Homeomorphic Embedding modulo Combinations of Associativity and Commutativity Axioms *

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

[1] DSIC-ELP, Universitat Politècnica de València, Spain.
{alpuente,acuenca,sescobar}@dsic.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 increasingly efficient formulations of the homeomorphic embedding relation modulo associativity and commutativity axioms. From our experimental results, we conclude that our 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, $v = s(0+s(X)) * s(X+Y)$ embeds $u = s(X) * s(Y)$. The usefulness of homeomorphic embedding for ensuring termination is given by the following well-known property of well-quasi-orderings: given a finite signature, 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 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.

* This work has been partially supported by the EU (FEDER) and the Spanish MINECO under grant TIN 2015-69175-C4-1-R, and by Generalitat Valenciana under grant PROMETEOII/2015/013. Jose Meseguer was partially supported by NRL under contract number N00173-17-1-G002. Angel Cuenca-Ortega has been supported by the SENESCYT, Ecuador (scholarship program 2013)

In [2], an extension of homeomorphic embedding modulo equational axioms, such as associativity and commutativity, was defined as a key component of the symbolic partial evaluator Victoria. 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 involved than their corresponding "syntactic" counterparts, and achieving efficient implementations has required years of significant investigation.

Our contribution. In this paper, we introduce four different formulations of 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. We propose an order-sorted, equational homeomorphic embedding formulation $\trianglelefteq_B^{sml}$ that runs up to 5 orders of magnitude faster than the original definition of $\trianglelefteq_B$ 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 [5].

Plan of the paper. After some preliminaries in Section 2, Section 3 recalls the homeomorphic equational embedding relation of [2] that extends the "syntactically simpler" homeomorphic embedding on nonground terms to the order-sorted case *modulo* equational axioms. Section 4 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 and commutativity axioms. Section 5 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, $1+(2+3)$ gets flattened to $+(1,2,3)$). An improvement of the algorithm is also achieved by replacing the classical boolean operators by short-circuit, strategic versions of these operators. We provide an experimental performance evaluation of the proposed formulations showing that we can efficiently deal with realistic embedding problems modulo axioms.

2 Preliminaries

We introduce some key concepts of order-sorted rewriting logic theories, see [5] for further details.

Given an *order-sorted signature* Σ, with a finite poset of sorts $(S,\leq)$, we consider an S-sorted family $\mathscr{X} = \{\mathscr{X}_{\mathsf{s}}\}_{\mathsf{s}\in\mathsf{S}}$ of disjoint variable sets. $\mathscr{T}_{\Sigma}(\mathscr{X})_{\mathsf{s}}$ and $\mathscr{T}_{\Sigma\mathsf{s}}$ denote the sets of terms and ground terms of sorts s, respectively. We also write $\mathscr{T}_{\Sigma}(\mathscr{X})$ and $\mathscr{T}_{\Sigma}$ for the corresponding term algebras. In order to simplify the presentation, we often disregard sorts when no confusion can arise. 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). $t|_p$ 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. A *substitution* σ is a sorted mapping from a finite subset of $\mathscr{X}$ 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 $t\sigma$.

A Σ*-equation* is an unoriented pair $t = t'$, where $t, t' \in \mathscr{T}_\Sigma(\mathscr{X})_{\mathsf{s}}$ 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 $t, t' \in \mathscr{T}_\Sigma(\mathscr{X})$ (see [4]). An *equational theory* (Σ, E) is a pair with Σ being an order-sorted signature and E a set of Σ-equations. A substitution θ is more (or equally) general than σ modulo E, denoted by $\theta \leq_E \sigma$, if there is a substitution γ such that $\sigma =_E \theta\gamma$, i.e., for all $x \in \mathscr{X}, x\sigma =_E x\theta\gamma$. A substitution σ is called a renaming if $\sigma = \{X_1 \mapsto Y_1, \ldots, X_n \mapsto Y_n\}$, the sorts of X_i and Y_i coincide, and variables $Y_1, \ldots, Y_n$ are pairwise distinct. The renaming substitution σ is a renaming for expression E if $(\mathscr{V}ar(E) - \{X_, \ldots, X_n\}) \cap \{Y_1, \ldots, Y_n\} = \emptyset$.

A *rewrite theory* is a triple $\mathscr{R} = (\Sigma, E, R)$, where (Σ, E) is the equational theory modulo that we rewrite and R is a set of rewrite rules. Rules are of the form $l \to r$ where terms $l, r \in \mathscr{T}_\Sigma(\mathscr{X})_{\mathsf{s}}$ for some sort s are respectively called the *left-hand side* (or *lhs*) and the *right-hand side* (or *rhs*) of the rule and $\mathscr{V}ar(r) \subseteq \mathscr{V}ar(l)$. 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 relation $\to_{R,E}$ on $\mathscr{T}_\Sigma(\mathscr{X})$ by $t \to_{p,R,E} t'$ (or simply $t \to_{R,E} t'$) 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 $t|_p =_E l\sigma$ and $t' = t[r\sigma]_p$.

