Electronic Notes in Theoretical Computer Science 70 No. 6 (2002)
URL: `http://www.elsevier.nl/locate/entcs/volume70.html` 20 pages

Approximations for Strategies and Termination

Aart Middeldorp [1,2]

Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba 305-8573, Japan

Abstract

The theorem of Huet and Lévy stating that for orthogonal rewrite systems (i) every reducible term contains a needed redex and (ii) repeated contraction of needed redexes results in a normal form if the term under consideration has a normal form, forms the basis of all results on optimal normalizing strategies for orthogonal rewrite systems. However, needed redexes are not computable in general.

In the paper we illustrate, based on the framework introduced in [6], how the use of approximations and their associated tree automata results allows one to obtain decidable conditions in a simple and elegant way.

We further show how the very same ideas can be used to improve [18] the dependency pair method of Arts and Giesl [1] for proving termination of rewrite systems automatically. More precisely, we show how approximations and tree automata techniques provide a better estimation of the dependency graph. This graph determines the ordering constraints that have to be solved in order to conclude termination. Furthermore, we present a new estimation of the dependency graph that does not rely on computationally expensive tree automata techniques.

1 Introduction and Preliminaries

This paper is about strategies and termination in first-order term rewriting. At first sight, these two topics seem to have little in common. Research on strategies is concerned with term rewrite systems (TRSs for short) that admit infinite rewrite sequences and addresses questions like how to compute normal forms in an optimal way. Research on termination is concerned with developing methods which can be used to show the absence of infinite rewrite sequences. In this paper we argue that the concept of approximation is very

[1] Supported by the Grant-in-Aid for Scientific Research (C)(2) 13224006 of the Ministry of Education, Culture, Sports, Science and Technology of Japan.
[2] Email: `ami@is.tsukuba.ac.jp`.

useful to (1) characterize large decidable classes of TRSs that admit a computable optimal strategy for computing normal forms and (2) improve the dependency pair method of Arts and Giesl for proving termination of TRSs automatically.

The remainder of the paper is organized as follows. In Section 2 we give a brief introduction to call-by-need strategies. In Section 3 we do the same for the dependency pair method. Section 4 introduces various approximations and the tree automata results that explain their usefulness. We apply these results in Sections 5 and 6 to the study of call-by-need strategies and automatic termination proofs, respectively. Section 6 also contains a new estimation of the dependency graph that does not rely on the results of Section 4. Except for this latter part, all results presented in this paper appeared before in Durand and Middeldorp [6] (strategies) and Middeldorp [18] (termination).

2 Call by Need Strategies

We assume familiarity with the basics of term rewriting (e.g. [2]). We assume that the rewrite rules $l \to r$ of a TRS satisfy $l \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(l)$. If these conditions are not imposed we find it useful to speak of *extended* TRSs (eTRSs). Such systems arise naturally when we approximate TRSs or orient the rewrite rules from right to left, as explained in Section 4. Note that eTRSs which are not TRSs can never be terminating, but in Section 6 we will make clear that such eTRSs are very useful for automatically proving termination of TRSs. Throughout the paper we assume that all (e)TRSs are finite. Moreover, we consider rewriting on ground terms only. So we assume that the set of ground terms is non-empty. These requirements entail no loss of generality.

Example 2.1 Consider the TRS $\mathcal{R}$ consisting of the rewrite rules

$$\begin{aligned} 0 + y &\to y & \mathsf{fib} &\to \mathsf{f}(\mathsf{s}(0),\mathsf{s}(0)) \\ \mathsf{s}(x) + y &\to \mathsf{s}(x+y) & \mathsf{f}(x,y) &\to x : \mathsf{f}(y, x+y) \\ \mathsf{nth}(0, y:z) &\to y & \mathsf{nth}(\mathsf{s}(x), y:z) &\to \mathsf{nth}(x,z) \end{aligned}$$

for computing Fibonacci numbers. The term $t = \mathsf{nth}(\mathsf{s}(\mathsf{s}(0)),\mathsf{fib})$ admits the normal form $\mathsf{s}(\mathsf{s}(0))$: [3]

$$\begin{aligned} t &\to \mathsf{nth}(2,\mathsf{f}(1,1)) \to \mathsf{nth}(2, 1 : \mathsf{f}(1,1+1)) \to \mathsf{nth}(1,\mathsf{f}(1,1+1)) \\ &\to \mathsf{nth}(1,\mathsf{f}(1,\mathsf{s}(0+1))) \to \mathsf{nth}(1,\mathsf{f}(1,2)) \to \mathsf{nth}(1, 1 : \mathsf{f}(2,1+2)) \\ &\to \mathsf{nth}(0,\mathsf{f}(2,1+2)) \to \mathsf{nth}(0,\mathsf{f}(2,\mathsf{s}(0+2))) \to \mathsf{nth}(0,\mathsf{f}(2,3)) \\ &\to \mathsf{nth}(0, 2 : \mathsf{f}(3,2+3)) \to 2 \end{aligned}$$

but an eager (innermost) strategy will produce an infinite rewrite sequence.

[3] In the rewrite sequence we denote $\mathsf{s}^n(0)$ by n for $n = 1, 2, 3$.

If a term t has a normal form then we can always compute a normal form of t by computing the reducts of t in a breadth-first manner until we encounter a normal form. However, this is a highly inefficient way to compute normal forms. In practice, normal forms are computed by adopting a suitable strategy for selecting the redexes which are to be contracted in each step. A strategy is called *normalizing* if it succeeds in computing normal forms for all terms that admit a normal form. For the class of *orthogonal* TRSs several normalizations results are known (see e.g. Klop [14]). For instance, O'Donnell [20] proved that the *parallel-outermost* strategy (which contracts in a single step all outermost redexes in parallel) is normalizing for all orthogonal TRSs. However, parallel-outermost is not an optimal strategy as it may perform useless steps.

Example 2.2 Consider the TRS $\mathcal{R}$ consisting of the rewrite rules

$$\begin{aligned} 0 + y &\to y & 0 \times y &\to 0 \\ \mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{s}(x) \times y &\to (x \times y) + y \end{aligned}$$

Faced with the term $t = (0 \times \mathsf{s}(0)) \times (0 + \mathsf{s}(0))$, the parallel-outermost strategy computes its normal form 0 by contracting three redexes in two steps:

$$\underline{(0 \times \mathsf{s}(0))} \times \underline{(0 + \mathsf{s}(0))} \to^* \underline{0 \times \mathsf{s}(0)} \to 0$$

The normal form 0 can also be reached by contracting just two redexes:

$$\underline{(0 \times \mathsf{s}(0))} \times (0 + \mathsf{s}(0)) \to \underline{0 \times \mathsf{s}(0)} \to 0$$

So redex $0 + \mathsf{s}(0)$ in t is not needed to reach the normal form.

An optimal strategy selects only *needed* redexes. Formally, a redex Δ in a term t is needed if in every rewrite sequence from t to normal form a descendant of Δ is contracted.

Example 2.1 (continued) In the displayed rewrite sequence $\mathsf{nth}(2, \mathsf{fib}) \to^* 2$ non-needed redexes are contracted. For instance, redex $1 + 2$ in the term $\mathsf{nth}(0, \mathsf{f}(2, 1 + 2))$ is non-needed:

$$\mathsf{nth}(0, \underline{\mathsf{f}(2, 1 + 2)}) \to \underline{\mathsf{nth}(0, 2 : \mathsf{f}(1 + 2, 2 + (1 + 2)))} \to 2$$

Huet and Lévy [11] proved the following important result.

Theorem 2.1 *Let $\mathcal{R}$ be an orthogonal TRS.*

(i) *Every reducible term contains a needed redex.*

(ii) *Repeated contraction of needed redexes results in a normal form, whenever the term under consideration has a normal form.*

So, for orthogonal TRSs, the strategy that always selects a needed redex for contraction is normalizing and optimal.[4] Unfortunately, needed redexes are not computable in general. Hence, in order to obtain a *computable* optimal strategy, we need to find (1) decidable approximations of neededness and (2) (decidable) classes of rewrite systems which ensure that every reducible term has a needed redex identified by (1).

Starting with the seminal work of Huet and Lévy [11] on *strong sequentiality*, these issues have been extensively investigated in the literature (e.g. [3,12,13,14,15,22,28]). In all these works Huet and Lévy's notions of index, ω-reduction, and sequentiality figure prominently. Basically, to determine whether an outermost redex Δ in a term $t = C[\Delta]$ is needed, Δ is replaced by a fresh symbol $\bullet$ and all other outermost redexes in t are replaced by Ω which represents an unknown term. It is then investigated whether $\bullet$ can disappear from the resulting Ω-term t' by using some computable notion of partial reduction. If this is not the case, then we may conclude that redex Δ in t is needed. Since neededness of redex Δ in t is solely determined by its position in t (cf. Lemma 5.1), replacing redex Δ in t by $\bullet$ incurs no loss of generality. However, by replacing all other outermost redexes by Ω essential information may be lost for determining the neededness of Δ. This is illustrated in the following example, which shows that needed redexes are not independent of other redexes.

