Logical Methods in Computer Science
Vol. 7 (3:01) 2011, pp. 1–38
www.lmcs-online.org

Submitted Oct. 11, 2010
Published Jul. 12, 2011

THE DERIVATIONAL COMPLEXITY INDUCED BY THE DEPENDENCY PAIR METHOD

GEORG MOSER [a] AND ANDREAS SCHNABL [b]

Institute of Computer Science, University of Innsbruck, Austria
e-mail address: georg.moser@uibk.ac.at, andreas.schnabl@uibk.ac.at

Abstract. We study the derivational complexity induced by the dependency pair method, enhanced with standard refinements. We obtain upper bounds on the derivational complexity induced by the dependency pair method in terms of the derivational complexity of the base techniques employed. In particular we show that the derivational complexity induced by the dependency pair method based on some direct technique, possibly refined by argument filtering, the usable rules criterion, or dependency graphs, is primitive recursive in the derivational complexity induced by the direct method. This implies that the derivational complexity induced by a standard application of the dependency pair method based on traditional termination orders like KBO, LPO, and MPO is exactly the same as if those orders were applied as the only termination technique.

1. Introduction

Several notions to assess the complexity of a terminating term rewrite system (TRS) have been proposed in the literature, compare [7, 8, 16, 20]. The conceptually simplest one was suggested by Hofbauer and Lautemann in [20]: the complexity of a given TRS is measured as the maximal length of derivation sequences. More precisely, the *derivational complexity function* with respect to a terminating TRS $\mathcal{R}$ relates the maximal derivation height to the size of the initial term. We adopt this notion as our central definition of the complexity of a TRS.

For termination proofs by direct methods a considerable number of results establish essentially optimal upper bounds on the growth rate of the derivational complexity function. See for example [18, 20, 22, 26, 29, 30, 31, 37, 38] for results in this direction. However, for transformation techniques like semantic labelling [40] or the dependency pair method [1] the situation changes. For semantic labelling, it is a trivial observation that the derivational complexity of the original TRS is bounded from above by the derivational complexity of the labelled system. However, if the domain of the used (quasi-)models is infinite, the labelled

1998 ACM Subject Classification: F.4.1, F.2.2, D.2.4, D.2.8.
Key words and phrases: derivational complexity analysis, termination, dependency pair method.
A preliminary version of this article appeared as [27].
[a,b] Supported by FWF (Austrian Science Fund) project P20133-N15.
[b] Supported by a grant of the University of Innsbruck.

DOI:10.2168/LMCS-7 (3:01) 2011

TRS is generally infinite, as well. Estimating the derivational complexity of such systems is harder than for finite systems: for some termination proof methods, such as the multiset path order (MPO for short) or the lexicographic path order (LPO for short) the complexity results only hold for finite TRSs [18, 38]. For the Knuth-Bendix order (KBO for short) the situation is better. If some weak conditions are in place, then the bound on the derivational complexity with respect to finite TRSs extends to infinite TRSs [25]. With respect to the dependency pair method, in [2, 16, 17, 23, 24, 32, 39] the bounds on derivation heights induced by the dependency pair method or its framework are investigated. However only variations on the original definition of the dependency pair method were analysed.

In this paper we give a derivational complexity analysis of the dependency pair method. It should be emphasised that the notion of dependency pair method studied here amounts to the original technique as introduced by Arts and Giesl [1] (see also Hirokawa and Middeldorp [14]). As the dependency pair method is a transformation technique, we can only give a parametrised analysis. We call those techniques that are applied on the transformed system: *base* techniques. Let us exemplify this notation on the next example.

Example 1.1. Consider the TRS $\mathcal{R}_1$ given below:

$$\begin{aligned} \mathsf{i}(x) \circ (y \circ z) &\to \mathsf{f}(x, \mathsf{i}(x)) \circ (\mathsf{i}(\mathsf{i}(y)) \circ z) & \mathsf{i}(x) &\to x \\ \mathsf{i}(x) \circ (y \circ (z \circ w)) &\to \mathsf{f}(x, \mathsf{i}(x)) \circ (z \circ (y \circ w)) & \mathsf{f}(x, y) &\to x \ . \end{aligned}$$

$\mathcal{R}_1$ is a variation of a TRS encoding the Ackermann function, introduced by Hofbauer [20, Proposition 5.9] (also compare [19]). Note that $\mathcal{R}_1$ is not simply terminating and the derivational complexity of $\mathcal{R}_1$ grows as fast as the Ackermann function. However, termination can be (automatically) shown by the dependency pair method in conjunction with argument filtering and KBO (we give all necessary definitions in Sections 2 and 3).

In Example 1.1 we cannot apply KBO directly (the TRS $\mathcal{R}_1$ is not simply terminating), but we apply KBO as base technique. In order to measure the strength of the dependency pair method itself, we express the induced complexity relative to the (maximal) complexities of the base techniques. With respect to Example 1.1 it is not difficult to see that the derivational complexity of $\mathcal{R}_1$ belongs to $\mathrm{Ack}(\Theta(n), 0)$, where n is the size of the start term. Essentially this follows from [19, Proposition 5.9], due to the closeness of $\mathcal{R}_1$ to Hofbauer's original example. As this is also the complexity induced by KBO [22] it may appear that the dependency pair method does not add any power. Our results provide a clear picture of the true connection. With respect to upper bounds on the derivational complexity, we establish the following (technical) results:

(1) For the basic dependency pair method (potentially using argument filterings) the induced derivational complexity is bounded triple exponentially in the derivational complexity of the base technique used. If we restrict to string rewrite systems, then the induced derivational complexity is exponential in the derivational complexity of the base technique.
(2) If we consider the basic dependency pair method using the usable rules refinement, then the induced derivational complexity is primitive recursive in the derivational complexity of the base technique.
(3) Finally, if we consider the dependency pair method in conjunction with dependency graphs, then the induced derivational complexity is primitive recursive in the (maximal) derivational complexity of the base techniques employed.

Complementing these results, we present results on lower bounds. For the basic dependency pair method, we present an example which shows that at least two of the three exponentials in its upper bound can actually be reached. If we restrict to string rewriting, this bound reduces to a single exponential. Hence the corresponding upper bound mentioned in result (1) is optimal. For the usable rules refinement, we show that the growth rate of the derivational complexity function may be nonelementary. Furthermore we show that the bound for dependency graphs given by result (3) is essentially optimal.

To exemplify these results, we momentarily focus on polynomial interpretations as base technique. It is well-known that polynomial interpretations induce a double exponential bound on the derivational complexity [20]. Let $\mathcal{R}$ be a TRS and suppose termination of $\mathcal{R}$ has been established by applying the basic dependency pair method, where polynomial interpretations are used to define the employed reduction pair. According to result (1) the derivational complexity function with respect to $\mathcal{R}$ is bounded by $2_5(\mathsf{O}(n))$, i.e, by a tower of 2s of height 5 in n. On the other hand, if in addition the usable rules criterion or dependency graphs are used, then results (2) and (3) yield that the derivational complexity is (at most) primitive recursive.

Thus seemingly easy refinements of the dependency pair method like dependency graphs may lead to noteworthy speed-ups of the growth rates of the derivational complexity function. On the other hand if strong techniques (with respect to the complexity induced) are employed in conjunction with the dependency pair method, then the derivational complexity of the analysed TRS may only depend on the complexity induced by the base technique.

Re-consider the TRS $\mathcal{R}_1$ given in Example 1.1. There are nine dependency pairs.

$$\begin{array}{ll}
\mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{f}(x, \mathsf{i}(x)) \circ^\sharp (\mathsf{i}(\mathsf{i}(y)) \circ z) & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to \mathsf{f}(x, \mathsf{i}(x)) \circ^\sharp (z \circ (y \circ w)) \\
\mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{f}^\sharp(x, \mathsf{i}(x)) & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to \mathsf{f}^\sharp(x, \mathsf{i}(x)) \\
\mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{i}(\mathsf{i}(y)) \circ^\sharp z & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to z \circ^\sharp (y \circ w) \\
\mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{i}^\sharp(\mathsf{i}(y)) & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to y \circ^\sharp w \\
\mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{i}^\sharp(y) \; . &
\end{array}$$

To show termination of $\mathcal{R}_1$ one may use the argument filtering π: $\pi(\mathsf{f}) = \pi(\mathsf{f}^\sharp) = \pi(\mathsf{i}^\sharp) = 1$, $\pi(\mathsf{i}) = [1]$, $\pi(\circ) = \pi(\circ^\sharp) = [1, 2]$ and the reduction pair $(\gtrsim^\pi_{\mathsf{KBO}}, >^\pi_{\mathsf{KBO}})$, where $(\gtrsim^\pi_{\mathsf{KBO}}, >^\pi_{\mathsf{KBO}})$ is induced by the (admissible) weight function w with $w_0 = 1$, $\mathsf{w}(\circ) = \mathsf{w}(\circ^\sharp) = 1$, and $\mathsf{w}(\mathsf{i}) = 0$. Furthermore the precedence $\succ$ fulfils: $\mathsf{i} \succ \circ, \circ^\sharp$. Due to result (1) and [22] the derivational complexity of $\mathcal{R}_1$ belongs to $\mathrm{Ack}(\Theta(n), 0)$.

In contrast to the case for polynomial interpretations, the complexity induced by the base technique belongs to a class of functions closed under primitive recursion. Hence it is already so huge, that the inherent complexity of the dependency pair method becomes negligible.

Note the challenges of our investigation: In order to estimate the derivational complexity of a rewrite system we only consider the (maximal) derivation complexities induced by the base techniques employed. This entails that we exploit the upper bound on the maximal number of dependency pair steps to bound the length of derivations.

Some of the results in this paper appeared in an earlier conference paper [27]. Apart from correcting some shortcomings of the conference paper the journal version extends [27] by providing a full analysis of the dependency graph refinement (see Section 8). Furthermore

the treatment of the usable rules criterion is new (see Section 7), as well as the improvement of the lower- and upper-bound in the context of string rewriting (see Section 6).

The technically most involved result is the proof of the triple exponential upper bound for the basic dependency pair method. Our proof rests on the observation that it suffices to bound the maximal depth of a term during a given derivation. We show that the depth of any term occurring in a derivation is bounded exponentially in the maximal number of dependency pair steps. Based on this result the triple exponential upper bound follows by standard observations. Due to this ground work the analysis of the usable rules refinement is relatively straightforward. On the other hand, for the analysis of the dependency graph refinement we employ a different, but conceptually simpler technique. Essentially, it suffices to embed the analysed TRS in a generic simulating TRS whose derivational complexity can be analysed directly.

The rest of this paper is organised as follows. In Sections 2 and 3 we present basic notions and starting points of the paper. Sections 4 and 5 establish result (1). The mentioned improvement for string rewriting is given in Section 6. In Section 7, we extend our considerations to usable rules and thus show result (2). In Section 8, we consider dependency graphs and show result (3). Finally we conclude in Section 9. To ease the presentation some technical results have been moved to the appendix.

2. Preliminaries

We assume familiarity with the basics of term rewriting, see [4, 34]. Below we recall central definitions and notions of rewriting which are relevant to this paper.

Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a signature of function symbols with fixed arities. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted as $\mathcal{T}(\mathcal{F},\mathcal{V})$. The set of ground terms over $\mathcal{F}$ is denoted as $\mathcal{T}(\mathcal{F})$. The (proper) subterm relation is denoted as $\trianglelefteq$ ($\triangleleft$); we write $\trianglerighteq$ ($\triangleright$) for the reversed (proper) subterm relation. The *root symbol* (denoted as $\mathsf{rt}(t)$) of a term t is either t itself, if $t \in \mathcal{V}$, or the symbol f, if $t = f(t_1,\dots,t_n)$. We denote the set of variables occurring in a term t as $\mathcal{V}\mathsf{ar}(t)$, and the set of function symbols occurring in t as $\mathcal{F}\mathsf{un}(t)$. A *position* is a finite sequence of positive integers. The root position is the empty sequence denoted by ϵ, and pq denotes the concatenation of positions p and q. The set of positions of a term t is denoted as $\mathcal{P}\mathsf{os}(t)$. We write $p \leqslant q$ ($p < q$) to denote that p is a (proper) prefix of q, and $p \parallel q$ if neither $p \leqslant q$ nor $q \leqslant p$. The subterm of t at position p is denoted as $t|_p$. We write $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ ($\mathcal{P}\mathsf{os}_{\mathcal{V}}(t)$) for the set of positions p such that $\mathcal{F}$ ($\mathcal{V}$) contains $\mathsf{rt}(t|_p)$. To simplify the exposition, we often confuse terms and their tree representations: a *branch* of a term t is a maximal set of positions B in t such that for all pairs of positions $q, q' \in B$, we have $q \leqslant q'$ or $q' \leqslant q$. The *size* (denoted as $|t|$) of a term t is the number of variables and function symbols occurring in t. The *depth* (denoted as $\mathsf{dp}(t)$) of a term t is 0 if t is a variable or a constant, and defined as follows if $t = f(t_1,\dots,t_n)$: $\mathsf{dp}(t) := 1 + \max\{\mathsf{dp}(t_i) \colon 1 \leqslant i \leqslant n\}$. A *substitution* is a mapping $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F},\mathcal{V})$. The result of applying a substitution σ to a term t is denoted as $t\sigma$. We introduce a fresh constant $\Box$ (the *hole*) and define a *context* C as a term (over $\mathcal{F} \cup \{\Box\}$ and $\mathcal{V}$) containing exactly one $\Box$. For a term t and a context C, $C[t]$ denotes the replacement of $\Box$ by t.

A *term rewrite system* (*TRS* for short) $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is a *finite* set of rewrite rules $l \to r$ with $l, r \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $l \notin \mathcal{V}$, and $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(l)$. Given a TRS $\mathcal{R}$ and two terms s, t, we say that s *rewrites* to t (denoted as $s \to_{\mathcal{R}} t$) if there exist a context C, a substitution σ and a rewrite rule $l \to r$ in $\mathcal{R}$ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. If no confusion can

arise, we write $s \to t$, instead. We write $\to_{\mathcal{R}}^{+}$ for the transitive closure of this relation. The reflexive closure is $\to_{\mathcal{R}}^{=}$. The reflexive and transitive closure is denoted as $\to_{\mathcal{R}}^{*}$. We write $\to_{\mathcal{R}}^{n}$ to express n-fold composition of $\to_{\mathcal{R}}$. If we wish to indicate the redex position p and the applied rewrite rule $l \to r$ in a reduction from s to t, we write $s \to_{p,l\to r} t$. A TRS $\mathcal{R}$ is *terminating* if there exists no infinite chain of terms $t_0, t_1, \ldots$ such that $t_i \to_{\mathcal{R}} t_{i+1}$ for each $i \in \mathbb{N}$.

A function symbol f is *defined* if $f = \mathsf{rt}(l)$ for some rewrite rule $l \to r$ in the considered TRS $\mathcal{R}$, otherwise it is called a *constructor*. The set of defined function symbols of $\mathcal{R}$ is denoted as $\mathcal{D}_{\mathcal{R}}$, while the constructor symbols are collected in $\mathcal{C}_{\mathcal{R}}$ (we only write $\mathcal{D}$ and $\mathcal{C}$, respectively, if no confusion can arise). We write $\mathcal{P}\mathsf{os}_{\mathcal{D}}(t)$ ($\mathcal{P}\mathsf{os}_{\mathcal{C}}(t)$) for the set of positions p such that $\mathcal{D}$ ($\mathcal{C}$) contains $\mathsf{rt}(t|_p)$. We recall the notion of *relative rewriting*, c.f. [11, 34]. Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs. We write $\to_{\mathcal{R}/\mathcal{S}}$ for $\to_{\mathcal{S}}^{*} \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^{*}$ and we call $\to_{\mathcal{R}/\mathcal{S}}$ the *relative rewrite relation* of $\mathcal{R}$ modulo $\mathcal{S}$. Clearly, we have that $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{R}}$, if $\mathcal{S} = \varnothing$. We write $\mathsf{NF}(\mathcal{R})$, $\mathsf{NF}(\mathcal{R}/\mathcal{S})$ to denote the set of normal forms of $\to_{\mathcal{R}}$, $\to_{\mathcal{R}/\mathcal{S}}$ respectively.

The *derivation height* of a term s with respect to a finitely branching, well-founded binary relation $\to$ on terms is defined as $\mathsf{dh}(s, \to) := \max\{n \mid \exists t\ s \to^n t\}$. The *derivational complexity function* of $\mathcal{R}$ is defined as:

$$\mathsf{dc}_{\mathcal{R}}(n) := \max\{\mathsf{dh}(t, \to_{\mathcal{R}}) \mid |t| \leqslant n\} .$$

In analogy to the mapping dh we define functions tracing the depth or size of reducts. The *potential depth* of a term s with respect to $\to$ is defined as follows: $\mathsf{pdp}(s, \to) := \max\{\mathsf{dp}(t) \mid s \to^* t\}$; the *potential size* is defined by $\mathsf{psz}(s, \to) := \max\{|t| \mid s \to^* t\}$. If termination of $\mathcal{R}$ by some termination proof technique implies an upper bound on $\mathsf{dc}_{\mathcal{R}}$, we call that bound the derivational complexity *induced* by that technique, or simply the derivational complexity of that technique.

An *$\mathcal{F}$-algebra* $\mathcal{A}$ for a signature $\mathcal{F}$ consists of a *carrier* A and, for every function symbol $f \in \mathcal{F}$, an *interpretation function* $f_{\mathcal{A}} : A^n \to A$, where n is the arity of f. Given an *assignment* $\alpha : \mathcal{V} \to A$, we denote the evaluation of a term t in $\mathcal{A}$ by $[\alpha]_{\mathcal{A}}(t)$. A *monotone $\mathcal{F}$-algebra* is a pair $(\mathcal{A}, \succ)$ where $\mathcal{A}$ is an $\mathcal{F}$-algebra and $\succ$ is a proper order such that for every function symbol $f \in \mathcal{F}$, $f_{\mathcal{A}}$ is strictly monotone in all coordinates with respect to $\succ$. A *weakly monotone $\mathcal{F}$-algebra* $(\mathcal{A}, \succcurlyeq)$ is defined similarly, but for every function symbol $f \in \mathcal{F}$, it suffices that $f_{\mathcal{A}}$ is monotone in all coordinates (with respect to the preorder $\succcurlyeq$). A monotone $\mathcal{F}$-algebra $(\mathcal{A}, \succ)$ is called *well-founded* if $\succ$ is well-founded. Similarly, a weakly monotone $\mathcal{F}$-algebra $(\mathcal{A}, \succcurlyeq)$ is well-founded, if the proper order $\succ$ induced by $\succcurlyeq$ is well-founded. Any well-founded monotone $\mathcal{F}$-algebra $(\mathcal{A}, \succ)$ induces a reduction order $\succ_{\mathcal{A}}$ on terms: define $s \succ_{\mathcal{A}} t$ if and only if $[\alpha]_{\mathcal{A}}(s) \succ [\alpha]_{\mathcal{A}}(t)$ for all assignments α. We say $(\mathcal{A}, \succ)$ is *compatible* with a TRS $\mathcal{R}$ if $\mathcal{R} \subseteq \succ_{\mathcal{A}}$. Similarly, given a weakly monotone algebra $(\mathcal{A}, \succcurlyeq)$, we define $s \succcurlyeq_{\mathcal{A}} t$ if and only if $[\alpha]_{\mathcal{A}}(s) \succcurlyeq [\alpha]_{\mathcal{A}}(t)$, and $s \succ_{\mathcal{A}} t$ if and only if $[\alpha]_{\mathcal{A}}(s) \succ [\alpha]_{\mathcal{A}}(t)$ for all assignments α. A *polynomial interpretation* is an interpretation into a well-founded monotone (weakly monotone) algebra $(\mathcal{A}, >)$ ($(\mathcal{A}, \geqslant)$) such that $A \subseteq \mathbb{N}$, $>$ ($\geqslant$) is the standard strict order (preorder) on the natural numbers, and $f_{\mathcal{A}}$ is a polynomial for every function symbol f [21].

We briefly recall the definition of the class of *primitive recursive functions*. The following number-theoretic functions are *initial*: (i) the constant zero functions of all arities: $\mathsf{z}_n(x_1, \ldots, x_n) = 0$, (ii) the successor function $\mathsf{s}(x) = x + 1$, and (iii) the projection functions $\pi_i^n(x_1, \ldots, x_n) = x_i$. A class $\mathcal{C}$ of number-theoretic functions is *closed under composition* if

for all m-ary $g \in \mathcal{C}$ and n-ary $h_1, \dots, h_m \in \mathcal{C}$, the function

$$f(x_1, \dots, x_n) = g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)) \ ,$$

is contained in $\mathcal{C}$, as well. It is *closed under primitive recursion* if for all n-ary $g \in \mathcal{C}$ and $n+2$-ary $h \in \mathcal{C}$ the following function f is contained in $\mathcal{C}$, as well:

$$\begin{aligned} f(0, x_1, \dots, x_n) &= g(x_1, \dots, x_n) \\ f(y+1, x_1, \dots, x_n) &= h(f(y, x_1, \dots, x_n), y, x_1, \dots, x_n) \ . \end{aligned}$$

The class of *primitive recursive functions* is the smallest set of number-theoretic functions which contains all initial functions and is closed under composition and primitive recursion. The definition schemata for primitive recursive functions can be translated to rewrite rules in the obvious way, see for example [9, Definition 2.6].

