Approximating Dependency Graphs using Tree Automata Techniques*

Aart Middeldorp

Institute of Information Sciences and Electronics
University of Tsukuba, Tsukuba 305-8573, Japan
ami@is.tsukuba.ac.jp

Abstract. The dependency pair method of Arts and Giesl is the most powerful technique for proving termination of term rewrite systems automatically. We show that the method can be improved by using tree automata techniques to obtain better approximations of the dependency graph. This graph determines the ordering constraints that need to be solved in order to conclude termination. We further show that by using our approximations the dependency pair method provides a decision procedure for termination of right-ground rewrite systems.

1 Introduction

In the area of term rewriting termination has been studied for several decades and many powerful techniques have been developed. Three general directions can be distinguished:

1. Syntactic methods that compare terms by constructing an explicit well-founded order. These methods are fully automatable but have limited power. Well-known examples are the recursive path order of Dershowitz [10] and the Knuth-Bendix order [18].
2. Semantic methods that compare terms by interpreting them in some well-founded domain. These methods can be very powerful in theory but their implementations rely on heuristics that greatly reduce this power. Well-known examples are the polynomial interpretations of Lankford [22] and the semantic path order of Kamin and Lévy [17].
3. Transformation methods which do not attempt to prove termination directly but rather transform the given rewrite system into another rewrite system such that termination of the latter system is easier to prove and implies termination of the former system. Examples include the transformation order of Bellegarde and Lescanne [6], and Zantema's distribution elimination [27] and semantic labelling [28]. Transformations differ in their degree of automation.

Since termination is an undecidable property of rewrite systems, even for systems that consist of a single rewrite rule, no method will work in all cases. Recently a new *automatable* technique emerged: the dependency pair method of Arts and Giesl. In this method a rewrite system is transformed into a set of ordering

* Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'01), Siena, Lecture Notes in Articial Intelligence 2083, pp. 593–610, 2001.

constraints such that termination of the rewrite system is equivalent to the solvability of the constraints. The generated constraints are typically solved by standard techniques (polynomial interpretations, path orders), even when these techniques are not applicable to the original rewrite system. The power of the dependency pair method has been amply illustrated in a sequence of papers by Arts and Giesl [2–4].

The ordering constraints in the dependency pair method are generated by analyzing the cycles in the *dependency graph*. This graph summarizes the relationships between the dependency pairs of the rewrite system. More precisely, there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in the dependency graph if some instance of t rewrites to some instance of u. Since this is undecidable in general, the dependency graph has to be estimated by a decidable approximation. Arts and Giesl proposed the following simple algorithm for this purpose: there is an arrow in the so-called estimated dependency graph from $s \to t$ to $u \to v$ if the term obtained from t by replacing all outermost defined symbols by variables and a subsequent linearization unifies with u.

The approximation of Arts and Giesl often results in an unnecessarily large graph and hence a large number of constraints. Sometimes, as examples in this paper will demonstrate, this causes the failure of the termination proof. The aim of this paper is to show that by using tree automata techniques we obtain a much better estimation of the dependency graph. Our approach is based on the following two ingredients:

1. The powerful framework of Durand and Middeldorp for the study of decidable call-by-need computations in orthogonal term rewriting. This framework is parameterized by so-called approximation mappings. An approximation mapping abstracts from certain parts of the terms in the rewrite rules such that the set of terms that rewrite to a term in an arbitrary regular tree language is again regular.
2. The folklore result that it is decidable whether the set of ground instances of an arbitrary term intersects with a regular tree language. This result is well-known for linear terms but it also holds for non-linear terms.

We show that by adopting the so-called *nv* approximation we always obtain an estimation of the dependency graph which is at least as good as the one of Arts and Giesl but often better. Interestingly, we can automatically prove termination of rewrite systems outside the class of so-called DP quasi-simply terminating systems. This class, proposed by Giesl and Ohlebusch [14], consists of all rewrite systems "where an automated termination proof using dependency pairs is potentially feasible".

The remainder of the paper is organized as follows. In the next section we briefly recall the basics of the dependency pair technique. Section 3 contains background material on tree automata. In Section 4 we define new approximations of the dependency graph. We compare our approximations with the one of Arts and Giesl in Section 5. We also include a comparison with the approximation of Kusakari and Toyama [19, 21]. In Section 6 we show that by using our approximations the dependency pair method provides a decision procedure for termination of right-ground rewrite systems.

2 Dependency Pairs

We assume familiarity with the basics of term rewriting ([5]). A term rewrite system (TRS for short) consists of rewrite rules $l \to r$ that 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 this paper we will make clear that such eTRSs are very useful for automatically proving termination of TRSs.

Below we recall the basic notions and results of the dependency pair technique of Arts and Giesl. We refer to [2, 4, 12] for motivations and further refinements. We adopt the notation of [13, 20]. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. As usual, root symbols of left-hand sides of rewrite rules are called defined. Let $\mathcal{F}^\sharp$ denote the union of $\mathcal{F}$ and $\{f^\sharp \mid f \text{ is a defined symbol of } \mathcal{R}\}$ where $f^\sharp$ has the same arity as f. Given a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with f defined, we write $t^\sharp$ for the term $f^\sharp(t_1, \ldots, t_n)$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol 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 often write F for $\mathsf{f}^\sharp$.

An argument filtering for a signature $\mathcal{F}$ is a mapping π that associates with every n-ary function symbol an argument position $i \in \{1, \ldots, n\}$ or a (possibly empty) list $[i_1, \ldots, i_m]$ 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 $[i_1, \ldots, i_m]$, 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:

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

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

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. The following theorem presents the basic dependency pair approach.

Theorem 1 (Arts and Giesl [4]). *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if there exists an argument filtering π for $\mathcal{F}^\sharp$ and a reduction pair $(\gtrsim, >)$ such that $\pi(\mathcal{R}) \subseteq \gtrsim$ and $\pi(\mathsf{DP}(\mathcal{R})) \subseteq >$.* □

Because rewrite rules are just pairs of terms, $\pi(\mathcal{R}) \subseteq \gtrsim$ is a shorthand for $\pi(l) \gtrsim \pi(r)$ for every rewrite rule $l \to r \in \mathcal{R}$. From now on we assume that all (e)TRSs are finite.

Rather than considering all dependency pairs at the same time, like in the above theorem, 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 of $\mathsf{DP}(\mathcal{R})$ 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 2 (Arts and Giesl [2])**.** *A TRS $\mathcal{R}$ is terminating if and only if for every cycle $\mathcal{C}$ in $\mathsf{DG}(\mathcal{R})$ there exists an argument filtering π and a reduction pair $(\gtrsim, >)$ such that $\pi(\mathcal{R}) \subseteq \gtrsim$, $\pi(\mathcal{C}) \subseteq \gtrsim \cup >$, and $\pi(\mathcal{C}) \cap > \neq \varnothing$.* $\square$

Note that $\pi(\mathcal{C}) \cap > \neq \varnothing$ denotes the situation that $\pi(s) > \pi(t)$ for at least one dependency pair $s \to t \in \mathcal{C}$.

