Match-Bounds with Dependency Pairs for Proving Termination of Rewrite Systems

Martin Korp and Aart Middeldorp

Institute of Computer Science
University of Innsbruck
Austria

Abstract. The match-bound technique is a recent and elegant method to prove the termination of rewrite systems using automata techniques. To increase the applicability of the method we incorporate it into the dependency pair framework. The key to this is the introduction of two new enrichments which take the special properties of dependency pair problems into account.

1 Introduction

The use of word automata for proving the termination of string rewrite systems was proposed by Geser, Hofbauer, and Waldmann [5]. In [8] tree automata were used to cover left-linear term rewrite systems. We extended the latter work to arbitrary term rewrite systems in [17] by considering so-called quasi-deterministic tree automata. Variations and improvements of the basic method for string rewrite systems are discussed in [6, 7]. The fact that the method has been implemented in several different termination provers ([10, 16, 21–23]) is a clear witness of the success of the automata based approach.

In this paper we integrate the method into the dependency pair framework [11, 20]. To guarantee a successful integration we need to modularise the method in order to be able to simplify dependency pair problems. We achieve this by introducing two new enrichments which exploit the special properties of dependency pair problems.

The remainder of the paper is organised as follows. In the next section we recall basic definitions concerning the automata theory approach for proving termination and the dependency pair framework. In Section 3 we introduce the concept of e-DP-bounds which is based on two new enrichments that allow us to simplify dependency pair problems. In Section 4 we consider right-hand sides of forward closures to reduce the set of terms which have to be considered. To simplify the discussion, we restrict ourselves to left-linear rewrite systems. The extension to non-left-linear term rewrite systems is sketched in Section 5. Experimental data is presented in Section 6. Missing proofs can be found in the full version, which is available from `http://cl-informatik.uibk.ac.at/~mkorp/`.

2 Preliminaries

We assume familiarity with term rewriting [2] and tree automata [3]. General knowledge of the match-bound technique [5, 8] and dependency pairs [1, 11, 13, 15] will be helpful.

Match-Bounds

Let $\mathcal{F}$ be a signature, $\mathcal{R}$ a term rewrite system (TRS for short) over $\mathcal{F}$, and $L \subseteq \mathcal{T}(\mathcal{F})$ a set of ground terms. The set $\{t \in \mathcal{T}(\mathcal{F}) \mid s \to_{\mathcal{R}}^* t \text{ for some } s \in L\}$ of descendants of L is denoted by $\to_{\mathcal{R}}^*(L)$. Given a set $N \subseteq \mathbb{N}$ of natural numbers, the signature $\mathcal{F} \times N$ is denoted by $\mathcal{F}_N$. Here function symbols (f, n) with $f \in \mathcal{F}$ and $n \in N$ have the same arity as f and are written as f_n. The mappings $\mathrm{lift}_c \colon \mathcal{F} \to \mathcal{F}_{\mathbb{N}}$, $\mathrm{base} \colon \mathcal{F}_{\mathbb{N}} \to \mathcal{F}$, and $\mathrm{height} \colon \mathcal{F}_{\mathbb{N}} \to \mathbb{N}$ are defined as $\mathrm{lift}_c(f) = f_c$, $\mathrm{base}(f_i) = f$, and $\mathrm{height}(f_i) = i$ for all $f \in \mathcal{F}$ and $c, i \in \mathbb{N}$. They are extended to terms and to sets of terms in the obvious way. For a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and a constant $c \in \mathbb{N}$ the mapping $\mathrm{lift}_c^\epsilon \colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$ is defined as follows: $\mathrm{lift}_c^\epsilon(t) = t$ if t is a variable and $\mathrm{lift}_c^\epsilon(t) = f_c(\mathrm{lift}_0(t_1), \ldots, \mathrm{lift}_0(t_n))$ if $t = f(t_1, \ldots, t_n)$. Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ and e a function that maps every rewrite rule $l \to r \in \mathcal{R}$ to a nonempty subset of $\mathcal{FP}\mathrm{os}(l)$, where $\mathcal{FP}\mathrm{os}(l) = \{p \in \mathcal{P}\mathrm{os}(l) \mid l(p) \in \mathcal{F}\}$. The TRS $e(\mathcal{R})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rewrite rules $l' \to \mathrm{lift}_c(r)$ for which there exists a rule $l \to r \in \mathcal{R}$ such that $\mathrm{base}(l') = l$ and $c = 1 + \min\{\mathrm{height}(l'(p)) \mid p \in e(l, r)\}$. Let $c \in \mathbb{N}$. The restriction of $e(\mathcal{R})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $e_c(\mathcal{R})$. We consider three concrete functions e in the following: $\mathrm{top}(l \to r) = \{\epsilon\}$, $\mathrm{roof}(l \to r) = \{p \in \mathcal{FP}\mathrm{os}(l) \mid \mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l|_p)\}$, and $\mathrm{match}(l \to r) = \mathcal{FP}\mathrm{os}(l)$. Let $e \in \{\mathrm{top}, \mathrm{roof}, \mathrm{match}\}$ and L a set of terms. The TRS $\mathcal{R}$ is called *e-bounded* for L if there exists a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms in $\to_{e(\mathcal{R})}^*(\mathrm{lift}_0(L))$ is at most c. If we want to precise the bound c, we say that $\mathcal{R}$ is e-bounded for L *by* c. If we do not specify the set of terms L then it is assumed that $L = \mathcal{T}(\mathcal{F})$.

