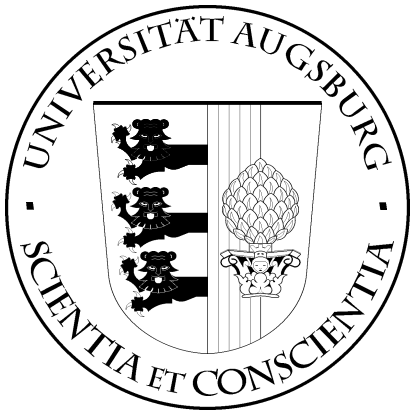
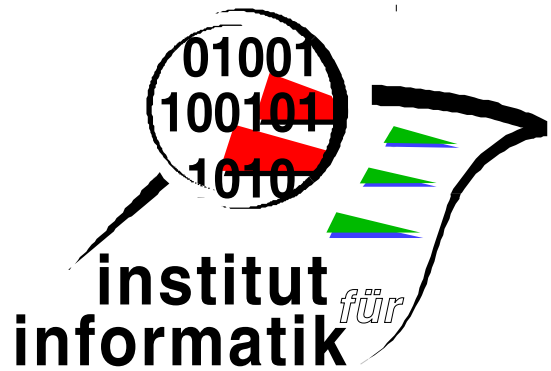
Universität Augsburg



Termination of Ground Non-Symmetric Knuth-Bendix Completion

Georg Struth

Report 2002-9 Mai 2002



Institut für Informatik
D-86135 Augsburg

Termination of Ground Non-Symmetric Knuth-Bendix Completion

Georg Struth

Institut für Informatik, Universität Augsburg
Universitätsstr. 14, D-86135 Augsburg, Germany
Tel:+49-821-598-3109, Fax:+49-821-598-2274,
struth@informatik.uni-augsburg.de

Abstract In the most natural approach, ground Knuth-Bendix completion procedures for non-symmetric transitive relations and quasiorderings, as specified in [16], need not terminate. We use a two-step transformation on the input expressions to enforce termination after $O(n)$ steps in the size of the input. We apply the completion procedure for developing rule-based declarative and dynamic algorithms for detecting embeddings in ground rewrite sequences. These results are interesting for the constraint-based analysis of state transition systems, for partial evaluation and for the reachability and termination analysis of ground term rewrite systems.
Keywords: Knuth-Bendix completion, transitive relations, quasiorderings, reachability, termination, homeomorphic embedding.

1 Introduction

Non-symmetric Knuth-Bendix completion is a method for reasoning about reachability in transitive relations, orderings, graphs and constraint systems [13,16], in particular, when these systems have infinitely many states. Although this procedure generalizes concepts and techniques of equational completion, there are significant differences [16]. There are, for instance, no term normal forms and decision procedures are not don't care non-deterministic, but search-based. Moreover, there are two severe obstacles to applicability. First, unlike the equational case, even ground non-symmetric completion need not terminate, at least in the most natural approach. Second, the non-ground procedure involves variable critical pairs that are usually modeled by context variables. It therefore requires new types of rewrite orderings and depends on a possibly undecidable unification problem [3,9].

Here, we show how to circumvent the first obstacle. We enforce termination via a two-step transformation on the input expressions to the procedure. First, all ground terms are transformed such that functions become constants; instead one single new binary function is introduced. In terms of functional languages this step *curries* terms. Second, all curried terms are flattened to height at most two and size at most three. The whole transformation increases the size of the input by a linear factor. Inspection of the possible completion steps with transformed

terms shows that their number is finitely bounded and in fact linear in the size of the input. It follows that the ground non-symmetric Knuth-Bendix completion procedure takes polynomial running time.

Non-symmetric Knuth-Bendix completion has interesting applications in the fields of declarative programming and constraint systems. It has been proposed as a metaprocedure for developing rule-based memoization and dynamic cycle elimination algorithms [12]. Our termination result opens the way to further applications. We present two rule-based extensions of the procedure that characterize classes of algoritms for detecting (homeomorphic) embeddings in ground rewrite sequences. These decision procedures are highly non-deterministic and therefore can easily be refined to efficient algorithms via execution strategies. Since they rely directly on a mathematical specification of the problem, correctness proofs are especially simple and precise. Moreover, the algorithms are generic; most data-structures and implementation details are hidden in the implementation of the metaprocedure. Finally, the algorithms easily adapt to dynamically changing environments. Applications are reachability and termination analysis of rewrite systems [4], programs [2,8] and state transition systems in general as well as partial evaluation and program transformation [6].

The remainder of this text is organized as follows. Section 2 contains some preliminary definitions. Section 3 sketches the basics of rewriting and ground Knuth-Bendix completion quasiorderings. Section 4 defines the two transformation steps on the input ground term rewrite system. Section 5 analyzes termination of the procedure. Section 6 shows two declarative algorithms for detecting embeddings as applications. Section 7 contains a conclusion and points out interesting future work.

2 Preliminaries

Let T_Σ be a (finite) set of ground terms with signature Σ. Σ_n denotes the set of n-ary function symbols in Σ. Elements of Σ_0 are constants. Let C be a denumerably infinite set of constants disjoint from Σ. We write $T_\Sigma(C)$ instead of $T_{\Sigma\cup C}$. As usually, terms are identified with Σ-labeled trees with nodes or *positions* in the monoid $\mathbb{N}^*$. We ambiguously write $s[t]_p$, if t is the subterm of s at position p or if the subterm $s = r|_p$ of r at position p is replaced by t. For a constant $c \in \Sigma$, an n-ary function $f \in \Sigma$ and terms $t_1, \ldots, t_n \in T_\Sigma$ we recursively define the *height* $h : T_\Sigma \to \mathbb{N}$ of a term as

$$h(c) = 1, \qquad h(f(t_1, \ldots, t_n)) = 1 + \max\{h(t_1), \ldots, h(t_n)\}$$

and the *size* $|.| : T_\Sigma \to \mathbb{N}$ of a term as

$$|c| = 1, \qquad |f(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t_i|.$$

Let $\to$ be a binary relation on a set A. We write $\leftarrow$ for its converse, $\leftrightarrow$ for its symmetric closure, $\to^+$ for its transitive closure and $\to^*$ for its reflexive

transitive closure. Juxtaposition of relations denotes relational product. $\rightarrow$ is a *quasiordering*, if it is reflexive and transitive. Let $\rightarrow$ be a binary relation on a ground term algebra A with associated set of ground terms T_Σ. The operation f^A denoted by the n-ary operation symbol f is *monotone* (in each argument), if it satisfies the formula

$$s_1 \rightarrow t_1, \ldots, s_n \rightarrow t_n \Longrightarrow f(s_1, \ldots, s_n) \rightarrow f(t_1, \ldots, t_n),$$

for all $s_1, \ldots, s_n, t_1, \ldots, t_n \in T_\Sigma$. A (ground) *rewrite rule* is a pair of (ground) terms. Given some set of rewrite rules, a (ground) *rewrite relation* is the smallest relation $\rightarrow_R$ such that $s[l]_p \rightarrow_R s[r]_p$, whenever $l \rightarrow r$ is a rewrite rule. Of course all functions corresponding to predecessors of p must be monotonic. We often write $\rightarrow_R$ both for the set of rewrite rules and the associated rewrite relation.