The i^{th} iterate of a unary function f is denoted as f^i, a similar notation is used for the i^{th} iterate of a function symbol. Finally, we define the function 2_n as follows:

$$2_0(m) := m \qquad 2_{n+1}(m) := 2^{2_n(m)} \ .$$

3. Dependency Pair Method

We recall the central notions of the dependency pair method [1, 14]. Let t be a term. We set $t^\sharp := t$ if $t \in \mathcal{V}$, and $t^\sharp := f^\sharp(t_1, \dots, t_n)$ if $t = f(t_1, \dots, t_n)$. Here $f^\sharp$ is a new n-ary function symbol called *dependency pair symbol*. For a signature $\mathcal{F}$, we define $\mathcal{F}^\sharp := \mathcal{F} \cup \{f^\sharp \mid f \in \mathcal{F}\}$. The set $\mathsf{DP}(\mathcal{R})$ of *dependency pairs* of a TRS $\mathcal{R}$ is defined as $\{l^\sharp \to u^\sharp \mid l \to r \in \mathcal{R}, u \trianglelefteq r, \mathsf{rt}(u) \in \mathcal{D}, u \not\trianglelefteq l\}$. We recall the following characterisation of termination of a TRS.

Proposition 3.1. *A TRS $\mathcal{R}$ is terminating if and only if there exists no infinite derivation of the form $t_1^\sharp \to_{\mathcal{R}}^* t_2^\sharp \to_{\mathsf{DP}(\mathcal{R})} t_3^\sharp \to_{\mathcal{R}}^* \dots$ such that for all $i > 0$, $t_i^\sharp$ is terminating with respect to $\mathcal{R}$.* □

An *argument filtering* (for a signature $\mathcal{F}$) is a mapping π that assigns to every n-ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1, \dots, n\}$ or a (possibly empty) list $[i_1, \dots, i_m]$ of argument positions with $1 \leqslant i_1 < \dots < i_m \leqslant n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f)$ is some list $[i_1, \dots, i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$, also denoted by π:

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

An argument filtering π is extended in the usual way to a TRS $\mathcal{R}$. Let R be a binary relation, then we write $\pi(\mathcal{R}) \subseteq R$ to indicate that for all $l \to r \in \mathcal{R}$, $\pi(l)\, R\, \pi(r)$ holds.

A *reduction pair* $(\succcurlyeq, \succ)$ consists of a preorder $\succcurlyeq$ which is closed under contexts and substitutions, and a compatible well-founded order $\succ$ which is closed under substitutions. Here compatibility means the inclusion $\succcurlyeq \cdot \succ \cdot \succcurlyeq \subseteq \succ$. Recall that any well-founded weakly monotone algebra $(\mathcal{A}, \succcurlyeq)$ gives rise to a pair $(\succcurlyeq_{\mathcal{A}}, \succ_{\mathcal{A}})$ of relations over terms. It is well known that the pair $(\succcurlyeq_{\mathcal{A}}, \succ_{\mathcal{A}})$ forms a reduction pair.

Proposition 3.2. *A TRS $\mathcal{R}$ is terminating if and only if there exist an argument filtering π and a reduction pair $(\succcurlyeq, \succ)$ such that $\pi(\mathsf{DP}(\mathcal{R})) \subseteq \succ$ and $\pi(\mathcal{R}) \subseteq \succcurlyeq$.* □

We write $f \blacktriangleright_{\mathcal{R}} g$ if there exists a rewrite rule $l \to r \in \mathcal{R}$ such that $\mathsf{rt}(l) = f$ and g is a defined function symbol in $\mathcal{F}\mathsf{un}(r)$. For a set $\mathcal{G}$ of defined function symbols we denote by $\mathcal{R} \restriction \mathcal{G}$ the set of rewrite rules $l \to r \in \mathcal{R}$ with $\mathsf{rt}(l) \in \mathcal{G}$. The set $\mathcal{U}_{\mathcal{R}}(t)$ of usable rules of a term t is defined as $\mathcal{R} \restriction \{g \mid f \blacktriangleright^*_{\mathcal{R}} g \text{ for some defined function symbol } f \text{ in } \mathcal{F}\mathsf{un}(t)\}$. Finally, if $\mathcal{P}$ is a set of dependency pairs then $\mathcal{U}_{\mathcal{R}}(\mathcal{P}) := \bigcup_{l \to r \in \mathcal{P}} \mathcal{U}_{\mathcal{R}}(r)$. We write $\mathcal{U}(\mathcal{P})$ instead of $\mathcal{U}_{\mathcal{R}}(\mathcal{P})$ if $\mathcal{R}$ is clear from the context. We use $\mathcal{C}_\epsilon$ to denote the two rules $\mathsf{cons}(x, y) \to x$ and $\mathsf{cons}(x, y) \to y$ for some fresh binary function symbol cons.

Proposition 3.3 ([12, 15])**.** *Let $\mathcal{R}$ be a TRS. If there exist an argument filtering π and a reduction pair $(\succcurlyeq, \succ)$ such that $\pi(\mathsf{DP}(\mathcal{R})) \subseteq \succ$ and $\pi(\mathcal{U}(\mathsf{DP}(\mathcal{R}))) \cup \mathcal{C}_\epsilon \subseteq \succcurlyeq$, then $\mathcal{R}$ is terminating.* □

The *dependency graph* of a TRS $\mathcal{R}$ (denoted by $\mathsf{DG}(\mathcal{R})$) is a graph whose nodes are the dependency pairs of $\mathcal{R}$. It contains an edge from $s \to t$ to $u \to v$ whenever there exist substitutions σ and τ such that $t\sigma \to^*_{\mathcal{R}} u\tau$. A *strongly connected component* (*SCC* for short) of $\mathsf{DG}(\mathcal{R})$ is a maximal subset of nodes such that for each pair of nodes $s \to t$, $u \to v$, there exists a path from $s \to t$ to $u \to v$. We call an SCC *trivial* if it consists of a single node $s \to t$ such that the only path from that node to itself is the empty path. All other SCCs are called *nontrivial*.

Proposition 3.4. *A TRS $\mathcal{R}$ is terminating if and only if for every nontrivial SCC $\mathcal{P}$ in $\mathsf{DG}(\mathcal{R})$ there exist an argument filtering π and a reduction pair $(\succcurlyeq, \succ)$ such that $\pi(\mathcal{P}) \subseteq \succ$ and $\pi(\mathcal{R}) \subseteq \succcurlyeq$.* □

4. Progenitor and Progeny

In this and the next section we show that for the basic dependency pair method (potentially using argument filterings) the induced derivational complexity is triple exponentially bounded in the derivational complexity induced by the base technique employed.

Before proceeding into the technical construction, we outline the proof plan. We aim to bound the length of derivations in a given TRS. Since any derivation in a terminating TRS is non-cycling, the *length* of any derivation is bounded exponentially in the *size* of the occurring terms. On the other hand, the *size* of any term is bounded exponentially in its *depth*. Thus it suffices to show that the *depth* of any term occurring in a derivation is exponentially bounded in the number of admitted dependency pair steps, which in turn is bounded by the derivational complexity induced by the base technique employed.

In the proof, we introduce the *progeny* relation (see Definition 4.1), which is an extension of the descendant relation [34, Chapter 4]. We use the progeny relation in order to extract derivations over $\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}$ from a given derivation over a TRS $\mathcal{R}$ (see Definition 4.8). In Definition 5.1 we exploit this notion to define the *progenitor graph*, which constitutes a suitable restriction of the progeny relation for a given derivation A. The intuition behind progenitor graphs is to define a graph that captures the dependency pair steps of the $\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}$-derivations extracted from A. Moreover the graph is constructed such that its size linearly bounds the height of the last term in A and the height of its components is bounded by the number of admitted dependency pair steps.

For the remainder of this paper, let $\mathcal{R}$ be a TRS. We recall the definition of descendants. Let $A\colon s \to_{p',l\to r} t$ be a rewriting step, and let $p \in \mathcal{P}\mathsf{os}(s)$. Then the *descendants of p in t*

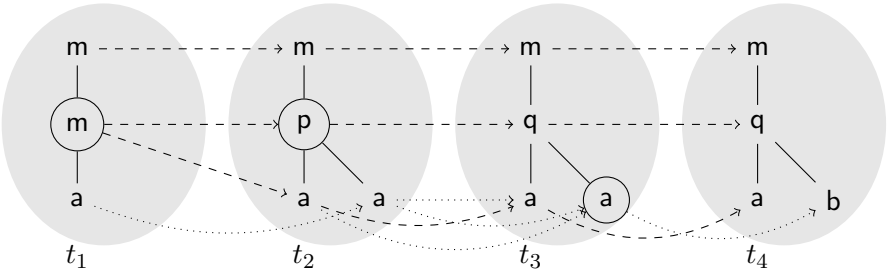

Figure 1: A Derivation, its Progeny Relation and Redex Positions

(denoted by $p\backslash A$) are defined as follows:

$$p\backslash A := \begin{cases} \{p\} & \text{if } p < p' \text{ or } p \parallel p', \\ \{p'q_3q_2 \mid r|_{q_3} = l|_{q_1}\} & \text{if } p = p'q_1q_2 \text{ with } q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(l), \\ \varnothing & \text{otherwise .} \end{cases}$$

We also want to keep track of redex positions, not just of positions in the context or the substitution of the rewrite rule. This intuition is cast into the following definition.

Definition 4.1. Let $A\colon s \to_{p',l\to r} t$ be a rewriting step, and let $p \in \mathcal{P}\mathsf{os}(s)$. Then the *progenies of p in t* (denoted by $p \between A$) are:

$$p \between A := \begin{cases} \{p\} & \text{if } p < p' \text{ or } p \parallel p', \\ \{p'q_3q_2 \mid r|_{q_3} = l|_{q_1}\} & \text{if } p = p'q_1q_2 \text{ with } q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(l), \\ \{p'q_2 \mid r|_{q_2} = l|_{q_1}\} & \text{if } p = p'q_1 \text{ with } q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l) - \{\epsilon\}, \\ \{pq_1 \mid r|_{q_1} \not\trianglelefteq l \wedge q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)\} & \text{if } p = p' . \end{cases}$$

If $q \in p \between A$, then we also say that p is a *progenitor of q in s*. We denote the set of progenitors of q in s by $A \between q$, i.e., we have $q \in p \between A$ if and only if $p \in A \between q$. For a set $P \subseteq \mathcal{P}\mathsf{os}(s)$, we define $P \between A := \bigcup_{p\in P} p \between A$.

Remark 4.2. Note that the distinction between the last two cases corresponds to the exclusion of rules $l^\sharp \to u^\sharp$ from $\mathsf{DP}(\mathcal{R})$ where $u \lhd l$, see Section 3. If we were not considering the exclusion of those rules, we could omit the third case in Definition 4.1, and drop the condition $r|_{q_1} \not\trianglelefteq l$ from the last case.

Example 4.3. Consider the TRS $\mathcal{R}_2$ consisting of the following three rewrite rules:

$$\mathsf{m}(x) \to \mathsf{p}(\mathsf{a}, x) \qquad \mathsf{p}(x, x) \to \mathsf{q}(x, x) \qquad \mathsf{a} \to \mathsf{b} \; .$$

Let A be the derivation

$$\underbrace{\mathsf{m}(\mathsf{m}(\mathsf{a}))}_{t_1} \to \underbrace{\mathsf{m}(\mathsf{p}(\mathsf{a},\mathsf{a}))}_{t_2} \to \underbrace{\mathsf{m}(\mathsf{q}(\mathsf{a},\mathsf{a}))}_{t_3} \to \underbrace{\mathsf{m}(\mathsf{q}(\mathsf{a},\mathsf{b}))}_{t_4} \; ,$$

which is represented in Figure 1. Redex positions are marked by circles, the progeny relation is marked by dotted and dashed lines (the two kinds of lines will be distinguished in Example 4.14 below). Note that each position in a term may have several progenitors. For instance, $(t_2 \to t_3) \between 11 = \{11, 12\}$.

Lemma 4.4. *Let $A\colon s \to t$, let $p \in \mathcal{P}\mathsf{os}(s)$, and let $q \in \mathcal{P}\mathsf{os}(t)$. If $q \in p \mathbin{\backslash\!\backslash} A$ and $\mathsf{rt}(t|_q) \in \mathcal{D}$, then $\mathsf{rt}(s|_p) \in \mathcal{D}$ and $(s|_p)^\sharp \to^{=}_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}} (t|_q)^\sharp$.*

Proof. Suppose that A is $s \to_{p',l\to r} t$. If $p < p'$ or $p \parallel p'$, then by definition, we have $p = q$ and thus $(s|_p)^\sharp \to^{=}_{\mathcal{R}} (t|_q)^\sharp$. On the other hand, if $p = p'$, then there exists $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$ such that $q = p'q_1$. Moreover, $t|_q \not\trianglelefteq s|_p$. By assumption $\mathsf{rt}(t|_q) \in \mathcal{D}$ and thus we obtain $(s|_p)^\sharp \to_{\mathsf{DP}(\mathcal{R})} (t|_q)^\sharp$. Finally, if $p > p'$, then by definition of the progeny relation, we have $s|_p = t|_q$. Then again, $(s|_p)^\sharp \to^{=}_{\mathcal{R}} (t|_q)^\sharp$ follows trivially. □

Lemma 4.5. *Let $A\colon s \to t$. Then for every $q \in \mathcal{P}\mathsf{os}(t)$, we have $A \mathbin{\backslash\!\backslash} q \neq \varnothing$.*

Proof. Suppose A denotes the step $s \to_{p',l\to r} t$. If $q < p'$ or $q \parallel p'$, then $A \mathbin{\backslash\!\backslash} q = \{q\}$. If $q = p'q_1$, $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$, and $r|_{q_1} \not\trianglelefteq l$, then $A \mathbin{\backslash\!\backslash} q = \{p'\}$. If $q = p'q_1$, $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$, and $r|_{q_1} \trianglelefteq l$, then there is some p_1 such that $l|_{p_1} = r|_{q_1}$, so $p'p_1 \in A \mathbin{\backslash\!\backslash} q$. Last, if $q = p'q_1q_2$ and $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(r)$, then there is some p_1 such that $l|_{p_1} = r|_{q_1}$ because $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(l)$. Therefore, $p'p_1q_2 \in A \mathbin{\backslash\!\backslash} q$. □

Definition 4.6. Let $A\colon s \to^* t$ be a derivation, and let $p \in \mathcal{P}\mathsf{os}(s)$. Then the *progenies of p in t* (also denoted by $p \mathbin{\backslash\!\backslash} A$) are defined as follows:

(1) If A is the empty derivation, then $p \mathbin{\backslash\!\backslash} A = \{p\}$.
(2) Otherwise, we can split A into $A_1 : s \to s'$ and $A_2 : s' \to^* t$. Then $p \mathbin{\backslash\!\backslash} A = (p \mathbin{\backslash\!\backslash} A_1) \mathbin{\backslash\!\backslash} A_2$.

We say p is a *progenitor* of q if $p \in A \mathbin{\backslash\!\backslash} q$, which holds if $q \in p \mathbin{\backslash\!\backslash} A$. Moreover, we have $q \in P \mathbin{\backslash\!\backslash} A$ if and only if $q \in p \mathbin{\backslash\!\backslash} A$ for some $p \in P$.

Lemma 4.7. *Let $A\colon s \to^* t$ be a derivation and let $p \in \mathcal{P}\mathsf{os}(s)$, $q \in \mathcal{P}\mathsf{os}(t)$. Then the set $A \mathbin{\backslash\!\backslash} q$ of progenitors of q is not empty. Moreover if $q \in p \mathbin{\backslash\!\backslash} A$ with $\mathsf{rt}(t|_q) \in \mathcal{D}$, then $\mathsf{rt}(s|_p) \in \mathcal{D}$ and $(s|_p)^\sharp \to^*_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}} (t|_q)^\sharp$.*

Proof. Straightforward induction using Lemmata 4.5 and 4.4. □

Using Lemma 4.7, we can extract derivations over $\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}$ from a given derivation in a TRS $\mathcal{R}$ using positions connected by the progeny relation.

Definition 4.8. Let $t_1, \ldots, t_n$ be terms, and let $p_1, \ldots, p_n$ be positions in $t_1, \ldots, t_n$, respectively, such that $\mathsf{rt}(t_n|_{p_n}) \in \mathcal{D}$, and for all $1 \leqslant i \leqslant n-1$, we have $A_i\colon t_i \to_{\mathcal{R}} t_{i+1}$ and $p_{i+1} \in p_i \mathbin{\backslash\!\backslash} A_i$. Then we call $A\colon (t_1|_{p_1})^\sharp \to^*_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}} (t_n|_{p_n})^\sharp$ the *implicit dependency pair derivation* with respect to $t_1, \ldots, t_n$ and $p_1, \ldots, p_n$. We denote the number of $\mathsf{DP}(\mathcal{R})$-steps in A as $\mathsf{DPI}(A)$.

Example 4.9 (continued from Example 4.3)**.** The implicit dependency pair derivation with respect to the terms t_1, t_2, t_3 and the positions 1, 11, 12 is given as follows:

$$\mathsf{m}^\sharp(\mathsf{a}) \;\to_{\mathsf{DP}(\mathcal{R}_2)}\; \mathsf{a}^\sharp \;\to^{=}_{\mathcal{R}_2}\; \mathsf{a}^\sharp\,.$$

Note that the length of this implicit dependency pair derivation is smaller than the length of the original derivation A. Moreover, all terms occurring in this implicit dependency pair derivation are proper subterms of the respective terms of A (modulo marking top symbols by a $\sharp$). In contrast, the implicit dependency pair derivation with respect to the terms t_1, t_2, t_3, t_4, and the positions ϵ, ϵ, ϵ, ϵ is given by $t_1^\sharp \to_{\mathcal{R}_2} t_2^\sharp \to_{\mathcal{R}_2} t_3^\sharp \to_{\mathcal{R}_2} t_4^\sharp$.

The following lemma shows that given two positions $q \leqslant q'$ in the same branch of a term, and a progenitor p_0 of q, we can always find a progenitor p_0' of q' such that $p_0 \leqslant p_0'$.

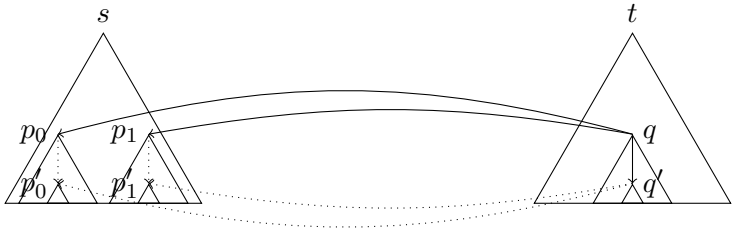

Figure 2: Intuition for Lemma 4.10

This is graphically depicted in Figure 2, where the drawn lines indicate the assumption of the lemma and the dotted lines the conclusion. The lemma entails that for any branch B of a term, we can find progenitors of all positions in B in a single branch again.

Lemma 4.10. *Let $A\colon s \to t$ and let $q, q' \in \mathcal{P}\mathsf{os}(t)$. If $q \leqslant q'$, then for any $p_0 \in A \setminus\!\setminus q$, there exists $p_0' \in A \setminus\!\setminus q'$ such that $p_0 \leqslant p_0'$.*

Proof. Suppose A has the form $s \to_{p',l\to r} t$. According to Definition 4.1, there are four cases for q'.