Theorem 1 (Geser *et al.* [8])**.** *If a left-linear TRS $\mathcal{R}$ is top-bounded, roof-bounded, or both right-linear and match-bounded for a language L then $\mathcal{R}$ is terminating on L.* □

In order to prove that a TRS $\mathcal{R}$ is e-bounded for some language L and some $e \in \{\mathrm{top}, \mathrm{roof}, \mathrm{match}\}$, the idea is to construct a tree automaton that is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$. A tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ is said to be *compatible* with some TRS $\mathcal{R}$ and some language L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma \colon \mathcal{V}\mathrm{ar}(l) \to Q$ such that $l\sigma \to_\Delta^* q$ it holds that $r\sigma \to_\Delta^* q$.

Dependency Pairs

The dependency pair method [1] is a powerful approach for proving termination of TRSs. The dependency pair framework [12, 20] is a modular reformulation and

improvement of this approach. We present a simplified version which is sufficient for our purposes.

Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The signature $\mathcal{F}$ is extended with symbols $f^\sharp$ for every symbol $f \in \{\mathrm{root}(l) \mid l \to r \in \mathcal{R}\}$, where $f^\sharp$ has the same arity as f, resulting in the signature $\mathcal{F}^\sharp$. If $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with $\mathrm{root}(t)$ defined then $t^\sharp$ denotes the term that is obtained from t by replacing its root symbol with $\mathrm{root}(t)^\sharp$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with a defined root symbol that is not a proper subterm of l then the rule $l^\sharp \to t^\sharp$ is a *dependency pair* of $\mathcal{R}$. The set of dependency pairs of $\mathcal{R}$ is denoted by $\mathrm{DP}(\mathcal{R})$. A *DP problem* is a pair of TRSs $(\mathcal{P}, \mathcal{R})$ such that symbols in $\{\mathrm{root}(l), \mathrm{root}(r) \mid l \to r \in \mathcal{P}\}$ do neither occur in $\mathcal{R}$ nor in proper subterms of the left and right-hand sides of rules in $\mathcal{P}$. The problem is said to be *finite* if there is no infinite sequence $s_1 \xrightarrow{\epsilon}_{\mathcal{P}} t_1 \to^*_{\mathcal{R}} s_2 \xrightarrow{\epsilon}_{\mathcal{P}} t_2 \to^*_{\mathcal{R}} \cdots$ such that all terms $t_1, t_2, \ldots$ are terminating with respect to $\mathcal{R}$. Such an infinite sequence is said to be *minimal*. Here the ϵ in $\xrightarrow{\epsilon}_{\mathcal{P}}$ denotes that the application of the rule in $\mathcal{P}$ takes place at the root position. The main result underlying the dependency pair approach states that a TRS $\mathcal{R}$ is terminating if and only if the DP problem $(\mathrm{DP}(\mathcal{R}), \mathcal{R})$ is finite.

In order to prove finiteness of a DP problem a number of so-called *DP processors* have been developed. DP processors are functions that take a DP problem as input and return a set of DP problems as output. In order to be employed to prove termination they need to be sound, that is, if all DP problems in a set returned by a DP processor are finite then the initial DP problem is finite. In addition, to ensure that a DP processor can be used to prove non-termination it must be complete which means that if one of the DP problems returned by the DP processor is not finite then the original DP problem is not finite.

3 DP-Bounds

To prove finiteness of a DP problem $(\mathcal{P}, \mathcal{R})$ it must be shown that it admits no minimal rewrite sequence. This is done by removing step by step those rewrite rules in $\mathcal{P}$ which cannot be used infinitely often in any minimal rewrite sequence. In each step a different DP processor can be applied. As soon as $\mathcal{P}$ is empty, we can conclude that the DP problem $(\mathcal{P}, \mathcal{R})$ is finite.

It is easy to incorporate the match-bound technique into the DP framework by defining a processor that checks for e-boundedness of $\mathcal{P} \cup \mathcal{R}$.

Theorem 2. *The DP processor*

$$(\mathcal{P}, \mathcal{R}) \mapsto \begin{cases} \varnothing & \text{if } \mathcal{P} \cup \mathcal{R} \text{ is left-linear and either top-bounded,} \\ & \text{roof-bounded, or both linear and match-bounded} \\ & \text{for } \mathcal{T}(\mathcal{F}) \\ \{(\mathcal{P}, \mathcal{R})\} & \text{otherwise} \end{cases}$$

where $\mathcal{F}$ is the signature of $\mathcal{P} \cup \mathcal{R}$, is sound and complete. □

This DP processor either succeeds by proving that the combined TRS $\mathcal{P} \cup \mathcal{R}$ is e-bounded or, when the e-boundedness of $\mathcal{P} \cup \mathcal{R}$ cannot be proved, it returns the initial DP problem. Since the construction of a compatible tree automaton does not terminate for TRSs that are not e-bounded, the latter situation typically does not happen. Hence the DP processor of Theorem 2 is applicable only at the leaves of the DP search tree, which means that it can be used only as a last option in a termination proving strategy. So it cannot cooperate with other DP processors.

Below we address this problem by adapting the match-bound technique in such a way that it can remove single rules of $\mathcal{P}$. We introduce two new enrichments top-DP$(\mathcal{P}, s \to t, \mathcal{R})$ and match-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to achieve this. The basic idea behind these TRSs is that every height increasing infinite sequence descends from an infinite sequence of $(\mathcal{P}, \mathcal{R})$ in which the rule $s \to t$, which is to be removed from $\mathcal{P}$, is used infinitely often.

To simplify the presentation we consider only left-linear TRSs. The extension to non-left-linear TRSs is briefly discussed in Section 5.

