Match-Bounds Revisited

Martin Korp*

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, to increase the applicability of the method we show how it can be incorporated into the dependency pair framework. To achieve this we introduce two new enrichments which take the special properties of dependency pair problems into account.

Key words: term rewriting, termination, match-bounds, dependency pairs, tree automata, automation

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 [1], the method has recently been extended to left-linear term rewrite systems [2]. Variations and improvements are discussed in [3, 4, 5]. The fact that the method has been implemented in several different termination provers ([6, 7, 8, 9]) is a clear witness of the success of the approach.

The method is not only useful for proving uniform termination. Two key features of the match-bound technique are that it can be employed to prove termination of a regular subset of all terms of a term rewrite system and that it implies linear derivational complexity [2]. The former is exploited by the "right-hand sides of forward closures" transformation which allows to conclude uniform termination from termination of a modified rewrite system on a restricted set of terms. The latter makes it one of the most powerful methods that can be used to establish (linear) runtime complexity.[1]

In this paper we extend the method in two directions. The first extension is the removal of the left-linearity restriction. 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 second extension is the integration of the method into the dependency pair framework [10, 11], a powerful framework for automatically proving termination and non-termination of rewrite systems. To guarantee a successful integration we need to modularise the method in order to be able to simplify dependency

*Corresponding author

Email addresses: martin.korp@uibk.ac.at (Martin Korp), aart.middeldorp@uibk.ac.at (Aart Middeldorp)

[1] http://colo5-c703.uibk.ac.at:8080/termcomp/

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 organized as follows. In the next section we recall the basic definitions concerning the automata theory approach to proving termination of rewrite systems and we introduce raise rules to overcome the problem caused by non-left-linear rules. In Section 3 quasi-deterministic tree automata are introduced and it is explained how these automata are used to infer termination. The notion of raise-consistency is introduced in Section 4 for a proper treatment of raise rules. In Section 5 we present an alternative notion of compatibility of tree automata and rewrite systems in order to treat raise rules in a more efficient way. In Section 6 we recall the basic definitions concerning the dependency pair framework and we introduce the concept of e-DP-bounds which is based on two new enrichments that allow us to simplify dependency pair problems. Usable rules are incorporated in Section 7 and in Section 8 we increase the power of the match-bound method by considering right-hand sides of forward closures. Experimental data is presented in Section 9 and we conclude in Section 10. Some of the more technical proofs can be found in Appendices A and B.

Many of the results presented here appeared in earlier conference papers [12, 13]. New contributions include quasi-compatible tree automata in Section 5 and the incorporation of usable rules in Section 7. Furthermore, we explain in detail how e-DP-bounds can be extended to non-left-linear TRSs.

2. Proving Termination using Bounds

We assume familiarity with term rewriting [14] and tree automata [15]. Below we recall some important definitions needed in the remainder of the paper.

A signature consists of function symbols equipped with fixed arities. The set of terms constructed from a signature $\mathcal{F}$ and a set of variables $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. The set of variables in a term t is denoted by $\mathcal{V}\mathrm{ar}(t)$ and the set of function symbols of t is denoted by $\mathcal{F}\mathrm{un}(t)$. Positions are used to address symbol occurrences in terms. Given a term t and a position $p \in \mathcal{P}\mathrm{os}(t)$, we write $t(p)$ for the symbol at position p. We use $\mathcal{FP}\mathrm{os}(t)$ to denote the subset of positions $p \in \mathcal{P}\mathrm{os}(t)$ such that $t(p)$ is a function symbol. Let $\mathcal{R}$ be a finite or infinite term rewrite system (TRS for short) over a finite or infinite signature $\mathcal{F}$. The *restriction* of $\mathcal{R}$ to a finite signature $\mathcal{G} \subseteq \mathcal{F}$ is defined as $\{l \to r \in \mathcal{R} \mid l, r \in \mathcal{T}(\mathcal{G},\mathcal{V})\}$. We call $\mathcal{R}$ *locally terminating* if every restriction of $\mathcal{R}$ to a finite signature $\mathcal{G} \subseteq \mathcal{F}$ is terminating.

Example 1. *Consider the infinite TRS* $\mathcal{R} = \{\mathsf{f}_i(x) \to \mathsf{f}_{i+1}(x) \mid i \geqslant 0\}$ *over the signature* $\mathcal{F} = \{\mathsf{f}_i \mid i \geqslant 0\}$. *It is easy to see that* $\mathcal{R}$ *is both non-terminating and locally terminating.*

Let $\mathcal{R}$ be a finite TRS over a finite signature $\mathcal{F}$. Given a set $L \subseteq \mathcal{T}(\mathcal{F})$ of ground terms, we say that $\mathcal{R}$ is *terminating on* L if none of the terms in L admits an infinite rewrite sequence. 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 $\to_{\mathcal{R}}^*(L)$. For a set $N \subseteq \mathbb{N}$ of natural numbers, the signature $\mathcal{F} \times N$ is abbreviated 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

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

for all $f \in \mathcal{F}$ and $c, i \in \mathbb{N}$. The application of $\phi \in \{\mathrm{lift}_c, \mathrm{base}\}$ to a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as

$$\phi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \phi(f)(\phi(t_1), \ldots, \phi(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

These mappings are extended to sets of terms in the obvious way.

2.1. Bounds for Left-Linear TRSs

To prove termination of a TRS $\mathcal{R}$ over the signature $\mathcal{F}$ using the match-bound technique [1, 2], first an enriched system over the new signature $\mathcal{F}_{\mathbb{N}}$ is constructed that simulates the original derivations. The idea behind the new TRSs is that after a rewrite step, the minimal height of the rewritten part is greater than the minimal height of the contracted redex. Below we introduce three different enrichments.

Let t be a term in $\mathcal{T}(\mathcal{F},\mathcal{V})$ and $V \subseteq \mathcal{V}\mathrm{ar}(t)$ a set of variables. A position $p \in \mathcal{FP}\mathrm{os}(t)$ is a *roof position* in t for V if $V \subseteq \mathcal{V}\mathrm{ar}(t|_p)$. 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:

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

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 $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $e_c(\mathcal{R})$. We write $e(l \to r)$ for $e(\{l \to r\})$.

Example 2. *Consider the TRS $\mathcal{R}$ consisting of the rewrite rule $\mathsf{f}(\mathsf{g}(x,\mathsf{h}(y))) \to \mathsf{g}(\mathsf{h}(\mathsf{f}(x)),y)$. Then* $\mathrm{top}(\mathcal{R})$ *contains the rewrite rules*

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

$\mathrm{roof}(\mathcal{R})$ *contains*

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

and $\mathrm{match}(\mathcal{R})$ *contains*

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

Note that all three TRSs have infinitely many rewrite rules.

To be able to use $e(\mathcal{R})$ for proving termination of $\mathcal{R}$ it must be guaranteed that $\mathcal{R}$ is terminating whenever $e(\mathcal{R})$ is terminating. Geser *et al.* [2] obtained the following result.

Lemma 3. *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.* □

By definition, $e(\mathcal{R})$ has an infinite signature and infinitely many rewrite rules whenever $\mathcal{R} \neq \varnothing$. The idea is now to check whether there exists a finite subset of $e(\mathcal{R})$ which simulates all derivations of $\mathcal{R}$. Let $e \in \{\mathrm{top},\mathrm{roof},\mathrm{match}\}$ and L a set of terms. A 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 indicate 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.

Theorem 4 (Geser *et al.* [2]). *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 [2] it is shown that match-bounds are strictly more powerful than roof-bounds and roof-bounds are strictly more powerful than top-bounds. So in general one would prefer $\mathrm{roof}(\mathcal{R})$ to $\mathrm{top}(\mathcal{R})$, and one will use $\mathrm{match}(\mathcal{R})$ for non-duplicating TRSs. The reason for introducing $\mathrm{top}(\mathcal{R})$ is that we have to resort to it in Section 6.

We conclude this section with an example.

Example 5. *Consider the TRS $\mathcal{R}$ of Example 2 over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$. We show that $\mathcal{R}$ is not top-bounded for $\mathcal{T}(\mathcal{F})$. Consider the substitutions $\sigma = \{x \mapsto \mathsf{g}_0(x, \mathsf{h}_0(\mathsf{a}_0))\}$, $\tau = \{x \mapsto \mathsf{a}_0\}$ $\mu_i = \{x \mapsto \mathsf{g}_i(\mathsf{h}_i(x), \mathsf{a}_0)\}$, and $\nu_i = \{x \mapsto \mathsf{f}_i(\mathsf{a}_0)\}$ for all $i \geqslant 1$. We have*

$$\mathsf{f}_0(\mathsf{g}_0(x, \mathsf{h}_0(\mathsf{a}_0)))\sigma^i\tau \to^*_{\mathrm{top}(\mathcal{R})} \mathsf{g}_1(\mathsf{h}_1(x), \mathsf{a}_0)\mu_2 \cdots \mu_i\nu_i$$

for all $i \geqslant 1$. However, $\mathcal{R}$ is roof-bounded by 2 and match-bounded by 1. In Section 3 it is explained how this can be automatically checked.

2.2. Raise-Bounds for Non-Left-Linear TRSs

The first problem that arises if one wants to extend the match-bound technique to non-left-linear TRSs is that e-bounded TRSs need not be terminating.

Example 6. *Consider the non-terminating TRS $\mathcal{R} = \{\mathsf{f}(x, x) \to \mathsf{f}(\mathsf{a}, x)\}$. The TRSs* $\mathrm{match}(\mathcal{R})$, $\mathrm{roof}(\mathcal{R})$, *and* $\mathrm{top}(\mathcal{R})$ *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 $e \in \{\mathrm{top}, \mathrm{roof}, \mathrm{match}\}$.*

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 $\mathsf{a}_0 \neq \mathsf{a}_1$. To overcome this problem we introduce *raise-rules* which increase the heights of function symbols.

Definition 7. 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. The restriction of $\mathrm{raise}(\mathcal{F})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $\mathrm{raise}_c(\mathcal{F})$. For terms $s, t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$ we write $s \geqslant t$ if $t \to^*_{\mathrm{raise}(\mathcal{F})} s$ and $s \uparrow t$ for the least term u with $u \geqslant s$ and $u \geqslant t$. 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. Note that $\uparrow S$ is undefined whenever S contains two terms s and t such that $\mathrm{base}(s) \neq \mathrm{base}(t)$.

The following result corresponds to Lemma 3. 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 8. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The TRSs* $\mathrm{top}(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ *and* $\mathrm{roof}(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ *are locally terminating. If $\mathcal{R}$ is non-duplicating then* $\mathrm{match}(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ *is locally terminating.*

Proof. First we consider $e(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ with $e \in \{\mathrm{top}, \mathrm{roof}\}$. From the proof of [2, Lemma 16] we know that the rewrite rules in $e(\mathcal{R})$ are oriented from left to right by the recursive path order [16] induced by the precedence $>$ on $\mathcal{F}_{\mathbb{N}}$ defined as $f > g$ if and only if $\mathrm{height}(f) < \mathrm{height}(g)$. The same holds for the rules in $\mathrm{raise}(\mathcal{F})$. Since the precedence $>$ is well-founded on any finite subset of $\mathcal{F}_{\mathbb{N}}$, we conclude that $e(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ is locally terminating. Next we show that $\mathrm{match}(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ is locally terminating. Let $\mathcal{MF}\mathrm{un}(t)$ denote the multiset of the function symbols that occur in the term t. From the proof of [2, Lemma 17] we know that for a non-duplicating TRS $\mathcal{R}$, $\mathcal{MF}\mathrm{un}(s) >_{\mathrm{mul}} \mathcal{MF}\mathrm{un}(t)$ whenever $s \to_{\mathrm{match}(\mathcal{R})} t$. Here $>_{\mathrm{mul}}$ denotes the multiset extension of the precedence $>$ on $\mathcal{F}_{\mathbb{N}}$. If $s \to_{\mathrm{raise}(\mathcal{F})} t$ then $\mathcal{MF}\mathrm{un}(t) = (\mathcal{MF}\mathrm{un}(s) \setminus f_i) \cup \{f_{i+1}\}$ for some function symbol $f \in \mathcal{F}$ and height $i \in \mathbb{N}$, and thus $\mathcal{MF}\mathrm{un}(s) >_{\mathrm{mul}} \mathcal{MF}\mathrm{un}(t)$. Since $>_{\mathrm{mul}}$ inherits well-foundedness from $>$, we conclude that $\mathrm{match}(\mathcal{R}) \cup \mathrm{raise}(\mathcal{F})$ is locally terminating. □

Since raise($\mathcal{F}$) is non-terminating, in order to use $e(\mathcal{R}) \cup \text{raise}(\mathcal{F})$ 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 9. 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 $p \in \mathcal{P}\text{os}(s)$, a context C, and terms $s_1, \ldots, s_n$ such that $l = C[x_1, \ldots, x_n]$ with all variables displayed, $s|_p = C[s_1, \ldots, s_n]$, $\text{base}(s_i) = \text{base}(s_j)$ 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 $\xrightarrow{\geqslant}_{e(\mathcal{R})} = \to_{e(\mathcal{R})}$ for left-linear TRSs $\mathcal{R}$. The following example illustrates how implicit raise-steps are used in $\xrightarrow{\geqslant}_{e(\mathcal{R})}$ to simulate original derivations.

Example 10. *Consider the TRS $\mathcal{R}$ consisting of the rewrite 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 $\mathcal{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{g}\}$. 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 $\xrightarrow{\geqslant}_{\text{match}(\mathcal{R})}$. For instance,*

$$\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}), \mathsf{f}(\mathsf{a},\mathsf{b})) \to_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{g}(\mathsf{a},\mathsf{a})), \mathsf{f}(\mathsf{a},\mathsf{b})) \to_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{a},\mathsf{b})) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a}, \mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a},\mathsf{b})))$$

is turned into

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

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

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

Definition 11. 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 belonging to $\xrightarrow{\geqslant}{}^*_{e(\mathcal{R})}(\text{lift}_0(L))$ is at most c.

Note that e-raise-boundedness coincides with e-boundedness for left-linear TRSs. An immediate consequence of the next lemma is that every derivation in $\mathcal{R}$ can be simulated using the rewrite relation $\xrightarrow{\geqslant}_{e(\mathcal{R})}$. This result is used to infer termination from e-raise-boundedness in Theorem 13.

Lemma 12. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. If $s \to_{\mathcal{R}} t$ then for all terms s' with* $\text{base}(s') = s$ *there exists a term t' such that* $\text{base}(t') = t$ *and $s' \xrightarrow{\geqslant}_{e(\mathcal{R})} t'$.*

Proof. Straightforward. □

Theorem 13. *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 12 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 8. □

We conclude this section with an example.

Example 14. *Consider the TRS $\mathcal{R}$ over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{f}\}$ of Example 6. Using the rewrite relation $\xrightarrow{\geqslant}_{\mathrm{match}(\mathcal{R})}$ instead of $\to_{\mathrm{match}(\mathcal{R})}$ we obtain the following infinite rewrite sequence:*

$$\mathsf{f}_0(\mathsf{a}_0, \mathsf{a}_0) \xrightarrow{\geqslant}_{e(\mathcal{R})} \mathsf{f}_1(\mathsf{a}_1, \mathsf{a}_0) \xrightarrow{\geqslant}_{e(\mathcal{R})} \mathsf{f}_2(\mathsf{a}_2, \mathsf{a}_1) \xrightarrow{\geqslant}_{e(\mathcal{R})} \mathsf{f}_3(\mathsf{a}_3, \mathsf{a}_2) \xrightarrow{\geqslant}_{e(\mathcal{R})} \cdots$$

Hence $\mathcal{R}$ is not $\mathrm{match}(\mathcal{R})$*-raise-bounded for any $L \subseteq \mathcal{T}(\mathcal{F})$ that contains $\mathsf{f}(\mathsf{a}, \mathsf{a})$. Using techniques introduced in the next section, the TRS $\mathcal{R}$ of Example 10 can be (automatically) shown to be match-raise-bounded by* 2 *and hence terminating by Theorem 13.*

3. Compatible Tree Automata

In order to prove automatically that a left-linear TRS is e-bounded for some language L, Geser *et al.* [2] introduced the notion of *compatible* tree automata.