Example 2.2 (continued) An arbitrary redex Δ is needed in the term $(0 + \mathsf{s}(0)) \times \Delta$ but not in the term $(0 \times \mathsf{s}(0)) \times \Delta$: $\underline{(0 \times \mathsf{s}(0))} \times \Delta \to \underline{0 \times \Delta} \to 0$.

In Section 5 we present the framework of Durand and Middeldorp [6] for decidable call-by-need computations in which issues (1) and (2) are addressed directly.

3 Dependency Pairs

In the area of term rewriting, termination has been studied for several decades and many powerful techniques have been developed (see [5,24,30] for surveys). Since termination is an undecidable property of rewrite systems, no method will work in all cases. The traditional techniques for *automated* termination proofs of TRSs are simplification orders like the recursive path order, the Knuth-Bendix order, and (most) polynomial orders. Recently, the termination proving power of these techniques has been significantly extended by the *dependency pair method* of Arts and Giesl. In this method a TRS is transformed into a set of ordering constraints such that termination of the TRS is equivalent to the solvability of the constraints. The generated constraints are typically solved by traditional simplification orders. The power of the dependency pair method has been amply illustrated in a sequence of papers by Arts, Giesl, Ohlebusch, and Urbain [1,8,9,21,29].

[4] We ignore here the problem of duplication of (needed) redexes, which can be solved if common subterms are shared.

In this section we recall the basic notions and results related to the dependency pair technique. We refer to [1,8,9] for motivations and further refinements. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The subset $\mathcal{D} \subseteq \mathcal{F}$ of *defined* symbols consists of all root symbols of left-hand sides of rewrite rules. Let $\mathcal{F}^\sharp$ denote the union of $\mathcal{F}$ and $\{f^\sharp \mid f \in \mathcal{D}\}$ where $f^\sharp$ has the same arity as f. Given a term $t = f(t_1, \dots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with f defined, we write $t^\sharp$ for the term $f^\sharp(t_1, \dots, t_n)$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with $\mathsf{root}(t) \in \mathcal{D}$ then the rewrite rule $l^\sharp \to t^\sharp$ is called a dependency pair of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by $\mathsf{DP}(\mathcal{R})$. In examples we write F for $\mathsf{f}^\sharp$.

Example 3.1 The TRS $\mathcal{R}$ consisting of the rewrite rules

$$\begin{aligned} \mathsf{even}(x) &\to \mathsf{eo}(x, 0) & \mathsf{eo}(x, 0) &\to \mathsf{not}(\mathsf{eo}(x, \mathsf{s}(0))) \\ \mathsf{odd}(x) &\to \mathsf{eo}(x, \mathsf{s}(0)) & \mathsf{eo}(0, \mathsf{s}(0)) &\to \mathsf{false} \\ \mathsf{not}(\mathsf{true}) &\to \mathsf{false} & \mathsf{eo}(\mathsf{s}(x), \mathsf{s}(0)) &\to \mathsf{eo}(x, 0) \\ \mathsf{not}(\mathsf{false}) &\to \mathsf{true} \end{aligned}$$

has five dependency pairs:

$$\begin{aligned} \mathsf{EVEN}(x) &\to \mathsf{EO}(x, 0) \quad (1) & \mathsf{EO}(x, 0) &\to \mathsf{NOT}(\mathsf{eo}(x, \mathsf{s}(0))) \quad (3) \\ \mathsf{ODD}(x) &\to \mathsf{EO}(x, \mathsf{s}(0)) \quad (2) & \mathsf{EO}(x, 0) &\to \mathsf{EO}(x, \mathsf{s}(0)) \quad (4) \\ & & \mathsf{EO}(\mathsf{s}(x), \mathsf{s}(0)) &\to \mathsf{EO}(x, 0) \quad (5) \end{aligned}$$

A preorder is a transitive and reflexive relation. A rewrite preorder is a preorder $\gtrsim$ on terms that is closed under contexts and substitutions. A *reduction pair* consists of a rewrite preorder $\gtrsim$ and a compatible well-founded order $>$ which is closed under substitutions. Here compatibility means that the inclusion $\gtrsim \cdot > \subseteq >$ or the inclusion $> \cdot \gtrsim \subseteq >$ holds.

Theorem 3.2 (Arts and Giesl [1]) *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if there exists a reduction pair $(\gtrsim, >)$ such that $\mathcal{R} \subseteq \gtrsim$ and $\mathsf{DP}(\mathcal{R}) \subseteq >$.*

Because rewrite rules are just pairs of terms, $\mathcal{R} \subseteq \gtrsim$ is a shorthand for $l \gtrsim r$ for every rewrite rule $l \to r \in \mathcal{R}$ and $\mathsf{DP}(\mathcal{R}) \subseteq >$ denotes that $l > r$ for every dependency pair $l \to r \in \mathsf{DP}(\mathcal{R})$.

Example 3.1 (continued) According to Theorem 3.2, termination of $\mathcal{R}$ amounts to finding a reduction pair $(\gtrsim, >)$ such that

$$\begin{aligned} \mathsf{even}(x) &\gtrsim \mathsf{eo}(x, 0) & \mathsf{eo}(x, 0) &\gtrsim \mathsf{not}(\mathsf{eo}(x, \mathsf{s}(0))) \\ \mathsf{odd}(x) &\gtrsim \mathsf{eo}(x, \mathsf{s}(0)) & \mathsf{eo}(0, \mathsf{s}(0)) &\gtrsim \mathsf{false} \\ \mathsf{not}(\mathsf{true}) &\gtrsim \mathsf{false} & \mathsf{eo}(\mathsf{s}(x), \mathsf{s}(0)) &\gtrsim \mathsf{eo}(x, 0) \\ \mathsf{not}(\mathsf{false}) &\gtrsim \mathsf{true} & \mathsf{EO}(x, 0) &> \mathsf{NOT}(\mathsf{eo}(x, \mathsf{s}(0))) \\ \mathsf{EVEN}(x) &> \mathsf{EO}(x, 0) & \mathsf{EO}(x, 0) &> \mathsf{EO}(x, \mathsf{s}(0)) \\ \mathsf{ODD}(x) &> \mathsf{EO}(x, \mathsf{s}(0)) & \mathsf{EO}(\mathsf{s}(x), \mathsf{s}(0)) &> \mathsf{EO}(x, 0) \end{aligned}$$

In order to benefit from the fact that the second component of a reduction pair need not be closed under contexts, the constraints generated by Theorem 3.2 may be simplified by applying a so-called *argument filtering*. An argument filtering for a signature $\mathcal{F}$ is a mapping π that associates with every n-ary function symbol an argument position $i \in \{1, \dots, n\}$ or a (possibly empty) list $[i_1, \dots, i_m]$ of argument positions with $1 \leqslant i_1 < \cdots < i_m \leqslant n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f)$ is some list $[i_1, \dots, i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$, also denoted by π:

$$\pi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \pi(t_i) & \text{if } t = f(t_1, \dots, t_n) \text{ and } \pi(f) = i \\ f(\pi(t_{i_1}), \dots, \pi(t_{i_m})) & \text{if } t = f(t_1, \dots, t_n) \text{ and } \pi(f) = [i_1, \dots, i_m] \end{cases}$$

Thus, an argument filtering is used to replace function symbols by one of their arguments or to eliminate certain arguments of function symbols.

Theorem 3.3 ([1]) *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if there exist an argument filtering π for $\mathcal{F}^\sharp$ and a reduction pair $(\gtrsim, >)$ such that $\pi(\mathcal{R}) \subseteq \gtrsim$ and $\pi(\mathsf{DP}(\mathcal{R})) \subseteq >$.*

Example 3.1 (continued) Even if we apply an arbitrary argument filtering π to the ordering constraints given earlier, the resulting constraints cannot be satisfied by a reduction pair $(\gtrsim, >)$ with $>$ a simplification order. This can be seen as follows. The constraint $\mathsf{EO}(x, 0) > \mathsf{EO}(x, \mathsf{s}(0))$ can only be satisfied when $\pi(\mathsf{s}) = [\,]$. But then the constraint $\mathsf{EO}(\mathsf{s}(x), \mathsf{s}(0)) > \mathsf{EO}(x, 0)$ reduces to $\mathsf{EO}(\mathsf{s}, \mathsf{s}) > \mathsf{EO}(x, 0)$ and hence we must have $\pi(\mathsf{EO}) \in \{[2], 2\}$. So the two constraints $\mathsf{EO}(x, 0) > \mathsf{EO}(x, \mathsf{s}(0))$ and $\mathsf{EO}(\mathsf{s}(x), \mathsf{s}(0)) > \mathsf{EO}(x, 0)$ reduce to either $\mathsf{EO}(0) > \mathsf{EO}(\mathsf{s})$ and $\mathsf{EO}(\mathsf{s}) > \mathsf{EO}(0)$ or $0 > \mathsf{s}$ and $\mathsf{s} > 0$, contradicting the well-foundedness of $>$.