We also consider syntactic orderings $\prec$ on (ground) terms. We assume that $\prec$ is total and noetherian and that it contains the proper subterm relation.

3 Non-Symmetric Rewriting and Completion

We presuppose the basic concepts and notation of equational and non-symmetric rewriting and completion [5,9,13,16]. A combination algorithm of non-symmetric and equational completion with applications to graph algorithms has been presented in [12]. We therefore briefly recall only the most important notions. For the sake of simplicity, we restrict our attention to quasiorderings. The adaption to transitive relations is nevertheless straightforward.

A *non-symmetric ground term rewrite system* (GTRS) is a triple $(R, S, \prec)$ of sets R and S of rewrite rules and a reduction ordering $\prec$ such that $l \succ r$ for all $l \rightarrow_R r$ and $l \prec s$ for all $l \rightarrow_S r$. Let $(R, S, \prec)$ be a GTRS that partitions a finite presentation $\rightarrow_I$ of some (monotonic) quasiordering $\leq$, discarding the reflexive part of $\rightarrow_I^*$. Let $\rightarrow_R \cup \leftarrow_S$ be noetherian and assume that every two-step rewrite sequence of the form $\rightarrow_S \rightarrow_R$ (a *peak*) can be replaced by a rewrite sequence in $\rightarrow_R^* \rightarrow_S^*$ (a *valley*). Then reachability in $\rightarrow_I$ can be decided by searching for a common vertex in the $\rightarrow_R$ dag from the initial vertex and the $\leftarrow_S$ dag from the final vertex of the query. Of course only finitely many rewrite rules apply to each term and therefore the dags are finitely branching. A *critical pair* is a pair of terms that is connected by a peak and that can possibly not be replaced by a valley.

As usually, a Knuth-Bendix completion procedure (KB-procedure) transforms an initial set of expressions in $\rightarrow_I$ into a GTRS that supports this decision procedure. It orients inequalities, computes critical pairs and simplifies expressions. We call the final GTRS a *normal system*. By definition, all critical peaks can be joined by a rewrite proof, $\rightarrow_R$ and $\leftarrow_S$ are noetherian and no rule from $\rightarrow_R$ or $\rightarrow_S$ can be deleted. A KB-procedure is a program that implements a state transition system together with a syntactic reduction ordering $\prec$ on terms, (oriented) inequalities and sequences of inequalities and rewrite steps. States are tuples of sets of inequalities or rewrite rules. The transition relation is specified by transition rules of two kinds. First, deductive inference rules that

add consequences to a state corresponding to critical pair computations. Second, simplification rules that combine deduction steps with deletions implementing an (approximative) notion of redundancy: An (oriented) inequality is *redundant*, if it can be replaced by a sequence of smaller steps (oriented or unoriented) with respect to $\prec$.

For a quasiordering $\leq$ presented by a set I_0 of ground inequalities, states are of the form $q = (I, R, S)$, where I is a set of ground inequalities and R and S are sets of (decreasing and increasing) rewrite rules. In the initial state q_0, R_0 and S_0 are empty and I_0—the *initial specification*—is irreflexive. The set of transition rules is denoted by C. A *run* of the procedure is a (finite or infinite) sequence $q_0, q_1, q_2, \ldots$ of states such that q_0 is an initial state and between all consecutive states there is a transition that applies some rule in C. A run *fails*, if I is non-empty in the limit. It *succeeds*, if it does not fail and the limit sets R_∞ and S_∞ yield a normal system. A run is *fair*, if every enabled transition is eventually executed. C is *correct*, if every fair run that does not fail succeeds and $\to^*_{R_\infty \cup S_\infty} = \to^*_{I_0}$. The deduction and simplification rules in C are defined as follows (c.f. [16]).

$$\frac{(I, R, S)}{I \cup \{s \to_I t\}, R, S)}, \qquad \text{(Deduce)}$$

if (s, t) is a critical pair. This rule can also be written as a pair of inference rules.

$$\frac{l_2 \to_S r_2 \qquad l_1[r_2]_p \to_R r_1}{l_1[l_2]_p \to_I r_1}, \qquad \frac{l_2 \to_S r_2[l_1]_p \qquad l_1 \to_R r_1}{l_2 \to_I r_2[r_1]_p}.$$

$$\frac{(I \cup \{s \to_I t\}, R, S)}{(I, R \cup \{s \to_R t\}, S)}, \qquad \frac{(I \cup \{t \to_I s\}, R, S)}{(I, R, S \cup \{t \to_S s\})}, \qquad \text{(Orient)}$$

if $s \succ t$.

$$\frac{(I \cup \{s \to_I t\}, R, S)}{(I, R, S)}, \quad \frac{(I, R \cup \{s \to_R t\}, S)}{(I, R, S)}, \quad \frac{(I, R, S \cup \{s \to_S t\})}{(I, R, S)}, \qquad \text{(Simplify)}$$

if $s \to_I t$ is redundant. Analogous rules simplify expressions in $\to_R$ and $\to_S$.

$$\frac{(I \cup \{s \to_I s\}, R, S)}{(I, R, S)}, \qquad \text{(Delete)}$$

Simplification rules and in particular Delete should be eagerly applied.

Theorem 1 ([16]). C *is sound and correct.*

A peculiarity is that Deduce rules are indispensible even in the ground case. Thus not all rules in C are simplification rules. Therefore C is more similar to ground ordered resolution than to ground equational completion and correctness is a limit property that need not be finitely satisfiable.

Lemma 1 ([16]). C *need not terminate.*

Proof. Let $b \succ f \succ a$ be a precedence for constants a, b and function f of arity one that is extended to a reduction ordering containing the subterm ordering. The relations $f(b) \to_I b$ and $f(a) \to_I b$ are oriented as $f(b) \longrightarrow_R b$ and $f(a) \longrightarrow_S b$. From the first rule, inequalities $f^n(a) \to_I b$ can be computed by DEDUCE for arbitrary n. All rules remain irredundant during the entire process, thus can never be simplified. □

4 Transforming the Initial Specification

We now define a two-step transformation $\tau_{12} = \tau_2 \circ \tau_1$, $\tau_{12} : Q \to Q'$ on states of C that allow us to enforce termination of C. Thereby Q is defined with respect to signature Σ and Q' with respect to a new signature Σ' defined below.