Definition 15. Let $\mathcal{R}$ be a left-linear TRS, $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ a tree automaton, and L a language. 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 $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q$ such that $l\sigma \to^*_\Delta q$ it holds that $r\sigma \to^*_\Delta q$.

Example 16. *Consider the TRS $\mathcal{R}$ over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{f}\}$ consisting of the rewrite rules*

$$\mathsf{f}(x, x) \to \mathsf{f}(\mathsf{a}, \mathsf{b}) \qquad \mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{a} \qquad \mathsf{f}(\mathsf{b}, \mathsf{b}) \to \mathsf{b}$$

and the tree automaton $\mathcal{A} = (\mathcal{F}, \{1, 2\}, \{1, 2\}, \Delta)$ with the transitions

$$\mathsf{a} \to 1 \qquad \mathsf{b} \to 2 \qquad \mathsf{f}(1, 1) \to 1 \qquad \mathsf{f}(2, 2) \to 2$$

accepting the language $L = \mathcal{T}(\{\mathsf{a}, \mathsf{f}\}) \cup \mathcal{T}(\{\mathsf{b}, \mathsf{f}\})$. Since $\mathsf{f}(x, x) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a}, \mathsf{b})$ and $\mathsf{f}(1, 1) \to 1$ but $\mathsf{f}(\mathsf{a}, \mathsf{b}) \not\to^*_\Delta 1$, $\mathcal{A}$ is not compatible with $\mathcal{R}$ and L. Adding the transitions $\mathsf{f}(1, 2) \to 1$ and $\mathsf{f}(1, 2) \to 2$ to Δ produces a tree automaton that is compatible with $\mathcal{R}$ and L.

As the above definition already indicates, any compatible tree automaton $\mathcal{A}$ is closed under left-linear rewriting.

Theorem 17 (Geser *et al.* [2]). *Let $\mathcal{R}$ be a left-linear TRS and L a language. Let $\mathcal{A}$ be a tree automaton. If $\mathcal{A}$ is compatible with $\mathcal{R}$ and L then $\to^*_{\mathcal{R}}(L) \subseteq \mathcal{L}(\mathcal{A})$.* □

So, as soon as we have constructed a tree automaton $\mathcal{A}$ that is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ for some left-linear TRS $\mathcal{R}$, we can conclude that $\to^*_{e(\mathcal{R})}(\mathrm{lift}_0(L)) \subseteq \mathcal{L}(\mathcal{A})$ and hence that $\mathcal{R}$ is e-bounded for L because $\mathcal{A}$ consists of finitely many symbols. Since the set $\to^*_{e(\mathcal{R})}(\mathrm{lift}_0(L))$ need not be regular, even for left-linear $\mathcal{R}$ and regular L [2], we cannot hope to give an exact automaton construction. The general idea [17, 2] is to look for violations of the compatibility requirement: $l\sigma \to^*_\Delta q$ and not $r\sigma \to^*_\Delta q$ for some rewrite rule $l \to r$, state substitution $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q$, 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 new transitions as possible 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.

To use Theorem 13 for proving termination it is necessary to construct a language that accepts at least all terms that are reachable from $\mathrm{lift}_0(L)$ via $\xrightarrow{\geq}_{e(\mathcal{R})}$. As before we want to do that by using compatible tree automata. However there is one problem. To cope with non-left-linear TRSs, non-deterministic tree automata cannot be used [2]. The reason is that given a non-deterministic tree automaton it is possible that terms can be only rewritten by reducing equivalent subterms to different states. A common approach to handle non-linearity with automata techniques is to consider *deterministic* tree automata (cf. [15, 18, 19]). 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 18. 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 $\{q \mid l \to q \in \Delta\}$ 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 $f(q_1, \ldots, q_n) \to q \in \Delta$ and $i \in \{1, \ldots, n\}$ with $q_i \in Q(l)$, the transition $f(q_1, \ldots, q_{i-1}, p, q_{i+1}, \ldots, q_n) \to q$ 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 for each left-hand side l there is a unique designated state in $Q(l)$, 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 $l \to q$ that satisfy $q = p_l$ is denoted by Δ_d.

Example 19. *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* $\Delta = \{\mathsf{a} \to 1, \mathsf{a} \to 2, \mathsf{f}(1, 2) \to 1\}$ *is not quasi-deterministic. This is due to the fact that for the left-hand side* a *neither* 2 *nor* 1 *can be used as designated state. If we take* $p_{\mathsf{a}} = 1$ *then we should be able to replace state* 2 *in the transition* $\mathsf{f}(1, 2) \to 1$ *by* 1, *i.e., the transition* $\mathsf{f}(1, 1) \to 1$ *should belong to* Δ. *Similarly, if we take* $p_{\mathsf{a}} = 2$ *then the transition* $\mathsf{f}(2, 2) \to 1$ *should belong to* Δ.

The key feature of a quasi-deterministic tree automaton $(\mathcal{F}, Q, Q_f, \Delta)$ is that it accepts the same language as $(\mathcal{F}, Q, Q_f, \Delta_d)$. To prove this, we need the following result.

Lemma 20. *Let* $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ *be a quasi-deterministic tree automaton. If* $t \to^*_\Delta q$ *then* $t \to^*_{\Delta_d} \cdot \to_\Delta q$ *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 $t \to^*_\Delta f(q_1, \ldots, q_n) \to_\Delta q$. The induction hypothesis yields for every $i \in \{1, \ldots, n\}$ a left-hand side $l_i \in \mathrm{lhs}(\Delta)$ such that $t_i \to^*_{\Delta_d} l_i \to_\Delta q_i$. Since $\mathcal{A}$ is quasi-deterministic, $l_i \to_{\Delta_d} p_{l_i}$, and $q_i \in Q(l_i)$. According to the definition of p_{l_1} the transition $f(p_{l_1}, q_2, \ldots, q_n) \to q$ belongs to Δ. Repeating this argument $n - 1$ times yields that the transition $f(p_{l_1}, \ldots, p_{l_n}) \to q$ belongs to Δ. Thus $t \to^*_{\Delta_d} f(p_{l_1}, \ldots, p_{l_n}) \to_\Delta q$. □

Lemma 21. *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 $t \to^*_\Delta q$ then $t \to^*_{\Delta_d} p_l$ and $q \in Q(l)$ for some $l \in \mathrm{lhs}(\Delta)$.

We use induction on t. If t is a constant then $t \to q \in \Delta$. Hence $t \in \mathrm{lhs}(\Delta)$, $q \in Q(t)$, and $t \to p_t \in \Delta_d$. Let $t = f(t_1, \ldots, t_n)$. The sequence from t to q can be written as $t \to^*_\Delta f(q_1, \ldots, q_n) \to_\Delta q$. From the previous lemma we know that $t \to^*_{\Delta_d} f(p_1, \ldots, p_n) \to_\Delta q$. Let $l = f(p_1, \ldots, p_n)$. We have $l \in \mathrm{lhs}(\Delta)$, $q \in Q(l)$, and $l \to p_l \in \Delta_d$. It follows that $t \to^*_{\Delta_d} p_l$. This completes the proof of the claim. Now let $t \in \mathcal{L}(\mathcal{A})$. So $t \to^*_\Delta q_f$ for some $q_f \in Q_f$. From the claim we obtain $t \to^*_{\Delta_d} p_l$ and $q_f \in Q(l)$ for some $l \in \mathrm{lhs}(\Delta)$. 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 22. *Consider the tree automaton $\mathcal{A}$ of Example 19. 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:*

$$\mathsf{a} \to \{1,2\} \qquad \mathsf{f}(\{1\},\{2\}) \to \{1\} \qquad \mathsf{f}(\{1,2\},\{2\}) \to \{1\}$$
$$\mathsf{f}(\{1\},\{1,2\}) \to \{1\} \qquad \mathsf{f}(\{1,2\},\{1,2\}) \to \{1\}$$

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

$$\mathsf{a} \to 1 \qquad \mathsf{a} \to \{1,2\} \qquad \mathsf{f}(1,2) \to 1 \qquad \mathsf{f}(\{1,2\},2) \to 1$$
$$\mathsf{a} \to 2 \qquad\qquad \mathsf{f}(1,\{1,2\}) \to 1 \qquad \mathsf{f}(\{1,2\},\{1,2\}) \to 1$$

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

Because we will use quasi-deterministic tree automata rather than non-deterministic tree automata to construct $\xrightarrow{\geqslant}{}^*_{e(\mathcal{R})}(\mathrm{lift}_0(L))$, we adapt the definition of compatible tree automata to make it more suitable for our purpose.

Definition 23. Let $\mathcal{R}$ be a non-left-linear TRS and L a language. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be 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 $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta_d} q$ it holds that $r\sigma \to^*_{\Delta} q$.

The reason for requiring $r\sigma \to^*_{\Delta} q$ rather than $r\sigma \to^*_{\Delta_d} q$ is that it is easier to construct a path $r\sigma \to^*_{\Delta} q$ because one can reuse more transitions.

Assume that we have constructed a quasi-deterministic tree automata $\mathcal{A}$ that is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$. To infer that $\mathcal{R}$ is e-raise-bounded for L, it must be guaranteed that $\mathcal{A}$ accepts at least $\xrightarrow{\geqslant}{}^*_{e(\mathcal{R})}(\mathrm{lift}_0(L))$. In the following we show that compatibility of $\mathcal{A}$ yields $\to^*_{e(\mathcal{R})}(\mathrm{lift}_0(L)) \subseteq \mathcal{L}(\mathcal{A})$ for any TRS $\mathcal{R}$. However that is not enough to conclude e-raise-boundedness. We also have to ensure that $\mathcal{A}$ is closed under the implicit raise steps caused by the rewrite relation $\xrightarrow{\geqslant}$. How this can be done automatically is explained in Section 4.

Theorem 24. *Let $\mathcal{R}$ be a TRS, L a language, and $\mathcal{A}$ 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 21, 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 $\tau\colon \mathcal{V}\mathrm{ar}(l) \to Q_d$ 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 20. Hence $t = s[r\sigma]_p \to^*_{\Delta_d} \cdot \to_{\Delta} s[q]_p \to^*_{\Delta_d} q_f$ and thus $t \in \mathcal{L}(\mathcal{A})$. □

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. This is illustrated in the next example.

Example 25. *Consider the TRS $\mathcal{R}$ over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{f}\}$ of Example 16 and the initial tree automaton $\mathcal{A} = (\mathcal{F}_{\{0\}}, \{1\}, \{1\}, \Delta)$ with the following transitions:*

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

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

- *If we reuse both transitions then we only need to add the transition* $\mathsf{f}_1(1,1) \to 1$ *in order to obtain* $\mathsf{f}_1(\mathsf{a}_1, \mathsf{b}_1) \to^* 1$*. This however gives rise to further violations of compatibility:* $\mathsf{f}_1(\mathsf{a}_1, \mathsf{a}_1) \to_{\mathrm{match}(\mathcal{R})} \mathsf{a}_2$ *with* $\mathsf{f}_1(\mathsf{a}_1, \mathsf{a}_1) \to^* 1$*,* $\mathsf{f}_1(\mathsf{b}_1, \mathsf{b}_1) \to_{\mathrm{match}(\mathcal{R})} \mathsf{b}_2$ *with* $\mathsf{f}_1(\mathsf{b}_1, \mathsf{b}_1) \to^* 1$ *and* $\mathsf{f}_1(x, x) \to_{\mathrm{match}(\mathcal{R})} \mathsf{f}_2(\mathsf{a}_2, \mathsf{b}_2)$ *with* $\mathsf{f}_1(1,1) \to 1$*. To solve the first two violations the transitions* $\mathsf{a}_2 \to 1$ *and* $\mathsf{b}_2 \to 1$ *have to be added. Afterwards the automaton consist of the following transitions:*

  $$\begin{array}{llll} \mathsf{a}_0 \to 1 & \mathsf{a}_2 \to 1 & \mathsf{b}_1 \to 1 & \mathsf{f}_0(1,1) \to 1 \\ \mathsf{a}_1 \to 1 & \mathsf{b}_0 \to 1 & \mathsf{b}_2 \to 1 & \mathsf{f}_1(1,1) \to 1 \end{array}$$

  It is easy to see that the new situation is similar to the one at the beginning: We have to establish $\mathsf{f}_2(\mathsf{a}_2, \mathsf{b}_2) \to^* 1$ *and may reuse one or both of the transitions* $\mathsf{a}_2 \to 1$ *and* $\mathsf{b}_2 \to 1$*.*

- *Suppose we reuse* $\mathsf{a}_1 \to 1$ *but not* $\mathsf{b}_1 \to 1$*. That means we have to add a new state* 2 *and transitions* $\mathsf{b}_1 \to 2$ *and* $\mathsf{f}_1(1,2) \to 1$ *resulting in the following transitions:*

  $$\begin{array}{llll} \mathsf{a}_0 \to 1 & \mathsf{b}_0 \to 1 & & \mathsf{f}_0(1,1) \to 1 \\ \mathsf{a}_1 \to 1 & \mathsf{b}_1 \to 1 & \mathsf{b}_1 \to 2 & \mathsf{f}_1(1,2) \to 1 \end{array}$$

  Making these transitions deterministic produces an automaton that includes $\mathsf{b}_0 \to 1$*,* $\mathsf{f}_0(1,1) \to 1$ *and* $\mathsf{b}_1 \to \{1,2\}$*. Because the transition* $\mathsf{b}_1 \to 1$ *was removed, the second violation of compatibility that we considered,* $\mathsf{f}_0(\mathsf{b}_0, \mathsf{b}_0) \to_{\mathrm{match}(\mathcal{R})} \mathsf{b}_1$ *and* $\mathsf{f}_0(\mathsf{b}_0, \mathsf{b}_0) \to 1$*, reappears. So we have to add* $\mathsf{b}_1 \to 1$ *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:

$$\begin{array}{llll} \mathsf{a}_0 \to 1 & \mathsf{b}_0 \to 1 & & \mathsf{f}_0(1,1) \to 1 \\ \mathsf{a}_1 \to 1 \mid 2 \mid 4 & \mathsf{b}_1 \to 1 \mid 3 \mid 5 & & \mathsf{f}_1(2,3) \to 1 \\ \mathsf{f}_0(1,4) \to 1 & \mathsf{f}_0(1,5) \to 1 & \mathsf{f}_0(4,1) \to 1 & \mathsf{f}_0(4,4) \to 1 \\ \mathsf{f}_0(4,5) \to 1 & \mathsf{f}_0(5,1) \to 1 & \mathsf{f}_0(5,4) \to 1 & \mathsf{f}_0(5,5) \to 1 \\ \mathsf{f}_1(2,5) \to 1 & \mathsf{f}_1(4,3) \to 1 & \mathsf{f}_1(4,5) \to 1 & \end{array}$$

The path $\mathsf{f}_1(\mathsf{a}_1, \mathsf{b}_1) \to^* 1$ *has been established by adding the new states* 2 *and* 3 *and the transitions* $\mathsf{a}_1 \to 2$*,* $\mathsf{b}_1 \to 3$*, and* $\mathsf{f}_1(2,3) \to 1$*. Furthermore* 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 18. The resulting automaton is compatible with* $\mathrm{match}(\mathcal{R})$*.*

4. Raise-Consistent Tree Automata

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 $f_i(x_1,\dots,x_n) \to f_{i+1}(x_1,\dots,x_n)$ for which f_j with $j \geqslant i+1$ appears in the current set of transitions. The following example shows that this approach may over-approximate the essential raise steps too much.

