Submitted to the International Conference on Rewriting Techniques and Applications
http://rewriting.loria.fr/rta/

CLOSING THE GAP BETWEEN RUNTIME COMPLEXITY AND POLYTIME COMPUTABILITY

MARTIN AVANZINI [1] AND GEORG MOSER [1]

[1] Institute of Computer Science, University of Innsbruck, Austria

E-mail address: {martin.avanzini,georg.moser}@uibk.ac.at

ABSTRACT. In earlier work, we have shown that for confluent term rewrite systems (TRSs for short), innermost polynomial runtime complexity induces polytime computability of the functions defined. In this paper, we generalise this result to full rewriting, for that we exploit graph rewriting. We give a new proof of the adequacy of graph rewriting for full rewriting that allows for a precise control of the resources copied. In sum we completely describe an implementation of rewriting on a Turing machine (TM for short). We show that the runtime complexity of the TRS and the runtime complexity of the TM is polynomially related. Our result strengthens the evidence that the complexity of a rewrite system is truthfully represented through the length of derivations. Moreover our result allows the classification of deterministic as well as nondeterministic polytime-computation based on runtime complexity analysis of rewrite systems.

1. Introduction

Recently we see increased interest in studies of the (maximal) derivation length of term rewrite system, compare for example [10, 9, 15, 11, 14]. We are interested in techniques to automatically classify the complexity of term rewrite systems (TRS for short) and have introduced the *polynomial path order* POP* and extensions of it, cf. [1, 2]. POP* is a restriction of the multiset path order [18] and whenever compatibility of a TRS $\mathcal{R}$ with POP* can be shown then the (innermost) *runtime complexity* of $\mathcal{R}$ is polynomially bounded. Here the runtime complexity of a TRS measures the maximal number of rewrite steps as a function in the size of the initial term, where the initial terms are restricted argument normalised terms (aka *basic* terms).

We have successfully implemented this technique.[1] As a consequence we have a fully automatic (but of course incomplete) procedure that verifies for a given TRS $\mathcal{R}$ whether it admits at most polynomial runtime complexity. In this paper, we study the question whether such techniques are restricted to *runtime complexity*, or whether they can be applied

1998 ACM Subject Classification: F 1.2, F 1.3, F 4.2.
Key words and phrases: term rewriting, graph rewriting, complexity analysis, polytime computability.
This research is supported by FWF (Austrian Science Fund) projects P20133.

[1] Our implementation forms part of the *Tyrolean Complexity Tool* (TCT for short). For further information, see http://cl-informatik.uibk.ac.at/software/tct/.

also for the (automated) classification of the *computational complexity* of the functions computed by the given TRS $\mathcal{R}$. For motivation consider the TRS $\mathcal{R}_{\mathsf{sat}}$ given in the next example. It is not difficult to see that $\mathcal{R}_{\mathsf{sat}}$ encodes the function problem FSAT associated to the well-known satisfiability problem SAT. FSAT is complete for the class of *function problems over* NP (FNP for short), compare [16].

Example 1.1. Consider the following TRS $\mathcal{R}_{\mathsf{sat}}$:

$$
\begin{array}{rrcl}
1\colon & \mathsf{if}(\mathsf{tt},t,e) & \to & t \\
2\colon & \mathsf{if}(\mathsf{ff},t,e) & \to & e \\
3\colon & \mathsf{choice}(x:xs) & \to & x \\
4\colon & \mathsf{choice}(x:xs) & \to & \mathsf{choice}(xs) \\
5\colon & \mathsf{guess}(\mathsf{nil}) & \to & \mathsf{nil} \\
6\colon & \mathsf{guess}(c:cs) & \to & \mathsf{choice}(c):\mathsf{guess}(cs) \\
7\colon & \mathsf{in}(x,\mathsf{nil}) & \to & \mathsf{ff} \\
8\colon & \mathsf{in}(x,y:ys) & \to & \mathsf{if}(x=y,\mathsf{tt},\mathsf{in}(x,ys)) \\
9\colon & \neg\mathsf{1}(x) & \to & \mathsf{0}(x) \\
10\colon & \neg\mathsf{0}(x) & \to & \mathsf{1}(x) \\
11\colon & \varepsilon=\varepsilon & \to & \mathsf{tt} \\
12\colon & \mathsf{1}(x)=\mathsf{1}(y) & \to & x=y \\
13\colon & \mathsf{1}(x)=\mathsf{0}(y) & \to & \mathsf{ff} \\
14\colon & \mathsf{0}(x)=\mathsf{1}(y) & \to & \mathsf{ff} \\
15\colon & \mathsf{0}(x)=\mathsf{0}(y) & \to & x=y \\
16\colon & \mathsf{verify}(\mathsf{nil}) & \to & \mathsf{tt} \\
17\colon & \mathsf{verify}(l:ls) & \to & \mathsf{if}(\mathsf{in}(\neg l,ls),\mathsf{ff},\mathsf{verify}(ls)) \\
18\colon & \mathsf{sat}'(a) & \to & \mathsf{if}(\mathsf{verify}(a),a,\mathsf{unsat}) \\
19\colon & \mathsf{sat}(c) & \to & \mathsf{sat}'(\mathsf{guess}(c))
\end{array}
$$

These rules are compatible with POP* and as a result we conclude that the innermost runtime complexity of $\mathcal{R}_{\mathsf{sat}}$ is polynomially bounded.[2]

This leads to the question, whether a characterisation of the runtime complexity of $\mathcal{R}_{\mathsf{sat}}$ suffices to conclude that the functional problem expressed by $\mathcal{R}_{\mathsf{sat}}$ belongs to the class FNP. The purpose of this paper is to provide a positive answer to this question. More precisely, we establish the following results:

- We re-consider graph rewriting and provide a new proof of the adequacy of graph rewriting for full rewriting. This overcomes obvious inefficiencies of rewriting, when it comes to the duplication of terms.
- We provide a precise analysis of the resources needed in implementing graph rewriting on a Turing machine (TM for short).
- Combining these results we obtain an efficient implementation of rewriting on a TM. More precisely, we show that for a given TRS $\mathcal{R}$, for any term s, some normal-form of s is computable in deterministic time $\mathrm{O}(\log(\ell^3)*\ell^7)$, and further, any normal-form of s is computable in nondeterministic time $\mathrm{O}(\log(\ell^2)*\ell^5)$. Here ℓ refers to an upper bound on the length of derivations starting from s (the bound ℓ is supposed to be at least as large as the size of s). Based on this implementation our main result on the correspondence between polynomial runtime complexity and polytime computability follows.

Our result strengthens the evidence that the complexity of a rewrite system is truthfully represented through the length of derivations. Furthermore, our result allows the classification of deterministic as well as nondeterministic polytime-computation based on runtime complexity analysis of rewrite systems. This extends previous work (see [3]) that shows that for confluent TRSs, innermost polynomial runtime complexity induces polytime

[2] To our best knowledge $\mathsf{T_C T}$ is currently the only complexity tool that can provide a complexity certificate for the TRS $\mathcal{R}_{\mathsf{sat}}$, compare `http://termcomp.uibk.ac.at`.

computability of the functions defined. Moreover, it extends related work by Dal Lago and Martini [8, 7] that studies the complexity of *orthogonal* TRSs, also applying graph rewriting techniques (c.f. also Section 6).

The paper is structured as follows. In Section 2 we present basic notions, in Section 3 we (briefly) recall the central concepts of our employed notion of graph rewriting. The adequacy theorem is provided in Section 4 and in Section 5 we show how rewriting can be implemented efficiently. Finally we discuss our results in Section 6, where the above application to computational complexity is made precise. Missing proofs are available in the technical report [4].

2. Preliminaries

We assume familiarity with the basics of term rewriting, see [5, 18]. No familiarity with graph rewriting (see [18]) is assumed. Let R be a binary relation on a set S. We write R^+ for the transitive and R^* for the transitive and reflexive closure of R. An element $a \in S$ is *R-minimal* if there exists no $b \in S$ such that $a\ R\ b$. We write $a\ R^!\ b$ if $a\ R^*\ b$ and b is R-minimal.

Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a signature. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted as $\mathcal{T}(\mathcal{F}, \mathcal{V})$ or $\mathcal{T}$ for short. The *size* $|t|$ of a term t is defined as usual. A *term rewrite system* (*TRS* for short) $\mathcal{R}$ over $\mathcal{T}$ is a *finite* set of rewrite rules $l \to r$, such that $l \notin \mathcal{V}$ and $\mathcal{V}\mathrm{ar}(l) \supseteq \mathcal{V}\mathrm{ar}(r)$. We write $\to_{\mathcal{R}}$ for the induced rewrite relation. The set of defined function symbols is denoted as $\mathcal{D}$, while the constructor symbols are collected in $\mathcal{C}$, clearly $\mathcal{F} = \mathcal{D} \cup \mathcal{C}$. We use $\mathrm{NF}(\mathcal{R})$ to denote the set of normal-forms of $\mathcal{R}$. We define the set of *values* $\mathcal{V}\mathrm{al} := \mathcal{T}(\mathcal{C}, \mathcal{V})$, and we define $\mathcal{B} := \{f(v_1, \ldots, v_n) \mid f \in \mathcal{D} \text{ and } v_i \in \mathcal{V}\mathrm{al}\}$ as the set of *basic terms*. Let $\Box$ be a fresh constant. Terms over $\mathcal{F} \cup \{\Box\}$ and $\mathcal{V}$ are called *contexts*. The empty context is denoted as $\Box$. For a context C with n holes, we write $C[t_1, \ldots, t_n]$ for the term obtained by replacing the holes from left to right in C with the terms $t_1, \ldots, t_n$.