The first transformation $\tau_1 : T_\Sigma \to T_{\Sigma'}$ maps every $f \in \Sigma$ to a constant $f \in \Sigma'_0$. Σ' contains one single additional binary function symbol @. Terms are transformed as follows.

$$\tau_1(a) = a, \qquad \tau_1(f(t_1, \ldots, t_n)) = @(f, @(\tau_1(t_1), \ldots, @(\tau_1(t_{n-1}), \tau_1(t_n)) \ldots)),$$

for every constant $a \in \Sigma$, function $f \in \Sigma$ of arity n and terms $t_1, \ldots, t_n \in T_\Sigma$. We often write $f c_1 \ldots c_n$ instead of $@(f, @(c_1, \ldots, @(c_{n-1}, c_n) \ldots))$. τ_1 corresponds to currying in functional languages. τ_1 is extended homomorphically to pairs of terms, states and sets of states, setting

$$\begin{aligned}
\tau_1((s,t)) &= (\tau_1(s), \tau_1(t)), \\
\tau_1(I_1 \cup I_2) &= \tau_1(I_1) \cup \tau_1(I_2), \qquad \text{(and similarly for } R \text{ and } S\text{)}, \\
\tau_1(I,R,S) &= (\tau_1(I), \tau_1(R), \tau_1(S)), \\
\tau_1(Q_1 \cup Q_2) &= \tau_1(Q_1) \cup \tau_1(Q_2).
\end{aligned}$$

Lemma 2. *$|\tau_1(t)| = 2|t| - 1$ for all $t \in T_\Sigma$.*

Proof. By induction on the size of terms. In the base case, $|a| = 1$ for a constant a. $|\tau_1(a)| = |c_a| = 1 = 2 \cdot 1 - 1 = 2|a| - 1$.

In the induction step,

$$\begin{aligned}
|\tau_1(f(t_1,\ldots,t_k))| &= |@(f,@(\tau_1(t_1),\ldots @(\tau_1(t_{k-1},\tau_1(t_k)))))| \\
&= k+1+\sum_{i=1}^{k}|\tau_1(t_i)| \\
&= k+1+\sum_{i=1}^{k}(2|t_i|-1) \\
&= 1+\sum_{i=1}^{k}2|t_i| \\
&= 2(1+\sum_{i=1}^{k}|t_i|)-1 \\
&= 2|f(t_1,\ldots,t_k)|-1
\end{aligned}$$

□

Proposition 1. *Let Γ be a set of T_Σ-inequalities and let Q denote the quasiordering axioms (reflexivity and transitivity) for the relation $\leq$ together with monotonicity of all functions in Σ. Then for all $s,t \in T_\Sigma$,*

$$\Gamma \cup Q \models s \leq t \qquad \Leftrightarrow \qquad \tau_1(\Gamma \cup Q) \models \tau_1(s \leq t).$$

Proof. We first consider a proof in T_Σ. By induction on the size of proofs.

(i) If $s \leq t \in \Gamma$, then $\tau_1(s) \leq \tau_1(t) \in \tau_1(\Gamma)$.

(ii) If the last step in the proof has been reflexivity, then $s = t$ and therefore $\tau_1(s) = \tau_1(t)$.

(iii) If the last step in the proof has been transitivity, then there exists a T_Σ-term r and proofs of $s \leq r$ and $r \leq t$. The induction hypothesis yields proofs of $\tau_1(s) \leq \tau_1(r)$ and $\tau_1(r) \leq \tau_1(t)$. Then $\tau_1(s) \leq \tau_1(t)$ follows immediately from transitivity.

(iv) If the last step in the proof has been monotonicity, then $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$, say, and the induction hypothesis yields proofs of $s_1 \leq t_1$ to $s_n \leq t_n$ together with proofs of $\tau_1(s_1) \leq \tau_1(t_1)$ to $\tau_1(s_n) \leq \tau_1(t_n)$. Monotonicity of @ then yields in n steps the proofs

$$\begin{gathered}
s_{n-1}, s_n \leq t_{n-1}t_n, \\
s_{n-2}s_{n-1}s_n \leq t_{n-2}t_{n-1}t_n \\
\cdots\cdots \\
fs_1\ldots s_n \leq ft_1\ldots t_n.
\end{gathered}$$

The last line is $\tau_1(s) \leq \tau_1(t)$.

The converse direction is similar. □

We henceforth assume that @ has maximal weight in the signature.

The second transformation $\tau_2 : T_\Sigma \times T_\Sigma \to Q$ flattens terms. It is also standard. Intuitively, each subterm s of a term t is renamed by a fresh constant c_s and a new pair of definitional rewrite rules $s \to_R c_s$ and $c_s \to_S s$ is added to the initial specification.[1] After application of τ_2 (and orientation), the specification consists of two types of rewrite rules. *P-rules* (or *presentation rules*) of the form

$$f(c_1, \ldots, c_n) \to_{I \cup R} c_0, \qquad c_0 \to_{I \cup S} f(c_1, \ldots, c_n), \qquad c_1 \to_{I \cup R \cup S} c_2$$

for constants $c_1, \ldots, c_n \in C$ and function f of arity n that represent the initial inequalities and *D-rules* (or *definitional rules*) of the form

$$f(c_1, \ldots, c_n) \to_R c_0, \qquad c_0 \to_S f(c_1, \ldots, c_n),$$

for an n-ary function $f \in \Sigma$ and constants $c_0, \ldots, c_n \in C$ that represent the subterm structure. P- and D-rules are not necessarily disjoint. We also classify these rules in a different way. We call *F-rule* a P- or D-rule involving a function symbol and *C-rule* a P-rule involving only constant symbols. We extend τ_2 homomorphically to states and sets of states. We also extend $|.|$ homomorphically from terms to pairs of terms and states, setting

$$|(s,t)| = |s| + |t|,$$
$$|(I,R,S)| = \sum_{(s,t) \in I \cup R \cup S} |(s,t)|.$$

Lemma 3. *Let q be a state. Then*

$$|\tau_2(q)| \leq 2(k+2)|q| + 2n(k+1),$$

where k is the maximal arity of a function symbol occuring in q and $n \leq |q|$ is the number of rewrite rules in q. Moreover, every term in $\tau_2(q)$ has height at most 2.

Proof. For every term l in a pair $l \to r$ in q, $|l|$ equals the number of its subterms. $\tau_2(l)$ constists of two D-rules for every subterm of l. Since not all subterms need to be distinct, there are at most $2|l|$ D-rules stemming from $\tau_2(l)$. Moreover there is one P-rule, wich is also a C-rule, representing the rewrite rule $l \to r$. Let k be the maximal arity of a function symbol occuring in q. Then the size of a D-rule in $\tau_2(q)$ can be at most $k+2$. The size of a P-rule can be at most 2. This yields

$$|\tau_2(l \to r)| \leq 2(k+2)(|l| + |r|) + 2$$

[1] Alternatively one could also use the combined Knuth-Bendix procedure for equalities and inequalities in [12] and in particular the memoization algorithm defined there for a formalization. Then one would add an equational rewrite rule $s \to_T c_s$ to the combined specification.

and consequently

$$\begin{aligned}|\tau_2(q)| &= \sum_{l\to r\in G} \tau_2(l\to r)\\ &\leq 2(k+2)\sum_{l\to r\in q}(|l|+|r|)+\sum_{l\to r\in q} 2\\ &= 2(k+2)|q|+2n,\end{aligned}$$

where n is the number of rewrite rules in q.

