Termination Proofs in the Dependency Pair Framework May Induce Multiple Recursive Derivational Complexity*

Georg Moser[1] and Andreas Schnabl[1]

1 Institute of Computer Science
University of Innsbruck, Austria
{georg.moser,andreas.schnabl}@uibk.ac.at

Abstract

We study the complexity of rewrite systems shown terminating via the dependency pair framework using processors for reduction pairs, dependency graphs, or the subterm criterion. The complexity of such systems is bounded by a multiple recursive function, provided the complexity induced by the employed base techniques is at most multiple recursive. Moreover this upper bound is tight.

1998 ACM Subject Classification F.2.2, F.4.1, D.2.4, D.2.8

Keywords and phrases Complexity, DP Framework, Multiple Recursive Functions

Digital Object Identifier 10.4230/LIPIcs.RTA.2011.235

Category Regular Research Paper

1 Introduction

Several notions to assess the complexity of a terminating term rewrite system (TRS) have been proposed in the literature, compare [3, 13, 4, 11]. The conceptually simplest one was suggested by Hofbauer and Lautemann in [13]: 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. However, we emphasise that our results immediately carry over to other complexity measures of TRSs (compare Section 5). Hence our results are conceivable as an investigation into *implicit computational complexity theory* (see [2] for an overview). To motivate our study consider the following example.

▶ **Example 1.1.** Let $\mathcal{R}_1$ be the TRS defined by the following rules:

$$\mathsf{Ack}(0,y) \to \mathsf{S}(y) \qquad \mathsf{Ack}(\mathsf{S}(x),\mathsf{S}(y)) \to \mathsf{Ack}(x,\mathsf{Ack}(\mathsf{S}(x),y))$$
$$\mathsf{Ack}(\mathsf{S}(x),0) \to \mathsf{Ack}(x,\mathsf{S}(0))$$

The $\mathcal{R}_1$ encodes the Ackermann function. Hence its derivational complexity function grows faster than any primitive recursive functions. Furthermore it is easy to see that the derivational complexity with respect to $\mathcal{R}_1$ is bounded by a multiple recursive function.

* This research is supported by FWF (Austrian Science Fund) project P20133 and a grant of the University of Innsbruck.

22nd International Conference on Rewriting Techniques and Applications (RTA'11).
Editor: M. Schmidt-Schauß; pp. 235–250
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We show termination of $\mathcal{R}_1$ by an application of the *dependency pair framework* (*DP framework* for short), compare [9, 23]. (All used notions will be defined in Section 2.) The set of dependency pairs $\mathsf{DP}(\mathcal{R}_1)$ with respect to $\mathcal{R}_1$ is given below:

$$\mathsf{Ack}^\sharp(\mathsf{S}(x),0) \to \mathsf{Ack}^\sharp(x,\mathsf{S}(0)) \qquad \mathsf{Ack}^\sharp(\mathsf{S}(x),\mathsf{S}(y)) \to \mathsf{Ack}^\sharp(x,\mathsf{Ack}(\mathsf{S}(x),y))$$
$$\mathsf{Ack}^\sharp(\mathsf{S}(x),\mathsf{S}(y)) \to \mathsf{Ack}^\sharp(\mathsf{S}(x),y)$$

These six rules together constitute the DP problem $(\mathsf{DP}(\mathcal{R}_1),\mathcal{R}_1)$. We apply the subterm criterion processor Φ^{SC} with respect to the simple projection π_1 that projects the first argument of the dependency pair symbol $\mathsf{Ack}^\sharp$. Thus $\Phi^{\mathsf{SC}}((\mathsf{DP}(\mathcal{R}_1),\mathcal{R}_1))$ consists of the single DP problem $(\mathcal{P},\mathcal{R}_1)$, where $\mathcal{P} = \{\mathsf{Ack}^\sharp(\mathsf{S}(x),\mathsf{S}(y)) \to \mathsf{Ack}^\sharp(\mathsf{S}(x),y))\}$. Another application of Φ^{SC}, this time with the simple projection π_2 projecting on the second argument of $\mathsf{Ack}^\sharp$ yields the DP problem $(\varnothing,\mathcal{R}_1)$, which is trivially finite. Thus termination of $\mathcal{R}_1$ follows.