Example 26. *Let us continue the previous example. We have* $\mathsf{f}_0(x,y) \to_{\mathrm{raise}(\mathcal{F})} \mathsf{f}_1(x,y)$ *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* $\mathsf{f}_1(x,x) \to_{\mathrm{match}(\mathcal{R})} \mathsf{f}_2(\mathsf{a}_2,\mathsf{b}_2)$ *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* $\mathsf{f}_1(x,y) \to_{\mathrm{raise}(\mathcal{F})} \mathsf{f}_2(x,y)$ *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 $f_i(q_1,\dots,q_n) \to q$ 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 $f_j(q_1,\dots,q_n) \to p \in \Delta$ with $j < i$ we add $f_i(q_1,\dots,q_n) \to p$ to Δ and for every transition $f_j(q_1,\dots,q_n) \to p \in \Delta$ with $j > i$ we add $f_j(q_1,\dots,q_n) \to q$ to Δ. The automata resulting from this implicit handling of raise rules satisfy the property defined below.

Definition 27. 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 $f_i(q_1,\dots,q_n) \to q$ and $f_j(q_1,\dots,q_n) \to p$ in Δ with $i < j$, the transition $f_j(q_1,\dots,q_n) \to q$ belongs to Δ.

Let us illustrate the above definition on an example.

Example 28. *The tree automaton* $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ *with* $\mathcal{F} = \{\mathsf{a}_0, \mathsf{a}_1, \mathsf{f}_0, \mathsf{f}_2\}$, $Q = \{1,2\}$, $Q_f = \{1\}$, *and transitions*

$$\mathsf{a}_0 \to 1 \qquad \mathsf{a}_1 \to 1 \mid 2 \qquad \mathsf{f}_0(1,2) \to 1 \qquad \mathsf{f}_2(1,2) \to 2$$

is not raise-consistent because $\mathsf{f}_0(1,2) \to 1$ *but not* $\mathsf{f}_2(1,2) \to 1$. *Adding the latter transition to* Δ *makes* $\mathcal{A}$ *raise-consistent.*

In the remainder of the section we show that by constructing a quasi-deterministic and raise-consistent tree automaton $\mathcal{A}$ that is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ it is guaranteed that $\mathcal{A}$ accepts $\xrightarrow{\geqslant}{}^*_{e(\mathcal{R})}(\mathrm{lift}_0(L))$.

Lemma 29. *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 $t \geqslant s$ then $s \uparrow t = t$ and $p \in Q(t)$ by the definition of raise-consistency. If $s \geqslant t$ 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 $s = f_j(s_1,\dots,s_n)$ and $t = f_k(t_1,\dots,t_n)$ with $s \to^*_\Delta f_j(p_1,\dots,p_n) \to_\Delta p$ and $t \to^*_\Delta f_k(q_1,\dots,q_n) \to_\Delta q$. The induction hypothesis yields left-hand sides $l_1,\dots,l_n \in \mathrm{lhs}(\Delta)$ such that $s_i \uparrow t_i \to^*_{\Delta_d} l_i$ with $p_i, q_i \in Q(l_i)$ for all $i \in \{1,\dots,n\}$. Let $m = \max\{j,k\}$. Clearly $s \uparrow t = f_m(s_1 \uparrow t_1,\dots,s_n \uparrow t_n)$. Let $l = f_m(p_{l_1},\dots,p_{l_n})$. We have $l_i \to p_{l_i} \in \Delta_d$ for all $i \in \{1,\dots,n\}$ by the definition of designated state and thus $s \uparrow t \to^*_{\Delta_d} f_m(l_1,\dots,l_n) \to^*_{\Delta_d} l$. Because $\mathcal{A}$ is quasi-deterministic, $f_j(p_{l_1},\dots,p_{l_n}) \to p$ and $f_k(p_{l_1},\dots,p_{l_n}) \to q$ belong to Δ. It follows that $l \in \mathrm{lhs}(\Delta)$. Raise-consistency yields $p,q \in Q(l)$. □

Theorem 30. *Let* $\mathcal{R}$ *be a TRS and* L *a language. Let* $\mathcal{A}$ *be a raise-consistent and quasi-deterministic 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 $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ for some finite subset N of $\mathbb{N}$. We have $\mathrm{lift}_0(L) \subseteq L(\mathcal{A})$. Let $s \in L(\mathcal{A})$ and $s \xrightarrow{\geqslant}_{l\to r} t$ with $l \to r \in e(\mathcal{R})$. Then there is a term s' such that $s \to^*_{\mathrm{raise}(\mathcal{F})} s' \to_{l\to r} t$. 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 $l = f(x, x)$. Let p the position at which the rewrite rule $l \to r$ is applied. We may write $s = s[f(s_1, s_2)]_p$ and $s' = s[f(u, u)]_p$ with $\mathrm{base}(s_1) = \mathrm{base}(s_2)$ 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 $s \to^*_\Delta s[f(p_1, p_2)]_p \to_\Delta s[q]_p \to^*_\Delta q_f$. In order to conclude $s' \in L(\mathcal{A})$ we show that $f(u, u) \to^*_\Delta q$. The previous lemma yields a left-hand side $l \in \mathrm{lhs}(\Delta)$ such that $u \to^*_{\Delta_d} l$ and $p_1, p_2 \in Q(l)$. We obtain $f(u, u) \to^*_{\Delta_d} f(l, l) \to^*_{\Delta_d} f(p_l, p_l)$. Quasi-determinism yields $f(p_l, p_l) \to q \in \Delta$ and thus $f(u, u) \to^*_\Delta q$ 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 24. □

Example 31. *Since the final quasi-deterministic tree automaton in Example 25 is raise-consistent and compatible with* $\mathrm{match}(\mathcal{R})$ *and* $\mathrm{lift}_0(\mathcal{T}(\mathcal{F}))$*, $\mathcal{R}$ is match-raise-bounded by Theorem 30.*

5. Quasi-Compatible Tree Automata

By using the explicit approach for handling raise rules described in the first paragraph of Section 4 or the implicit approach using raise-consistent tree automata, it is often the case that a transition is duplicated by increasing the height of the function symbol of the left-hand side. As soon as this happens, the transition with the smaller height is useless since in each further compatibility violation the new one with the greater height can be used instead. To be able to simplify tree automata by removing such transitions we introduce the notion of *quasi-compatible* tree automata.

Definition 32. Let $\mathcal{R}$ be a TRS and L a language. Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton with N a finite subset of $\mathbb{N}$. We say that $\mathcal{A}$ is *quasi-compatible* with $\mathcal{R}$ and L if for all $t \in L$ there is a term $t' \in \mathcal{L}(\mathcal{A})$ such that $t' \geqslant t$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta_d} q$ it holds that $r'\sigma \to^*_\Delta q$ for some $r' \geqslant r$.

In the following we show that each quasi-deterministic and raise-consistent tree automaton $\mathcal{A}$ that is quasi-compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ can be transformed into a quasi-deterministic and raise-consistent tree automaton that is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$. As an immediate consequence we obtain that $\mathcal{R}$ is e-raise-bounded for L if $\mathcal{A}$ is quasi-compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$.

Definition 33. 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 *height-complete* if for all $f_i(q_1, \ldots, q_n) \to q \in \Delta$ we have $f_j(q_1, \ldots, q_n) \to q \in \Delta$ for all $0 \leqslant j < i$.

Lemma 34. *Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ be a raise-consistent and quasi-deterministic tree automaton with N a finite subset of $\mathbb{N}$. Let $\mathcal{A}' = (\mathcal{F}_{N'}, Q, Q_f, \Delta')$ be the smallest height-complete tree automaton such that $\Delta \subseteq \Delta'$. Then $\mathcal{A}'$ is quasi-deterministic and raise-consistent.*

PROOF. First we show that $\mathcal{A}'$ is quasi-deterministic. If $\mathcal{A}'$ is not quasi-deterministic then there is an $l' \in \mathrm{lhs}(\Delta')$ such that for all states $p \in Q_{\Delta'}(l')$ there is a transition $f_c(q_1, \ldots, q_n) \to q \in \Delta'$ and a $j \in \{1, \ldots, n\}$ with $q_j \in Q_{\Delta'}(l')$ and $f_c(q_1, \ldots, q_{j-1}, p, q_{j+1}, \ldots, q_n) \to q \notin \Delta'$. According to Definition 33 there is an $l \in \mathrm{lhs}(\Delta)$ such that $l \geqslant l'$. Consider the state $p_l \in Q_\Delta(l)$ whose existence is guaranteed because $\mathcal{A}$ is quasi-deterministic. As $\mathcal{A}'$ is the smallest height-complete tree-automaton such that $\Delta \subseteq \Delta'$ we have $Q_{\Delta'}(l') = Q_\Delta(l)$ and thus $p_l \in Q_{\Delta'}(l')$. By assumption there is a transition $f_c(q_1, \ldots, q_n) \to q \in \Delta'$ and a $j \in \{1, \ldots, n\}$ such that $q_j \in Q_{\Delta'}(l)$ and $f_c(q_1, \ldots, q_{j-1}, p_l, q_{j+1}, \ldots, q_n) \to q \notin \Delta'$. Because $\mathcal{A}'$ is the smallest height-complete extension of $\mathcal{A}$, there exists a transition $f_{c'}(q_1, \ldots, q_n) \to q \in \Delta$ for some $c' \geqslant c$. Since $\mathcal{A}$ is quasi-deterministic and $q_j \in Q_\Delta(l)$ we have $f_{c'}(q_1, \ldots, q_{j-1}, p_l, q_{j+1}, \ldots, q_n) \to q \in \Delta \subseteq \Delta'$. Height-completeness of $\mathcal{A}'$ yields $f_c(q_1, \ldots, q_{j-1}, p_l, q_{j+1}, \ldots, q_n) \to q \in \Delta'$, providing the desired contradiction. Hence $\mathcal{A}'$ is quasi-deterministic. Raise-consistency is an immediate consequence of Definitions 27 and 33. □

Theorem 35. *Let $\mathcal{R}$ be a TRS and L a language. Let $\mathcal{A}$ be a raise-consistent and quasi-deterministic tree automaton. If $\mathcal{A}$ is quasi-compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ then $\mathcal{R}$ is e-raise-bounded for L.*

PROOF. Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ for some finite subset N of $\mathbb{N}$. Let $\mathcal{A}' = (\mathcal{F}_{N'}, Q, Q_f, \Delta')$ be the smallest height-complete tree-automaton such that $\Delta \subseteq \Delta'$. We prove the theorem by showing that $\mathcal{A}'$ is compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$. Assume to the contrary that this does not hold. Then there is a rewrite rule $l \to r \in e(\mathcal{R})$ and a state substitution $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta'_d} q$ but not $r\sigma \to^*_{\Delta'} q$. By construction of $\mathcal{A}'$ there exists a term $l' \geqslant l$ such that $l'\sigma \to^*_{\Delta_d} q$. Let $r' \geqslant r$ such that $l' \to r' \in e(\mathcal{R})$. Since $\mathcal{A}$ is quasi-compatible with $e(\mathcal{R})$ and $\mathrm{lift}_0(L)$ there must be a term $r'' \geqslant r'$ such that $r''\sigma \to^*_{\Delta} q$ and thus also $r''\sigma \to^*_{\Delta'} q$. Let $c \in N$ such that $\mathrm{lift}_c(\mathrm{base}(r)) = r$ and let $l_1 \to p_1, \ldots, l_n \to p_n$ the transitions in Δ' which are used in the derivation $r''\sigma \to^*_{\Delta'} q$. From $r'' \geqslant r$ and the height-completeness of $\mathcal{A}'$ we infer that $\mathrm{lift}_c(\mathrm{base}(l_1)) \to p_1, \ldots, \mathrm{lift}_c(\mathrm{base}(l_n)) \to p_n \in \Delta'$. Hence $r\sigma \to^*_{\Delta'} q$, contradicting our assumption. To conclude the proof we remark that $\mathcal{A}'$ is quasi-deterministic and raise-consistent due to Lemma 34. Hence $\mathcal{R}$ is e-raise-bounded for L according to Theorem 30. □

The general idea for constructing a quasi-compatible tree automaton is quite similar to the procedure described in Section 3 for constructing a compatible tree automaton. At first we look for violations of the quasi-compatibility requirement: $l\sigma \to^*_{\Delta_d} q$ and for some rewrite rule $l \to r$, state substitution $\sigma\colon \mathcal{V}\mathrm{ar}(l) \to Q_d$, state q, but not $r'\sigma \to^*_{\Delta} q$ for any $r' \geqslant r$. After $r\sigma \to^*_{\Delta} q$ has been established by adding new states and transitions to the current automaton, we delete all transitions $f_i(q_1, \ldots, q_n) \to q$ for which there is a base-equivalent transition $f_j(q_1, \ldots, q_n) \to q$ with $j > i$. This process is repeated until a quasi-compatible tree automaton is obtained, which may never happen if new states are kept being added.

Example 36. *A quasi-deterministic and raise-consistent tree automaton that is quasi-compatible with the TRS of Example 25 has the following transitions:*

$$\begin{array}{llll}
\mathsf{a}_1 \to 1 \mid 2 \mid 4 & \mathsf{b}_1 \to 1 \mid 3 \mid 5 & & \\
\mathsf{f}_0(1,1) \to 1 & \mathsf{f}_0(1,4) \to 1 & \mathsf{f}_0(1,5) \to 1 & \mathsf{f}_0(4,1) \to 1 \\
\mathsf{f}_0(4,4) \to 1 & \mathsf{f}_0(5,1) \to 1 & \mathsf{f}_0(5,4) \to 1 & \mathsf{f}_0(5,5) \to 1 \\
\mathsf{f}_1(2,3) \to 1 & \mathsf{f}_1(2,5) \to 1 & \mathsf{f}_1(4,3) \to 1 & \mathsf{f}_1(4,5) \to 1
\end{array}$$

With respect to the quasi-deterministic, raise-consistent, and compatible tree automaton given in Example 25, the transitions $\mathsf{a}_0 \to 1$, $\mathsf{b}_0 \to 1$ *and* $\mathsf{f}_0(4,5) \to 1$ *are removed.*

Because e-raise-boundedness coincides with e-boundedness for left-linear TRSs it is obvious that quasi-compatible tree automata can be also used to verify e-bounds.

6. Combining Dependency Pairs and Bounds

The dependency pair method [20] is a powerful approach for proving termination of TRSs. The dependency pair framework [21, 11] is a modular reformulation and improvement of this approach. After presenting a simplified version of it which is sufficient for our purposes, we show how the match-bound technique can be integrated into the dependency pair framework.

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 $\mathrm{DP}(\mathcal{R})$.

Example 37. *Consider the TRS* $\mathcal{R}$ *consisting of the two rewrite rules* $\mathsf{f}(\mathsf{g}(x), y) \to \mathsf{g}(\mathsf{h}(x, y))$ *and* $\mathsf{h}(x, y) \to \mathsf{f}(x, \mathsf{g}(y))$. *The dependency pairs of* $\mathcal{R}$ *are* $\mathsf{f}^\sharp(\mathsf{g}(x), y) \to \mathsf{h}^\sharp(x, y)$ *and* $\mathsf{h}^\sharp(x, y) \to \mathsf{f}^\sharp(x, \mathsf{g}(y))$. *To ease readability, we often write* F *instead of* $\mathsf{f}^\sharp$, *etc.*

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. We say that $(\mathcal{P}, \mathcal{R})$ is finite *on* a language $L \subseteq \mathcal{T}(\mathcal{F}^\sharp)$ if there is no minimal rewrite sequence starting at a term $s \in L$. 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.