2.1 Pure homeomorphic embedding

The pure (syntactic) homeomorphic embedding relation known from term algebra [11] was introduced by Dershowitz for variable-arity symbols in [6] and for fixed-arity symbols in [7]. In the following, we consider only fixed-arity symbols.

Definition 1 (Homeomorphic embedding, Dershowitz [7]). *The homeomorphic embedding relation* $\trianglelefteq$ *over* $\mathscr{T}_\Sigma$ *is defined as follows, with* $n \geq 0$*:*

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

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 $\preceq$ 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 j, k with $j < k$ and $t_j \preceq t_k$.

Theorem 1 (Tree Theorem, Kruskal [11]). *The embedding relation* $\trianglelefteq$ *is a well-quasi-ordering on* $\mathscr{T}_\Sigma$.

The derivability relation given by $\underline{\blacktriangleleft}$ is mechanized in [15] by introducing the following term rewriting system $Emb(\Sigma)$ as follows: $t \,\underline{\blacktriangleleft}\, t'$ if and only if $t' \rightarrow^*_{Emb(\Sigma)} t$.

Definition 2 (Homeomorphic embedding rewrite rules, Middeldorp [15]). *Let Σ be a signature. The 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 $i \in \{1, \cdots, n\}$.*

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. The following definition from [12, 16] adapts the pure (syntactic) homeomorphic embedding from [6] 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 varyadic symbols and infinite signatures are investigated in [13].

Definition 3 (Variable-extended homeomorphic embedding, Leuschel [12]). *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 equal to the pure (syntactic) homeomorphic embedding definition.*

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

Fig. 1. Variable-extended homeomorphic embedding

The extended embedding relation $\trianglelefteq$ is a well-quasi-ordering on the set of terms $\mathcal{T}_\Sigma(\mathcal{X})$ [12, 16]. An alternative characterization without the hassle of explicitly handling variables can be proved as follows.

Lemma 1 (Variable-less characterization of $\trianglelefteq$). *Given a signature Σ, let $\Sigma^\sharp$ be an extension of Σ with a new constant $\sharp$, and let $t^\sharp$ denote the (ground) instance of t where all variables have been replaced by $\sharp$. Given terms t_1 and t_2, $t_1 \trianglelefteq t_2$ iff $t_1^\sharp \trianglelefteq t_2^\sharp$ iff $t_1^\sharp \,\underline{\blacktriangleleft}\, t_2^\sharp$.*

Moreover, Lemma 1 above allows the variable-extended relation $\trianglelefteq$ of Definition 3 to be mechanized in a way similar to the rewriting relation $\rightarrow^*_{Emb(\Sigma)}$ used in Definition 2 for the embedding $\underline{\blacktriangleleft}$ of Definition 1: $t_1 \trianglelefteq t_2$ if and only if $t_2^\sharp \rightarrow^*_{Emb(\Sigma^\sharp)} t_1^\sharp$. By abuse of notation, from now on, we will indistinctly consider either terms with variables or ground terms with $\sharp$, whenever one formulation is simpler than the other.

3 Homeomorphic embedding modulo equational axioms

The following definition given in [2] extends the "syntactically simpler" homeomorphic embedding relation on nonground terms to the order-sorted case *modulo* a set of axioms B. The (order-sorted) relation $\trianglelefteq_B$ is called B–embedding (or embedding modulo B). We define $v \overset{ren}{=}_B v'$ iff there is a renaming substitution σ for v' such that $v =_B v'\sigma$.

Definition 4 ((Order-sorted) homeomorphic embedding modulo B). *We define the B–embedding relation $\unlhd_B$ (or embedding modulo B) as $(\overset{ren}{=}_B).(\unlhd).(\overset{ren}{=}_B)$.*

Example 1. Consider the following rewrite theory (written in Maude syntax) that defines the signature of natural numbers, with sort `Nat` and constructor operators `0`, and `suc` for sort `Nat`. We also define the associative and commutative addition operator symbol `_+_`.

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

We have $+(1,X{:}Nat) \unlhd_B +(Y{:}Nat,+(1,3))$ because $+(Y{:}Nat,+(1,3))$ is equal to $+(1,+(Y{:}Nat,3))$ modulo AC, and $+(1,X{:}Nat) \unlhd +(1,+(Y{:}Nat,3))$.

The following result extends Kruskal's Tree Theorem for the equational theories considered in this paper. We have to restrict it to the class of finite equational theories in order to prove the result. $\mathscr{B}$ is called *class-finite* if all $\mathscr{B}$-equivalence classes are finite. This includes the class of permutative equational theories. An equational theory $\mathscr{E}$ is permutative if for all terms t, t', the fact that $t =_{\mathscr{E}} t'$ implies that the terms t and t' contain the same symbols with the same number of occurrences [10]. Permutative theories include any theory with any combination of symbols obeying any combination of associativity and commutativity axioms.