Rather than considering all dependency pairs at the same time, like in the preceding theorems, it is advantageous to treat groups of dependency pairs separately. These groups correspond to cycles in the *dependency graph* $\mathsf{DG}(\mathcal{R})$ of $\mathcal{R}$. The nodes of $\mathsf{DG}(\mathcal{R})$ are the dependency pairs of $\mathcal{R}$ and there is an arrow from $s \to t$ to $u \to v$ if and only if there exist substitutions σ and τ such that $t\sigma \to_{\mathcal{R}}^* u\tau$. (By renaming variables in different occurrences of dependency pairs we may assume that $\sigma = \tau$.) A *cycle* is a non-empty subset $\mathcal{C}$ of dependency pairs if for every two (not necessarily distinct) pairs $s \to t$ and $u \to v$ in $\mathcal{C}$ there exists a non-empty path in $\mathcal{C}$ from $s \to t$ to $u \to v$.

Theorem 3.4 ([9]) *A TRS $\mathcal{R}$ is terminating if and only if for every cycle $\mathcal{C}$ in $\mathsf{DG}(\mathcal{R})$ there exist an argument filtering π and a reduction pair $(\gtrsim, >)$ such that $\pi(\mathcal{R}) \subseteq \gtrsim$, $\pi(\mathcal{C}) \subseteq \gtrsim \cup >$, and $\pi(\mathcal{C}) \cap > \neq \varnothing$.*

The last condition in Theorem 3.4 denotes the situation that $\pi(s) > \pi(t)$ for at least one dependency pair $s \to t \in \mathcal{C}$.

Example 3.1 (continued) The dependency graph $\mathsf{DG}(\mathcal{R})$ has six arrows:

$$\begin{tikzcd}
(1) \arrow[r] \arrow[d] & (3) & \\
(4) \arrow[r, bend left] & (5) \arrow[u] \arrow[l, bend left] & (2) \arrow[l]
\end{tikzcd}$$

The constraints corresponding to the only cycle $\mathcal{C} = \{(4), (5)\}$ consist of the rule constraints

$$\begin{array}{rlrl}
\mathsf{even}(x) & \gtrsim \mathsf{eo}(x, 0) & \mathsf{eo}(x, 0) & \gtrsim \mathsf{not}(\mathsf{eo}(x, \mathsf{s}(0))) \\
\mathsf{odd}(x) & \gtrsim \mathsf{eo}(x, \mathsf{s}(0)) & \mathsf{eo}(0, \mathsf{s}(0)) & \gtrsim \mathsf{false} \\
\mathsf{not}(\mathsf{true}) & \gtrsim \mathsf{false} & \mathsf{eo}(\mathsf{s}(x), \mathsf{s}(0)) & \gtrsim \mathsf{eo}(x, 0) \\
\mathsf{not}(\mathsf{false}) & \gtrsim \mathsf{true} & &
\end{array}$$

and the dependency pair constraints

$$\mathsf{EO}(x, 0) >' \mathsf{EO}(x, \mathsf{s}(0)) \qquad \mathsf{EO}(\mathsf{s}(x), \mathsf{s}(0)) >'' \mathsf{EO}(x, 0)$$

with $>', >'' \in \{>, \gtrsim\}$ such that at least one of $>'$ and $>''$ equals $>$. Letting $>' = \gtrsim$, $>'' = >$, and using the argument filtering π defined as $\pi(\mathsf{EO}) = 1$, $\pi(\mathsf{eo}) = \pi(\mathsf{not}) = \pi(0) = [\,]$, and $\pi(\mathsf{s}) = \pi(\mathsf{even}) = \pi(\mathsf{odd}) = [1]$, these constraints reduce to

$$\begin{array}{lllll}
\mathsf{even}(x) \gtrsim \mathsf{eo} & \mathsf{not} \gtrsim \mathsf{false} & \mathsf{eo} \gtrsim \mathsf{not} & \mathsf{eo} \gtrsim \mathsf{eo} & \mathsf{s}(x) > x \\
\mathsf{odd}(x) \gtrsim \mathsf{eo} & \mathsf{not} \gtrsim \mathsf{true} & \mathsf{eo} \gtrsim \mathsf{false} & x \gtrsim x &
\end{array}$$

and are satisfied by the recursive path order (i.e., $> \;=\; \succ_{\text{rpo}}$ and $\gtrsim \;=\; \succ_{\text{rpo}}^{=}$) with precedence $\mathsf{even}, \mathsf{odd} \succ \mathsf{eo} \succ \mathsf{not} \succ \mathsf{false}, \mathsf{true}$.

Since it is undecidable whether there exist substitutions σ, τ such that $t\sigma \rightarrow_{\mathcal{R}}^{*} u\tau$, the dependency graph cannot be computed in general. Hence, in order to mechanize the termination criterion of Theorem 3.4 one has to approximate the dependency graph. To this end, Arts and Giesl proposed a simple algorithm.

Definition 3.5 Let $\mathcal{R}$ be a TRS. The nodes of the *estimated* dependency graph $\mathsf{EDG}(\mathcal{R})$ are the dependency pairs of $\mathcal{R}$ and there is an arrow from $s \to t$ to $u \to v$ if and only if $\mathsf{REN}(\mathsf{CAP}(t))$ and u are unifiable. Here CAP replaces all outermost subterms with a defined root symbol by distinct fresh variables and REN replaces all occurrences of variables by distinct fresh variables.

Lemma 3.6 ([1]) *Let $\mathcal{R}$ be a TRS.*

(i) $\mathsf{EDG}(\mathcal{R})$ *is computable.*

(ii) $\mathsf{DG}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$.

One easily verifies that for the TRS $\mathcal{R}$ of Example 3.1, $\mathsf{EDG}(\mathcal{R})$ coincides with $\mathsf{DG}(\mathcal{R})$.

Example 3.7 Consider the TRS $\mathcal{R}$ consisting of the well-known rewrite rule of Toyama [27]:

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

There is one dependency pair: $\mathsf{F}(\mathsf{a},\mathsf{b},x) \to \mathsf{F}(x,x,x)$. As there are no terms s and t such that $\mathsf{F}(s,s,s) \to_{\mathcal{R}}^{*} \mathsf{F}(\mathsf{a},\mathsf{b},t)$, $\mathsf{DG}(\mathcal{R})$ contains no arrows and hence termination of $\mathcal{R}$ is a trivial consequence of Theorem 3.4. However, since $\mathsf{REN}(\mathsf{CAP}(\mathsf{F}(x,x,x))) = \mathsf{F}(x_1,x_2,x_3)$ unifies with $\mathsf{F}(\mathsf{a},\mathsf{b},x)$, $\mathsf{EDG}(\mathcal{R})$ contains a cycle. Since solving the resulting constraints $\mathsf{f}(\mathsf{a},\mathsf{b},x) \gtrsim \mathsf{f}(x,x,x)$ and $\mathsf{F}(\mathsf{a},\mathsf{b},x) > \mathsf{F}(x,x,x)$ is just as hard as proving the termination of $\mathcal{R}$ directly, *automatically* proving termination with the dependency pair method seems impossible.

The TRS in the above example is not *DP quasi-simply terminating.* The class of DP quasi-simply terminating TRSs (Giesl and Ohlebusch [10]) is supposed to "capture all TRSs where an automated termination proof using dependency pairs is potentially feasible". In Section 6 we will see that by replacing the estimated dependency graph by better (computable) approximations of the dependency graph, automatically proving termination of the TRS of Example 3.7 becomes trivial.

4 Approximations

We begin this section by recalling some basic definitions and results concerning tree automata. Much more information can be found in [4].

A (finite bottom-up) *tree automaton* is a quadruple $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ consisting of a finite signature $\mathcal{F}$, a finite set Q of states, disjoint from $\mathcal{F}$, a subset $Q_f \subseteq Q$ of final states, and a set of transition rules Δ. Every transition rule has the form $f(q_1,\dots,q_n) \to q$ with $f \in \mathcal{F}$ and $q_1,\dots,q_n,q \in Q$. So a tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ is simply a finite ground TRS $(\mathcal{F} \cup Q, \Delta)$ whose rewrite rules have a special shape, together with a subset Q_f of Q. The induced rewrite relation on $\mathcal{T}(\mathcal{F} \cup Q)$ is denoted by $\to_{\mathcal{A}}$. A ground term $t \in \mathcal{T}(\mathcal{F})$ is accepted by $\mathcal{A}$ if $t \to_{\mathcal{A}}^{*} q$ for some $q \in Q_f$. The set of all such terms is denoted by $L(\mathcal{A})$. A subset $L \subseteq \mathcal{T}(\mathcal{F})$ is called *regular* if there exists a tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ such that $L = L(\mathcal{A})$. It is well-known that the set of ground instances $\Sigma(t)$ of a linear term t is regular. Moreover, the set of ground normal forms $\mathsf{NF}(\mathcal{R})$ of a left-linear TRS $\mathcal{R}$ is regular. Below we make use of the well-known fact that it is decidable whether a given tree automaton accepts the empty language.