(1) If $q' < p'$ or $q' \parallel p'$, then also $q < p'$ or $q \parallel p'$. Therefore, $A \setminus\!\setminus q = \{q\}$ and $A \setminus\!\setminus q' = \{q'\}$.
(2) If $q' = p'q_1'$, $q_1' \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$, and $r|_{q_1'} \not\trianglelefteq l$, then either $q < p'$, or $q = p'q_1$, $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$, and $r|_{q_1} \not\trianglelefteq l$. We have $A \setminus\!\setminus q = \{p_0\}$ and $A \setminus\!\setminus q' = \{p'\}$ with $p_0 = q$ or $p_0 = p'$. In both cases, $p_0 \leqslant p'$, so the lemma follows.
(3) If $q' = p'q_1'$, $q_1' \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$, and $r|_{q_1'} \trianglelefteq l$, then $A \setminus\!\setminus q' = \{p'q_2' \mid q_2' \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l) \wedge r|_{q_1'} = l|_{q_2'}\}$. From the three cases in Definition 4.1 applicable for q, we only consider the last one, where $q = p'q_1$, $q_1 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$ and $r|_{q_1} \trianglelefteq l$, then $A \setminus\!\setminus q = \{p'q_2 \mid q_2 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l) \wedge r|_{q_1} = l|_{q_2}\}$. Since $q \leqslant q'$, there exists some q_3' such that $q' = qq_3'$. Hence, for any $p'q_2 \in A \setminus\!\setminus q$, we also have $p'q_2q_3' \in A \setminus\!\setminus q'$, entailing the lemma.
(4) If $q' = p'q_1'q_2'$ with $q_1' \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(r)$, then $A \setminus\!\setminus q' = \{p'q_3'q_2' \mid r|_{q_1'} = l|_{q_3'}\}$. Except for $q \parallel p'$, all cases are possible for q. Again, we restrict to one of these cases and assume that $q = p'q_1'q_2$. Then $A \setminus\!\setminus q = \{p'q_3q_2 \mid r|_{q_1'} = l|_{q_3}\}$. Since $q \leqslant q'$, there exists q_4' such that $q_2' = q_2q_4'$. Hence, for any $p'q_3q_2 \in A \setminus\!\setminus q$, we also have $p'q_3q_2' \in A \setminus\!\setminus q'$, thus the lemma follows. □

In order to simplify the structure of the progeny relation we restrict the progenies and progenitors to a single branch in each term. The definition rests on the idea that for a derivation $A\colon s \to^* t$ and a *main branch* B' in t it is possible to find a main branch B in s such that each position $q \in B'$ has a (unique) progenitor in B. See Figure 3 for an illustration. The bold lines denote the main branches of s and t, and the thin lines denote other branches of s containing progenitors of all positions in the main branch of t.

Definition 4.11. Let $A\colon t_1 \to^* t_n$ denote a derivation built up from the rewrite steps $A_i\colon t_i \to t_{i+1}$ for $i = 1, \ldots, n-1$. Then the *main branch* of each term in A is inductively defined:

(1) The main branch of t_n is the leftmost branch among all branches of maximal length in t_n.
(2) Suppose the main branch of t_{i+1} is denoted as B_{i+1}, $1 \leqslant i \leqslant n-1$. Then consider all branches b in t_i such that for every $q \in B_{i+1}$, the set of progenitors $A_i \setminus\!\setminus q$ of q has

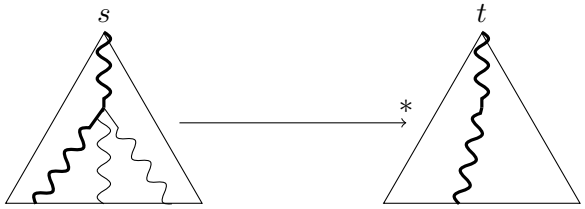

Figure 3: Depiction of the Main Branch in a Derivation

nonempty intersection with b. The leftmost of these branches is the main branch of t_i, denoted as B_i.

In the above definition, the restriction to the leftmost of all candidate branches is arbitrary and can be suitably replaced. The second clause is well-defined by Lemmata 4.7 and 4.10. Note that a branch of maximal size is chosen for the final term of the given derivation since it reflects the depth of this term, c.f. Section 5. The next definition specialises progenies and progenitors to the main branch.

Definition 4.12. Let $A' : s \to t$ be a rewriting step, let $p \in \mathcal{P}\mathsf{os}(s)$, and let B and B' be branches in s and t, respectively. Then the set of *main progenies of p in t* (with respect to A') (denoted as $p \supset^B_{B'} A'$) is defined as follows:

$$p \supset^B_{B'} A' := \begin{cases} \varnothing & \text{if } p \notin B \\ B' \cap (p \setminus\setminus A') & \text{if } p \in B\,. \end{cases}$$

If the (main) branches B and B' are clear from context, we write $p \supset A'$ instead of $p \supset^B_{B'} A'$. If $q \in p \supset A'$, then we also say that p is a *main progenitor of q in s* (with respect to A'). We denote the set of main progenitors of q in s by $A' \supset q$. For a set $P \subseteq \mathcal{P}\mathsf{os}(s')$, we define $P \supset A' := \bigcup_{p\in P} p \supset A'$. We naturally extend the definition to derivations $A\colon s \to^* t$, analogous to Definition 4.6: if A is the empty derivation, then $p \supset^B_{B'} A = \{p\}$. Otherwise, we can split A into $A_1 : s \to s'$ and $A_2 : s' \to^* t$. Let B'' be the main branch in s'. Then $p \supset^B_{B'} A = (p \supset^B_{B''} A_1) \supset^{B''}_{B'} A_2$.

Lemma 4.13. *Let $A\colon u \to^* s \to^n t \to^* w$ and denote $A'\colon s \to^n t$. Let $B(s)$ and $B(t)$ denote the main branches of s and t in A, respectively. Then for any $q \in B(t)$, the main progenitor of q in the branch $B(s)$ is unique, i.e., $|A' \supset q| = 1$.*

Proof. By Definition 4.11, q has at least one main progenitor in s. We show that there exists at most one by induction on n. For $n = 0$ the claim is trivial. Hence assume $n > 0$ and let $A'\colon s \to s' \to^{n-1} t$. Let $B(s')$ denote the main branch in s' with respect to A. By induction hypothesis there exists a unique position p_1 in $B(s')$ such that $(s' \to^{n-1} t) \supset q = \{p_1\}$. Let $A''\colon s \to_{p',l\to r} s'$ denote the first rewrite step in A'. Suppose $p_1 < p'$ or $p_1 \parallel p'$. Then by definition $A'' \setminus\setminus p_1 = \{p_1\}$. Hence the main progenitor of q in $B(s)$ is unique. On the other hand suppose $p_1 = p'p_2$ with $p_2 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$ such that $r|_{p_2} \not\trianglelefteq l$. Then $A'' \setminus\setminus p_1 = \{p'\}$ and $A' \supset q$ is a singleton as it should be. Now suppose $p_1 = p'p_2$ with $p_2 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)$ such that $r|_{p_2} \triangleleft l$. Then by definition $A'' \setminus\setminus p_1 = \{p'p_3 \mid p_3 \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l) \wedge l|_{p_3} = r|_{p_2}\}$. Note that $A' \supset q = A'' \setminus\setminus p_1 \cap B(s)$, which is again a singleton. Finally, if $p_1 = p'p_2p_3$ with $p_2 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(r)$, then $A'' \setminus\setminus p_1 = \{p'p_4p_3 \mid p_4 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(l) \wedge l|_{p_4} = r|_{p_2}\}$. As before, the intersection of the latter

set with $B(s)$ is a singleton. Hence the main progenitor of q in $B(s)$ is unique. This concludes the inductive proof. □

Observe that we cannot define main progenies for (multi-step) derivations directly by restricting the progeny relation to the main branches; it is indeed necessary to use the inductive definition given above. In particular, Lemma 4.13 would be incorrect for that definition, as exemplified below.

Example 4.14 (continued from Example 4.3)**.** Consider the derivation A again. We split A into $A_1 : t_1 \to t_2$, $A_2 : t_2 \to t_3$, and $A_3 : t_3 \to t_4$. The "central" branch of each term in Figure 1 is its main branch, and the dashed lines denote the main progeny relation. Note that $1 \in A \supset 11$, since $11 \in A_3 \supset 11$, $11 \in A_2 \supset 11$, and $1 \in A_1 \supset 11$. Furthermore, we do not have $11 \in A \supset 11$, even though $11 \in A_3 \setminus\!\setminus 11$, $12 \in A_2 \setminus\!\setminus 11$, $11 \in A_1 \setminus\!\setminus 12$, and therefore $11 \in A \setminus\!\setminus 11$.

For positions pointing to non-defined symbols, we also have the reverse of Lemma 4.13.

Lemma 4.15. *We assume the same notation as in Lemma 4.13. For any $p \in B(s)$ such that $\mathsf{rt}(s|_p) \in \mathcal{C} \cup \mathcal{V}$, we have $|p \supset A'| \leqslant 1$, i.e., the number of main progenies for a position such that the root of the corresponding subterm is non-defined is at most* 1*.*

Proof. By induction on n. It suffices to consider the case $n > 0$, so $A' \colon s \to s' \to^{n-1} t$. Let $A'' \colon s \to_{p',l\to r} s'$ denote the first rewrite step in A'. If $p < p'$ or $p \parallel p'$, then $p \supset A'' = \{p\}$. If $p > p'$, then for any $p_1 \in p \setminus\!\setminus A''$, we have $s|_p = s'|_{p_1}$, so again, $p \setminus\!\setminus A'' \cap B(s')$ is a singleton. In all of these cases, the claim follows by induction hypothesis as $\mathsf{rt}(s|_p) = \mathsf{rt}(s'|_{p_1})$ for any $p_1 \in p \supset A''$. This concludes the proof, as the case $p = p'$ is impossible. Otherwise, we derive a contradiction to the assumption that the root of $s|_p$ is not a defined symbol. □

5. Dependency Pairs and Complexity

Let $A \colon t_1 \to^*_{\mathcal{R}} t_n$ be a derivation with respect to $\mathcal{R}$, and let m be the maximum number of $\mathsf{DP}(\mathcal{R})$-steps in any implicit dependency pair derivation corresponding to A. In this section we show that the length n of A is bounded triple exponentially in m. As mentioned at the beginning of Section 4, it suffices to show that the depth of any term occurring in A is exponentially bounded in m. More precisely, as we consider an arbitrary derivation A, it even suffices to show that the depth of the term t_n is exponentially bounded in m, c.f. Lemma 5.11.

Notation. In the sequel, we fix the derivation A and let $B_1, \ldots, B_n$ denote the main branches of $t_1, \ldots, t_n$ with respect to A. Let G be the progenitor graph of A (see Definition 5.1 below). For the remainder of this paper, let $C := \max(\{2\} \cup \{\mathsf{dp}(r)+1 \mid l \to r \in \mathcal{R}\})$. We call C the *branching constant* of $\mathcal{R}$.

In the next definition we formalise *progenitor graphs*.

Definition 5.1. The *progenitor graph* G of A is defined as follows.

(1) The nodes are all pairs (t_i, p) such that $p \in B_i$ with $\mathsf{rt}(t_i|_p)$ defined and either $i = 1$ or the single element of $(t_{i-1} \to t_i) \supset p$ and the redex position in the rewrite step $t_{i-1} \to t_i$ coincide.

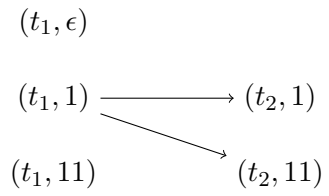

Figure 4: Progenitor Graph

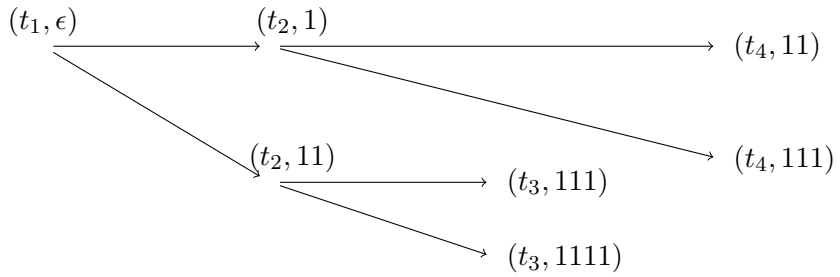

Figure 5: Progenitor Graph: Full Binary Tree

(2) There is an edge from (t_i, p) to (t_j, q) whenever $i < j$, $(t_i \to^* t_j) \ni q = \{p\}$, and for all $i \leqslant k < j - 1$, the single element of $(t_k \to^* t_j) \ni q$ and the redex position in the rewrite step $t_k \to t_{k+1}$ do *not* coincide.

With respect to the definition of edges note that the single element of $(t_{j-1} \to t_j) \ni q$ and the redex position in the rewrite step $t_{j-1} \to t_j$ coincide. Also note that G is a forest, and the root of each tree in G is (t_1, p) for some $p \in B_1$.

Example 5.2. Consider the derivation A from Example 4.3 again. Its progenitor graph G is shown in Figure 4. Observe that (t_2, ϵ) is not contained in G since the single element of $(t_1 \to t_2) \ni \epsilon$ is ϵ, and ϵ is not the redex position of the step $t_1 \to t_2$. For similar reasons, (t_3, ϵ), $(t_3, 11)$, (t_4, ϵ), $(t_4, 1)$, and $(t_4, 11)$ are not contained in G. Moreover $(t_3, 1)$ is not contained in G either because $\mathsf{rt}(t_3|_1) = \mathsf{q}$ is not defined. Furthermore, $(t_2, 12)$, $(t_3, 12)$, and $(t_4, 12)$ are not contained in G because 12 is not a member of the main branch of t_2, t_3, and t_4, respectively.

However, (t_1, ϵ), $(t_1, 1)$, and $(t_1, 11)$ are still contained in G because all positions of t_1 pointing to defined symbols are in G. Moreover $(t_2, 1)$ is contained in G because $\mathsf{rt}(t_2|_1) \in \mathcal{D}$, and the single element of $(t_1 \to t_2) \ni 1$ is the redex position of the step $t_1 \to t_2$. For the same reason, $(t_2, 11)$ is contained in G.

The main factor of the exponentially faster growth of $\mathsf{dp}(t_n)$ compared to the maximal height of all trees in G is the difference between that maximal height and the size of G (which is linearly related to $\mathsf{dp}(t_n)$). This becomes apparent in our next example, where G is a full binary tree.

Example 5.3. Consider the TRS $\mathcal{R}_3 := \{\mathsf{full}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{full}(\mathsf{full}(x)))\}$ together with the following derivation A:

$$\underbrace{\mathsf{full}(\mathsf{s}(\mathsf{s}(0)))}_{t_1} \to \underbrace{\mathsf{s}(\mathsf{full}(\mathsf{full}(\mathsf{s}(0))))}_{t_2} \to \underbrace{\mathsf{s}(\mathsf{full}(\mathsf{s}(\mathsf{full}(\mathsf{full}(0)))))}_{t_3} \to \underbrace{\mathsf{s}(\mathsf{s}(\mathsf{full}(\mathsf{full}(\mathsf{full}(\mathsf{full}(0))))))}_{t_4} .$$

The progenitor graph of A is shown in Figure 5.

Lemma 5.4. *If there is an edge from (t_i, p) to (t_j, q) in G, then there exists $q' \in B_{j-1}$ such that there is a derivation $(t_i|_p)^\sharp \to_{\mathcal{R}}^* (t_{j-1}|_{q'})^\sharp \to_{\mathsf{DP}(\mathcal{R})} (t_j|_q)^\sharp$.*

Proof. By definition, $q \in p \supset (t_i \to^* t_j)$. Therefore, by Lemma 4.7, we have the implicit dependency pair derivation $A' \colon (t_i|_p)^\sharp \to_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}}^* (t_j|_q)^\sharp$. We have $(t_{j-1} \to t_j) \supset q = \{q'\}$, where by definition q' is the redex position of the step $t_{j-1} \to t_j$. Therefore, the last step of A' is a $\mathsf{DP}(\mathcal{R})$-step (see also the last clause of Definition 4.1). Note that for $i \leqslant k < j-1$, the single element of $(t_k \to^* t_j) \supset q$ and the redex position in $t_k \to t_{k+1}$ do not coincide. Hence, if there are rewrite steps before the last step, these are $\mathcal{R}$-steps and the lemma follows. □

The next lemma shows that Definition 5.1 is well-defined in the sense that only those nodes that do not contribute to the branching of the progenitor graph are left out.

Lemma 5.5. *Let $p \in B_i$ and $q \in B_j$ such that $i < j$ and $(t_i \to^* t_j) \supset q = \{p\}$. If for all $i \leqslant k \leqslant j-1$, the single element of $(t_k \to^* t_j) \supset q$ and the redex position in the rewrite step $t_k \to t_{k+1}$ do not coincide, then $p \supset (t_i \to^* t_j) = \{q\}$.*

Proof. We show the lemma by induction on $j - i$. If $i = j$ then the claim trivially holds. Otherwise, the derivation $t_i \to^* t_j$ can be split into $t_i \to t_{i+1} \to^* t_j$. Let p' be the redex position in $t_i \to t_{i+1}$. If $p \parallel p'$, $p < p'$, or $p > p'$, then as in Lemma 4.15, $|p \supset (t_i \to t_{i+1})| \leqslant 1$, and the lemma follows by induction hypothesis. The remaining case is again impossible, since by assumption, p and p' do not coincide. □

In the following sequence of lemmata we show the properties which allow us to bound $\mathsf{dp}(t_n)$ in the maximal height of all trees in G. First, we prove that almost each position in B_n is "covered" by a node in G. Next, we show that there exists a fixed upper bound on the number of positions in B_n each node in G can cover, and finally, we show that there is a fixed upper bound on the branching factor of G, as well.

Lemma 5.6. *Let $k \in \{1, \dots, n\}$. For every $q \in B_k$, there exists a unique $p \in B_1$ such that either $\mathsf{rt}(t_1|_p) \in \mathcal{C} \cup \mathcal{V}$ and $(t_1 \to^* t_k) \supset q = \{p\}$, or there exists a unique node (t_i, p_i) in G where $q \in p_i \supset (t_i \to^* t_k)$ and for any direct successor node (t_j, p_j) of (t_i, p_i) in G, we have $q \notin p_j \supset (t_j \to^* t_k)$.*

Proof. By Lemma 4.13, $(t_1 \to^* t_k) \supset q = \{p\}$ for some $p \in B_1$. If $\mathsf{rt}(t_1|_p) \in \mathcal{C} \cup \mathcal{V}$, the first alternative of the lemma holds. If $\mathsf{rt}(t_1|_p) \in \mathcal{D}$, then $(t_1, p) \in G$. Therefore, there exists a maximal natural number i such that $(t_i, p_i) \in G$ and $q \in p_i \supset (t_i \to^* t_k)$ for some $p_i \in B_i$, so the second alternative of the lemma holds for (t_i, p_i). □

Lemma 5.6 suggest the following definition.

Definition 5.7. Let $k \in \{1, \dots, n\}$ and let $q \in B_k$. Suppose $(t_1 \to^* t_k) \supset q = \{p\}$ such that $\mathsf{rt}(t_1|_p) \notin \mathcal{C} \cup \mathcal{V}$. Furthermore let (t_i, p_i) be the unique node in G where $q \in p_i \supset (t_i \to^* t_k)$ and for any direct successor node (t_j, p_j) of (t_i, p_i) in G, we have $q \notin p_j \supset (t_j \to^* t_k)$. Then (t_i, p_i) is said to *cover* the position $q \in B_k$.

As shown in the next lemma, C is an upper bound on the number of positions in B_n each node in G can cover.

Lemma 5.8. *For every node (t_i, p) in G, there are at most C many positions $q \in B_n$ covered by (t_i, p).*

Proof. If there is no $i \leqslant j < n$ such that the redex position of the step $t_j \to t_{j+1}$ and an element of $p \supset (t_i \to^* t_j)$ coincide, then it follows from Lemma 5.5 that $|p \supset (t_i \to^* t_n)| \leqslant 1 < C$.

Otherwise, let k be the smallest number such that $k \geqslant i$ and $p \supset (t_i \to^* t_k) = \{p_k\}$, where p_k is the redex position of $t_k \to t_{k+1}$. By Definitions 4.1 and 4.12, $|p_k \supset (t_k \to t_{k+1})| \leqslant C$. In the next paragraph, we show for each $p_{k+1} \in p_k \supset (t_k \to t_{k+1})$ that $|p_{k+1} \supset (t_{k+1} \to^* t_n)| \leqslant 1$. Hence the node (t_k, p_k) can cover at most C many positions in B_n.

For each $p_{k+1} \in p_k \supset (t_k \to t_{k+1})$, if $\mathsf{rt}(t_{k+1}|_{p_{k+1}})$ is defined, then (t_{k+1}, p_{k+1}) is a successor node of (t_i, p), and for any main progeny q of p_{k+1}, by definition we have $q \in p_{k+1} \supset (t_{k+1} \to^* t_n)$, which violates the definition of being covered by (t_i, p). On the other hand, suppose $\mathsf{rt}(t_{k+1}|_{p_{k+1}})$ is a constructor symbol or a variable. Then by Lemma 4.15, $|p_{k+1} \supset (t_{k+1} \to^* t_n)| \leqslant 1$. □

The following example illustrates the role of Lemma 5.8.

Example 5.9. Let $\mathcal{R}_4$ be the TRS consisting of the single rewrite rule

$$\mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \ .$$

Let $t_1 = \mathsf{d}(\mathsf{s}(\mathsf{s}(0)))$, $t_2 = \mathsf{s}(\mathsf{s}(\mathsf{d}(\mathsf{s}(0))))$, and $t_3 = \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{d}(0)))))$. We have the derivation $A\colon t_1 \to t_2 \to t_3$ and the following progenitor graph:

$$(t_1, \epsilon) \xrightarrow{\qquad} (t_2, 11) \xrightarrow{\qquad} (t_3, 1111)$$

Note that G leaves out all function symbols s above the d in each term. However, by Lemma 5.8, the number of positions in the last term of A which are hidden in this way is bounded linearly in the size of the progenitor graph.