A TRS is called *confluent* if for all $s, t_1, t_2 \in \mathcal{T}$ with $s \to^*_{\mathcal{R}} t_1$ and $s \to^*_{\mathcal{R}} t_2$ there exists a term u such that $t_1 \to^*_{\mathcal{R}} u$ and $t_2 \to^*_{\mathcal{R}} u$. The *derivation height* of a terminating term s with respect to $\to$ is defined as $\mathrm{dl}(s, \to) := \max\{n \mid \exists t.\ s \to^n t\}$, where $\to^n$ denotes the n-fold application of $\to$. The *runtime complexity function* $\mathrm{rc}_{\mathcal{R}}$ with respect to a TRS $\mathcal{R}$ is defined as $\mathrm{rc}_{\mathcal{R}}(n) := \max\{\mathrm{dl}(t, \to_{\mathcal{R}}) \mid t \in \mathcal{B} \text{ and } |t| \leqslant n\}$.

3. Term Graph Rewriting

In the sequel we introduce the central concepts of *term graph rewriting* or *graph rewriting* for short. We closely follow the presentation of [3], for further motivation of the presented notions we kindly refer the reader to [3]. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. We keep $\mathcal{R}$ and $\mathcal{F}$ fixed for the remaining of this paper.

A *directed graph* $G = (\mathrm{V}_G, \mathrm{Succ}_G, \mathrm{L}_G)$ over the set $\mathcal{L}$ of *labels* is a structure such that V_G is a finite set, the *nodes* or *vertices*, $\mathrm{Succ}\colon \mathrm{V}_G \to \mathrm{V}_G^*$ is a mapping that associates a node u with an (ordered) sequence of nodes, called the *successors* of u. Note that the sequence of successors of u may be empty: $\mathrm{Succ}_G(u) = []$. Finally $\mathrm{L}_G\colon \mathrm{V}_G \to \mathcal{L}$ is a mapping that associates each node u with its *label* $\mathrm{L}_G(u)$. Typically the set of labels $\mathcal{L}$ is clear from context and not explicitly mentioned. In the following, nodes are denoted by $u, v, \ldots$ possibly followed by subscripts. We drop the reference to the graph G from V_G,

Succ_G, and L_G, i.e., we write $G = (\mathrm{V}, \mathrm{Succ}, \mathrm{L})$ if no confusion can arise from this. Further, we also write $u \in G$ instead of $u \in \mathrm{V}$.

Let $G = (\mathrm{V}, \mathrm{Succ}, \mathrm{L})$ be a graph and let $u \in G$. Consider $\mathrm{Succ}(u) = [u_1, \ldots, u_k]$. We call u_i $(1 \leqslant i \leqslant k)$ the *i-th successor* of u (denoted as $u \xrightarrow{i} u_i$). If $u \xrightarrow{i} v$ for some i, then we simply write $u \rightharpoonup v$. A node v is called *reachable* from u if $u \xrightarrow{*} v$, where $\xrightarrow{*}$ denotes the reflexive and transitive closure of $\rightharpoonup$. We write $\xrightarrow{+}$ for $\rightharpoonup \cdot \xrightarrow{*}$. A graph G is *acyclic* if $u \xrightarrow{+} v$ implies $u \neq v$ and G is *rooted* if there exists a unique node u such that every other node in G is reachable from u. The node u is called the *root* $\mathrm{rt}(G)$ of G. The *size* of G, i.e., the number of nodes, is denoted as $|G|$. The *depth* of G, i.e., the length of the longest path in G, is denoted as $\mathrm{dp}(G)$. We write $G \restriction u$ for the subgraph of G reachable from u.

Let G and H be two term graphs, possibly sharing nodes. We say that G and H are *properly sharing* if $u \in G \cap H$ implies $\mathrm{L}_G(u) = \mathrm{L}_H(u)$ and $\mathrm{Succ}_G(u) = \mathrm{Succ}_H(u)$. If G and H are properly sharing, we write $G \cup H$ for their union.

Definition 3.1. A *term graph* (with respect to $\mathcal{F}$ and $\mathcal{V}$) is an *acyclic* and *rooted* graph $S = (\mathrm{V}, \mathrm{Succ}, \mathrm{L})$ over labels $\mathcal{F} \cup \mathcal{V}$. Let $u \in S$ and suppose $\mathrm{L}(u) = f \in \mathcal{F}$ such that f is k-ary. Then $\mathrm{Succ}(u) = [u_1, \ldots, u_k]$. On the other hand, if $\mathrm{L}(u) \in \mathcal{V}$ then $\mathrm{Succ}(u) = []$. We demand that every variable node is *shared*. That is, for $u \in S$ with $\mathrm{L}(u) \in \mathcal{V}$, if $\mathrm{L}(u) = \mathrm{L}(v)$ for some $v \in \mathrm{V}$ then $u = v$.

Below $S, T, \ldots$ and L, R, possibly followed by subscripts, always denote term graphs. We write $\mathcal{G}\mathrm{raph}$ for the set of all term graphs with respect to $\mathcal{F}$ and $\mathcal{V}$. Abusing notation from rewriting we set $\mathcal{V}\mathrm{ar}(S) := \{u \mid u \in S, \mathrm{L}(u) \in \mathcal{V}\}$, the set of *variable nodes* in S. We define the term $\mathrm{term}(S)$ *represented* by S as follows: $\mathrm{term}(S) := x$ if $\mathrm{L}(\mathrm{rt}(S)) = x \in \mathcal{V}$ and $\mathrm{term}(S) := f(\mathrm{term}(S \restriction u_1), \ldots, \mathrm{term}(S \restriction u_k))$ for $\mathrm{L}(\mathrm{rt}(S)) = f \in \mathcal{F}$ and $\mathrm{Succ}(\mathrm{rt}(S)) = [u_1, \ldots, u_k]$.

We adapt the notion of *positions* in terms to positions in graphs in the obvious way. Positions are denoted as $p, q, \ldots$, possibly followed by subscripts. For positions p and q we write pq for their concatenation. We write $p \leqslant q$ if p is a prefix of q, i.e., $q = pp'$ for some position p'. The size $|p|$ of position p is defined as its length. Let $u \in S$ be a node. The set of *positions* $\mathcal{P}os_S(u)$ of u is defined as $\mathcal{P}os_S(u) := \{\varepsilon\}$ if $u = \mathrm{rt}(S)$ and $\mathcal{P}os_S(u) := \{i_1 \cdots i_k \mid \mathrm{rt}(S) \xrightarrow{i_1} \cdots \xrightarrow{i_k} u\}$ otherwise. The set of all positions in S is $\mathcal{P}os_S := \bigcup_{u \in S} \mathcal{P}os_S(u)$. Note that $\mathcal{P}os_S$ coincides with the set of positions of $\mathrm{term}(S)$. If $p \in \mathcal{P}os_S(u)$ we say that u *corresponds* to p. In this case we also write $S \restriction p$ for the subgraph $S \restriction u$. This is well defined since exactly one node corresponds to a position p. One verifies $\mathrm{term}(S \restriction p) = \mathrm{term}(S)|_p$ for all $p \in \mathcal{P}os_S$. We say that u is (strictly) *above* a position p if u corresponds to a position q with $q \leqslant p$ $(q < p)$. Conversely, the node u is *below* p if u corresponds to q with $p \leqslant q$.

By exploiting different degrees of *sharing*, a term t can often be represented by more than one term graph. Let S be a term graph and let $u \in S$ be a node. We say that u is *shared* if the set of positions $\mathcal{P}os_S(u)$ is not singleton. Note that in this case, the node u represents more than one subterm of $\mathrm{term}(S)$. If $\mathcal{P}os_S(u)$ is singleton, then u is *unshared*. The node u is *minimally shared* if it is either unshared or a variable node (recall that variable nodes are always shared). We say u is *maximally shared* if $\mathrm{term}(S \restriction u) = \mathrm{term}(S \restriction v)$ implies $u = v$ for all nodes $v \in S$. The term graph S is called *minimally sharing* (*maximally sharing*) if all nodes $u \in S$ are minimally shared (maximally shared). Let s be a term. We collect

all minimally sharing term graphs representing s in the set $\triangle(s)$. Maximally sharing term graphs representing s are collected in $\triangledown(s)$.

We now introduce a notion for replacing a subgraph $S\restriction u$ of S by a graph H.

Definition 3.2. Let S be a term graph and let $u, v \in S$ be two nodes. Then $S[u \leftarrow v]$ denotes the *redirection* of node u to v: define the mapping r such that $r(u) := v$ and $r(w) := w$ for all $w \in S\backslash\{u\}$. Set $\mathrm{V}' := (\mathrm{V}_S \cup \{v\})\backslash\{u\}$ and for all $w \in \mathrm{V}'$, $\mathrm{Succ}'(w) := r^*(\mathrm{Succ}_S(w))$ where r^* is the extension of r to sequences. Finally, set $S[u \leftarrow v] := (\mathrm{V}', \mathrm{Succ}', \mathrm{L}_S)$.

Let H be a rooted graph over $\mathcal{F} \cup \mathcal{V}$. We define $S[H]_u := (S[u \leftarrow \mathrm{rt}(H)] \cup H)\restriction v$ where $v = \mathrm{rt}(H)$ if $u = \mathrm{rt}(S)$ and $v = \mathrm{rt}(S)$ otherwise. Note that $S[H]_u$ is again a term graph if $u \notin H$ and H acyclic.

The following notion of *term graph morphism* plays the role of substitutions.

Definition 3.3. Let L and S be two term graphs. A *morphism* from L to S (denoted $m\colon L \to S$) is a function $m\colon \mathrm{V}_L \to \mathrm{V}_S$ such that $m(\mathrm{rt}(L)) = \mathrm{rt}(S)$, and for all $u \in L$ with $\mathrm{L}_L(u) \in \mathcal{F}$, (i) $\mathrm{L}_L(u) = \mathrm{L}_S(m(u))$ and (ii) $m^*(\mathrm{Succ}_L(u)) = \mathrm{Succ}_S(m(u))$.

The next lemma follows essentially from Assertion (ii) of Definition 3.3.

Lemma 3.4. *If $m\colon L \to S$ then for any $u \in L$ we have $m\colon L\restriction u \to S\restriction m(u)$.*

Let $m\colon L \to S$ be a morphism from L to S. The *induced substitution* $\sigma_m\colon \mathcal{V}\mathrm{ar}(L) \to \mathcal{T}$ is defined as $\sigma_m(x) := \mathrm{term}(S\restriction m(u))$ for any $u \in L$ such that $\mathrm{L}(u) = x \in \mathcal{V}$. As an easy consequence of Lemma 3.4 we obtain the following.