Let $\mathcal{R}$ be an eTRS over a signature $\mathcal{F}$ and $L \subseteq \mathcal{T}(\mathcal{F})$. In the following we denote the set of all terms $s \in \mathcal{T}(\mathcal{F})$ such that $s \to_{\mathcal{R}}^{*} t$ for some term $t \in L$ by $(\to_{\mathcal{R}}^{*})[L]$.

The reason for the undecidability of neededness and the fact that the dependency graph is not computable is simply that reduction $(\to_{\mathcal{R}}^{*})$ is undecidable. More precisely, it is undecidable whether a term rewrites to a term that belongs to a certain set. In the case of neededness, this set is the set of normal forms (that do not contain $\bullet$, see the next section for precise statements). Since $\mathsf{NF}(\mathcal{R})$ is regular for a left-linear TRS $\mathcal{R}$, it follows that the set $(\rightarrow_{\mathcal{R}}^*)[\mathsf{NF}(\mathcal{R})]$ is not regular. The key to decidability is to extend $\rightarrow_{\mathcal{R}}^*$ to $\rightarrow_{\mathcal{S}}^*$ for some suitable eTRS $\mathcal{S}$ such that $(\rightarrow_{\mathcal{S}}^*)[\mathsf{NF}(\mathcal{R})]$ becomes regular.

Definition 4.1 Let $\mathcal{R}$ and $\mathcal{S}$ be eTRSs over the same signature. We say that $\mathcal{S}$ *approximates* $\mathcal{R}$ if $\rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{S}}^*$ and $\mathsf{NF}(\mathcal{R}) = \mathsf{NF}(\mathcal{S})$. An *approximation mapping* is a mapping α from eTRSs to eTRSs with the property that $\alpha(\mathcal{R})$ approximates $\mathcal{R}$ for all eTRSs $\mathcal{R}$. We say that α is *regularity preserving* if $(\rightarrow_{\mathcal{R}_\alpha}^*)[L]$ is regular for all eTRSs $\mathcal{R}$ and regular L.

In the remainder of this section we define three approximation mappings that are known to be regularity preserving. Our definitions are slightly different from the ones found in the literature because we have to deal with possibly non-left-linear TRSs (when approximating the dependency graph in Section 6).

Definition 4.2 Let $\mathcal{R}$ be an eTRS. The *strong approximation* $\mathcal{R}_{\mathsf{s}}$ is obtained from $\mathcal{R}$ by replacing the right-hand side and all occurrences of variables in the left-hand side of every rewrite rule by distinct fresh variables, i.e., $\mathcal{R}_{\mathsf{s}} = \{\mathsf{REN}(l) \rightarrow z \mid l \rightarrow r \in \mathcal{R} \text{ and } z \text{ is a fresh variable}\}$.

Example 2.2 (continued) The eTRS $\mathcal{R}_{\mathsf{s}}$ consists of the following rules:

$$\begin{aligned} 0 + y &\rightarrow z & \qquad 0 \times y &\rightarrow z \\ \mathsf{s}(x) + y &\rightarrow z & \qquad \mathsf{s}(x) \times y &\rightarrow z \end{aligned}$$

The idea of approximating a TRS by ignoring the right-hand sides of its rewrite rules is due to Huet and Lévy [11]. A better approximation is obtained by preserving the *n*on-*v*ariable parts of the right-hand sides of the rewrite rules.

Definition 4.3 The *nv approximation* $\mathcal{R}_{\mathsf{nv}}$ is obtained from $\mathcal{R}$ by replacing all occurrences of variables in the rewrite rules by distinct fresh variables: $\mathcal{R}_{\mathsf{nv}} = \{\mathsf{REN}(l) \rightarrow \mathsf{REN}(r) \mid l \rightarrow r \in \mathcal{R}\}$.

Example 2.2 (continued) The eTRS $\mathcal{R}_{\mathsf{nv}}$ consists of the following rules:

$$\begin{aligned} 0 + y &\rightarrow y' & \qquad 0 \times y &\rightarrow 0 \\ \mathsf{s}(x) + y &\rightarrow \mathsf{s}(x' + y') & \qquad \mathsf{s}(x) \times y &\rightarrow (x' \times y') + y'' \end{aligned}$$

The idea of approximating a TRS by ignoring the variables in the right-hand sides of the rewrite rules is due to Oyamaguchi [22]. Note that $\mathcal{R}_{\mathsf{nv}} = \mathcal{R}$ whenever $\mathcal{R}$ is left-linear and right-ground.

Definition 4.4 An eTRS is called *growing* if for every rewrite rule $l \rightarrow r$ the variables in $\mathcal{V}\mathsf{ar}(l) \cap \mathcal{V}\mathsf{ar}(r)$ occur at depth 1 in l. The *growing approximation*

$\mathcal{R}_{\mathsf{g}}$ is defined as any left-linear growing eTRS that is obtained from $\mathcal{R}$ by linearizing the left-hand sides and renaming the variables in the right-hand sides that occur at a depth greater than 1 in the corresponding left-hand sides.

Example 2.2 (continued) The eTRS $\mathcal{R}_{\mathsf{g}}$ consists of the following rules:

$$\begin{array}{cc} 0 + y \to y & 0 \times y \to 0 \\ \mathsf{s}(x) + y \to \mathsf{s}(x' + y) & \mathsf{s}(x) \times y \to (x' \times y) + y \end{array}$$

Note that the occurrences of y in the right-hand sides of the rules of $\mathcal{R}$ are not renamed since they occur at depth 1 in the corresponding left-hand sides.

Growing TRSs, introduced by Jacquemard [12], are a proper extension of the shallow TRSs considered by Comon [3]. The growing approximation defined above stems from Nagaya and Toyama [19]. It extends the growing approximation in [12] in that the right-linearity requirement is dropped.

As a further example, consider a TRS $\mathcal{R}$ that contains the rewrite rule $\mathsf{f}(x, \mathsf{g}(x), y) \to \mathsf{f}(x, x, \mathsf{g}(y))$. Then $\mathcal{R}_{\mathsf{s}}$ contains $\mathsf{f}(x, \mathsf{g}(x'), y) \to z$, $\mathcal{R}_{\mathsf{nv}}$ contains $\mathsf{f}(x, \mathsf{g}(x'), y) \to \mathsf{f}(x'', x''', \mathsf{g}(y'))$, and $\mathcal{R}_{\mathsf{g}}$ contains $\mathsf{f}(x, \mathsf{g}(x'), y) \to \mathsf{f}(x, x, \mathsf{g}(y))$ or $\mathsf{f}(x', \mathsf{g}(x), y) \to \mathsf{f}(x'', x'', \mathsf{g}(y))$. The former is preferred as it is closer to the original rule. The ambiguity in the definition of $\mathcal{R}_{\mathsf{g}}$ causes no problems in the sequel.

Theorem 4.5 *The approximation mappings* s, nv, *and* g *are regularity preserving.*

Nagaya and Toyama [19] proved the above result for the growing approximation; the tree automaton that recognizes $(\to^*_{\mathcal{R}_{\mathsf{g}}})[L]$ is defined as the limit of a finite saturation process. This saturation process is similar to the ones defined in Comon [3] and Jacquemard [12], but by working exclusively with deterministic tree automata, non-right-linear rewrite rules can be handled. For the strong and nv approximation simpler constructions using ground tree transducers are possible ([6]).

Takai *et al.* [25] introduced the class of left-linear inverse finite path overlapping rewrite systems and showed that the preceding theorem is true for the corresponding approximation mapping. Growing rewrite systems constitute a proper subclass of the class of inverse finite path overlapping rewrite systems. Since the definition of this class is rather difficult, we do not consider the inverse finite path overlapping approximation here. We note however that our results easily extend. Another complicated regularity preserving approximation mapping can be extracted from the recent paper by Seki *et al.* [23].

5 Approximations for Strategies

Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. We assume the existence of a constant $\bullet$ not appearing in $\mathcal{F}$ and we view $\mathcal{R}$ as a TRS over the extended signature

$\mathcal{F}_\bullet = \mathcal{F} \cup \{\bullet\}$. So $\mathsf{NF}(\mathcal{R})$, the set of ground normal forms of $\mathcal{R}$, consists of all terms in $\mathcal{T}(\mathcal{F}_\bullet)$ that are in normal form. Let $\mathcal{R}_\bullet$ be the TRS $\mathcal{R} \cup \{\bullet \to \bullet\}$. Note that $\mathsf{NF}(\mathcal{R}_\bullet)$ coincides with $\mathsf{NF}(\mathcal{R}) \cap \mathcal{T}(\mathcal{F})$, the set of ground normal forms that do not contain the symbol $\bullet$. The following easy lemma gives an alternative definition of neededness, not depending on the notion of descendant.