Theorem 2. *For class-finite theories, the embedding relation $\unlhd_B$ is a well-quasi ordering of the set $\mathscr{T}_\Sigma(\mathscr{X})$ for finite Σ, that is, $\unlhd_B$ is a quasi-order.*

Function symbols with variable arity are sometimes seen as associative operators. Let us briefly discuss the homeomorphic embedding modulo axioms $\unlhd_B$ of Definition 4 in comparison to the variadic extension $\trianglelefteq^v$ of Definition 1 as given in [6]:

Diving

$$\frac{\exists i{\in}\{1,\dots,n\} \,:\, s \trianglelefteq^v t_i}{s \trianglelefteq^v f(t_1,\dots,t_n)}$$

Coupling

$$\frac{\forall i{\in}\{1,\dots,m\} \,:\, s_i \trianglelefteq^v t_{j_i}, \text{with } 1{\le}j_1{<}j_2{<}\cdots{<}j_m{\le}n}{f(s_1,\dots,s_m) \trianglelefteq^v f(t_1,\dots,t_n)}$$

Example 2. Consider a variadic version of the addition symbol $+$ of Example 1 that allows any number of natural numbers to be used as arguments; for instance, $+(1,2,3)$. On the one hand, $+(1) \trianglelefteq^v +(1,2,3)$ whereas $+(1) \not\!\unlhd_B +(1,2,3)$, with B consisting of the associativity and commutativity axioms for the operator $+$ (actually, $+(1)$ is ill-formed). On the other hand, we have both $+(1,2) \trianglelefteq^v +(1,0,3,2)$ and $+(1,2) \unlhd_B +(1,0,3,2)$. 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, 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[4] *canonical representative* $+(0,1,2,3)$, hence also $+(1,2) \trianglelefteq_B +(0,1,2,3)$. A more detailed explanation of flat terms can be found in Section 5. However, note that $+(2,1) \trianglelefteq_B +(1,0,3,2)$ but $+(2,1) \not\trianglelefteq^{v} +(1,0,3,2)$ because the $\trianglelefteq^{v}$ does not consider the commutativity of symbol $+$.

Roughly speaking, in the worst case, the homeomorphic embedding modulo axioms of Definition 4, $t \trianglelefteq_B t'$, amounts to considering all the elements in the B-equivalence classes of t and t' and then checking for standard homeomorphic embedding, $u \trianglelefteq u'$, every pair u and u' of such terms, one term from each class. According to Definition 2, checking $u \trianglelefteq u'$ essentially boils down to the reachability analysis given by $u' \rightarrow^*_{Emb(\Sigma)} u$. Unfortunately, the enumeration of all terms in a B-equivalence class is impractical, as shown in the following example.

Example 3. Consider the AC binary symbol $+$ of Example 1 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 computing 24 reachability problems $u' \rightarrow^*_{Emb(\Sigma)} u$ in order to decide $t \trianglelefteq_{AC} t'$, in the worst case. Moreover, we know a priori that half of these reachability 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 $\trianglelefteq_B$ can be achieved by lifting Definition 2 to the order-sorted and *modulo* case in a natural way. However, ill-formed terms can be produced by naïvely applying the rules $f(X_1,\ldots,X_n) \rightarrow X_i$ of Definition 2 to typed (i.e., order-sorted) terms. For example, "$(\texttt{0} \leq \texttt{1})$ `or true`" $\rightarrow$ "`0 or true`".

In the order-sorted context we can overcome this drawback as follows. Assume that Σ has no ad-hoc overloading. We can extend Σ to a new signature $\Sigma^{\mathscr{U}}$ by adding a new top sort $\mathscr{U}$ that is bigger than all other sorts. For each $f : A_1,\ldots,A_n \rightarrow A$ in Σ, we add the rules $f(X_1{:}\mathscr{U},\ldots,X_n{:}\mathscr{U}) \rightarrow X_i{:}\mathscr{U}$, $1 \leq i \leq n$. In this way, rewriting with $\rightarrow^*_{Emb(\Sigma^{\mathscr{U}}),B}$ becomes a relation between well-formed $\Sigma^{\mathscr{U}}$-terms, see [2].

Definition 5 ((Order-sorted) homeomorphic embedding rewrite rules modulo B [2]). *Let Σ be an order-sorted signature and B be a set of axioms. Let us introduce the following signature transformation $\Sigma \ni (f : \mathsf{s}_1 \ldots \mathsf{s}_n \rightarrow \mathsf{s}) \mapsto (f : \mathscr{U} \overset{n}{\ldots} \mathscr{U} \rightarrow \mathscr{U}) \in \Sigma^u$, where $\mathscr{U}$ conceptually represents a universal supersort of all sorts in Σ. Also, for any Σ-term t, t^u leaves the term t unchanged but regards all its variable as unsorted (i.e., of sort $\mathscr{U}$). We define the TRS Emb(Σ) that consists of all rewrite rules $f(X_1{:}\mathscr{U},\ldots,X_n{:}\mathscr{U}) \rightarrow X_i{:}\mathscr{U}$ for each $f : A_1,\ldots,A_n \rightarrow A$ in Σ and $i \in \{1,\ldots,n\}$.*

In the sequel, we consider equational theories B that may contain any combination of associativity and/or commutativity axioms for any binary symbol in the signature. Also, for the sake of simplicity we often omit sorts when no confusion can arise.