Lemma 3.5. *Let L and S be term graphs, and suppose $m\colon L \to S$ for some morphism m. Let σ_m be the substitution induced by m. Then* $\mathrm{term}(L)\sigma_m = \mathrm{term}(S)$.

Proof. The lemma has been shown in [3, Lemma 14]. ∎

We write $S \geqslant_m T$ (or $S \geqslant T$ for short) if $m\colon S \to T$ is a morphism such that for *all* $u \in \mathrm{V}_S$, Property (i) and Property (ii) in Definition 3.3 are fulfilled. For this case, S and T represent the same term. We write $S >_m T$ (or $S > T$ for short) when the graph morphism m is additionally *non-injective*. If both $S \geqslant T$ and $T \geqslant S$ holds then S and T are *isomorphic*, in notation $S \cong T$. Recall that $|S|$ denotes the number of nodes in S.

Lemma 3.6. *For all term graph S and T, $S \geqslant_m T$ implies* $\mathrm{term}(S) = \mathrm{term}(T)$ *and* $|S| \geqslant |T|$. *If further $S >_m T$ holds then $|S| > |T|$.*

Let L and R be two properly sharing term graphs. Suppose $\mathrm{rt}(L) \notin \mathcal{V}\mathrm{ar}(L)$, $\mathcal{V}\mathrm{ar}(R) \subseteq \mathcal{V}\mathrm{ar}(L)$ and $\mathrm{rt}(L) \notin R$. Then the graph $L \cup R$ is called a *graph rewrite rule* (*rule* for short), denoted by $L \to R$. The graph L, R denotes the left-hand, right-hand side of $L \to R$ respectively. A *graph rewrite system* (*GRS* for short) $\mathcal{G}$ is a set of graph rewrite rules.

Let $\mathcal{G}$ be a GRS, let $S \in \mathcal{G}\mathrm{raph}$ and let $L \to R$ be a rule. A rule $L' \to R'$ is called a *renaming* of $L \to R$ with respect to S if $(L' \to R') \cong (L \to R)$ and $\mathrm{V}_S \cap \mathrm{V}_{L' \to R'} = \varnothing$. Let $L' \to R'$ be a renaming of a rule $(L \to R) \in \mathcal{G}$ for S, and let $u \in S$ be a node. We say S *rewrites* to T at *redex* u with rule $L \to R$, denoted as $S \longrightarrow_{\mathcal{G},u,L\to R} T$, if there exists a morphism $m\colon L' \to S\restriction u$ and $T = S[m(R')]_u$. Here $m(R')$ denotes the structure obtained by replacing in R' every node $v \in \mathrm{dom}(m)$ by $m(v) \in S$, where the labels of $m(v) \in m(R')$ are the labels of $m(v) \in S$. We also write $S \longrightarrow_{\mathcal{G},p,L\to R} T$ if $S \longrightarrow_{\mathcal{G},u,L\to R} T$ for position p corresponding to u in S. We set $S \longrightarrow_{\mathcal{G}} T$ if $S \longrightarrow_{\mathcal{G},u,L\to R} T$ for some $u \in S$ and

$(L \to R) \in \mathcal{G}$. The relation $\longrightarrow_{\mathcal{G}}$ is called the *graph rewrite relation* induced by $\mathcal{G}$. Again abusing notation, we denote the set of normal-forms with respect to $\longrightarrow_{\mathcal{G}}$ as $\mathrm{NF}(\mathcal{G})$.

4. Adequacy of Graph Rewriting for Term Rewriting

In earlier work [3] we have shown that graph rewriting is adequate for innermost rewriting without further restrictions on the studied TRS $\mathcal{R}$. In this section we generalise this result to full rewriting. The adequacy theorem presented here (see Theorem 4.15) is not essentially new. Related results can be found in the extensive literature, see for example [18]. In particular, in [17] the adequacy theorem is stated for full rewriting and unrestricted TRSs. In this work, we take a fresh look from a complexity related point of view. We give a new proof of the adequacy of graph rewriting for full rewriting that allows for a precise control of the resources copied. This is essential for the accurate characterisation of the implementation of graph rewriting given in Section 5.

Definition 4.1. The *simulating graph rewrite system* $\mathcal{G}(\mathcal{R})$ of $\mathcal{R}$ contains for each rule $(l \to r) \in \mathcal{R}$ some rule $L \to R$ such that $L \in \triangle(l)$, $R \in \triangle(r)$ and $\mathrm{V}_L \cap \mathrm{V}_R = \mathcal{V}\mathrm{ar}(R)$.

The next two Lemmas establish soundness in the sense that derivations with respect to $\mathcal{G}(\mathcal{R})$ correspond to $\mathcal{R}$-derivations.

Lemma 4.2. *Let S be a term graph and let $L \to R$ be a renaming of a graph rewrite rule for S, i.e., $S \cap R = \varnothing$. Suppose $m\colon L \to S$ for some morphism m and let σ_m be the substitution induced by m. Then* $\mathrm{term}(R)\sigma_m = \mathrm{term}(T)$ *where* $T := (m(R) \cup S) \restriction \mathrm{rt}(m(R))$.

Proof. The lemma has been shown in [3, Lemma 15]. ∎

In Section 2 we introduced $\Box$ as designation of the empty context. Below we write $\Box$ for the unique (up-to isomorphism) graph representing the constant $\Box$.

Lemma 4.3. *Let S and T be two properly sharing term graphs, let $u \in S \setminus T$ and $C =$* $\mathrm{term}(S[\Box]_u)$. *Then* $\mathrm{term}(S[T]_u) = C[\mathrm{term}(T), \ldots, \mathrm{term}(T)]$.

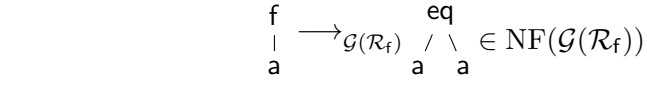
Proof. The lemma has been shown in [3, Lemma 16]. Note that the set of positions of $\Box$ in C corresponds to $\mathcal{P}os_S(u)$. ∎

For non-left-linear TRSs $\mathcal{R}$, $\longrightarrow_{\mathcal{G}(\mathcal{R})}$ does not suffice to mimic $\to_{\mathcal{R}}$. This is clarified in the following example.

Example 4.4. Consider the TRS $\mathcal{R}_{\mathsf{f}} := \{\mathsf{f}(x) \to \mathsf{eq}(x, \mathsf{a});\ \mathsf{eq}(x, x) \to \top\}$. Then $\mathcal{R}_{\mathsf{f}}$ admits the derivation

$$\mathsf{f}(\mathsf{a}) \to_{\mathcal{R}_{\mathsf{f}}} \mathsf{eq}(\mathsf{a}, \mathsf{a}) \to_{\mathcal{R}_{\mathsf{f}}} \top$$

but $\mathcal{G}(\mathcal{R}_{\mathsf{f}})$ cannot completely simulate the above sequence:



Let $L \to R$ be the rule in $\mathcal{G}(\mathcal{R}_{\mathsf{f}})$ corresponding to $\mathsf{eq}(x, x) \to \top$, and let S, $\mathrm{term}(S) = \mathsf{eq}(\mathsf{a}, \mathsf{a})$, be the second graph in the above sequence. Then $L \to R$ is inapplicable as we cannot simultaneously map the unique variable node in L to both leaves in S via a graph morphism. Note that the situation can be repaired by sharing the two arguments in S.

For maximally sharing graphs S we can prove that redexes of $\mathcal{R}$ and (positions corresponding to) redexes of $\mathcal{G}(\mathcal{R})$ coincide. This is a consequence of the following Lemma.

Lemma 4.5. *Let l be a term and $s = l\sigma$ for some substitution σ. If $L \in \triangle(l)$ and $S \in \nabla(s)$, then there exists a morphism $m\colon L \to S$. Further, $\sigma(x) = \sigma_m(x)$ for the induced substitution σ_m and all variables $x \in \mathcal{V}\mathrm{ar}(l)$.*

Proof. We prove the lemma by induction on l. It suffices to consider the induction step. Let $l = f(l_1, \ldots, l_k)$ and $s = f(l_1\sigma, \ldots, l_k\sigma)$. Suppose $\mathrm{Succ}_L(\mathrm{rt}(L)) = [u_1, \ldots, u_k]$ and $\mathrm{Succ}_S(\mathrm{rt}(S)) = [v_1, \ldots, v_k]$. By induction hypothesis there exist morphisms $m_i\colon L\restriction u_i \to S\restriction v_i$ $(1 \leqslant i \leqslant k)$ of the required form. Define $m\colon \mathrm{V}_L \to \mathrm{V}_S$ as follows. Set $m(\mathrm{rt}(L)) = \mathrm{rt}(S)$ and for $w \neq \mathrm{rt}(L)$ define $m(w) = m_i(w)$ if $w \in \mathrm{dom}(m_i)$. We claim $w \in (\mathrm{dom}(m_i) \cap \mathrm{dom}(m_j))$ implies $m_i(w) = m_j(w)$. For this, suppose $w \in (\mathrm{dom}(m_i) \cap \mathrm{dom}(m_j))$. Since $L \in \triangle(l)$, only variable nodes are shared, hence w needs to be a variable node, say $\mathrm{L}_L(w) = x \in \mathcal{V}$. Then

$$\mathrm{term}(S\restriction m_i(w)) = \sigma_{m_i}(x) = \sigma(x) = \sigma_{m_j}(x) = \mathrm{term}(S\restriction m_j(w))$$

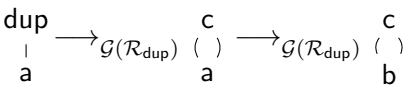
by induction hypothesis. As $S \in \nabla(s)$ is maximally shared, $m_i(w) = m_j(w)$ follows. We conclude m is a well-defined morphism, further $m\colon L \to S$. ∎

A second problem is introduced by non-eager evaluation. Consider the following.