The next lemma shows that the "branching factor" of G, i.e., the maximal number of direct successors of a node in G, is bounded by the branching constant C.

Lemma 5.10. *Every node in G has at most C many direct successor nodes.*

Proof. Let (t_i, p) be a node in G. If there is no $i \leqslant j < n$ such that the redex position of the step $t_j \to t_{j+1}$ and an element of $p \supset (t_i \to^* t_j)$ coincide, then (t_i, p) has no successor node, so the claim holds. Otherwise, let j be the smallest number greater than i such that $p \supset (t_i \to^* t_j) = \{q\}$, where q is the redex position of $t_j \to t_{j+1}$. By Definitions 4.1 and 4.12, $|q \supset (t_j \to t_{j+1})| \leqslant C$. Hence, (t_i, p) has at most C many direct successor nodes. □

We are ready to prove the main lemma of this section.

Lemma 5.11. *Let $\mathcal{R}$ be terminating and let $f(t) := \max\{\mathsf{dh}(u^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \mid u \trianglelefteq t\}$. Then there exists $d \in \mathbb{N}$ such that for all terms t: $\mathsf{pdp}(t, \to_{\mathcal{R}}) \leqslant (\mathsf{dp}(t) + 1) \cdot 2^{d \cdot (f(t)+2)}$.*

Proof. Consider any derivation $A\colon s \to^*_{\mathcal{R}} t$ and let $A'\colon (u)^\sharp \to^*_{\mathsf{DP}(\mathcal{R})/\mathcal{R}} (v)^\sharp$ be a maximal derivation over $\mathsf{DP}(\mathcal{R})$ modulo $\mathcal{R}$ such that $u \trianglelefteq s$. Set $m := \mathsf{DPI}(A')$. Let k be the number of defined symbols in the main branch of s. If $k = 0$, then s is a normal form, hence $s = t$, and the lemma follows trivially. In the following we assume $k > 0$. Note that $k \leqslant \mathsf{dp}(s) + 1$.

It is easy to see that the progenitor graph G forms a forest consisting of k distinct trees $T_1, \ldots, T_k$.

Due to Lemma 5.4 the height of each tree $T_1, \ldots, T_k$ in G is bounded by m. Here the height of a tree is the number of edges on the longest path from the root to a leaf. Recall that any C-ary tree of height m has at most $\frac{C^{m+1}-1}{C-1} \leqslant C^{m+1}$ many nodes. Hence, due to Lemma 5.10, each of the trees T_i $(1 \leqslant i \leqslant k)$ has at most C^{m+1} many nodes. Thus G can have at most $k \cdot C^{m+1}$ many nodes.

The main branch of t consists of $\mathsf{dp}(t) + 1$ many positions, all of which have to fulfil one of the two properties in Lemma 5.6. By Lemma 4.15, the first case applies to at most $\mathsf{dp}(s) + 1 - k$ many positions. Furthermore, due to Lemma 5.8 each node in G can cover at most C many positions in the main branch of t. In sum we obtain the following upper bound on the depth of t:

$$\mathsf{dp}(t) \leqslant (k \cdot C^{m+1}) \cdot C + \mathsf{dp}(s) - k \leqslant (\mathsf{dp}(s) + 1) \cdot C^{m+2} \,,$$

where we have applied $k \leqslant \mathsf{dp}(s) + 1$ in the second inequality. By definition $m = \mathsf{DPI}(A') = f(s)$ from which the lemma follows immediately. □

All that is left to show is that the derivational complexity of a finite and terminating TRS is bounded double exponentially in its depth growth. This can be achieved by two easy observations.

Lemma 5.12. *Let $\mathcal{R}$ be terminating. Then there exists $d \in \mathbb{N}$ such that for all terms t: $\mathsf{dh}(t, \to_{\mathcal{R}}) \leqslant 2^{2^{d \cdot \mathsf{pdp}(t, \to_{\mathcal{R}})}}$.*

Proof. We show that there exist constants e and e', such that for all terms t, the inequalities $\mathsf{psz}(t, \to_{\mathcal{R}}) \leqslant 2^{e \cdot \mathsf{pdp}(t, \to_{\mathcal{R}})}$ and $\mathsf{dh}(t, \to_{\mathcal{R}}) \leqslant 2^{e' \cdot \mathsf{psz}(t, \to_{\mathcal{R}})}$ hold. Then the lemma follows easily by choosing $d = e + e'$, for instance.

(1) For any term t, we have $|t| \leqslant k^{\mathsf{dp}(t)+1}$, where k is the maximum arity of any function symbol in the signature. This proves the first inequality.
(2) On the other hand, by assumption the signature $\mathcal{F}$ of $\mathcal{R}$ is finite. Moreover without loss of generality the considered derivation in $\mathcal{R}$ is ground. Hence we can build only $2^{e' \cdot m}$ different terms of size at most m, where e' depends only on $\mathcal{F}$. This proves the second inequality. □

Theorem 5.13. *Let $\mathcal{R}$ be terminating and let*

$$f(n) := \max\{\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \mid |t| \leqslant n\} \,.$$

Then there exists $D \in \mathbb{N}$ such that $\mathsf{dc}_{\mathcal{R}}(n) \leqslant 2^{2^{n \cdot 2^{D \cdot (f(n)+2)}}}$.

Proof. The theorem follows directly from Lemmata 5.11 and 5.12. □

We also call the function f defined in the theorem the *dependency pair complexity* function. Observe that for any argument filtering π and any terms s, t, we have that $s^\sharp \to_{\mathcal{R}} t^\sharp$ implies $\pi(s^\sharp) \to^{=}_{\pi(\mathcal{R})} \pi(t^\sharp)$. Furthermore $s^\sharp \to_{\mathsf{DP}(\mathcal{R})} t^\sharp$ implies $\pi(s^\sharp) \to_{\pi(\mathsf{DP}(\mathcal{R}))} \pi(t^\sharp)$. These observations are sufficient to extend Theorem 5.13 to argument filtering.

Corollary 5.14. *Let $\mathcal{R}$ be terminating, let π be an argument filtering, and let*

$$f(n) := \max\{\mathsf{dh}(\pi(t^\sharp), \to_{\pi(\mathsf{DP}(\mathcal{R}))/\pi(\mathcal{R})}) \mid |t| \leqslant n\} \,.$$

Then $\mathsf{dc}_{\mathcal{R}}(n) \leqslant 2^{2^{n \cdot 2^{D \cdot (f(n)+2)}}}$, where D is defined as above. □

This concludes that termination proofs by the basic dependency pair method combined with some base technique (possibly enhanced by argument filtering) imply a complexity bound that is triple exponential in the derivational complexity of the base technique. For instance, if polynomial interpretations are used as base technique, then the derivational complexity of the TRS under consideration is bounded by $2_5(\mathsf{O}(n))$. On the other hand, if KBO is used as a base technique, then the derivational complexity of the TRS is bounded by $\mathrm{Ack}(\mathsf{O}(n), 0)$.

So by Theorem 5.13, the derivational complexity of a TRS $\mathcal{R}$ is bounded triple exponentially in its dependency pair complexity. This yields an upper-bound. The following TRS establishes a double exponential lower-bound.

Example 5.15. Consider the following TRS $\mathcal{R}_5$, extending the TRS $\mathcal{R}_3$:

$$1\colon\ \mathsf{full}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{full}(\mathsf{full}(x))) \qquad 2\colon\ \mathsf{full}(x) \to \mathsf{cons}(x, x)\ .$$

We show that $\mathcal{R}_5$ has linear dependency pair complexity, but admits derivations of double exponential length.

Let $C(x)$ be the shorthand for $\mathsf{cons}(x, x)$. Now, consider the starting term $\mathsf{full}(\mathsf{s}^n(\mathsf{0}))$. As can be easily seen, this term rewrites to $\mathsf{s}^n(\mathsf{full}^{2^n}(\mathsf{0}))$ in $2^n - 1$ steps using rule 1. Now, we can use rule 2 and an outermost strategy to reach $\mathsf{s}^n(C^{2^n}(\mathsf{0}))$ in $2^{2^n} - 1$ steps, so $\mathsf{dc}_{\mathcal{R}_5}$ is at least double exponential.

On the other hand consider $\mathsf{DP}(\mathcal{R}_5)$:

$$3\colon\ \mathsf{full}^\sharp(\mathsf{s}(x)) \to \mathsf{full}^\sharp(\mathsf{full}(x)) \qquad 4\colon\ \mathsf{full}^\sharp(\mathsf{s}(x)) \to \mathsf{full}^\sharp(x)\ .$$

We define a (very restricted) polynomial interpretation $\mathcal{A}$ as follows: $\mathsf{full}^\sharp_{\mathcal{A}}(m) = \mathsf{full}_{\mathcal{A}}(m) = m$, $\mathsf{s}_{\mathcal{A}}(m) = m + 1$, $\mathsf{cons}_{\mathcal{A}}(m, n) = \mathsf{0}_{\mathcal{A}} = 0$, where $\mathcal{R}_5 \subseteq \succcurlyeq_{\mathcal{A}}$ and $\mathsf{DP}(\mathcal{R}_5) \subseteq >_{\mathcal{A}}$, and $(\succcurlyeq_{\mathcal{A}}, >_{\mathcal{A}})$ forms a reduction pair. Thus the dependency pair complexity function with respect to $\mathcal{R}_5$ is at most linear.

Note that from the proof of Theorem 5.13 one can distill the following three facts, where each of them is responsible for one of the exponentials in the upper-bound:

(1) the number of nodes in a progenitor graph may be exponential in its height,
(2) the size of a term may be exponential in its depth, and
(3) the number of terms of size n is exponential in n.

For an optimal example, we would have to utilise all three criteria, while the just given TRS $\mathcal{R}_5$ utilises only the criteria (1) and (2). To us, it seems impossible to enumerate enough terms of exponential depth and double exponential size so that this is possible. Moreover, we believe that the first and the last criterion can be merged into a single exponential, as shown for string rewriting in the next section. Hence, we conjecture that the upper-bound given in Theorem 5.13 can be improved to a double exponential one.

6. String Rewriting

In this short section we consider *string rewrite systems* (*SRSs* for short), i.e., TRSs where all function symbols are unary or nullary.[1] Since the size and the height of strings are linearly (and not just exponentially) related, the upper bound from Theorem 5.13 immediately breaks down to a double exponential one. However, we can further improve this bound to a

[1]This is sometimes called *unary rewriting*, as opposed to "true" string rewriting, where only unary function symbols are allowed. The results presented in this section hold for both flavours of string rewriting.

single exponential one. Showing this is the purpose of this section. Our main proof idea is to relate rewrite steps and nodes in the progenitor graph directly.

As in Section 5, we fix a finite SRS $\mathcal{S}$, a derivation $A : t_1 \to^* t_n$ over $\mathcal{S}$, and the progenitor graph G of A. In this section, the terms $t_1, \ldots, t_n$ do not contain any function symbols with arity greater than 1. Therefore, each of them only consists of a single branch, which in turn is its main branch.

Lemma 6.1. *The number of nodes in G is at least $n - 1$.*

Proof. For each $1 \leqslant k \leqslant n-1$, let p_k be the redex position of the step $t_k \to t_{k+1}$, so that $\mathsf{rt}(t_k|_{p_k})$ is defined.

Since we consider string rewriting, p_k is in the main branch of t_k. Hence by Lemma 5.6, there exists a node (t_i, p) in G that covers (t_k, p_k). Moreover, for any j $(i \leqslant j < k)$, the single element of $(t_j \to^* t_k) \ni p_k$ and p_j do not coincide. Otherwise (t_j, p_j) would be a successor of (t_i, p) that covers (t_k, p_k). This would contradict the choice of (t_i, p). Therefore, by Lemma 5.5, we have $p \ni (t_i \to^* t_k) = \{p_k\}$. This yields a one to one correspondence between all $n-1$ redex positions and a subset of the nodes of G, entailing the lemma. □

Theorem 6.2. *Let $\mathcal{S}$ be terminating and let $f(n) := \max\{\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{S})/\mathcal{S}}) \mid |t| \leqslant n\}$. Then there exists $d \in \mathbb{N}$ such that $\mathsf{dc}_{\mathcal{S}}(n) \leqslant n \cdot 2^{d \cdot (f(n)+1)}$.*

Proof. Recall the branching constant $C = \max(\{2\} \cup \{\mathsf{dp}(r)+1 \mid l \to r \in \mathcal{S}\})$. Let $A\colon s \to^n_{\mathcal{S}} t$ denote any derivation with respect to $\mathcal{S}$.

Let G be the progenitor graph of A. G has k many connected components, where k is the number of defined symbols in s. By Lemma 5.10, each of them contains at most $C^{\mathsf{DPI}(A')+1}$ many nodes, where A' is an implicit dependency pair derivation corresponding to A such that $\mathsf{DPI}(A')$ is maximal. Hence the total size of G is most $k \cdot C^{\mathsf{DPI}(A')+1}$. By Lemma 6.1, the size of G is at least $n-1$, from which we obtain:

$$n \leqslant k \cdot C^{\mathsf{DPI}(A')+1} + 1 \leqslant |s| \cdot C^{\mathsf{DPI}(A')+1} + 1 \,.$$

We obtain that the length n of A is less than or equal to $|s| \cdot C^{\mathsf{DPI}(A')+1} + 1$. From this the theorem is immediate. □

Thus, for string rewriting, termination proofs by the basic dependency pair method combined with some base technique and an argument filtering imply a complexity bound that is only single exponential in the derivational complexity of the base technique. If polynomial interpretations are used as base technique, then the derivational complexity of the SRS under consideration is bounded by $2_3(\mathsf{O}(n))$. If the base technique is KBO, then the derivational complexity of the SRS is bounded by $\mathrm{Ack}(\mathsf{O}(n), 0)$.

Example 6.3. If we restrict $\mathcal{R}_5$ to its first rule (i.e., we consider $\mathcal{R}_3$), we can see in the same way as in Example 5.15 that $\mathsf{dc}_{\mathcal{R}_3}$ is at least exponential, and the dependency pair complexity function with respect to $\mathcal{R}_3$ is at most linear. Therefore, the upper bound given in Theorem 6.2 is tight.

This concludes our complexity analysis of the basic dependency pair method. The purpose of the next sections is to analyse the usable rules and dependency graph refinements.

7. Usable Rules

In this section we extend the results in Section 5 to termination proofs employing the dependency pair method in connection with the usable rules criterion, c.f. Proposition 3.3.

Notation. For the rest of this section, we use the following constants depending only on the TRS $\mathcal{R}$. Let $E := \max\{a, b, 2 \cdot c\} + 3$, where a is the maximum arity of all function symbols, b the number of rules in $\mathcal{R}$, c is chosen such that it is larger than the size of any right hand side of any rule in $\mathcal{R}$, and hence also larger than the number of occurrences of any variable on the right hand side. Furthermore let $F := \max\{a, D\}$, where D is defined as in Theorem 5.13.

Perhaps surprisingly, the usable rules criterion strengthens the power of the termination technique considerably, as witnessed by the following example.

Example 7.1. Consider the TRS $\mathcal{R}_6$:

$$\begin{aligned} \mathsf{d}(\mathsf{0}) &\to \mathsf{0} & \mathsf{e}(\mathsf{0}, x) &\to x \\ \mathsf{d}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) & \mathsf{e}(\mathsf{s}(x), y) &\to \mathsf{e}(x, \mathsf{d}(y)) \ . \end{aligned}$$

The dependency pairs of $\mathcal{R}_6$ are given by $\mathsf{d}^\sharp(\mathsf{s}(x)) \to \mathsf{d}^\sharp(x)$, $\mathsf{e}^\sharp(\mathsf{s}(x), y) \to \mathsf{e}^\sharp(x, \mathsf{d}(y))$, and $\mathsf{e}^\sharp(\mathsf{s}(x), y) \to \mathsf{d}^\sharp(y)$.

We have the following usable rules with respect to $\mathcal{R}_6$:

$$\mathsf{d}(\mathsf{0}) \to \mathsf{0} \qquad \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \ .$$

The rules $\mathsf{DP}(\mathcal{R}_6) \cup \mathcal{U}(\mathsf{DP}(\mathcal{R}_6)) \cup \mathcal{C}_\epsilon$ admit only double exponentially many dependency pair steps from any starting term $t^\sharp$ with $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Consider for instance the algebra $\mathcal{A}$ over $\mathbb{N}$ defined as follows:

$$\begin{aligned} \mathsf{e}^\sharp_\mathcal{A}(m, n) &= 2^m \cdot (n+1) + 1 & \mathsf{d}^\sharp_\mathcal{A}(m) &= m & \mathsf{d}_\mathcal{A}(m) &= 2 \cdot m \\ \mathsf{0}_\mathcal{A} &= 0 & \mathsf{s}_\mathcal{A}(m) &= m + 1 & \mathsf{cons}_\mathcal{A}(m, n) &= m + n \ . \end{aligned}$$

It is easy to check that $\mathsf{DP}(\mathcal{R}_6) \subseteq >_\mathcal{A}$ and $\mathcal{U}(\mathsf{DP}(\mathcal{R}_6)) \cup \mathcal{C}_\epsilon \subseteq \geqslant_\mathcal{A}$, and for any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $[\alpha]_\mathcal{A}(t^\sharp)$ is double exponentially bounded in $|t|$.

On the other hand the derivational complexity with respect to $\mathcal{R}_6$ is clearly super-exponential. Observe that also the rules $\mathsf{DP}(\mathcal{R}_6) \cup \mathcal{R}_6$ allow a super-exponential number of DP-steps, e.g. for the family of starting terms $\mathsf{e}^\sharp(E^k(\mathsf{0}), \mathsf{s}(\mathsf{0}))$, where $E(x)$ is a shorthand for $\mathsf{e}(x, \mathsf{s}(\mathsf{0}))$.

Note that in Example 7.1 it is essential that arbitrary starting terms, as for example $\mathsf{e}(E^k(\mathsf{0}), \mathsf{s}(\mathsf{0}))$, are considered. If we would restrict the starting terms to *basic* terms, i.e., terms of the form $f(t_1, \ldots, t_n)$ such that f is defined and $t_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ for all $1 \leqslant i \leqslant n$, then the results from Section 5 directly extend to Proposition 3.3. This is a consequence of [16, Lemma 16]. We can generalise Example 7.1 to primitive recursion by employing the Ackermann function.

Example 7.2. We employ a unary notation for the Ackermann function: we write $\mathrm{Ack}_i(x)$ instead of $\mathrm{Ack}(i, x)$. Consider the following family of TRSs $\mathcal{R}_7(l)$, parametrised by $l \in \mathbb{N}$. Here we assume $0 \leqslant i < l$.

$$\begin{aligned} \mathsf{Ack}_0(x) &\to \mathsf{s}(x) & \mathsf{I}(\mathsf{0}, x) &\to \mathsf{Ack}_l(x) \\ \mathsf{Ack}_{i+1}(\mathsf{0}) &\to \mathsf{Ack}_i(\mathsf{s}(\mathsf{0})) & \mathsf{I}(\mathsf{s}(x), y) &\to \mathsf{I}(x, \mathsf{Ack}_l(y)) \\ \mathsf{Ack}_{i+1}(\mathsf{s}(x)) &\to \mathsf{Ack}_i(\mathsf{Ack}_{i+1}(x)) \ . \end{aligned}$$

Note that the last two rules are not contained in $\mathcal{U}(\mathsf{DP}(\mathcal{R}_7(l)))$. The number of dependency pair steps admitted by the rules $\mathsf{DP}(\mathcal{R}_7(l)) \cup \mathcal{U}(\mathsf{DP}(\mathcal{R}_7(l))) \cup \mathcal{C}_\epsilon$ from any starting term $t^\sharp$ with $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is then bounded by $\mathrm{Ack}^2_{l+1}(\mathsf{O}(|t|))$, as witnessed by the algebra $\mathcal{A}$ over $\mathbb{N}$, defined as follows.

$$(\mathsf{Ack}^\sharp_i)_\mathcal{A}(m) = \mathrm{Ack}_i(m) + i \qquad \mathsf{I}^\sharp_\mathcal{A}(m,n) = \mathrm{Ack}^{m+1}_l(n) + m + l + 1 \qquad \mathsf{s}_\mathcal{A}(m) = m + 1$$
$$(\mathsf{Ack}_i)_\mathcal{A}(m) = \mathrm{Ack}_i(m) \qquad \mathsf{cons}_\mathcal{A}(m,n) = m + n \qquad \mathsf{0}_\mathcal{A} = 0$$

It is easy to check that $\mathsf{DP}(\mathcal{R}_7(l)) \subseteq >_\mathcal{A}$ and $\mathcal{U}(\mathsf{DP}(\mathcal{R}_7(l))) \cup \mathcal{C}_\epsilon \subseteq \geqslant_\mathcal{A}$. On the other hand the derivational complexity with respect to $\mathcal{R}_7(l)$ is bounded from below by $\mathrm{Ack}_{l+2}(\Omega(n))$, as witnessed by derivations starting from the family of terms $F^k(\mathsf{0})$, where $F(x)$ is a shorthand for $\mathsf{I}(x, \mathsf{s}(\mathsf{0}))$.

