Proving Termination of Rewrite Systems using Bounds

Martin Korp and Aart Middeldorp

Institute of Computer Science
University of Innsbruck
Austria

Abstract. The use of automata techniques to prove the termination of string rewrite systems and left-linear term rewrite systems is advocated by Geser *et al.* in a recent sequence of papers. We extend their work to non-left-linear rewrite systems. The key to this extension is the introduction of so-called raise rules and the use of tree automata that are not quite deterministic. Furthermore, we present negative solutions to two open problems related to string rewrite systems.

1 Introduction

Using automata techniques is a relatively new and elegant approach for automatically proving the termination of rewrite systems. Initially proposed for string rewriting by Geser, Hofbauer, and Waldmann [7], the method has recently been extended to left-linear term rewrite systems [10]. Variations and improvements are discussed in [5, 8, 9]. The fact that the method has been implemented in several different termination provers ([4, 12, 16, 17]) is a clear witness of the success of the approach.

In this paper we look at two extremes. On the one hand, we present a negative solution to the problem whether a given string rewrite system can be proved terminating by the method if no a priori bound is given. A simple reduction from the undecidable termination problem for string rewrite systems does the trick. We further show that failure of the method is not completely characterized by the presence of a so-called witnessing set. Both results settle open problems in [7].

On the other hand, we extend the method to term rewrite systems containing rules that are not left-linear. This turns out to be surprisingly challenging. First of all, the theory on which the method is based does not work without further ado for non-left-linear rewrite systems. So-called raise rules are introduced to solve this issue. Second, the usual approach of using deterministic tree automata for dealing with non-left-linear rewrite rules appears to be incompatible with the method. We introduce quasi-deterministic tree automata to overcome this problem. Finally, the raise rules need special care to enable the automata construction to terminate.

The remainder of the paper is organized as follows. In the next section we recall basic definitions concerning the automata theory approach to proving termination of rewrite systems. In Section 3 we introduce raise rules to overcome the problem caused by non-left-linear rules. Quasi-deterministic tree automata are introduced in Section 4. In Sections 5 and 6 it is explained how these automata are used to infer termination. Like in the linear case, the power of the method is increased by considering right-hand sides of forward closures. This is explained in Section 7. We present experimental data in Section 8. In Section 9 we present our negative solutions to the open problems for string rewrite systems.

2 Preliminaries

We assume familiarity with term rewriting [1] and tree automata [2]. General knowledge of the match-bound technique [7, 10] will be helpful. Below we recall important definitions and results from the latter paper.

A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called *locally terminating* if every restriction of $\mathcal{R}$ to a finite signature $\mathcal{G} \subseteq \mathcal{F}$ is terminating. Given a set $L \subseteq \mathcal{T}(\mathcal{F})$ of ground terms, we denote the set $\{t \in \mathcal{T}(\mathcal{F}) \mid s \rightarrow_{\mathcal{R}}^* t \text{ for some } s \in L\}$ of descendants of L by $\rightarrow_{\mathcal{R}}^*(L)$. 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. Let $\mathcal{F}$ be a signature. 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 follows:

$$\mathrm{lift}_c(f) = f_c \qquad \mathrm{base}(f_i) = f \qquad \mathrm{height}(f_i) = i$$

for all $f \in \mathcal{F}$ and $c, i \in \mathbb{N}$. They are extended to terms and to set of terms in the obvious way.

Let t be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and $V \subseteq \mathcal{V}\mathsf{ar}(t)$ a set of variables. A position $p \in \mathcal{FP}\mathrm{os}(t)$ is a *roof position* in t for V if $V \subseteq \mathcal{V}\mathsf{ar}(t|_p)$. The set of all roof positions in t for V is denoted by $\mathcal{RP}\mathrm{os}_V(t)$. Let l and r be two terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The mappings top, roof, and match are defined as follows:

$$\mathrm{top}(l, r) = \{\epsilon\} \qquad \mathrm{roof}(l, r) = \mathcal{RP}\mathrm{os}_{\mathcal{V}\mathsf{ar}(r)}(l) \qquad \mathrm{match}(l, r) = \mathcal{FP}\mathrm{os}(l)$$

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 $\mathcal{FP}\mathrm{os}(l)$. 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 $\mathcal{F}_{\{0,\dots,c\}}$ is denoted by $e_c(\mathcal{R})$. 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 $\rightarrow_{e(\mathcal{R})}^*(\mathrm{lift}_0(L))$ is at most c. If we want to precise the bound c, we say that $\mathcal{R}$ is e-bounded for L *by* c. In the following we do not mention L if we have the set of all ground terms in mind.

Lemma 1 ([10]). *Let $\mathcal{R}$ be a TRS. The TRSs* $\mathrm{top}(\mathcal{R})$ *and* $\mathrm{roof}(\mathcal{R})$ *are locally terminating. If $\mathcal{R}$ is right-linear then* $\mathrm{match}(\mathcal{R})$ *is locally terminating.* □

3 Raise-Bounds

The following example shows that e-bounded TRSs need not be terminating.

Example 2. Consider the non-terminating TRS $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{a},x)\}$. The TRSs match($\mathcal{R}$), roof($\mathcal{R}$), and top($\mathcal{R}$) coincide and consist of the rules

$$\mathsf{f}_i(x,x) \to \mathsf{f}_{i+1}(\mathsf{a}_{i+1},x)$$

for all $i \geqslant 0$. It is not difficult to see that with these rules we can never reach height 2 starting from a term in $\mathcal{T}(\{\mathsf{a}_0,\mathsf{f}_0\})$. Hence $\mathcal{R}$ is e-bounded by 1 for all $e \in \{\text{top},\text{roof},\text{match}\}$.

The problem is that even though every single $\mathcal{R}$-step can be simulated by an $e(\mathcal{R})$-step, this does not hold for consecutive $\mathcal{R}$-steps. We have $\mathsf{f}(\mathsf{a},\mathsf{a}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a},\mathsf{a}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a},\mathsf{a})$ but after the step $\mathsf{f}_0(\mathsf{a}_0,\mathsf{a}_0) \to_{e(\mathcal{R})} \mathsf{f}_1(\mathsf{a}_1,\mathsf{a}_0)$ we are stuck because $\mathsf{a}_0 \neq \mathsf{a}_1$.

Definition 3. *Let $\mathcal{F}$ be a signature. The TRS* 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. The restriction of raise($\mathcal{F}$) *to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by* raise$_c$($\mathcal{F}$). *For terms $s,t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}},\mathcal{V})$ we write $s \leqslant t$ if $s \to^*_{\text{raise}(\mathcal{F})} t$ and $s \uparrow t$ for the least term u with $s \leqslant u$ and $t \leqslant u$. The latter 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 following result corresponds to Lemma 1. The right-linearity condition is weakened to non-duplication in order to cover more non-left-linear TRSs. (A TRS is duplicating if there exist a rewrite rule $l \to r$ and a variable x that occurs more often in r than in l.)