To simplify the presentation we first consider left-linear TRSs. The extension to non-left-linear TRSs is discussed in Section 6.2. Finally in Section 6.3 it is explained how (quasi-deterministic and raise-consistent) tree automata can be used to infer finiteness of DP problems.

6.1. DP-Bounds for Left-Linear DP Problems

The general procedure for proving finiteness of a DP problem $(\mathcal{P}, \mathcal{R})$ tries to remove 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 38. *The DP processor*

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

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

Proof. Assume that $\mathcal{P} \cup \mathcal{R}$ is e-bounded for $\mathcal{T}(\mathcal{F})$. By Theorem 4 we conclude that $\mathcal{P} \cup \mathcal{R}$ is terminating. Because $\mathcal{P} \cup \mathcal{R}$ does not admit an infinite rewrite sequence we know that $(\mathcal{P}, \mathcal{R})$ does not admit a minimal rewrite sequence. Hence $(\mathcal{P}, \mathcal{R})$ is finite. □

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 38 is applicable only at the leaves of the DP search tree, which means that it cannot be used to (partly) simplify a DP problem. 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-$\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ and match-$\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ to achieve this. The basic idea behind these TRSs is that every infinite sequence of $(\mathcal{P}, \mathcal{R})$ in which $s \to t$, the rule that is to be removed from $\mathcal{P}$, is used infinitely often is simulated by a height increasing infinite sequence.

Definition 39. Let $\mathcal{S}$ be a TRS over a signature $\mathcal{F}$. The TRS e-$\mathrm{DP}(\mathcal{S})$ 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-$\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ is defined as the union of e-$\mathrm{DP}((\mathcal{P} \setminus \{s \to t\}) \cup \mathcal{R})$ and $e(s \to t)$. The restriction of e-$\mathrm{DP}(\mathcal{S})$ and e-$\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by e-$\mathrm{DP}_c(\mathcal{S})$ and e-$\mathrm{DP}_c(\mathcal{P}, s \to t, \mathcal{R})$.

Example 40. *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*

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

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

$$\mathsf{H}_0(x,y)\to\mathsf{F}_0(x,\mathsf{g}_0(y))\qquad \mathsf{H}_1(x,y)\to\mathsf{F}_1(x,\mathsf{g}_1(y))\qquad \mathsf{H}_2(x,y)\to\mathsf{F}_2(x,\mathsf{g}_2(y))\qquad\cdots$$

and match$(s\to t)$ *contains*

$$\mathsf{F}_0(\mathsf{g}_0(x),y)\to\mathsf{H}_1(x,y)\qquad \mathsf{F}_1(\mathsf{g}_0(x),y)\to\mathsf{H}_1(x,y)\qquad \mathsf{F}_0(\mathsf{g}_1(x),y)\to\mathsf{H}_1(x,y)\qquad\cdots$$

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*

$$\mathsf{F}_0(\mathsf{g}_0(x),y)\to\mathsf{H}_0(x,y)\qquad \mathsf{F}_1(\mathsf{g}_0(x),y)\to\mathsf{H}_1(x,y)\qquad \mathsf{F}_0(\mathsf{g}_1(x),y)\to\mathsf{H}_0(x,y)\qquad\cdots$$

we obtain the TRS match-DP$(\mathcal{P}\cup\mathcal{R})$*. Note that all TRSs have infinitely many rewrite rules.*

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 41. 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 number $c\in\mathbb{N}$ such that the height of function symbols occurring in terms in $\to^*_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}(\mathrm{lift}_0(L))$ is at most c.

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 39. The following example shows what goes wrong if we would simplify the definition.

Example 42. *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* $\mathcal{P}=\mathrm{DP}(\mathcal{R})$ *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 39 to* $c=\min\{\mathrm{height}(l'(p))\mid p\in e(\mathrm{base}(l'),r)\}$ *then for* $s\to t=\mathsf{F}(x)\to\mathsf{G}(y)$ *we have*

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

and it would follow that $(\mathcal{P},\mathcal{R})$ *is* match*-DP-bounded for* $\mathsf{F}(x)\to\mathsf{G}(x)$*. As we will see later, this would imply that we can remove* $\mathsf{F}(x)\to\mathsf{G}(x)$ *from* $\mathcal{P}$*. Because the remaining DP problem is finite we would falsely conclude termination of the original TRS* $\mathcal{R}$*.*

An immediate consequence of the next lemma is that every derivation according to the DP problem $(\mathcal{P},\mathcal{R})$ can be simulated using the rules in e-DP$(\mathcal{P},s\to t,\mathcal{R})$.

Lemma 43. *Let* $(\mathcal{P},\mathcal{R})$ *be a left-linear DP problem and* $s\to t\in\mathcal{P}$*. If* $u\to_{s\to t}v$ *or* $u\to_{(\mathcal{P}\setminus\{s\to t\})\cup\mathcal{R}}v$ *then for all terms* u' *with* $\mathrm{base}(u')=u$ *there exists a term* v' *such that* $\mathrm{base}(v')=v$ *and* $u'\to_{e(s\to t)}v'$ *or* $u'\to_{e\text{-DP}((\mathcal{P}\setminus\{s\to t\})\cup\mathcal{R})}v'$*.*

PROOF. Straightforward. □

To be able to use the concept of e-DP-boundedness to simplify DP problems, we need to ensure that no restriction of $e\text{-DP}(\mathcal{P}, s \to t, \mathcal{R})$ to a finite signature admits 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 use $e\text{-DP}(\mathcal{P} \cup \mathcal{R})$ instead of $e\text{-DP}(\mathcal{P}, s \to t, \mathcal{R})$ then this property does not hold because every rewrite sequence in $\mathcal{P} \cup \mathcal{R}$ can be simulated by an $e\text{-DP}_0(\mathcal{P} \cup \mathcal{R})$-sequence.

Lemma 44. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem, let $s \to t \in \mathcal{P}$, and let $c \geqslant 0$. The TRS $\text{top-DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ 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 $\text{top-DP}(\mathcal{P} \setminus \{s \to t\})$ and $\text{top}(s \to t)$ are applied at root positions. Every rewrite rule $l \to r$ in $\text{top-DP}(\mathcal{P} \setminus \{s \to t\})$ has the property that $\text{height}(l(\epsilon)) = \text{height}(r(\epsilon))$. Hence $\text{height}(t_i(\epsilon)) = \text{height}(s_{i+1}(\epsilon))$ for all $i \geqslant 1$. By definition, for every $l \to r \in \text{top}(s \to t)$ we have $\text{height}(r(\epsilon)) = \text{height}(l(\epsilon)) + 1$ and thus $\text{height}(t_i(\epsilon)) = \text{height}(s_i(\epsilon)) + 1$ for all $i \geqslant 1$. It follows that $\text{height}(t_{c+1}(\epsilon)) \geqslant c + 1$, contradicting the assumption. □

Theorem 45. *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 on L if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite on 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 on L. If $(\mathcal{P}, \mathcal{R})$ is not finite on 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 $\text{top-DP}(\mathcal{P}, s \to t, \mathcal{R})$ rewrite sequence starting from $\text{lift}_0(s_1)$. Since the original sequence contains infinitely many $\xrightarrow{\epsilon}_{s\to t}$ rewrite steps the lifted sequence contains infinitely many $\xrightarrow{\epsilon}_{\text{top}(s\to t)}$ 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 $\text{top-DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ and therefore $\text{top-DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ contains a minimal rewrite sequence consisting of infinitely many $\xrightarrow{\epsilon}_{\text{top}(s\to t)}$ rewrite steps. This however is excluded by Lemma 44. □

If we restrict Lemma 44 to *minimal* rewrite sequences, it also holds for $e = \text{match}$ provided $\mathcal{R}$ and $\mathcal{P}$ are non-duplicating. The proof is considerably more complicated and given in Appendix A.

Lemma 46. *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 $\text{match-DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ does not admit minimal rewrite sequences with infinitely many $\xrightarrow{\epsilon}_{\text{match}(s\to t)}$ rewrite steps.*

Theorem 47. *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 on L if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite on L.*

PROOF. Similarly to the proof of Theorem 45, using Lemma 46 instead of Lemma 44. Note that in the presence of left-linearity, the non-duplicating requirement in Lemma 46 is equivalent to linearity. □

We conjecture that Lemma 44 also holds for $e = \text{roof}$. A positive solution is important as roof-bounds are strictly more powerful than top-bounds (see Section 9 and [2]).

Corollary 48. *The DP processor*

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

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

Proof. Immediate consequence of Theorems 45 and 47. □

6.2. Raise-DP-Bounds for Non-Left-Linear DP Problems

In order to apply the DP processor of Theorem 38 to non-left-linear TRSs, we use e-raise-bounds instead of e-bounds.

Theorem 49. *The DP processor*

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

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

Proof. Similar to the proof of Theorem 38 by using Theorem 13 instead of Theorem 4. □

Similar as in the case of e-bounds, e-DP-bounds can be used only for DP problems $(\mathcal{P},\mathcal{R})$ consisting of left-linear TRSs $\mathcal{P}$ and $\mathcal{R}$. The reason is that without left-linearity, rewrite sequences in $(\mathcal{P},\mathcal{R})$ cannot be lifted to sequences in $e\text{-DP}(\mathcal{P}, s \to t, \mathcal{R})$, cf. Lemma 43. As described in Section 2 one can solve that problem by considering the relation[2] $\xrightarrow{\geqslant}_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}$ which uses raise rules to deal with non-left-linear rewrite rules.

Definition 50. 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 $\xrightarrow{\geqslant}{}^*_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}(\mathrm{lift}_0(L))$ is at most c.

Note that for left-linear DP problems, e-raise-DP-boundedness coincides with e-DP-boundedness. An immediate consequence of the next lemma is that every derivation according to the DP problem $(\mathcal{P},\mathcal{R})$ can be simulated using the rewrite relation $\xrightarrow{\geqslant}_{e\text{-DP}(\mathcal{P},s\to t,\mathcal{R})}$.

Lemma 51. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and $s \to t \in \mathcal{P}$. If $u \to_{s\to t} v$ or $u \to_{(\mathcal{P}\setminus\{s\to t\})\cup\mathcal{R}} v$ then for all terms u' with $\mathrm{base}(u') = u$ there exists a term v' such that $\mathrm{base}(v') = v$ and $u' \xrightarrow{\geqslant}_{e(s\to t)} v'$ or $u' \xrightarrow{\geqslant}_{e\text{-DP}((\mathcal{P}\setminus\{s\to t\})\cup\mathcal{R})} v'$.*

Proof. Straightforward. □

The following two results correspond to Lemmata 44 and 46.

Lemma 52. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem, let $s \to t \in \mathcal{P}$, and let $c \geqslant 0$. The TRS $\text{top-DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ does not admit $\xrightarrow{\geqslant}$ rewrite sequences with infinitely many $\xrightarrow{\geqslant}_{\mathrm{top}(s\to t)}$ root-rewrite steps.*

Proof. Similar to the proof of Lemma 44, using $\xrightarrow{\geqslant}_{\text{top-DP}(\mathcal{P}\setminus\{s\to t\}\cup\mathcal{R})}$ instead of $\to_{\text{top-DP}(\mathcal{P}\setminus\{s\to t\}\cup\mathcal{R})}$ and $\xrightarrow{\geqslant}_{\mathrm{top}(s\to t)}$ instead of $\xrightarrow{\epsilon}_{\mathrm{top}(s\to t)}$. □

[2] This relation is obtained by replacing $e(\mathcal{R})$ with $e\text{-DP}(\mathcal{P}, s \to t, \mathcal{R})$ in Definition 9.

Lemma 53. *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-$\mathrm{DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ *does not admit minimal $\xrightarrow{\geqslant}$ rewrite sequences with infinitely many $\xrightarrow{\geqslant}_{\mathrm{match}(s\to t)}$ root-rewrite steps.*

Proof. Straightforward adaption of the proof of Lemma 46 given in Appendix A. □

Theorem 54. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem, $s \to t \in \mathcal{P}$, and L a set of terms. If $(\mathcal{P},\mathcal{R})$ is top-raise-DP-bounded for $s \to t$ and L then $(\mathcal{P},\mathcal{R})$ is finite on L if and only if $(\mathcal{P} \setminus \{s \to t\},\mathcal{R})$ is finite on L. If $\mathcal{P}$ and $\mathcal{R}$ are non-duplicating and $(\mathcal{P},\mathcal{R})$ is match-raise-DP-bounded for $s \to t$ and L then $(\mathcal{P},\mathcal{R})$ is finite on L if and only if $(\mathcal{P} \setminus \{s \to t\},\mathcal{R})$ is finite on 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 on L. If $(\mathcal{P},\mathcal{R})$ is not finite on 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 $s_1 \in L$. By Lemma 51, this rewrite sequence can be lifted to an infinite $\xrightarrow{\geqslant}_{e\text{-}\mathrm{DP}(\mathcal{P},s\to t,\mathcal{R})}$ 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 $\xrightarrow{\geqslant}_{e(s\to t)}$ root-rewrite steps. Moreover, because $(\mathcal{P},\mathcal{R})$ is e-raise-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 $e\text{-}\mathrm{DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ and therefore $e\text{-}\mathrm{DP}_c(\mathcal{P}, s \to t, \mathcal{R})$ contains a minimal $\xrightarrow{\geqslant}$ rewrite sequence consisting of infinitely many $\xrightarrow{\geqslant}_{e(s\to t)}$ root-rewrite steps. This however is excluded by Lemmata 52 and 53. □

Corollary 55. *The DP processor*

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

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

Proof. Immediate consequence of Theorem 54. □

6.3. Compatible Tree Automata

In order to prove automatically that a DP problem is e(-raise)-DP-bounded for some language L we use compatible (quasi-deterministic) tree automata as defined in Section 3.

Lemma 56. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem such that $\mathcal{P}$ and $\mathcal{R}$ are left-linear and let $s \to t \in \mathcal{P}$. Let $\mathcal{A}$ be a tree automaton. If $\mathcal{A}$ is compatible with $e\text{-}\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ and $\mathrm{lift}_0(L)$ then $(\mathcal{P},\mathcal{R})$ is e-DP-bounded for $s \to t$ and L.*

Proof. Easy consequence of Theorem 17. □

Lemma 57. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem, $s \to t \in \mathcal{P}$ and L a language. Let $\mathcal{A}$ be a quasi-deterministic and raise-consistent tree automaton. If $\mathcal{A}$ is compatible with $e\text{-}\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$ and $\mathrm{lift}_0(L)$ then $(\mathcal{P},\mathcal{R})$ is e-raise-DP-bounded for $s \to t$ and L.*

Proof. Similar as the proof of Theorem 30 if we take $\mathcal{F}$ to be the signature of $\mathcal{P} \cup \mathcal{R}$ and replace $e(\mathcal{R})$ by $e\text{-}\mathrm{DP}(\mathcal{P}, s \to t, \mathcal{R})$. □

Example 58. *We show that the DP problem $(\mathcal{P},\mathcal{R})$ of Example 40 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 Corollary 48 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 38 fails on $(\mathcal{P},\mathcal{R})$.*

Similar as for e-raise-bounds we can optimize the completion procedure by constructing a quasi-deterministic and raise-consistent tree-automaton that is *quasi-compatible* with e-DP$(\mathcal{P},s\to t,\mathcal{R})$ and $\text{lift}_0(L)$.