It follows that the derivational complexity of the base technique used in the termination proof of $\bigcup_{i=0}^{l} \mathcal{R}_7(l)$ belongs to level $l+1$ of the Ackermann function, while the derivational complexity of the considered TRS belongs to level $l+2$.

Due to Examples 7.1 and 7.2 we cannot have an elementary relationship between the derivational complexity of the original TRS $\mathcal{R}$ and the complexity induced by the termination technique employed in conjunction with Proposition 3.3.

Still, we can give an upper bound on the derivational complexity with respect to $\mathcal{R}$. This follows from a close study of the correctness proof of Proposition 3.3 given in [15], compare also [12]. The main ingredient of this proof is the definition of the interpretation $\mathcal{I}_\mathcal{G}$.

Definition 7.3 ([15])**.** Let $\mathcal{G} \subseteq \mathcal{F}$. The *interpretation* $\mathcal{I}_\mathcal{G}$ is a mapping from terminating terms in $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{V})$ to terms in $\mathcal{T}(\mathcal{F}^\sharp \cup \{\mathsf{nil}, \mathsf{cons}\}, \mathcal{V})$, where nil is a fresh function symbol and cons is the function symbol introduced by $\mathcal{C}_\epsilon$, inductively defined as follows:

$$\mathcal{I}_\mathcal{G}(t) := \begin{cases} t & \text{if } t \text{ is a variable} \\ f(\mathcal{I}_\mathcal{G}(t_1), \ldots, \mathcal{I}_\mathcal{G}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin \mathcal{G} \\ \mathsf{cons}(f(\mathcal{I}_\mathcal{G}(t_1), \ldots, \mathcal{I}_\mathcal{G}(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \mathcal{G} \end{cases}$$

where in the last clause t' denotes the term $\mathsf{order}(\{\mathcal{I}_\mathcal{G}(u) \mid t \to_\mathcal{R} u\})$ with

$$\mathsf{order}(T) := \begin{cases} \mathsf{nil} & \text{if } T = \varnothing \\ \mathsf{cons}(t, \mathsf{order}(T - \{t\})) & \text{if } t \text{ is the minimum element of } T \end{cases}$$

Here an arbitrary but fixed total order on $\mathcal{T}(\mathcal{F}^\sharp \cup \{\mathsf{nil}, \mathsf{cons}\}, \mathcal{V})$ is assumed.

According to [15, Theorem 20], any $\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}$-derivation starting from t can be transformed into a $\mathsf{DP}(\mathcal{R}) \cup \mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathcal{C}_\epsilon$-derivation starting from $\mathcal{I}_\mathcal{G}(t)$, where $\mathcal{G}$ is the set of defined symbols of $\mathcal{R} - \mathcal{U}(\mathsf{DP}(\mathcal{R}))$. Therefore, estimating $|\mathcal{I}_\mathcal{G}(t)|$ is the key to the connection between $\mathsf{dh}(t, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})$ and $\mathsf{dh}(t, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{U}(\mathsf{DP}(\mathcal{R}))\cup\mathcal{C}_\epsilon})$. Suppose there exists a function f that bounds $\mathsf{dh}(t, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{U}(\mathsf{DP}(\mathcal{R}))\cup\mathcal{C}_\epsilon})$ in $|t|$. Then $\mathsf{dh}(t, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})$ can be bounded in $|t|$ by $f(|\mathcal{I}_\mathcal{G}(t)|)$.

However, the difficulty of this estimation lies in the following mutual dependence between the definition of the interpretation $\mathcal{I}_\mathcal{G}$ and the derivation height. On one hand, we bound $\mathsf{dh}(t, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})$ in $|t|$ by $f(|\mathcal{I}_\mathcal{G}(t)|)$. On the other hand, $\mathcal{I}_\mathcal{G}(t)$ depends on $\mathsf{dh}(t, \to_\mathcal{R})$ since $\mathsf{dh}(t, \to_\mathcal{R})$ determines the number of recursive calls of the shape $\{\mathcal{I}_\mathcal{G}(u) \mid t \to_\mathcal{R} u\}$ in the definition of $\mathcal{I}_\mathcal{G}(t)$. The following sequence of lemmata shows how this mutual dependence can be resolved.

Definition 7.4. Let $g\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the function satisfying the following recursive definition:

$$g(m,n) := \begin{cases} E & \text{if } m = 0 \\ E \cdot g(m-1,0) & \text{if } m > 0 \text{ and } n = 0 \\ E \cdot g(m-1,n) + E \cdot m \cdot g(E \cdot m, n-1) & \text{otherwise .} \end{cases}$$

The next two lemmata estimate the size $|\mathcal{I}_{\mathcal{G}}(t)|$ of the interpretation $\mathcal{I}_{\mathcal{G}}(t)$ in the size of t and the derivation height of t (with respect to $\mathcal{R}$).

Lemma 7.5.
(1) *The function g is well-defined and strictly monotone in each argument.*
(2) *For all m, n: $E \leqslant g(m,n)$.*
(3) *For any term t: $|\mathcal{I}_{\mathcal{G}}(t)| \leqslant g(|t|, \mathsf{dh}(t, \to_{\mathcal{R}}))$.*

Proof. We only show property (3). The proof proceeds by induction on the lexicographic order over the pair $(\mathsf{dh}(t, \to_{\mathcal{R}}), |t|)$. It suffices to consider the interesting case, where $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{G}$ and $\mathsf{dh}(t, \to_{\mathcal{R}}) > 0$. We obtain

$$\begin{aligned} |\mathcal{I}_{\mathcal{G}}(t)| &= 2 + \sum_{i=1}^{n} |\mathcal{I}_{\mathcal{G}}(t_i)| + |\mathsf{order}(\{\mathcal{I}_{\mathcal{G}}(u) \mid t \to_{\mathcal{R}} u\})| \\ &\leqslant 2 + \sum_{i=1}^{n} |\mathcal{I}_{\mathcal{G}}(t_i)| + 1 + b \cdot |t| \cdot (1 + \max\{|\mathcal{I}_{\mathcal{G}}(u)| \mid t \to_{\mathcal{R}} u\}) \\ &\leqslant 3 + n \cdot \max_{1 \leqslant i \leqslant n} \{g(|t_i|, \mathsf{dh}(t_i, \to_{\mathcal{R}}))\} + b \cdot |t| \cdot (1 + \max\{g(|u|, \mathsf{dh}(u, \to_{\mathcal{R}})) \mid t \to_{\mathcal{R}} u\}) \\ &\leqslant 3 + a \cdot g(|t| - 1, \mathsf{dh}(t, \to_{\mathcal{R}})) + b \cdot |t| + b \cdot |t| \cdot g(c \cdot |t| + c, \mathsf{dh}(t, \to_{\mathcal{R}}) - 1) \\ &\leqslant E \cdot g(|t| - 1, \mathsf{dh}(t, \to_{\mathcal{R}})) + E \cdot |t| \cdot g(E \cdot |t|, \mathsf{dh}(t, \to_{\mathcal{R}}) - 1) \\ &= g(|t|, \mathsf{dh}(t, \to_{\mathcal{R}})) \ . \end{aligned}$$

In the second line we use the fact that any term t has at most $b \cdot |t|$ many reducts. In the third line, we apply the induction hypothesis. In the fourth line, we use that $|u| \leqslant c \cdot |t| + c$ whenever $t \to_{\mathcal{R}} u$. □

Lemma 7.6. *Let g be defined as in Lemma 7.5 above. Then there exists a minimal $d \in \mathbb{N}$ such that for all $m, n \in \mathbb{N}$, we have $g(m,n) \leqslant 2^{2^{d \cdot (m+n+1)}} =: G(m,n)$.*

Proof. It can be shown by straightforward induction on the lexicographic order over the pair (m,n) that $g(m,n) \leqslant (E \cdot (n+1))^{(n+1) \cdot E^{2 \cdot m + 1}}$ holds. It is easy to see that for suitable d we have $(E \cdot (n+1))^{(n+1) \cdot E^{2 \cdot m + 1}} \leqslant 2^{2^{d \cdot (m+n+1)}}$. Thus the lemma follows. □

For the remainder of the section, let the function G be defined as in Lemma 7.6 above. Then we define the function $H[f]\colon \mathbb{N} \to \mathbb{N}$ parametrised in a mapping f from the naturals to the naturals as follows:

$$H[f](m) := f(1 + F \cdot G(m, h(m,m))) \ ,$$

where $h(m,n) := 2^{2^{m \cdot 2^{F \cdot (n+2)}}}$.

Lemma 7.7. *Let $\mathcal{R}$ be terminating and let the function f be defined by $f(n) := \max(\{n\} \cup \{\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathcal{C}_\epsilon}) \mid |t| \leqslant n\})$. Then $\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \leqslant (H[f])^{|t|}(1)$.*

Proof. We show the lemma by induction on $t^\sharp$. If $t^\sharp = t$ is a variable, the lemma is trivial. Otherwise, assume $t^\sharp = f^\sharp(t_1, \dots, t_n)$ and recall that $f^\sharp \notin \mathcal{G}$ for any $f \in \mathcal{F}$. Due to Definition 7.3 in conjunction with Lemma 7.5 there exists $i \in \{1, \dots, n\}$ such that

$$|\mathcal{I}_\mathcal{G}(t^\sharp)| = |f^\sharp(\mathcal{I}_\mathcal{G}(t_1), \dots, \mathcal{I}_\mathcal{G}(t_n))| \leqslant 1 + F \cdot g(|t_i|, \mathsf{dh}(t_i, \to_\mathcal{R})) \; .$$

By Theorem 5.13, we have

$$\mathsf{dh}(t_i, \to_\mathcal{R}) \leqslant 2^{2^{|t_i| \cdot 2^{D \cdot (\mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) + 2)}}} \; .$$

Due to $D \leqslant F$, we conclude $\mathsf{dh}(t_i, \to_\mathcal{R}) \leqslant h(|t_i|, \mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}))$.

As mentioned above, any $\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}$-derivation starting from t can be transformed into a $\mathsf{DP}(\mathcal{R}) \cup \mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathcal{C}_\epsilon$-derivation starting from $\mathcal{I}_\mathcal{G}(t)$. Thus, we also have $\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \leqslant f(|\mathcal{I}_\mathcal{G}(t^\sharp)|)$. In sum we obtain:

$$\begin{aligned}
\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) &\leqslant f(|\mathcal{I}_\mathcal{G}(t^\sharp)|) \\
&\leqslant f(1 + F \cdot g(|t_i|, \mathsf{dh}(t_i, \to_\mathcal{R}))) \\
&\leqslant f(1 + F \cdot g(|t_i|, h(|t_i|, \mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})))) \\
&\leqslant f(1 + F \cdot G(|t_i|, h(|t_i|, \mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})))) \\
&\leqslant H[f](\max\{|t_i|, \mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})\}) \; .
\end{aligned}$$

It is easy to verify that $|t_i| \leqslant (H[f])^{|t_i|}(1) \leqslant (H[f])^{|t|-1}(1)$. Moreover by induction hypothesis we have $\mathsf{dh}(t_i^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \leqslant (H[f])^{|t_i|}(1) \leqslant (H[f])^{|t|-1}(1)$. From this the lemma follows. □

Theorem 7.8. *Let $\mathcal{R}$ be terminating and let*

$$f(n) := \max(\{n\} \cup \{\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\,\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathcal{C}_\epsilon}) \mid |t| \leqslant n\}) \; .$$

Then there exist a function f' which is elementary in f, and an elementary function e such that $\mathsf{dc}_\mathcal{R}(n) \leqslant e(n, (f')^n(1))$.

Proof. We choose $f' = H[f]$ and $e = h$. Let t be a term. By Theorem 5.13 we have that $\mathsf{dh}(t, \to_\mathcal{R}) \leqslant h(|t|, \mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}))$. Furthermore by Lemma 7.7 we obtain that $\mathsf{dh}(t^\sharp, \to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}) \leqslant (H[f])^{|t|}(1)$. Combining these two observations the theorem is immediate. □

Consider any TRS $\mathcal{R}$ whose termination can be shown by the basic dependency pair method and some base technique enhanced by the usable rules criterion. Let f, e, and f' be defined as in Theorem 7.8 and set $j(n) := e(n, (f')^n(1))$. For instance, if polynomial interpretations are used as a base technique, then f is bounded by a double exponential function. Therefore j (and thus also $\mathsf{dc}_\mathcal{R}$) is superexponentially bounded. On the other hand, if LPO is used as a base technique, then f, and hence also j and $\mathsf{dc}_\mathcal{R}$ are bounded by multiply recursive functions. Note that the derivational complexity induced by LPO (as a direct method) is multiply recursive [38]. Clearly the class of multiply recursive functions is closed under primitive recursion. Hence the complexity of the dependency pair method (in conjunction with the usable rules refinement) becomes negligible.

8. Dependency Graphs

We now consider *dependency graphs*, i.e., we establish an upper bound on the complexity of TRSs whose termination can be shown by Proposition 3.4. As already mentioned in the introduction the derivational complexity analysis of Proposition 3.4 does not employ the techniques developed in Sections 5–7, but a conceptually simpler technique. Essentially it suffices to embed the TRS $\mathcal{R}$ in a generic simulating TRS $\mathcal{R}_{\mathsf{sim}}$ (see Definition 8.7), whose derivational complexity can be analysed directly.

Notation. For the rest of this section, we use the following constants depending only on the TRS $\mathcal{R}$. Let k be the number of (trivial and nontrivial) SCCs in $\mathsf{DG}(\mathcal{R})$, a the maximum arity of any function symbol occurring in $\mathcal{R}$, and recall that C denotes the branching constant of $\mathcal{R}$.

At first glance, it might seem that the number of dependency pair steps admitted by a TRS is bounded linearly in the number of dependency pair steps admitted within the "worst" SCC of the dependency graph. However, this is not the case.

Example 8.1. Consider the following family of TRSs, denoted as $\mathcal{R}_8(l)$, and parametrised by $l \in \mathbb{N}$. The system of TRSs $\mathcal{R}_8(l)$ generalises a TRS given by Hofbauer in [19].

$$\mathsf{i}(x) \circ_k (y \circ_{k-1} z) \to x \circ_k (\mathsf{i}(\mathsf{i}(y)) \circ_{k-1} z) \qquad 2 \leqslant k \leqslant l$$
$$\mathsf{i}(x) \circ_k (y \circ_{k-1} (z \circ_{k-2} w)) \to x \circ_k (z \circ_{k-1} (y \circ_{k-2} w)) \qquad 3 \leqslant k \leqslant l$$

For all $m, n \geqslant 0$, set

$$t_{m,n} := \mathsf{i}^{2(n+1)}(\mathsf{e}) \circ_{m+2} (\mathsf{e} \circ_{m+1} (\ldots (\mathsf{e} \circ_1 \mathsf{e}) \ldots)) \,.$$

Then $\mathsf{dh}(t_{m,n}, \to_{\mathcal{R}_8(l)}) \geqslant \mathrm{Ack}(m, n)$, whenever $l \geqslant m + 2$. This follows from Proposition 5.9 in [19]. Hence, for every primitive recursive function f there exists some l such that $\mathsf{dc}_{\mathcal{R}_8(l)}$ dominates f. Due to Theorem 5.13 the same property holds for the dependency pair complexities of the TRSs $\mathcal{R}_8(l)$.

On the other hand, we can show termination of $\mathcal{R}_8(l)$ by orienting every nontrivial SCC of $\mathsf{DG}(\mathcal{R}_8(l))$ by a uniform and restricted polynomial interpretation $\mathcal{A}$. We define $(\circ_k^\sharp)_\mathcal{A}(m, n) = m$, $(\circ_k)_\mathcal{A}(m, n) = 0$, $\mathsf{i}_\mathcal{A}(m) = m + 1$, where $k \in \{1, \ldots, l\}$. Note that $\mathcal{A}$ yields a linear upper bound on the number of dependency pair steps in each SCC.

Remark 8.2. In [27, Section 6] we falsely claimed that the derivational complexity induced by Proposition 3.4 would be *elementary* in the complexity of the base techniques. Example 8.1 contradicts this claim.

Example 8.1 exemplifies the fact that the bound on the maximal number of dependency pair steps possible within a specific SCC $\mathcal{P}$ is related to the size of the considered term at the moment of entering the SCC $\mathcal{P}$, and not to the size of the starting term of the full derivation. Thus, Theorem 5.13 and the techniques developed in the preceding sections, cannot be applied directly. Instead, one needs to argue inductively so that in each step in this induction, Theorem 5.13 is on one hand employed to estimate the number of $\mathcal{R}$-steps and on the other hand used to provide an upper bound on the size of terms.

However, this inductive argument becomes rather involved. Thus we establish a new technique in this section, where we employ a simulating TRS $\mathcal{R}_{\mathsf{sim}}$. In this way the inductive argument becomes hidden in the termination proof of $\mathcal{R}_{\mathsf{sim}}$. The argument needs some

preparations. Let f be a monotone function over $\mathbb{N}$ defined as follows:

$$f(n) := \max(\{1\} \cup \{\mathsf{dh}(t^\sharp, \to_{\mathcal{P}/\mathcal{R}}) \mid |t| \leqslant n, \mathcal{P} \text{ is SCC of } \mathsf{DG}(\mathcal{R})\}) \;, \tag{8.1}$$

such that f dominates the maximal number of $\mathcal{P}/\mathcal{R}$ steps in any SCC $\mathcal{P} \in \mathsf{DG}(\mathcal{R})$. Consider the unary function f defined in (8.1). Note that it is an easy task to define a TRS $\mathcal{R}'$ (employing the constructors s, $\mathsf{0}$) that computes the function f, whenever f is computable. That is, there exist a TRS $\mathcal{R}'$ and a defined function symbol f such that $\mathsf{f}(\mathsf{s}^n(\mathsf{0})) \to^*_{\mathcal{R}'} \mathsf{s}^{f(n)}(\mathsf{0})$.

Note that if f is a primitive recursive function it is straightforward to define the TRS $\mathcal{R}'$ in such a way that the derivational complexity function $\mathsf{dc}_{\mathcal{R}'}$ is primitive recursive [18]. Furthermore it is not difficult to see that this generalises to any class of (computable) functions [13].

Let $\mathcal{P}$ and $\mathcal{Q}$ denote different (trivial or nontrivial) SCCs in $\mathsf{DG}(\mathcal{R})$, respectively. Then we call $\mathcal{Q}$ *reachable* from $\mathcal{P}$ if there exist nodes $u \in \mathcal{P}$, $v \in \mathcal{Q}$ and a path in $\mathsf{DG}(\mathcal{R})$ connecting u with v. Let $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_k$ be all (trivial and nontrivial) SCCs in $\mathsf{DG}(\mathcal{R})$. Let $\mathsf{rk}\colon \{\mathcal{Q}_1, \ldots, \mathcal{Q}_k\} \to \{1, \ldots, k\}$ be an arbitrary but fixed mapping respecting the topological ordering of $\mathsf{DG}(\mathcal{R})$, i.e. $\mathsf{rk}(\mathcal{Q}_i) > \mathsf{rk}(\mathcal{Q}_j)$ whenever $\mathcal{Q}_j$ is reachable from $\mathcal{Q}_i$. We call $\mathsf{rk}(\mathcal{P})$ the *rank* of an SCC $\mathcal{P}$.

Definition 8.3. The *rank of a dependency pair* $s \to t$, denoted by $\mathsf{rk}(s \to t)$, is the rank of $\mathcal{P}$ such that $s \to t \in \mathcal{P}$. Let u be a term and suppose there exists an SCC $\mathcal{P}$ such that $u^\sharp$ is *not* a normal form with respect to $\to_{\mathcal{P}/\mathcal{R}}$. The *rank of the term* u is defined as follows:

$$\mathsf{rk}(u) := \max\{\mathsf{rk}(s \to t) \mid \text{there exists } \sigma \text{ such that } u^\sharp \to^*_{\mathcal{R}} s\sigma\} \;.$$

Observe that $\mathsf{rk}(u)$ need not be defined, although u has a redex at the root position. This is due to the fact that this redex need not be governed by a dependency pair. On the other hand observe that if $u \notin \mathsf{NF}(\mathcal{P}/\mathcal{R})$ for some SCC $\mathcal{P}$, then $\mathsf{rk}(u)$ is defined. Furthermore in this case $\mathsf{rk}(u) > 0$ and $\mathsf{dh}(u, \to_{\mathcal{P}/\mathcal{R}}) > 0$.