Since it is undecidable whether there exists substitutions σ, τ such that $t\sigma \to_{\mathcal{R}}^{*} u\tau$, the dependency graph cannot be computed in general. Hence, in order to mechanize the termination criterion of Theorem 2 one has to approximate the dependency graph. To this end, Arts and Giesl proposed a simple algorithm.

Definition 3. *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 4 (Arts and Giesl [4])**.** *Let $\mathcal{R}$ be a TRS.*

1. $\mathsf{EDG}(\mathcal{R})$ *is computable.*
2. $\mathsf{DG}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$. $\square$

3 Tree Automata

We briefly recall some basic definitions and results concerning tree automata. Much more information can be found in [8].

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 $f(q_1, \ldots, q_n) \to q$ with $f \in \mathcal{F}$ and $q_1, \ldots, q_n, q \in Q$. 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 every regular language is accepted by a deterministic tree automaton without inaccessible states. A deterministic automaton has no two different rules with the same left-hand side. A state is inaccessible if no ground term reduces to it. In this paper we make use of the additional properties mentioned below.

Lemma 5. *The set of ground instances of a linear term is regular.* □

We write $\Sigma(t)$ for the set of ground instances of the term t. The next result states that it is decidable whether a ground instance of an arbitrary term is accepted by a given tree automaton. For a linear term t this is obvious since (1) $\Sigma(t)$ is regular by Lemma 5, (2) regular languages are effectively closed under intersection, and (3) emptiness is decidable for regular languages. The point is that the problem remains decidable for non-linear terms. This extension 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).

Theorem 6 (Tison [26])**.** *The following problem is decidable:*

instance: *tree automaton* $\mathcal{A}$, *term* t
question: $\Sigma(t) \cap L(\mathcal{A}) = \varnothing$?

Proof. First we transform $\mathcal{A}$ into an equivalent deterministic tree automaton $\mathcal{B} = (\mathcal{F}, Q, Q_f, \Delta)$ without inaccessible states. We claim that $\Sigma(t) \cap L(\mathcal{A}) \neq \varnothing$ if and only if there exists a mapping $\sigma \colon \mathcal{V}\mathsf{ar}(t) \to Q$ such that $t\sigma \in L(\mathcal{B})$.

- $\Rightarrow$ Suppose $\Sigma(t) \cap L(\mathcal{A}) \neq \varnothing$. So there exists a substitution $\tau \colon \mathcal{V}\mathsf{ar}(t) \to \mathcal{T}(\mathcal{F})$ such that $t\tau \in L(\mathcal{A}) = L(\mathcal{B})$. Hence $t\tau \to_\Delta^* q$ for some $q \in Q_f$. In this sequence every subterm $\tau(x)$ of $t\tau$ is reduced to some state. Because $\mathcal{B}$ is deterministic, different occurrences of $\tau(x)$ in $t\tau$ reduce to the same state, say $q_x \in Q$. Define the mapping $\sigma \colon \mathcal{V}\mathsf{ar}(t) \to Q$ by $\sigma(x) = q_x$ for every $x \in \mathcal{V}\mathsf{ar}(t)$. Clearly $t\tau \to_\Delta^* t\sigma \to_\Delta^* q$ and hence $t\sigma \in L(\mathcal{B})$.
- $\Leftarrow$ Suppose $t\sigma \in L(\mathcal{B})$ for some mapping $\sigma \colon \mathcal{V}\mathsf{ar}(t) \to Q$. So $t\sigma \to_\Delta^* q$ for some $q \in Q_f$. Since all states of $\mathcal{B}$ are accessible, there exists a substitution $\tau \colon \mathcal{V}\mathsf{ar}(t) \to \mathcal{T}(\mathcal{F})$ such that $\tau(x) \to_\Delta^* \sigma(x)$ for all $x \in \mathcal{V}\mathsf{ar}(t)$. Hence $t\tau \to_\Delta^* t\sigma \to_\Delta^* q$ and thus $t\tau \in L(\mathcal{B}) = L(\mathcal{A})$. In other words, $\Sigma(t) \cap L(\mathcal{A}) \neq \varnothing$.

Since there are only finitely many mappings from $\mathcal{V}\mathsf{ar}(t)$ to Q, this yields a decision procedure. □

We stress that for a linear term t there is no need to perform the expensive determinization of $\mathcal{A}$.

4 Approximations

In this section we define new approximations of the dependency graph. Our approximations are based on the framework of Durand and Middeldorp [11] for the study of decidable call-by-need computations in orthogonal term rewriting.

If $\mathcal{R}$ is an eTRS over a signature $\mathcal{F}$ and $L \subseteq \mathcal{T}(\mathcal{F})$ then $(\to_{\mathcal{R}}^*)[L]$ denotes the set of all terms $s \in \mathcal{T}(\mathcal{F})$ such that $s \to_{\mathcal{R}}^* t$ for some term $t \in L$.

Definition 7. *An* approximation mapping *is a mapping α from eTRSs to eTRSs with the property that $\to_{\mathcal{R}} \subseteq \to_{\alpha(\mathcal{R})}^*$ for every eTRS $\mathcal{R}$. In the following we write $\mathcal{R}_\alpha$ instead of $\alpha(\mathcal{R})$. We say that α is* regularity preserving *if $(\to_{\mathcal{R}_\alpha}^*)[L]$ is regular for all eTRSs $\mathcal{R}$ and regular L.*

In [11] an approximation mapping α is also required to satisfy the condition that the ground normal forms of $\mathcal{R}$ and $\mathcal{R}_\alpha$ coincide, but we do not need that condition here. Next 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.

Definition 8. *Let $\mathcal{R}$ be an eTRS. The* strong *approximation $\mathcal{R}_{\mathrm{s}}$ 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., $\mathcal{R}_{\mathrm{s}} = \{\mathsf{REN}(l) \to x \mid l \to r \in \mathcal{R}$ and x is a fresh variable$\}$. The* nv *approximation $\mathcal{R}_{\mathrm{nv}}$ is obtained from $\mathcal{R}$ by replacing all occurrences of variables in the rewrite rules by distinct fresh variables: $\mathcal{R}_{\mathrm{nv}} = \{\mathsf{REN}(l) \to \mathsf{REN}(r) \mid l \to r \in \mathcal{R}\}$. An eTRS is called growing if for every rewrite rule $l \to r$ the variables in $\mathcal{V}\mathsf{ar}(l) \cap \mathcal{V}\mathsf{ar}(r)$ occur at depth 1 in l. The* growing *approximation $\mathcal{R}_{\mathrm{g}}$ 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.*

For instance, if $\mathcal{R}$ contains the rewrite rule $\mathsf{f}(x, \mathsf{g}(x), y) \to \mathsf{f}(x, x, \mathsf{g}(y))$ then $\mathcal{R}_{\mathrm{s}}$ contains $\mathsf{f}(x, \mathsf{g}(x'), y) \to z$, $\mathcal{R}_{\mathrm{nv}}$ contains $\mathsf{f}(x, \mathsf{g}(x'), y) \to \mathsf{f}(x'', x''', \mathsf{g}(y'))$, and $\mathcal{R}_{\mathrm{g}}$ 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 $\mathcal{R}_{\mathrm{g}}$ causes no problems in the sequel.)