Definition 3. *Let $\mathcal{S}$ be a TRS over a signature $\mathcal{F}$. The TRS e-DP$(\mathcal{S})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rules $l' \to \mathrm{lift}_c(r)$ such that $\mathrm{base}(l') \to r \in \mathcal{S}$ and*

$$c = \min\left(\{\mathrm{height}(l'(\epsilon))\} \cup \{1 + \mathrm{height}(l'(p)) \mid p \in e(\mathrm{base}(l'), r)\}\right)$$

Given a DP problem $(\mathcal{P}, \mathcal{R})$ and a rule $s \to t \in \mathcal{P}$, the TRS e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ is defined as the union of e-DP$((\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R})$ and $e(s \to t)$. The restriction of e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by e-DP$_c(\mathcal{P}, s \to t, \mathcal{R})$.

Example 4. Consider the DP problem $(\mathcal{P}, \mathcal{R})$ with $\mathcal{R}$ consisting of the rewrite rules $\mathsf{f}(\mathsf{g}(x), y) \to \mathsf{g}(\mathsf{h}(x, y))$ and $\mathsf{h}(x, y) \to \mathsf{f}(x, \mathsf{g}(y))$, and $\mathcal{P} = \mathrm{DP}(\mathcal{R})$ consisting of $\mathsf{F}(\mathsf{g}(x), y) \to \mathsf{H}(x, y)$ and $\mathsf{H}(x, y) \to \mathsf{F}(x, \mathsf{g}(y))$. Let $s \to t$ be the first of the two dependency pairs. Then match-DP$(\mathcal{R})$ contains the rules

$$\begin{array}{ll} \mathsf{f}_0(\mathsf{g}_0(x), y) \to \mathsf{g}_0(\mathsf{h}_0(x, y)) & \mathsf{h}_0(x, y) \to \mathsf{f}_0(x, \mathsf{g}_0(y)) \\ \mathsf{f}_0(\mathsf{g}_1(x), y) \to \mathsf{g}_0(\mathsf{h}_0(x, y)) & \mathsf{h}_1(x, y) \to \mathsf{f}_1(x, \mathsf{g}_1(y)) \\ \mathsf{f}_2(\mathsf{g}_0(x), y) \to \mathsf{g}_1(\mathsf{h}_1(x, y)) & \cdots \end{array}$$

match-DP$(\mathcal{P} \setminus \{s \to t\})$ contains

$$\begin{array}{ll} \mathsf{H}_0(x, y) \to \mathsf{F}_0(x, \mathsf{g}_0(y)) & \mathsf{H}_1(x, y) \to \mathsf{F}_1(x, \mathsf{g}_1(y)) \\ \mathsf{H}_2(x, y) \to \mathsf{F}_2(x, \mathsf{g}_2(y)) & \cdots \end{array}$$

and match$(s \to t)$ contains

$$\begin{array}{ll} \mathsf{F}_0(\mathsf{g}_0(x), y) \to \mathsf{H}_1(x, y) & \mathsf{F}_1(\mathsf{g}_0(x), y) \to \mathsf{H}_1(x, y) \\ \mathsf{F}_0(\mathsf{g}_1(x), y) \to \mathsf{H}_1(x, y) & \cdots \end{array}$$

The union of these three infinite TRSs constitutes match-DP$(\mathcal{P}, s \to t, \mathcal{R})$. If we replace match$(s \to t)$ by match-DP$(\{s \to t\})$, which consists of the rules

$$\begin{aligned} \mathsf{F}_0(\mathsf{g}_0(x), y) &\to \mathsf{H}_0(x, y) & \mathsf{F}_1(\mathsf{g}_0(x), y) &\to \mathsf{H}_1(x, y) \\ \mathsf{F}_0(\mathsf{g}_1(x), y) &\to \mathsf{H}_0(x, y) & &\cdots \end{aligned}$$

we obtain the TRS match-DP$(\mathcal{P} \cup \mathcal{R})$.

The idea now is to use the enrichment e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to simplify the DP problem $(\mathcal{P}, \mathcal{R})$ into $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$. For that we need the property defined below.

Definition 5. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and let $s \to t \in \mathcal{P}$. We call $(\mathcal{P}, \mathcal{R})$ e-DP-bounded for $s \to t$ and a set of terms L if there exists a $c \in \mathbb{N}$ such that the height of function symbols occurring in terms in $\to^*_{e\text{-DP}(\mathcal{P}, s \to t, \mathcal{R})}(\text{lift}_0(L))$ is at most c.*

Moreover, we need to ensure that every restriction of e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to a finite signature does not admit minimal rewrite sequences with infinitely many $\xrightarrow{\epsilon}_{e(s \to t)}$ rewrite steps. For $e = \text{top}$ this is shown below. Note that if we would use e-DP$(\mathcal{P} \cup \mathcal{R})$ instead of e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ then this property would not hold because every rewrite sequence in $\mathcal{P} \cup \mathcal{R}$ can be simulated by an e-DP$_0(\mathcal{P} \cup \mathcal{R})$-sequence.