Lemma 4. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The TRSs* top($\mathcal{R}$) $\cup$ raise($\mathcal{F}$) *and* roof($\mathcal{R}$) $\cup$ raise($\mathcal{F}$) *are locally terminating. If $\mathcal{R}$ is non-duplicating then* match($\mathcal{R}$) $\cup$ raise($\mathcal{F}$) *is locally terminating.*

Proof. Straightforward adaptations of the proofs of Lemmata 16 and 17 in [10]. □

An immediate consequence of the next lemma states that every derivation in $\mathcal{R}$ can be simulated using the rules in $e(\mathcal{R})$ and raise($\mathcal{F}$).

Lemma 5. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. If $s \to_{\mathcal{R}} t$ then for all s' with* base(s') $= s$ *there exists a term t' such that* base(t') $= t$ *and $s' \to^+_{e(\mathcal{R})\cup\text{raise}(\mathcal{F})} t'$.*

Proof. Straightforward. □

However, since raise($\mathcal{F}$) is non-terminating, in order to use $e(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ to infer termination of $\mathcal{R}$, we have to restrict the rules of raise($\mathcal{F}$) to those that are really needed to simulate derivations in $\mathcal{R}$. We do this by defining a new relation $\xrightarrow{\geqslant}_{e(\mathcal{R})}$ in which the necessary raise steps are built in. The idea is that $s \xrightarrow{\geqslant}_{e(\mathcal{R})} t$ if t can be obtained from s by doing the minimum number of raise steps to ensure the applicability of a non-left-linear rewrite rule in $e(\mathcal{R})$.

Definition 6. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. We define the relation $\xrightarrow{\geqslant}_{e(\mathcal{R})}$ on $\mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$ as follows: $s \xrightarrow{\geqslant}_{e(\mathcal{R})} t$ if and only if there exist a rewrite rule $l \to r \in e(\mathcal{R})$, 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:*

$$\theta(x) = \begin{cases} \uparrow \{s_i \mid x_i = x\} & \text{if } x \in \{x_1, \ldots, x_n\} \\ x & \text{otherwise} \end{cases}$$

Note that $\xrightarrow{\geqslant}_{e(\mathcal{R})} = \rightarrow_{e(\mathcal{R})}$ for left-linear TRSs $\mathcal{R}$.

Definition 7. *The TRS $\mathcal{R}$ is called* e-raise-bounded *for L if there exists a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms in $\xrightarrow{\geqslant}{}^{*}_{e(\mathcal{R})}(\mathrm{lift}_0(L))$ is at most c.*

For left-linear TRSs, e-raise-boundedness coincides with e-boundedness.

Lemma 8. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. If $s \to_{\mathcal{R}} t$ then for all terms s' with $\mathrm{base}(s') = s$ there exists a term t' such that $\mathrm{base}(t') = t$ and $s' \xrightarrow{\geqslant}_{e(\mathcal{R})} t'$.*

Proof. Straightforward. □

Theorem 9. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and let $L \subseteq \mathcal{T}(\mathcal{F})$. If $\mathcal{R}$ is top-raise-bounded or roof-raise-bounded for L then $\mathcal{R}$ is terminating on L. If $\mathcal{R}$ is non-duplicating and match-raise-bounded for L then $\mathcal{R}$ is terminating on L.*

Proof. Assume to the contrary that there exists an infinite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ with $t_1 \in L$. With help of Lemma 8 this sequence is lifted to an infinite $\xrightarrow{\geqslant}_{e(\mathcal{R})}$ sequence starting from $\mathrm{lift}_0(t_1)$. Since $\mathcal{R}$ is e-raise-bounded for L, all terms in this latter sequence belong to $\mathcal{T}(\mathcal{F}_{\{0,\ldots,c\}})$ for some $c \in \mathbb{N}$. Hence the employed rules must come from $e_c(\mathcal{R}) \cup \mathrm{raise}_c(\mathcal{F})$ and therefore $e_c(\mathcal{R}) \cup \mathrm{raise}_c(\mathcal{F})$ is non-terminating. This is impossible because $e(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ is locally terminating according to Lemma 4. □

We conclude this section with an example.

Example 10. Consider the TRS $\mathcal{R}$ consisting of the rules $\mathsf{f}(x, x) \to \mathsf{f}(\mathsf{a}, \mathsf{g}(\mathsf{a}, x))$ and $\mathsf{g}(x, x) \to \mathsf{b}$ over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{f}, \mathsf{g}\}$. With the rules

$$\mathsf{f}_0(x, x) \to \mathsf{f}_1(\mathsf{a}_1, \mathsf{g}_1(\mathsf{a}_1, x)) \qquad \mathsf{g}_0(x, x) \to \mathsf{b}_1 \qquad \mathsf{g}_1(x, x) \to \mathsf{b}_2$$

of match($\mathcal{R}$), arbitrary derivations in $\mathcal{R}$ can be simulated using the relation $\xrightarrow{\geqslant}_{\mathrm{match}(\mathcal{R})}$. For instance,

$$\begin{aligned}\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{f}(\mathsf{a},\mathsf{b})) &\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{a},\mathsf{a})),\mathsf{f}(\mathsf{a},\mathsf{b}))\\ &\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{f}(\mathsf{a},\mathsf{b}))\\ &\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{b})))\end{aligned}$$

is turned into

$$\begin{aligned}\mathsf{f}_0(\mathsf{f}_0(\mathsf{a}_0,\mathsf{a}_0),\mathsf{f}_0(\mathsf{a}_0,\mathsf{b}_0)) &\xrightarrow{\geqslant}_{\mathrm{match}(\mathcal{R})} \mathsf{f}_0(\mathsf{f}_1(\mathsf{a}_1,\mathsf{g}_1(\mathsf{a}_1,\mathsf{a}_0)),\mathsf{f}_0(\mathsf{a}_0,\mathsf{b}_0))\\ &\xrightarrow{\geqslant}_{\mathrm{match}(\mathcal{R})} \mathsf{f}_0(\mathsf{f}_1(\mathsf{a}_1,\mathsf{b}_2),\mathsf{f}_0(\mathsf{a}_0,\mathsf{b}_0))\\ &\xrightarrow{\geqslant}_{\mathrm{match}(\mathcal{R})} \mathsf{f}_1(\mathsf{a}_1,\mathsf{g}_1(\mathsf{a}_1,\mathsf{f}_1(\mathsf{a}_1,\mathsf{b}_2)))\end{aligned}$$

Here the following raise rules are used implicitly to enable the application of the non-left-linear rules in match($\mathcal{R}$):

$$\begin{aligned}\mathsf{a}_0 &\rightarrow \mathsf{a}_1 & \mathsf{b}_1 &\rightarrow \mathsf{b}_2\\ \mathsf{b}_0 &\rightarrow \mathsf{b}_1 & \mathsf{f}_0(x,y) &\rightarrow \mathsf{f}_1(x,y)\end{aligned}$$

It can be shown that $\mathcal{R}$ is match-raise-bounded by 2.