Theorem 9. *The approximation mappings* s, nv, *and* g *are regularity preserving.* □

Nagaya and Toyama [24] proved the above result for the growing approximation; the tree automaton that recognizes $(\to_{\mathcal{R}_{\mathrm{g}}}^*)[L]$ is defined as the limit of a finite saturation process. This saturation process is similar to the ones defined in Comon [7] and Jacquemard [16], 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 (see e.g. Durand and Middeldorp [11]).

Recently, 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 and the construction in the proof of regularity preservingness very complicated, we do not consider the inverse finite path overlapping approximation here. We note however that our results easily extend.

Definition 10. *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.

Lemma 11. *Let $\mathcal{R}$ be a TRS and α an approximation mapping.*

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

Proof.

1. Let $s \to t$ and $u \to v$ be dependency pairs of $\mathcal{R}$. Because $\mathsf{REN}(u)$ is a linear term, $\Sigma(\mathsf{REN}(u))$ is regular (Lemma 5). Since α is regularity preserving, $(\to^*_{\mathcal{R}_\alpha})[\Sigma(\mathsf{REN}(u))]$ is regular. Hence, according to Theorem 6, it is decidable whether $\Sigma(t)$ intersects with $(\to^*_{\mathcal{R}_\alpha})[\Sigma(\mathsf{REN}(u))]$. By the same reasoning it follows that it is decidable whether $\Sigma(u)$ and $(\to^*_{(\mathcal{R}^{-1})_\alpha})[\Sigma(\mathsf{REN}(t))]$ intersect. Hence it is decidable whether there exists an arrow from $s \to t$ to $u \to v$ in $\mathsf{DG}_\alpha(\mathcal{R})$.
2. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}(\mathcal{R})$. So $t\sigma \to^*_{\mathcal{R}} u\tau$ for some substitutions σ and τ. We may assume without loss of generality that $t\sigma$ and $u\tau$ are ground terms. Hence $t\sigma \in \Sigma(t) \subseteq \Sigma(\mathsf{REN}(t))$ and $u\tau \in \Sigma(u) \subseteq \Sigma(\mathsf{REN}(u))$. Consequently, $\Sigma(t) \cap (\to^*_{\mathcal{R}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$ and $\Sigma(u) \cap (\to^*_{\mathcal{R}^{-1}})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$. Because α is an approximation mapping, $\to^*_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}_\alpha}$ and $\to^*_{\mathcal{R}^{-1}} \subseteq \to^*_{(\mathcal{R}^{-1})_\alpha}$. Therefore $\Sigma(t) \cap (\to^*_{\mathcal{R}_\alpha})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$ and $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})_\alpha})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$. In other words, there exists an arrow from $s \to t$ to $u \to v$ in $\mathsf{DG}_\alpha(\mathcal{R})$. □

It should be clear that a better approximation mapping results in a better approximation of the dependency graph. Hence we have the following result.

Lemma 12. $\mathsf{DG}_{\mathrm{g}}(\mathcal{R}) \subseteq \mathsf{DG}_{\mathrm{nv}}(\mathcal{R}) \subseteq \mathsf{DG}_{\mathrm{s}}(\mathcal{R})$ *for every TRS $\mathcal{R}$.* □

The reason for considering the strong and nv approximations in this paper is that DG_{s} and $\mathsf{DG}_{\mathrm{nv}}$ are easier to compute than DG_{g}, cf. the paragraph following Theorem 9.

5 Comparison

In this section we compare our α-approximated dependency graph with the estimated dependency graph of Arts and Giesl and the approximation of the dependency graph defined by Kusakari and Toyama [19, 21].

The first two examples show that the s-approximated dependency graph and the estimated dependency graph are incomparable in general.

Example 13. Consider the TRS $\mathcal{R}$ consisting of the two rewrite rules

$$\begin{array}{rcl} \mathsf{f}(\mathsf{g}(\mathsf{a})) & \rightarrow & \mathsf{f}(\mathsf{a}) \\ \mathsf{a} & \rightarrow & \mathsf{b} \end{array}$$

There are two dependency pairs:

$$\begin{array}{rcll} \mathsf{F}(\mathsf{g}(\mathsf{a})) & \rightarrow & \mathsf{F}(\mathsf{a}) & \qquad (1) \\ \mathsf{F}(\mathsf{g}(\mathsf{a})) & \rightarrow & \mathsf{A} & \qquad (2) \end{array}$$

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:

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

We have $(\mathcal{R}^{-1})_\mathrm{s} = \{\mathsf{f}(\mathsf{a}) \rightarrow x, \mathsf{b} \rightarrow x\}$. Hence $(\rightarrow^*_{(\mathcal{R}^{-1})_\mathrm{s}})[\{\mathsf{F}(\mathsf{a})\}]$ 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 $\mathsf{DG}_\mathrm{s}(\mathcal{R})$.

Example 14. Consider the TRS $\mathcal{R}$ consisting of the single rewrite rule

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

There is one dependency pair:

$$\mathsf{F}(x, x) \;\rightarrow\; \mathsf{F}(\mathsf{a}, \mathsf{b})$$

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 $\Sigma(\mathsf{F}(\mathsf{a}, \mathsf{b})) \cap (\rightarrow^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(\mathsf{F}(x, x)))]$ and $\Sigma(\mathsf{F}(x, x)) \cap (\rightarrow^*_{(\mathcal{R}^{-1})_\mathrm{s}})[\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a}, \mathsf{b})))]$ 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-left-linearity in the preceding example is essential. This is shown in Lemma 16 below. In the proof we make use of the following lemma. Here $\leftarrow^*_{\mathcal{R}_\mathrm{s}}$ is the inverse of the relation $\rightarrow^*_{\mathcal{R}_\mathrm{s}}$ (which is different from $\rightarrow^*_{(\mathcal{R}^{-1})_\mathrm{s}}$).