Example 4.6. Let $\mathcal{R}_{\mathsf{dup}} := \{\mathsf{dup}(x) \to \mathsf{c}(x, x);\ \mathsf{a} \to \mathsf{b}\}$. Then $\mathcal{R}_{\mathsf{dup}}$ admits the derivation

$$\mathsf{dup}(\mathsf{a}) \to_{\mathcal{R}_{\mathsf{dup}}} \mathsf{c}(\mathsf{a}, \mathsf{a}) \to_{\mathcal{R}_{\mathsf{dup}}} \mathsf{c}(\mathsf{b}, \mathsf{a})$$

but applying the corresponding rules in $\mathcal{G}(\mathcal{R}_{\mathsf{dup}})$ yields:



Application of the first rule produces a shared redex. Consequently the second step amounts to a parallel step in $\mathcal{R}_{\mathsf{dup}}$.

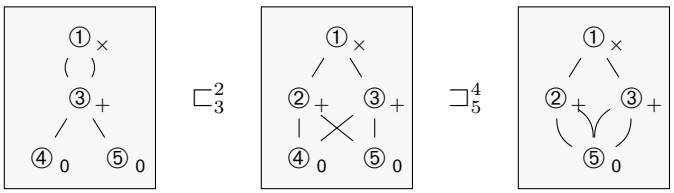
To prove adequacy of graph rewriting for term rewriting and unrestricted TRSs, we follow the standard approach [18, 17] where *folding* (also called *collapsing*) and *unfolding* (also referred to as *copying*) is directly incorporated in the graph rewrite relation. Unlike in the cited literature, we employ a very restrictive form of folding and unfolding. To this extend, we define for positions p relations $\blacktriangleright_p$ and $\lhd_p$ on term graphs. Both relations preserve term structure. However, when $S \blacktriangleright_p T$ holds then the subgraph $T\restriction p$ admits strictly more sharing than $S\restriction p$. Conversely, when $S \lhd_p T$ holds, nodes above p in T admit less sharing than nodes above p in S. Extending the graph rewrite relation $\longrightarrow_{\mathcal{G}(\mathcal{R}),p}$ by $\blacktriangleright_p$ and $\lhd_p$ addresses both problems highlighted in Example 4.4 and Example 4.6.

The relations $\blacktriangleright_p$ and $\lhd_p$ are based on *single step* approximations $\sqsupseteq^u_v$ of $>_m$.

Definition 4.7. Let $\succ$ denote some total quasi-order on nodes, let $\succcurlyeq$ denote the reflexive closure of $\succ$. Let S be a term graph, and let $u, v \in S$ be nodes satisfying $u \succcurlyeq v$. We define $S \sqsupseteq^u_v T$ for term graph T if $S \geqslant_m T$ for the morphism m identifying u and v, more precisely, $m(u) = v$ and $m(w) = w$ for all $w \in S \setminus \{u\}$. We define $S \sqsupset^u_v T$ if $S \sqsupseteq^u_v T$ and $u \neq v$.

We write $S \sqsupseteq_v T$ $(S \sqsupset_v T)$ if there exists $u \in S$ such that $S \sqsupseteq^u_v T$ $(S \sqsupset^u_v T)$ holds. Similar $S \sqsupseteq T$ $(S \sqsupset T)$ if there exist nodes $u, v \in S$ such that $S \sqsupseteq^u_v T$ $(S \sqsupset^u_v T)$ holds.

Example 4.8. Consider the term $t = (0 + 0) \times (0 + 0)$. Then t is represented by the following three graphs that are related by $\sqsubset_3^2$ and $\sqsupset_5^4$ respectively.



T_1 $\sqsubset_3^2$ T_2 $\sqsupset_5^4$ T_3

Put otherwise, the term graph T_2 is obtained from T_1 by copying node 3, introducing the fresh node 2. The graph T_3 is obtained from T_2 by collapsing node 4 onto node 5.

Suppose $S \sqsupset_v^u T$. Then the morphism underlying $\sqsupset_v^u$ defines the identity on $\mathrm{V}_S \setminus \{u\}$. In particular, it defines the identity on successors of $u, v \in S$. Thus the following is immediate.

Lemma 4.9. *Let S be a term graph, and let $u, v \in S$ be two distinct nodes. Then there exists a term graph T such that $S \sqsupset_v^u T$ if and only if $\mathrm{L}_S(u) = \mathrm{L}_S(v)$ and $\mathrm{Succ}_S(u) = \mathrm{Succ}_S(v)$.*

The restriction $u \succcurlyeq v$ was put onto $\sqsupseteq_v^u$ so that $\sqsupseteq_v$ enjoys the following diamond property. Otherwise, the peak $\sqsubset_v^u \cdot \sqsupset_u^v \subseteq \cong$ cannot be joined.

Lemma 4.10. $\sqsubseteq_u \cdot \sqsupseteq_v \;\subseteq\; \sqsupseteq_{w_1} \cdot \sqsubseteq_{w_2}$ *where* $w_1, w_2 \in \{u, v\}$.

Proof. Assume $T_1 \sqsubseteq_u^{u'} S \sqsupseteq_v^{v'} T_2$ for some term graphs S, T_1 and T_2. The only non-trivial case is $T_1 \sqsubset_u^{u'} S \sqsupset_v^{v'} T_2$ for $u' \neq v'$ and $u \neq v$. We prove $T_1 \sqsupset_{w_1} \cdot \sqsubset_{w_2} T_2$ for $w_1, w_2 \in \{u, v\}$ by case analysis. We highlight two interesting cases. The remaining cases follow by similar reasoning, c.f. [4].

- Case $T_1 \sqsubset_w^{u'} S \sqsupset_w^{v'} T_2$ for $v' \neq u'$. We claim $T_1 \sqsupset_w^{v'} \cdot \sqsubset_w^{u'} T_2$. Let m_1 be the morphism underlying $\sqsubset_w^{u'}$ and let m_2 be the morphism underlying $\sqsupset_w^{v'}$ (c.f. Definition 4.7). We first show $\mathrm{L}_{T_1}(v') = \mathrm{L}_{T_1}(w)$ and $\mathrm{Succ}_{T_1}(v') = \mathrm{Succ}_{T_1}(w)$. Using Lemma 4.9, $S \sqsupset_w^{v'} T_2$ yields $\mathrm{L}_S(v') = \mathrm{L}_S(w)$. Employing $v' \neq u'$ and $w \neq u'$ we see
$$\begin{aligned}\mathrm{L}_{T_1}(v') = \mathrm{L}_{T_1}(m_1(v')) &= \mathrm{L}_S(v') \\ &= \mathrm{L}_S(w) = \mathrm{L}_{T_1}(m_1(w)) = \mathrm{L}_{T_1}(w)\ .\end{aligned}$$
where we employ $m_1(v') = v'$ and $m_1(w) = w$. Again by Lemma 4.9, we see $\mathrm{Succ}_S(u') = \mathrm{Succ}_S(w)$ and $\mathrm{Succ}_S(v') = \mathrm{Succ}_S(w)$ by the assumption $T_1 \sqsubset_w^{u'} S \sqsupset_w^{v'} T_2$. We conclude $\mathrm{Succ}_S(v') = \mathrm{Succ}_S(w)$ and thus
$$\begin{aligned}\mathrm{Succ}_{T_1}(v') = \mathrm{Succ}_{T_1}(m_1(v')) &= m_1^*(\mathrm{Succ}_S(v')) \\ &= m_1^*(\mathrm{Succ}_S(w)) = \mathrm{Succ}_{T_1}(m_1(w)) = \mathrm{Succ}_{T_1}(w)\ .\end{aligned}$$
By Lemma 4.9 we obtain term graph U_1 such that $T_1 \sqsupset_w^{v'} U_1$. Symmetrically, we can prove $T_2 \sqsupset_w^{u'} U_2$ for some term graph U_2. Hence $T_1 \sqsupset_w^{v'} \cdot \sqsubset_w^{u'} T_2$ holds if $U_1 = U_2$. To prove the latter, one shows $m_2 \cdot m_1 = m_1 \cdot m_2$ by a straightforward case analysis.
- Case $T_1 \sqsubset_u^{u'} S \sqsupset_v^{v'} T_2$ for pairwise distinct u', u, v' and v. We show $T_1 \sqsupset_v^{v'} \cdot \sqsubset_u^{u'} T_2$. Let m be the morphism underlying $\sqsupset_u^{u'}$. Observe $m(v) = v$ and $m(v') = v'$ by our assumption. Hence $\mathrm{L}_{T_1}(v') = \mathrm{L}_S(v') = \mathrm{L}_S(v) = \mathrm{L}_{T_1}(v)$ and $\mathrm{L}_{T_1}(v') = m^*(\mathrm{L}_S(v')) = m^*(\mathrm{L}_S(v)) = \mathrm{L}_{T_1}(v)$. We obtain $T_1 \sqsupset_v^{v'} U_1$ and symmetrically $T_2 \sqsupset_u^{u'} U_2$ for some term graphs U_1 and U_2. Finally, one verifies $U_1 = U_2$ by case analysis as above. ∎

The above lemma implies confluence of $\sqsupseteq$. Since $\sqsupset^* = \sqsupseteq^*$, $\sqsupset$ is also confluent.

Definition 4.11. Let S be a term graph and let p be a position in S. We say that S *folds strictly below p to the term graph T*, in notation $S \blacktriangleright_p T$, if $S \sqsupset_v^u T$ for nodes $u, v \in S$ strictly below p in S. The graph S *unfolds above p to the term graph T*, in notation $S \lhd_p T$, if $S \sqsubset_v^u T$ for some unshared node $u \in T$ above p, i.e., $\mathcal{P}os_T(u) = \{q\}$ for $q \leqslant p$.

Example 4.12. Reconsider the term graphs T_1, T_2 and T_3 with $T_1 \sqsubset_3^2 T_2 \sqsupset_5^4 T_3$ from Example 4.8. Then $T_1 \lhd_2 T_2$ since node 3 is an unshared node above position 2 in T_2. Further $T_2 \blacktriangleright_2 T_3$ since both nodes 4 and 5 are strictly below position 2 in T_2.