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. For example [25] studies the derivational complexity induced by the lexicographic path order (LPO). LPO induces multiple recursive derivational complexity. In recent years the research focused on automatable proof methods that induce polynomially bounded (derivational) complexity (see for example [26, 18, 24]). The focus and the re-newed interest in this area was partly triggered by the integration of a dedicated complexity category in the annual international termination competition.[1].

None of these results can be applied to our motivating example. Abstracting from Example 1.1 suppose termination of a given TRS $\mathcal{R}$ can be shown via the DP framework in conjunction with a well-defined set of processors. Furthermore assume that all *base techniques* employed in the termination proof (employed within a processor) induce at most multiple recursive derivational complexity. Kindly note that this assumption is rather weak as for all termination techniques whose complexity has been analysed a multiple recursive upper bound exists. Then we show that the derivational complexity with respect to $\mathcal{R}$ is bounded by a multiply recursive function. We restrict our attention to simple DP processors like the *reduction pair processor*, the *dependency graph processor* or the *subterm criterion processor*. Furthermore we show that this upper bound is tight, even if we restrict to base techniques that induce linear derivational complexity. This result can be understood as a negative result: using the above mentioned DP processors, it is theoretically impossible to prove termination of any TRS whose (derivational) complexity is not bounded by a multiply recursive function. One famous example of such a TRS is Dershowitz's system `TRS/D33-33`, aka the Hydra battle rewrite system (see [6, 14]). On the other hand, our result immediately turns termination provers into automatic complexity provers, albeit rather weak ones. Furthermore it provides the basis for further investigations into termination techniques that induced more feasible (derivational) complexities.

The rest of this paper is organised as follows. In Section 2 we present basic notions and starting points of the paper. Section 3 states our main result and provides suitable examples to show that the multiple recursive bound presented is tight. The technical construction is given in Section 4. Finally, we conclude in Section 5. Due to space restriction, some technical proofs have been replaced by sketches. For the full proofs, we refer to the extended version of this paper [17].

[1] `http://termcomp.uibk.ac.at`

2 Preliminaries

We assume familiarity with term rewriting (see [22]) and in particular with the DP method and the DP framework (see [9, 10, 23]). Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a signature. Without loss of generality we assume that $\mathcal{F}$ contains at least one constant. The set of (ground) terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$ ($\mathcal{T}(\mathcal{F})$). The (proper) subterm relation is denoted as $\trianglelefteq$ ($\triangleleft$) and the subterm of t at position p is denoted as $t|_p$. Let t be a term. The *root symbol* of t is denoted as $\mathsf{rt}(t)$; the *positions* of t are denoted as $\mathsf{Pos}(t)$; the *size* (*depth*) of t is denoted as $|t|$ ($\mathsf{dp}(t)$). Let $\mathcal{R}$ and $\mathcal{S}$ be finite TRSs over $\mathcal{F}$. We write $\to_\mathcal{R}$ (or simply $\to$) for the induced rewrite relation. Let $s \xrightarrow{\epsilon}_\mathcal{R} t$ ($s \xrightarrow{\succ\epsilon}_\mathcal{R} t$) denote rewrite steps at (below) the root. We recall the notion of *relative rewriting* [8]. Let $\mathcal{R}$ and $\mathcal{S}$ be finite TRSs over $\mathcal{F}$. The *relative rewrite relation* $\to_{\mathcal{R}/\mathcal{S}}$ is defined as $\to_\mathcal{S}^* \cdot \to_\mathcal{R} \cdot \to_\mathcal{S}^*$. We use $\mathsf{NF}(\mathcal{R})$ to denote the set of normal-forms of $\mathcal{R}$, and $\mathsf{NF}(\mathcal{R}/\mathcal{S})$ for the set of normal forms of $\to_{\mathcal{R}/\mathcal{S}}$. The n-fold composition of $\to$ is denoted as $\to^n$ and 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: $\mathsf{dc}_\mathcal{R}(n) = \max\{\mathsf{dh}(t,\to_\mathcal{R}) \mid |t| \leqslant n\}$. Let $\mathcal{R}$ be a TRS and M a termination method. We say M *induces* a certain derivational complexity, if $\mathsf{dc}_\mathcal{R}$ is bounded by a function of this complexity, whenever termination of $\mathcal{R}$ follows by M.