Lemma 5.1 *Let $\mathcal{R}$ be an orthogonal TRS over a signature $\mathcal{F}$. Redex Δ in term $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is needed if and only if there is no term $t \in \mathsf{NF}(\mathcal{R}_\bullet)$ such that $C[\bullet] \to_{\mathcal{R}}^* t$.*

An immediate consequence of this lemma is the folklore result that only the position of a redex in a term is important for determining neededness. So if redex Δ in term $C[\Delta]$ is needed then so is redex Δ' in $C[\Delta']$.

Definition 5.2 Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and α an approximation mapping. We say that redex Δ in $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is *α-needed* if $C[\bullet] \notin (\to_{\mathcal{R}_\alpha}^*)[\mathsf{NF}(\mathcal{R}_\bullet)]$. The set of all such terms $C[\bullet]$ is denoted by $\mathsf{NEED}(\mathcal{R}_\alpha)$.

So redex Δ in $C[\Delta]$ is α-needed if and only if $C[\bullet] \in \mathsf{NEED}(\mathcal{R}_\alpha)$. In examples we abbreviate $\to_{\mathcal{R}_\alpha}$ to $\to_\alpha$.

Example 2.2 (continued) Let Δ_1 and Δ_2 be redexes and consider the term

$$t = (\underbrace{0 + \mathsf{s}(\Delta_1)}_{\Delta_3}) + \Delta_2$$

All three redexes are needed (since $\mathcal{R}$ is non-erasing). The following rewrite sequences show that Δ_1 and Δ_2 are not s-needed:

$$\begin{aligned}(0 + \mathsf{s}(\bullet)) + \Delta_2 &\to_{\mathsf{s}} 0 + \Delta_2 \to_{\mathsf{s}} 0\\ (0 + \mathsf{s}(\Delta_1)) + \bullet &\to_{\mathsf{s}} 0 + \bullet \to_{\mathsf{s}} 0\end{aligned}$$

Redex Δ_3 is s-needed since all s-reducts of $\bullet + \Delta_2$ are of the form $\bullet + t'$. For the nv approximation the situation is the same. Redexes Δ_1 and Δ_2 are not nv-needed—the above s-rewrite sequences are also nv-rewrite sequences—but Δ_3 is. With respect to the growing approximation, Δ_1 is not g-needed:

$$(0 + \mathsf{s}(\bullet)) + \Delta_2 \to_{\mathsf{g}} \mathsf{s}(\bullet) + \Delta_2 \to_{\mathsf{g}} \mathsf{s}(0 + \Delta_2) \to_{\mathsf{g}} \mathsf{s}(\Delta_2) \to_{\mathsf{g}}^* t'$$

for some normal form t' (which depends on redex Δ_2). However, Δ_2 is g-needed. The reason is that we cannot get rid of $\bullet$ in the term $(0 + \mathsf{s}(\Delta_1)) + \bullet$ since the second argument of $+$ is never erased by the rules in $\mathcal{R}_{\mathsf{g}}$.

Lemma 5.3 *Let $\mathcal{R}$ be an orthogonal TRS and α an approximation mapping. Every α-needed redex is needed.*

Only in Lemma 5.3 we require orthogonality. For decidability issues, left-linearity suffices.

Lemma 5.4 *Let $\mathcal{R}$ be a left-linear TRS and α an approximation mapping. If α is regularity preserving then $\mathsf{NEED}(\mathcal{R}_\alpha)$ is regular.*

Since membership for regular tree languages is decidable, we obtain the following result.

Corollary 5.5 *Let $\mathcal{R}$ be a left-linear TRS and α a regularity preserving approximation mapping. It is decidable whether a redex in a term is α-needed.*

Naturally, a better approximation can identify more needed redexes.

Lemma 5.6 $\mathsf{NEED}(\mathcal{R}_\mathsf{s}) \subseteq \mathsf{NEED}(\mathcal{R}_\mathsf{nv}) \subseteq \mathsf{NEED}(\mathcal{R}_\mathsf{g})$ *for every TRS $\mathcal{R}$.*

Lemma 5.3 and Corollary 5.5 take care of the first issue mentioned in the paragraph following Theorem 2.1, to find decidable approximations of neededness. In the following we address the second issue, to identify classes of TRSs with the property that every reducible term has a computable needed redex.

Definition 5.7 Let α be an approximation mapping. The class of TRSs such that every reducible ground term has an α-needed redex is denoted by CBN_α.

The proof of the following theorem relies on standard properties of regular tree languages and ground tree transducers.

Theorem 5.8 *Let $\mathcal{R}$ be a left-linear TRS and α a regularity preserving approximation map. It is decidable whether $\mathcal{R} \in \mathsf{CBN}_\alpha$.*

It should not come as a surprise that a better approximation covers a larger class of TRSs. This is expressed formally in the next lemma.

Lemma 5.9 $\mathsf{CBN}_\mathsf{s} \subsetneq \mathsf{CBN}_\mathsf{nv} \subsetneq \mathsf{CBN}_\mathsf{g}$.

It is interesting to note that the class CBN_s *properly* includes the class of strongly sequential TRSs introduced by Huet and Lévy (see [7, Example 1]). The class CBN_nv is much larger than the class of NV-sequential TRSs introduced by Oyamaguchi [22]. For instance, CBN_nv contains all right-ground TRSs. As a consequence, the proof that the first inclusion in the previous lemma is strict is very easy. The TRS consisting of the three rules

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

belongs to CBN_nv (as it is right-ground) but not to CBN_s since none of the occurrences of redex Δ in the term $\mathsf{f}(\Delta,\Delta,\Delta)$ is s-needed:

$$\begin{aligned}
\mathsf{f}(\bullet,\Delta,\Delta) &\to_\mathsf{s} \mathsf{f}(\bullet,\mathsf{a},\Delta) \to_\mathsf{s} \mathsf{f}(\bullet,\mathsf{a},\mathsf{b}) \to_\mathsf{s} \mathsf{a}\\
\mathsf{f}(\Delta,\bullet,\Delta) &\to_\mathsf{s} \mathsf{f}(\mathsf{b},\bullet,\Delta) \to_\mathsf{s} \mathsf{f}(\mathsf{b},\bullet,\mathsf{a}) \to_\mathsf{s} \mathsf{a}\\
\mathsf{f}(\Delta,\Delta,\bullet) &\to_\mathsf{s} \mathsf{f}(\mathsf{a},\Delta,\bullet) \to_\mathsf{s} \mathsf{f}(\mathsf{a},\mathsf{b},\bullet) \to_\mathsf{s} \mathsf{a}
\end{aligned}$$

In contrast, the proof that the class of NV-sequential TRSs properly includes all strongly sequential TRSs is rather complicated (cf. [22]). The relationships between several classes of TRSs that admit decidable call by need computations to normal form are summarized in Fig. 1. The interested reader is referred to Durand and Middeldorp [6,7] for further results on decidable call-by-need computations.

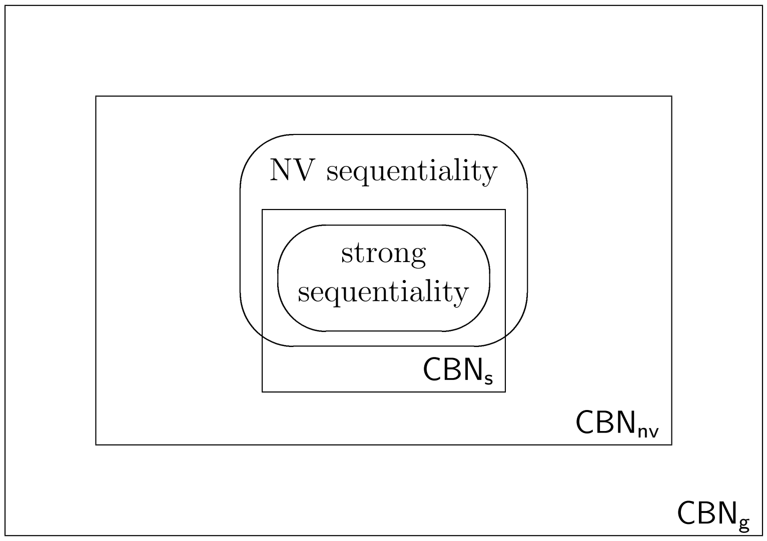

Fig. 1. Comparison.

6 Approximations for Termination

Recall that there exists an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}(\mathcal{R})$ if and only if $t\sigma \to_{\mathcal{R}}^* u\tau$ for some substitutions σ and τ. Since $t\sigma \in \Sigma(t)$ and $u\tau \in \Sigma(u)$, this is equivalent to

$$\Sigma(t) \cap (\to_{\mathcal{R}}^*)[\Sigma(u)] \neq \varnothing$$

If we want to use tree automata to decide the non-emptiness of the intersection of $\Sigma(t)$ and $(\to_{\mathcal{R}}^*)[\Sigma(u)]$, the obvious idea is to approximate these sets by regular tree languages that contain them. It is well-known that $\Sigma(u)$ need not be regular if u is a non-linear term. However, $\mathsf{REN}(u)$ is a linear term and the inclusion $\Sigma(u) \subseteq \Sigma(\mathsf{REN}(u))$ clearly holds. Based on the results of Section 4, it is natural to approximate $(\to_{\mathcal{R}}^*)[\Sigma(u)]$ by $(\to_{\mathcal{R}_\alpha}^*)[\Sigma(\mathsf{REN}(u))]$ for a regularity preserving approximation mapping α. However, there is no need to approximate $\Sigma(t)$ by $\Sigma(\mathsf{REN}(t))$ in order to obtain decidability. The reason is expressed in the following result.