Note that for $S \sqsupset_v^u T$ the sets of positions $\mathcal{P}os_S$ and $\mathcal{P}os_T$ coincide, thus the n-fold composition $\lhd_p^n$ of $\lhd_p$ (and the n-fold composition $\blacktriangleright_p^n$ of $\blacktriangleright_p$) is well-defined for $p \in \mathcal{P}os_S$. In the next two lemmas we prove that relations $\lhd_p$ and $\blacktriangleright_p$ fulfill their intended purpose.

Lemma 4.13. *Let S be a term graph and p a position in S. If S is $\lhd_p$-minimal then the node corresponding to p is unshared.*

Proof. By way of contradiction, suppose S is $\lhd_p$-minimal but the node w corresponding to p is shared. We construct T such that $S \lhd_p T$. We pick an unshared node $v \in S$, and shared node $v_i \in S$, above p such that $v \rightharpoonup v_i$. By a straightforward induction on p we see that v and v_i exist as w is shared. For this, note that at least the root of S is unshared.

Define $T := (\mathrm{V}_T, \mathrm{L}_T, \mathrm{Succ}_T)$ as follows: let u be a fresh node such that $u \succ v_i$. set $\mathrm{V}_T := \mathrm{V}_S \cup \{u\}$; set $\mathrm{L}_T(u) := \mathrm{L}_S(v_i)$ and $\mathrm{Succ}_T(u) := \mathrm{Succ}_S(v_i)$; further replace the edge $v \overset{i}{\rightharpoonup} v_i$ by $v \overset{i}{\rightharpoonup} u$, that is, set $\mathrm{L}_T(v) := [v_1, \ldots, u, \ldots, v_l]$ for $\mathrm{L}_S(v) = [v_1, \ldots, v_i, \ldots, v_l]$. For the remaining cases, define $\mathrm{L}_T(w) := \mathrm{L}_S(u)$ and $\mathrm{Succ}_T(w) := \mathrm{Succ}_S(w)$. One easily verifies $T \sqsupset_{v_i}^u S$. Since by way of construction u is an unshared node above p, $S \lhd_p T$ holds. ∎

Lemma 4.14. *Let S be a term graph, let p be a position in S. If S is $\blacktriangleright_p$-minimal then the subgraph $S \restriction p$ is maximally shared.*

Proof. Suppose $S \restriction p$ is not maximally shared. We show that S is not $\blacktriangleright_p$-minimal. Pick some node $u \in S \restriction p$ such that there exists a distinct node $v \in S \restriction p$ with $\mathrm{term}(S \restriction u) = \mathrm{term}(S \restriction v)$. For that we assume that u is $\rightharpoonup$-*minimal* in the sense that there is no node u' with $u \overset{+}{\rightharpoonup} u'$ such that u' would fulfill the above property. Clearly $\mathrm{L}_S(u) = \mathrm{L}_S(v)$ follows from $\mathrm{term}(S \restriction u) = \mathrm{term}(S \restriction v)$. Next, suppose $u \overset{i}{\rightharpoonup} u_i$ and $v \overset{i}{\rightharpoonup} v_i$ for some nodes $u_i \neq v_i$. But then u_i contradicts minimality of u, and so we conclude $u_i = v_i$. Consequently $\mathrm{Succ}_S(u) = \mathrm{Succ}_S(v)$ follows as desired. Without loss of generality, suppose $u \succ v$. By Lemma 4.9 there exists a term graph T such that $S \sqsupset_v^u T$. Since $u, v \in S \restriction p$, $S \blacktriangleright_p T$ follows. ∎

Theorem 4.15 (Adequacy)**.** *Let s be a term and let S be a term graph such that* $\mathrm{term}(S) = s$*. Then*

$$s \to_{\mathcal{R},p} t \text{ if and only if } S \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$$

for some term graph T with $\mathrm{term}(T) = t$*.*

Proof. First, we consider the direction from right to left. Suppose $S \lhd_p^! U \blacktriangleright_p^! V \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$. Note that $\blacktriangleright_p$ preserves $\lhd_p$-minimality. We conclude V is $\lhd_p$-minimal as U is. Let $v \in V$ be the node corresponding to p. By Lemma 4.13 we see $\mathcal{P}os_U(v) = \{p\}$. Now consider the step $V \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$. There exists a renaming $L' \to R'$ of $(L \to R) \in \mathcal{G}(\mathcal{R})$ such that $m \colon L' \to V \restriction v$ is a morphism and $T = V[m(R')]_v$. Set $l := \mathrm{term}(L')$ and $r := \mathrm{term}(R')$, by

definition $(l \to r) \in \mathcal{R}$. By Lemma 3.5 we obtain $l\sigma_m = \mathrm{term}(V \restriction v)$ for the substitution σ_m induced by the morphism m. Define the context $C := \mathrm{term}(V[\Box]_v)$. As v is unshared, C admits exactly one occurrence of $\Box$, moreover the position of $\Box$ in C is p. By Lemma 4.3,

$$\mathrm{term}(V) = \mathrm{term}(V[V \restriction v]_v) = C[\mathrm{term}(V \restriction v)] = C[l\sigma_m] \,.$$

Set $T_v := (m(R') \cup V) \restriction \mathrm{rt}(m(R'))$, and observe $T = V[m(R')]_v = V[T_v]_v$. Using Lemma 4.3 and Lemma 4.2 we see

$$\mathrm{term}(T) = \mathrm{term}(V[T_v]_v) = C[\mathrm{term}(T_v)] = C[r\sigma_m] \,.$$

As $\mathrm{term}(S) = \mathrm{term}(V)$ by Lemma 3.6, $\mathrm{term}(S) = C[l\sigma_m] \to_{\mathcal{R},p} C[r\sigma_m] = \mathrm{term}(T)$ follows.

Finally, consider the direction from left to right. For this suppose $s = C[l\sigma] \to_{\mathcal{R},p} C[r\sigma] = t$ where the position of the hole in C is p. Suppose $S \lhd_p^! U \blacktriangleright_p^! V$ for $\mathrm{term}(S) = s$. We prove that there exists T such that $V \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$ and $\mathrm{term}(T) = t$. Note that V is $\blacktriangleright_p$-minimal and, as observed above, it is also $\lhd_p$-minimal. Let $v \in V$ be the node corresponding to p, by Lemma 4.13 the node v is unshared. Next, observe $l\sigma = s|_p = \mathrm{term}(S \restriction p) = \mathrm{term}(V \restriction v)$ since $\mathrm{term}(S) = \mathrm{term}(V)$ (c.f. Lemma 3.6). Additionally, Lemma 4.14 reveals $V \restriction v \in \triangledown(l\sigma)$. Further, by Lemma 4.3 we see

$$s = C[l\sigma] = \mathrm{term}(V) = \mathrm{term}(V[V \restriction v]_v) = \mathrm{term}(V[\Box]_v)[l\sigma] \,.$$

Since the position of the hole in C and $\mathrm{term}(V[\Box]_v)$ coincides, we conclude $C = \mathrm{term}(V[\Box]_v)$.

Let $L \to R \in \mathcal{G}(\mathcal{R})$ be the rule corresponding to $(l \to r) \in \mathcal{R}$, let $(L' \to R') \cong (L \to R)$ be a renaming for V. As $L' \in \triangle(l)$ and $V \restriction v \in \triangledown(l\sigma)$, by Lemma 4.5 there exists a morphism $m \colon L' \to V \restriction v$ and hence $V \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$ for $T = V[m(R')]_v$. Note that for the induced substitution σ_m and $x \in \mathcal{V}\mathrm{ar}(l)$, $\sigma_m(x) = \sigma(x)$. Set $T_v := (m(R') \cup V) \restriction \mathrm{rt}(m(R'))$, hence $T = V[T_v]_v$ and moreover $r\sigma = r\sigma_m = \mathrm{term}(T_v)$ follows as in the first half of the proof. Employing Lemma 4.3 we obtain

$$t = C[r\sigma] = \mathrm{term}(V[\Box]_v)[r\sigma] = \mathrm{term}(V[T_v]_v) = \mathrm{term}(T) \,.$$

■

We define $S \mathrel{-\!\!\blacktriangleright\!\!\to}_{\mathcal{G}(\mathcal{R}),p} T$ if and only if $S \lhd_p^! \cdot \blacktriangleright_p^! U \longrightarrow_{\mathcal{G}(\mathcal{R}),p} T$. Employing this notion we can rephrase the conclusion of the Adequacy Theorem as: $s \to_{\mathcal{R},p} t$ if and only if $S \mathrel{-\!\!\blacktriangleright\!\!\to}_{\mathcal{G}(\mathcal{R}),p} T$ for $\mathrm{term}(S) = s$ and $\mathrm{term}(T) = t$.

5. Implementing Term Rewriting Efficiently

Opposed to term rewriting, graph rewriting induces linear size growth in the length of derivations. The latter holds as a single step $\longrightarrow_{\mathcal{G}}$ admits constant size growth:

Lemma 5.1. *If $S \longrightarrow_{\mathcal{G}} T$ then $|T| \leqslant |S| + \Delta$ for some $\Delta \in \mathbb{N}$ depending only on $\mathcal{G}$.*

Proof. Set $\Delta := \max\{|R| \mid (L \to R) \in \mathcal{G}\}$ and the lemma follows by definition. ■

It is easy to see that a graph rewrite step $S \longrightarrow_{\mathcal{G}} T$ can be performed in time polynomial in the size of the term graph S. By the above lemma we obtain that S can be normalised in time polynomial in $|S|$ and the length of derivations. In the following, we prove a result similar to Lemma 5.1 for the relation $\rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}}$, where (restricted) folding and unfolding is incorporated. The main obstacle is that due to unfolding, size growth of $\rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}}$ is not bound by a constant in general. We now investigate the relation $\lhd_p$ and $\blacktriangleright_p$.