Proposition 1. *Given Σ, B, and t,t' in $\mathscr{T}_\Sigma(\mathscr{X})$, $t \trianglelefteq_B t'$ iff $(t'^u)^\sharp \rightarrow^*_{Emb((\Sigma^{\mathscr{U}})^\sharp),B} (t^u)^\sharp$.*

Example 4. Consider the order-sorted signature for natural numbers of Example 1. Let us represent by sort `U` in Maude the unique (top) sort of the transformed signature:

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

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

Likewise, the terms expressed in Σ must also be transformed to be expressed as $\Sigma^{\mathscr{U}}$-terms. For instance, given the Σ-terms $t = \texttt{X:Nat}$[5] and $t' = \texttt{suc(Y:Nat)}$, the corresponding $\Sigma^{\mathscr{U}}$-terms are $t = \texttt{X:U}$ and $\texttt{suc(Y:U)}$, respectively.

The associated TRS $Emb(\Sigma)$ contains the following two rules: $+(X_1{:}U, X_2{:}U) \to X_1{:}U$ and $+(X_1{:}U, X_2{:}U) \to X_2{:}U$. However, since the rules of $Emb(\Sigma)$ are applied modulo the commutativity of symbol $+$, in practice, we can get rid of either of the two rules above since only one is required in Maude.

Example 5. Following Example 3, instead of comparing pairwisely all terms in the equivalence classes of t and t', we choose $Emb(\Sigma)$ to contain just the rewrite rule $+(X_1{:}U, X_2{:}U) \to X_2{:}U$, we use it to prove the rewrite step $+(2,+(3,1)) \to_{Emb(\Sigma),B} +(2,1)$, and finally we check that $+(2,1) =_B +(1,2)$, with $B = \{A, C\}$. However, there are six alternative rewriting steps stemming from the initial term $+(2,+(3,1))$, all of which result from applying the very same rewrite rule above to the term (modulo AC), five of which are useless for proving the considered embedding (the selected redex is underlined):

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

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.

Moreover, there are three problems with Definition 5. First, the intrinsic non-determinism of the rules may unnecessarily produce an extremely large search space. Second, as shown in Example 5, 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 stepwisely in the sequel.

4 Goal-driven homeomorphic embedding modulo B

The formulation of homeomorphic embedding as a reachability problem by using the rewrite rules of Definition 5 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 (or *goal*), $t \trianglelefteq_B t'$, it inductively processes the terms t and t' in a top-down manner.

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

[5] The expression $X{:}S$ represents an explicit definition of a variable X of sort S in Maude.

4.1 An homeomorphic embedding calculus modulo B

Let us introduce a calculus for embeddability goals $t \trianglelefteq^{gd}_B t'$ that directly handles in the deduction system the algebraic axioms of B, with B being any combination of A and/or C axioms for the theory operators. Roughly speaking, this is achieved by specializing w.r.t. B the coupling rule of Definition 3.

Definition 6 (Goal-driven 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 2: (i) the three inference rules (Variable, Diving, and Coupling) of Definition 3 for any function symbol; (ii) one extra coupling rule for the case of a commutative symbol with or without associativity (Coupling$_C$); (iii) two extra coupling rules for the case of an associative symbol with or without commutativity (Coupling$_A$); and (iv) two extra coupling rules for the case of an associative-commutative symbol (Coupling$_{AC}$).*

$$\textbf{Coupling}_C \quad \frac{s_0 \trianglelefteq^{gd}_B t_1 \ \wedge\ s_1 \trianglelefteq^{gd}_B t_0}{f(s_0,s_1) \trianglelefteq^{gd}_B f(t_0,t_1)}$$

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

$$\textbf{Coupling}_{AC} \quad \frac{f(s_0,s_1) \trianglelefteq^{gd}_B t_1 \ \wedge\ s_2 \trianglelefteq^{gd}_B t_0}{f(s_0,f(s_1,s_2)) \trianglelefteq^{gd}_B f(t_0,t_1)} \qquad \frac{s_1 \trianglelefteq^{gd}_B f(t_0,t_1) \ \wedge\ s_0 \trianglelefteq^{gd}_B t_2}{f(s_0,s_1) \trianglelefteq^{gd}_B f(t_0,f(t_1,t_2))}$$

Fig. 2. Extra coupling rules for A, C, AC symbols

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

Example 6. Consider the binary symbol $+$ obeying associativity and commutativity axioms, and the terms $t = +(1,2)$ and $t' = +(2,+(3,1))$ of Example 5. We can prove $t \trianglelefteq^{gd}_B t'$ by

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

We can also prove a more complex embedding goal by first using the right inference rule for AC of Figure 2 and then the generic Coupling and Diving inference rules.

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

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

4.2 Reachability-based, goal-driven homeomorphic embedding formulation

Let us provide a more operational goal-driven characterization of the homeomorphic embedding modulo B. We formalize it in the reachability style of Definition 5. The main challenge here is how to generate a suitable rewrite theory $R^{rogd}(\Sigma,B)$ that can decide embedding modulo B by running a reachability goal.