Obviously, all terms in $|\tau_2(q)|$ have height at most 2, since all rewrite rules are either P- or D-rules. □

This upper bound is very rough, since the size change induced by τ_2 depends strongly on the amount of subterm sharing in terms in q.

Proposition 2. *Under the conditions of proposition 1, for all* $s, t \in T_\Sigma$,

$$\Gamma \cup Q \models s \leq t \qquad \Leftrightarrow \qquad \tau_2(\Gamma \cup Q) \models \tau_2(s \leq t).$$

Proposition 2 holds since the transformed expressions are definitional extensions. See [12] for a similar proof.

Corollary 1. *Let* q *be a state. Then*

$$|\tau_{12}(q)| \leq 16|q| - 14n,$$

where $n \leq |q|$ *is the number of pairs in* q.

Proof. First observe that the maximal arity in $\tau_1(q)$ is $k = 2$. Using this and the results of lemma 2 and lemma 3 yields

$$\begin{aligned}|\tau_2(\tau_1(q))| &\leq 8|\tau_1(q)| + 2n\\ &= 8(\sum_{l\to r\in q} |\tau_1(l)| + |\tau_1(r)|) + 2n\\ &= 8(\sum_{l\to r\in q} 2|l| - 1 + 2|r| - 1) + 2n\\ &= 16|q| - 14n.\end{aligned}$$

□

The following corollary follows immediately from corollary 1 and corollary 2.

Corollary 2. *Under the conditions of proposition 1, for all* $s, t \in T_\Sigma$,

$$\Gamma \cup Q \models s \leq t \qquad \Leftrightarrow \qquad \tau_{12}(\Gamma \cup Q) \models \tau_{12}(s \leq t).$$

We henceforth assume that all terms are built from one binary non-constant function symbol and that they are flat. We also presuppose without further discussion that both transformations have polynomial running time.

5 Termination Analysis

With the two-step transformation τ_{12} and a precedence $\prec$ that gives @ greater weight than all constants, the termination analysis of C is rather simple.

Theorem 2. *Let $q = \tau_{12}(q')$ for some $q' \in Q$. Then* C *terminates on q after $O(|q|)$ steps.*

Proof. By assumption, all pairs in q that can contribute to a DEDUCE-step are F- or C-rules of depth at most 2 and size at most 4 of the form

$$c_1c_2 \to_R c_3, \qquad c_1 \to_R c_2c_3, \tag{F}$$

$$c_1 \to_R c_2, \qquad c_1 \to_S c_2. \tag{C}$$

Thereby c_1, c_2 and c_3 are constants in C. If k is the number of contants in q, then q contains at most $2k^2(k-1)$ F-rules and $k(k-1)$ C-rules, since both kind of rules must contain at least two different constants. This and the sizes of F- and C-rules imply that $|q| \leq 8k^3 - 6k^2 - 2k$.

Let n be the number of (P- and D-)rules in q.

We analyze the possible DEDUCE-steps.

- F/F-overlaps. Consider the DEDUCE-step

$$\frac{c_1 \to_S c_2c_3 \qquad c_2c_3 \to_R c_4}{c_1 \to_I c_4}.$$

  The conclusion is a C-rule. This is the only way two F-rules can overlap.
- F/C-overlaps. Consider the DEDUCE-steps

$$\frac{c_1 \to_S c_2 \qquad c_2c_3 \to_R c_4}{c_1c_3 \to_I c_4}, \qquad \frac{c_1 \to_S c_2c_3 \qquad c_3 \to_R c_4}{c_1 \to_I c_2c_4}.$$

  The conclusions are F-rules. There are two similar cases, where the C-rule matches with c_3 in the left-hand and with c_2 in the right-hand F rule.
- C/C-overlaps. Consider the DEDUCE-step

$$\frac{c_1 \to_S c_2 \qquad c_2 \to_R c_3}{c_1 \to_I c_3}.$$

  The conclusion is a C-rule. This is the only way two C-rules can overlap.

We see that the DEDUCE-steps introduce only F- and C-rules. In particular, no new constants from C are introduced. Therefore the number of DEDUCE-steps in C is bounded by the total number of rules in Q as $2k^3 - k^2 - k - n$. k rules can at most be deleted, $2k^3 - k^2 - k - n$ rules can at most be oriented or simplified. Thus the overall number of steps in C is $O(k^3)$, which by our above estimation for $|q|$ is equal to $O(|q|)$. □

The subtle point in the analysis is that there is no need to introduce new constants. If one would continue the renaming during the completion and introduce a new constant for every new term that is generated by DEDUCE-steps in C, then the procedure need not terminate. See [12] for an example.

Theorem 3. *Ground non-symmetric Knuth-Bendix completion terminates in polynomial time.*

Proof. Let q_0 be the initial specification with $R_0 = S_0 = \emptyset$. According to the results of section 4, the transformation $\tau_{12} : q_0 \mapsto q$ takes polynomial running time and increases the size of q_0 by a linear factor. According to theorem 2, C takes polynomially many steps in the size of q, hence also in the size of q_0.

We may assume that each DEDUCE, ORIENT and DELETE step may be executed in constant (or at least polynomial) time. See [7] for a discussion of a related procedure. It remains to consider the cost of SIMPLIFY.

Unlike the equational KB-procedures, the SIMPLIFY-steps of C are search-based. We represent q as a directed graph with vertices consisting of constants or pairs of constants from C. There is an egde between vertices for every rule $l \to_{I \cup R \cup S} r$ in q. If q is built from k constants, then there may be at most $k + k^2$ vertices. According to the proof of theorem 2, there may be at most $k(k-1)$ edges corresponding to C-rules and $2k^2(k-1)$ edges corresponding to F-rules. We can then check for simplification by depth-first or breadth-first search along these edges, which is linear in the number of vertices plus the number of edges, hence polynomial. Thus every SIMPLIFY-step can be performed in polynomial time and the whole procedure therefore has polynomial running time in the size of the initial specification. □

Correctness of C does not depend on performing SIMPLIFY-steps. Thus the running time can be considerably improved by using no or an approximate implementation of SIMPLIFY. The proof of theorem 3 has the following consequence.

Corollary 3. *The decision procedure of ground non-symmetric rewriting (c.f. section 3) has polynomial running time in the size of the normal GTRS.*

Another direct consequence of theorem 3 is the following (c.f. [11]).