Lemma 5.2. *Let S be a term graph and let p be a position in S.*

1) *If $S \lhd_p^\ell T$ then $\ell \leqslant |p|$ and $|T| \leqslant |S| + |p|$.*
2) *If $S \blacktriangleright_p^\ell T$ then $\ell \leqslant |S \restriction p|$ and $|T| \leqslant |S|$.*

Proof. We consider the first assertion. For term graphs U, let $P_U = \{w \mid \mathcal{P}os_U(w) = \{q\} \text{ and } q \leqslant p\}$ be the set of unshared nodes above p. Consider $U \lhd_p V$. Observe that $P_U \subset P_V$ holds: By definition $U \sqsubset_v^u V$ where $\mathcal{P}os_V(u) = \{q\}$ with $q \leqslant p$. Clearly, $P_U \subseteq P_V$, but moreover $u \in P_V$ whereas $u \notin P_U$. Hence for $(S \lhd_p^\ell T) = S = S_0 \lhd_p \ldots \lhd_p S_\ell = T$, we observe $P_S = P_{S_0} \subset \ldots P_{S_\ell} = P_T$. Note that $|P_S| \geqslant 1$ since $\mathrm{rt}(S) \in P_s$. Moreover, $|P_T| = |p| + 1$ since the node corresponding to p in T is unshared (c.f. Lemma 4.13). Thus from $P_{S_i} \subset P_{S_{i+1}}$ $(0 \leqslant i < \ell)$ we conclude $\ell \leqslant |p|$. Next, we see $|T| \leqslant |S| + |p|$ as $|T| = |S| + \ell$ by definition of $\lhd_p$.

Finally, the second assertion can be proved as above, where we employ that $U \lhd_p V$ implies $|V| = |U| - 1$, c.f. the technical report [4]. ∎

By combining the above two lemmas we derive the following:

Lemma 5.3. *If $S \rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}} T$ then $|T| \leqslant |S| + \mathrm{dp}(S) + \Delta$ and $\mathrm{dp}(T) \leqslant \mathrm{dp}(S) + \Delta$ for some $\Delta \in \mathbb{N}$ depending only on $\mathcal{G}$.*

Proof. Consider $S \rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}} T$, i.e., $S \lhd_p^! U \blacktriangleright_p^! V \longrightarrow_{\mathcal{G}} T$ for some position p and term graphs U and V. Lemma 5.2 reveals $|U| \leqslant |S| + |p|$ and further $|V| \leqslant |U|$ for $\Delta := \max\{|R| \mid (L \to R) \in \mathcal{G}\}$. As $|p| \leqslant \mathrm{dp}(S)$ we see $|V| \leqslant |S| + \mathrm{dp}(S)$. Since $V \longrightarrow_{\mathcal{G}} T$ implies $|T| \leqslant |V| + \Delta$ (c.f. Lemma 5.1) we establish $|T| \leqslant |S| + \mathrm{dp}(S) + \Delta$. Finally, $\mathrm{dp}(T) \leqslant \mathrm{dp}(S) + \Delta$ follows from the easy observation that both $U \lhd_p V$ and $U \blacktriangleright_p V$ imply $\mathrm{dp}(U) = \mathrm{dp}(V)$, likewise $V \longrightarrow_{\mathcal{G}} T$ implies $\mathrm{dp}(T) \leqslant V + \Delta$. ∎

Lemma 5.4. *If $S \rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}}^\ell T$ then $|T| \leqslant (\ell+1)|S| + \ell^2\Delta$ for $\Delta \in \mathbb{N}$ depending only on $\mathcal{G}$.*

Proof. We prove the lemma by induction on ℓ. The base case follows trivially, so suppose the lemma holds for ℓ, we establish the lemma for $\ell+1$. Consider a derivation $S \rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}}^\ell T \rightarrowtail\!\!\!\!\!\rightarrow_{\mathcal{G}} U$. By induction hypothesis, $|T| \leqslant (\ell+1)|S| + \ell^2\Delta$. Iterative application of Lemma 5.3 reveals $\mathrm{dp}(T) \leqslant \mathrm{dp}(S) + \ell\Delta$. Thus

$$\begin{aligned} |U| &\leqslant |T| + \mathrm{dp}(T) + \Delta \\ &\leqslant \big((\ell+1)|S| + \ell^2\Delta\big) + \big(\mathrm{dp}(S) + \ell\Delta\big) + \Delta \leqslant (\ell+2)|S| + (\ell+1)^2\Delta\,. \end{aligned}$$

∎

In the sequel, we prove that an arbitrary graph rewrite step $S \rightarrowtail\!\!\!\!\!\rightarrow T$ can be performed in time cubic in the size of S. Lemma 5.4 then allows us to lift the bound on steps to a polynomial bound on derivations in the size of S and the length of derivations. We closely follow the notions of [12]. As model of computation we use k-tape *Turing Machines* (TM for short) with dedicated input- and output-tape. If not explicitly mentioned otherwise, we will use deterministic TMs. We say that a (possibly nondeterministic) TM computes a

relation $R \subseteq \Sigma^* \times \Sigma^*$ if for all $(x, y) \in R$, on input x there exists an accepting run such that y is written on the output tape.

We fix a *standard encoding* for term graphs S. We assume that for each function symbol $f \in \mathcal{F}$ a corresponding tape-symbols is present. Nodes and variables are represented by natural numbers, encoded in binary notation and possibly padded by zeros. We fix the invariant that natural numbers $\{1, \dots, |S|\}$ are used for nodes and variables in the encoding of S. Thus variables (nodes) of S are representable in space $\mathrm{O}(\log(|S|))$. Finally, term graphs S are encoded as a list of *node specifications*, i.e., triples of the form $\langle v, \mathrm{L}(v), \mathrm{Succ}(v)\rangle$ for all $v \in S$ (see [18, Section 13.3]). For a suitable encoding of tuples and lists, a term graph S is representable in size $\mathrm{O}(\log(|S|) * |S|)$. For this, observe that the length of $\mathrm{Succ}(v)$ is bound by the maximal arity of the fixed signature $\mathcal{F}$. In this spirit, we define the *representation size* of a term graph S as $\|S\| := \mathrm{O}(\log(|S|) * |S|)$.

We investigate into the computational complexity of $\lhd_p$ and $\blacktriangleright_p$ first.

Lemma 5.5. *Let S be a term graph and let p a position in S. A term graph T such that $S \lhd_p^! T$ is computable in time $\mathrm{O}(\|S\|^2)$.*

Proof. Suppose $S = S_0 \lhd_p S_1 \lhd_p \cdots \lhd_p S_\ell = T$. By Lemma 5.2, $\ell \leqslant |p| \leqslant |S| \leqslant \|S\|$. One verifies that S_{i+1} is computable from S_i $(0 \leqslant i < \ell)$ in time linear in S_i, and thus linear in S (compare Lemma 5.2). From this it is easy to see that there exists a deterministic TM operating in time quadratic in $\|S\|$ (c.f. the technical report [4]).

■

Lemma 5.6. *Let S be a term graph and p a position in S. The term graph T such that $S \blacktriangleright_p^! T$ is computable in time $\mathrm{O}(\|S\|^2)$.*

Proof. Define the *height* $ht_U(u)$ of a node u in a term graph U inductively as usual: $ht_U(u) := 0$ if $\mathrm{Succ}(u) = []$ and $ht_U(v) := 1 + \max_{v \in \mathrm{Succ}(u)} ht_U(v)$ otherwise. We drop the reference to the graph U in $ht_U(u)$ in the analysis of the normalising sequence $S \blacktriangleright_p^! T$ below. This is justified as the height of nodes remain stable under $\sqsupset$-reductions.

Recall the definition of $\blacktriangleright_p$: $U \blacktriangleright_p V$ if there exist nodes u, v strictly below p with $U \sqsupset_v^u V$. Clearly, for u, v given, the graph V is constructable from U in time linear in $|U|$. However, finding arbitrary nodes u and v such that $U \sqsupset_v^u V$ takes time quadratic in $|U|$ worst case. Since up to linearly many $\sqsupset$-steps in $|S|$ need to be performed, a straightforward implementation admits cubic runtime complexity. To achieve a quadratic bound in the size of the starting graph S, we construct a TM that implements a bottom up reduction-strategy. More precisely, the machine implements the maximal sequence

$$S = S_1 \sqsupset_{u_1}^! S_2 \sqsupset_{u_2}^! \cdots \sqsupset_{u_{\ell-1}}^! S_\ell \tag{a}$$

satisfying, for all $1 \leqslant i < \ell - 1$, (i) either $ht(u_i) = ht(u_{i+1})$ and $u_i \prec u_{i+1}$ or $ht(u_i) < ht(u_{i+1})$, and (ii) for $S_i \sqsupset_{u_i}^{v_{i,1}} \dots \sqsupset_{u_i}^{v_{i,k}} S_{i+1}$, u_i and $v_{i,j}$ $(1 \leqslant j \leqslant k)$ are strictly below p.

By definition $S \blacktriangleright_p^* S_\ell$, it remains to show that the sequence (a) is normalising, i.e., S_ℓ is $\blacktriangleright_p$-minimal. Set $d := \mathrm{dp}(S \restriction p)$ and define, for $0 \leqslant h \leqslant d$,

$$\sqsupset_{(h)} := \bigcup_{u,v \in S \restriction p \wedge ht(v) = h} \sqsupset_v^u .$$

Observe that each $\sqsupset_{u_i}$-step in the sequence (a) corresponds to a step $\sqsupset_{(h)}$ for some $0 \leqslant h \leqslant d$. Moreover, it is not difficult to see that

$$S = S_{i_0} \sqsupset_{(0)}^! S_{i_1} \sqsupset_{(1)}^! \cdots \sqsupset_{(d)}^! S_{i_{d+1}} = S_\ell \tag{b}$$

for $\{S_{i_0}, \ldots, S_{i_d+1}\} \subseteq \{S_1, \ldots, S_{\ell-1}\}$.