4 Quasi-Deterministic Tree Automata

A common approach to handle non-linearity with automata techniques is to consider *deterministic* tree automata (cf. [2, 14, 15]). The weaker property defined below turns out to be more suitable for our purposes. To simplify the presentation we consider tree automata without ϵ-transitions.

Definition 11. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a tree automaton. For a left-hand side $l \in \mathrm{lhs}(\Delta)$ of a transition, we denote the set $\{q \mid l \rightarrow q \in \Delta\}$ of possible right-hand sides by $Q(l)$. We call $\mathcal{A}$* quasi-deterministic *if for every $l \in \mathrm{lhs}(\Delta)$ there exists a state $p \in Q(l)$ such that for all transitions $f(q_1, \dots, q_n) \rightarrow q \in \Delta$ and $i \in \{1, \dots, n\}$ with $q_i \in Q(l)$, the transition $f(q_1, \dots, q_{i-1}, p, q_{i+1}, \dots, q_n) \rightarrow q$ belongs to Δ. Moreover, we require that $p \in Q_f$ whenever $Q(l)$ contains a final state.*

Deterministic tree automata are trivially quasi-deterministic because $Q(l)$ is a singleton set for every left-hand side $l \in \mathrm{lhs}(\Delta)$. In general, $Q(l)$ may contain more than one state that satisfies the above property. In the following we assume that there is a unique designated state, which we denote by p_l. The set of all designated states is denoted by Q_d and the restriction of Δ to transition rules $l \rightarrow q$ that satisfy $q = p_l$ is denoted by Δ_d.

Lemma 12. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton. If $t \rightarrow_\Delta^* q$ then $t \rightarrow_{\Delta_d}^* \cdot \rightarrow_\Delta q$ for all terms $t \in \mathcal{T}(\mathcal{F})$ and states $q \in Q$.*

Proof. We use induction on t. If t is a constant the claim holds trivially. Let $t = f(t_1, \ldots, t_n)$. The sequence from t to q can be written as $t \rightarrow_{\Delta}^{*} f(q_1, \ldots, q_n) \rightarrow_{\Delta} q$. The induction hypothesis yields for every $i \in \{1, \ldots, n\}$ a left-hand side $l_i \in \text{lhs}(\Delta)$ such that $t_i \rightarrow_{\Delta_d}^{*} l_i \rightarrow_{\Delta} q_i$. Since $\mathcal{A}$ is quasi-deterministic, $l_i \rightarrow_{\Delta_d} p_{l_i}$ and $q_i \in Q(l_i)$. According to the definition of p_{l_1} the transition $f(p_{l_1}, q_2, \ldots, q_n) \rightarrow q$ belongs to Δ. Repeating this argument $n-1$ times yields that the transition $f(p_{l_1}, \ldots, p_{l_n}) \rightarrow q$ belongs to Δ. Thus $t \rightarrow_{\Delta_d}^{*} f(p_{l_1}, \ldots, p_{l_n}) \rightarrow_{\Delta} q$. □

Lemma 13. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton. The tree automaton $\mathcal{A}_d = (\mathcal{F}, Q, Q_f, \Delta_d)$ is deterministic and $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_d)$.*

Proof. From the definition it is obvious that $\mathcal{A}_d$ is deterministic. The inclusion $\mathcal{L}(\mathcal{A}_d) \subseteq \mathcal{L}(\mathcal{A})$ is trivial. In order to show the reverse inclusion, we prove the following claim for all terms $t \in \mathcal{T}(\mathcal{F})$ and states $q \in Q$:

If $t \rightarrow_{\Delta}^{*} q$ then $t \rightarrow_{\Delta_d}^{*} p_l$ and $q \in Q(l)$ for some $l \in \text{lhs}(\Delta)$.

We use induction on t. If t is a constant then $t \rightarrow q \in \Delta$. Hence $t \in \text{lhs}(\Delta)$, $q \in Q(t)$, and $t \rightarrow p_t \in \Delta_d$. Let $t = f(t_1, \ldots, t_n)$. The sequence from t to q can be written as $t \rightarrow_{\Delta}^{*} f(q_1, \ldots, q_n) \rightarrow_{\Delta} q$. From the previous lemma we know that $t \rightarrow_{\Delta_d}^{*} f(p_1, \ldots, p_n) \rightarrow_{\Delta} q$. Let $l = f(p_1, \ldots, p_n)$. We have $l \in \text{lhs}(\Delta)$, $q \in Q(l)$, and $l \rightarrow p_l \in \Delta_d$. It follows that $t \rightarrow_{\Delta_d}^{*} p_l$. This completes the proof of the claim. Now let $t \in \mathcal{L}(\mathcal{A})$. So $t \rightarrow_{\Delta}^{*} q_f$ for some $q_f \in Q_f$. From the claim we obtain $t \rightarrow_{\Delta_d}^{*} p_l$ and $q_f \in Q(l)$ for some $l \in \text{lhs}(\Delta)$. Since $Q(l)$ contains a final state, we have $p_l \in Q_f$ by definition. Hence $t \in \mathcal{L}(\mathcal{A}_d)$. □

A simple procedure to turn an arbitrary tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ into an equivalent quasi-deterministic one *without losing any transitions of* Δ is the following:

1. Use the subset construction to transform $\mathcal{A}$ into a deterministic tree automaton $\mathcal{A}' = (\mathcal{F}, Q', Q'_f, \Delta')$.
2. Take the union of $\mathcal{A}$ and $\mathcal{A}'$ after identifying states $\{q\} \in Q'$ with $q \in Q$.

Let us illustrate this on a small example.

Example 14. The tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ with $\mathcal{F} = \{\mathsf{a}, \mathsf{f}\}$, $Q = \{1, 2\}$, $Q_f = \{1\}$, and $\Delta = \{\mathsf{a} \rightarrow 1, \mathsf{a} \rightarrow 2, \mathsf{f}(1, 2) \rightarrow 1\}$ is not quasi-deterministic; we have $Q(\mathsf{a}) = \{1, 2\}$ but if we take $p_{\mathsf{a}} = 1$ then the transition $\mathsf{f}(1, 1) \rightarrow 1$ is missing and if we take $p_{\mathsf{a}} = 2$ then the transition $\mathsf{f}(2, 2) \rightarrow 1$ is missing. The subset construction produces $\mathcal{A}' = (\mathcal{F}, Q', Q'_f, \Delta')$ with $Q' = \{\{1\}, \{2\}, \{1, 2\}\}$, $Q'_f = \{\{1\}, \{1, 2\}\}$, and Δ' consisting of the following transitions:

$$\begin{array}{lrr} \mathsf{a} \rightarrow \{1,2\} & \mathsf{f}(\{1\},\{2\}) \rightarrow \{1\} & \mathsf{f}(\{1,2\},\{2\}) \rightarrow \{1\} \\ & \mathsf{f}(\{1\},\{1,2\}) \rightarrow \{1\} & \mathsf{f}(\{1,2\},\{1,2\}) \rightarrow \{1\} \end{array}$$

Combining $\mathcal{A}$ and $\mathcal{A}'$ after identifying $\{1\}$ with 1 and $\{2\}$ with 2 produces the following transitions:

$$\begin{array}{lrr} \mathsf{a} \rightarrow \{1,2\} & \mathsf{f}(1,2) \rightarrow 1 & \mathsf{f}(\{1,2\},2) \rightarrow 1 \\ \mathsf{a} \rightarrow 1 & \mathsf{f}(1,\{1,2\}) \rightarrow 1 & \mathsf{f}(\{1,2\},\{1,2\}) \rightarrow 1 \\ \mathsf{a} \rightarrow 2 & & \end{array}$$

The final states are 1 and $\{1,2\}$, and $p_{\mathsf{a}} = \{1,2\}$.