Definition 8.4. We define the mapping $\mathsf{dh}^\sharp$ from terms to $\mathbb{N} \times \mathbb{N}$ as follows:

$$\mathsf{dh}^\sharp(t) := \begin{cases} (i, \mathsf{dh}(t^\sharp, \to_{\mathcal{P}_i/\mathcal{R}})) & \text{if } \mathsf{rk}(t) \text{ is defined and } \mathsf{rk}(t) = i \;, \\ (0,1) & \text{if } t^\sharp \in \mathsf{NF}(\mathcal{P}/\mathcal{R}) \text{ for all SCCs } \mathcal{P} \text{ and } \mathsf{rt}(t) \text{ is defined} \;, \\ (0,0) & \text{otherwise} \;. \end{cases}$$

In the following we write $\mathcal{P}_i$ for an SCC with rank i. Note that any SCC in $\mathsf{DG}(\mathcal{R})$ is uniquely defined by its rank. We give some intuition for Definition 8.4 that is made precise in Lemma 8.6 below. Consider a term t and suppose t is not in normal form with respect to $\to_{\mathcal{P}_i/\mathcal{R}}$. Then the second component of $\mathsf{dh}^\sharp(t)$ estimates the remaining rewrite steps with respect to $\mathcal{P}_i$ modulo $\mathcal{R}$. The other cases in Definition 8.4 take care of the possibility that $t \in \mathsf{NF}(\mathcal{Q}/\mathcal{R})$ for all SCCs $\mathcal{Q}$ in $\mathsf{DG}(\mathcal{R})$. Note that, if $\mathsf{dh}^\sharp(t) = (i, j)$ with $i > 0$, then $j > 0$, as well.

Example 8.5 (continued from Example 4.3)**.** There are two dependency pairs with respect to $\mathcal{R}_2$:

$$\mathsf{m}^\sharp(x) \to \mathsf{p}^\sharp(\mathsf{a}, x) \qquad \mathsf{m}^\sharp(x) \to \mathsf{a}^\sharp$$

The dependency graph of $\mathcal{R}_2$ contains no edges, so it only consists of two (trivial) SCCs. Let $\mathsf{rk}(\mathsf{m}^\sharp(x) \to \mathsf{p}^\sharp(\mathsf{a}, x)) = 1$, and $\mathsf{rk}(\mathsf{m}^\sharp(x) \to \mathsf{a}^\sharp) = 2$, so $\mathcal{P}_1 = \{\mathsf{m}^\sharp(x) \to \mathsf{p}^\sharp(\mathsf{a}, x)\}$ and $\mathcal{P}_2 = \{\mathsf{m}^\sharp(x) \to \mathsf{a}^\sharp\}$.

We now give the values of $\mathsf{dh}^\sharp$ for all subterms of t_1 and t_2. All function symbols in t_1 and t_2 are defined, however the terms $\mathsf{a}^\sharp$ and $\mathsf{p}^\sharp(\mathsf{a},\mathsf{a})$ are normal forms with respect to $\to_{\mathsf{DP}(\mathcal{R}_2)/\mathcal{R}_2}$. Therefore, $\mathsf{dh}^\sharp(\mathsf{a}) = \mathsf{dh}^\sharp(\mathsf{p}(\mathsf{a},\mathsf{a})) = (0,1)$. On the other hand, we have

$$\begin{aligned}\mathsf{dh}(\mathsf{m}^\sharp(\mathsf{m}(\mathsf{a})), \to_{\mathcal{P}_2/\mathcal{R}_2}) &= \mathsf{dh}(\mathsf{m}^\sharp(\mathsf{a}), \to_{\mathcal{P}_2/\mathcal{R}_2}) \\ &= \mathsf{dh}(\mathsf{m}^\sharp(\mathsf{p}(\mathsf{a},\mathsf{a})), \to_{\mathcal{P}_2/\mathcal{R}_2}) = 1 \ .\end{aligned}$$

Thus, $\mathsf{dh}^\sharp(\mathsf{m}(\mathsf{m}(\mathsf{a}))) = \mathsf{dh}^\sharp(\mathsf{m}(\mathsf{a})) = \mathsf{dh}^\sharp(\mathsf{m}(\mathsf{p}(\mathsf{a},\mathsf{a}))) = (2,1)$.

We write $>^{\mathrm{lex}}$ for the lexicographic extension of the standard order $>$ on the natural numbers.

Lemma 8.6. *Let $A\colon s \to_{\mathcal{R}} t$, let $p \in \mathcal{P}\mathsf{os}(s)$, and let $q \in \mathcal{P}\mathsf{os}(t)$. Suppose that $q \in p \setminus\!\setminus A$. Then $\mathsf{dh}^\sharp(s|_p) \geqslant^{\mathrm{lex}} \mathsf{dh}^\sharp(t|_q)$. Let p' be the redex position of A. If $p = p'$, then $\mathsf{dh}^\sharp(s|_p) >^{\mathrm{lex}} \mathsf{dh}^\sharp(t|_q)$.*

Proof. By assumption $q \in p \setminus\!\setminus A$. Suppose that $\mathsf{rt}(t|_q)$ is defined. Otherwise let $\mathsf{rt}(t|_q) \in \mathcal{C}$. Then by Definition 8.4, $\mathsf{dh}^\sharp(t|_q) = (0,0)$ and the lemma is trivial. Hence we assume $\mathsf{rt}(t|_q)$ is defined. Then by Lemma 4.4 $(s|_p)^\sharp \to^{=}_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}} (t|_q)^\sharp$.

Suppose further $p = p'$. Then $(s|_p)^\sharp \to_{\mathsf{DP}(\mathcal{R})} (t|_q)^\sharp$ follows from the proof of Lemma 4.4. Thus $\mathsf{rk}(s|_p)$ is defined and $\mathsf{rk}(s|_p) = i > 0$. By Definition 8.3 we have $(s|_p)^\sharp \to_{\mathcal{P}_j} (t|_q)^\sharp$ for some $j \leqslant i$. Thus $i > \mathsf{rk}(t|_q)$, or $i = \mathsf{rk}(t|_q)$ and

$$\mathsf{dh}((s|_p)^\sharp, \to_{\mathcal{P}_i/\mathcal{R}}) > \mathsf{dh}((t|_q)^\sharp, \to_{\mathcal{P}_i/\mathcal{R}}) \ .$$

The lemma follows.

Suppose otherwise $p \neq p'$. Then

$$\mathsf{dh}^\sharp(s|_p) \geqslant^{\mathrm{lex}} \mathsf{dh}^\sharp(t|_q) \ ,$$

by Definition 8.4. This concludes the proof of the lemma. □

The simulating TRS $\mathcal{R}_{\mathsf{sim}}$ is based on a mapping tr (see Definition 8.8 below) such that $s \to_{\mathcal{R}} t$ implies $\mathsf{tr}(s) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{tr}(t)$. Essentially, $\mathsf{tr}(t)$ encodes $\mathsf{dh}^\sharp(t')$ for all subterms t' of t. This is done by transforming t into a term with an $a+1$-ary root symbol g_i, where i is the first component of $\mathsf{dh}^\sharp(t)$, the first argument of g_i encodes the second component of $\mathsf{dh}^\sharp(t)$, and the remaining arguments contain the transformations of the direct subterms of t.

The main tool for achieving the simulation of a rewrite step $s \to_{\mathcal{R}} t$ are rules which create the progenies of the redex position p' of the step. The set $p' \setminus\!\setminus (s \to_{\mathcal{R}} t)$ contains at most a^C many elements. We indicate how this behaviour is overapproximated in derivations over $\mathcal{R}_{\mathsf{sim}}$. For instance, if $a = 2$ and $C = 2$, we make use of the following derivation:

$$\mathsf{g}_i(\mathsf{s}(x), x_1, x_2) \to^* \mathsf{g}_i(\mathsf{g}_i(x, \mathsf{g}_i(x, x_1, x_2), \mathsf{g}_i(x, x_1, x_2)), \mathsf{g}_i(x, \mathsf{g}_i(x, x_1, x_2), \mathsf{g}_i(x, x_1, x_2))) \ .$$

Recall the above definition of the TRS $\mathcal{R}'$ computing the function f defined in (8.1). The definition of the simulating TRS $\mathcal{R}_{\mathsf{sim}}$ employs the TRS $\mathcal{R}'$.

Definition 8.7. Consider the following TRS $\mathcal{R}_{\mathsf{sim}}$, where $0 \leqslant i \leqslant k$, $1 \leqslant i' \leqslant k$, and $1 \leqslant j \leqslant a$.

$$
\begin{aligned}
1_i\colon &\quad \mathsf{g}_i(\mathsf{s}(x), x_1, \dots, x_a) &&\to \mathsf{tree}_i(\mathsf{s}^C(\mathsf{0}), x, x_1, \dots, x_a)\\
2_{i'}\colon &\quad \mathsf{g}_{i'}(x, x_1, \dots, x_a) &&\to \mathsf{g}_{i'-1}(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(\mathsf{0}, x_1, \dots, x_a))), x_1, \dots, x_a)\\
3_{i,j}\colon &\quad \mathsf{size}(\mathsf{g}_i(x, x_1, \dots, x_a)) &&\to \mathsf{d}_a(\mathsf{size}(x_j))\\
4\colon &\quad \mathsf{size}(\mathsf{c}) &&\to \mathsf{s}(\mathsf{0})\\
5\colon &\quad \mathsf{d}_a(\mathsf{s}(x)) &&\to \mathsf{s}^a(\mathsf{d}_a(x))\\
6\colon &\quad \mathsf{d}_a(\mathsf{0}) &&\to \mathsf{0}\\
7\colon &\quad \mathsf{g}_0(x, x_1, \dots, x_a) &&\to \mathsf{c}\\
8_{i,j}\colon &\quad \mathsf{g}_i(x, x_1, \dots, x_a) &&\to x_j\\
9\colon &\quad \mathsf{g}(x) &&\to \mathsf{g}_k(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(\mathsf{0}, x, \dots, x))), x, \dots, x)\\
10\colon &\quad \mathsf{z} &&\to \mathsf{g}_k(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(\mathsf{0}, \mathsf{c}, \dots, \mathsf{c}))), \mathsf{c}, \dots, \mathsf{c})\\
11_i\colon &\quad \mathsf{tree}_i(\mathsf{0}, x, x_1, \dots, x_a) &&\to \mathsf{g}_i(x, x_1, \dots, x_a)\\
12_i\colon &\quad \mathsf{tree}_i(\mathsf{s}(y), x, x_1, \dots, x_a) &&\to \mathsf{g}_i(x, \mathsf{tree}_i(y, x, x_1, \dots, x_a), \dots, \mathsf{tree}_i(y, x, x_1, \dots, x_a))\ .
\end{aligned}
$$

These rules are augmented by $\mathcal{R}'$ defining the function symbol f. Without loss of generality we can assume that the signatures of $\mathcal{R}'$ and $\mathcal{R}_{\mathsf{sim}}$ are disjoint with the exception of f and the constructors s and $\mathsf{0}$.

Observe that $\mathcal{R}_{\mathsf{sim}}$ depends only on the constants a, C, k, and the function f. Some comments: The rules 1_i ($0 \leqslant i \leqslant k$) are the main rules for the simulation of the effects of a single step $s \to_{\mathcal{R}} t$ in $\mathcal{R}_{\mathsf{sim}}$. These rules have already been motivated above. The rules $2_{i'}$ ($1 \leqslant i' \leqslant k$) simulate that each of the new positions q created by $s \to_{\mathcal{R}} t$ might be of rank j ($i' > j$). Observe that by definition of the function f we have

$$f(|t|_q|) \geqslant \mathsf{dh}((t|_q)^\sharp, \to_{\mathcal{P}_j/\mathcal{R}})\ , \tag{8.2}$$

which explains the occurrence of the first argument of the right-hand side of these rules. The rules $3_{i,j}$–6 ($0 \leqslant i \leqslant k$, $1 \leqslant j \leqslant a$) define the function symbol size, that is, $\mathsf{size}(s)$ reduces to a numeral $\mathsf{s}^l(\mathsf{0})$ such that $l \geqslant |s|$, see Lemma 8.10(3) below. The rules 7–$8_{i,j}$ ($0 \leqslant i \leqslant k$, $1 \leqslant j \leqslant a$) make sure that any superfluous positions and copies of subterms created by the rules of type 1_i can be deleted. The rules 9 and 10 guarantee that the simulating derivation can be started with a suitably small initial term. Note that it is in general not the case that we have $|\mathsf{tr}(s)| \leqslant |s|$, compare Definition 8.8 below. Finally, the rules 11_i–12_i define the function symbols tree_i introduced by the rules 1_i, which essentially unfold a full a-ary tree using the function symbol g_i.

Through a sequence of lemmata we show that the TRS $\mathcal{R}_{\mathsf{sim}}$ indeed simulates $\mathcal{R}$ as requested. Let $\mathcal{F}$, $\mathcal{F}_{\mathsf{sim}}$ denote the signatures of $\mathcal{R}$ and $\mathcal{R}_{\mathsf{sim}}$, respectively.

Definition 8.8. The mapping $\mathsf{tr}\colon \mathcal{T}(\mathcal{F}) \to \mathcal{T}(\mathcal{F}_{\mathsf{sim}})$ is defined as follows. Suppose $t = f(t_1, \dots, t_n)$ and $\mathsf{dh}^\sharp(t) = (i, l)$. Then we define:

$$\mathsf{tr}(t) := \mathsf{g}_i(\mathsf{s}^l(\mathsf{0}), \mathsf{tr}(t_1), \dots, \mathsf{tr}(t_n), \mathsf{c}, \dots, \mathsf{c})\ .$$

Note that for any constant t, we have $\mathsf{tr}(t) = \mathsf{g}_i(\mathsf{s}^l(\mathsf{0}), \mathsf{c}, \dots, \mathsf{c})$ for some $i, l \in \mathbb{N}$. To simplify the presentation, we often compress sequences of c as follows: $\mathsf{tr}(t) = \mathsf{g}_i(\mathsf{s}^l(\mathsf{0}), \overline{\mathsf{c}})$. We exemplify the role played by the simulating TRS $\mathcal{R}_{\mathsf{sim}}$ below.

Example 8.9 (continued from Example 8.5)**.** Using the results of $\mathsf{dh}^\sharp$ for all subterms of t_1 and t_2, we get

$$\mathsf{tr}(\mathsf{m}(\mathsf{m}(\mathsf{a}))) = \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c}), \mathsf{c})$$
$$\mathsf{tr}(\mathsf{m}(\mathsf{p}(\mathsf{a}, \mathsf{a}))) = \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}})), \mathsf{c}) \ .$$

For this example, we use $\mathcal{R}_{\mathsf{sim}}$ with the parameters $k = a = C = 2$, and $f(n) = 1$. Hence, a suitable TRS $\mathcal{R}'$ for defining f consists of the single rewrite rule:

$$\mathsf{f}(x) \to \mathsf{s}(0) \ .$$

The following derivation over $\mathcal{R}_{\mathsf{sim}} \cup \mathcal{R}'$ rewrites $\mathsf{tr}(t_1)$ into $\mathsf{tr}(t_2)$, The underlined part in each term is the redex used in the next step in the derivation. In each rewrite step using a rule from $\mathcal{R}_{\mathsf{sim}}$, the applied rule is indicated.

$$\begin{aligned}
&\mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}, \mathsf{c}) \\
&\quad \to_{1_2} \mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{tree}_2(\mathsf{s}(\mathsf{s}(0)), 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}, \mathsf{c}) \\
&\quad \to_{12_2} \mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{g}_2(0, \mathsf{tree}_2(\mathsf{s}(0), 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c}), \mathsf{tree}_2(\mathsf{s}(0), 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c}))}, \mathsf{c}) \\
&\quad \to_{8_{2,1}} \mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{tree}_2(\mathsf{s}(0), 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}, \mathsf{c}) \\
&\quad \to_{12_2} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_2(0, \underline{\mathsf{tree}_2(0, 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}, \mathsf{tree}_2(0, 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})), \mathsf{c}) \\
&\quad \to_{11_2} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_2(0, \underline{\mathsf{g}_2(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}, \mathsf{tree}_2(0, 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})), \mathsf{c}) \\
&\quad \to_{8_{2,1}} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_2(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \underline{\mathsf{tree}_2(0, 0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}), \mathsf{c}) \\
&\quad \to_{11_2} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_2(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \underline{\mathsf{g}_2(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{c})}), \mathsf{c}) \\
&\quad \to_{8_{2,1}} \mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{g}_2(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}))}, \mathsf{c}) \\
&\quad \to_{2_2} \mathsf{g}_2(\mathsf{s}(0), \underline{\mathsf{g}_1(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}})))), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}))}, \mathsf{c}) \\
&\quad \to_{2_1} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_0(\underline{\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}))))}, \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}})), \mathsf{c}) \\
&\quad \to_{\mathcal{R}'} \mathsf{g}_2(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}}), \mathsf{g}_0(\mathsf{s}(0), \overline{\mathsf{c}})), \mathsf{c}) \ .
\end{aligned}$$

In the remainder of this section we show that the derivational complexity of $\mathcal{R}_{\mathsf{sim}}$, and thus of $\mathcal{R}$, is primitive recursive in f. We define the equivalence $s \approx t$ on $\mathcal{T}(\mathcal{F}_{\mathsf{sim}})$. If $s = \mathsf{c}$, then $t = \mathsf{c}$. Otherwise if $s = \mathsf{g}_i(\mathsf{s}^m(0), s_1, \ldots, s_a)$, then $t = \mathsf{g}_{i'}(\mathsf{s}^n(0), t_1, \ldots, t_a)$, such that $m, n \in \mathbb{N}$, $1 \leqslant i, i' \leqslant k$ and $s_j \approx t_j$ for all $1 \leqslant j \leqslant a$.

Lemma 8.10. *Let $i \in \{0, \ldots, k\}$. Then the following properties of $\mathcal{R}_{\mathsf{sim}}$ hold:*

(1) $\mathsf{g}_i(\mathsf{s}(x), x_1, \ldots, x_a) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{g}_i(x, x_1, \ldots, x_a)$.
(2) $\mathsf{g}_i(x, x_1, \ldots, x_a) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{c}$
(3) *For all ground terms s such that $t \approx \mathsf{tr}(s)$, we have $\mathsf{size}(t) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{s}^l(0)$ where $l \geqslant |s|$.*
(4) *If $s \to^+_{\mathcal{R}} t$ and $\mathsf{tr}(s) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{tr}(t)$ then for any n-ary function symbol f, we have that $\mathsf{tr}(f(u_1, \ldots, s, \ldots, u_n)) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{tr}(f(u_1, \ldots, t, \ldots, u_n))$.*

Proof. The first two assertions are obvious. We show the third part by induction on $|s|$. As s is a ground term, $s = f(s_1, \ldots, s_n)$. Without loss of generality we set $t = \mathsf{g}_0(0, t_1, \ldots, t_n, \overline{\mathsf{c}})$, where $s_j \approx \mathsf{tr}(t_j)$ for all $1 \leqslant j \leqslant n$. If $|s| = 1$, then $n = 0$. Thus $\mathsf{size}(t) \to \mathsf{size}(\mathsf{c}) \to \mathsf{s}(0)$ by applying rules $8_{0,1}$ and 4. Otherwise, suppose $|s| > 1$. Then let j be such that $|s_j|$ is maximal. By induction hypothesis, we have $\mathsf{size}(t_j) \to^+ \mathsf{s}^{l_j}(0)$ with $l_j \geqslant |s_j|$. Hence, by

applying rules $3_{0,j}$, 5, and 6, we obtain

$$\mathsf{size}(t) \to \mathsf{d}_a(\mathsf{size}(t_j)) \to^* \mathsf{d}_a(\mathsf{s}^{l_j}(0)) \to^* \mathsf{s}^{a \cdot l_j}(0) \ .$$

Due to $a \cdot l_j \geqslant |s|$, property (3) follows.

Finally, we show property (4). We set $u := f(u_1, \ldots, t, \ldots, u_n)$ and we suppose that $\mathsf{dh}^\sharp(f(u_1, \ldots, s, \ldots, u_n)) = (i, m)$ and $\mathsf{dh}^\sharp(f(u_1, \ldots, t, \ldots, u_n)) = (i', m')$. Then we set

$$\begin{aligned} s' &:= \mathsf{tr}(f(u_1, \ldots, s, \ldots, u_n)) = \mathsf{g}_i(m, \mathsf{tr}(u_1), \ldots, \mathsf{tr}(s), \ldots, \mathsf{tr}(u_n)) \\ t' &:= \mathsf{tr}(f(u_1, \ldots, t, \ldots, u_n)) = \mathsf{g}_{i'}(m', \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) \ . \end{aligned}$$

By assumption, $f(u_1, \ldots, s, \ldots, u_n)$ rewrites to $f(u_1, \ldots, t, \ldots, u_n)$ by some derivation A. Moreover $\epsilon \in \epsilon \setminus\!\setminus A$ and thus by Lemma 8.6 $(i, m) \geqslant^{\mathrm{lex}} (i', m')$.