Lemma 15. $(\leftarrow^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(t))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}(t)))$ *for every 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 the root symbol of t is a defined symbol then $\mathsf{CAP}(t)$ is a variable and hence $\Sigma(\mathsf{REN}(\mathsf{CAP}(t))) = \mathcal{T}(\mathcal{F})$ and thus trivially $(\leftarrow^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(t))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}(t)))$. Suppose $t = f(t_1, \ldots, t_n)$ with f a constructor. Because the left-hand side of every rule in $\mathcal{R}_\mathrm{s}$ starts with a defined symbol and the arguments of $\mathsf{REN}(t)$ do not share variables, $(\leftarrow^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(t))] = \{f(s_1, \ldots, s_n) \mid s_i \in \Sigma(\mathsf{REN}(t_i))\}$. Also $\Sigma(\mathsf{REN}(\mathsf{CAP}(t))) = \{f(s_1, \ldots, s_n) \mid s_i \in \Sigma(\mathsf{REN}(\mathsf{CAP}(t_i)))\}$. Hence the desired inclusion follows from the induction hypothesis. $\square$

The previous lemma does not hold for eTRSs. For instance, consider the eTRS $\mathcal{R} = \{x \to \mathsf{a}\}$ over the signature consisting of the constants a and b. If $t = \mathsf{b}$ then $(\leftarrow^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(t))] = \{\mathsf{a}, \mathsf{b}\}$ and $\Sigma(\mathsf{REN}(\mathsf{CAP}(t))) = \{\mathsf{b}\}$.

Lemma 16. *If $\mathcal{R}$ is a left-linear TRS then $\mathsf{DG}_\mathrm{s}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$.*

Proof. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}_\mathrm{s}(\mathcal{R})$. By definition, $\Sigma(t) \cap (\to^*_{\mathcal{R}_\mathrm{s}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. Since $\mathcal{R}$ is left-linear, u is a linear term and thus $\Sigma(\mathsf{REN}(u)) = \Sigma(u)$. Hence there exist ground substitutions σ and τ such that $t\sigma \to^*_{\mathcal{R}_\mathrm{s}} u\tau$. Clearly $t\sigma \in \Sigma(\mathsf{REN}(t))$. According to the preceding lemma $u\tau \in \Sigma(\mathsf{REN}(\mathsf{CAP}(t)))$. Since $\mathsf{REN}(\mathsf{CAP}(t))$ and u do not share variables, they are unifiable and thus there exists an arrow from $s \to t$ to $u \to v$ in $\mathsf{EDG}(\mathcal{R})$. □

Actually, with the strong approximation we can never benefit from non-linearity. This is formally expressed in the following lemma.

Lemma 17. *Let $\mathcal{R}$ be a nonempty eTRS, t a term, and L a set of ground terms. The following statements are equivalent:*

1. *$\Sigma(t) \cap (\to^*_{\mathcal{R}_\mathrm{s}})[L] \neq \varnothing$,*
2. *$\Sigma(\mathsf{REN}(t)) \cap (\to^*_{\mathcal{R}_\mathrm{s}})[L] \neq \varnothing$.*

Proof.

⇒ Obvious since $\Sigma(t) \subseteq \Sigma(\mathsf{REN}(t))$.

⇐ Let Δ be an arbitrary ground redex and define the substitution $\sigma = \{x \mapsto \Delta \mid x \in \mathcal{V}\mathsf{ar}(t)\}$. Because in $\mathcal{R}_\mathrm{s}$ a redex can be rewritten to any term, $t\sigma \to^*_{\mathcal{R}_\mathrm{s}} t'$ for every $t' \in \Sigma(\mathsf{REN}(t))$. Hence, if $t' \in \Sigma(\mathsf{REN}(t)) \cap (\to^*_{\mathcal{R}_\mathrm{s}})[L]$ then $t\sigma \in \Sigma(t) \cap (\to^*_{\mathcal{R}_\mathrm{s}})[L]$. □

As a consequence, the strong approximation is not all that useful for approximating dependency graphs. For the nv approximation matters are quite different. Our next result states that the nv-approximated dependency graph is always a subgraph of the estimated dependency graph. In order to prove this, we need the following preliminary result.

Lemma 18. $(\mathcal{R}_\mathrm{nv})^{-1} = (\mathcal{R}^{-1})_\mathrm{nv}$ *for every eTRS $\mathcal{R}$.*

Proof. Since $\mathcal{R}_\mathrm{nv} = \{\mathsf{REN}(l) \to \mathsf{REN}(r) \mid l \to r \in \mathcal{R}\}$, the result is obvious. □

We stress that the above lemma is not true for the strong and growing approximations. For the strong approximation the TRSs of Examples 13 and 14 serve as counterexample.

Theorem 19. $\mathsf{DG}_\mathrm{nv}(\mathcal{R}) \subseteq \mathsf{EDG}(\mathcal{R})$ *for every TRS $\mathcal{R}$.*

Proof. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$. By definition, $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})_{\mathrm{nv}}})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$. According to Lemmata 18 and 15, and using the observation that $\to^*_{\mathcal{R}_{\mathrm{nv}}}$ is a subrelation of $\to^*_{\mathcal{R}_{\mathrm{s}}}$, $(\to^*_{(\mathcal{R}^{-1})_{\mathrm{nv}}})[\Sigma(\mathsf{REN}(t))] = (\leftarrow^*_{\mathcal{R}_{\mathrm{nv}}})[\Sigma(\mathsf{REN}(t))] \subseteq (\leftarrow^*_{\mathcal{R}_{\mathrm{s}}})$ $[\Sigma(\mathsf{REN}(t))] \subseteq \Sigma(\mathsf{REN}(\mathsf{CAP}(t)))$. Hence $\Sigma(u) \cap \Sigma(\mathsf{REN}(\mathsf{CAP}(t))) \neq \varnothing$ and hence u and $\mathsf{REN}(\mathsf{CAP}(t))$ are unifiable. Therefore the arrow from $s \to t$ to $u \to v$ also exists in $\mathsf{EDG}(\mathcal{R})$. □

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

Example 20. Consider the TRS $\mathcal{R}$ consisting of the two rewrite rules

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

There is one dependency pair:

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

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. We have $\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x))) = \{\mathsf{F}(\mathsf{a},\mathsf{b},t) \mid t \in \mathcal{T}(\mathcal{F})\}$ and $\mathcal{R}_{\mathrm{nv}} = \{\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{f}(x_1,x_2,x_3), \mathsf{a} \to \mathsf{c}\}$. Consequently $(\to^*_{\mathcal{R}_{\mathrm{nv}}})[\Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x)))] = \Sigma(\mathsf{REN}(\mathsf{F}(\mathsf{a},\mathsf{b},x)))$ and since no instance of $\mathsf{F}(x,x,x)$ belongs to this set, $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$ contains no arrow. Therefore $\mathcal{R}$ is trivially terminating.