5 Compatibility

The reason why we prefer quasi-deterministic tree automata over deterministic automata is the importance of preserving existing transitions when constructing an automaton that satisfies the compatibility condition defined below. This will be illustrated in Example 17.

Definition 15. *Let $\mathcal{R}$ be a TRS, L a language, and $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ a quasi-deterministic tree automaton. We say that $\mathcal{A}$ is* compatible *with $\mathcal{R}$ and L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathcal{V}\mathsf{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta_d} q$ it holds that $r\sigma \to^*_{\Delta} q$.*

Theorem 16. *Let $\mathcal{R}$ be a TRS and L a language. Let $\mathcal{A}$ be a quasi-deterministic tree automaton. If $\mathcal{A}$ is compatible with $\mathcal{R}$ and L then $\to^*_{\mathcal{R}}(L) \subseteq \mathcal{L}(\mathcal{A})$.*

Proof. Let s and t be two ground terms such that $s \in \mathcal{L}(\mathcal{A})$ and $s \to_{\mathcal{R}} t$. We show that $t \in \mathcal{L}(\mathcal{A})$. The desired result then follows by induction. There exist a rewrite rule $l \to r \in \mathcal{R}$, a position $p \in \mathcal{P}\mathrm{os}(s)$, and a ground substitution σ such that $s = s[l\sigma]_p \to_{\mathcal{R}} s[r\sigma]_p = t$. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$. Because $s \in \mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_d)$, there exist states $q \in Q$ and $q_f \in Q_f$ such that $s = s[l\sigma]_p \to^*_{\Delta_d} s[q]_p \to^*_{\Delta_d} q_f$. Because $\mathcal{A}_d$ is deterministic by Lemma 13, different occurrences of $x\sigma$ in $l\sigma$ are reduced to the same state in the sequence from $s[l\sigma]_p$ to $s[q]_p$. Hence there exists a mapping $\tau\colon \mathcal{V}\mathsf{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta_d} l\tau \to^*_{\Delta_d} q$. We have $r\sigma \to^*_{\Delta_d} r\tau \to^*_{\Delta_d} \cdot \to_{\Delta} q$ by the definition of compatibility and Lemma 12. Hence $t = s[r\sigma]_p \to^*_{\Delta_d} \cdot \to_{\Delta} s[q]_p \to^*_{\Delta_d} q_f$ and thus $t \in \mathcal{L}(\mathcal{A})$. $\square$

Since the set $\xrightarrow{\geqslant}{}^*_{e(\mathcal{R})}(\mathrm{lift}_0(L))$ need not be regular, even for left-linear $\mathcal{R}$ and regular L [10], we cannot hope to give an exact automaton construction. The general idea [6, 10] is to look for violations of the compatibility requirement: $l\sigma \to^*_{\Delta_d} q$ and not $r\sigma \to^*_{\Delta} q$ for some rewrite rule $l \to r$, state substitution $\sigma\colon \mathcal{V}\mathsf{ar}(l) \to Q$, and state q. Then we add new states and transitions to the current automaton to ensure $r\sigma \to^*_{\Delta} q$. There are several ways to do this, ranging from establishing a completely new path $r\sigma \to^*_{\Delta} q$ to adding as few as possible new transitions by reusing transitions from the current automaton. After $r\sigma \to^*_{\Delta} q$ has been established, we look for further violations of compatibility. This process is repeated until a compatible automaton is obtained, which may never happen if new states are kept being added.

The following example explains why we prefer quasi-deterministic automata over deterministic ones.

Example 17. Consider the TRS $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{c} \to \mathsf{a}, \mathsf{c} \to \mathsf{b}\}$ over the signature $\mathcal{F} = \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{f}\}$ and the initial tree automaton $\mathcal{A} = (\mathcal{F}_{\{0\}}, \{1\}, \{1\}, \Delta)$ with the following transitions:

$$\mathsf{a}_0 \to 1 \qquad \mathsf{b}_0 \to 1 \qquad \mathsf{c}_0 \to 1 \qquad \mathsf{f}_0(1,1) \to 1$$

Suppose we look for a deterministic automaton that is compatible with match($\mathcal{R}$) and $\text{lift}_0(\mathcal{T}(\mathcal{F}))$. Note that $\mathcal{L}(\mathcal{A}) = \text{lift}_0(\mathcal{T}(\mathcal{F}))$. Since $\mathsf{c}_0 \rightarrow_{\text{match}(\mathcal{R})} \mathsf{a}_1$ and $\mathsf{c}_0 \rightarrow 1$, we add the transition $\mathsf{a}_1 \rightarrow 1$. Similarly, $\mathsf{c}_0 \rightarrow_{\text{match}(\mathcal{R})} \mathsf{b}_1$ gives rise to the transition $\mathsf{b}_1 \rightarrow 1$. Next we consider $\mathsf{f}_0(1,1) \rightarrow_{\text{match}(\mathcal{R})} \mathsf{f}_1(\mathsf{a}_1, \mathsf{b}_1)$ with $\mathsf{f}_0(1,1) \rightarrow 1$. In order to ensure $\mathsf{f}_1(\mathsf{a}_1,\mathsf{b}_1) \rightarrow^* 1$ we may reuse one or both of the transitions $\mathsf{a}_1 \rightarrow 1$ and $\mathsf{b}_1 \rightarrow 1$. Let us consider the various alternatives.

- If we reuse both transitions then we only need to add the transition $\mathsf{f}_1(1,1) \rightarrow 1$ in order to obtain $\mathsf{f}_1(\mathsf{a}_1,\mathsf{b}_1) \rightarrow^* 1$. This gives rise to a further violation of compatibility, $\mathsf{f}_1(1,1) \rightarrow_{\text{match}(\mathcal{R})} \mathsf{f}_2(\mathsf{a}_2,\mathsf{b}_2)$ with $\mathsf{f}_1(1,1) \rightarrow 1$, which is similar to the previous one.
- Suppose we reuse $\mathsf{a}_1 \rightarrow 1$ but not $\mathsf{b}_1 \rightarrow 1$. That means we have to add a new state 2 and transitions $\mathsf{b}_1 \rightarrow 2$ and $\mathsf{f}_1(1,2) \rightarrow 1$ resulting in the following transitions:

$\mathsf{a}_0 \rightarrow 1$	$\mathsf{b}_0 \rightarrow 1$	$\mathsf{c}_0 \rightarrow 1$	$\mathsf{f}_0(1,1) \rightarrow 1$
$\mathsf{a}_1 \rightarrow 1$	$\mathsf{b}_1 \rightarrow 1$	$\mathsf{b}_1 \rightarrow 2$	$\mathsf{f}_1(1,2) \rightarrow 1$

  Making these transitions deterministic produces an automaton that includes $\mathsf{c}_0 \rightarrow 1$ and $\mathsf{b}_1 \rightarrow \{1,2\}$. Because the transition $\mathsf{b}_1 \rightarrow 1$ was removed, the second violation of compatibility that we considered, $\mathsf{c}_0 \rightarrow_{\text{match}(\mathcal{R})} \mathsf{b}_1$ and $\mathsf{c}_0 \rightarrow 1$, reappears. So we have to add $\mathsf{b}_1 \rightarrow 1$ again, but each time we make the automaton deterministic this transition is deleted.
- The remaining options would be to choose a fresh state for a_1 or for both a_1 and b_1. However they all give rise to the same situation.