If $i = i'$, then $m \geqslant m'$, and we have

$$\begin{aligned} s' &\to^+ \mathsf{g}_i(m, \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) \\ &\to^* \mathsf{g}_i(m', \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) = t' \ . \end{aligned}$$

Here we apply the assumption $\mathsf{tr}(s) \to^+ \mathsf{tr}(t)$ in the first line and property (1) in the second. Otherwise, $i > i'$ and we obtain the following derivation:

$$\begin{aligned} s' &\to^+ \mathsf{g}_i(m, \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) \\ &\to^* \mathsf{g}_{i'}(\mathsf{f}(\mathsf{size}(t')), \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) \\ &\to^* \mathsf{g}_{i'}(m, \mathsf{tr}(u_1), \ldots, \mathsf{tr}(t), \ldots, \mathsf{tr}(u_n)) = t' \ . \end{aligned}$$

Here the second line follows by applying rules 2_i to $2_{i'+1}$ such that $t' \approx \mathsf{tr}(u)$. In the third line, we firstly make use of property (3) to conclude $\mathsf{size}(t') \to^+ \mathsf{s}^l(0)$ for some $l \geqslant |u|$. Secondly Definition 8.4 yields $f(l) \geqslant m'$. Thus by (8.2) we have $\mathsf{f}(\mathsf{s}^l(0)) \to^* \mathsf{s}^{f(l)}(0)$. Finally property (1) is applied. This completes the proof of the lemma. □

We arrive at the main lemma of this section.

Lemma 8.11. *For any ground terms s and t, $s \to_{\mathcal{R}} t$ implies $\mathsf{tr}(s) \to^+_{\mathcal{R}_{\mathsf{sim}}} \mathsf{tr}(t)$.*

Proof. Let $l \to r$ be the rewrite rule applied in the step $s \to t$. Then there exist some position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ and some substitution σ such that $l\sigma = s|_p$ and $r\sigma = t|_p$. It is not difficult to see that there exists a position $q \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\mathsf{tr}(s))$ such that $\mathsf{tr}(l\sigma) = \mathsf{tr}(s)|_q$ and $\mathsf{tr}(r\sigma) = \mathsf{tr}(t)|_q$.

First, we show $\mathsf{tr}(l\sigma) \to^+ \mathsf{tr}(r\sigma)$. Let $\mathsf{dh}^\sharp(l\sigma) = (i, m)$. Since l is not a variable, we have $l = f(l_1, \ldots, l_n)$. Hence, $\mathsf{tr}(l\sigma) = \mathsf{g}_i(\mathsf{s}^m(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}})$. Since $\mathsf{rt}(l)$ is defined, we have $m > 0$. By rules 1_i, $8_{i,1}$ and 12_i, we have

$$\mathsf{tr}(l\sigma) \to^+ \mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(r)}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \ .$$

We show the following claim by induction on $\mathsf{dp}(u)$.

Claim 8.12. *If $u \trianglelefteq r$, then $\mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(u)}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \to^* \mathsf{tr}(u\sigma)$, where $\mathsf{dh}^\sharp(l\sigma) = (i, m)$.*

Since $r \trianglelefteq r$ and $\mathsf{dp}(r) \leqslant C$, the claim entails $\mathsf{tr}(l\sigma) \to^+ \mathsf{tr}(r\sigma)$. Applying Lemma 8.10(4) then yields $\mathsf{tr}(s) \to^+ \mathsf{tr}(t)$ and the lemma follows. Hence, the remainder of this proof is devoted to showing the claim.

In proof of the claim, it suffices to consider the interesting case that $u \not\lhd l$. Since $\mathcal{V}\mathsf{ar}(l) \supseteq \mathcal{V}\mathsf{ar}(r) \supseteq \mathcal{V}\mathsf{ar}(u)$, u is not a variable. Hence, $u = g(u_1, \ldots, u_{n'})$. Let $\mathsf{dh}^\sharp(u) = (i', m')$. By induction hypothesis, for all $1 \leqslant j \leqslant n'$ we have

$$\mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(u_j)}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \to^* \mathsf{tr}(u_j\sigma) \; . \tag{8.3}$$

Moreover, employing instances of the rules $8_{i,1}$ and 12_i, we obtain:

$$\begin{aligned}\mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(u)-1}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \\ \to^* \mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(u_j)}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \; .\end{aligned} \tag{8.4}$$

From rule 12_i together with (8.4) and (8.3), we obtain

$$\mathsf{tree}_i(\mathsf{s}^{\mathsf{dp}(u)}(0), \mathsf{s}^{m-1}(0), \mathsf{tr}(l_1\sigma), \ldots, \mathsf{tr}(l_n\sigma), \overline{\mathsf{c}}) \to^* \mathsf{g}_i(\mathsf{s}^{m-1}(0), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}}) \; ,$$

employing Lemma 8.10(2). We distinguish two subcases for i': either $i = i'$, or $i > i'$. (Note that $i' > i$ is impossible since $(i, m) >^{\mathrm{lex}} (i', m')$ due to Lemma 8.6.) Suppose $i' = i$, then $m > m'$ due to Lemma 8.6. We obtain

$$\mathsf{g}_i(\mathsf{s}^{m-1}(0), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}}) \to^* \mathsf{tr}(u\sigma) \; .$$

Here we use Lemma 8.10(1). Otherwise, if $i > i'$, from $\mathsf{g}_i(\mathsf{s}^{m-1}(0), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}})$, we reach the term $\mathsf{g}_{i'}(\mathsf{f}(\mathsf{size}(u')), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}})$ for a suitable $u' \approx \mathsf{tr}(u\sigma)$, applying rules 2_i to $2_{i'+1}$. Thus by Lemma 8.10(1,3) and (8.2), we obtain the following derivation:

$$\mathsf{g}_{i'}(\mathsf{f}(\mathsf{size}(u')), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}}) \to^* \mathsf{g}_{i'}(\mathsf{s}^{m'}(0), \mathsf{tr}(u_1\sigma), \ldots, \mathsf{tr}(u_{n'}\sigma), \overline{\mathsf{c}}) = \mathsf{tr}(u\sigma) \; .$$

This concludes the proof of the claim, and thus of the lemma. □

Lemma 8.11 yields that the length of any derivation in $\mathcal{R}$ can be estimated by the maximal derivation height with respect to $\mathcal{R}_{\mathsf{sim}}$. To extend this to measure the derivational complexity function $\mathsf{dc}_{\mathcal{R}}$ via the function $\mathsf{dc}_{\mathcal{R}_{\mathsf{sim}}}$ we make use of the following lemma; note that $|\mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z})| \leqslant |t|$.

Lemma 8.13. *For any ground term t, we have $\mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z}) \to^*_{\mathcal{R}_{\mathsf{sim}}} \mathsf{tr}(t)$.*

Proof. We show the slightly more general assertion that if $l \geqslant \mathsf{dp}(t)$, then $\mathsf{g}^l(\mathsf{z}) \to^* \mathsf{tr}(t)$. We proceed by induction on l. Since t is ground, $t = f(t_1, \ldots, t_n)$. Let $\mathsf{dh}^\sharp(t) = (i, m)$. We distinguish two cases: either $l = 0$ or $l > 0$. Assume $l = 0$, then $\mathsf{dp}(t) = 0$, hence t is a constant. We obtain the derivation

$$\mathsf{z} \to \mathsf{g}_k(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \overline{\mathsf{c}}))), \overline{\mathsf{c}}) \to^* \mathsf{g}_i(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \overline{\mathsf{c}}))), \overline{\mathsf{c}}) \to^* \mathsf{g}_i(\mathsf{f}(\mathsf{s}^{l'}(0)), \overline{\mathsf{c}}) \to^* \mathsf{g}_i(\mathsf{s}^{f(|t|)}(0), \overline{\mathsf{c}}) \; ,$$

where $l' \geqslant |t|$. Here we apply the rules 10 and 2_k to 2_{i+1} in conjunction with Lemma 8.10(3) for the derivation $\mathsf{z} \to^* \mathsf{g}_i(\mathsf{f}(\mathsf{s}^{l'}(0)), \overline{\mathsf{c}})$. Note that $\mathsf{g}_0(0, \overline{\mathsf{c}}) \approx \mathsf{tr}(t)$ holds. Furthermore $\mathsf{g}_i(\mathsf{f}(\mathsf{s}^{l'}(0)), \overline{\mathsf{c}}) \to^* \mathsf{g}_i(\mathsf{s}^{f(|t|)}(0), \overline{\mathsf{c}})$ is due to Lemma 8.10(1) together with (8.2).

On the other hand assume $l > 0$. It is easy to see that $\mathsf{g}^l(\mathsf{z}) \to^* \mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z})$. Furthermore by an application of rule 9, we obtain

$$\mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z}) \to \mathsf{g}_k(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \mathsf{g}^{\mathsf{dp}(t)-1}(\mathsf{z}), \ldots, \mathsf{g}^{\mathsf{dp}(t)-1}(\mathsf{z})))), \mathsf{g}^{\mathsf{dp}(t)-1}(\mathsf{z}), \ldots, \mathsf{g}^{\mathsf{dp}(t)-1}(\mathsf{z})) \; .$$

Note that $\mathsf{g}(x) \to \mathsf{g}_k(\mathsf{f}(\ldots), x, \ldots, x) \to^* \mathsf{c}$ by Lemma 8.10(2), $\mathsf{z} \to^* \mathsf{c}$, and for all j, $\mathsf{g}^{\mathsf{dp}(t)-1}(\mathsf{z}) \to^* \mathsf{tr}(t_j)$ by induction hypothesis. Thus the right-hand side of the above equation rewrites to

$$\mathsf{g}_k(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(0, \mathsf{tr}(t_1), \ldots, \mathsf{tr}(t_n), \overline{\mathsf{c}}))), \mathsf{tr}(t_1), \ldots, \mathsf{tr}(t_n), \overline{\mathsf{c}}) \; ,$$

which in turn rewrites to $\mathsf{g}_i(\mathsf{f}(\mathsf{size}(s)), \mathsf{tr}(t_1), \ldots, \mathsf{tr}(t_n), \overline{\mathsf{c}})$ for suitable s with $s \approx \mathsf{tr}(t)$. Finally, Lemma 8.10(1),(3) and (8.2) yield $\mathsf{g}_i(\mathsf{s}^m(\mathsf{0}), \mathsf{tr}(t_1), \ldots, \mathsf{tr}(t_n), \overline{\mathsf{c}})$. This concludes the proof. □

It remains to verify that $\mathcal{R}_{\mathsf{sim}}$ is terminating and that $\mathsf{dc}_{\mathcal{R}_{\mathsf{sim}}}$ is primitive recursive in f. This is non-trivial, due to the rules 1_i.

Theorem 8.14. *There exists a well-founded monotone algebra $\mathcal{I}$ such that $\mathcal{I}$ is compatible with $\mathcal{R}_{\mathsf{sim}}$ and for all $g \in \mathcal{F}_{\mathsf{sim}}$, the function $g_{\mathcal{I}}$ is primitive recursive in f. In particular $\mathcal{R}_{\mathsf{sim}}$ is terminating and $\mathsf{dc}_{\mathcal{R}_{\mathsf{sim}}}$ is primitive recursive in f.*

Proof. The proof is given in the appendix. □

We arrive at the main result of this section.

Theorem 8.15. *Let $\mathcal{R}$ be a terminating TRS and let f be the following function over $\mathbb{N}$:*

$$f(n) := \max(\{1\} \cup \{\mathsf{dh}(t^\sharp, \to_{\mathcal{P}/\mathcal{R}}) \mid |t| \leqslant n, \mathcal{P} \text{ is SCC of } \mathsf{DG}(\mathcal{R})\}) \ .$$

Then $\mathsf{dc}_{\mathcal{R}}$ is primitive recursive in f. This upper bound is essentially optimal if f is at least linear.

Proof. Let t be a term. Without loss of generality we can assume that t is ground. Due to Lemmata 8.11 and 8.13 we have the following inequalities.

$$\mathsf{dh}(t, \to_{\mathcal{R}}) \leqslant \mathsf{dh}(\mathsf{tr}(t), \to_{\mathcal{R}_{\mathsf{sim}}}) \leqslant \mathsf{dh}(\mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z}), \to_{\mathcal{R}_{\mathsf{sim}}}) \ .$$

Note that $|\mathsf{g}^{\mathsf{dp}(t)}(\mathsf{z})| \leqslant |t|$. Hence for all n: $\mathsf{dc}_{\mathcal{R}}(n) \leqslant \mathsf{dc}_{\mathcal{R}_{\mathsf{sim}}}(n)$. Due to Theorem 8.14, $\mathsf{dc}_{\mathcal{R}_{\mathsf{sim}}}$ is primitive recursive in f. Thus $\mathsf{dc}_{\mathcal{R}}$ is bounded by a function primitive recursive in f. It follows from Example 8.1 that this bound is essentially optimal. □

Consider any TRS $\mathcal{R}$ whose termination can be shown by the basic dependency pair method in conjunction with dependency graphs and some base technique for each SCC of the dependency graph. Let f be defined as in Theorem 8.15. As an example, if only polynomial interpretations and MPO are used as base techniques, then both f and $\mathsf{dc}_{\mathcal{R}}$ are bounded by primitive recursive functions. Note that the derivational complexity induced by MPO (as a direct method) is primitive recursive [18]. By definition the primitive recursive functions are closed under primitive recursion. Hence the complexity of the dependency pair method (in conjunction with the dependency graph refinement) becomes negligible.

9. Conclusion

In this paper we have investigated the derivational complexity induced by the dependency pair method, where the object of our investigation is the standard formulation of the dependency pair method [1, 14] together with natural refinements.

We have established the following results: Firstly, for the basic dependency pair method (potentially using argument filterings) the induced derivational complexity is triple exponentially bounded in the derivational complexity of the base technique used. For string rewrite systems we have an optimal exponential upper bound and for the general case, we presented a double exponential lower bound. Secondly, if we consider the dependency pair method using the usable rules refinement, then the induced derivational complexity is primitive recursive in the derivational complexity of the base technique. Here we have provided a nonelementary lower bound. Finally, if we consider the dependency pair method

in conjunction with dependency graphs, then the induced derivational complexity is again primitive recursive in the derivational complexity of the base techniques employed. This result is essentially optimal. It is worthy of note that this is the very first analysis of the dependency pair method (without any dilutions) from a complexity analysis point of view. It remains to clarify to what extent such results hold for other notions of complexity.

As briefly mentioned in the introduction the *derivational complexity* is not the only measure of the complexity of a TRS suggested in the literature. In particular, alternative approaches have been suggested by Choppy et al. [7], Cichon and Lescanne [8], and Hirokawa and the first author [16]. In [16] the *runtime complexity* with respect to a TRS is defined as a refinement of the derivational complexity, by restricting the set of admitted initial terms. This notion has first been suggested in [7], where it is augmented by an *average case* analysis. Finally [8] studies the complexity of the functions *computed* by a given TRS. This latter notion is often studied within *implicit computational complexity theory* (see [5] for an overview).

We have chosen to present our results in terms of derivational complexity as this simplifies the comparison to well-known results in this area. However, it is easy to see that all upper bound results hold as well, if we would study the runtime complexity of a TRS. Furthermore, the runtime complexity of a TRS is an invariant cost model [10] and thus it is straightforward to rephrase our results in terms of the complexity of the function computed by the TRS in question. Let f be a function computable by a TRS $\mathcal{R}$ and let g denote a function that grows at least linearly. Suppose the runtime complexity of $\mathcal{R}$ is bounded by $g(n)$. Then there exists a Turing machine running in time polynomial in $g(n)$ that computes f [3]. Thus our results also characterises the complexity of functions computed by rewrite systems, whose termination has been shown by the dependency pair method together with natural refinements.

From the original viewpoint of derivational complexity analysis, as an analysis of the strength of termination methods, the implications of our results are easy to state. For example, our results imply that the (technically simple) extensions of the dependency pair method with the dependency graph refinement greatly increase the strength of the method. On the other hand our results also provide limitations on the strength of the studied techniques. For instance consider the following example.

Example 9.1. Consider the TRS $\mathcal{R}_9$ introduced by Touzet in [36].[2]

$$\mathsf{b}(\mathsf{u}(x)) \to \mathsf{b}(\mathsf{s}(x)) \qquad \mathsf{s}(\mathsf{b}(\mathsf{s}(x))) \to \mathsf{b}(\mathsf{t}(x)) \qquad \mathsf{t}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{s}(x)) \qquad \mathsf{t}(\mathsf{s}(x)) \to \mathsf{t}(\mathsf{t}(x))$$
$$\mathsf{s}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))) \quad \mathsf{s}(\mathsf{u}(x)) \to \mathsf{s}(\mathsf{s}(x)) \quad \mathsf{t}(\mathsf{b}(\mathsf{s}(x))) \to \mathsf{u}(\mathsf{t}(\mathsf{b}(x))) \quad \mathsf{t}(\mathsf{u}(x)) \to \mathsf{u}(\mathsf{t}(x))$$

$\mathcal{R}_9$ encodes the Ackermann function [36] and therefore the derivational complexity function belongs to $\mathrm{Ack}(\Theta(n), 0)$.

Our results imply that any successful termination proof of $\mathcal{R}_9$ has to employ techniques that go beyond the basic dependency pair method and the refinements studied here. Very recently, Sternagel and Middeldorp presented in [33] an automatic termination proof of $\mathcal{R}_9$. Based on our work it is indeed no surprise that this proof makes crucial use of an extension of the dependency pair method, the dependency pair *framework* [12, 35].

Motivated by this and like-minded examples we have very recently started investigations into the complexity induced by the dependency pair framework. A first result in this

[2]This is example Zantema_04/z090 in the termination problems database, see `http://termcomp.uibk.ac.at/`.

direction shows that the complexity of the dependency pair framework may be multiply recursive [28]. Furthermore, for a clearly defined subset of processors, this bound is optimal.

In recent years (derivational) complexity results mainly focused on crafting new methods that induce low-complexity upper bounds, like for example polynomial upper bounds. We exemplarily mention results by Neurauter et al. studying the use of matrix interpretations to polynomially bound the derivational complexity of TRSs [31]. Moreover, in the area of implicit computational complexity, Bonfante et al. study the use of quasi-interpretations to characterise complexity classes like LINSPACE, PTIME, or PSPACE [6]. In the context of our results these classes are clearly of a low complexity.

With respect to this motivation our results are arguably negative: our results clearly show that the undiluted dependency pair method is not a suitable tool to yield low complexity upper bound. Again it does not matter much whether we consider derivational complexity or runtime complexity: the example given for the double exponential lower bound for the basic dependency pair method also shows a double exponential lower bound for the runtime complexity.

Recently a number of variants of the dependency pair method have been proposed in the literature [2, 16, 17, 23, 24, 32, 39]. We believe that our results can also be profitably employed in the crafting of variants of the dependency pair method or framework in the context of *polynomial* complexity analysis. This will be subject to future work.

Acknowledgement

We thank the reviewers for constructive suggestions that helped to improve the quality of the presentation of the paper.

References

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1,2):133–178, 2000.

[2] Martin Avanzini and Georg Moser. Dependency pairs and polynomial path orders. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *LNCS*, pages 48–62, 2009.

[3] Martin Avanzini and Georg Moser. Closing the gap between runtime complexity and polytime computability. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, volume 6 of *LIPIcs*, pages 33–48, 2010.

[4] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[5] Patric Baillot, Jean-Yves Marion, and Simona Ronchi Della Rocca. Guest editorial: Special issue on implicit computational complexity. *ACM Trans. Comput. Log.*, 10(4), 2009.

[6] Guillaume Bonfante, Jean-Yves Marion, and Jean-Yves Moyen. Quasi-interpretations: A way to control resources. *Theoretical Computer Science*, 412(25):2776–2796, 2011.

[7] Christine Choppy, Stéphane Kaplan, and Michèle Soria. Complexity analysis of term-rewriting systems. *Theoretical Computer Science*, 67(2–3):261–282, 1989.

[8] Adam Cichon and Pierre Lescanne. Polynomial interpretations and the complexity of algorithms. In *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *LNCS*, pages 139–147, 1992.