The TRS in the above example is not *DP quasi-simply terminating*. The class of DP quasi-simply terminating TRSs was introduced by Giesl and Ohlebusch [14] and supposed to "capture all TRSs where an automated termination proof using dependency pairs is potentially feasible". We note that the various refinements of the dependency pair method (narrowing, rewriting, instantiation; see Giesl and Arts [12]) are not applicable and moreover that proving innermost termination (which is easy with the standard dependency pair technique) is insufficient for termination as the TRS does not belong to a known class for which termination and innermost termination coincide.

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 21. Consider the TRS $\mathcal{R}$ consisting of the three rewrite rules

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

There are three dependency pairs:

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

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

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

In particular, $\mathsf{F}(\mathsf{a}, \mathsf{g}(\mathsf{a}, \mathsf{b})) \to_{\mathcal{R}_{\rm nv}} \mathsf{F}(\mathsf{a}, \mathsf{a})$ which explains the arrows from (1) to (1) and (2). The problematic cycle $\{(1)\}$ does not exist in $\mathsf{DG}_{\rm g}(\mathcal{R})$ because no ground instance of $\mathsf{F}(x, \mathsf{g}(x, \mathsf{b}))$ rewrites in $\mathcal{R}_{\rm g}$ 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 (obtained from Theorem 2) are easily satisfied (e.g. by taking $\pi(\mathsf{f}) = 1$ in combination with the lexicographic path order with precedence $\mathsf{G} > \mathsf{h}$ and $\mathsf{g} > \mathsf{h}$).[1]

In the final part of this section we compare our α-approximated dependency graph with the approximation of the dependency graph defined by Kusakari and Toyama [19, 21]. Their approximation relies on the concepts of ω-reduction and Ω-reduction. The first concept stems from Huet and Lévy [15].

Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. Let Ω be a fresh constant. The set of ground terms over the extended signature $\mathcal{F} \cup \{\Omega\}$ is denoted by $\mathcal{T}_\Omega(\mathcal{F})$. Given a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, the term in $\mathcal{T}_\Omega(\mathcal{F})$ obtained from t by replacing all variables by Ω is denoted by t_Ω. The prefix order $\geqslant$ on $\mathcal{T}_\Omega(\mathcal{F})$ is defined by the following two clauses:

- $t \geqslant \Omega$ for every $t \in \mathcal{T}_\Omega(\mathcal{F})$,
- $f(s_1, \dots, s_n) \geqslant f(t_1, \dots, t_n)$ if $s_i \geqslant t_i$ for every $1 \leqslant i \leqslant n$.

Two terms $s, t \in \mathcal{T}_\Omega(\mathcal{F})$ are compatible, denoted by $s \uparrow t$, if there exists a term $u \in \mathcal{T}_\Omega(\mathcal{F})$ such that both $u \geqslant s$ and $u \geqslant t$. Finally, ω-reduction is the relation $\to_\omega$ on $\mathcal{T}_\Omega(\mathcal{F})$ defined as follows: $s \to_\omega t$ if and only if $s = C[s']$ and $t = C[\Omega]$ such that $\Omega \neq s' \uparrow l_\Omega$ for some $l \to r \in \mathcal{R}$. It is easy to prove that ω-reduction is terminating and confluent. Hence every term $t \in \mathcal{T}_\Omega(\mathcal{F})$ has a unique normal form, which is denoted by $\omega(t)$. It is well-known that ω-reduction is closely related to the strong approximation. Below we make use of the following well-known facts (for all terms $s, t \in \mathcal{T}_\Omega(\mathcal{F})$):

- $\omega(t) \leqslant t$,
- if $s \leqslant t$ then $\omega(s) \leqslant \omega(t)$.

The concept of Ω-reduction corresponds to the nv approximation and is defined as follows: $s \to_\Omega t$ if and only if $s = C[s']$ and $t = C[r_\Omega]$ for some $l \to r \in \mathcal{R}$ such that $\Omega \neq s' \uparrow l_\Omega$. Unlike ω-reduction, Ω-reduction is in general neither confluent nor terminating.

[1] Again, the TRS is not DP quasi-simply terminating. Unlike the previous example, proving innermost termination is sufficient for termination, but the estimated innermost dependency graph coincides with $\mathsf{EDG}(\mathcal{R}) = \mathsf{DG}_{\rm nv}(\mathcal{R})$ and the narrowing refinement for innermost termination fails to make the requirements for an automatic proof easier.

Lemma 22. *Let $\mathcal{R}$ be a TRS. If $s \rightarrow^*_{\mathcal{R}_{\text{nv}}} t$ and $s' \leqslant s$ then $s' \rightarrow^*_{\Omega} t'$ for some $t' \leqslant t$.*

Proof. Induction on the length of $s \rightarrow^*_{\mathcal{R}_{\text{nv}}} t$, using the easy to prove fact that if $s \rightarrow_{\mathcal{R}_{\text{nv}}} t$ and $s' \leqslant s$ then $s' \rightarrow^{=}_{\Omega} t'$ for some $t' \leqslant t$. ☐

We now have all ingredients to define Kusakari and Toyama's approximation of the dependency graph. Actually, their definition applies to AC rewriting, an extension that we do not consider in this paper. The definition below is the specialization to ordinary term rewriting.

Definition 23. *Let $\mathcal{R}$ be a TRS. For every $n \geqslant 0$ we define the graph $\mathsf{DG}^n_{\Omega}(\mathcal{R})$ as follows. Its nodes are the dependency pairs of $\mathcal{R}$ and there is an arrow from $s \rightarrow t$ to $u \rightarrow v$ if and only if there exists a term $t' \in \mathcal{T}_{\Omega}(\mathcal{F})$ such that $t' \uparrow u_{\Omega}$ and either $t_{\Omega} \rightarrow^m_{\Omega} t'$ with $m < n$ or $t_{\Omega} \rightarrow^n_{\Omega} \cdot \rightarrow^!_{\omega} t'$. (Note that the latter condition is equivalent to $t_{\Omega} \rightarrow^n_{\Omega} t''$ and $t' = \omega(t'')$ for some term $t'' \in \mathcal{T}_{\Omega}(\mathcal{F})$.)*

Lemma 24 (Kusakari and Toyama [19, 21])**.** *Let $\mathcal{R}$ be a TRS and $n \geqslant 0$.*

1. *$\mathsf{DG}^n_{\Omega}(\mathcal{R})$ is computable.*
2. *$\mathsf{DG}^n_{\Omega}(\mathcal{R}) \subseteq \mathsf{DG}(\mathcal{R})$.* ☐

It is not difficult to show that $\mathsf{EDG}(\mathcal{R})$ and $\mathsf{DG}^n_{\Omega}(\mathcal{R})$ are incomparable in general, for all $n \geqslant 0$ (contradicting the remark in Kusakari and Toyama [21] that their algorithm for approximating the dependency graph is more powerful than the one of Arts and Giesl). For instance, for the TRS $\mathcal{R}$ of Example 14 $\mathsf{DG}^n_{\Omega}(\mathcal{R})$ contains a cycle for every $n \geqslant 0$ whereas $\mathsf{EDG}(\mathcal{R})$ is empty. The same holds for $\mathsf{DG}_{\text{s}}(\mathcal{R})$. However, it is easy to prove that $\mathsf{DG}_{\text{s}}(\mathcal{R})$ is always a subgraph of $\mathsf{DG}^0_{\Omega}(\mathcal{R})$ and sometimes a proper subgraph, like in Example 13 where $\mathsf{DG}^0_{\Omega}(\mathcal{R})$ coincides with $\mathsf{EDG}(\mathcal{R})$. Below we compare Kusakari and Toyama's approximation with our nv-approximated dependency graph.