So by using deterministic automata we will never achieve compatibility. The problem is clearly the removal of transitions that were added in an earlier stage to ensure compatibility and that is precisely the reason why we introduced quasi-deterministic automata. Starting from the transitions in the last case above, the following quasi-deterministic tree automaton is constructed:

$\mathsf{a}_0 \rightarrow 1$	$\mathsf{b}_0 \rightarrow 1$	$\mathsf{c}_0 \rightarrow 1$	$\mathsf{f}_0(1,1) \rightarrow 1$
$\mathsf{a}_1 \rightarrow 1 \mid 2 \mid 4$	$\mathsf{b}_1 \rightarrow 1 \mid 3 \mid 5$		$\mathsf{f}_1(2,3) \rightarrow 1$
$\mathsf{f}_0(1,4) \rightarrow 1$	$\mathsf{f}_0(1,5) \rightarrow 1$	$\mathsf{f}_0(4,5) \rightarrow 1$	$\mathsf{f}_0(4,4) \rightarrow 1$
$\mathsf{f}_0(4,1) \rightarrow 1$	$\mathsf{f}_0(5,1) \rightarrow 1$	$\mathsf{f}_0(5,4) \rightarrow 1$	$\mathsf{f}_0(5,5) \rightarrow 1$
$\mathsf{f}_1(2,5) \rightarrow 1$	$\mathsf{f}_1(4,3) \rightarrow 1$	$\mathsf{f}_1(4,5) \rightarrow 1$	

Here 4 (abbreviating $\{1,2\}$) is the designated state for a_1 and 5 (abbreviating $\{1,3\}$) is the designated state for b_1. The transitions in the last three rows are added to satisfy the condition of Definition 11. The resulting automaton is compatible with match($\mathcal{R}$).

To conclude match-raise-boundedness in the previous example, it is not enough to construct a tree automaton that is compatible with match($\mathcal{R}$). We also have to ensure that the automaton is closed under the implicit raise steps caused by $\xrightarrow{\geqslant}_{\text{match}(\mathcal{R})}$. How this can be done is explained in the next section.

6 Raise-Consistency

A naive (and sound) approach to guarantee that the implicit raise rules in the definition of $\xrightarrow{\geqslant}_{e(\mathcal{R})}$ are taken into account would be to require compatibility with all raise rules $f_i(x_1, \ldots, x_n) \to f_{i+1}(x_1, \ldots, x_n)$ for which f_{i+1} appears in the current set of transitions. The following example shows that this approach may over-approximate the essential raise steps too much.

Example 18. Continuing the previous example, we have $\mathsf{f}_0(1,1) \to_{\mathrm{raise}(\mathcal{F})} \mathsf{f}_1(1,1)$ with $\mathsf{f}_0(1,1) \to 1$. Compatibility requires the addition of the transition $\mathsf{f}_1(1,1) \to 1$, causing a new compatibility violation $\mathsf{f}_1(1,1) \to_{\mathrm{match}(\mathcal{R})} \mathsf{f}_2(\mathsf{a}_2, \mathsf{b}_2)$ with $\mathsf{f}_1(1,1) \to 1$. After establishing the path $\mathsf{f}_2(\mathsf{a}_2, \mathsf{b}_2) \to^* 1$, f_2 will make its appearance and thus we have to consider $\mathsf{f}_1(1,1) \to_{\mathrm{raise}(\mathcal{F})} \mathsf{f}_2(1,1)$ with $\mathsf{f}_1(1,1) \to 1$. This yields the transition $\mathsf{f}_2(1,1) \to 1$. Clearly, this process will not terminate.

To avoid the behaviour in the previous example, we now outline a better way to handle the raise rules. Let $f_i(q_1, \ldots, q_n) \to q$ be a transition that we add to the current set Δ of transitions, either to resolve a compatibility violation or to satisfy the quasi-determinism condition. Then, for every transition $f_j(q_1, \ldots, q_n) \to p \in \Delta$ with $j < i$ we add $f_i(q_1, \ldots, q_n) \to p$ to Δ and for every transition $f_j(q_1, \ldots, q_n) \to p \in \Delta$ with $j > i$ we add $f_j(q_1, \ldots, q_n) \to q$ to Δ. The automata resulting from this implicit handling of raise rules satisfy the property defined below.

Definition 19. *Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ be a tree automaton with N a finite subset of $\mathbb{N}$. We say that $\mathcal{A}$ is* raise-consistent *if for every pair of transitions $f_i(q_1, \ldots, q_n) \to q$ and $f_j(q_1, \ldots, q_n) \to p$ in Δ with $i < j$, the transition $f_j(q_1, \ldots, q_n) \to q$ belongs to Δ.*

Lemma 20. *Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton. If $\mathcal{A}$ is raise-consistent then for all terms $s, t \in \mathcal{T}(\mathcal{F}_N)$ and states $p, q \in Q$ with* $\mathrm{base}(s) = \mathrm{base}(t)$*, $s \to_\Delta^* p$, and $t \to_\Delta^* q$ there exists a left-hand side $l \in \mathrm{lhs}(\Delta)$ such that $s \uparrow t \to_{\Delta_d}^* l$ and $p, q \in Q(l)$.*

Proof. We prove the lemma by induction on s and t. If s and t are constants then $s \uparrow t \in \{s, t\}$. If $s \leqslant t$ then $s \uparrow t = t$ and $p \in Q(t)$ by the definition of raise-consistency. If $t \leqslant s$ then $s \uparrow t = s$ and $q \in Q(s)$. So in both cases we can take $l = s \uparrow t$. For the induction step suppose that $s = f_j(s_1, \ldots, s_n)$ and $t = f_k(t_1, \ldots, t_n)$ with $s \to_\Delta^* f_j(p_1, \ldots, p_n) \to_\Delta p$ and $t \to_\Delta^* f_k(q_1, \ldots, q_n) \to_\Delta q$. The induction hypothesis yields left-hand sides $l_1, \ldots, l_n \in \mathrm{lhs}(\Delta)$ such that $s_i \uparrow t_i \to_{\Delta_d}^* l_i$ with $p_i, q_i \in Q(l_i)$ for all $i \in \{1, \ldots, n\}$. Let $m = \max\{j, k\}$. Clearly $s \uparrow t = f_m(s_1 \uparrow t_1, \ldots, s_n \uparrow t_n)$. Let $l = f_m(p_{l_1}, \ldots, p_{l_n})$. We have $s \uparrow t \to_{\Delta_d}^* f_m(l_1, \ldots, l_n) \to_{\Delta_d}^* l$. Because $\mathcal{A}$ is quasi-deterministic, $f_j(p_{l_1}, \ldots, p_{l_n}) \to p$ and $f_k(p_{l_1}, \ldots, p_{l_n}) \to q$ belong to Δ. It follows that $l \in \mathrm{lhs}(\Delta)$. Raise-consistency yields $p, q \in Q(l)$. □