Theorem 59. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and L a language. Let $\mathcal{A}$ be a quasi-deterministic and raise-consistent tree automaton. If $\mathcal{A}$ is quasi-compatible with e-DP$(\mathcal{P},s\to t,\mathcal{R})$ and $\text{lift}_0(L)$ then $(\mathcal{P},\mathcal{R})$ is e-raise-DP-bounded for $s\to t$ and L.*

Proof. Similar to the proof of Theorem 35; just replace $e(\mathcal{R})$ by e-DP$(\mathcal{P},s\to t,\mathcal{R})$. □

7. Usable Rules

A widely used approach to increase the power of DP processors is to consider only those rewrite rules of $\mathcal{R}$ which are *usable* [20, 21, 22, 23]. Let $\mathcal{R}$ be a TRS and t be a term. The function $\text{tcap}(\mathcal{R},t)$ [21] is defined as $\text{tcap}(\mathcal{R},t)=f(\text{tcap}(\mathcal{R},t_1),\ldots,\text{tcap}(\mathcal{R},t_n))$ if $t=f(t_1,\ldots,t_n)$ and $f(\text{tcap}(\mathcal{R},t_1),\ldots,\text{tcap}(\mathcal{R},t_n))$ does not unify with any $l\in\text{lhs}(\mathcal{R})$. Otherwise $\text{tcap}(\mathcal{R},t)=x$ for some fresh variable x. For a DP problem $(\mathcal{P},\mathcal{R})$, the set of *usable rules* is defined as

$$\mathcal{U}(\mathcal{P},\mathcal{R})=\bigcup_{s\to t\in\mathcal{P}}\mathcal{U}(t)$$

where $\mathcal{U}(t)\subseteq\mathcal{R}$ denotes the smallest set of rules such that

- $\mathcal{U}(r)\subseteq\mathcal{U}(t)$ if $l\to r\in\mathcal{U}(t)$,
- $\mathcal{U}(u)\subseteq\mathcal{U}(t)$ if u is a subterm of t, and
- $l\to r\in\mathcal{U}(t)$ if $t=f(t_1,\ldots,t_n)$ and $f(\text{tcap}(t_1),\ldots,\text{tcap}(t_n))$ unifies with a $l\in\text{lhs}(\mathcal{R})$.

Furthermore, in the case that $\mathcal{P}$ or $\mathcal{R}$ is duplicating we have $c(x,y)\to x\in\mathcal{U}(\mathcal{P},\mathcal{R})$ and $c(x,y)\to y\in\mathcal{U}(\mathcal{P},\mathcal{R})$ for some fresh function symbol c. The two projection rules ensure that $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ admits an infinite rewrite sequence whenever $(\mathcal{P},\mathcal{R})$ is not finite.

Let us illustrate the above definitions on a small example.

Example 60. *Consider the DP problem $(\mathcal{P},\mathcal{R})$ with $\mathcal{R}$ consisting of the rewrite rules*

$$\mathsf{p}(\mathsf{s}(x)) \to x \qquad \mathsf{fac}(\mathsf{s}(x)) \to \mathsf{s}(x) \times \mathsf{fac}(\mathsf{p}(\mathsf{s}(x))) \qquad \mathsf{fac}(0) \to 0$$

and $\mathcal{P}$ consisting of the two dependency pairs $\mathsf{Fac}(\mathsf{s}(x)) \to \mathsf{Fac}(\mathsf{p}(\mathsf{s}(x)))$ and $\mathsf{Fac}(\mathsf{s}(x)) \to \mathsf{P}(\mathsf{s}(x))$ of $\mathcal{R}$. We have $\mathrm{tcap}(\mathcal{R},x) = y$, $\mathrm{tcap}(\mathcal{R},\mathsf{s}(x)) = \mathsf{s}(y)$, and $\mathrm{tcap}(\mathcal{R},\mathsf{p}(\mathsf{s}(x))) = y$ as well as $\mathcal{U}(\mathsf{Fac}(\mathsf{p}(\mathsf{s}(x)))) = \{\mathsf{p}(\mathsf{s}(x)) \to x\}$ and $\mathcal{U}(\mathsf{P}(\mathsf{s}(x))) = \varnothing$. Hence $\mathcal{U}(\mathcal{P},\mathcal{R}) = \{\mathsf{p}(\mathsf{s}(x)) \to x\}$.

Since in general the transformation from $(\mathcal{P},\mathcal{R})$ to $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ does not preserve minimality (i.e., the property that the terms t_1, t_2, ... are terminating in the definition on page 13) of rewrite sequences if $\mathcal{P}$ or $\mathcal{R}$ is duplicating [24], it must be guaranteed that the DP processors of Theorems 38 and 49 as well as Corollaries 48 and 55 do not rely on the minimality of infinite rewrite sequences. For the DP processors of Theorems 38 and 49 this is obviously the case, since e(-raise)-bounds take all infinite rewrite sequences into account. For the DP processors of Corollaries 48 and 55 with $e = \mathrm{top}$ this follows from Lemmata 44 and 52. For $e = \mathrm{match}$ there is also no problem since $e = \mathrm{match}$ can only be used for non-duplicating systems and it is known that usable rules can be used without restrictions for non-duplicating systems.[3] Thus we obtain the following results.

Corollary 61. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and let L be a language. If $\mathcal{P} \cup \mathcal{U}(\mathcal{P},\mathcal{R})$ is both left-linear and e-bounded for L or e-raise-bounded for L then $(\mathcal{P},\mathcal{R})$ is finite.* □

Corollary 62. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem, $s \to t \in \mathcal{P}$ and let L be a language. If $\mathcal{P} \cup \mathcal{U}(\mathcal{P},\mathcal{R})$ is left-linear and $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ is e-DP-bounded for $s \to t$ and L then $(\mathcal{P},\mathcal{R})$ is finite if and only if $(\mathcal{P}\setminus\{s \to t\},\mathcal{R})$ is finite. If $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ is e-raise-DP-bounded for $s \to t$ and L then $(\mathcal{P},\mathcal{R})$ is finite if and only if $(\mathcal{P} \setminus \{s \to t\},\mathcal{R})$ is finite.* □

Example 63. *Consider again the DP problem $(\mathcal{P},\mathcal{R})$ of Example 60. We have $\mathcal{U}(\mathcal{P},\mathcal{R}) = \{\mathsf{p}(\mathsf{s}(x)) \to x\}$. Let $s \to t$ be the first of the two dependency pairs. By using Corollary 48 together with Corollary 62 we can show that $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ is match-DP-bounded for $s \to t$ by 1. Without using usable rules the DP processor of Corollary 48 fails. The reason is that for $(\mathcal{P},\mathcal{R})$, top-DP-bounds must be used because $(\mathcal{P},\mathcal{R})$ is duplicating. However by using top-DP-bounds we do not succeed in constructing a tree automaton that is compatible with* top-DP$(\mathcal{P}, s \to t, \mathcal{R})$.

Note that the DP processors of Theorems 38 and 49 and Corollaries 48 and 55 are in general incomparable to the ones obtained from Corollaries 61 and 62. The reason is that by using $\mathcal{U}(\mathcal{P},\mathcal{R})$ instead of $\mathcal{R}$ it is possible that for duplicating $\mathcal{P} \cup \mathcal{R}$, $(\mathcal{P},\mathcal{U}(\mathcal{P},\mathcal{R}))$ admits an infinite sequence with infinitely many $s \to t$ rewrite steps whereas $(\mathcal{P},\mathcal{R})$ does not.

Example 64. *Consider the TRS $\mathcal{R}$ consisting of the rewrite rule $\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{f}(x,x,x)$. There is one dependency pair, namely $\mathsf{F}(\mathsf{a},\mathsf{b},x) \to \mathsf{F}(x,x,x)$. By using Corollary 48 one can easily check that $(\mathrm{DP}(\mathcal{R}),\mathcal{R})$ is top-DP-bounded for $\mathsf{F}(\mathsf{a},\mathsf{b},x) \to \mathsf{F}(x,x,x)$ by 1 and hence finite. If we combine the DP processor of Corollary 48 with usable rules, finiteness of $(\mathrm{DP}(\mathcal{R}),\mathcal{R})$ can no longer be shown since $(\mathrm{DP}(\mathcal{R}),\mathcal{U}(\mathrm{DP}(\mathcal{R}),\mathcal{R}))$ admits the following minimal cyclic sequence:*

$$\begin{aligned}\mathsf{F}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b})) &\to_{\mathrm{DP}(\mathcal{R})} \mathsf{F}(\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b}))\\ &\to_{\mathcal{U}(\mathrm{DP}(\mathcal{R}),\mathcal{R})} \mathsf{F}(\mathsf{a},\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b})) \to_{\mathcal{U}(\mathrm{DP}(\mathcal{R}),\mathcal{R})} \mathsf{F}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b}))\end{aligned}$$

Here $\mathcal{U}(\mathrm{DP}(\mathcal{R}),\mathcal{R}) = \{\mathsf{g}(x,y) \to x, \mathsf{g}(x,y) \to y\}$.

[3] In [10, Example 14] and [23, Theorem 23] this has been shown for a slightly different definition of usable rules. Nevertheless, this result carries over to the present setting without any problems.

8. Forward Closures

When proving the termination of a TRS $\mathcal{R}$ that is non-overlapping [25] or right-linear [26] it is sufficient to restrict attention to the set $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})$ 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 narrowing. More formally, $\mathrm{RFC}_L(\mathcal{R})$ 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. Dershowitz [26] obtained the following result.

Theorem 65. *A right-linear TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}$ is terminating on* $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})$. □

If we want to prove termination using dependency pairs, we can benefit from the properties of DP problems.

Lemma 66. *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 65 and the definition of DP problems. □

Lemma 66 can be used in connection with the DP processors of Theorems 38 and 49. For the DP processors of Corollaries 48 and 55 we can do better. Since the proof is considerably more complicated than the previous one it is deferred to Appendix B.

Lemma 67. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem and let $s \to t \in \mathcal{P}$. 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* $\mathrm{RFC}_{\{t\}}(\mathcal{P}\cup\mathcal{R})$.

The following concept has been introduced in [2]. 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 $\mathrm{RFC}_L(\mathcal{R})$ for linear $\mathcal{R}$.

Definition 68. 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 $l \to r \in \mathcal{R}_\#$ and $p \in \mathcal{FP}\mathrm{os}(l) \setminus \{\epsilon\}$ then $l[\#]_p \to r\sigma \in \mathcal{R}_\#$. Here the substitution σ is defined by $\sigma(x) = \#$ if $x \in \mathcal{V}\mathrm{ar}(l|_p)$ 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 [2].

Lemma 69. *Let $\mathcal{R}$ be a linear TRS and L a set of linear terms. We have* $\mathrm{RFC}_L(\mathcal{R})\sigma_\# = \to^*_{\mathcal{R}_\#}(L\sigma_\#)$. □

Corollary 70. *If a linear TRS $\mathcal{R}$ is match-bounded for $\to^*_{\mathcal{R}_\#}(\mathrm{rhs}(\mathcal{R})\sigma_\#)$ then $\mathcal{R}$ is terminating.* □

In the case that we consider a DP problem $(\mathcal{P},\mathcal{R})$ the following results can be derived from Lemma 69.

Corollary 71. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem that is match-bounded for $\to^*_{(\mathcal{P}\cup\mathcal{R})_\#}(\mathrm{rhs}(\mathcal{P})\sigma_\#)$. If $\mathcal{P}$ and $\mathcal{R}$ are linear then $(\mathcal{P},\mathcal{R})$ is finite.*

Proof. Since $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})\sigma_\#$ is equal to $\to^*_{(\mathcal{P}\cup\mathcal{R})_\#}(\mathrm{rhs}(\mathcal{P})\sigma_\#)$, $\mathcal{P}\cup\mathcal{R}$ is also match-raise-bounded for $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})\sigma_\#$. (Recall that for linear $\mathcal{P}$ and $\mathcal{R}$, match-boundedness coincide with match-raise-boundedness.) Theorem 13 yields the termination of $\mathcal{P}\cup\mathcal{R}$ on $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})\sigma_\#$. Since rewriting is closed under substitution, $\mathcal{P}\cup\mathcal{R}$ is terminating on $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})$ and hence $(\mathcal{P},\mathcal{R})$ is finite for $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{P})}(\mathcal{P}\cup\mathcal{R})$. Applying Lemma 66 yields the finiteness of $(\mathcal{P},\mathcal{R})$. □

Corollary 72. *Let $(\mathcal{P},\mathcal{R})$ be a DP problem with linear $\mathcal{P}$ and $\mathcal{R}$, and let $s \to t \in \mathcal{P}$. If $(\mathcal{P},\mathcal{R})$ is match-DP-bounded for $s \to t$ and $\to^*_{(\mathcal{P}\cup\mathcal{R})_\#}(\{t\}\sigma_\#)$ then $(\mathcal{P},\mathcal{R})$ is finite if and only if $(\mathcal{P}\setminus\{s\to t\},\mathcal{R})$ is finite.*

PROOF. Since $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})\sigma_\#$ is equal to $\to^*_{(\mathcal{P}\cup\mathcal{R})_\#}(\{t\}\sigma_\#)$, $(\mathcal{P}, \mathcal{R})$ is also match-raise-DP-bounded for $s \to t$ and $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})\sigma_\#$. (Recall that for linear $\mathcal{P}$ and $\mathcal{R}$, match-DP-boundedness coincide with match-raise-DP-boundedness.) Theorem 54 yields that $(\mathcal{P}, \mathcal{R})$ is finite on $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})\sigma_\#$ if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite on $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})\sigma_\#$. Since rewriting is closed under substitution, $(\mathcal{P}, \mathcal{R})$ is finite on $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})$ if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite on $\mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})$. From Lemma 67 we conclude that $(\mathcal{P}, \mathcal{R})$ is finite if and only if $(\mathcal{P} \setminus \{s \to t\}, \mathcal{R})$ is finite. □

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

Definition 73. Let t be a term. The set of linear terms s with $\mathcal{V}\mathrm{ar}(t) \subseteq \mathcal{V}\mathrm{ar}(s)$ for which there exists a variable substitution $\tau\colon \mathcal{V}\mathrm{ar}(s) \setminus \mathcal{V}\mathrm{ar}(t) \to \mathcal{V}\mathrm{ar}(t)$ such that $s\tau = t$ is denoted by $\mathrm{linear}(t)$. Let $\mathcal{R}$ be a TRS. The set of rewrite rules $\{l' \to r \mid l \to r \in \mathcal{R} \text{ and } l' \in \mathrm{linear}(l)\}$ is denoted by $\mathrm{linear}(\mathcal{R})$.

In the following we write $\mathcal{R}'_\#$ for $\mathrm{linear}(\mathcal{R})_\#$. Note that $\mathcal{R}_\# = \mathcal{R}'_\#$ for linear TRSs $\mathcal{R}$. In general $\mathrm{linear}(\mathcal{R})$ and hence $\mathcal{R}'_\#$ consists of infinitely many rewrite rules since variables in $\mathcal{V}\mathrm{ar}(l') \setminus \mathcal{V}\mathrm{ar}(l)$ are not constrained. When using $\mathcal{R}'_\#$ to approximate $\mathrm{RFC}_L(\mathcal{R})\sigma_\#$ it is enough to consider a finite subset of $\mathcal{R}'_\#$ which ignores different variants of rules.

Lemma 74. *Let $\mathcal{R}$ be a right-linear TRS and L a set of linear terms. We have* $\mathrm{RFC}_L(\mathcal{R})\sigma_\# \subseteq \to^*_{\mathcal{R}'_\#}(L\sigma_\#)$.