Lemma 6. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem, let $s \to t \in \mathcal{P}$, and let $c \geqslant 0$. The TRS* top-DP$_c(\mathcal{P}, s \to t, \mathcal{R})$ *does not admit rewrite sequences with infinitely many $\xrightarrow{\epsilon}_{\text{top}(s \to t)}$ rewrite steps.*

Proof. Assume to the contrary that there is such an infinite rewrite sequence

$$s_1 \xrightarrow{\epsilon}_{\text{top}(s \to t)} t_1 \to^*_{\text{top-DP}((\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R})} s_2 \xrightarrow{\epsilon}_{\text{top}(s \to t)} t_2 \to^*_{\text{top-DP}((\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R})} \cdots$$

Because the root symbols in $\mathcal{P}$ do not appear anywhere else in $\mathcal{P}$ or $\mathcal{R}$, we know that only rewrite rules from top-DP$(\mathcal{P} \setminus \{s \to t\})$ and top$(s \to t)$ are applied at root positions. Every rewrite rule $l \to r$ in top-DP$(\mathcal{P} \setminus \{s \to t\})$ has the property that height$(l(\epsilon)) =$ height$(r(\epsilon))$. Hence height$(t_i(\epsilon)) =$ height$(s_{i+1}(\epsilon))$ for all $i \geqslant 1$. By definition, for every $l \to r \in \text{top}(s \to t)$ we have height$(r(\epsilon)) =$ height$(l(\epsilon)) + 1$ and thus height$(t_i(\epsilon)) =$ height$(s_i(\epsilon)) + 1$ for all $i \geqslant 1$. It follows that height$(t_{c+1}(\epsilon)) \geqslant c + 1$, contradicting the assumption. □

Theorem 7. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and let $s \to t \in \mathcal{P}$ such that $(\mathcal{P}, \mathcal{R})$ is top-DP-bounded for $s \to t$ and a set of terms L. If $\mathcal{P} \cup \mathcal{R}$ is left-linear then $(\mathcal{P}, \mathcal{R})$ is finite for L if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite for L.*

Proof. The only-if direction is trivial. For the if direction, suppose that the DP problem $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite for L. If $(\mathcal{P}, \mathcal{R})$ is not finite for L then there exists a minimal rewrite sequence

$$s_1 \xrightarrow{\epsilon}_{s \to t} t_1 \to^*_{(\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R}} s_2 \xrightarrow{\epsilon}_{s \to t} t_2 \to^*_{(\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R}} s_3 \xrightarrow{\epsilon}_{s \to t} \cdots$$

with $s_1 \in L$. Due to left-linearity, this sequence can be lifted to an infinite top-DP$(\mathcal{P}, s \to t, \mathcal{R})$ rewrite sequence starting from $\mathrm{lift}_0(s_1)$. Since the original sequence contains infinitely many $\xrightarrow{\epsilon}_{s\to t}$ rewrite steps the lifted sequence contains infinitely many $\xrightarrow{\epsilon}_{\mathrm{top}(s\to t)}$ rewrite steps. Moreover, because $(\mathcal{P}, \mathcal{R})$ is top-DP-bounded for L, there is a $c \geqslant 0$ such that the height of every function symbol occurring in a term in the lifted sequence is at most c. Hence the employed rules must come from top-DP$_c(\mathcal{P}, s \to t, \mathcal{R})$ and therefore top-DP$_c(\mathcal{P}, s \to t, \mathcal{R})$ contains an infinite rewrite sequence consisting of infinitely many $\xrightarrow{\epsilon}_{\mathrm{top}(s\to t)}$ rewrite steps. This however is excluded by Lemma 6. □

If we restrict Lemma 6 to *minimal* rewrite sequences, it also holds for $e =$ match provided $\mathcal{P}$ and $\mathcal{R}$ are non-duplicating. The proof is considerably more complicated and omitted for reasons of space.

Lemma 8. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem, let $s \to t \in \mathcal{P}$, and let $c \geqslant 0$. If $\mathcal{P} \cup \mathcal{R}$ is non-duplicating then the TRS* match-DP$_c(\mathcal{P}, s \to t, \mathcal{R})$ *does not admit minimal rewrite sequences with infinitely many $\xrightarrow{\epsilon}_{\mathrm{match}(s\to t)}$ rewrite steps.* □

Theorem 9. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and let $s \to t \in \mathcal{P}$ such that $(\mathcal{P}, \mathcal{R})$ is match-DP-bounded for $s \to t$ and a set of terms L. If $\mathcal{P} \cup \mathcal{R}$ is linear then $(\mathcal{P}, \mathcal{R})$ is finite for L if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite for L.*

Proof. Similarly to the proof of Theorem 7, using Lemma 8 instead of Lemma 6. Note that in the presence of left-linearity, the non-duplicating requirement in Lemma 8 is equivalent to linearity. □

We conjecture that Lemma 6 also holds for $e =$ roof. A positive solution is important as roof-bounds are strictly more powerful than top-bounds [8, 17].

Theorem 10. *The DP processor*

$$(\mathcal{P}, \mathcal{R}) \mapsto \begin{cases} \{(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})\} & \text{if } \mathcal{P} \cup \mathcal{R} \text{ is left-linear and top-DP-bounded} \\ & \text{or linear and match-DP-bounded for } s \to t \\ & \text{and } \mathcal{T}(\mathcal{F}) \\ \{(\mathcal{P}, \mathcal{R})\} & \text{otherwise} \end{cases}$$

where $\mathcal{F}$ is the signature of $\mathcal{P} \cup \mathcal{R}$, is sound and complete.