Theorem 21. *Let $\mathcal{R}$ be a TRS and L a language. Let $\mathcal{A}$ be a quasi-deterministic and raise-consistent tree automaton. If $\mathcal{A}$ is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ then $\mathcal{R}$ is e-raise-bounded for L.*

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R}$ and let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$. We have $\text{lift}_0(L) \subseteq L(\mathcal{A})$. Let $s \in L(\mathcal{A})$ and $s \xrightarrow{\geqslant}_{e(\mathcal{R})} t$. Then there is a term s' such that $s \rightarrow^*_{\text{raise}(\mathcal{F})} s' \rightarrow_{e(\mathcal{R})} t$. We show that $s' \in L(\mathcal{A})$. If l is linear then $s = s'$ and we are done. Suppose l is non-linear. To simplify the notation we assume that $l = f(x, x)$. We may write $s = s[f(s_1, s_2)]_p$ and $s' = s[f(u, u)]_p$ with $\text{base}(s_1) = \text{base}(s_2)$ and $u = s_1 \uparrow s_2$. Since $s \in L(\mathcal{A})$, there exist states $p_1, p_2, q \in Q$ and $q_f \in Q_f$ such that $s \rightarrow^*_\Delta s[f(p_1, p_2)]_p \rightarrow_\Delta s[q]_p \rightarrow^*_\Delta q_f$. In order to conclude $s' \in L(\mathcal{A})$ we show that $f(u, u) \rightarrow^*_\Delta q$. The previous lemma yields a left-hand side $l \in \text{lhs}(\Delta)$ such that $u \rightarrow^*_{\Delta_d} l$ and $p_1, p_2 \in Q(l)$. We obtain $f(u, u) \rightarrow^*_{\Delta_d} f(l, l) \rightarrow^*_{\Delta_d} f(p_l, p_l)$. Quasi-determinism yields $f(p_l, p_l) \rightarrow q \in \Delta$ and thus $f(u, u) \rightarrow^*_\Delta q$ as desired. Now that $s' \in L(\mathcal{A})$ is established, we obtain $t \in L(\mathcal{A})$ from the compatibility of $\mathcal{A}$ and $e(\mathcal{R})$, as in the proof of Theorem 16. □

Example 22. Since the resulting quasi-deterministic tree automaton in Example 17 is raise-consistent and compatible with $\text{match}(\mathcal{R})$ and $\text{lift}_0(\mathcal{T}(\mathcal{F}))$, $\mathcal{R}$ is match-raise-bounded by Theorem 21.

7 Forward Closures

When proving termination of a TRS $\mathcal{R}$ that is non-overlapping [11] or right-linear [3] it is sufficient to restrict attention to the set $\text{RFC}(\mathcal{R})$ 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, $\text{RFC}(\mathcal{R})$ is the least extension of $\text{rhs}(\mathcal{R})$ such that

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

Dershowitz [3] obtained the following result.

Theorem 23. *A right-linear TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}$ is terminating on* $\text{RFC}(\mathcal{R})$. □

The following concept has been introduced in [10]. It enables the simulation of narrowing in the definition of right-hand sides of forward closures by rewriting. This makes it possible to use tree automata to compute an approximation of $\text{RFC}(\mathcal{R})$ for linear $\mathcal{R}$.

Definition 24. *Let $\mathcal{R}$ be a TRS. The TRS $\mathcal{R}_\#$ is defined as the least extension of $\mathcal{R}$ that is closed under the following operation. If $l \rightarrow r \in \mathcal{R}_\#$ and $p \in \mathcal{FP}\text{os}(l) \setminus \{\epsilon\}$ then $l[\#]_p \rightarrow r\sigma \in \mathcal{R}_\#$. Here the substitution σ is defined by $\sigma(x) = \#$ if $x \in \mathcal{V}\mathsf{ar}(l|_p)$ and $\sigma(x) = x$ otherwise. The substitution that maps all variables to $\#$ is denoted by $\sigma_\#$. Here $\#$ is a fresh function symbol.*

The following results are proved in [10].

Lemma 25. *If $\mathcal{R}$ is a linear TRS then* $\mathrm{RFC}(\mathcal{R})\sigma_{\#} = \rightarrow^*_{\mathcal{R}_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$. ☐

Corollary 26. *If a linear TRS $\mathcal{R}$ is match-bounded for $\rightarrow^*_{\mathcal{R}_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$ then $\mathcal{R}$ is terminating.* ☐

In order to obtain a corresponding result for right-linear TRSs, we linearize left-hand sides of rewrite rules.

Definition 27. *Let t be a term. The set of linear terms s with $\mathcal{V}\mathsf{ar}(t) \subseteq \mathcal{V}\mathsf{ar}(s)$ for which there exists a variable substitution $\tau\colon \mathcal{V}\mathsf{ar}(s) \setminus \mathcal{V}\mathsf{ar}(t) \to \mathcal{V}\mathsf{ar}(t)$ such that $s\tau = t$ is denoted by* $\mathrm{linear}(s)$. *Let $\mathcal{R}$ be a TRS. The set of rewrite rules $\{l' \to r \mid l \to r \in \mathcal{R}$ and $l' \in \mathrm{linear}(l)\}$ is denoted by* $\mathrm{linear}(\mathcal{R})$. *We write $\mathcal{R}'_{\#}$ for* $\mathrm{linear}(\mathcal{R})_{\#}$.

Lemma 28. *If $\mathcal{R}$ is right-linear then* $\mathrm{RFC}(\mathcal{R})\sigma_{\#} \subseteq \rightarrow^*_{\mathcal{R}'_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$.

Proof. We obviously have $\mathrm{rhs}(\mathcal{R}) = \mathrm{rhs}(\mathrm{linear}(\mathcal{R}))$. Applying Lemma 25 to $\mathrm{linear}(\mathcal{R})$ yields $\mathrm{RFC}(\mathrm{linear}(\mathcal{R}))\sigma_{\#} = \rightarrow^*_{\mathcal{R}'_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$. Hence it is sufficient to prove the inclusion $\mathrm{RFC}(\mathcal{R})\sigma_{\#} \subseteq \mathrm{RFC}(\mathrm{linear}(\mathcal{R}))\sigma_{\#}$. We omit the straightforward details. ☐

The following example shows that the reverse inclusion does not hold.