PROOF. Applying Lemma 69 to $\mathrm{linear}(\mathcal{R})$ yields $\mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))\sigma_\# = \to^*_{\mathcal{R}'_\#}(L\sigma_\#)$ Hence it suffices to prove that $\mathrm{RFC}_L(\mathcal{R})\sigma_\#$ is a subset of $\mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))\sigma_\#$.

First we prove that every term $t \in \mathrm{RFC}_L(\mathcal{R})$ is linear. We use induction on the derivation of t. If $t \in \mathrm{rhs}(\mathcal{R})$ then t is linear because $\mathcal{R}$ is right-linear. Let $t = s[r]_p\sigma$ with $s \in \mathrm{RFC}_L(\mathcal{R})$, $l \to r$ a fresh variant of a rewrite rule in $\mathcal{R}$, and σ a most general unifier of $s|_p$ and l. According to the induction hypothesis s is linear. Hence $\mathcal{V}\mathrm{ar}(s|_p) \cap \mathcal{V}\mathrm{ar}(s[\Box]_p) = \varnothing$. From the linearity of r, $\mathcal{V}\mathrm{ar}(l) \cap \mathcal{V}\mathrm{ar}(s) = \varnothing$, $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l)$, and the fact that σ is a most general unifier, we obtain that $r\sigma$ is linear and $\mathcal{V}\mathrm{ar}(r\sigma) \cap \mathcal{V}\mathrm{ar}(s\sigma[\Box]_p) = \varnothing$. It follows that t is linear.

Next we show that $\mathrm{RFC}_L(\mathcal{R}) \subseteq \mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))$, which immediately gives the inclusion $\mathrm{RFC}_L(\mathcal{R})\sigma_\# \subseteq \mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))\sigma_\#$. Assume to the contrary that this does not hold. This is only possible if there are a term $t \in \mathrm{RFC}_L(\mathcal{R})$ a position $p \in \mathcal{FP}\mathrm{os}(t)$, a fresh variant $l \to r$ of a rewrite rule in $\mathcal{R}$, and a most general unifier σ of $t|_p$ and l such that $t[r]_p\sigma \in \mathrm{RFC}_L(\mathcal{R})$, and $t[r]_p\sigma \notin \mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))$. Since l and t do not share variables, we may assume without loss of generality that σ is idempotent. In order to arrive at a contradiction, we construct a term $l' \in \mathrm{linear}(l)$ and a most general unifier σ' of l' and $t|_p$ such that $t[r]_p\sigma' = t[r]_p\sigma$. Write $l = C[x_1, \ldots, x_n]$ with all variables displayed. Let $q_1, \ldots, q_n$ be the positions of these variables. Because σ is idempotent and t is linear, for every variable $x \in \mathcal{V}\mathrm{ar}(l)$ with $x\sigma \neq x$ there exists a position $q_x \in \{q_1, \ldots, q_n\}$ such that $x\sigma = t|_{pq_x}$. We now replace every x_i in l with $q_i \neq q_{x_i}$ by a fresh variable. This yields a term $l' \in \mathrm{linear}(l)$. Let σ' be an idempotent most general unifier of l' and $t|_p$. It follows from the construction of l' that $\sigma(x) = \sigma'(x)$ for all variables $x \in \mathcal{V}\mathrm{ar}(l) \subseteq \mathcal{V}\mathrm{ar}(l')$. Since $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l)$, $t[r]_p\sigma' = t[r]_p\sigma$ as desired. □

The following example shows what can go wrong if we would not consider all linearizations of the TRS $\mathcal{R}$.

Example 75. *Consider the TRS $\mathcal{R}$ consisting of the following rewrite rules:*

$$\mathsf{f}(x, x) \to \mathsf{f}(\mathsf{h}(x), \mathsf{a}) \qquad \mathsf{f}(\mathsf{h}(x), x) \to \mathsf{g}(x) \qquad \mathsf{g}(x) \to \mathsf{f}(x, \mathsf{a})$$

For the language $L = \mathrm{rhs}(\mathcal{R})$, $\mathrm{RFC}_L(\mathcal{R})\sigma_\#$ consists of the following terms:

$$\mathsf{g}(\#) \qquad \mathsf{g}(\mathsf{a}) \qquad \mathsf{f}(\#, \mathsf{a}) \qquad \mathsf{f}(\mathsf{a}, \mathsf{a}) \qquad \mathsf{f}(\mathsf{h}(\#), \mathsf{a}) \qquad \mathsf{f}(\mathsf{h}(\mathsf{a}), \mathsf{a})$$

If $\mathrm{linear}(\mathcal{R})$ *would consist of the rewrite rules*

$$\mathsf{f}(x,x') \to \mathsf{f}(\mathsf{h}(x),\mathsf{a}) \qquad \mathsf{f}(\mathsf{h}(x),x') \to \mathsf{g}(x) \qquad \mathsf{g}(x) \to \mathsf{f}(x,\mathsf{a})$$

then $\mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))\sigma_\# = \{\mathsf{g}(\mathsf{h}^i(\#)), \mathsf{f}(\mathsf{h}^i(\#),\mathsf{a}) \mid i \geqslant 0\}$. *Observe that* $\mathsf{g}(\mathsf{a})$ *is missing, invalidating Lemma 74. In the proof the linearization* $\mathsf{f}(\mathsf{h}(x'),x) \to \mathsf{g}(x)$ *of* $\mathsf{f}(\mathsf{h}(x),x) \to \mathsf{g}(x)$ *is constructed because the right-hand side* $\mathsf{f}(\mathsf{h}(y),\mathsf{a})$ *is unified with* $\mathsf{f}(\mathsf{h}(x),x)$ *to produce the term* $\mathsf{g}(\mathsf{a})$ *and only the second occurrence of* x *is mapped to a subterm of* $\mathsf{f}(\mathsf{h}(y),\mathsf{a})$. *We remark that* $\mathcal{R}$ *is non-terminating since it admits the cycle* $\mathsf{g}(\mathsf{a}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a},\mathsf{a}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{h}(\mathsf{a}),\mathsf{a}) \to_{\mathcal{R}} \mathsf{g}(\mathsf{a})$. *However, it is easy to see that* $\mathcal{R}$ *is terminating on* $\mathrm{RFC}_L(\mathrm{linear}(\mathcal{R}))\sigma_\#$*:* $\mathsf{f}(\mathsf{h}^i(\#),\mathsf{a})$ *is a normal form and* $\mathsf{g}(\mathsf{h}^i(\#))$ *rewrites only to* $\mathsf{f}(\mathsf{h}^i(\#),\mathsf{a})$, *for all* $i \geqslant 0$.

The following example shows that the reverse inclusion of Lemma 74 does not hold.

Example 76. *For the TRS* $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{b},\mathsf{g}(x)), \mathsf{a} \to \mathsf{b}\}$ *we have* $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})\sigma_\# = \{\mathsf{f}(\mathsf{b},\mathsf{g}(\#)),\mathsf{b}\}$ *and* $\to^*_{\mathcal{R}'_\#}(\mathrm{rhs}(\mathcal{R})\sigma_\#) = \{\mathsf{b}, \mathsf{f}(\mathsf{b},\mathsf{g}^i(\#)), \mathsf{f}(\mathsf{b},\mathsf{g}^i(\mathsf{b})) \mid i \geqslant 1\}$.

Corollary 77. *Let* $\mathcal{R}$ *be a right-linear TRS. If* $\mathcal{R}$ *is match-raise-bounded for* $\to^*_{\mathcal{R}'_\#}(\mathrm{rhs}(\mathcal{R})\sigma_\#)$ *then* $\mathcal{R}$ *is terminating.*

Proof. Since $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})\sigma_\#$ is a subset of $\to^*_{\mathcal{R}'_\#}(\mathrm{rhs}(\mathcal{R})\sigma_\#)$ according to Lemma 74, $\mathcal{R}$ is also match-raise-bounded for $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})\sigma_\#$. Theorem 13 yields the termination of $\mathcal{R}$ on $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})\sigma_\#$. Since rewriting is closed under substitution, $\mathcal{R}$ is terminating on $\mathrm{RFC}_{\mathrm{rhs}(\mathcal{R})}(\mathcal{R})$. From Theorem 65 we conclude that $\mathcal{R}$ is terminating on all terms. □

The above result extends to DP problems without problems; simply replace $(\mathcal{P} \cup \mathcal{R})_\#$ by $(\mathcal{P} \cup \mathcal{R})'_\#$ in the proofs of Corollaries 71 and 72.

Corollary 78. *Let* $(\mathcal{P},\mathcal{R})$ *be a DP problem with right-linear* $\mathcal{P}$ *and* $\mathcal{R}$. *If* $\mathcal{P} \cup \mathcal{R}$ *is match-raise-bounded for* $\to^*_{(\mathcal{P}\cup\mathcal{R})'_\#}(\mathrm{rhs}(\mathcal{P})\sigma_\#)$ *then* $(\mathcal{P},\mathcal{R})$ *is finite.* □

Corollary 79. *Let* $(\mathcal{P},\mathcal{R})$ *be a DP problem such that* $\mathcal{P}$ *and* $\mathcal{R}$ *are right-linear TRSs. Let* $s \to t \in \mathcal{P}$. *If* $(\mathcal{P},\mathcal{R})$ *is match-raise-DP-bounded for* $s \to t$ *and* $\to^*_{(\mathcal{P}\cup\mathcal{R})'_\#}(\{t\sigma_\#\})$ *then* $(\mathcal{P},\mathcal{R})$ *is finite if and only if* $(\mathcal{P} \setminus \{s \to t\},\mathcal{R})$ *is finite.* □

In order to show that a DP problem $(\mathcal{P},\mathcal{R})$ is match(-raise)-DP-bounded for a rewrite rule $s \to t$ and $L = \to^*_{(\mathcal{P}\cup\mathcal{R})'_\#}(\{t\sigma_\#\})$ we have to construct a (quasi-deterministic and raise-consistent) tree automaton that is (quasi-)compatible with match-DP$(\mathcal{P}, s \to t, \mathcal{R})$ and $\mathrm{lift}_0(L)$. We do that by performing two steps. At first we construct a tree automaton $\mathcal{A}$ that is compatible with $(\mathcal{P} \cup \mathcal{R})'_\#$ and $\{t\sigma_\#\}$. Since $(\mathcal{P} \cup \mathcal{R})'_\#$ is left-linear we know by Theorem 17 that $\mathcal{L}(\mathcal{A}) \supseteq L$. In a second step we search for a (quasi-deterministic and raise-consistent) tree automaton that is (quasi-)compatible with match-DP$(\mathcal{P}, s \to t, \mathcal{R})$ and $\mathrm{lift}_0(L)$ as described in Section 3 (Section 5). If such an automaton has been found we know that $(\mathcal{P},\mathcal{R})$ is match(-raise)-DP-bounded for $s \to t$ and L. If $\mathcal{P} \cup \mathcal{R}$ is left-linear the two steps can be combined in an optimized way as described in [2].

9. Experimental Results

The techniques described in the preceding sections are implemented in the termination prover T$_\mathsf{T}$T$_\mathsf{2}$ [27]. T$_\mathsf{T}$T$_\mathsf{2}$ is written in OCaml[4] and consists of about 30000 lines of code. About 15% is used to implement the match-bound technique.

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

Table 1: Summary e-raise-bounds

	no RFC						RFC						
	explicit			implicit			explicit			implicit			
	t	r	rm	t	r	rm	t	r	rm	t	r	rm	
# successes	11	11	12	17	17	19	32	32	32	39	40	41	using c
average time	25	26	24	62	24	23	12	12	13	98	249	243	
# timeouts	147	147	146	141	141	139	126	126	126	119	118	117	
# successes	11	11	12	17	17	19	32	32	32	40	40	41	using qc
average time	11	12	11	39	20	18	8	8	8	963	188	185	
# timeouts	147	147	146	141	141	139	126	126	126	118	118	117	

Since quasi-determinisation is expensive, T2 collects and resolves all (quasi-)compatibility violations with respect to the current automaton before making the automaton quasi-deterministic. Then new (quasi-) compatibility violations are determined and the process is repeated. The violations are resolved by adding new transitions according to the following strategy, which is a variation of the one used by Matchbox [8]. To establish a path $r\sigma \to_\Delta^* q$, T2

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

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, T2 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 T2 on the 1331 TRSs in version 5.0 of the Termination Problem Data Base that fulfill the variable condition, i.e., $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l)$ for each rewrite rule $l \to r \in \mathcal{R}$.[5] 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 Tables 1 and 2.[6] 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 Table 1 we deal with non-left-linear systems (158 TRSs in total) and test for e-raise-boundedness, both with the explicit approach for handling raise rules described in the first paragraph of Section 4 and the implicit approach using raise-consistent tree automata. To simplify the representation we use the abbreviations `t`, `r` and `m` to indicate that we test for top-, roof-, and match-raise-boundedness. Since match-raise-bounds can be only used if the given TRS is non-duplicating, we combine match-raise-bounds with roof-raise-bounds (indicated by `rm`). That means that if the TRS under consideration is non-duplicating we test for match-raise-boundedness; duplicating TRSs are tested for roof-raise-boundedness. In the upper part of the table we construct compatible tree automata (indicated by `c`) whereas in the lower part quasi-compatible tree automata (indicated by `qc`) are used. The positive effect of forward closures (Corollary 77) is clearly visible. By constructing quasi-compatible tree automata instead of compatible tree automata we get an average speed up of 1.4. Furthermore, our results confirm that match-bounds are more powerful than roof-bounds, which in turn are more powerful than top-bounds.

[5] `http://dev.aspsimon.org/projects/termcomp/downloads/`

[6] Full experimental data can be found at `http://cl-informatik.uibk.ac.at/software/ttt2/experiments/matchbounds/`.

Table 2: Summary e(-raise)-DP-bounds

	no RFC					RFC				
		no `ur`		`ur`		no `ur`		`ur`		
	`sp`	`spb`	`spd`	`spb`	`spd`	`spb`	`spd`	`spb`	`spd`	
# successes	498	559	587	585	612	575	589	606	616	using `c`
average time	111	101	190	98	223	106	152	133	152	
# timeouts	12	772	744	746	719	756	742	725	715	
# successes	498	559	589	586	614	575	591	606	618	using `qc`
average time	111	102	218	150	249	111	156	131	165	
# timeouts	12	772	742	745	717	756	740	725	713	

Table 2 shows our results for e(-raise)-DP-bounds. Besides the recursive SCC algorithm [28] and the improved estimated dependency graph processor [21], we use the following four DP processors:

- `s` the subterm criterion of [23],
- `p` polynomial orderings with 0/1 coefficients [24],
- `b` the DP processor of Theorem 38 for left-linear DP problems and the one of Theorem 49 for non-left-linear DP problems,
- `d` the DP processor of Corollary 48 for left-linear DP problems and the one of Corollary 55 for non-left-linear DP problems.

For the latter two, if the DP problem is non-duplicating we take e = match. For duplicating problems we take e = roof for `b` and e = top for `d`. The usage of usable rules (see Corollaries 61 and 62) is indicated by `ur`. The advantage of the DP processors of Corollaries 48 and 55 over the naive ones of Theorems 38 and 49 is clear, although the difference decreases when usable rules and RFC are in effect. Furthermore, by using quasi-compatible tree automata instead of compatible tree automata we obtain some additional termination proofs.

Although not visible from the data in Table 2, our experiments confirm the claim at the end of Section 7 that usable rules can have an adverse effect. For instance, the TRS `Zantema/z28` can be proved terminating by T$_{\mathsf{T}}$T$_2$ using top-DP-bounds. If we compute usable rules in advance, termination can no longer be shown because the added projection rules cause the (quasi-)completion procedure to loop. However, restricting the computation of usable rules to non-duplicating systems in order to avoid these projection rules is not a good strategy since there are duplicating TRSs such as `SchneiderKamp/trs/otto01` which can only be proved terminating with help of usable rules.