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*. The set $\mathsf{DP}(\mathcal{R})$ of *dependency pairs* of $\mathcal{R}$ is defined as $\{l^\sharp \to u^\sharp \mid l \to r \in \mathcal{R}, u \trianglelefteq r$, but $u \not\trianglelefteq l$, and $\mathsf{rt}(u)$ is defined$\}$.[2] A *DP problem* is a pair $(\mathcal{P},\mathcal{R})$, where $\mathcal{P}$ and $\mathcal{R}$ are sets of rewrite rules.[3] It is *finite* if there exists no infinite sequence of rules $s_1 \to t_1, s_2 \to t_2, \dots$ from $\mathcal{P}$ such that for all $i > 0$, t_i is terminating with respect to $\mathcal{R}$, and there exist substitutions σ and τ with $t_i\sigma \to_\mathcal{R}^* s_{i+1}\tau$. A DP problem of the form $(\varnothing,\mathcal{R})$ is trivially finite. We recall the following (well-known) characterisation of termination of a TRS. A TRS $\mathcal{R}$ is terminating if and only if the DP problem $(\mathsf{DP}(\mathcal{R}),\mathcal{R})$ is finite. A *DP processor* is a mapping from DP problems to sets of DP problems. A DP processor Φ is *sound* if for all DP problems $(\mathcal{P},\mathcal{R})$, $(\mathcal{P},\mathcal{R})$ is finite whenever all DP problems in $\Phi((\mathcal{P},\mathcal{R}))$ are finite. 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 reduction pair $(\succcurlyeq_\mathcal{A},\succ_\mathcal{A})$.

▶ **Proposition 2.1** ([9]). *Let $(\succcurlyeq,\succ)$ be a reduction pair. Then the following DP processor (*reduction pair processor*) Φ^{RP} is sound:*

$$\Phi^{\mathsf{RP}}((\mathcal{P},\mathcal{R})) = \begin{cases} \{(\mathcal{P}',\mathcal{R})\} & \text{if } \mathcal{P}' \cup \mathcal{R} \subseteq \succcurlyeq \text{ and } \mathcal{P} \setminus \mathcal{P}' \subseteq \succ \\ \{(\mathcal{P},\mathcal{R})\} & \text{otherwise .} \end{cases}$$

The *dependency graph* of a DP problem $(\mathcal{P},\mathcal{R})$ (denoted by $\mathsf{DG}(\mathcal{P},\mathcal{R})$) is a graph whose nodes are the elements of $\mathcal{P}$. 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 $\mathsf{DG}(\mathcal{P},\mathcal{R})$ is a maximal subset of nodes such that for each pair of nodes $s \to t$, $u \to v$, there exists a path (possibly empty) from $s \to t$ to $u \to v$. Note that this is the

[2] The observation that pairs $l^\sharp \to u^\sharp$ such that $u \triangleleft l$ need not be considered is due to Dershowitz.
[3] We use a simpler definition of DP problems than [9, 23], which suffices in our context.

standard definition of SCC from graph theory (cf. [5], for instance), which slightly differs from the definition that is often used in the termination literature. 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 2.2** ([9]). *The following DP processor (*dependency graph processor*)* Φ^{DG} *is sound:* $\Phi^{\mathsf{DG}}((\mathcal{P},\mathcal{R})) = \{(\mathcal{P}',\mathcal{R}) \mid \mathcal{P}' \text{ is a nontrivial SCC of } \mathsf{DG}(\mathcal{P},\mathcal{R})\}$.