Proof. Immediate consequence of Theorems 7 and 9. □

Example 11. We show that the DP problem $(\mathcal{P}, \mathcal{R})$ of Example 4 over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{F}, \mathsf{H}\}$ is match-DP-bounded for $\mathsf{F}(\mathsf{g}(x), y) \to \mathsf{H}(x, y)$ by constructing a compatible tree automaton. As starting point we consider the initial tree automaton

$$\begin{array}{lll} \mathsf{a}_0 \to 1 & \mathsf{f}_0(1,1) \to 1 & \mathsf{g}_0(1) \to 1 \\ \mathsf{h}_0(1,1) \to 1 & \mathsf{F}_0(1,1) \to 2 & \mathsf{H}_0(1,1) \to 2 \end{array}$$

which accepts the set of all ground terms that have F_0 or H_0 as root symbol and a_0, f_0, g_0, and h_0 below the root. Since $\mathsf{F}_0(\mathsf{g}_0(x), y) \to_{\text{match}(s\to t)} \mathsf{H}_1(x, y)$ and $\mathsf{F}_0(\mathsf{g}_0(1), 1) \to^* 2$, we add the transition $\mathsf{H}_1(1, 1) \to 2$. Next we consider $\mathsf{H}_1(x, y) \to_{\text{match-DP}(\mathcal{P}\setminus\{s\to t\})} \mathsf{F}_1(x, \mathsf{g}_1(y))$ with $\mathsf{H}_1(1, 1) \to 2$. By adding the transitions $\mathsf{F}_1(1, 3) \to 2$ and $\mathsf{g}_1(1) \to 3$ this compatibility violation is solved. After that the rewrite rule $\mathsf{F}_1(\mathsf{g}_0(x), y) \to_{\text{match}(s\to t)} \mathsf{H}_1(x, y)$ and the derivation $\mathsf{F}_1(\mathsf{g}_0(1), 3) \to^* 2$ give rise to the transition $\mathsf{H}_1(1, 3) \to 2$. Finally we have $\mathsf{H}_1(x, y) \to_{\text{match-DP}(\mathcal{P}\setminus\{s\to t\})} \mathsf{F}_1(x, \mathsf{g}_1(y))$ and $\mathsf{H}_1(1, 3) \to 2$. In order to ensure $\mathsf{F}_1(1, \mathsf{g}_1(3)) \to^* 2$ we reuse the transition $\mathsf{F}_1(1, 3) \to 2$ and add the new transition $\mathsf{g}_1(3) \to 3$. After that step, the obtained tree automaton is compatible with match-DP$(\mathcal{P}, s \to t, \mathcal{R})$. Hence the DP problem $(\mathcal{P}, \mathcal{R})$ is match-DP-bounded for $\mathsf{F}(\mathsf{g}(x), y) \to \mathsf{H}(x, y)$ by 1. Applying the DP processor of Theorem 10 yields the new DP problem $(\{\mathsf{H}(x, y) \to \mathsf{F}(x, \mathsf{g}(y))\}, \mathcal{R})$, which is easily (and automatically by numerous DP processors) shown to be finite. We note that the DP processor of Theorem 2 fails on $(\mathcal{P}, \mathcal{R})$.

To ensure that the TRS e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ can assist to prove finiteness of the DP problem $(\mathcal{P}, \mathcal{R})$, it is crucial that every minimal rewrite sequence in $(\mathcal{P}, \mathcal{R})$ with infinitely many $\xrightarrow{\epsilon}_{s\to t}$ rewrite steps can be simulated by an infinite height increasing sequence in e-DP$(\mathcal{P}, s \to t, \mathcal{R})$. To this end it is important that rewrite rules in e-DP$((\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R})$ do not propagate the minimal height of the contracted redex unless the height of the root symbol of the redex is minimal. This is the reason for the slightly complicated definition of c in Definition 3. The following example shows what goes wrong if we would simplify the definition.

Example 12. Consider the DP problem $(\mathcal{P}, \mathcal{R})$ with $\mathcal{R}$ consisting of the rewrite rules $\mathsf{f}(x) \to \mathsf{g}(x)$ and $\mathsf{g}(\mathsf{a}(x)) \to \mathsf{f}(\mathsf{a}(x))$ and $\mathcal{P} = \text{DP}(\mathcal{R})$ consisting of $\mathsf{F}(x) \to \mathsf{G}(x)$ and $\mathsf{G}(\mathsf{a}(x)) \to \mathsf{F}(\mathsf{a}(x))$. The DP problem $(\mathcal{P}, \mathcal{R})$ is not finite because the term $\mathsf{G}(\mathsf{a}(x))$ admits a minimal rewrite sequence. If we change the definition of c in Definition 3 to

$$c = \min\{\text{height}(l'(p)) \mid p \in e(\text{base}(l'), r)\}$$

then for $s \to t = \mathsf{F}(x) \to \mathsf{G}(y)$ we have

$$\mathsf{F}_0(\mathsf{a}_0(x)) \to_{\text{match}(s\to t)} \mathsf{G}_1(\mathsf{a}_0(x)) \to_{\text{match-DP}(\mathcal{P}\setminus\{s\to t\})} \mathsf{F}_0(\mathsf{a}_0(x))$$

and it would follow that $(\mathcal{P}, \mathcal{R})$ is match-DP-bounded for $\mathsf{F}(x) \to \mathsf{G}(x)$. However, removing this rule from $\mathcal{P}$ would leave a finite DP problem and hence we would falsely conclude termination of the original TRS $\mathcal{R}$.

4 Forward Closures

When proving the termination of a TRS $\mathcal{R}$ that is non-overlapping [9] or right-linear [4] it is sufficient to restrict attention to the set RFC$(\mathcal{R})$ of right-hand sides of forward closures. This set is defined as the closure of the right-hand sides of the rules in $\mathcal{R}$ under variable renaming and narrowing. More formally, RFC$_L(\mathcal{R})$ is the least extension of L such that

– $t[r]_p\sigma \in \mathrm{RFC}_L(\mathcal{R})$ whenever $t \in \mathrm{RFC}_L(\mathcal{R})$ and there exist a position $p \in \mathcal{FP}\mathrm{os}(t)$ and a fresh variant $l \to r$ of a rewrite rule in $\mathcal{R}$ with σ a most general unifier of $t|_p$ and l,
– $t\sigma \in \mathrm{RFC}_L(\mathcal{R})$ whenever $t \in \mathrm{RFC}_L(\mathcal{R})$ and σ is a variable renaming.