Corollary 4. *Reachability of a (equational) ground term rewrite system is decidable in polynomial time.*

This is particular interesting, when the rewrite system is non-terminating and therefore infinite.

Example 1. Consider again the GTRS from lemma 1. The transformation τ_{12} yields the rewrite rules $fb \to_R b$ and $fa \to_R b$. There are no critical pairs.

6 Detecting Embeddings

We now apply C as a metaprocedure to the development of two rule-based declarative and dynamic algorithms for detecting simple and homeomorphic embeddings in a state transition or constraint system represented by a set of ground inequalities. The inference rules are highly non-deterministic and therefore characterize classes of algorithms rather than particular instances. Concrete algorithms

can be obtained by refinement, imposing execution strategies. The decision procedures are interesting for the termination analysis of programs [2,8] and term rewrite systems [4] and for partial evaluation and program transformation [6].

We first define a simple *embedding relation* $\trianglelefteq$ by

$$t \trianglelefteq f(\dots, t, \dots)$$

for all $t \in T_\Sigma$. We say that a term s is *embedded* into a term t, if $s \trianglelefteq^* t$. Let $q = (I, R, S)$ be a state. A $\rightarrow_{I \cup R \cup S}$-sequence $s = t_0, t_1, \dots$ *contains an embedding*, if $t_i \trianglelefteq^* t_j$ holds for some t_i, t_i in s and $i \leq j$.

We first present two technical lemmata.

Lemma 4. *$s \trianglelefteq^* t$ iff $t = t[s]_p$ for some position p.*

Proof. (i) $s \trianglelefteq^* t$ implies $t = t[s]_p$ by induction on $\trianglelefteq^n$. For $n = 1$, $s \trianglelefteq t$ iff $t = f(\dots, s, \dots)$ or $s = t \in \Sigma_0$. Thus $t = t[s]_p$.

Now let $s \trianglelefteq^{n+1} t = f(t_1, \dots, t_n)$. Thus, by the definition of $\trianglelefteq$, $s \trianglelefteq^n t_i$ for some $1 \leq i \leq n$ and $t_i = t_i[s]_p$ by the induction hypothesis. Hence also $t = t[s]_{p'}$.

(ii) $t = t[s]_p$ implies $s \trianglelefteq^* t$ by induction on the size of terms. If $t = s \in \Sigma_0$, then trivially $s \trianglelefteq^* t$.

If $t = f(\dots, t_i[s], \dots)$, then $s \trianglelefteq^* t_i$ follows from the induction hypothesis and therefore $s \trianglelefteq^* \trianglelefteq t$ by definition of $\trianglelefteq$, which is equivalent to $s \trianglelefteq^* t$. □

Lemma 5. *s is a subterm of t iff $\tau_1(s)$ is a subterm of $\tau_1(t)$.*

Proof. (i) $t = t[s]$ implies $\tau_1(t) = \tau_1[\tau_1(s)]$ by induction on the size of t. If $t \in \Sigma_0$, then the result follows trivially from the definition of τ_1. If $t = f(t_1, \dots, t_n)$, then either $s = t$ or $t_i = t_i[s]$ for some $1 \leq i \leq n$. The first case is trivial. In the second case, the induction hypothesis yields $\tau_1(t_i) = \tau_1(t_i)[\tau_1(s)]$. Then $i - 1$ applications of @ yield $\tau_1(t) = \tau_1(t)[\tau_1(s)]$, according to the definition of τ_1.

(ii) $\tau_1(t) = \tau_1[\tau_1(s)]$ implies $t = t[s]$ by induction on the size of terms. The base case is similar to (i). Let $\tau_1(t) = f\tau_1(t_1) \dots \tau_1(t_n)$. Then either $s = t$ (which is trivial) or $\tau_1(t_i) = \tau_1(t_i)[\tau_1(s)]$ for some $1 \leq i \leq n$. Then the induction hypothesis yields $t_i = t_i[s]$. This implies $t = t[s]$ by definition of τ_1. □

We assume that all rules are labeled with P, D or both, depending on whether they are P- or D-rules.

Lemma 6. *Let $q = (I, R, S)$ be a state. The following statements are equivalent.*

(i) $s \rightarrow^*_{I \cup R \cup S} t$ *is an embedding.*
(ii) $\tau_1(s) \rightarrow^*_{I' \cup R' \cup S'} \tau_1(t)$ *is an embedding, where* $(I', R', S') = \tau_1(q)$.
(iii) $\tau_{12}(s) \rightarrow_{I'' \cup R'' \cup S''} \tau_{12}(t)$ *holds in the state* $\tau_{12}(q) = (I'', R'', S'')$, *where $\tau_{12}(s)$ names a subterm of $\tau_1(t)$ and all rules in the sequence are P-rules.*

Proof. ((i) equivalent to (ii)). First note that $s \rightarrow^*_{I \cup R \cup S} t$ iff $\tau_1(s) \rightarrow^*_{I' \cup R' \cup S'} \tau_1(t)$ by lemma 1. Moreover, $s \trianglelefteq^* t$ iff s is a subterm of t by lemma 4, iff $\tau_1(s)$ is a subterm of $\tau_1(t)$ by lemma 5, iff $\tau_1(t) = \tau_1(t)[\tau_1(s)]$ again by lemma 4.

((ii) equivalent to (iii)). $\tau_1(s) \rightarrow^*_{I' \cup R' \cup S'} \tau_1(t)$ iff $\tau_{12}(s) \rightarrow_{I'' \cup R'' \cup S''} \tau_{12}(t)$ by lemma 2, obviously by a sequence of P-rules. Now let $\tau_1(s) \trianglerighteq^* \tau_1(t)$. By lemma 4, this is equivalent to the fact that $\tau_1(s)$ is a subterm of $\tau_1(t)$. By definition of τ_2, equivalently, there is $\tau_{12}(s)$ names a subterm of $\tau_1(t)$. □

Lemma 6 is the basis for detecting embeddings.

Theorem 4. *Let $q_0 = (I_0, R_0, S_0)$ be an initial specification and $q = \tau_{12}(q_0) = (I, R, S)$. Let C_e be C together with the rule*

$$\frac{(I, R \cup \{c_1 c_2 \to_R^D c_3)\}, S)}{(I \cup \{c_3 \to_I^P c_1, c_3 \to_I^P c_2\}, R \cup \{c_1 c_2 \to_R^D c_3)\}, S)}. \qquad \text{(BACK)}$$

(i) An I_0-sequence in q_0 contains an embedding, iff a fair run of C_e constructs a cycle of P-rules from q.
(ii) C_e detects all embeddings in q_0 in polynomial time.