Theorem 6.1 (Tison [26]) *The following problem is decidable:*

instance: *tree automaton $\mathcal{A}$, term t*

question: $\Sigma(t) \cap L(\mathcal{A}) = \varnothing$*?*

This result will turn out to be very important for automatically proving termination of TRSs that rely on non-linearity (i.e., by linearizing the rewrite rules the TRS becomes non-terminating).

For a proper understanding of the next definition, it is helpful to realize that $\Sigma(t) \cap (\to^*_{\mathcal{R}})[\Sigma(u)] \neq \varnothing$ is equivalent to $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})})[\Sigma(t)] \neq \varnothing$.

Definition 6.2 Let $\mathcal{R}$ be a TRS and α an approximation mapping. The nodes of the *α-approximated* dependency graph $\mathsf{DG}_\alpha(\mathcal{R})$ are the dependency pairs of $\mathcal{R}$ and there is an arrow from $s \to t$ to $u \to v$ if and only if both $\Sigma(t) \cap (\to^*_{\mathcal{R}_\alpha})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$ and $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})_\alpha})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$.

So we draw arrow from $s \to t$ to $u \to v$ if a ground instance of t rewrites in $\mathcal{R}_\alpha$ to a ground instance of $\mathsf{REN}(u)$ *and* a ground instance of u rewrites in $(\mathcal{R}^{-1})_\alpha$ to a ground instance of $\mathsf{REN}(t)$. The reason for having both conditions is that (1) for decidability t or u should be made linear and (2) depending on α and $\mathcal{R}$, $\mathcal{R}_\alpha$ may better approximate $\mathcal{R}$ than $(\mathcal{R}^{-1})_\alpha$ approximates $\mathcal{R}^{-1}$, or vice-versa. Also, the more conditions one imposes, the closer one gets to the real dependency graph.

Theorem 6.3 *Let $\mathcal{R}$ be a TRS and α an approximation mapping.*

(i) *If α is regularity preserving then $\mathsf{DG}_\alpha(\mathcal{R})$ is computable.*

(ii) $\mathsf{DG}(\mathcal{R}) \subseteq \mathsf{DG}_\alpha(\mathcal{R})$.

Naturally, a better approximation mapping results in a better approximation of the dependency graph. Hence we have the following result.

Lemma 6.4 $\mathsf{DG}(\mathcal{R}) \subseteq \mathsf{DG}_{\mathsf{g}}(\mathcal{R}) \subseteq \mathsf{DG}_{\mathsf{nv}}(\mathcal{R}) \subseteq \mathsf{DG}_{\mathsf{s}}(\mathcal{R})$ *for every TRS $\mathcal{R}$.*

We now compare our α-approximated dependency graph with the estimated dependency graph of Arts and Giesl. The first two examples show that the s-approximated dependency graph and the estimated dependency graph are incomparable in general.

Example 6.5 Consider the TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{g}(\mathsf{a})) \to \mathsf{f}(\mathsf{a}), \mathsf{a} \to \mathsf{b}\}$. There are two dependency pairs:

$$\mathsf{F}(\mathsf{g}(\mathsf{a})) \to \mathsf{F}(\mathsf{a}) \quad (1) \qquad\qquad \mathsf{F}(\mathsf{g}(\mathsf{a})) \to \mathsf{A} \quad (2)$$

Because $\mathsf{REN}(\mathsf{CAP}(\mathsf{F}(\mathsf{a}))) = \mathsf{F}(x)$ unifies with $\mathsf{F}(\mathsf{g}(\mathsf{a}))$, $\mathsf{EDG}(\mathcal{R})$ contains two arrows:

$$\begin{tikzcd} (1) \arrow[loop left] \arrow[r] & (2) \end{tikzcd}$$

We have $(\mathcal{R}^{-1})_{\mathsf{s}} = \{\mathsf{f}(\mathsf{a}) \to x, \mathsf{b} \to x\}$. Hence $(\to^*_{(\mathcal{R}^{-1})_{\mathsf{s}}})[\{\mathsf{F}(\mathsf{a})\}]$ consists of all terms of the form $\mathsf{f}^n(\mathsf{a})$, $\mathsf{f}^n(\mathsf{b})$, $\mathsf{F}(\mathsf{f}^n(\mathsf{a}))$, $\mathsf{F}(\mathsf{f}^n(\mathsf{b}))$ with $n \geqslant 0$. The term $\mathsf{F}(\mathsf{g}(\mathsf{a}))$ clearly does not belong to this set and hence there are no arrows in $\mathsf{DG}_{\mathsf{s}}(\mathcal{R})$.

Example 6.6 Consider the TRS $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{f}(\mathsf{a},\mathsf{b})\}$. We have $\mathsf{DP}(\mathcal{R}) = \{\mathsf{F}(x,x) \to \mathsf{F}(\mathsf{a},\mathsf{b})\}$. Because $\mathsf{REN}(\mathsf{CAP}(\mathsf{F}(\mathsf{a},\mathsf{b}))) = \mathsf{F}(\mathsf{a},\mathsf{b})$ and $\mathsf{F}(x,x)$ are not unifiable, $\mathsf{EDG}(\mathcal{R})$ contains no arrows. However, both $\Sigma(\mathsf{F}(\mathsf{a},\mathsf{b})) \cap (\to^*_{\mathcal{R}_{\mathsf{s}}})$

$[\Sigma(\mathsf{REN}(\mathsf{F}(x,x)))]$ and $\Sigma(\mathsf{F}(x,x)) \cap (\rightarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b})))]$ are non-empty, as witnessed by the terms $\mathsf{F}(\mathsf{a},\mathsf{b})$ and $\mathsf{F}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{f}(\mathsf{a},\mathsf{b}))$.

The non-linearity in the preceding example is essential.

Lemma 6.7 $\mathsf{DG}_\mathsf{s}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$ *for every left-linear TRS* $\mathcal{R}$.

The next result states that the nv-approximated dependency graph is always a subgraph of the estimated dependency graph.

Theorem 6.8 $\mathsf{DG}_\mathsf{nv}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$ *for every TRS* $\mathcal{R}$.

The next example shows that the nv-approximated dependency graph is in general a proper subgraph of the estimated dependency graph.

Example 3.7 (continued) We have $\mathcal{R}_\mathsf{nv} = \{\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{f}(x_1,x_2,x_3)\}$ and $\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x))) = \{\mathsf{F}(\mathsf{a},\mathsf{b},t) \mid t \text{ is a ground term}\}$. Consequently, $(\rightarrow^*_{\mathcal{R}_\mathsf{nv}})$ $[\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x)))]$ equals $\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x)))$ and since no instance of the term $\mathsf{F}(x,x,x)$ belongs to this set, $\mathsf{DG}_\mathsf{nv}(\mathcal{R})$ contains no arrow.

We note that the various refinements of the dependency pair method (narrowing, rewriting, instantiation; see Giesl and Arts [8]) are not applicable to the TRS of Example 3.7.

The next example shows a TRS that cannot be proved terminating with the nv approximation but whose (automatic) termination proof becomes easy with the growing approximation.

Example 6.9 Consider the TRS $\mathcal{R}$ consisting of the three rewrite rules

$$\mathsf{f}(x,\mathsf{a}) \to \mathsf{f}(x,\mathsf{g}(x,\mathsf{b})) \qquad \begin{aligned} \mathsf{g}(\mathsf{h}(x),y) &\to \mathsf{g}(x,\mathsf{h}(y)) \\ \mathsf{g}(\mathsf{a},y) &\to y \end{aligned}$$

There are three dependency pairs:

$$\begin{aligned} \mathsf{F}(x,\mathsf{a}) &\to \mathsf{F}(x,\mathsf{g}(x,\mathsf{b})) \quad (1) \qquad & \mathsf{G}(\mathsf{h}(x),y) &\to \mathsf{G}(x,\mathsf{h}(y)) \quad (3) \\ \mathsf{F}(x,\mathsf{a}) &\to \mathsf{G}(x,\mathsf{b}) \quad (2) & & \end{aligned}$$

One easily verifies that $\mathsf{DG}_\mathsf{nv}(\mathcal{R})$ contains two cycles:

$$\begin{tikzcd} (1) \arrow[loop left] \arrow[r] & (2) \arrow[r] & (3) \arrow[loop right] \end{tikzcd}$$