Next observe $S_i \sqsupset_{(h_1)} S_{i+1} \sqsupset_{(h_2)} S_{i+2}$ and $h_1 > h_2$ implies $S_i \sqsupset_{(h_2)} \cdot \sqsupset_{(h_1)} S_{i+2}$: suppose $S_i \sqsupset_u^{u'} S_{i+1} \sqsupset_v^{v'} S_{i+2}$ where $ht(u) > ht(v)$ and $u', u, v, v' \in S \restriction p$, we show $S_i \sqsupset_v^{v'} \cdot \sqsupset_u^{u'} S_{i+2}$. Inspecting the proof of Lemma 4.10 we see $\sqsubset_u^{u'} \cdot \sqsupset_v^{v'} \subseteq \sqsupset_v^{v'} \cdot \sqsubset_u^{u'}$ for the particular case that u', u, v and v' pairwise distinct. The latter holds as $ht(u') = ht(u) \neq ht(v) = ht(v')$. Hence it remains to show $S_i \sqsupset_v^{v'} S'_{i+1}$ for some term graph S'_{i+1}, or equivalently $\mathrm{L}_{S_i}(v) = \mathrm{L}_{S_i}(v')$ and $\mathrm{Succ}_{S_i}(v) = \mathrm{Succ}_{S_i}(v')$ by Lemma 4.9. The former equality is trivial, for the latter observe $ht(u') = ht(u) > ht(v) = ht(v')$ and thus neither $u' \notin \mathrm{Succ}_{S_i}(v')$ nor $u' \notin \mathrm{Succ}_{S_i}(v)$. We see $\mathrm{Succ}_{S_i}(v) = \mathrm{Succ}_{S_{i+1}}(v) = \mathrm{Succ}_{S_{i+1}}(v') = \mathrm{Succ}_{S_i}(v')$.

Now suppose that S_ℓ is not $\blacktriangleright_p$-minimal, i.e, $S_\ell \sqsupset_{(h)} U$ for some $0 \leqslant h \leqslant d$ and term graph U. But then we can permute steps in the reduction (b) such that $S_{i_{h+1}} \sqsupset_{(h)} V$ for some term graph V. This contradicts $\sqsupset^!_{(h)}$-minimality of $S_{i_{h+1}}$. We conclude that S_ℓ is $\blacktriangleright_p$-minimal. Thus sequence (a) is $\blacktriangleright_p$-normalising.

Finally, using the derivation (a) it is not difficult to show that there exists a TM operating in time $\mathrm{O}(\|S\|^2)$ that, on input S and p, computes T such that $S \blacktriangleright_p T$. For the construction we kindly refer the reader to the technical report [4]. ∎

Lemma 5.7. *Let S be a term graph, let p be a position of S and let $L \to R$ be a rewrite rule of the simulating graph rewrite system. It is decidable in time $\mathrm{O}(\|S\|^2 * 2^{\|L \to R\|})$ whether $S \longrightarrow_{p, L \to R} T$ for some term graph T. Moreover, the term graph T is computable from S, p and $L \to R$ in time $\mathrm{O}(\|S\|^2 * 2^{\|L \to R\|})$.*

Proof. For the first assertion we can use the *matching*-algorithm as described in [3, Lemma 24]. Based on the morphism returned by this procedure, it is easy to construct a TM that computes the graph T under the stated bound, c.f. the technical report [4]. ∎

Lemma 5.8. *Let S be a term graph and let $\mathcal{G}(\mathcal{R})$ be the simulating graph rewrite system of $\mathcal{R}$. If S is not a normal-form of $\mathcal{G}(\mathcal{R})$ then there exists a position p and rule $(L \to R) \in \mathcal{G}(\mathcal{R})$ such that a term graph T with $S \mathrel{\blacktriangleright\!\!\!\rightarrow}_{\mathcal{G}(\mathcal{R}), p, L \to R} T$ is computable in time $\mathrm{O}(\|S\|^3)$.*

Proof. The TM searches for a rule $(L \to R) \in \mathcal{G}$ and position p such that $S \mathrel{\blacktriangleright\!\!\!\rightarrow}_{\mathcal{G}(\mathcal{R}), p, L \to R} T$ for some term graph T. For this, it enumerates the rules $(L \to R) \in \mathcal{G}$ on a separate working tape. For each rule $L \to R$, each node $u \in S$ and some $p \in \mathcal{P}os_S$ it computes S_1 such that $S \blacktriangleright^!_p S_1$ in time quadratic in $\|S\|$ (c.f. Lemma 5.6). Using the machine of Lemma 5.7, it decides in time $2^{\mathrm{O}(\|L\|)} * \mathrm{O}(\|S_1\|^2)$ whether rule $L \to R$ applies to S_1 at position p. Since $\mathcal{R}$ is fixed, $2^{\mathrm{O}(\|L\|)}$ is constant, thus the TM decides whether rule $L \to R$ applies in time $\mathrm{O}(\|S_1\|^2) = \mathrm{O}(\|S\|^2)$. Note that the choice of $p \in \mathcal{P}os_S(u)$ is irrelevant, since $S \blacktriangleright^!_{p_i} S_1$ and $S \blacktriangleright^!_{p_j} S_2$ for $p_i, p_j \in \mathcal{P}os_S(u)$ implies $S_1 \cong S_2$. Hence the node corresponding to p_i in S_1 is a redex with respect to $L \to R$ if and only if the node corresponding to p_j is. Suppose rule $L \to R$ applies at $S_1 \restriction p$. One verifies $S_1 \restriction p \cong S_2 \restriction p$ for term graph S_2 such that $S \lhd^!_p \cdot \blacktriangleright^!_p S_2$. We conclude $S \mathrel{\blacktriangleright\!\!\!\rightarrow}_{\mathcal{G}(\mathcal{R}), p, L \to R} T$ for some position p and rule $(L \to R) \in \mathcal{G}(\mathcal{R})$ if and only if the above procedure succeeds. From u one can extract some position $p \in \mathcal{P}os_S(u)$ in time quadratic in $\|S\|$. This can be done for instance by implementing the function $\mathsf{pos}(u) = \varepsilon$ if $u = \mathrm{rt}(S)$ and $\mathsf{pos}(u) = pi$ for some node $v \in S$ with $v \stackrel{i}{\rightharpoonup} u$ and $\mathsf{pos}(v) = p$. Overall, the position $p \in \mathcal{P}os_S$ and rule $(L \to R) \in \mathcal{G}$ is found if and only if $S \mathrel{\blacktriangleright\!\!\!\rightarrow}_{p, L \to R} T$ for some term graph T. Since $|S| \leqslant \|S\|$, and only a constant number of rules have to be checked, the overall runtime is $\mathrm{O}(\|S\|^3)$.

To obtain T from S, p, and $L \to R$, the machine now combines the machines from Lemma 5.5, Lemma 5.6 and Lemma 5.7. These steps can even be performed in time $\mathrm{O}(\|S\|^2)$, employing that the size of intermediate graphs is bound linear in the size of S (compare Lemma 5.2) and that sizes of $(L \to R) \in \mathcal{G}(\mathcal{R})$ are constant. ∎

Lemma 5.9. *Let S be a term graph and let $\ell := \mathrm{dl}(S, \twoheadrightarrow_{\mathcal{G}(\mathcal{R})})$. Suppose $\ell = \Omega(|S|)$.*

1) *Some normal-form of S is computable in deterministic time $\mathrm{O}(\log(\ell)^3 * \ell^7)$.*
2) *Any normal-form of S is computable in nondeterministic time $\mathrm{O}(\log(\ell)^2 * \ell^5)$.*

Proof. We prove the first assertion. Consider the normalising derivation

$$S = T_0 \twoheadrightarrow_{\mathcal{G}(\mathcal{R})} \ldots \twoheadrightarrow_{\mathcal{G}(\mathcal{R})} T_l = T \qquad (\dagger)$$

where, for $0 \leqslant i < l$, T_i is obtained from T_{i+1} as given by Lemma 5.8. By Lemma 5.4, we see $|T_i| \leqslant (\ell + 1)|S| + \ell^2\Delta = \mathrm{O}(\ell^2)$. Here the latter equality follows by the assumption $\ell = \Omega(|S|)$. Recall $\|T_i\| = \mathrm{O}(\log(|T_i|) * |T_i|)$ $(0 \leqslant i < l)$ and hence $\|T_i\| = \mathrm{O}(\log(\ell^2) * \ell^2) = \mathrm{O}(\log(\ell) * \ell^2)$. From this, and Lemma 5.8, we obtain that T_{i+1} is computable from T_i in time $\mathrm{O}(\|T_i\|^3) = \mathrm{O}(\log(\ell)^3 * \ell^6)$. Since $l \leqslant \mathrm{dl}(S, \twoheadrightarrow_{\mathcal{G}(\mathcal{R})}) = \ell$ we conclude the first assertion.

We now consider the second assertion. Reconsider the proof of Lemma 5.8. For a given rewrite-position p, a step $S \twoheadrightarrow_{\mathcal{G}(\mathcal{R})} T$ can be performed in time $\mathrm{O}(\|S\|^2)$. A nondeterministic TM can guess some position p, and verify whether the node corresponding to p is a redex in time $\mathrm{O}(\|S\|^2)$. In total, the reduct T can be obtained in nondeterministic time $\mathrm{O}(\|S\|^2)$. Hence, following the proof of the first assertion, one easily verifies the second assertion. ∎

6. Discussion

We present an application of our result in the context of *implicit computational complexity theory*. We define semantics of TRSs as follows.

Definition 6.1. Let $\mathcal{N} \subseteq \mathcal{V}\mathrm{al}$ be a finite set of *non-accepting patterns*. We call a term t *accepting* (with respect to $\mathcal{N}$) if there exists no $p \in \mathcal{N}$ such that $p\sigma = t$ for some substitution σ. We say that $\mathcal{R}$ *computes the relation* $R \subseteq \mathcal{V}\mathrm{al} \times \mathcal{V}\mathrm{al}$ with respect to $\mathcal{N}$ if there exists $\mathsf{f} \in \mathcal{D}$ such that for all $s, t \in \mathcal{V}\mathrm{al}$,

$$s \; R \; t :\Longleftrightarrow \mathsf{f}(s) \to^{!}_{\mathcal{R}} t \text{ and } t \text{ is accepting} .$$

On the other hand, we say that a relation R is computed by $\mathcal{R}$ if R is defined by the above equations with respect to *some* set $\mathcal{N}$ of non-accepting patterns.