A *simple projection* is an argument filtering π such that for each function symbol $f \in \mathcal{F}$ of arity n, we have $\pi(f) = [1,\dots,n]$, and for each dependency pair symbol $f \in \mathcal{F}^\sharp \setminus \mathcal{F}$, $\pi(f) = i$ for some $1 \leqslant i \leqslant n$.

▶ **Proposition 2.3** ([10, 23]). *Let* π *be a simple projection. Then the following DP processor (*subterm criterion processor*)* Φ^{SC} *is sound:*

$$\Phi^{\mathsf{SC}}((\mathcal{P},\mathcal{R})) = \begin{cases} \{(\mathcal{P}',\mathcal{R})\} & \text{if } \pi(\mathcal{P}') \subseteq \trianglerighteq \text{ and } \pi(\mathcal{P}\setminus\mathcal{P}') \subseteq \triangleright \\ \{(\mathcal{P},\mathcal{R})\} & \text{otherwise .} \end{cases}$$

Let $\mathcal{R}$ be a TRS. A *proof tree* of $\mathcal{R}$ is a tree satisfying the following: the nodes are DP problems, the root is $(\mathsf{DP}(\mathcal{R}),\mathcal{R})$, each leaf is a DP problem of the shape $(\varnothing,\mathcal{R})$, and for each inner node $(\mathcal{P},\mathcal{R}')$, there exists a sound DP processor Φ such that each element of $\Phi((\mathcal{P},\mathcal{R}'))$ is a child of $(\mathcal{P},\mathcal{R}')$, and each of the edges from $(\mathcal{P},\mathcal{R}')$ to the elements of $\Phi((\mathcal{P},\mathcal{R}'))$ is labelled by Φ.

▶ **Theorem 2.4.** *Let* $\mathcal{R}$ *be a TRS such that there exists a proof tree* PT *of* $\mathcal{R}$*. Suppose that each edge label of that proof tree is a reduction pair, dependency graph, or subterm criterion processor. Then* $\mathcal{R}$ *is terminating.*

Furthermore, we recall some essentials of recursion theory, compare [20, 21]. We call the following functions over $\mathbb{N}$ *initial functions*: the constant zero function $z_n(x_1,\dots,x_n) = 0$ of all arities, the unary successor function $s(x) = x + 1$, and all projection functions $\pi^i_n(x_1,\dots,x_n) = x_i$ for $1 \leqslant i \leqslant n$. A class $\mathcal{C}$ of functions over $\mathbb{N}$ is *closed under composition* if for all $f\colon \mathbb{N}^m \to \mathbb{N}$ and $g_1,\dots,g_m\colon \mathbb{N}^n \to \mathbb{N}$ in $\mathcal{C}$, the function $h(x_1,\dots,x_n) = f(g_1(x_1,\dots,x_n),\dots,g_m(x_1,\dots,x_n))$ is in $\mathcal{C}$, as well. It is *closed under primitive recursion* if for all $f\colon \mathbb{N}^n \to \mathbb{N}$ and $g\colon \mathbb{N}^{n+2} \to \mathbb{N}$, the function h defined by $h(0,x_1,\dots,x_n) = f(x_1,\dots,x_n)$ and $h(y+1,x_1,\dots,x_n) = f(y,h(y,x_1,\dots,x_n),x_1,\dots,x_n)$ is contained in $\mathcal{C}$, as well. The *k-ary Ackermann function* A_k for $k \geqslant 2$ is defined recursively as follows:

$$\begin{aligned} A_k(0,\dots,0,x_k) &= x_k + 1 \\ A_k(x_1,\dots,x_{k-2},x_{k-1}+1,0) &= A_k(x_1,\dots,x_{k-1},1) \\ A_k(x_1,\dots,x_{k-2},x_{k-1}+1,x_k+1) &= A_k(x_1,\dots,x_{k-1},A_k(x_1,\dots,x_{k-2},x_{k-1}+1,x_k)) \\ A_k(x_1,\dots,x_{i-1},x_i+1,0,\dots,0,x_k) &= A_k(x_1,\dots,x_i,x_k,0,\dots,0,x_k) \end{aligned}$$

Here, the last equation is a schema instantiated for all $1 \leqslant i \leqslant k-2$. The set of *primitive recursive functions* is the smallest set of functions over $\mathbb{N}$ which contains all initial functions and is closed under composition and primitive recursion. The set of *multiply recursive functions* is the smallest set of functions over $\mathbb{N}$ which contains all initial functions and k-ary Ackermann functions, and is closed under composition and primitive recursion.