Lemma 25. *Let $\mathcal{R}$ be a TRS. If $n > 0$ then $\mathsf{DG}^n_{\Omega}(\mathcal{R}) \subseteq \mathsf{DG}^{n-1}_{\Omega}(\mathcal{R})$.*

Proof. Suppose there is an arrow from dependency pair $s \rightarrow t$ to dependency pair $u \rightarrow v$ in $\mathsf{DG}^n_{\text{nv}}(\mathcal{R})$. So there exists a term $t' \in \mathcal{T}_{\Omega}(\mathcal{F})$ such that $t' \uparrow u_{\Omega}$ and either $t_{\Omega} \rightarrow^m_{\Omega} t'$ with $m < n$ or $t_{\Omega} \rightarrow^n_{\Omega} \cdot \rightarrow^!_{\omega} t'$. First suppose that $t_{\Omega} \rightarrow^m_{\Omega} t'$ with $m < n$. If $m < n - 1$ then the arrow from $s \rightarrow t$ to $u \rightarrow v$ also exists in $\mathsf{DG}^{n-1}(\mathcal{R})$. If $m = n - 1$ then we reason as follows. Since $\omega(t') \leqslant t'$, $\omega(t') \uparrow u_{\Omega}$. Clearly $t_{\Omega} \rightarrow^{n-1}_{\Omega} t' \rightarrow^!_{\omega} \omega(t')$. Hence the arrow from $s \rightarrow t$ to $u \rightarrow v$ exists in $\mathsf{DG}^{n-1}(\mathcal{R})$. Finally consider the case that $t_{\Omega} \rightarrow^n_{\Omega} \cdot \rightarrow^!_{\omega} t'$. So there exists terms t_1 and t_2 such that $t_{\Omega} \rightarrow^{n-1}_{\Omega} t_1 \rightarrow_{\Omega} t_2 \rightarrow^!_{\omega} t'$. Clearly $t_1 \rightarrow_{\omega} t'_2$ with $t'_2 \leqslant t_2$. We have $\omega(t_1) = \omega(t'_2) \leqslant \omega(t_2) = t'$. Hence $\omega(t_1)$ and u_{Ω} are compatible. Since $t_{\Omega} \rightarrow^{n-1}_{\Omega} t_1 \rightarrow^!_{\omega} \omega(t_1)$, the arrow from $s \rightarrow t$ to $u \rightarrow v$ exists in $\mathsf{DG}^{n-1}_{\Omega}(\mathcal{R})$. ☐

The following result is the key to showing that our nv-approximated dependency graph is a subgraph of $\mathsf{DG}^n_{\Omega}(\mathcal{R})$, for all $n \geqslant 0$.

Lemma 26. *Let $s \to t$ and $u \to v$ be dependency pairs of $\mathcal{R}$. If $\Sigma(\mathsf{REN}(t)) \cap (\to^*_{\mathcal{R}_{\rm nv}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$ then there is an arrow from $s \to t$ to $u \to v$ in $\mathsf{DG}^n_\Omega(\mathcal{R})$ for all $n \geqslant 0$.*

Proof. Suppose $\Sigma(\mathsf{REN}(t)) \cap (\to^*_{\mathcal{R}_{\rm nv}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. So $\mathsf{REN}(t)\sigma \to^*_{\mathcal{R}_{\rm nv}} \mathsf{REN}(u)\tau$ for some ground substitutions σ and τ. Since $t_\Omega \leqslant \mathsf{REN}(t)\sigma$, an application of Lemma 22 yields $t_\Omega \to^*_\Omega u'$ for some term $u' \leqslant \mathsf{REN}(u)\tau$. Because also $u_\Omega \leqslant \mathsf{REN}(u)\tau$, $u' \uparrow u_\Omega$. Let m be the length of the Ω-reduction sequence from t_Ω to u'. It follows that there is an arrow from $s \to t$ to $u \to v$ in $\mathsf{DG}^n_\Omega(\mathcal{R})$ for all $n > m$. According to Lemma 25, the arrow exists also in $\mathsf{DG}^n_\Omega(\mathcal{R})$ for $n \leqslant m$. □

Theorem 27. $\mathsf{DG}_{\rm nv}(\mathcal{R}) \subseteq \mathsf{DG}^n_\Omega(\mathcal{R})$ *for every TRS $\mathcal{R}$ and $n \geqslant 0$.*

Proof. Suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}_{\rm nv}(\mathcal{R})$. By definition, $\Sigma(t) \cap (\to^*_{\mathcal{R}_{\rm nv}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. Since $\Sigma(t) \subseteq \Sigma(\mathsf{REN}(t))$, also $\Sigma(\mathsf{REN}(t)) \cap (\to^*_{\mathcal{R}_{\rm nv}})[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. According to Lemma 26 the arrow from $s \to t$ to $u \to v$ exists in $\mathsf{DG}^n_\Omega(\mathcal{R})$ for all $n \geqslant 0$. □

The reverse inclusion does not hold. Consider for instance the TRS $\mathcal{R}$ of Example 20. Since $\mathsf{F}(\Omega, \Omega, \Omega)$ is compatible with $\mathsf{F}(\mathsf{a}, \mathsf{b}, \Omega)$, $\mathsf{DG}^n_\Omega(\mathcal{R})$ contains a cycle for all $n \geqslant 0$.

In retrospect, Kusakari and Toyama's approximation suffers from the following two problems: (1) since all variables are replaced by Ω, TRSs that are terminating because of non-linearity cannot be handled appropriately, and (2) there is no need to bound the number of Ω-reduction steps, rather, by avoiding such a bound we can make effective use of tree automata techniques.

6 Decidable Classes

Termination is known to be decidable for several subclasses of TRSs. In this section we investigate whether these decidability results can be obtained with the dependency pair technique. The best known class of TRSs with a decidable termination problem is the class of right-ground TRSs (Dershowitz [9]). The following easy result states that in principle the dependency pair technique is very suitable for deciding termination of right-ground TRSs.