Example 29. For the TRS $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{b},\mathsf{g}(x)), \mathsf{a} \to \mathsf{b}\}$ we have $\mathrm{RFC}(\mathcal{R})\sigma_{\#} = \{\mathsf{f}(\mathsf{b},\mathsf{g}(\#)), \mathsf{b}\}$ and $\rightarrow^*_{\mathcal{R}'_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#}) = \{\mathsf{f}(\mathsf{b},\mathsf{g}^i(\#)), \mathsf{b}, \mathsf{f}(\mathsf{b},\mathsf{g}^i(\mathsf{b})) \mid i \geqslant 1\}$.

Corollary 30. *Let $\mathcal{R}$ be a right-linear TRS. If $\mathcal{R}$ is match-raise-bounded for $\rightarrow^*_{\mathcal{R}'_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$ then $\mathcal{R}$ is terminating.*

Proof. Since $\mathrm{RFC}(\mathcal{R})\sigma_{\#}$ is a subset of $\rightarrow^*_{\mathcal{R}'_{\#}}(\mathrm{rhs}(\mathcal{R})\sigma_{\#})$, $\mathcal{R}$ is also match-raise-bounded for $\mathrm{RFC}(\mathcal{R})\sigma_{\#}$. Theorem 9 yields the termination of $\mathcal{R}$ on $\mathrm{RFC}(\mathcal{R})\sigma_{\#}$. Since rewriting is closed under substitution, $\mathcal{R}$ is terminating on $\mathrm{RFC}(\mathcal{R})$. From Theorem 23 we conclude that $\mathcal{R}$ is terminating on all terms. ☐

We conclude this section by stating a simple criterion that allows us to restrict the set of terms that have to be considered for termination of TRSs that are non-duplicating but not necessarily right-linear. The easy proof is omitted. For right-linear systems the use of forward closures is more powerful.

Lemma 31. *Let $\mathcal{R}$ be a non-duplicating TRS over a signature $\mathcal{F}$ and let $\mathcal{G} \subseteq \mathcal{F}$ consist of all function symbols in* $\mathrm{rhs}(\mathcal{R})$. *Then $\mathcal{R}$ is terminating if and only if $\mathcal{R}$ is terminating on $\mathcal{T}(\mathcal{G})$.* ☐

Table 1. 87 non-left-linear TRSs

	28 right-linear				59 non-right-linear	
			RFC			
	explicit	implicit	explicit	implicit	explicit	implicit
# successes	8	9	21	21	4	8
average time	3	4	6	6	3421	549
# timeouts	20	19	7	7	55	51

8 Experimental Results

The techniques described in the preceding sections are implemented in TTTbox.[1] TTTbox is written in OCaml[2] and consists of about 5000 lines of code.

Since quasi-determinisation is expensive, TTTbox collects and resolves all compatibility violations with respect to the current automaton before making the automaton quasi-deterministic. Then new compatibility violations are determined and the process is repeated. Compatibility violations are resolved by adding new transitions according to the following strategy, which is a variation of the one used by Matchbox [16]. To establish a path $r\sigma \to_\Delta^* q$, TTTbox

1. calculates all contexts $C[\square, \dots, \square]$, $D_1[\square, \dots, \square], \dots, D_n[\square, \dots, \square]$ and terms $t_1, \dots, t_m \in \mathcal{T}(\mathcal{F}, Q)$ such that $C[D_1[t_1, \dots, t_i], \dots, D_n[t_j, \dots, t_m]] = r\sigma$, $C[q_1, \dots, q_n] \to_\Delta^* q$ and $t_i \to_\Delta^* q_{t_i}$ for states $q_1, \dots, q_n, q_{t_1}, \dots, q_{t_m} \in Q$,
2. chooses among all possibilities one where the combined size of $D_1[\square, \dots, \square]$, $\dots$, $D_n[\square, \dots, \square]$ is minimal,
3. adds new transitions involving new states to achieve $D_1[q_{t_1}, \dots, q_{t_i}] \to^* q_1$, $\dots$, $D_n[q_{t_j}, \dots, q_{t_m}] \to^* q_n$.

We report on the experiments we performed with TTTbox on the 87 non-left-linear TRSs in version 3.2 of the Termination Problem Data Base.[3] All tests were performed on a server equipped with two Intel® Xeon$^{\text{TM}}$ processors running at a CPU rate of 2.40 GHz and 2048 MB of system memory. Our results are summarized in Table 1. 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. In the left part of the table we deal with right-linear systems and test for match-raise-boundedness, both with the explicit approach for handling raise rules described in the first paragraph of Section 6 and the implicit approach using raise-consistency. The positive effect of forward closures (Corollary 30) is clearly visible. In the right part of Table 1 we deal with non-right-linear TRSs. If the TRS under consideration is non-duplicating we test for match-raise-boundedness; duplicating TRSs are tested for roof-raise-boundedness.

[1] `http://cl-informatik.uibk.ac.at/~mkorp/TTTbox.html`
[2] `http://caml.inria.fr/`
[3] `http://www.lri.fr/~marche/tpdb`

All non-left-linear TRSs in the Termination Problem Data Base that can be proved terminating with $\mathsf{T_TTbox}$ can also be proved terminating with other termination provers. Nevertheless there are non-left-linear TRSs which can currently only be handled by $\mathsf{T_TTbox}$. One such TRS consists of the following rules:

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

Using the implicit approach, $\mathsf{T_TTbox}$ certifies in 0.059 seconds that this TRS is match-raise-bounded for the set of right-hand sides of forward closures by 2. It produces a quasi-deterministic and raise-consistent tree automaton that has 47 states and 213 transitions. None of the tools that participated in last year's termination competition can prove termination within 300 seconds.

9 Two Results for Match-Bounded String Rewriting

For the class of string rewrite systems (SRSs in the following), the match-bound technique has some properties which do not hold for the more general case of (left-linear) term rewriting. One of these is regularity preservingness [7]. The following result of [7] states that match-boundedness is decidable if the bound c is fixed.

Theorem 32. *The following problem is decidable:*

instance: *an SRS $\mathcal{R}$, a regular set L, a bound $c \in \mathbb{N}$*
question: *is $\mathcal{R}$ match-bounded for L by c?*

An efficient and exact algorithm for finding a compatible automaton that solves the problem of Theorem 32 is described in Endrullis *et al.* [5] and implemented in `Jambox`. When c is not fixed, match-boundedness becomes undecidable. This settles an open problem in [7].[4]

Theorem 33. *The following problem is undecidable:*

instance: *an SRS $\mathcal{R}$ and a regular set L*
question: *is $\mathcal{R}$ match-bounded for L?*

Proof. Let t be an arbitrary string and consider $L = \{t\}$. We show that $\mathcal{R}$ is match-bounded for L if and only if t is terminating. Since the termination problem for SRSs is undecidable [13], the result follows. If $\mathcal{R}$ is match-bounded for L then t is terminating according to [7, Theorem 2]. Otherwise, for each $c \in \mathbb{N}$ there exists a string u and a derivation $\mathrm{lift}_0(t) \to^*_{\mathrm{match}(\mathcal{R})} u$ such that u contains a function symbol of height c. Because the relation $\to_{\mathrm{match}(\mathcal{R})}$ is finitely branching, it follows that there must be an infinite match($\mathcal{R}$)-derivation starting from $\mathrm{lift}_0(t)$. Erasing the heights from this derivation produces an infinite $\mathcal{R}$-derivation from t. Hence t is not terminating. □

[4] The question whether an SRS is match-bounded for the set of all strings remains open.