▶ **Proposition 2.5** ([21], Chapter 1). *For every multiply recursive function* f*, there exists a* k *such that* A_k *asymptotically dominates* f*.*

3 Main Theorem

In this short section we show that there exist TRSs whose termination is shown via Theorem 2.4 such that the derivational complexity cannot be bounded by a primitive recursive function. Furthermore we state our main result in precise terms.

▶ **Example 3.1.** Consider the following TRS $\mathcal{R}_2$, taken from [13, 12]: $\mathsf{i}(x) \circ (y \circ z) \to x \circ (\mathsf{i}(\mathsf{i}(y)) \circ z)$ and $\mathsf{i}(x) \circ (y \circ (z \circ w)) \to x \circ (z \circ (y \circ w))$. It is shown in [12] that $\mathsf{dc}_{\mathcal{R}_2}$ is not primitive recursive as the system encodes the Ackermann function.

Following Endrullis et al. [7, Example 11] we show termination of $\mathcal{R}_2$ employing Theorem 2.4. The dependency pairs with respect to $\mathcal{R}_2$ are:

$$\begin{array}{lll} 1\colon & \mathsf{i}(x) \circ^\sharp (y \circ z) \to x \circ^\sharp (\mathsf{i}(\mathsf{i}(y)) \circ z) & 2\colon \quad \mathsf{i}(x) \circ^\sharp (y \circ z) \to \mathsf{i}(\mathsf{i}(y)) \circ^\sharp z \\ 3\colon & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to x \circ^\sharp (z \circ (y \circ w)) & 4\colon \quad \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to z \circ^\sharp (y \circ w) \\ 5\colon & \mathsf{i}(x) \circ^\sharp (y \circ (z \circ w)) \to y \circ^\sharp w & \end{array}$$

First, consider the reduction pair induced by the polynomial algebra $\mathcal{A}$ defined as follows: $\circ^\sharp_{\mathcal{A}}(x, y) = y$, $\circ_{\mathcal{A}}(x, y) = y + 1$, and $\mathsf{i}_{\mathcal{A}}(x) = 0$. An application of the processor Φ^{RP} removes the dependency pairs $\{2, 4, 5\}$. Next, we apply the reduction pair induced by the polynomial algebra $\mathcal{B}$ with $\circ^\sharp_{\mathcal{B}}(x, y) = x$, $\circ_{\mathcal{B}}(x, y) = 0$, and $\mathsf{i}_{\mathcal{B}}(x) = x + 1$, which removes the remaining pairs $\{1, 3\}$. Hence we conclude termination of $\mathcal{R}$.

As a corollary we see that the derivational complexity function $\mathsf{dc}_{\mathcal{R}_2}$ is bounded from below by a function that grows faster than any primitive recursive function. On the other hand the complexity induced by the base techniques is linear. Let $\mathcal{P}_1$ denote the set of dependency pairs $\{2, 4, 5\}$ and let $\mathcal{P}_2 = \{1, 3\}$. It is easy to infer from the polynomial algebras $\mathcal{A}$ and $\mathcal{B}$ employed in the two applications of Φ^{RP} that the derivation height functions $\mathsf{dh}(t^\sharp, \to_{\mathcal{P}_1/(\mathcal{P}_2 \cup \mathcal{R})})$ and $\mathsf{dh}(t^\sharp, \to_{\mathcal{P}_2/\mathcal{R}})$ are linear in $|t|$. In order to obtain a tight lower bound we generalise Example 1.1