Theorem 28. *A right-ground TRS $\mathcal{R}$ is terminating if and only if* $\mathsf{DG}(\mathcal{R})$ *contains no cycles.*

Proof.

⇒ Suppose $\mathsf{DG}(\mathcal{R})$ contains a cycle $\mathcal{C} = \{s_i \to t_i \mid 1 \leqslant i \leqslant n\}$. We show that $\mathcal{R}$ is non-terminating. Without loss of generality we assume that $\mathcal{C}$ is minimal. Since $\mathcal{R}$ is right-ground, there exist substitutions σ_i such that $t_i \to^*_{\mathcal{R}} s_{i+1}\sigma_i$ for all $1 \leqslant i \leqslant n$ and with $s_{n+1} = s_1$. By definition of dependency pairs, for every $1 \leqslant i \leqslant n$ there exists a rewrite rule $l_i \to r_i \in \mathcal{R}$ and a subterm u_i of

r_i such that $s_i = l_i^\sharp$ and $t_i = u_i^\sharp$. Let C_i be the context such that $r_i = C_i[u_i]$. Since all steps in $u_i^\sharp \to_{\mathcal{R}}^* l_{i+1}^\sharp\sigma_i$ take place below the root position, we also have $u_i \to_{\mathcal{R}}^* l_{i+1}\sigma_i$ and thus $r_i = C_i[u_i] \to_{\mathcal{R}}^* C_i[l_{i+1}\sigma_i]$ (with $l_{n+1} = l_1$). Therefore

$$\begin{aligned} l_1 \to_{\mathcal{R}} r_1 \to_{\mathcal{R}}^* C_1[l_2\sigma_1] \to_{\mathcal{R}} C_1[r_2] \to_{\mathcal{R}}^* C_1[C_2[l_3\sigma_2]] \\ \to_{\mathcal{R}} \quad \cdots \quad \to_{\mathcal{R}}^* C_1[C_2[\cdots[C_n[l_1\sigma_n]]\cdots]] \end{aligned}$$

which gives rise to an infinite rewrite sequence.

$\Leftarrow$ If there are no cycles in $\mathsf{DG}(\mathcal{R})$ then the conditions of Theorem 2 are trivially satisfied and thus $\mathcal{R}$ is terminating. □

So as far as termination of right-ground TRSs is concerned, the only thing that matters is a good approximation of the dependency graph. Next we consider how the various approximations of the dependency graph deal with right-ground TRSs.

Theorem 29. *For every left-linear right-ground TRS $\mathcal{R}$, $\mathsf{DG}(\mathcal{R}) = \mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$.*

Proof. According to Lemma 11 it suffices to show that $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R}) \subseteq \mathsf{DG}(\mathcal{R})$. So suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$. Hence $\Sigma(t) \cap (\to_{\mathcal{R}_{\mathrm{nv}}}^*)[\Sigma(\mathsf{REN}(u))] \neq \varnothing$. Because $\mathcal{R}$ is left-linear and right-ground, $\mathcal{R}_{\mathrm{nv}} = \mathcal{R}$, t is a ground term, and u is linear. Hence $t \in (\to_{\mathcal{R}}^*)[\Sigma(u)]$ and thus $t \to_{\mathcal{R}}^* u\sigma$ for some ground substitution σ. Therefore the arrow from $s \to t$ to $u \to v$ also exists in $\mathsf{DG}(\mathcal{R})$. □

The following example shows that without the left-linearity condition $\mathsf{DG}(\mathcal{R})$ and $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$ may differ.

Example 30. Consider the right-ground TRS $\mathcal{R}$ consisting of the three rewrite rules

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

There are four dependency pairs:

$$\begin{array}{rcll rcll} \mathsf{F}(\mathsf{a}) & \to & \mathsf{G}(\mathsf{h}(\mathsf{a},\mathsf{b})) & (1) & \mathsf{G}(\mathsf{g}(\mathsf{a})) & \to & \mathsf{F}(\mathsf{b}) & (3) \\ \mathsf{F}(\mathsf{a}) & \to & \mathsf{H}(\mathsf{a},\mathsf{b}) & (2) & \mathsf{H}(x,x) & \to & \mathsf{G}(\mathsf{a}) & (4) \end{array}$$

Because $\mathcal{R}_{\mathrm{nv}}$ contains the rewrite rule $\mathsf{h}(x,y) \to \mathsf{g}(\mathsf{a})$ and $(\mathcal{R}_{\mathrm{nv}})^{-1} = (\mathcal{R}^{-1})_{\mathrm{nv}}$, $\mathsf{DG}_{\mathrm{nv}}(\mathcal{R})$ contains an arrow from (1) to (3). However, since $\mathsf{G}(\mathsf{h}(\mathsf{a},\mathsf{b}))$ does not rewrite to $\mathsf{G}(\mathsf{g}(\mathsf{a}))$ in $\mathcal{R}$, this arrow does not exist in $\mathsf{DG}(\mathcal{R})$.

Note that in the previous example $\mathcal{R}_{\mathrm{g}}$ also contains the rule $\mathsf{h}(x,y) \to \mathsf{g}(\mathsf{a})$ but the corresponding rule in $(\mathcal{R}^{-1})_{\mathrm{g}}$ is $\mathsf{g}(\mathsf{a}) \to \mathsf{h}(x,x)$ and therefore $\mathsf{G}(\mathsf{g}(\mathsf{a}))$ does not belong to $(\to_{(\mathcal{R}^{-1})_{\mathrm{g}}}^*)[\{\mathsf{G}(\mathsf{h}(\mathsf{a},\mathsf{b}))\}]$. Hence there is no arrow from (1) to (3) in $\mathsf{DG}_{\mathrm{g}}(\mathcal{R})$. This holds in general.

Theorem 31. *For every right-ground TRS $\mathcal{R}$,* $\mathsf{DG}(\mathcal{R}) = \mathsf{DG}_{\mathrm{g}}(\mathcal{R})$.

Proof. According to Lemma 11 it suffices to show that $\mathsf{DG}_{\mathrm{g}}(\mathcal{R}) \subseteq \mathsf{DG}(\mathcal{R})$. So suppose there is an arrow from dependency pair $s \to t$ to dependency pair $u \to v$ in $\mathsf{DG}_{\mathrm{g}}(\mathcal{R})$. Hence $\Sigma(u) \cap (\to^*_{(\mathcal{R}^{-1})_{\mathrm{g}}})[\Sigma(\mathsf{REN}(t))] \neq \varnothing$. Because $\mathcal{R}$ is right-ground, $(\mathcal{R}^{-1})_{\mathrm{g}} = \mathcal{R}^{-1}$ and t is a ground term. Hence $\Sigma(u) \cap (\leftarrow^*_{\mathcal{R}})[\{t\}] \neq \varnothing$ and thus $t \to^*_{\mathcal{R}} u\sigma$ for some ground substitution σ. Therefore the arrow from $s \to t$ to $u \to v$ also exists in $\mathsf{DG}(\mathcal{R})$. □

The above results provide an easy decision procedure for termination of right-ground TRSs $\mathcal{R}$: Compute the dependency graph of $\mathcal{R}$ using the growing (nv, if $\mathcal{R}$ is left-linear) approximation and determine whether there are any cycles. We stress that the above results are not true for the estimated dependency graph.