Proof. (ad i) By lemma 6, q_0 contains an embedding, iff q contains a rewrite sequence of P-rules that connects the name c_s of the initial term s with the (name c_t of the) final term t of the embedding. By lemma 4, s must be a subterm of t. By lemma 5 and lemma 6, subterms and embeddings are preserved by τ_1. Thus for each name of a term in q_0, BACK-steps eventually produces a rewrite sequence to the names of all its subterms. This yields a cyle iff there is an embedding.

(ad ii) The number of rules generated by BACK is $O(|q|)$ and therefore $O(|q_0|)$ by lemma 1 and the transformation τ_{12}; the closure under BACK can be done in polynomial time. Cycle detection can also be done in polynomial time with C and therefore C_e. This has been shown for graph structures in [12]. The basic idea, which applies also here, is that by the ordering constraints, every cycle must contain at least an R- and and S-step, hence a critical peak. Each cycle with k-edges can therefore eventually be collapsed into a cycle with two edges, which is simple to detect (in polynomial time). Therefore detection of embeddings has polynomial running time in the size of q_0. □

The cycle detection algorithm can be refined to the running time of depth-first search using a strategy and an on-the-fly construction of the precedence. See [12] for details. The main idea is as follows. As mentioned in the proof of theorem 3, the input state q to C can be represented as a directed graph with nodes consisting of constants and pairs of constants from C. Now the weight of each vertex can also be determined on the fly in a depth-first search along $\to_I$-rules, given vertices that are discovered later smaller weight. Using this strategy, cycle detection has the running time of depth-first search.

We now consider a more complex kind of embedding. A binary relation $\trianglelefteq$ on T_Σ is a *homeomorphic embedding*, if the following holds (c.f. [17]). For all $s = f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$ in $T_\Sigma(X)$, $s \trianglelefteq t$, iff either

(h1) $f \preceq g$ and $s_i \trianglelefteq t_{ji}$ for all $i = 1, \ldots, m$ and some $j_1, \ldots, j_m$ with $1 \leq j_1 < j_2 < \cdots < j_m \leq n$, or
(h2) $s \trianglelefteq t_i$ for some $i = 1, \ldots, n$, or
(h3) $s \prec t$, if $s, t \in \Sigma_0$.

Lemma 7. *Let $q = (I, R, S)$ be a state. Then $s \to_{I \cup R \cup S} t$ is a homeomorphic embedding iff $\tau_1(s) \to_{I' \cup R' \cup S'} \tau_1(t)$ is a homeomorphic embedding, where $(I', R', S') = \tau_1(q)$.*

Proof. $s \to_{I \cup R \cup S} t$ iff $\tau_1(s) \to_{I' \cup R' \cup S'} \tau_1(t)$ follows from lemma 1.

(i) Let $s \trianglelefteq t$. We argue by induction on the size of terms and assume that the precedence $\prec$ of Σ is inherited by Σ'. If $t \in \Sigma_0$, then $s \preceq t$ and therefore also $\tau_1(s) \prec \tau_1(q)$ by (h3), since $\tau_1(s), \tau_1(q) \in \Sigma'_0$ by definition of τ_1.

Now let $s = f(s_1, \ldots s_m)$ and $t = g(t_1, \ldots, t_n)$.

(case h1). Let $f \preceq g$ and $s_i \trianglelefteq t_{j_i}$ for all $1 \leq i \leq m$ and some $j_1, \ldots, j_m$ such that $1 \leq j_1 < j_2 < \cdots < j_m \leq n$. Then by definition of τ_1, $\tau_1(f), \tau_1(g) \in \Sigma'_0$ and $\tau_1(f) \preceq \tau_1(g)$, therefore also $\tau_1(f) \trianglelefteq \tau_1(g)$. Moreover the induction hypotesis yields $\tau_1(s_i) \trianglelefteq \tau_1(t_{j_i})$. Applying (h1) m times and (h2) $n - m$ times yields $gs_1 \ldots s_m \trianglelefteq f\tau_1(t_1) \ldots \tau_1(t_n)$, hence $\tau_1(s) \trianglelefteq \tau_1(t)$. Thereby (h2) is used to fill in those $\tau_1(t_j)$ that are not related by $\trianglelefteq$ to an $\tau_1(s_i)$.

(case h2). Let $s \trianglelefteq t_i$ for some $1 \leq i \leq n$. Then the induction hypothesis yields $\tau_1(s) \trianglelefteq \tau_1(t_i)$. Then n applications of (h2) yield $\tau_1(s) \trianglelefteq g\tau_1(t_1) \ldots \tau_1(t_n)$, hence $\tau_1(s) \trianglelefteq \tau_1(t)$.

(ii) Let $\tau_1(s) \trianglelefteq \tau_1(t)$. We argue again by induction on the size of terms. The cases are similar to those in (i), only the packing and unpacking of terms is done in the opposite direction. □

Lemma 8. *Let $q = (I, R, S)$ be a state. Then $s \to_{I \cup R \cup S} t$ is a homeomorphic embedding iff*

(i) $\tau_{12}(s) \to_{I'' \cup R'' \cup S''} \tau_{12}(t)$ *holds in the state* $\tau_{12}(q) = (I'', R'', S'')$ *and*
(ii) $\tau_{12}(s) \trianglelefteq \tau_{12}(t)$ *follows from the* embedding axioms

$$c_1 \trianglelefteq c'_1, c_2 \trianglelefteq c'_2, c_1 c_2 \to^D_R c_3, c'_1 c'_2 \to_R c'_3 \Rightarrow c_3 \trianglelefteq c'_3, \tag{h1'}$$

$$c \trianglelefteq c'_1, c'_1 c'_2 \to^D_R c'_3 \Rightarrow c \trianglelefteq c'_3, \tag{h2'}$$

$$c \trianglelefteq c'_2, c'_1 c'_2 \to^D_R c'_3 \Rightarrow c \trianglelefteq c'_3, \tag{h2'}$$

$$c \preceq c' \Rightarrow c \trianglelefteq c', \qquad \text{if } c, c' \in \Sigma_0. \tag{h3'}$$

Thereby all elements of Σ preserve their names and the precedence on Σ is inherited.

Proof. $s \to_{I \cup R \cup S} t$ iff $\tau_{12}(s) \to_{I'' \cup R'' \cup S''} \tau_{12}(t)$ holds by corollary 2. Homeomorphic embeddings are preserved under τ_1 by lemma 7. The remainder follows from the correctness of the encoding of (h1), (h2) and (h3), which is straightforward. In particular, the $\to_R$-rules in the definition trigger the choice of subterms according to (h1), (h2) and (h3). □

Lemma 8 is the basis for detecting homeomorphic embeddings.