Dershowitz [4] obtained the following result.

Theorem 13. *A right-linear TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}$ is terminating on* $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})$. □

In our setting we can benefit from the properties of DP problems.

Lemma 14. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem. If $\mathcal{P}$ and $\mathcal{R}$ are right-linear then $(\mathcal{P},\mathcal{R})$ is finite if and only if it is finite on* $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})$.

Proof. Easy consequence of Theorem 13 and the definition of DP problems. □

Lemma 14 can be used in connection with the DP processor of Theorem 2. For the DP processor of Theorem 10 we can do better.

Lemma 15. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and $s \to t \in \mathcal{P}$. If $\mathcal{P}$ and $\mathcal{R}$ are right-linear then $(\mathcal{P},\mathcal{R})$ admits a minimal rewrite sequence with infinitely many $\xrightarrow{\epsilon}_{s\to t}$ rewrite steps if and only if it admits such a sequence starting from a term in* $\mathrm{RFC}_{\{t\}}(\mathcal{P}\cup\mathcal{R})$. □

In general $\mathrm{RFC}_L(\mathcal{P}\cup\mathcal{R})$ is not computable. We can however over-approximate $\mathrm{RFC}_L(\mathcal{P}\cup\mathcal{R})$ by using tree automata as described in [8] and [17].

5 Raise-DP-Bounds

The reason why e-DP-bounds can be used only for DP problems $(\mathcal{P},\mathcal{R})$ consisting of left-linear TRSs $\mathcal{P}$ and $\mathcal{R}$ is that without left-linearity, rewrite sequences in $(\mathcal{P},\mathcal{R})$ cannot be lifted to sequences in e-DP$(\mathcal{P}, s \to t, \mathcal{R})$, cf. the proof of Theorem 7. As described in [17] one can solve that problem by using raise rules.

Definition 16. *Let $\mathcal{F}$ be a signature. The TRS* $\mathrm{raise}(\mathcal{F})$ *over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rules $f_i(x_1,\ldots,x_n) \to f_{i+1}(x_1,\ldots,x_n)$ with f an n-ary function symbol in $\mathcal{F}$, $i \in \mathbb{N}$, and $x_1,\ldots,x_n$ pairwise different variables. For terms $s,t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}},\mathcal{V})$ we write $s \leqslant t$ if $s \to^*_{\mathrm{raise}(\mathcal{F})} t$ and $s \uparrow t$ for the least term u with $s \leqslant u$ and $t \leqslant u$. Furthermore, this notion is extended to $\uparrow S$ for finite nonempty sets $S \subset \mathcal{T}(\mathcal{F}_{\mathbb{N}},\mathcal{V})$ in the obvious way.*

The raise rules are used below to modify the rewrite relation associated to e-DP$(\mathcal{P}, s \to t, \mathcal{R})$ in such a way that non-left-linear rules are handled properly.

Definition 17. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem over a signature $\mathcal{F}$. We define the relation $\xrightarrow{\geqslant}_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}$ on $\mathcal{T}(\mathcal{F}_{\mathbb{N}},\mathcal{V})$ as follows: $s \xrightarrow{\geqslant}_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})} t$ if and only if there exist a rewrite rule $l \to r \in e$-DP$(\mathcal{P}, s \to t, \mathcal{R})$, a position $p \in \mathcal{P}\mathrm{os}(s)$, a context C, and terms $s_1,\ldots,s_n$ such that $l = C[x_1,\ldots,x_n]$ with all variables displayed, $s|_p = C[s_1,\ldots,s_n]$, $\mathrm{base}(s_i) = \mathrm{base}(s_j)$ whenever $x_i = x_j$, and $t = s[r\theta]_p$. Here the substitution θ is defined as follows: $\theta(x) = \uparrow\{s_i \mid x_i = x\}$ if $x \in \{x_1,\ldots,x_n\}$ and $\theta(x) = x$ otherwise.*

Definition 18. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and let $s \to t \in \mathcal{P}$. We call $(\mathcal{P},\mathcal{R})$* e-raise-DP-bounded *for $s \to t$ and a set of terms L if there exists a $c \in \mathbb{N}$ such that the height of function symbols occurring in terms in $\stackrel{\geqslant}{\to}^*_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}(\mathrm{lift}_0(L))$ is at most c.*

For left-linear TRSs e-raise-DP-boundedness coincides with e-DP-boundedness. The following result is a straightforward adaption of Theorem 10.

Theorem 19. *The DP processor*

$$(\mathcal{P},\mathcal{R}) \mapsto \begin{cases} \{(\mathcal{P} \setminus \{s \to t\},\mathcal{R})\} & \text{if } \mathcal{P} \cup \mathcal{R} \text{ is top-raise-DP-bounded or} \\ & \text{non-duplicating and match-raise-DP-bounded} \\ & \text{for } s \to t \text{ and } \mathcal{T}(\mathcal{F}) \\ \{(\mathcal{P},\mathcal{R})\} & \text{otherwise} \end{cases}$$

where $\mathcal{F}$ is the signature of $\mathcal{P} \cup \mathcal{R}$, is sound and complete. □

In [17] we showed that deterministic tree automata—a common approach to handle non-linearity with automata techniques (cf. [3, 18, 19])—are unsuitable. The problem with deterministic automata is that during the construction of a compatible tree automaton $\mathcal{A}$, it can happen that $\mathcal{A}$ becomes non-deterministic. Making $\mathcal{A}$ deterministic could lead to the removal of transitions that were added in earlier stages to ensure compatibility. However as soon as we add those transitions again, they are removed since they cause $\mathcal{A}$ to be non-deterministic. In [17] we introduced *quasi-deterministic* tree automata to solve this problem. This carries over to the present setting without any problems.