Recently, Nagaya and Toyama [24] obtained the following decidability result.

Theorem 32. *Termination is decidable for almost orthogonal growing TRSs.* □

It should be noted that this result does not cover the preceding results due to the almost orthogonality requirement. (A TRS is called almost orthogonal if it is left-linear and all critical pairs are trivial overlays.) On the other hand, although it is very easy to prove that $\mathsf{DG}(\mathcal{R}) = \mathsf{DG}_{\mathrm{g}}(\mathcal{R})$ for every left-linear growing TRS $\mathcal{R}$, the dependency pair approach does not seem to give an easy decision procedure since the dependency graph may contain cycles, as shown in the following example.

Example 33. Consider the (almost) orthogonal growing TRS $\mathcal{R}$ consisting of the two rewrite rules

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

There are two dependency pairs:

$$\begin{aligned} \mathsf{F}(x) &\to \mathsf{G}(x) \qquad (1) \\ \mathsf{G}(\mathsf{a}) &\to \mathsf{F}(\mathsf{b}) \qquad (2) \end{aligned}$$

One easily verifies that $\mathsf{DG}(\mathcal{R})$ contains a cycle:

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

However, $\mathcal{R}$ is clearly terminating (and it is very easy to solve the constraints stemming from the dependency pair technique in Theorem 2).

7 Conclusion

In this paper we have shown that simple tree automata techniques are useful to obtain better approximations of the dependency graph and hence we can automatically prove termination of a larger class of TRSs. More sophisticated tree automata techniques have been developed for dealing with non-linearity, see [8, Chapter 4], but we are not aware of any preservation results for the corresponding language classes and hence it is unclear whether these techniques could further improve automatic termination techniques.

Obviously, our α-approximated dependency graphs are harder to compute than the estimated dependency graph of Arts and Giesl. Consequently, we do not propose to eliminate the estimated dependency graph. Rather, our approximations should be tried only if tools based on the estimated dependency graph (like [1]) fail to prove termination or maybe in parallel to the search for suitable argument filterings and orderings to satisfy the resulting constraints. Clearly experimentation is needed to determine when to invoke our approximations. Currently we are working on an implementation of our algorithms.

It is worthwhile to investigate whether our approach can be extended to AC termination ([21, 23]) and to innermost termination ([4]). For AC termination we do not expect any problems, but innermost termination seems more difficult. The reason is that the existence of an arrow from $s \to t$ to $u \to v$ in the *innermost* dependency graph does not only depend on whether a ground instance $t\sigma$ of t innermost rewrites to a ground instance $u\tau$ of u, but $s\sigma$ and $u\tau$ are additionally required to be normal forms. The latter condition is easily verified by tree automata techniques but it is unclear how to deal with the synchronization between the two conditions.

Since there are numerous examples of terminating TRSs whose dependency graphs do contain cycles, it goes without saying that the work reported in this paper is not the final answer to the problem of proving termination of rewrite systems automatically.

Acknowledgements

I thank Seitaro Yuuki for useful discussions. The paper benefitted from detailed comments of Jürgen Giesl and the anonymous referees.

References

1. T. Arts. System description: The dependency pair method. In *Proc. 11th RTA*, volume 1833 of *LNCS*, pages 261–264, 2000.
2. T. Arts and J. Giesl. Modularity of termination using dependency pairs. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 226–240, 1998.
3. T. Arts and J. Giesl. Applying rewriting techniques to the verification of Erlang processes. In *Proc. 13th CSL*, volume 1862 of *LNCS*, pages 457–471, 2000.
4. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
5. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
6. F. Bellegarde and P. Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, 1:79–96, 1990.
7. H. Comon. Sequentiality, monadic second-order logic and tree automata. *Information and Computation*, 157:25–51, 2000.

8. H. Comon, 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/`.
9. N. Dershowitz. Termination of linear rewriting systems (preliminary version). In *Proc. 8th ICALP*, volume 115 of *LNCS*, pages 448–458, 1981.
10. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
11. I. Durand and A. Middeldorp. Decidable call by need computations in term rewriting. In *Proc. 14th CADE*, volume 1249 of *LNAI*, pages 4–18, 1997.
12. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Applicable Algebra in Engineering, Communication and Computing*, 2001. To appear.
13. J. Giesl and A. Middeldorp. Eliminating dummy elimination. In *Proc. 17th CADE*, volume 1831 of *LNAI*, pages 309–323, 2000.
14. J. Giesl and E. Ohlebusch. Pushing the frontiers of combining rewrite systems farther outwards. In *Proc. FroCoS'98*, volume 7 of *Studies in Logic and Computation*, pages 141–160. Wiley, 2000.
15. G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, I and II. In *Computational Logic, Essays in Honor of Alan Robinson*, pages 396–443. The MIT Press, 1991. Original version: Report 359, Inria, 1979.
16. F. Jacquemard. Decidable approximations of term rewriting systems. In *Proc. 7th RTA*, volume 1103 of *LNCS*, pages 362–376, 1996.
17. S. Kamin and J.J. Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, University of Illinois, 1980.
18. D.E. Knuth and P. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
19. K. Kusakari. *Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems*. PhD thesis, JAIST, 2000.
20. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proc. 1st PPDP*, volume 1702 of *LNCS*, pages 48–62, 1999.
21. K. Kusakari and Y. Toyama. On proving AC-termination by AC-dependency pairs. Research Report IS-RR-98-0026F, School of Information Science, JAIST, 1998.
22. D. Lankford. On proving term rewriting systems are noetherian. Report MTP-3, Louisiana Technical University, 1979.
23. C. Marché and X. Urbain. Termination of associative-commutative rewriting by dependency pairs. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 241–255, 1998.
24. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. In *Proc. 10th RTA*, volume 1631 of *LNCS*, pages 256–270, 1999.
25. T. Takai, Y. Kaji, and H. Seki. Right-linear finite path overlapping term rewriting systems effectively preserve recognizability. In *Proc. 11th RTA*, volume 1833 of *LNCS*, pages 246–260, 2000.
26. S. Tison. Tree automata and term rewrite systems, July 2000. Invited tutorial at the 11th RTA.
27. H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.
28. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.