[9] Adam Cichon and Andreas Weiermann. Term rewriting theory for the primitive recursive functions. *Annals of Pure and Applied Logic*, 83(3):199–223, 1997.
[10] Ugo Dal Lago and Simone Martini. On constructor rewrite systems and the lambda-calculus. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, volume 5556 of *LNCS*, pages 163–174, 2009.
[11] Alfons Geser. *Relative Termination*. PhD thesis, Universität Passau, 1990.
[12] Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
[13] William G. Handley and Stanley S. Wainer. Equational derivation vs. computation. *Annals of Pure and Applied Logic*, 70(1):17–49, 1994.
[14] Nao Hirokawa and Aart Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.
[15] Nao Hirokawa and Aart Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474–511, 2007.
[16] Nao Hirokawa and Georg Moser. Automated complexity analysis based on the dependency pair method. In *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 364–379, 2008.
[17] Nao Hirokawa and Georg Moser. Complexity, graphs, and the dependency pair method. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 5330 of *LNCS*, pages 652–666, 2008.
[18] Dieter Hofbauer. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140, 1992.
[19] Dieter Hofbauer. *Termination Proofs and Derivation Lengths in Term Rewriting Systems*. PhD thesis, Technische Universität Berlin, 1992.
[20] Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications*, volume 355 of *LNCS*, pages 167–177, 1989.
[21] Dallas Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
[22] Ingo Lepper. Derivation lengths and order types of Knuth-Bendix orders. *Theoretical Computer Science*, 269(1,2):433–450, 2001.
[23] Jean-Yves Marion and Romain Péchoux. Characterizations of polynomial complexity classes with a better intensionality. In *Proceedings of the 10th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 79–88. ACM, 2008.
[24] Jean-Yves Marion and Romain Péchoux. Sup-interpretations, a semantic method for static analysis of program resources. *ACM Transactions on Computational Logic*, 10 (4), 2009.
[25] Georg Moser. Derivational complexity of Knuth Bendix orders revisited. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 4246 of *LNCS*, pages 75–89, 2006.
[26] Georg Moser and Andreas Schnabl. Proving quadratic derivational complexities using context dependent interpretations. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *LNCS*, pages 276–290, 2008.
[27] Georg Moser and Andreas Schnabl. The derivational complexity induced by the dependency pair method. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *LNCS*, pages 255–269, 2009.

[28] Georg Moser and Andreas Schnabl. Termination proofs in the dependency pair framework may induce multiply recursive derivational complexity. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications*, volume 10 of *LIPIcs*, pages 235–250, 2011.

[29] Georg Moser and Andreas Weiermann. Relating derivation lengths with the slow-growing hierarchy directly. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, volume 2706 of *LNCS*, pages 296–310, 2003.

[30] Georg Moser, Andreas Schnabl, and Johannes Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In *Proceedings of the 28th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2 of *LIPIcs*, pages 304–315, 2008.

[31] Friedrich Neurauter, Harald Zankl, and Aart Middeldorp. Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 6397 of *LNCS (ARCoSS)*, pages 550–564, 2010.

[32] Lars Noschinski, Fabian Emmes, and Jürgen Giesl. The dependency pair framework for automated complexity analysis of term rewrite systems. In *Proceedings of the 23rd International Conference on Automated Deduction*, LNCS, 2011. To appear.

[33] Christian Sternagel and Aart Middeldorp. Root-labeling. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *LNCS*, pages 336–350, 2008.

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

[35] René Thiemann. *The DP Framework for Proving Termination of Term Rewriting*. PhD thesis, University of Aachen, 2007.

[36] Hélène Touzet. A complex example of a simplifying rewrite system. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, volume 1443 of *LNCS*, pages 507–517, 1998.

[37] Johannes Waldmann. Polynomially bounded matrix interpretations. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, volume 6 of *LIPIcs*, pages 357–372, 2010.

[38] Andreas Weiermann. Termination proofs for term rewriting systems with lexicographic path orderings imply multiply recursive derivation lengths. *Theoretical Computer Science*, 139(1,2):355–362, 1995.

[39] Harald Zankl and Martin Korp. Modular complexity analysis via relative complexity. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, volume 6 of *LIPIcs*, pages 385–400, 2010.

[40] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1,2):89–105, 1995.

Appendix A. Termination of the Simulating TRS $\mathcal{R}_{\mathsf{sim}}$

Recall the definition of the simulating TRS $\mathcal{R}_{\mathsf{sim}}$ given in Section 8. In this appendix we define a well-founded monotone algebra $\mathcal{I} = (\mathbb{N}, >)$, where $>$ denotes the usual order on the natural numbers. Termination of $\mathcal{R}_{\mathsf{sim}}$ follows as $\mathcal{I}$ is compatible with $\mathcal{R}_{\mathsf{sim}}$. Furthermore, if the function f, defined in (8.1), is primitive recursive, then $\mathcal{I}$ makes only use of primitive recursive interpretation functions. The definition of $\mathcal{I}$ makes use of a family of fast growing functions, defined below. This definition is parametrised in d. The exact value of the parameter d will become clear from the termination proof. To simplify the notation we assume the function f is primitive recursive. Otherwise Definition A.1 has to be replaced by a function hierarchy that is parametrised in f.

Definition A.1. Let $d \geqslant 2$ be a given number. We define:

$$\mathsf{F}_0(m) := d^{m+1} \qquad \mathsf{F}_{n+1}(m) := \mathsf{F}_n^{m+1}(m) \; .$$

The following properties of the family of functions $\{\mathsf{F}_n \mid n \geqslant 0\}$ are easy to verify.

Lemma A.2. *Let n, m, a, and b be natural numbers.*

(1) $\mathsf{F}_n(a) \geqslant d^{a+1} \geqslant d \cdot a > a$.
(2) *If $a > b$, then $\mathsf{F}_n(a) > \mathsf{F}_n(b)$.*
(3) *If $n > m$, then $\mathsf{F}_n(a) > \mathsf{F}_m(a)$ for $a \geqslant 1$.*
(4) $\mathsf{F}_m(a + b) \geqslant \mathsf{F}_m(a) + b$ *and* $\mathsf{F}_m(a + 1) \geqslant 2 \cdot \mathsf{F}_m(a)$.
(5) *Each function F_n is primitive recursive.*
(6) *For every n-ary primitive recursive function g, there exists a number k such that for all numbers $m_1, \ldots, m_n$: $\mathsf{F}_k(\max\{m_1, \ldots, m_n\}) \geqslant g(m_1, \ldots, m_n)$.* □

The next proposition is due to Hofbauer [19].

Proposition A.3. *Let $\mathcal{A} = (\mathbb{N}, >)$ denote a weakly monotone algebra, compatible with a TRS $\mathcal{R}$ and let p be a strictly monotone unary function on $\mathbb{N}$ such that for all $f \in \mathcal{F}$*

$$p(n) \geqslant f_{\mathcal{A}}(n, \ldots, n) \qquad \text{for all } n \in \mathbb{N} \; .$$

Then we have $\mathsf{dc}_{\mathcal{R}}(n) \leqslant p^n(0)$. □

Recall that $\mathcal{F}_{\mathsf{sim}}$ denotes the signature of the TRS $\mathcal{R}_{\mathsf{sim}}$. By definition $\mathsf{f} \in \mathcal{F}_{\mathsf{sim}}$ and we assume that the function f is primitive recursive. The rules $\mathcal{R}'$ defining f constitute a (terminating) subset of $\mathcal{R}_{\mathsf{sim}}$, c.f. Definition 8.7. For the definition of the well-founded monotone algebra $\mathcal{I}$ it suffices to define primitive recursive mappings $f_{\mathcal{I}}$ for all $f \in \mathcal{F}_{\mathsf{sim}}$. A complication is the definition of $\mathsf{f}_{\mathcal{I}}$ as the TRS $\mathcal{R}'$ has only been defined implicitly above. However, following the construction in [18], we conclude the existence of a well-founded monotone algebra $\mathcal{J}$ compatible with $\mathcal{R}'$ such that $\mathsf{f}_{\mathcal{J}}$ is primitive recursive. More precisely, without loss of generality we can assume that there exists $\ell \in \mathbb{N}$ such that $\mathsf{f}_{\mathcal{J}}(n) = \mathsf{F}_\ell(n)$ and that $\mathsf{s}_{\mathcal{J}}(n) = n + 1$ and $\mathsf{0}_{\mathcal{J}} = 1$.

Preparing the definition of the well-founded monotone algebra $\mathcal{I}$, we define the interpretation functions $\mathsf{f}_{\mathcal{I}}$, $\mathsf{s}_{\mathcal{I}}$, and $\mathsf{0}_{\mathcal{I}}$ as follows:

$$\mathsf{f}_{\mathcal{I}}(n) := \mathsf{F}_\ell(n) \qquad \mathsf{s}_{\mathcal{I}}(n) = n + 1 \qquad \mathsf{0}_{\mathcal{I}} = 1 \; . \tag{A.1}$$

The next definition gives the mappings associated to the function symbols g_i $(0 \leqslant i \leqslant k)$. Let $d \geqslant \max\{C + 2, a + 1\}$.

$$(\mathsf{g}_i)_{\mathcal{I}}(n, x_1, \ldots, x_a) := \mathsf{F}_{\ell+2i}^{d^{n+1}\cdot(C+1)}(n + x_1 + \cdots + x_a) \; . \tag{A.2}$$

Before we continue the definition of $\mathcal{I}$ we give the following auxiliary result. Let α denote an arbitrary assignment. Let x be a variable and let $\overline{x}$ abbreviate $[\alpha]_{\mathcal{A}}(x)$.

Lemma A.4. *Let α be an assignment such that for all $x \in \mathcal{V}$, $\alpha(x) \geqslant 1$. Then there exists $q \in \mathbb{N}$ such that $\mathsf{F}_q(\overline{n} + \overline{x}_1 + \cdots + \overline{x}_a) \geqslant [\alpha]_{\mathcal{A}}(\mathsf{g}_i(n, x_1, \ldots, x_a))$.*

Proof. By definition $[\alpha]_{\mathcal{A}}(\mathsf{g}_i(n, x_1, \ldots, x_a)) = \mathsf{F}_{\ell+2i}^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n}+\overline{x}_1+\cdots+\overline{x}_a)$, we set $p := \ell+2i$ and abbreviate $\overline{n} + \overline{x}_1 + \cdots + \overline{x}_a$ as $\overline{n} + \overline{x}$. Due to Lemma A.2(1) and the assumption on d, we have $\mathsf{F}_1(\overline{n} + \overline{x}) \geqslant d^{\overline{n}+2+\overline{x}} \geqslant d^{\overline{n}+2} + \overline{x} > d^{\overline{n}+1} \cdot (C+1) + \overline{x}$ for $\overline{n} \geqslant 1$. In sum, we obtain:

$$\begin{aligned}
\mathsf{F}_{p+2}(\overline{n} + \overline{x}) &\geqslant \mathsf{F}_{p+1} \circ \mathsf{F}_{p+1}(\overline{n} + \overline{x}) \\
&\geqslant \mathsf{F}_{p+1} \circ \mathsf{F}_1(\overline{n} + \overline{x}) \\
&\geqslant \mathsf{F}_{p+1}(d^{\overline{n}+1} \cdot (C+1) + \overline{x}) \\
&\geqslant \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)+1}(d^{\overline{n}+1} \cdot (C+1) + \overline{x}) \\
&> \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n} + \overline{x}) \; .
\end{aligned}$$

Hence the lemma follows, if we set $q := p + 2$. □

The next definition gives the mappings associated to the function symbols tree_i ($0 \leqslant i \leqslant k$).

$$(\mathsf{tree}_i)_{\mathcal{I}}(m, n, x_1, \ldots, x_a) := \mathsf{F}_{\ell+2i}^{d^{n+2}\cdot(m+1)}(x + x_1 + \cdots + x_a) \; . \tag{A.3}$$

The interpretation functions given in (A.1)–(A.3) are sufficient to prove the main result of this appendix.

Theorem 8.14 (revisited). *There exists a well-founded monotone algebra $\mathcal{I}$, such that $\mathcal{I}$ is compatible with $\mathcal{R}_{\mathsf{sim}}$ and for all $g \in \mathcal{F}_{\mathsf{sim}}$, the function $g_{\mathcal{I}}$ is primitive recursive in the parameter function f. In particular $\mathcal{R}_{\mathsf{sim}}$ is terminating.*

Proof. Without loss of generality we can assume that f is primitive recursive. Otherwise a straightforward extension of Definition A.1 suffices to prove the more general proposition.

Set $\mathcal{I} = (\mathbb{N} - \{0\}, >)$ and recall that in (A.1), (A.2), and (A.3) the mappings $\mathsf{f}_{\mathcal{I}}$, $\mathsf{s}_{\mathcal{I}}$, $\mathsf{0}_{\mathcal{I}}$, $(\mathsf{g}_i)_{\mathcal{I}}$ and $(\mathsf{tree}_i)_{\mathcal{I}}$ have been defined, where $0 \leqslant i \leqslant k$ and $0 \leqslant j \leqslant C$ holds. We extend these definitions, by setting $\mathsf{c}_{\mathcal{I}} := 3$ and $\mathsf{size}_{\mathcal{I}}(n) := n$. Hence it remains to consider the mappings $\mathsf{d}_{\mathcal{I}}$, $\mathsf{g}_{\mathcal{I}}$, and $\mathsf{z}_{\mathcal{I}}$. Based on Lemma A.4 it is not difficult to define suitable interpretations such that the rules $3_{i,j}$—10 are strictly decreasing with respect to $\mathcal{I}$. We leave these definitions to the reader.

We write $f \circ g(n)$ for the function composition $f(g(n))$ and we abbreviate $\overline{x}_1 + \cdots + \overline{x}_a$ as $\overline{x}$. In proving compatibility, we restrict our attention to the (families of) rules 1_i, 11_i, 12_i ($i \in \{0, \ldots, k\}$) and $2_{i'}$ ($i' \in \{1, \ldots, k\}$). Let i be arbitrary, but fixed.

Consider the rule 1_i:

$$\mathsf{g}_i(\mathsf{s}(n), x_1, \ldots, x_a) \to \mathsf{tree}_i(\mathsf{s}^C(0), x, x_1, \ldots, x_a) \; .$$

Due to Lemma A.2(1) we obtain (for an arbitrary assignment α):

$$\begin{aligned}
[\alpha]_{\mathcal{A}}(\mathsf{g}_i(\mathsf{s}(n), x_1, \ldots, x_a)) &= \mathsf{F}_{\ell+2i}^{d^{\overline{n}+2}\cdot(C+1)}(\overline{n} + 1 + \overline{x}) \\
&> \mathsf{F}_{\ell+2i}^{d^{\overline{n}+2}\cdot(C+1)}(\overline{n} + \overline{x}) \\
&= [\alpha]_{\mathcal{A}}(\mathsf{tree}_i(\mathsf{s}^C(0), n, x_1, \ldots, x_a)) \; .
\end{aligned}$$

Consider the rule 11_i:

$$\mathsf{tree}_i(0, x, x_1, \dots, x_a) \to \mathsf{g}_i(x, x_1, \dots, x_a) \ .$$

Then we obtain:

$$\begin{aligned}[\alpha]_{\mathcal{A}}(\mathsf{tree}_i(0, n, x_1, \dots, x_a)) &= \mathsf{F}_p^{d^{\overline{n}+2}}(\overline{n} + \overline{x}_1 + \cdots + \overline{x}_a) \\ &> \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n} + \overline{x}_1 + \cdots + \overline{x}_a) \\ &= [\alpha]_{\mathcal{A}}(\mathsf{g}_i(n, x_1, \dots, x_a)) \ ,\end{aligned}$$

where we use Lemma A.2(1) together with the fact $d > C + 1$.

Consider the rule 12_i:

$$\mathsf{tree}_i(\mathsf{s}(y), x, x_1, \dots, x_a) \to \mathsf{g}_i(x, \mathsf{tree}_i(y, x, x_1, \dots, x_a), \dots, \mathsf{tree}_i(y, x, x_1, \dots, x_a)) \ .$$

We obtain:

$$\begin{aligned}[\alpha]_{\mathcal{A}}(\mathsf{tree}_i(\mathsf{s}(m), n, x_1, \dots, x_a)) &= \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+2)}(\overline{n} + \overline{x}) \\ &> \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)+d^{\overline{n}+2}-d^{\overline{n}+1}+1}(\overline{n} + \overline{x}) \\ &= \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)+d^{\overline{n}+1}\cdot(d-1)+1}(\overline{n} + \overline{x}) \\ &\geqslant \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)+d^{\overline{n}+1}\cdot(C+1)+1}(\overline{n} + \overline{x}) \\ &= \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)} \circ \mathsf{F}_p \circ \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)}(\overline{n} + \overline{x}) \ .\end{aligned}$$

Due to (A.3) $\mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)}(\overline{n} + \overline{x}) = [\alpha]_{\mathcal{A}}(\mathsf{tree}_i(m, n, x_1, \dots, x_a))$. Thus due to Lemma A.2(1) and $d \geqslant a + 1$, we obtain: $\mathsf{F}_p \circ \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)}(\overline{n} + \overline{x}) \geqslant \overline{n} + a \cdot [\alpha]_{\mathcal{A}}(\mathsf{tree}_i(m, n, x_1, \dots, x_a))$. Moreover, due to (A.2):

$$\begin{aligned}&\mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n} + a \cdot [\alpha]_{\mathcal{A}}(\mathsf{tree}_i(m, n, x_1, \dots, x_a))) = \\ &[\alpha]_{\mathcal{A}}(\mathsf{g}_i(n, \mathsf{tree}_i(m, n, x_1, \dots, x_a), \dots, \mathsf{tree}_i(m, n, x_1, \dots, x_a))) \ .\end{aligned}$$

In sum we obtain

$$\begin{aligned}\mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)} \circ \mathsf{F}_p \circ \mathsf{F}_p^{d^{\overline{n}+2}\cdot(\overline{m}+1)}(\overline{n} + \overline{x}) &\geqslant \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n} + a \cdot [\alpha]_{\mathcal{A}}(\mathsf{tree}_i(m, n, x_1, \dots, x_a))) = \\ &= [\alpha]_{\mathcal{A}}(\mathsf{g}_i(n, \mathsf{tree}_i(m, n, x_1, \dots, x_a), \dots, \mathsf{tree}_i(m, n, x_1, \dots, x_a))) \ ,\end{aligned}$$

where we employ Lemma A.2(2).

Finally, consider the family of rules $(2_{i'})_{1 \leqslant i' \leqslant k}$ and let $i' \in \{1, \dots, k\}$ be arbitrary, but fixed. Consider the rule $2_{i'}$:

$$\mathsf{g}_{i'}(n, x_1, \dots, x_a) \to \mathsf{g}_{i'-1}(\mathsf{g}(\mathsf{size}(\mathsf{g}_0(\mathsf{0}, x_1, \dots, x_a))), x_1, \dots, x_a) \ .$$

Set $p := \ell + 2i'$, hence $p - 2 = \ell + 2(i' - 1)$. Recall that $d \geqslant C + 2 \geqslant 4$.

We obtain for any assignment α such that $\alpha(x) \geqslant 1$ for all $x \in \mathcal{V}$:

$$\begin{aligned}
[\alpha]_{\mathcal{A}}(\mathsf{g}_{i'}(n, x_1, \ldots, x_a)) &= \mathsf{F}_p^{d^{\overline{n}+1}\cdot(C+1)}(\overline{n} + \overline{x}) \\
&> \mathsf{F}_p \circ \mathsf{F}_p \circ \mathsf{F}_p^{d\cdot(C+1)}(\overline{n} + \overline{x}) \\
&> \mathsf{F}_p \circ \mathsf{F}_p \circ \mathsf{F}_{p-2}^{d^{\overline{n}+2}\cdot(C+1)}(\overline{n} + \overline{x}) \\
&\geqslant \mathsf{F}_p \circ \mathsf{F}_\ell \circ \mathsf{F}_{p-2}^{d^{\overline{n}+2}\cdot(C+1)}(1 + \overline{x}) \\
&\geqslant \mathsf{F}_{p-2}^{d^{(\mathsf{F}_\ell \circ \mathsf{F}_{p-2}^{d^2\cdot(C+1)}(1+\overline{x}))+1}\cdot(C+1)} \circ \mathsf{F}_\ell \circ \mathsf{F}_{p-2}^{d^{\overline{n}+2}\cdot(C+1)}(1 + \overline{x}) \\
&\geqslant \mathsf{F}_{p-2}^{d^{\mathsf{F}_\ell(\mathsf{F}_\ell^{d^2\cdot(C+1)}(1+\overline{x}))+1}\cdot(C+1)}(\mathsf{F}_\ell(\mathsf{F}_\ell^{d^2\cdot(C+1)}(1 + \overline{x})) + \overline{x}) \\
&= [\alpha]_{\mathcal{A}}(\mathsf{g}_{i'-1}(\mathsf{f}(\mathsf{size}(\mathsf{g}_0(\mathsf{0}, x_1, \ldots, x_a))), x_1, \ldots, x_a)) \,.
\end{aligned}$$

In lines 3 and 5 we apply slight variants of the proof of Lemma A.4, and in line 6 we apply Lemma A.2(4).

This completes the proof of compatibility for the crucial families of rules 1_i, 11_i, 12_i ($i \in \{0, \ldots, k\}$) and $2_{i'}$ ($i' \in \{1, \ldots, k\}$). Hence the theorem follows. □