▶ **Example 3.2.** Let $k \geqslant 2$ and consider the following schematic rewrite rules, denoted as $\mathcal{R}_3^k$. It is easy to see that for fixed k, the TRS $\mathcal{R}_3^k$ encodes the k-ary Ackermann function:

$$\begin{aligned} \mathsf{Ack}_k(0, \ldots, 0, n) &\to \mathsf{S}(n) \\ \mathsf{Ack}_k(l_1, \ldots, l_{k-2}, \mathsf{S}(m), 0) &\to \mathsf{Ack}_k(l_1, \ldots, l_{k-2}, m, \mathsf{S}(0)) \\ \mathsf{Ack}_k(l_1, \ldots, l_{k-2}, \mathsf{S}(m), \mathsf{S}(n)) &\to \mathsf{Ack}_k(l_1, \ldots, l_{k-2}, m, \mathsf{Ack}_k(l_1, \ldots, l_{k-2}, \mathsf{S}(m), n)) \\ \mathsf{Ack}_k(l_1, \ldots, l_{i-1}, \mathsf{S}(l_i), 0, \ldots, 0, n) &\to \mathsf{Ack}_k(l_1, \ldots, l_i, n, 0, \ldots, 0, n) \end{aligned}$$

Here, the last rule is a schema instantiated for all $1 \leqslant i \leqslant k - 2$.

Following of the pattern of the termination proof of $\mathcal{R}_1$, we show termination of $\mathcal{R}_3^k$ by k applications of processor Φ^{SC}. The next lemma is a direct consequence of Proposition 2.5 and the above considerations.

▶ **Lemma 3.3.** *For any multiple recursive function f, there exists a TRS $\mathcal{R}$ whose derivational complexity function $\mathsf{dc}_{\mathcal{R}}$ majorises f. Furthermore termination of $\mathcal{R}$ follows by an application of Theorem 2.4.*

Lemma 3.3 shows that the DP *framework* admits much higher derivational complexities than the basic DP method. In [15, 16] we show that the derivational complexity induced by the DP method is *primitive recursive* in the complexity induced by the base technique, even

if standard refinements like *usable rules* or *dependency graphs* are considered. Examples 3.1 and 3.2 show that we cannot hope to achieve such a bound in the context of the DP framework. In the remainder of this paper we show that jumping to the next function class, the multiple recursive functions, suffices to bound the induced derivational complexities.

▶ **Definition 3.4.** Let $\mathcal{R}$ be a TRS whose termination is shown via Theorem 2.4, and PT the proof tree employed by the theorem. Let k be the maximum number of SCCs in any dependency graph employed by any instance of Φ^{DG} occurring in PT, and let $g\colon \mathbb{N} \to \mathbb{N}$ denote a monotone function such that:

$$g(n) \geqslant \max(\{k\} \cup \{\mathsf{dh}(t^\sharp, \to_{(\mathcal{P}\setminus\mathcal{Q})/(\mathcal{R}\cup\mathcal{Q})}) \mid \text{there exists an edge from } (\mathcal{P},\mathcal{R}) \text{ to } (\mathcal{Q},\mathcal{R}) \text{ in } \mathsf{PT} \text{ labelled by an instance of } \Phi^{\mathsf{RP}} \text{ and } |t| \leqslant n\}) \,.$$

Then g is called a *reduction pair function* of $\mathcal{R}$ with respect to PT.

Note that some reduction pair function can often be computed just by inspection of the employed instances of Φ^{RP}. Moreover, for most of the known reduction pairs (in particular, for virtually all reduction pairs currently applied by automatic termination provers), it is easily possible to compute a multiply recursive reduction pair function.