Theorem 5. *Let $q_0 = (I_0, R_0, S_0)$ be an initial specification and $q = \tau_{12}(q_0) = (I, R, S)$. Let C_h be C together with the rules*[2]

$$\frac{c_1 \to^h_I c_1' \quad c_2 \to^h_I c_2' \quad c_1 c_2 \to^D_R c_3 \quad c_1' c_2' \to^D_R c_3'}{c_3 \to^h_I c_3'} \tag{H1}$$

$$\frac{c_1' \to^h_I c \quad c_1' c_2' \to^D_R c_3'}{c_3' \to^h_I c} \qquad \frac{c_2' \to^h_I c \quad c_1' c_2' \to^D_R c_3'}{c_3' \to^h_I c} \tag{H2}$$

$$\frac{c \preceq c'}{c' \to^h_I c} \qquad \text{if } c, c' \in \Sigma_0. \tag{H3}$$

All h-rules are considered also as P-rules. (H1), (H2) and (H3) are eagerly applied[3].

(i) An I_0-sequence in q_0 contains an embedding, iff a fair run of C_h constructs a cycle of P-rules from q that contains precisely one h-rule.
(ii) C_h detects all homeomorphic embeddings in q_0 in polynomial time.

Proof. (ad i) By lemma 8, q_0 contains a homeomorphic embedding, iff q contains a rewrite sequence of P-rules that connects the name c_s of the initial term s with the (name c_t of the) final term t of the embedding. The rules (H1), (H2) and (H3) are constructed such that all homeomorphic embeddings are enumerated according to the rules (h1'), (h2') and (h3') in lemma 8. The two rule sets are identical (which establishes correctness of (H1), (H2) and (H3)), only the arrows $\to^h_I$ are inverted with respect to $\preceq$. Therefore, a cycle containing precisely one h-rule is eventually constructed by C_h iff there is an embedding.

(ad ii) The rules (H1), (H2) and (H3) do not introduce any new constants. (H3) introduces at most $|\Sigma_0|^2$ edges, that is C-rules. Also (H1) and (H2) introduce only C-rules. Their number is bounded by $k(k-1)$, where k is the number of constants in C that occur in q and therefore in $O(|q_0|)$. The closure under these rules can be done in polynomial time. The remainder of the proof is analogous to that of 4. In addition one must use an execution strategy for C_h that ensures that only one h-rule is used for every cycle detection. □

Note that the (H2)-rules of C_h are essentially the BACK-rule of C_e read backwards. Generally, C_e uses a top-down approach to cycle construction, whereas C_h takes a bottom-up approach. We have chosen the bottom-up approach, sinc it is then easier to model axiom (h1'), but in principle also a top-down approach should be possible. There should also be much space for refining the algorithms, using different execution strategies on the deduction and simplification rules of C_e and C_h. The algorithms can also easily be combined with other rule-based extensions of C and integrated into the combined KB-procedure for equalities and inequalities in [12].

[2] They are written as inference rules because of their length.
[3] Alternatively, and more non-deterministically, similar rules for R and S should be used.

Finally, the algorithms work also in dynamic environments. When a new rule $s \to_I t$ is added to a state q, it must first be transformed by τ_{12}. The transformation with τ_1 is completely local. The transformation τ_2 is non-local. Names of subterms that appear in q might be relevant to s and t, but conversely, one can avoid changes in q. Therefore addition of new rules is modular and, due to the non-determinism of C and its extensions, can easily be integrated at running time.

Example 2. Consider the one step rewrite sequence $s \to_I t$ for terms $s = \neg(\neg 0 \vee 1)$ and $t = \neg(\neg(\neg 1 \wedge (1 \vee (\neg 0 \vee 1))))$, where $\Sigma = \{0, 1, \vee\wedge, \neg\}$ is the signature of the two-element boolean algebra. Assume a precedence $\prec$ for which all elements of Σ are incomparable (for homeomorphic embeddings, $\prec$ needs only be a quasiordering). It is easy to check with (h1), (h2) and (h3) that $s \trianglelefteq t$ (c.f. [17]). Let $\Sigma' = \{0, 1, \vee, \wedge, \neg, @\}$ be a new signature for which $\neg$, $\vee$ and $\wedge$ are constants. We obtain

$$\begin{aligned}
\tau_1(s) &= @(\neg, @(\vee, @(@(\neg, 0), 1))) \\
&= (\neg(\vee(\neg 0)1)), \\
\tau_1(t) &= @(\neg, @(\neg, @(\vee, @(@(\neg, 1), @(\wedge, @(1, @(\vee, @(@(\neg, 0), 1)))))))) \\
&= (\neg(\neg(\vee(\neg 1)(\wedge 1(\vee(\neg 0)1))))).
\end{aligned}$$

τ_2 yields the following rewrite rules.

$$\neg 0 \to_R^D c_0, \tag{1}$$
$$c_0 1 \to_R^D c_1, \tag{2}$$
$$\vee c_1 \to_R^D c_2, \tag{3}$$
$$\neg c_2 \to_R^D c_3, \tag{4}$$

$$1 c_2 \to_R^D c_4, \tag{5}$$
$$\wedge c_4 \to_R^D c_5, \tag{6}$$
$$\neg 1 \to_R^D c_6, \tag{7}$$
$$c_6 c_5 \to_R^D c_7, \tag{8}$$
$$\vee c_7 \to_R^D c_8, \tag{9}$$
$$\neg c_8 \to_R^D c_9, \tag{10}$$
$$\neg c_9 \to_R^D c_{10}, \tag{11}$$

$$c_3 \to_I^P c_{10}. \tag{12}$$

We then obtain

$$\frac{c_2 \preceq c_2}{c_2 \to_I^h c_2} \tag{13}$$

by (H3),

$$\frac{c_2 \to_I^h c_2 \quad 1c_2 \to_R^D c_4}{c_4 \to_I^h c_2} \tag{14}$$

by (H2) from the conclusion of (13) and (5),

$$\frac{c_4 \to_I^h c_2 \quad \wedge c_4 \to_R^D c_5}{c_5 \to_I^h c_2} \tag{15}$$

by (H2) from the conclusion of (14) and (6),

$$\frac{c_5 \to_I^h c_2 \quad c_6 c_5 \to_R^D c_7}{c_7 \to_I^h c_2} \tag{16}$$

by (H2) from the conclusion of (15) and (8),

$$\frac{c_7 \to_I^h c_2 \quad \vee c_7 \to_R^D c_8}{c_8 \to_I^h c_2} \tag{17}$$

by (H2) from the conclusion of (16) and (9),

$$\frac{\neg \preceq \neg}{\neg \to_I^h \neg} \tag{18}$$

by (H3),

$$\frac{c_8 \to_I^h c_2 \quad \neg \to_I^h \neg \quad \neg c_8 \to_R^D c_9 \quad \neg c_2 \to_R^D c_3}{c_9 \to_I^h c_3} \tag{19}$$

by (H1) from the conclusion of (17) and (18),

$$\frac{c_9 \to_I^h c_3 \quad \neg c_9 \to_R^D c_{10}}{c_{10} \to_I^h c_3} \tag{20}$$

by (H2) from the conclusion of (19) and (11). We have thus obtained the 2-cycle

$$c_2 \to_I c_1 0 \to_I^h c_2$$

from (12) and the conclusion of (20), which can easily be detected. Note that orientation, for instance with the precedence $c_2 \prec c_{10}$ would yield the critical pair

$$c_2 \to_S c_1 0 \to_R^h c_2.$$

Of course, C_e will generate more rules, but their number will remain finite. We do not speculate on whether homeomorphic embedding between boolean expressions has any significant meaning.