In particular, $\mathsf{F}(\mathsf{a},\mathsf{g}(\mathsf{a},\mathsf{b})) \rightarrow_{\mathcal{R}_\mathsf{nv}} \mathsf{F}(\mathsf{a},\mathsf{a})$ which explains the arrows from (1) to (1) and (2). The problematic cycle $\{(1)\}$ does not exist in $\mathsf{DG}_\mathsf{g}(\mathcal{R})$ because no ground instance of $\mathsf{F}(x,\mathsf{g}(x,\mathsf{b}))$ rewrites in $\mathcal{R}_\mathsf{g}$ to a ground instance of $\mathsf{F}(x,\mathsf{a})$:

$$\begin{tikzcd} (1) & (2) \arrow[r] & (3) \arrow[loop right] \end{tikzcd}$$

As a consequence, the resulting ordering constraints

$$\begin{aligned} \mathsf{f}(x,\mathsf{a}) &\gtrsim \mathsf{f}(x,\mathsf{g}(x,\mathsf{b})) \qquad & \mathsf{g}(\mathsf{a},y) &\gtrsim y \\ \mathsf{g}(\mathsf{h}(x),y) &\gtrsim \mathsf{g}(x,\mathsf{h}(y)) & \mathsf{G}(\mathsf{h}(x),y) &> \mathsf{G}(x,\mathsf{h}(y)) \end{aligned}$$

are easily satisfied (e.g. by taking $\pi(\mathsf{f}) = 1$ in combination with the lexicographic path order with precedence $\mathsf{G} \succ \mathsf{h}$ and $\mathsf{g} \succ \mathsf{h}$). [5]

For a comparison of our α-approximated dependency graph with the approximation of the dependency graph defined by Kusakari and Toyama [16,17] we refer to [18].

We conclude this paper by introducing a new approximation of the dependency graph, which does not rely on tree automata techniques. Recall that there are two conditions for the existence of an arrow between two dependency pairs in the α-approximated dependency graph. The idea is now to incorporate the symmetry considerations that gave rise to the second conditions into the estimated dependency graph of Arts of Giesl.

Definition 6.10 Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The result of replacing all outermost subterms of a term t with a root symbol in $\mathcal{D}^{-1}$ is denoted by $\mathsf{CAP}^{-1}(t)$. Here

$$\mathcal{D}^{-1} = \begin{cases} \{\mathsf{root}(r) \mid l \to r \in \mathcal{R}\} & \text{if } \mathcal{R} \text{ is non-collapsing} \\ \mathcal{F} & \text{otherwise} \end{cases}$$

is the set of defined symbols of the eTRS $\mathcal{R}^{-1}$. The nodes of the *estimated** dependency graph $\mathsf{EDG}^*(\mathcal{R})$ are the dependency pairs of $\mathcal{R}$ and there is an arrow from $s \to t$ to $u \to v$ if and only if both $\mathsf{REN}(\mathsf{CAP}(t))$ and u are unifiable, and t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ are unifiable.

Lemma 6.11 *Let $\mathcal{R}$ be a TRS.*

(i) $\mathsf{EDG}^*(\mathcal{R})$ *is computable.*

(ii) $\mathsf{DG}(\mathcal{R}) \subseteq \mathsf{EDG}^*(\mathcal{R})$.

Proof. The first part is obvious. The second part is an immediate consequence of Lemma 6.4 and Theorem 6.17 below. □

The next two examples show the advantage of the new approximation over EDG.

Example 6.5 (continued) We have $\mathsf{REN}(\mathsf{CAP}^{-1}(\mathsf{F}(\mathsf{g}(\mathsf{a}))) = \mathsf{F}(\mathsf{g}(\mathsf{a}))$ since $\mathcal{D}^{-1} = \{\mathsf{b}, \mathsf{f}\}$. Because this term does not unify with $\mathsf{F}(\mathsf{a})$, $\mathsf{EDG}^*(\mathcal{R})$ contains no arrows.

Example 3.7 (continued) Since $\mathsf{REN}(\mathsf{CAP}^{-1}(\mathsf{F}(\mathsf{a}, \mathsf{b}, x)) = \mathsf{F}(\mathsf{a}, \mathsf{b}, x_1)$ does not unify with $\mathsf{F}(x, x, x)$, $\mathsf{EDG}^*(\mathcal{R})$ contains no arrow.

As soon as $\mathcal{R}$ contains a rewrite rule with a variable right-hand side, $\mathsf{EDG}^*(\mathcal{R})$ and $\mathsf{EDG}(\mathcal{R})$ coincide.

Lemma 6.12 $\mathsf{EDG}^*(\mathcal{R}) = \mathsf{EDG}(\mathcal{R})$ *for every collapsing TRS $\mathcal{R}$.*

[5] Again, the TRS is not DP quasi-simply terminating.

Proof. Clearly $\mathsf{EDG}^*(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$. Suppose there is an arrow from $s \to t$ to $u \to v$ in $\mathsf{EDG}(\mathcal{R})$. So $\mathsf{REN}(\mathsf{CAP}(t))$ and u are unifiable. We need to show that t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ are unifiable. Let $\mathcal{F}$ be the signature of $\mathcal{R}$. We have $\mathsf{root}(t) = \mathsf{root}(u) \in \mathcal{F}^\sharp$. Let $u = f^\sharp(u_1, \ldots, u_n)$. Because $\mathcal{R}$ is collapsing, $\mathcal{D}^{-1} = \mathcal{F}$. As $u_1, \ldots, u_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $\mathsf{REN}(\mathsf{CAP}^{-1}(u)) = f^\sharp(x_1, \ldots, x_n)$ for some fresh variables $x_1, \ldots, x_n$. It follows that $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ unifies with any term that has $f^\sharp$ as root symbol, in particular with t. ☐

As far as the comparison with $\mathsf{DG_s}$ and $\mathsf{DG_{nv}}$ is concerned, the results established for EDG carry over to EDG^*, except that we have to strengthen left-linearity to linearity in Lemma 6.7. This follows from the next two examples and the subsequent results.

Example 6.13 Consider the TRS $\mathcal{R}$ consisting of the two rewrite rules

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

There are two dependency pairs:

$$\mathsf{F}(\mathsf{g}(\mathsf{a})) \to \mathsf{F}(\mathsf{a}) \quad (1) \qquad\qquad \mathsf{F}(\mathsf{g}(\mathsf{a})) \to \mathsf{A} \quad (2)$$

Because $\mathsf{REN}(\mathsf{CAP}(\mathsf{F}(\mathsf{a}))) = \mathsf{F}(x)$ unifies with $\mathsf{F}(\mathsf{g}(\mathsf{a}))$ and $\mathsf{F}(\mathsf{a})$ unifies with $\mathsf{REN}(\mathsf{CAP}^{-1}(\mathsf{F}(\mathsf{g}(\mathsf{a})))) = \mathsf{F}(x)$, $\mathsf{EDG}^*(\mathcal{R})$ contains two arrows:

$$\begin{tikzcd}(1) \arrow[loop left] \arrow[r] & (2)\end{tikzcd}$$

However, since $\mathsf{F}(\mathsf{g}(\mathsf{a})) \notin (\to^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\{\mathsf{F}(\mathsf{a})\}]$, $\mathsf{DG_s}(\mathcal{R})$ contains no arrows.

Example 6.14 Consider the TRS $\mathcal{R}$ consisting of the two rewrite rules

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

There is one dependency pair: $\mathsf{F}(\mathsf{a}, \mathsf{b}, x) \to \mathsf{F}(x, x, x)$. Because $\mathsf{F}(\mathsf{a}, \mathsf{b}, x)$ unifies with $\mathsf{REN}(\mathsf{CAP}(\mathsf{F}(x, x, x))) = \mathsf{F}(x_1, x_2, x_3)$ and $\mathsf{F}(x, x, x)$ unifies with $\mathsf{REN}(\mathsf{CAP}^{-1}(\mathsf{F}(\mathsf{a}, \mathsf{b}, x)) = \mathsf{F}(x_1, \mathsf{b}, x_2)$, $\mathsf{EDG}^*(\mathcal{R})$ contains a cycle. One easily verifies that $\mathsf{DG_{nv}}(\mathcal{R})$ contains no arrows.

Lemma 6.15 $(\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(t))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(t)))$ *for every non-collapsing TRS* $\mathcal{R}$ *and term* t.

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R}$. We use induction on the structure of t. If t is a variable or if $\mathsf{root}(t) \in \mathcal{D}^{-1}$ then $\mathsf{CAP}^{-1}(t)$ is a variable and hence $\Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(t))) = \mathcal{T}(\mathcal{F}^\sharp)$ and thus trivially $(\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(t))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(t)))$. Suppose $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{F}^\sharp \setminus \mathcal{D}^{-1}$. Since $\mathcal{R}$ is non-collapsing, every right-hand side of a rule in $\mathcal{R}$ and thus also every left-hand side of a rule in $(\mathcal{R}^{-1})_\mathsf{s}$ starts with a symbol in $\mathcal{D}^{-1}$. Because $\mathsf{root}(t) \notin \mathcal{D}^{-1}$ and the arguments of $\mathsf{REN}(t)$ do not share variables, it follows that $(\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(t))] = \{f(s_1, \ldots, s_n) \mid s_i \in (\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(t_i))]\}$. Clearly

$\Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(t))) = \{f(s_1,\ldots,s_n) \mid s_i \in \Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(t_i)))\}$. Hence the desired inclusion follows from the induction hypothesis. □

Lemma 6.16 $\mathsf{DG_s}(\mathcal{R}) \subseteq \mathsf{EDG}^*(\mathcal{R})$ *for every linear TRS* $\mathcal{R}$.