Definition 7 (Goal-driven homeomorphic embedding rewrite rules modulo B). *Given Σ and B, we define the TRS $R^{rogd}(\Sigma,B)$ as follows.*

1. *We include in $R^{rogd}(\Sigma,B)$ a rewrite rule of the form $u \trianglelefteq_B^{rogd} v \to true$ for each (particular intance of the) inference rules of the form $\frac{}{u \trianglelefteq_B^{gd} v}$ given Definition 6 (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).*
2. *We include in $R^{rogd}(\Sigma,B)$ a rewrite rule of the form $u \trianglelefteq_B^{rogd} v \to u_1 \trianglelefteq_B^{rogd} v_1 \wedge \cdots \wedge u_k \trianglelefteq_B^{rogd} v_k$ for each (particular intance of the) inference rules of the form $\frac{u_1 \trianglelefteq_B^{gd} v_1 \wedge \cdots \wedge u_k \trianglelefteq_B^{gd} v_k}{u \trianglelefteq_B^{gd} v}$ given in Definition 6.*

Proposition 3. *Given Σ, B, and terms t,t', $t \trianglelefteq_B^{gd} t'$ iff $(t \trianglelefteq_B^{rogd} t') \to^*_{R^{rogd}(\Sigma,B),B} true$.*

Example 7. Consider the binary symbol $+$ of Example 1. According to Definition 6, there are twelve inference rules for $\trianglelefteq_B^{gd}$:

$$\begin{array}{ccc}
\text{Variable} & \text{Diving} & \text{Coupling} \\
\frac{}{x \trianglelefteq_B^{gd} y} & \frac{x \trianglelefteq_B^{gd} t_1}{x \trianglelefteq_B^{gd} suc(t_1)} & \frac{}{0 \trianglelefteq_B^{gd} 0} \\
 & \frac{x \trianglelefteq_B^{gd} t_1}{x \trianglelefteq_B^{gd} +(t_1,t_2)} & \frac{t_1 \trianglelefteq_B^{gd} t_1'}{suc(t_1) \trianglelefteq_B^{gd} suc(t_1')} \\
 & \frac{x \trianglelefteq_B^{gd} t_2}{x \trianglelefteq_B^{gd} +(t_1,t_2)} & \frac{t_1 \trianglelefteq_B^{gd} t_1' \wedge t_2 \trianglelefteq_B^{gd} t_2'}{+(t_1,t_2) \trianglelefteq_B^{gd} +(t_1',t_2')}
\end{array}$$

$$\begin{array}{ccc}
\text{Coupling}_C & \text{Coupling}_A & \text{Coupling}_{AC} \\
\frac{t_1 \trianglelefteq_B^{gd} t_2' \wedge t_2 \trianglelefteq_B^{gd} t_1'}{+(t_1,t_2) \trianglelefteq_B^{gd} +(t_1',t_2')} & \frac{+(t_0,t_1) \trianglelefteq_B^{gd} t_1' \wedge t_2 \trianglelefteq_B^{gd} t_2'}{+(t_0,+(t_1,t_2)) \trianglelefteq_B^{gd} +(t_1',t_2')} & \frac{+(t_0,t_1) \trianglelefteq_B^{gd} t_2' \wedge t_2 \trianglelefteq_B^{gd} t_1'}{+(t_0,+(t_1,t_2)) \trianglelefteq_B^{gd} +(t_1',t_2')} \\
 & \frac{t_1 \trianglelefteq_B^{gd} +(t_0',t_1') \wedge t_2 \trianglelefteq_B^{gd} t_2'}{+(t_1,t_2) \trianglelefteq_B^{gd} +(t_0',+(t_1',t_2'))} & \frac{t_2 \trianglelefteq_B^{gd} +(t_0',t_1') \wedge t_1 \trianglelefteq_B^{gd} t_2'}{+(t_1,t_2) \trianglelefteq_B^{gd} +(t_0',+(t_1',t_2'))}
\end{array}$$

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

$$\begin{array}{lrll}
\text{(Diving)} & x \trianglelefteq_B^{rogd} suc(T_1) & \to x \trianglelefteq_B^{rogd} T_1 \\
 & x \trianglelefteq_B^{rogd} +(T_1,T_2) & \to x \trianglelefteq_B^{rogd} T_1 \\
\text{(Coupling)} & \sharp \trianglelefteq_B^{rogd} \sharp & \to true \\
 & 0 \trianglelefteq_B^{rogd} 0 & \to true \\
 & suc(T_1) \trianglelefteq_B^{rogd} suc(T_1') & \to T_1 \trianglelefteq_B^{rogd} T_1' \\
\text{(Coupling}_{\emptyset,C,A,AC}) & +(T_1,T_2) \trianglelefteq_B^{rogd} +(T_1',T_2') & \to T_1 \trianglelefteq_B^{rogd} T_1' \wedge T_2 \trianglelefteq_B^{rogd} T_2'
\end{array}$$