The properties e-boundedness and e-raise-boundedness for $e \in \{\mathrm{match}, \mathrm{roof}, \mathrm{top}\}$ are likewise undecidable. In order to prove that a given SRS $\mathcal{R}$ is *not* match-bounded for a given set L, the following concept was introduced in [7].

Definition 34. *Let $\mathcal{R}$ be an SRS and W a nonempty set that does not contain the empty string. The set of all strings t for which there exist a string $s \in W$ and strings u, t', v such that $\mathrm{lift}_0(s) \to^*_{\mathrm{match}(\mathcal{R})} ut'v$, $\mathrm{base}(t') = t$, and the height of every symbol in t' is at least 1, is denoted by* $\mathrm{raised}(\mathcal{R}, W)$*. If $W \subseteq \mathrm{raised}(\mathcal{R}, W)$ then W is called a* witnessing *set for $\mathcal{R}$.*

Lemma 35 ([7])**.** *Let $\mathcal{R}$ be an SRS such that $\epsilon \notin \mathrm{lhs}(\mathcal{R})$. If W is a witnessing set for $\mathcal{R}$ then $\mathcal{R}$ is not match-bounded for W.* ☐

In [7] it is further shown that an SRS is *looping* if and only if it admits a *finite* witnessing set. We now show that the converse of Lemma 35 does not hold. We do this by exhibiting an SRS $\mathcal{R}$ without witnessing set that is not match-bounded (for the set of all strings). This settles an open problem in [7].

Lemma 36. *The SRS $\mathcal{R} = \{\mathsf{aab} \to \mathsf{ab}, \mathsf{bc} \to \mathsf{ab}\}$ is not match-bounded for $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^*$.*

Proof. Similar to the proof of Claim 2 in [7, Example 20]. First we prove by induction on n that

$$\mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^n-1} \to^*_{\mathrm{match}(\mathcal{R})} \mathsf{a}_{n+1}\mathsf{b}_{n+1}$$

for all $n \geqslant 1$. If $n = 1$ then $\mathsf{a}_1\mathsf{b}_1\mathsf{c}_0 \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_1\mathsf{a}_1\mathsf{b}_1 \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_2\mathsf{b}_2$. If $n > 1$ then

$$\begin{aligned}\mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^n-1} = \mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^{n-1}-1}\mathsf{c}_0^{2^{n-1}} &\to^*_{\mathrm{match}(\mathcal{R})} \mathsf{a}_n\mathsf{b}_n\mathsf{c}_0^{2^{n-1}} \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_n\mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^{n-1}-1} \\ &\to^*_{\mathrm{match}(\mathcal{R})} \mathsf{a}_n\mathsf{a}_n\mathsf{b}_n \to^*_{\mathrm{match}(\mathcal{R})} \mathsf{a}_{n+1}\mathsf{b}_{n+1}\end{aligned}$$

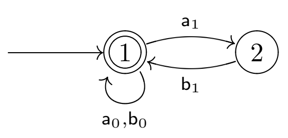
by applying the induction hypothesis twice. It follows that

$$\mathsf{a}_0\mathsf{b}_0\mathsf{c}_0^{2^n} \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_0\mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^n-1} \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_1\mathsf{b}_1\mathsf{c}_0^{2^{n-1}-1} \to^*_{\mathrm{match}(\mathcal{R})} \mathsf{a}_{n+1}\mathsf{b}_{n+1}$$

for all $n \geqslant 1$ and hence $\mathcal{R}$ is not match-bounded. ☐

Lemma 37. *The SRS $\mathcal{R} = \{\mathsf{aab} \to \mathsf{ab}, \mathsf{bc} \to \mathsf{ab}\}$ admits no witnessing set.*

Proof. Assume to the contrary that $W \subseteq \mathrm{raised}(\mathcal{R}, W)$ for some nonempty set $W \subseteq \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^+$. Since c does not appear in any right-hand side of $\mathcal{R}$ it can never reach a height greater than 0. Hence no string in W contains c and thus the rule $\mathsf{bc} \to \mathsf{ab}$ cannot be used in establishing the inclusion $W \subseteq \mathrm{raised}(\mathcal{R}, W)$. It follows that $W \subseteq \mathrm{raised}(\mathcal{R}', W)$ for the SRS $\mathcal{R}' = \{\mathsf{aab} \to \mathsf{ab}\}$. The following finite automaton certifies that $\mathcal{R}'$ is match-bounded for $\{\mathsf{a}, \mathsf{b}\}^*$ by 1:



Since $W \subseteq \{\mathsf{a}, \mathsf{b}\}^*$, $\mathcal{R}'$ is also match-bounded for W. But then $W \subseteq \mathrm{raised}(\mathcal{R}', W)$ cannot hold by Lemma 35. ☐

Acknowledgments

We thank Alfons Geser for simplifying our earlier example in Lemmata 36 and 37.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. 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.
3. N. Dershowitz. Termination of linear rewriting systems (preliminary version). In *Proc. 8th ICALP*, volume 115, pages 448–458, 1981.
4. J. Endrullis. Jambox: Automated termination proofs for string/term rewriting. Available from `http://joerg.endrullis.de/`, 2006.
5. J. Endrullis, D. Hofbauer, and J. Waldmann. Decomposing terminating rewrite relations. In *Proc. 8th WST*, pages 39–43, 2006.
6. T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 151–165, 1998.
7. A. Geser, D. Hofbauer, and J. Waldmann. Match-bounded string rewriting systems. *AAECC*, 15(3-4):149–171, 2004.
8. A. Geser, D. Hofbauer, and J. Waldmann. Termination proofs for string rewriting systems via inverse match-bounds. *JAR*, 34(4):365–385, 2005.
9. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. Finding finite automata that certify termination of string rewriting systems. *IJFCS*, 16(3):471–486, 2005.
10. 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.
11. O. Geupel. Overlap closures and termination of term rewriting systems. Report MIP-8922, Universität Passau, 1989.
12. 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.
13. G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. Rapport Laboria 283, INRIA, 1978.
14. A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proc. IJCAR*, volume 2083 of *LNAI*, pages 593–610, 2001.
15. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *I&C*, 178(2):499–514, 2002.
16. J. Waldmann. Matchbox: A tool for match-bounded string rewriting. In *Proc. 15th RTA*, volume 3091 of *LNCS*, pages 85–94, 2004.
17. H. Zantema. Termination of rewriting proved automatically. *JAR*, 34:105–139, 2005.