▶ **Theorem 3.5** (Main Theorem)**.** *Let $\mathcal{R}$ be a TRS whose termination is shown via Theorem 2.4 and let the reduction pair function g of $\mathcal{R}$ be multiple recursive. Then the derivational complexity function $\mathsf{dc}_\mathcal{R}$ with respect to $\mathcal{R}$ is bounded by a multiple recursive function. Furthermore this upper bound is tight.*

The proof of Theorem 3.5 makes use of a combinatorial argument, and is given in the next section. Here we present the proof plan. In proving the theorem we essentially use three different ideas. First, we exploit the given proof tree PT. We observe that, if we restrict our attention to termination of terms, we can focus on specific branches of the proof tree. Secondly, we define a TRS $\mathcal{S}$ simulating the initial TRS $\mathcal{R}$: $s \to_\mathcal{R} t$ implies $\mathsf{tr}(s) \to_\mathcal{S}^+ \mathsf{tr}(t)$. Here tr denotes a suitable interpretation of terms into the signature of the simulating TRS $\mathcal{S}$, compare Definition 4.11. The term $\mathsf{tr}(t)$ aggregates the termination arguments for t given by the DP processors in part of the proof tree which has been identified as particularly relevant for t in the first step. Finally, $\mathcal{S}$ will be simple enough to be compatible with a LPO so that we can employ Weiermann's result in [25] to deduce a multiple recursive upper bound on the derivational complexity with respect to $\mathcal{S}$ and conclusively with respect to $\mathcal{R}$.

Note that our proof technique is conceptually simpler than the technique we used in [15, 16] to show a triple exponential upper complexity bound on the most basic version of the dependency pair method: in order to establish the much lower bound in [15, 16], we constructed the whole proof argument from scratch, while in this paper we exploit Weiermann's analysis of the derivational complexity induced by LPO (see [25]). Still the here presented application of Weiermann's result is non-trivial and requires some preparation.

4 Proof of the Main Result

In this section we prove our main result, Theorem 3.5. We start with some preliminary definitions. Let $\mathcal{R}$ denote a TRS. We assume without loss of generality for each considered termination proof that whenever a DP processor Φ is applied to a DP problem $(\mathcal{P},\mathcal{R})$, then $\Phi((\mathcal{P},\mathcal{R})) \neq \{(\mathcal{P},\mathcal{R})\}$. For each of the DP processors Φ considered in this paper, the following facts are obvious: $(\mathcal{P}',\mathcal{R}') \in \Phi((\mathcal{P},\mathcal{R}))$ implies $\mathcal{P}' \subset \mathcal{P}$ and $\mathcal{R}' = \mathcal{R}$. Therefore, we assume throughout the rest of this paper that for each DP problem $(\mathcal{P},\mathcal{R})$, $\mathcal{P} \subseteq \mathsf{DP}(\mathcal{R})$.

In particular, each rule in $\mathcal{P}$ has the shape $s^\sharp \to t^\sharp$ for some $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. Moreover, $(\mathcal{P}',\mathcal{R}) \in \Phi((\mathcal{P},\mathcal{R}))$, $(\mathcal{P}'',\mathcal{R}) \in \Phi((\mathcal{P},\mathcal{R}))$, and $\mathcal{P}' \neq \mathcal{P}''$ imply $\mathcal{P}' \cap \mathcal{P}'' = \varnothing$. Therefore, each dependency pair can only appear in a single branch of a proof tree.