For example, the rewrite sequence proving $+(1,+(2,3)) \trianglelefteq_B^{rogd} +(+(4,2),+(3,1))$ is:

$$\begin{array}{l}
+(1,+(2,3)) \trianglelefteq_B^{rogd} +(+(4,2),+(3,1)) \\
\qquad \to_{R^{rogd}(\Sigma,B),B} +(2,3)) \trianglelefteq_B^{rogd} +(+(4,2),3) \wedge 1 \trianglelefteq_B^{rogd} 1 \\
\qquad \to_{R^{rogd}(\Sigma,B),B} 2 \trianglelefteq_B^{rogd} +(4,2) \wedge 3 \trianglelefteq_B^{rogd} 3 \\
\qquad \to_{R^{rogd}(\Sigma,B),B} 2 \trianglelefteq_B^{rogd} 2 \\
\qquad \to_{R^{rogd}(\Sigma,B),B} true
\end{array}$$

Although the improvement in performance achieved by using the rewriting relation $\to_{R^{rogd}(\Sigma,B),B}$ versus the rewriting relation $\to^*_{Emb(\Sigma),B}$ is important, the search space is still huge since the expression $+(1,+(2,3)) \trianglelefteq_B^{gd} +(+(4,2),+(3,1))$ matches the left-hand side $+(T_1,T_2) \trianglelefteq_B^{gd} +(T_1',T_2')$ 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 further optimize the 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).

5 Meta-Level deterministic goal-driven homeomorphic embedding modulo B

The meta-level representation of terms in Maude [5, Chapter 14] works with flattened versions of the terms that are rooted by poly-variadic versions of the associative (or associative-commutative) symbols. For instance, given an associative (or associative-commutative) 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

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

Given an associative (or associative-commutative) symbol f and a term $f(t_1,\ldots,t_n)$, we call *f-alien terms* (or simply *alien terms*) those terms among the $t_1,\ldots,t_n$ that are not rooted by f. In the following, we implicitly consider that all terms are in B-canonical form.

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 $\bar{t} =$ '$f[\bar{t_1},\ldots,\bar{t_n}]$.

Definition 8 (Meta-level homeomorphic embedding modulo B). *The meta-level homeomorphic embedding modulo B, $\trianglelefteq_B^{ml}$, is defined for term meta-representations by means of the equational theory E^{ml} of Figure 3, 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 that are specific to A and AC symbols. For the sake of readability, these new existential tests are also formulated (for ordinary terms instead of meta-level terms) as the inference rules Coupling$_A$ and Coupling$_{AC}$ of Figure 4.*

$$
\begin{array}{rll}
\sharp \trianglelefteq_B^{ml} \sharp & = \mathbf{true} & \\
F[\mathit{TermList}] \trianglelefteq_B^{ml} \sharp & = \mathbf{false} & \\
T \trianglelefteq_B^{ml} F[\mathit{TermList}] & = \mathbf{any}(T, \mathit{TermList}) & \text{if } \mathit{root}(T) \neq F \\
F[\mathit{TermList}1] \trianglelefteq_B^{ml} F[\mathit{TermList}2] & = \mathbf{any}(F[\mathit{TermList}1], \mathit{TermList}2) & \\
 & \quad \mathbf{or\ all}(\mathit{TermList}1, \mathit{TermList}2) & \\
F[U,V] \trianglelefteq_B^{ml} F[X,Y] & = \mathbf{any}(F[U,V],[X,Y]) & \text{if } F \text{ is } C \\
 & \quad \mathbf{or}(\ U \trianglelefteq_B^{ml} X \ \mathbf{and}\ V \trianglelefteq_B^{ml} Y\ ) & \\
 & \quad \mathbf{or}\ (\ U \trianglelefteq_B^{ml} Y \ \mathbf{and}\ V \trianglelefteq_B^{ml} X\ ) & \\
F[\mathit{TermList}1] \trianglelefteq_B^{ml} F[\mathit{TermList}2] & = \mathbf{any}(F[\mathit{TermList}1], \mathit{TermList}2) & \text{if } F \text{ is } A \\
 & \quad \mathbf{or\ all\_A}(\mathit{TermList}1, \mathit{TermList}2) & \\
F[\mathit{TermList}1] \trianglelefteq_B^{ml} F[\mathit{TermList}2] & = \mathbf{any}(F[\mathit{TermList}1], \mathit{TermList}2) & \text{if } F \text{ is } AC \\
 & \quad \mathbf{or\ all\_AC}(\mathit{TermList}1, \mathit{TermList}2) &
\end{array}
$$

$$
\begin{array}{rl}
\mathbf{any}(U, \mathit{nil}) & = \mathbf{false} \\
\mathbf{any}(U, V : L) & = U \trianglelefteq_B^{ml} V \ \mathbf{or\ any}(U, L)
\end{array}
$$