The TRS `secret07/TTT2/2` in the Termination Problem Data Base can be proved terminating by T$_{\mathsf{T}}$T$_2$ using match-DP-raise-bounds and RFC. None of the other tools that participated in the termination competitions of 2007[7] and 2008[8] could handle this TRS. The same holds for the TRS `TRCSR/Ex2_Luc02a_iGM` and the string rewrite system `Waldmann07b/size-12-alpha-3-num-469`. The former is handled by T$_{\mathsf{T}}$T$_2$ using top-DP-bounds and the latter using match-DP-bounds together with RFC. In 2008 T$_{\mathsf{T}}$T$_2$ found the following elegant termination proof.

Example 80. *The TRS* `secret06/matchbox/gen-25` *($\mathcal{R}$ in the following) consists of the following rewrite*

[7] `http://www.lri.fr/~marche/termination-competition/2007`
[8] `http://colo5-c703.uibk.ac.at:8080/termcomp`

rules:

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

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

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

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 T*T*2 *consists of the following transitions:*

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

Hence the DP processor of Corollary 48 is applicable. This results in the new DP problem $(\{2\},\mathcal{R})$, which is proved finite by the subterm criterion with the simple projection $\pi(C) = 1$.

10. Conclusion

In this paper we extended the match-bound technique in two directions. We showed how *non-left-linear* rules can be treated by raise rules. To verify e-raise-boundedness, we introduced quasi-deterministic tree automata. Furthermore, to be able to handle raise rules properly during the completion process we introduced the notion of raise-consistent tree automata. We further showed how the match-bound technique can be incorporated into the *dependency pair framework*. For that purpose we introduced two new enrichments which take care of the special properties of dependency pair problems. We showed how to strengthen the method by taking usable rules and forward closures into account. Experimental results demonstrated the usefulness of our results.

An important open question is whether we can use the roof enrichment in connection with dependency pairs. 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 $\xrightarrow{\epsilon}_{\mathrm{roof}(s\to t)}$ ($\xrightarrow{\geqslant}_{\mathrm{roof}(s\to t)}$) rewrite steps. We conjecture that this claim holds for arbitrary $\mathcal{P}$ and $\mathcal{R}$. A positive solution would make additional termination proofs possible: The number of successes in the `spd` columns in Table 2 increases by 6 (when usable rules are in effect) and 8 (without usable rules).

Acknowledgements

We thank the anonymous referees for providing numerous suggestions which helped to improve the presentation.

References

[1] A. Geser, D. Hofbauer, J. Waldmann, Match-bounded string rewriting systems, Applicable Algebra in Engineering, Communication and Computing 15 (3-4) (2004) 149–171.

[2] A. Geser, D. Hofbauer, J. Waldmann, H. Zantema, On tree automata that certify termination of left-linear term rewriting systems, Information and Computation 205 (4) (2007) 512–534.

[3] J. Endrullis, D. Hofbauer, J. Waldmann, Decomposing terminating rewrite relations, in: Proceedings of the 8th International Workshop on Termination (WST), 2006, pp. 39–43.

[4] A. Geser, D. Hofbauer, J. Waldmann, Termination proofs for string rewriting systems via inverse match-bounds, Journal of Automated Reasoning 34 (4) (2005) 365–385.

[5] A. Geser, D. Hofbauer, J. Waldmann, H. Zantema, Finding finite automata that certify termination of string rewriting systems, International Journal of Foundations of Computer Science 16 (3) (2005) 471–486.

[6] J. Giesl, P. Schneider-Kamp, R. Thiemann, AProVE 1.2: Automatic termination proofs in the dependency pair framework, in: Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR), Vol. 4130 of Lecture Notes in Artificial Intelligence, 2006, pp. 281–286.

[7] Jambox, Available from `http://joerg.endrullis.de/`.

[8] J. Waldmann, Matchbox: A tool for match-bounded string rewriting, in: Proceedings of the 15th International Conference on Rewriting Techniques and Applications (RTA), Vol. 3091 of Lecture Notes in Computer Science, 2004, pp. 85–94.

[9] H. Zantema, Termination of rewriting proved automatically, Journal of Automated Reasoning 34 (2) (2005) 105–139.

[10] J. Giesl, R. Thiemann, P. Schneider-Kamp, The dependency pair framework: Combining techniques for automated termination proofs, in: Proceedings of the 11th International Conference on Logic Programming and Automated Reasoning (LPAR), Vol. 3425 of Lecture Notes in Artificial Intelligence, 2004, pp. 301–331.

[11] R. Thiemann, The dp framework for proving termination of term rewriting, Ph.D. thesis, RWTH Aachen, available as technical report AIB-2007-17 (2007).

[12] M. Korp, A. Middeldorp, Proving termination of rewrite systems using bounds, in: Proceedings of the 18th International Conference on Rewriting Techniques and Applications (RTA), Vol. 4533 of Lecture Notes in Computer Science, 2007, pp. 273–287.

[13] M. Korp, A. Middeldorp, Match-bounds with dependency pairs for proving termination of rewrite systems, in: Proceedings of the 2nd International Conference on Language and Automata Theory and Applications (LATA), Vol. 5196 of Lecture Notes in Computer Science, 2008, pp. 321–332.

[14] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

[15] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, Available from `www.grappa.univ-lille3.fr/tata` (2002).

[16] N. Dershowitz, Orderings for term-rewriting systems, Theoretical Computer Science 17 (3) (1982) 279–301.

[17] T. Genet, Decidable approximations of sets of descendants and sets of normal forms, in: Proceedings of the 9th International Conference on Rewriting Techniques and Applications (RTA), Vol. 1379 of Lecture Notes in Computer Science, 1998, pp. 151–165.

[18] A. Middeldorp, Approximating dependency graphs using tree automata techniques, in: Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR), Vol. 2083 of Lecture Notes in Artificial Intelligence, 2001, pp. 593–610.

[19] T. Nagaya, Y. Toyama, Decidability for left-linear growing term rewriting systems, Information and Computation 178 (2) (2002) 499–514.

[20] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theoretical Computer Science 236 (1-2) (2000) 133–178.

[21] J. Giesl, R. Thiemann, P. Schneider-Kamp, Proving and disproving termination of higher-order functions, in: Proceedings of the 5th International Workshop on Frontiers of Combining Systems (FroCoS), Vol. 3717 of Lecture Notes in Artificial Intelligence, 2005, pp. 216–231.

[22] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Improving dependency pairs, in: Proceedings of the 10th International Conference on Logic Programming and Automated Reasoning (LPAR), Vol. 2850 of Lecture Notes in Artificial Intelligence, 2003, pp. 167–182.

[23] N. Hirokawa, A. Middeldorp, Tyrolean termination tool: Techniques and features, Information and Computation 205 (4) (2007) 474–511.

[24] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Mechanizing and improving dependency pairs, Journal of Automated Reasoning 37 (3) (2006) 155–203.

[25] O. Geupel, Overlap closures and termination of term rewriting systems, Report MIP-8922, Universität Passau (1989).

[26] N. Dershowitz, Termination of linear rewriting systems (preliminary version), in: Proceedings of the 8th International Colloquium on Automata, Languages and Programming (ICALP), Vol. 115, 1981, pp. 448–458.

[27] Tyrolean Termination Tool 2, Available from `http://cl-informatik.uibk.ac.at/software/ttt2`.

[28] N. Hirokawa, A. Middeldorp, Automating the dependency pair method, Information and Computation 199 (1-2) (2005) 172–199.

A. Soundness of Match(-Raise)-DP-Bounds

In this appendix we present the proof of Lemma 46. To prove that every restriction of match-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to a finite signature does not admit minimal rewrite sequences with infinitely many $\xrightarrow{\epsilon}_{\mathrm{match}(s\to t)}$ rewrite steps, we mark function symbols of terms as active and inactive to trace the propagation of heights. The idea to consider active and inactive areas of terms occurring in derivations originates from [26]. Below we recall the most important information.

For a signature $\mathcal{F}$, $\overline{\mathcal{F}}$ denotes the set $\{\overline{f} \mid f \in \mathcal{F}\}$ where $\overline{f}$ is a fresh function symbol with the same arity as f. The function symbols in $\overline{\mathcal{F}}$ are called *active* whereas those in $\mathcal{F}$ are called *inactive*. The mappings label$\colon \mathcal{F} \to \overline{\mathcal{F}}$ and unlabel$\colon \overline{\mathcal{F}} \to \mathcal{F}$ are defined as $\mathrm{label}(f) = \overline{f}$ and $\mathrm{unlabel}(\overline{f}) = f$. They are extended to terms and sets of terms in the obvious way. A term $s \in \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ is called *inactive* if $s = \mathrm{unlabel}(s)$. For a term $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ the set $\mathrm{mark}(s)$ consists of all terms t which can be divided into a context $C \in \mathcal{T}(\overline{\mathcal{F}} \cup \{\Box\}, \mathcal{V})$ and terms $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t = C[t_1, \dots, t_n]$ and $\mathrm{unlabel}(t) = s$. For terms $s, t \in \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ with $\mathrm{unlabel}(s) = \mathrm{unlabel}(t)$ we write $s \uparrow t$ for the term u that is uniquely determined by the two conditions (i) $\mathrm{unlabel}(u) = \mathrm{unlabel}(s)$ and (ii) for each position $p \in \mathcal{FP}\mathrm{os}(u)$ we have $u(p) \in \overline{\mathcal{F}}$ if and only if $s(p) \in \overline{\mathcal{F}}$ or $t(p) \in \overline{\mathcal{F}}$. We extend this notion to $\uparrow S$ for finite non-empty sets $S \subset \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ consisting of terms that have the same unlabeled image.

Definition 81. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The TRS $\overline{\mathcal{R}}$ over the signature $\overline{\mathcal{F}} \cup \mathcal{F}$ consists of all rewrite rules $l \to r$ for which there exists a rewrite rule $l' \to r' \in \mathcal{R}$ such that $l \in \mathrm{mark}(l')$, $\mathrm{unlabel}(r) = r'$, and $r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $l(\epsilon) \in \mathcal{F}$ and $r \in \mathcal{T}(\overline{\mathcal{F}}, \mathcal{V})$ if $l(\epsilon) \in \overline{\mathcal{F}}$.

Example 82. *Consider the TRS $\mathcal{R}$ consisting of the rewrite rules $\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{g}(x, \mathsf{p}(y)))$, $\mathsf{g}(\mathsf{p}(x), \mathsf{s}(y)) \to y$, and $\mathsf{g}(x, x) \to \mathsf{f}(\mathsf{s}(x), x)$. The TRS $\overline{\mathcal{R}}$ consists of the following rewrite rules:*

$$\begin{array}{lll}
\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{g}(x, \mathsf{p}(y))) & \mathsf{g}(\mathsf{p}(x), \mathsf{s}(y)) \to y & \mathsf{g}(x, x) \to \mathsf{f}(\mathsf{s}(x), x) \\
\overline{\mathsf{f}}(\mathsf{s}(x), y) \to \overline{\mathsf{s}}(\overline{\mathsf{g}}(x, \overline{\mathsf{p}}(y))) & \overline{\mathsf{g}}(\mathsf{p}(x), \mathsf{s}(y)) \to y & \overline{\mathsf{g}}(x, x) \to \overline{\mathsf{f}}(\overline{\mathsf{s}}(x), x) \\
\overline{\mathsf{f}}(\overline{\mathsf{s}}(x), y) \to \overline{\mathsf{s}}(\overline{\mathsf{g}}(x, \overline{\mathsf{p}}(y))) & \overline{\mathsf{g}}(\overline{\mathsf{p}}(x), \mathsf{s}(y)) \to y & \\
 & \overline{\mathsf{g}}(\mathsf{p}(x), \overline{\mathsf{s}}(y)) \to y & \\
 & \overline{\mathsf{g}}(\overline{\mathsf{p}}(x), \overline{\mathsf{s}}(y)) \to y &
\end{array}$$

Definition 83. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. We define the relation $\twoheadrightarrow$ on $\mathcal{T}(\mathcal{F} \cup \overline{\mathcal{F}}, \mathcal{V})$ as follows: $s \twoheadrightarrow_{\mathcal{R}} t$ if and only if there exist a rewrite rule $l \to r \in \overline{\mathcal{R}}$, a position $p \in \mathcal{P}\mathrm{os}(s)$, a context C, and terms $s_1, \dots, s_n$ such that $l = C[x_1, \dots, x_n]$ with all variables displayed, $s|_p = C[s_1, \dots, s_n]$, $\mathrm{unlabel}(s_i) = \mathrm{unlabel}(s_j)$ whenever $x_i = x_j$, and $t = s[r\theta]_p$. Here the substitution θ is defined as follows:

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

An immediate consequence of the next lemma is that every rewrite sequence in $\mathcal{R}$ can be lifted to a rewrite sequence in $\overline{\mathcal{R}}$.

Lemma 84. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. If $s \to_{\mathcal{R}} t$ then for all terms $s' \in \mathrm{mark}(s)$ there exists a term $t' \in \mathrm{mark}(t)$ such that $s' \twoheadrightarrow_{\mathcal{R}} t'$.* □

Definition 85. Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$, $l \to r \in \overline{\mathcal{R}}$ a rewrite rule, t a term, $p \in \mathcal{P}\mathrm{os}(t)$ a position, and σ a substitution. The rewrite step $t[l\sigma]_p \twoheadrightarrow t[r\sigma]_p$ is said to be *active* if $t(p) \in \overline{\mathcal{F}}$ and *inactive* if $t(p) \in \mathcal{F}$. We use $\xrightarrow{\mathrm{a}}$ to denote active steps and $\xrightarrow{\mathrm{i}}$ to denote inactive steps. An active rewrite step $t[l\sigma]_p \xrightarrow{\mathrm{a}} t[r\sigma]_p$ is said to be *strongly active* if $\mathcal{F}\mathrm{un}(l) \subseteq \overline{\mathcal{F}}$.

Example 86. *With respect to the TRS $\mathcal{R}$ of the previous example, the term $\overline{\mathsf{f}}(\mathsf{s}(\mathsf{p}(x)), \mathsf{g}(\mathsf{p}(y), \mathsf{s}(x)))$ admits the following rewrite sequence:*

$$\overline{\mathsf{f}}(\mathsf{s}(\mathsf{p}(x)), \mathsf{g}(\mathsf{p}(y), \mathsf{s}(x))) \xrightarrow{\mathrm{i}}_{\mathcal{R}} \overline{\mathsf{f}}(\mathsf{s}(\mathsf{p}(x)), x) \xrightarrow{\mathrm{a}}_{\mathcal{R}} \overline{\mathsf{s}}(\overline{\mathsf{g}}(\mathsf{p}(x), \overline{\mathsf{p}}(x))) \xrightarrow{\mathrm{a}}_{\mathcal{R}} \overline{\mathsf{s}}(\overline{\mathsf{f}}(\overline{\mathsf{s}}(\overline{\mathsf{p}}(x)), \overline{\mathsf{p}}(x)))$$

Note that the second active rewrite step is strongly active.

A second important property of $\overline{\mathcal{R}}$ is that active function symbols always stay above inactive function symbols.

Lemma 87. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and $s \in \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ a term such that $s \in \mathrm{mark}(\mathrm{unlabel}(s))$. If $s \to_{\mathcal{R}}^* t$ then $t \in \mathrm{mark}(\mathrm{unlabel}(t))$.* □

The proof of Lemma 46 is based on the observation that from some point on in each minimal rewrite sequence with respect to $\to_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})}$ only strongly active steps are applied. Below we prove the correctness of this observation. Minimal terms in Lemma 88 are terms that have the property that all proper subterms are terminating.