6 Experiments

The techniques described in the preceding sections are implemented in T$_\text{T}$T$_2$ [21]. T$_\text{T}$T$_2$ is written in OCaml[1] and consists of about 25000 lines of code. About 20% is used to implement the match-bound technique.

An important criterion for the success of e(-raise)-DP-bounds is the choice of the rewrite rule from $\mathcal{P}$ that should be removed from the DP problem $(\mathcal{P},\mathcal{R})$ under consideration. To find a suitable rule, T$_\text{T}$T$_2$ simply starts the construction of a (quasi-)compatible tree automaton for each $s \to t \in \mathcal{P}$ in parallel. As soon as one of the processes terminates the procedure stops and returns the corresponding rule.

Below we report on the experiments we performed with T$_\text{T}$T$_2$ on the 1321 TRSs in version 4.0 of the Termination Problem Data Base.[2] All tests were performed on a workstation equipped with an Intel® Pentium™ M processor running at a CPU rate of 2 GHz and 1 GB of system memory. Our results are summarized in Table 1.

[1] `http://caml.inria.fr/`

[2] `http://www.lri.fr/~marche/tpdb`

Table 1. Summary

		no RFC				RFC			
		no `ur`		`ur`		no `ur`		`ur`	
	`sp`	`spb`	`spd`	`spb`	`spd`	`spb`	`spd`	`spb`	`spd`
# successes	497	558	583	584	608	574	584	605	613
average time	119	116	185	112	162	127	132	213	132
# timeouts	12	763	738	737	713	747	737	716	708

We list the number of successful termination attempts, the average system time needed to prove termination (measured in milliseconds), and the number of timeouts. For all experiments we used a 60 seconds time limit. Besides the recursive SCC algorithm [14] and the improved estimated dependency graph processor [12], we use the following four DP processors:

`s` the subterm criterion of [15],
`p` polynomial orderings with 0/1 coefficients [13],
`b` the DP processor of Theorem 2 (extended to non-left-linear TRSs),
`d` the DP processor of Theorem 10 for left-linear TRSs and the one of Theorem 19 for non-left-linear TRSs.

For the latter two, if the DP problem is non-duplicating we take $e = \text{match}$. For duplicating problems we take $e = \text{roof}$ for `b` and $e = \text{top}$ for `d`.

A widely used approach to increase the power of DP processors is to consider only those rewrite rules of $\mathcal{R}$ which are usable [13, 15]. Since in general usable rules (`ur` in Table 1) do not preserve the minimality of rewrite sequences for duplicating TRSs [13], it must be guaranteed that the DP processors of Theorem 2, 10 and 19 do not rely on the minimality of infinite rewrite sequences. For the DP processor of Theorem 2 this is obviously the case, since e(-raise)-bounds take all infinite rewrite sequences into account. For the DP processor of Theorem 10 with $e = \text{top}$ this follows from Lemma 6. For $e = \text{match}$ there is also no problem since $e = \text{match}$ can only be used for non-duplicating systems and it is known that usable rules can be used without restrictions for non-duplication systems ([11, Example 29] and [15, Theorem 23]).

The advantage of the DP processors of Theorems 10 and 19 over the naive one of Theorem 2 is clear, although the difference decreases when usable rules and RFC are in effect. There are two TRSs that could not be proved terminating by any tool participating in last year's termination competition[3] but which can now be handled by T$_{\text{T}}$T$_2$ due to the results of this paper: `secret05-teparla3` and `secret06-matchbox-gen-25`.

Example 20. The TRS `secret06-matchbox-gen-25` ($\mathcal{R}$ in the following) consists of the following rewrite rules:

$$\begin{aligned} \mathsf{c}(\mathsf{c}(z, x, \mathsf{a}), \mathsf{a}, y) &\to \mathsf{f}(\mathsf{f}(\mathsf{c}(y, \mathsf{a}, \mathsf{f}(\mathsf{c}(z, y, x))))) \\ \mathsf{f}(\mathsf{f}(\mathsf{c}(\mathsf{a}, y, z))) &\to \mathsf{b}(y, \mathsf{b}(z, z)) \\ \mathsf{b}(\mathsf{a}, f(\mathsf{b}(\mathsf{b}(z, y), \mathsf{a}))) &\to z \end{aligned}$$

[3] `http://www.lri.fr/~marche/termination-competition/2007`

The dependency graph contains one strongly connected component, consisting of the dependency pairs

$$1\colon \mathsf{C}(\mathsf{c}(z,x,\mathsf{a}),\mathsf{a},y) \to \mathsf{C}(y,\mathsf{a},\mathsf{f}(\mathsf{c}(z,y,x)))$$
$$2\colon \mathsf{C}(\mathsf{c}(z,x,\mathsf{a}),\mathsf{a},y) \to \mathsf{C}(z,y,x)$$

Hence termination of $\mathcal{R}$ is reduced to finiteness of the DP problem $(\{1,2\},\mathcal{R})$. This problem is top-DP-bounded for rule 1; the compatible tree automaton computed by TT_2 consists of the following transitions:

$\mathsf{a}_0 \to 1$	$\mathsf{c}_0(2,2,2) \to 4$	$\mathsf{C}_0(1,5,1) \to 3$	$\mathsf{f}_1(13) \to 1,10,14$
$\mathsf{a}_1 \to 6$	$\mathsf{c}_1(1,1,1) \to 10$	$\mathsf{C}_0(2,1,5) \to 3$	$\mathsf{f}_1(17) \to 4$
$\mathsf{b}_0(1,1) \to 1$	$\mathsf{c}_1(1,2,1) \to 14$	$\mathsf{C}_1(5,6,8) \to 3$	$\mathsf{f}_1(20) \to 21$
$\mathsf{b}_1(1,1) \to 9$	$\mathsf{c}_1(1,5,1) \to 7$	$\mathsf{f}_0(1) \to 1$	$\mathsf{f}_1(22) \to 23$
$\mathsf{b}_1(1,9) \to 1$	$\mathsf{c}_1(1,6,11) \to 12$	$\mathsf{f}_0(4) \to 5$	$\mathsf{f}_1(23) \to 12$
$\mathsf{b}_1(6,18) \to 1,10,14$	$\mathsf{c}_1(1,11,1) \to 20$	$\mathsf{f}_1(7) \to 8$	$\mathsf{f}_1(24) \to 25$
$\mathsf{b}_1(6,19) \to 4$	$\mathsf{c}_1(1,15,1) \to 24$	$\mathsf{f}_1(10) \to 11$	$\mathsf{f}_1(26) \to 27$
$\mathsf{b}_1(11,11) \to 18$	$\mathsf{c}_1(2,6,15) \to 16$	$\mathsf{f}_1(12) \to 13$	$\mathsf{f}_1(27) \to 16$
$\mathsf{b}_1(15,15) \to 19$	$\mathsf{c}_1(11,6,21) \to 22$	$\mathsf{f}_1(14) \to 15$	$1 \to 2,9$
$\mathsf{c}_0(1,1,1) \to 1$	$\mathsf{c}_1(15,6,25) \to 26$	$\mathsf{f}_1(16) \to 17$	$6 \to 1$

Hence the DP processor of Theorem 10 is applicable. This results in the new DP problem $(\{2\},\mathcal{R})$, which is proved finite by the subterm criterion with the simple projection $\pi(C) = 1$.

7 Conclusion

In this paper we showed how the match-bound technique can be incorporated into the dependency pair framework. We introduced two new enrichments which take care of the special properties of DP problems. We also showed how to strengthen the method by taking right-hand sides of forward closures into account. Experimental results demonstrated the usefulness of our approach.

An important open question is whether we can use the roof enrichment in this setting. To ensure soundness of roof(-raise)-DP-bounds, it has to be proved that no restriction of roof-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to a finite signature admits a minimal rewrite sequence with infinitely many $\xrightarrow{\epsilon}_{\mathrm{roof}(s\to t)}$ rewrite steps. We conjecture that this claim holds for arbitrary $\mathcal{P}$ and $\mathcal{R}$. We anticipate that a positive solution would make additional termination proofs possible.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *TCS*, 236:133–178, 2000.

2. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
3. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available from `www.grappa.univ-lille3.fr/tata`, 2002.
4. N. Dershowitz. Termination of linear rewriting systems (preliminary version). In *Proc. 8th ICALP*, volume 115, pages 448–458, 1981.
5. A. Geser, D. Hofbauer, and J. Waldmann. Match-bounded string rewriting systems. *AAECC*, 15(3-4):149–171, 2004.
6. A. Geser, D. Hofbauer, and J. Waldmann. Termination proofs for string rewriting systems via inverse match-bounds. *JAR*, 34(4):365–385, 2005.
7. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. Finding finite automata that certify termination of string rewriting systems. *International Journal of Foundations of Computer Science*, 16(3):471–486, 2005.
8. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. *I&C*, 205(4):512–534, 2007.
9. O. Geupel. Overlap closures and termination of term rewriting systems. Report MIP-8922, Universität Passau, 1989.
10. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. 3rd IJCAR*, volume 4130 of *LNAI*, pages 281–286, 2006.
11. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. 11th LPAR*, volume 3425 of *LNAI*, pages 301–331, 2004.
12. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. 5th FroCoS*, volume 3717 of *LNAI*, pages 216–231, 2005.
13. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *JAR*, 37(3):155–203, 2006.
14. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *I&C*, 199(1,2):172–199, 2005.
15. N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *I&C*, 205(4):474–511, 2007.
16. Jambox. Available from `http://joerg.endrullis.de/`.
17. M. Korp and A. Middeldorp. Proving termination of rewrite systems using bounds. In *Proc. 18th RTA*, volume 4533 of *LNCS*, pages 273–287, 2007.
18. A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proc. IJCAR*, volume 2083 of *LNAI*, pages 593–610, 2001.
19. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *I&C*, 178(2):499–514, 2002.
20. R. Thiemann. *The DP Framework for Proving Termination of Term Rewriting*. PhD thesis, RWTH Aachen, 2007. Available as technical report AIB-2007-17.
21. Tyrolean Termination Tool 2. Available from `http://colo6-c703.uibk.ac.at/ttt2`.
22. J. Waldmann. Matchbox: A tool for match-bounded string rewriting. In *Proc. 15th RTA*, volume 3091 of *LNCS*, pages 85–94, 2004.
23. H. Zantema. Termination of rewriting proved automatically. *JAR*, 34:105–139, 2005.