$$
\begin{array}{rl}
\mathbf{all}(\mathit{nil}, \mathit{nil}) & = \mathbf{true} \\
\mathbf{all}(\mathit{nil}, U : L) & = \mathbf{false} \\
\mathbf{all}(U : L, \mathit{nil}) & = \mathbf{false} \\
\mathbf{all}(U : L1, V : L2) & = U \trianglelefteq_B^{ml} V \ \mathbf{and\ all}(L1, L2)
\end{array}
$$

$$
\begin{array}{rl}
\mathbf{all\_A}(\mathit{nil}, L) & = \mathbf{true} \\
\mathbf{all\_A}(U : L, \mathit{nil}) & = \mathbf{false} \\
\mathbf{all\_A}(U : L1, V : L2) & = (U \trianglelefteq_B^{ml} V \ \mathbf{and\ all\_A}(L1, L2)) \ \mathbf{or\ all\_A}(U : L1, L2))
\end{array}
$$

$$
\begin{array}{rl}
\mathbf{all\_AC}(\mathit{nil}, L) & = \mathbf{true} \\
\mathbf{all\_AC}(U : L1, L2) & = \mathbf{all\_AC\_Aux}(U : L1, L2, L2) \\
\mathbf{all\_AC\_Aux}(U : L1, \mathit{nil}, L3) & = \mathbf{false} \\
\mathbf{all\_AC\_Aux}(U : L1, V : L2, L3) & = (U \trianglelefteq_B^{ml} V \ \mathbf{and\ all\_AC}(L1, \mathbf{remove}(V, L3))) \\
 & \quad \mathbf{or\ all\_AC\_Aux}(U : L1, L2, L3))
\end{array}
$$

$$
\begin{array}{rl}
\mathbf{remove}(U, \mathit{nil}) & = \mathbf{nil} \\
\mathbf{remove}(U, V : L) & = \mathbf{if}\ U = V \ \mathbf{then}\ L \ \mathbf{else}\ V : \mathbf{remove}(U, L)
\end{array}
$$

Fig. 3. Meta-level homeomorphic embedding modulo axioms

Example 8. Given the embedding problem for $+(1,+(2,3))$ and $+(+(4,2),+(3,1))$, the corresponding call to the meta-level homeomorphic embedding $\trianglelefteq_B^{ml}$ of Definition 8 is '+['1, '2, '3] $\trianglelefteq_B^{ml}$ '+['4, '2, '3, '1].

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

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

Fig. 4. Coupling rule for associativity-commutativity functions

Proposition 4. *Given Σ, B, and terms t and t', $t \trianglelefteq_B^{gd} t'$ iff $(t \trianglelefteq_B^{ml} t')!_{E^{ml},B} = true$.*

Finally, a further optimized version of Definition 8 can be easily defined by replacing the Boolean conjunction (*and*) and disjunction (*or*) operators with the computationally more efficient Maude Boolean operators `and-then` and `or-else` that avoid evaluating the second argument when the result of evaluating the first one suffices to compute the result.

Definition 9 (Strategic meta-level deterministic embedding modulo B). *We define $\trianglelefteq_B^{sml}$ as the strategic version of relation $\trianglelefteq_B^{ml}$ that is obtained by replacing the Boolean operators* `and` *and* `or` *with the* `and-then` *operator for* short-circuit *version of conjunction and the* `or-else` *operator for short-circuit disjunction [5, Chapter 9.1], respectively.*

6 Experiments

We have implemented in Maude all four equational homeomorphic embedding formulations $\trianglelefteq_B$, $\trianglelefteq_B^{rogd}$, $\trianglelefteq_B^{ml}$, and $\trianglelefteq_B^{sml}$ of previous sections. The implementation consists of approximately 250 function definitions (2.2K lines of Maude source code) and is publicly available online[6]. In this section, we provide an experimental comparison of the four 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 [3]; (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

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

Benchmark	**$\sharp$ Axioms**				$\trianglelefteq_B$			$\trianglelefteq_B^{rogd}$			$\trianglelefteq_B^{ml}$, $\trianglelefteq_B^{sml}$		
	$\emptyset$	**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

Benchmark	**$\sharp$ Symbols**		**Size**		$\trianglelefteq_B$	$\trianglelefteq_B^{rogd}$	$\trianglelefteq_B^{ml}$	$\trianglelefteq_B^{sml}$
	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
NatList	1	2	5	5	2508	2892	1	1
				10	840310	640540	1	1
				100	TO	TO	8	2
				500	TO	TO	60	5
Maze	1	1	5	5	40	25	1	1
				10	TO	20790	4	1
				100	TO	TO	256	2
				500	TO	TO	19808	10
Dekker	1	1	5	5	50	40	1	1
				10	111468	110517	2	1
				100	TO	TO	5	3
				500	TO	TO	20	13

Table 2. Performance of equational homeomorphic embedding implementations

exclusion problem that appeared in [5]. As testing benchmarks we considered a set of representative embeddability problems for the four programs that are generated during the execution of the partial evaluator Victoria [2].

Tables 1, 2, and 3 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 both, $\trianglelefteq_B^{ml}$ and $\trianglelefteq_B^{sml}$, 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 $\trianglelefteq_B$ and $\trianglelefteq_B^{rogd}$, 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 $\trianglelefteq_B^{rogd}$ than for $\trianglelefteq_B$ (for instance, $\trianglelefteq_B^{rogd}$ is encoded by 823 rules for the Dekker program versus the 59 rules of $\trianglelefteq_B$). 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.