7 Conclusion and Further Work

We have shown that ground non-symmetric Knuth-Bendix terminates in polynomial time, using a two-step transformation on the initial specification, Here we presented the argument only for a KB-procedure for quasiorderings, but it immediately applies also to the KB-procedure for non-symmetric transitive relations from [13,16].

Corollary 5. *The ground KB-procedure for non-symmetric transitive relations from [16] terminates in polynomial time.*

The previous non-termination problem also concerns variants of ordered chaining calculi [1,15] for transitive relations and quasiorderings, that extend the respective KB-procedures to the clausal level. Traditionally, chaining calculi are presented in absence of monotonicity. But the definition of (ground) variants, where rewriting also occurs below contexts, seems simple. We conjecture that our termination result applies also to these extensions.

Another extension are the combined KB-procedures for equalities and inequalities [12]. As already mentioned, the termination analysis and in particular the renaming in transformation τ_2 could be done directly with this extension and in particular the memoization algorithm presented there. In particular, our termination result can easily be transfered (equational ground KB-completion terminates a fortiori).

Corollary 6. *The combined ground KB-procedure for equalities and inequalities from [12] terminates in polynomial time.*

A third interesting extension are ground non-symmetric KB-procedures modulo equational axioms. Here, since the equational axioms are non-ground, the KB-procedures need no longer terminate. A particular procedure for the associativity and commutativity axiom, even further normalizing with respect to idempotence has been given in [14]. For this particular case of ground non-symmetric KB-completion modulo AC, it is straightforward to show termination.

Lemma 9. *Ground non-symmetric Knuth-Bendix completion modulo AC terminates.*

Proof. The main obeservation is that renaming decouples the AC-part from the rest of the rewrite system. Equational rewriting is used to denote the renaming of terms with AC-symbols at the root. Therefore extended rules and overlaps between AC-terms arise only within the equational part, termination of which is standard. □

Depending on termination of the ground KB-procedure we have also developed two rule-based, declarative and dynamic (classes of) algorithms to detect embeddings and homeomorphic embeddings in ground rewrite sequences. These algorithms have interesting applications both in the field of termination analysis of term rewrite systems and programs and in constraint-based analysis, partial evaluation and program transformation.

In the first field, assume that the program is given by a finite set of rewrite rules. We have results at least for the ground case.

Lemma 10. *If a term rewrite system contains an embedding, then it does not terminate.*

But the converse need not hold. See [4] for a counterexample.

Lemma 11 ([4]). *If a term rewrite system does not terminate, then it contains a homeomorphic embedding.*

Again, the converse does not hold [4]. But at least, absence of homeomorphic embeddings implies termination. Our algorithm again decides only the ground case. In the non-ground case, there is the following negative result.

Theorem 6 ([10]). *It is undecidable, whether a term rewrite system contains a homeomorphic embedding.*

Due to the problems with variable critical pairs mentioned in the introduction, lifting our detection algorithms to the non-ground level is problematic. At least it is possible, when all terms in the initial specifications are linear (every variable occurs at most once in a term) [16]. Our algorithms may then serve as semi-decision procedures. Homeomorphic embeddings also arise in the field of partial deduction with and termination analysis of logic programs [2,8]. There, detection of embeddings also must be done online. The usual algorithms are not declarative and seem more difficult to understand, implement and analyze than ours.

In the field of ground term rewriting, many interesting properties are expressible in terms of reachability. It should be very interesting to revisit existing decision procedures from the completion point of view.

In the field of (online) partial evaluation and program transformation or optimization, there is a method called *supercomplation* (c.f. [6] for an introduction) that is based on homeomorphic embeddings. Here again, one usually works in a non-ground setting. Beyond detecting these embeddings, there are also several rules that determine how to handle them. It might be interesting to consider integrating the whole approach in the Knuth-Bendix completion framework.

References

1. L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 384–393. IEEE Computer Society Press, 1994.
2. R. Bol. Loop checking in partial deduction. *The Journal of Logic Programming*, 16(1,2):25–46, 1993.
3. H. Comon. Completion of rewrite systems with membership constraints. In *Int. Coll. on Automata, Languages and Programming, ICALP'92*, volume 632 of *LNCS*. Springer-Verlag, 1992.
4. N. Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3:69–116, 1987.

5. Baader F. and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
6. Robert Glück and Morten Heine Sørensen. A roadmap to metacomputation by supercompilation. In O. Danvy, R. Glück, and P. Thiemann, editors, *Partial Evaluation*, volume 1110, pages 137–160. Springer-Verlag, 1996.
7. D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *Rewriting Techniques and Applications, 8th International Conference, RTA-97*, volume 1232 of *LNCS*, pages 23–37. Springer-Verlag, 1997.
8. M. Leuschel. On the power of homeomorphic embedding for online termination. In G. Levi, editor, *Static Analysis Symposium, SAS'98*, volume 1503 of *LNCS*, pages 230–245. Springer-Verlag, 1998.
9. J. Levy and J. Agustí. Bi-rewrite systems. *J. Symbolic Comput.*, 22:279–314, 1996.
10. D. A. Plaisted. The undecidability of self embedding for term rewriting systems. *Inf. Proc. Lett.*, 20:61–64, 1985.
11. D.A. Plaisted. Polynomial time termination and constraint satisfaction tests. In C. Kirchner, editor, *Pewriting Techniques and Applications, 5th International Conference, RTA-93*, volume 690 of *LNCS*, pages 405–420. Springer-VErlag, 1993.
12. G. Struth. Declarative graph algorithms via Knuth-Bendix completion. submitted.
13. G. Struth. *Canonical Transformations in Algebra, Universal Algebra and Logic.* PhD thesis, Institut für Informatik, Universität des Saarlandes, 1998.
14. G. Struth. An algebra of resolution. In L. Bachmair, editor, *Rewriting Techniques and Applications, 11th International Conference*, volume 1833 of *LNCS*, pages 214–228. Springer-Verlag, 2000.
15. G. Struth. Deriving focused calculi for transitive relations. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference*, volume 2051 of *LNCS*, pages 291–305. Springer-Verlag, 2001.
16. G. Struth. Knuth-Bendix completion for non-symmetric transitive relations. In M. van den Brand and R. Verma, editors, *Second International Workshop on Rule-Based Programming (RULE2001)*, volume 59 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.
17. W. Wechler. *Universal Algebra for Computer Scientists.* Springer-Verlag, 1992.