Lemma 88. *Let $\mathcal{R}$ be a non-duplicating TRS over a signature $\mathcal{F}$ and $s \in \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ a minimal term such that the root symbol of s is active and all other function symbols are inactive. If s starts a rewrite sequence $s = s_1 \to_{\mathcal{R}} s_2 \to_{\mathcal{R}} \cdots$ then there exists an $i \geqslant 1$ such that all rewrite steps in the rewrite sequence starting from s_i are strongly active.*

Proof. The proof of this lemma is based on the following observations:

- active $\xrightarrow{\mathrm{a}}_{\mathcal{R}}$ steps cannot increase the number of inactive symbols because $\mathcal{R}$ is non-duplicating,
- proper subterms of s are terminating due to the minimality assumption,
- maximal inactive subterms of s_j for $j \geqslant 1$ can be traced back to inactive subterms of s.

The first two observations holds trivially. To show the correctness of the last one, we prove the following claim:

> If $t \to_{\mathcal{R}}^* u$ for some terms $t, u \in \mathcal{T}(\overline{\mathcal{F}} \cup \mathcal{F}, \mathcal{V})$ such that $t \in \mathrm{mark}(\mathrm{unlabel}(t))$, then for each inactive subterm u' of u there is an inactive subterm t' of t and a context C such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[u']$.

We prove the claim by induction on the length of the derivation. The base case is trivial. Assume now that $t \to_{\mathcal{R}}^+ u$. Then there are a position p, a substitution σ, and a rewrite rule $l \to r \in \overline{\mathcal{R}}$ such that $t \to_{\mathcal{R}}^* u[l\sigma]_p \to u[r\sigma]_p = u$. Let u' be an inactive subterm of u at some position $q \parallel p$. Then $u' = u[l\sigma]_p|_q$. Due to the induction hypothesis we know that there exists an inactive subterm t' of t and a context C such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[u']$. Assume now that u' is an inactive subterm of u at some position q such that either $q < p$ or $q \geqslant p$. We distinguish between these two cases.

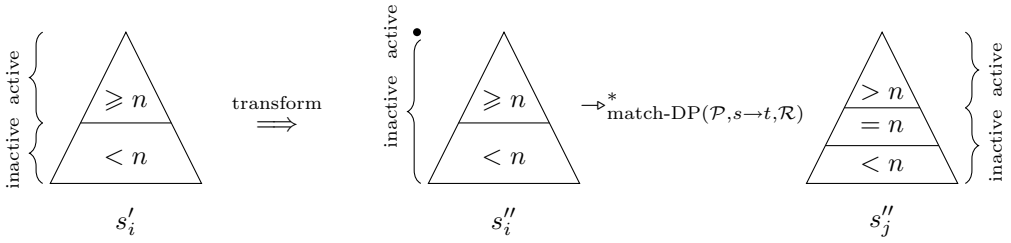
- If $q < p$ then $u[l\sigma]_p \xrightarrow{\mathrm{i}} u$. Let $v = (u[l\sigma]_p)|_q$. Obviously, v is an inactive subterm of $u[l\sigma]_p$ and $v \xrightarrow{\mathrm{i}}_{l\to r} u'$. The induction hypothesis yields an inactive subterm t' of t and a context C such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[v]$. Hence $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[v] \xrightarrow{\mathrm{i}}_{l\to r} C[u']$.
- If $q \geqslant p$ then either $u[l\sigma]_p \xrightarrow{\mathrm{i}} u$ or $u[l\sigma]_p \xrightarrow{\mathrm{a}} u$. In the former case we have $l\sigma \xrightarrow{\mathrm{i}}_{l\to r} D[u']$ for some context D. The induction hypothesis yields an inactive subterm t' of t and a context C such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[l\sigma]$. Hence $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[D[u']]$. Next suppose that $u[l\sigma]_p \xrightarrow{\mathrm{a}} u$. Since all function symbols of r are active, we conclude that u' is a subterm of $x\sigma$ for some variable $x \in \mathcal{V}\mathrm{ar}(r)$. Hence u' is an inactive subterm of $u[l\sigma]_p$. The induction hypothesis yields an inactive subterm t' of t and a context C such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* C[u']$.

This concludes the proof of the claim.

Now, from the above observations it follows that the infinite sequence starting from s contains only finitely many inactive $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps. Hence a tail of the sequence consists entirely of $\xrightarrow{\mathrm{a}}_{\mathcal{R}}$ steps. Steps in this tail that are not strongly active consume at least one inactive symbol whereas the strongly active steps do not increase the number of inactive symbols. Hence from some point, only strongly active steps are applied. This completes the proof of the lemma. □

The following example shows that the previous lemma does not hold for duplicating TRS. Hence it cannot be used to prove soundness of roof-DP($\mathcal{P}, s \to t, \mathcal{R}$).

Figure 1: The claim in the proof of Lemma 46



Example 89. *Consider the TRS $\mathcal{R}$ consisting of the rewrite rules $\mathsf{a} \to \mathsf{b}$ and $\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{f}(x,x,x)$. The term $\overline{\mathsf{f}}(\mathsf{a},\mathsf{b},\mathsf{a})$ admits the rewrite sequence $\overline{\mathsf{f}}(\mathsf{a},\mathsf{b},\mathsf{a}) \overset{\mathrm{a}}{\to\!\triangleright}_{\mathcal{R}} \overline{\mathsf{f}}(\mathsf{a},\mathsf{a},\mathsf{a}) \overset{\mathrm{i}}{\to\!\triangleright}_{\mathcal{R}} \overline{\mathsf{f}}(\mathsf{a},\mathsf{b},\mathsf{a}) \overset{\mathrm{a}}{\to\!\triangleright}_{\mathcal{R}} \cdots$. Since this sequence does not contain any strongly active rewrite steps, it is obvious that the previous lemma does not hold.*

By using the rewrite relation $\to\!\triangleright_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})}$ we are now ready to prove that no restriction of match-DP$(\mathcal{P}, s \to t, \mathcal{R})$ to a finite signature admits minimal rewrite sequences containing infinitely many $\overset{\epsilon}{\to}_{\text{match}(s\to t)}$ rewrite steps.

Proof (of Lemma 46). Assume to the contrary that there is a minimal rewrite sequence of the form

$$s_1 \overset{\epsilon}{\to}_{\text{match}(s\to t)} t_1 \to^*_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} s_2 \overset{\epsilon}{\to}_{\text{match}(s\to t)} t_2 \to^*_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} \cdots$$

where we assume without loss of generality that $s_1 \in \mathcal{T}(\mathcal{F}_{\{0\}}, \mathcal{V})$. Here $\mathcal{F}$ denotes the signature of $\mathcal{P} \cup \mathcal{R}$. To trace the propagation of heights in this sequence we switch from $\to$ to $\to\!\triangleright$. To simplify the representation we assume that for terms $s_i'^{\cdots\prime}$ we have $\text{unlabel}(s_i'^{\cdots\prime}) = s_i$. Moreover, the infinite rewrite sequence starting from $s_i'^{\cdots\prime}$ is projected onto the above sequence from s_i by applying the unlabel operation. The terms $s_i'^{\cdots\prime}$ will be constructed as we go along. To prove the lemma, we first prove the following claim (illustrated in Figure 1):

> Let s'_i be a term such that (i) all rewrite steps in the infinite rewrite sequence starting from s'_i are strongly active and (ii) the height of all active symbols in s'_i is at least n. Further let s''_i be the term obtained from s'_i by labeling all function symbols as inactive except the root symbol. Then there is a term s''_j with $j \geqslant i$ such that all rewrite steps of the infinite sequence starting at s''_j are strongly active and the height of every active function symbol in s''_j is at least $n + 1$.

Lemma 88 yields a term s''_j with $j \geqslant i$ such that all rewrite steps of the rewrite sequence starting from s''_j are strongly active. Because all rewrite steps in the infinite rewrite sequence starting from s'_i are strongly active, we know that whenever

$$s'_i \overset{\mathrm{a}}{\to\!\triangleright}^*_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} u = u[l\sigma]_p \overset{\mathrm{a}}{\to\!\triangleright}_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} u[r\sigma]_p$$

for some term u, rewrite rule $l \to r$, position p, and substitution σ, the minimal height of function symbols in l is at least n. Since the rewrite sequence starting from s'_i is equivalent to the rewrite sequence starting at the term s''_i after unlabeling, this property holds also for s''_i. Since the rewrite step $s''_i \to\!\triangleright_{\text{match}(s\to t)} t''_i$ is active, we know that the height of all active function symbols in t''_i is at least $n + 1$. Hence, whenever

$$t''_i \to\!\triangleright^*_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} u = u[l\sigma]_p \overset{\mathrm{a}}{\to\!\triangleright}_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} u[r\sigma]_p$$

the height of the root symbol of l is at least $n + 1$. Together with the fact that the minimal height of the function symbols in the redex pattern of an active rewrite step is at least n, we can conclude from Definition 39 that the height of each active function symbol in s''_j must be at least $n + 1$. This completes the proof of the claim.

Now let s'_1 be the term obtained from s_1 by marking the root symbol as active. Lemma 88 yields a term s'_{i_1} with $i_1 \geqslant 1$ such that $s'_1 \to^*_{\text{match-DP}(\mathcal{P},s\to t,\mathcal{R})} s'_{i_1}$ and all rewrite steps in the rewrite sequence starting from s'_{i_1} are strongly active. Since $s'_1 \xrightarrow{\mathrm{a}}_{\text{match}(s\to t)} t'_1$, we know that the height of every active function symbol in t'_1 is 1. It follows that the height of each active function symbol in s'_{i_1} is at least 1. Let s''_{i_1} be the term obtained from s'_{i_1} by inactivating all function symbols below the root. Applying the claim yields a term s''_{i_2} with $i_2 \geqslant i_1$ such that all rewrite steps in the infinite sequence starting from s''_{i_2} are strongly active and the height of all active function symbols in s''_{i_2} is at least 2. Repeating this argumentation produces increasingly greater heights. As soon as we reach height $c+1$ we obtain a contradiction with the assumptions of Lemma 46. □

B. Correctness of $\mathrm{RFC}_t(\mathcal{P} \cup \mathcal{R})$

In this appendix we present the proof of Lemma 67. Similar as in the proof of Lemma 46, we use the rewrite relation $\to\!\!\triangleright$ to obtain detailed information about derivations.

Proof (of Lemma 67). To prove the only-if direction, let

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

be a minimal rewrite sequence. To trace the application of rewrite rules in this sequence we switch from $\to$ to $\to\!\!\triangleright$. Let s'_1 be the term obtained from s_1 by marking the root symbol as active. Lemma 88 yields an $i \geqslant 1$ such that all rewrite steps in the rewrite sequence starting from s'_i are strongly active. Let $l_1 \to r_1, \ldots, l_n \to r_n$ be (fresh variants of) rewrite rules in $\mathcal{P} \cup \mathcal{R}$ such that

$$s'_1 \xrightarrow{\mathrm{a}}_{s\to t} t'_1 \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} u_1 \xrightarrow{\mathrm{a}}_{l_1\to r_1} v_1 \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} u_2 \xrightarrow{\mathrm{a}}_{l_2\to r_2} v_2 \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} \cdots \xrightarrow{\mathrm{a}}_{l_n\to r_n} v_n \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} s'_i$$

with all active steps displayed and let $p_1, \ldots, p_n$ be the positions at which the rewrite rules $l_1 \to r_1, \ldots, l_n \to r_n$ are applied. To prove the lemma we first show that we can characterize the active region of s'_i with the help of the definition of the right-hand sides of forward closures. Let $v_0 = t'_1$. We define linear terms $w_0, \ldots, w_n \in \mathcal{T}(\overline{\mathcal{F}}, \mathcal{V})$ and substitutions $\tau_0, \ldots, \tau_n \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that for all $0 \leqslant j \leqslant n$ the following properties hold: (1) $v_j = w_j\tau_j$ and (2) $\mathrm{unlabel}(w_j) \in \mathrm{RFC}_{\{t\}}(\mathcal{P} \cup \mathcal{R})$.

We perform induction on j. First consider $j = 0$. Define $w_0 = \mathrm{label}(t)$. Since $\mathrm{unlabel}(w_0) = t$, property (2) holds trivially. Since $\mathcal{P}$ is right-linear, w_0 is linear. Obviously, $w_0 \in \mathcal{T}(\overline{\mathcal{F}}, \mathcal{V})$. Since $t'_1 = v_0$ is an instance of $\mathrm{label}(t)$, there exists a substitution τ_0 such that $w_0\tau_0 = v_0$. We may assume that $\tau_0 \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ as there are no other active symbols in v_0 besides the ones in $\mathrm{label}(t)$. Hence property (1) also holds.

Now let $j \geqslant 1$. Since all steps that take place between v_{j-1} and u_j are inactive, we infer that the active part part of u_j is identical to the active part of v_{j-1}. Since the active rewrite step $u_j \xrightarrow{\mathrm{a}}_{l_j\to r_j} v_j$ requires that the root symbol of the contracted redex is active we know that p_j is an active position in v_{j-1} and thus a non-variable position in w_{j-1}. Let $l'_j \to r'_j$ be the rule in $\overline{\mathcal{P} \cup \mathcal{R}}$ that is used to rewrite u_j to v_j. (So $l_j = \mathrm{unlabel}(l'_j)$ and $r'_j = \mathrm{label}(r_j)$.) Since w_{j-1} is linear, it follows that $w_{j-1}|_{p_j}$ unifies with l'_j. Let σ_j be an idempotent most general unifier of these two terms. Define $w_j = w_{j-1}[r'_j]_{p_j}\sigma_j$. Since w_{j-1} is linear, $\mathcal{V}\mathrm{ar}(w_{j-1}) \cap \mathcal{V}\mathrm{ar}(l'_j) = \varnothing$, and σ_j is idempotent, $(\mathcal{V}\mathrm{ar}(w_{j-1}) \setminus \mathcal{V}\mathrm{ar}(w_{j-1}|_{p_j})) \cap \mathcal{D}\mathrm{om}(\sigma_j) = \varnothing$ and thus $w_j = w_{j-1}[r'_j\sigma_j]_{p_j}$. Because w_{j-1} contains only active function symbols, $x\sigma_j = \mathrm{label}(x\sigma_j)$ for all $x \in \mathcal{V}\mathrm{ar}(l'_j)$ and hence $w_j \in \mathcal{T}(\overline{\mathcal{F}}, \mathcal{V})$. Since $\mathcal{P}$ and $\mathcal{R}$ are right-linear, it follows that w_j is linear. We have $w_{j-1}[l'_j\sigma_j]_{p_j} \xrightarrow{\mathrm{a}}\!\!\triangleright\, w_{j-1}[r'_j\sigma_j]_{p_j} = w_j$. It follows that w_j represents the active region of v_j and thus $v_j = w_j\tau_j$ for some substitution $\tau_j \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, which proves property (1). Property (2) holds by construction.

Since $v_n \xrightarrow{\mathrm{i}}{}^*\!\!\triangleright\, s'_i$, the active part of s'_i is the same as the active part of v_n. It follows that $s'_i = w_n\tau$ for some substitution $\tau \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$. Since all rewrite steps in the infinite sequence starting from s'_i are strongly active, these steps can also be performed when starting from w_n. Removing all labels produces an infinite sequence starting at the term $\mathrm{unlabel}(w_n)$ with the desired properties.

The if direction of the lemma holds trivially. □