For all benchmarks $T1 \trianglelefteq_B^{\alpha} T2$ in Table 2, we have fixed to five the size of `T1` that is measured in the depth of (the non-flattened version of) the term. As for `T2`, we have

T1						T2						$\trianglelefteq_B$	$\trianglelefteq_B^{rogd}$	$\trianglelefteq_B^{ml}$	$\trianglelefteq_B^{sml}$
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	38	60	35
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	2183	6350	2005
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

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

considered terms with increasing depths: five, ten, one hundred, and five hundred. The ♯ `Symbols` column records the number of A (resp. AC) symbols occurring in the benchmarked goals.

The figures in Table 2 confirm our expectations regarding $\trianglelefteq_B$ and $\trianglelefteq_B^{rogd}$ that the search space is huge and increases exponentially with the size of `T2` (discussed for $\trianglelefteq_B$ in Example 5 and for $\trianglelefteq_B^{rogd}$ in Example 6). 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 $\trianglelefteq_B^{ml}$ and $\trianglelefteq_B^{sml}$ bring impressive speedups, with $\trianglelefteq_B^{sml}$ working outstandingly well in practice even for really complex terms.

The reader may wonder how big the impact is 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 confront the size of the original term (OT) versus 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 $\trianglelefteq_B$ and $\trianglelefteq_B^{rogd}$ for completeness, but they do not reveal a dramatic improvement of $\trianglelefteq_B^{rogd}$ with respect to $\trianglelefteq_B$ for the benchmarked (false) goals, contrary to what we initially expected. This means that $\trianglelefteq_B^{rogd}$ cannot be generally used in real applications due to the risk of intolerable embedding test times, even if $\trianglelefteq_B^{rogd}$ may be far less wasteful than $\trianglelefteq_B$ for succeeding goals, as discussed in Section 4. For $\trianglelefteq_B^{ml}$ and $\trianglelefteq_B^{sml}$, 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 5 we have compared on a logarithmic scale our best embedding definition, $\trianglelefteq_B^{sml}$, with a standard meta-level Prolog[7] implementation of the (syntactic) pure homeomorphic embedding $\trianglelefteq$ of Definition 3.

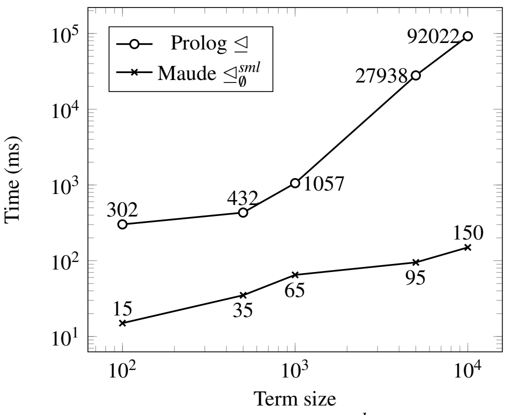

Fig. 5. Comparison of $\trianglelefteq$ vs. $\trianglelefteq_\emptyset^{sml}$ for NatList

We chose the NatList example and terms `T1` and `T2` that 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 $\trianglelefteq_B^{sml}$ (i.e. without search) outperforms the Prolog version so no penalty is incurred when syntactic embeddability tests are run in our equational implementation.

7 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 definition of homeomorphic embedding with a remarkably good performance for theories with symbols having any combination of associativity and commutativity. 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 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.

[7] To avoid any bias, we took the Prolog code for the homeomorphic embedding of the ECCE system [14] that is available at `https://github.com/leuschel/ecce`, and we run it in SWI-Prolog 7.6.3.

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, M. Falaschi, and G. Vidal. Partial Evaluation of Functional Logic Programs. *ACM TOPLAS*, 20(4):768–844, 1998.
4. A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and Proof in Membership Equational Logic. *Theor. Comput. Sci.*, 236(1-2):35–132, 2000.
5. 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.
6. 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.
7. Dershowitz, N. A Note on Simplification Orderings. *Information Processing Letters*, 9(5):212–215, 1979.
8. S. Eker. Single Elementary Associative-Commutative Matching. *J. Autom. Reasoning*, 28(1):35–51, 2002.
9. S. Escobar, J. Meseguer, and R. Sasse. Variant Narrowing and Equational Unification. *Electronic Notes Theoretical Computer Science*, 238(3):103–119, 2009.
10. H.J. Bürckert and A. Herold and M. Schmidt-Schau. On Equational Theories, Unification, and (Un)decidability. *Journal of Symbolic Computation*, 8(1–2):3–49, 1989.
11. J.B. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95:210–225, 1960.
12. 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.
13. 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.
14. 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.
15. A. Middeldorp and B. Gramlich. Simple Termination is Difficult. *Applicable Algebra in Engineering, Communication and Computing*, 6(2):115–128, 1995.
16. 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.