Proof. If $\mathcal{R}$ is collapsing then the result follows from Lemmata 6.7 and 6.12. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG_s}(\mathcal{R})$. Since $\mathsf{DG_s}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$ by Lemma 6.7, $\mathsf{REN}(\mathsf{CAP}(t))$ and u are unifiable by the definition of $\mathsf{EDG}(\mathcal{R})$. So it remains to show that t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ are unifiable. We have $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$ by the definition of $\mathsf{DG_s}(\mathcal{R})$. Since t and u are linear terms, $\Sigma(\mathsf{REN}(t)) = \Sigma(t)$ and $\Sigma(u) = \Sigma(\mathsf{REN}(u))$. It follows that $(\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(u))] \cap \Sigma(t) \neq \varnothing$ and thus $\Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(u))) \cap \Sigma(t) \neq \varnothing$ by Lemma 6.15. Since t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ do not share variables, they are indeed unifiable. □

Theorem 6.17 $\mathsf{DG_{nv}}(\mathcal{R}) \subseteq \mathsf{EDG}^*(\mathcal{R})$ *for every TRS* $\mathcal{R}$.

Proof. As in the preceding proof, we may assume that $\mathcal{R}$ is non-collapsing. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG_{nv}}(\mathcal{R})$. Since $\mathsf{DG_{nv}}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$ by Lemma 6.8, $\mathsf{REN}(\mathsf{CAP}(t))$ and u are unifiable by the definition of $\mathsf{EDG}(\mathcal{R})$. So it remains to show that t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ are unifiable. By definition, $\Sigma(t) \cap (\to^*_{\mathcal{R}_\mathsf{nv}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. Since $(\mathcal{R}_\mathsf{nv})^{-1} = (\mathcal{R}^{-1})_\mathsf{nv}$, $(\to^*_{\mathcal{R}_\mathsf{nv}})[\Sigma(\mathsf{REN}(u))] = (\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{nv}})[\Sigma(\mathsf{REN}(u))]$. Using Lemma 6.15 and the observation that $\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{nv}}$ is a subrelation of $\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}}$, it follows that $(\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{nv}})[\Sigma(\mathsf{REN}(u))] \subseteq (\leftarrow^*_{(\mathcal{R}^{-1})_\mathsf{s}})[\Sigma(\mathsf{REN}(u))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(u)))$. Hence $\Sigma(t) \cap \Sigma(\mathsf{REN}(\mathsf{CAP}^{-1}(u))) \neq \varnothing$ and thus t and $\mathsf{REN}(\mathsf{CAP}^{-1}(u))$ are unifiable. □

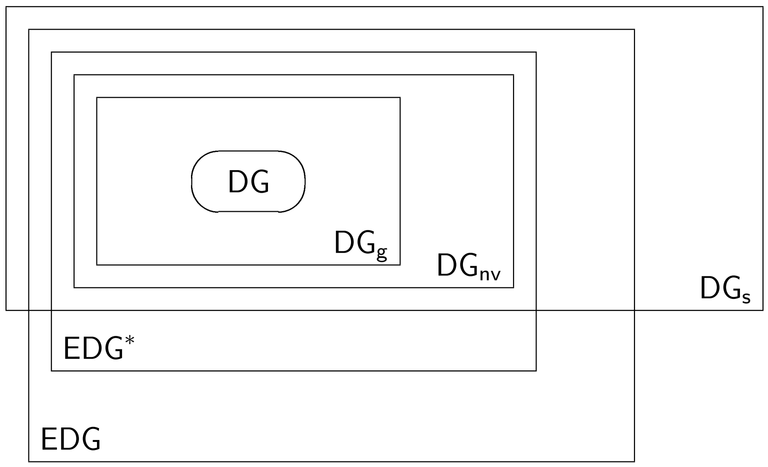

Fig. 2. Comparison.

Fig. 2 reveals how far the various approximations of the dependency graph are removed from the real object.

Acknowledgement

I am grateful to Nao Hirokawa for spotting an embarrassing mistake in an earlier version of the paper.

References

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

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

[3] Comon, H., *Sequentiality, monadic second-order logic and tree automata*, Information and Computation **157** (2000), pp. 25–51.

[4] Comon, H., M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, *Tree automata techniques and applications* (1999), draft, available from `http://www.grappa.univ-lille3.fr/tata/`.

[5] Dershowitz, N., *Termination of rewriting*, Journal of Symbolic Computation **3** (1987), pp. 69–116.

[6] Durand, I. and A. Middeldorp, *Decidable call by need computations in term rewriting (extended abstract)*, in: *Proc. 14th CADE*, LNAI **1249**, 1997, pp. 4–18.

[7] Durand, I. and A. Middeldorp, *On the modularity of deciding call-by-need*, in: *Proc. FoSSaCS*, LNCS **2030**, 2001, pp. 199–213.

[8] Giesl, J. and T. Arts, *Verification of Erlang processes by dependency pairs*, Applicable Algebra in Engineering, Communication and Computing **12** (2001), pp. 39–72.

[9] Giesl, J., T. Arts and E. Ohlebusch, *Modular termination proofs for rewriting using dependency pairs*, Journal of Symbolic Computation **34** (2002), pp. 21–58.

[10] Giesl, J. and E. Ohlebusch, *Pushing the frontiers of combining rewrite systems farther outwards*, in: *Proc. FroCoS'98*, Studies in Logic and Computation **7**, Wiley, 2000 pp. 141–160.

[11] Huet, G. and J.-J. Lévy, *Computations in orthogonal rewriting systems, I and II*, in: *Computational Logic, Essays in Honor of Alan Robinson*, The MIT Press, 1991, pp. 396–443.

[12] Jacquemard, F., *Decidable approximations of term rewriting systems*, in: *Proc. 7th RTA*, LNCS **1103**, 1996, pp. 362–376.

[13] Jouannaud, J.-P. and W. Sadfi, *Strong sequentiality of left-linear overlapping rewrite systems*, in: *Proc. 4th CTRS*, LNCS **968**, 1995, pp. 235–246.

[14] Klop, J., *Term rewriting systems*, in: *Handbook of Logic in Computer Science, Vol. 2*, Oxford University Press, 1992 pp. 1–116.

[15] Klop, J. and A. Middeldorp, *Sequentiality in orthogonal term rewriting systems*, Journal of Symbolic Computation **12** (1991), pp. 161–195.

[16] Kusakari, K., "Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems," Ph.D. thesis, JAIST (2000).

[17] Kusakari, K. and Y. Toyama, *On proving AC-termination by AC-dependency pairs*, Research Report IS-RR-98-0026F, School of Information Science, JAIST (1998).

[18] Middeldorp, A., *Approximating dependency graphs using tree automata techniques*, in: *Proc. IJCAR*, LNAI **2083**, 2001, pp. 593–610.

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

[20] O'Donnell, M., "Computing in Systems Described by Equations," LNCS **58**, 1977.

[21] Ohlebusch, E., *Hierarchical termination revisited*, Information Processing Letters **84** (2002), pp. 207–214.

[22] Oyamaguchi, M., *NV-sequentiality: A decidable condition for call-by-need computations in term rewriting systems*, SIAM Journal on Computation **22** (1993), pp. 114–135.

[23] Seki, H., T. Takai, Y. Fujinaka and Y. Kaji, *Layered transducing term rewriting system and its recognizability preserving property*, in: *Proc. 13th RTA*, LNCS **2378**, 2002, pp. 98–113.

[24] Steinbach, J., *Simplification orderings: History of results*, Fundamenta Informaticae **24** (1995), pp. 47–87.

[25] Takai, T., Y. Kaji and H. Seki, *Right-linear finite path overlapping term rewriting systems effectively preserve recognizability*, in: *Proc. 11th RTA*, LNCS **1833**, 2000, pp. 246–260.

[26] Tison, S., *Tree automata and term rewrite systems* (2000), invited tutorial at the 11th RTA.

[27] Toyama, Y., *Counterexamples to the termination for the direct sum of term rewriting systems*, Information Processing Letters **25** (1987), pp. 141–143.

[28] Toyama, Y., *Strong sequentiality of left-linear overlapping term rewriting systems*, in: *Proc. 7th LICS*, 1992, pp. 274–284.

[29] Urbain, X., *Automated incremental termination proofs for hierarchically defined term rewriting systems*, in: *Proc. IJCAR*, LNAI **2083**, 2001, pp. 485–498.

[30] Zantema, H., *Termination*, in: TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003 (to appear).