For the case that $\mathcal{R}$ is *confluent* we also say that $\mathcal{R}$ computes the (partial) *function* induced by the relation R. The reader may wonder why we restrict to binary relations, but this is only a non-essential simplification that eases the presentation. The assertion that for normal-forms t, t is accepting amounts to our notion of *accepting run* of a TRS $\mathcal{R}$. This aims to eliminate by-products of the computation that should not be considered as part of the relation R. (A typical example would be the constant $\bot$ if the TRS contains a rule of the form $l \to \bot$ and $\bot$ is interpreted as *undefined*.) The restriction that $\mathcal{N}$ is finite is essential for the simulation results below: If we implement the computation of $\mathcal{R}$ on a TM, then we also have to be able to effectively test whether t is accepting.

We briefly contrast Definition 6.1 to the way how semantics is given to TRSs in [6]. Basically, in [6] the result of a computation is defined as the *maximal* normal-form with respect

to some ordering on terms. So even non-confluent TRSs compute functions. This definition serves the purpose of characterising *optimisation problems*. In particular, restricted polynomial interpretations are used to characterise the class of functions OptP as introduced in [13]. Intuitively, an OptP function is computed by an NP machine that, at the end, outputs the *maximal* result of all accepting computation branches. Our intention is not to capture optimisation problems. Instead, we show below a tight correspondence between polynomial runtime-complexity and the class FNP of *functional problems* [16] associated with NP. It is well-known that $\mathsf{FNP} \subseteq \mathsf{OptP}$ and it is expected that the inclusion is strict, see [13, 16].

First, we show that polynomial runtime-complexity implies polytime computability of the relations defined in the sense of Definition 6.1. For this, we encode terms as graphs and perform rewriting on graphs instead.

Theorem 6.2. *Let $\mathcal{R}$ be a terminating TRS, moreover suppose $\mathrm{rc}_{\mathcal{R}}(n) = \mathrm{O}(n^k)$ for all $n \in \mathbb{N}$ and some $k \in \mathbb{N}$, $k \geqslant 1$. The relations computed by $\mathcal{R}$ are computable in nondeterministic time $\mathrm{O}(n^{5k+2})$. Further, if $\mathcal{R}$ is confluent then the functions computed by $\mathcal{R}$ are computable in deterministic time $\mathrm{O}(n^{7k+3})$.*

Proof. We investigate the complexity of a relation R computed by $\mathcal{R}$. For that, single out the corresponding defined function symbol f and fix some argument $s \in \mathcal{V}\mathrm{al}$. Suppose the underlying set of non-accepting patterns is $\mathcal{N}$. By definition, $s\ R\ t$ if and only if $\mathsf{f}(s) \to^{!}_{\mathcal{R}} t$ and $t \in \mathcal{V}\mathrm{al}$ is accepting with respect to $\mathcal{N}$. Let S be a term graph such that $\mathrm{term}(S) = \mathsf{f}(s)$ and recall that $|S| \leqslant |\mathsf{f}(s)|$. Set $\ell := \mathrm{dl}(S, \twoheadrightarrow_{\mathcal{G}(\mathcal{R})})$. By the Adequacy Theorem 4.15, we conclude $S \twoheadrightarrow^{!}_{\mathcal{G}(\mathcal{R})} T$ where $\mathrm{term}(T) = t$, and moreover, $\ell \leqslant \mathrm{rc}_{\mathcal{R}}(|\mathsf{f}(s)|) = \mathrm{O}(n^k)$. By Lemma 5.9 we see that T is computable from S in nondeterministic time $\mathrm{O}(\log(\ell)^2 * \ell^5) = \mathrm{O}(\log(n^k)^2 * n^{5k}) = \mathrm{O}(n^{5k+2})$. Clearly, we can decide in time linear in $\|T\| = \mathrm{O}(\ell^2) = \mathrm{O}(n^{2k})$ (c.f. Lemma 5.4) whether $\mathrm{term}(T) \in \mathcal{V}\mathrm{al}$, further in time quadratic in $\|T\|$ whether $\mathrm{term}(T)$ is accepting. For the latter, we use the matching algorithm of Lemma 5.7 on the fixed set of non-accepting patterns, where we employ $p\sigma = \mathrm{term}(T)$ if and only if there exists a morphism $m\colon P \to T$ for $P \in \triangle(p)$ (c.f. Lemma 4.5 and Lemma 3.5). Hence overall, the accepting condition can be checked in (even deterministic) time $\mathrm{O}(n^{4k})$. If the accepting condition fails, the TM rejects, otherwise it accepts a term graph T representing t. The machine does so in nondeterministic time $\mathrm{O}(n^{5k+2})$ in total. As s was chosen arbitrarily, we conclude the first half of the theorem.

Finally, the second half follows by identical reasoning, where we use the deterministic TM as given by Lemma 5.9 instead of the nondeterministic one. ∎

Let R be a binary relation that is decidable by some *nondeterministic* TM N. That is, the pair (x, y) is accepted by N if and only if $x\ R\ y$ holds. Furthermore, suppose N operates in time polynomial in the size of x. The *function problems* R_F associated with R is: given x, find *some* y such that $x\ R\ y$ holds. The class FNP is the class of all functional problems defined in the above way, compare [16]. FP is the subclass resulting if we only consider function problems in FNP that can be solved in polynomial time by some deterministic TM. As by-product of Theorem 6.2 we obtain:

Corollary 6.3. *Let $\mathcal{R}$ be a terminating TRS with polynomially bounded runtime complexity. Suppose $\mathcal{R}$ computes the relation R. Then $R_F \in \mathsf{FNP}$ for the function problem R_F associated with R. Moreover, if $\mathcal{R}$ is confluent then $R_F \in \mathsf{FP}$.*

Proof. The nondeterministic TM N as given by Theorem 6.2 (the deterministic TM N, respectively) can be used to decide whether $s\ R\ t$ holds. By the assumptions on $\mathcal{R}$, the

runtime of N is bounded polynomially in the size of s. Recall that s is represented as some term graph S, in particular s is encoded over the alphabet of N in size $\|S\| = \mathrm{O}(\log(|S|)*|S|)$ for $|S| \leqslant |s|$. Thus trivially N operates in time polynomially in the size of S. ∎

In the terminology of [7], we have shown that the number of rewrite steps is an *invariant cost model* for term rewriting. Not only does it reflect the complexity of a TRS in a very natural way, but in fact it truthfully reflects the complexity of rewriting on the standard computational model in complexity theory, the Turing machine. In [7] our result is proved for orthogonal TRSs and innermost or respectively outermost rewriting. Hence our work can be seen as a direct extension of [7], establishing that neither the restriction to orthogonality nor the use of particular reduction-strategies is essential. Based on [7], Dal Lago and Martini establish in [8] that the number of β-steps constitutes an invariant cost model for the *weak* λ-calculus (here reduction below λ-abstractions are disallowed), when reducing under a *call-by-value* or *call-by-need* reduction strategy. Their approach works by embedding β-steps into the rewrite relation as induced by specific orthogonal TRSs, and analysing the complexity of the latter relation. This raises the question whether our result can be used to extend the work by Dal Lago and Martini on λ-calculus. This is subject to further research.

References

[1] M. Avanzini and G. Moser. Complexity Analysis by Rewriting. In *Proc. of 9th FLOPS*, volume 4989 of *LNCS*, pages 130–146. Springer Verlag, 2008.

[2] M. Avanzini and G. Moser. Dependency Pairs and Polynomial Path Orders. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 48–62. Springer Verlag, 2009.

[3] M. Avanzini and G. Moser. Complexity Analysis by Graph Rewriting. In *Proc. of 11th FLOPS*, LNCS. Springer Verlag, 2010. To appear.

[4] M. Avanzini and G. Moser. Technical report: Complexity Analysis by Graph Rewriting Revisited. *CoRR*, cs/CC/1001.5404, 2010. Available at `http://www.arxiv.org/`.

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

[6] G. Bonfante, A. Cichon, J. Marion, and H. Touzet. Algorithms with Polynomial Interpretation Termination Proof. *JFP*, 11(1):33–53, 2001.

[7] U. Dal Lago and S. Martini. Derivational Complexity is an Invariant Cost Model. In *Proc. of 1st FOPARA*, 2009.

[8] U. Dal Lago and S. Martini. On Constructor Rewrite Systems and the Lambda-Calculus. In *Proc. of 36th ICALP*, volume 5556 of *LNCS*, pages 163–174. Springer Verlag, 2009.

[9] J. Endrullis, J. Waldmann, and H. Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. *JAR*, 40(3):195–220, 2008.

[10] A. Koprowski and J. Waldmann. Arctic Termination ... Below Zero. In *Proc. of 19th RTA*, volume 5117 of *LNCS*, pages 202–216. Springer Verlag, 2008.

[11] M. Korp and A. Middeldorp. Match-bounds revisited. *IC*, 207(11):1259–1283, 2009.

[12] D. C. Kozen. *Theory of Computation*. Springer Verlag, first edition, 2006.

[13] M. W. Krentel. The Complexity of Optimization Problems. In *Proc. of 18th STOC*, pages 69–76. ACM, 1986.

[14] G. Moser and A. Schnabl. The Derivational Complexity Induced by the Dependency Pair Method. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 255–260. Springer Verlag, 2009.

[15] G. Moser, A. Schnabl, and J. Waldmann. Complexity Analysis of Term Rewriting Based on Matrix and Context Dependent Interpretations. In *Proc. of 28th FSTTCS*, pages 304–315. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[16] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, second edition, 1995.

[17] D. Plump. Essentials of Term Graph Rewriting. *ENTCS*, 51:277–289, 2001.

[18] TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracks in Theoretical Computer Science*. Cambridge University Press, 2003.