Let $\mathcal{G}$ be a dependency graph; we order the SCCs of $\mathcal{G}$ by assigning a *rank* to each of them. Let $\mathcal{P}$, $\mathcal{Q}$ denote two distinct SCCs of $\mathcal{G}$. We call $\mathcal{Q}$ *reachable* from $\mathcal{P}$ if there exist nodes $u \in \mathcal{P}$, $v \in \mathcal{Q}$ and a path in $\mathcal{G}$ from u to v. Let k be the number of SCCs in $\mathcal{G}$. Consider a bijection $\mathsf{rk}(\mathcal{G},\cdot)$ from the set of SCCs of $\mathcal{G}$ to $\{1,\dots,k\}$ such that $\mathsf{rk}(\mathcal{G},\mathcal{P}) > \mathsf{rk}(\mathcal{G},\mathcal{Q})$ whenever $\mathcal{Q}$ is reachable from $\mathcal{P}$ in $\mathcal{G}$. We call $\mathsf{rk}(\mathcal{G},\mathcal{P})$ the *rank* of an SCC $\mathcal{P}$ in $\mathcal{G}$. The rank of a dependency pair $l \to r$, denoted by $\mathsf{rk}(\mathcal{G}, l \to r)$, is the rank of $\mathcal{P}$ in $\mathcal{G}$ such that $l \to r \in \mathcal{P}$. Finally, the rank of a term t such that $t^\sharp \notin \mathsf{NF}(\mathcal{P}/\mathcal{R})$ for some SCC $\mathcal{P}$ of $\mathcal{G}$ is defined by $\mathsf{rk}(\mathcal{G},t) := \max\{\mathsf{rk}(\mathcal{G}, l \to r) \mid \exists\sigma\ t^\sharp \to_{\mathcal{R}}^* l\sigma\}$. Observe that $\mathsf{rk}(\mathcal{G},t)$ need not be defined, although t 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 $t \notin \mathsf{NF}(\mathcal{P}/\mathcal{R})$ for some SCC $\mathcal{P}$ of $\mathcal{G}$, then $\mathsf{rk}(\mathcal{G},t)$ is defined. Furthermore in this case $\mathsf{rk}(\mathcal{G},t) > 0$ and $\mathsf{dh}(t^\sharp, \to_{\mathcal{P}/\mathcal{R}}) > 0$.

We now change the definition of proof trees to better suit our needs. The main change is that for Φ^{DG}, all SCCs of the respective dependency graph are taken into account (not just, as usual, the nontrivial ones). While termination of trivial SCCs follows trivially, they might still form a bridge between nontrivial SCCs in a dependency graph thus crucially increasing the length of derivations. Example 4.2 below illustrates this. Moreover, as mentioned above, we use the proof tree to track its currently relevant part with respect to showing termination of a given term. This relevant part may very well include the DP problem belonging to a trivial SCC of a dependency graph.

▶ **Definition 4.1.** We redefine *proof trees*. A proof tree PT of $\mathcal{R}$ is a tree satisfying:

1) The nodes of PT are DP problems and $(\mathsf{DP}(\mathcal{R}),\mathcal{R})$ is the root of PT.
2) For every inner node $(\mathcal{P},\mathcal{R})$ in PT, there exists a sound DP processor Φ such that for each DP problem $(\mathcal{Q},\mathcal{R}) \in \Phi((\mathcal{P},\mathcal{R}))$, there exists an edge from $(\mathcal{P},\mathcal{R})$ to $(\mathcal{Q},\mathcal{R})$ in PT labelled by Φ.
3) Further, suppose $\Phi = \Phi^{\mathsf{DG}}$. Then there exists an edge from $(\mathcal{P},\mathcal{R})$ to a leaf $(\mathcal{Q},\mathcal{R})$ (labelled by Φ) for every trivial SCC $\mathcal{Q}$ of $\mathsf{DG}(\mathcal{P},\mathcal{R})$. Moreover the successors of $(\mathcal{P},\mathcal{R})$ are ordered from left to right in decreasing order with respect to the function rk.

The positions of nodes in PT are defined as usual as finite sequences of numbers. We write Greek letters for positions in PT. It is easy to verify that there is a one-to-one correspondence between proof trees according to Section 2 and Definition 4.1.

▶ **Example 4.2.** Consider the TRS $\mathcal{R}_4$ given by the rewrite rules: $\mathsf{d}(\mathsf{0}) \to \mathsf{0}$, $\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$, $\mathsf{e}(\mathsf{S}(x),y) \to \mathsf{e}(x,\mathsf{d}(y))$, and $\mathsf{sexp}(\mathsf{S}(x),\mathsf{e}(\mathsf{0},y)) \to \mathsf{sexp}(x,\mathsf{e}(y,\mathsf{S}(\mathsf{0})))$. The dependency pairs $\mathsf{DP}(\mathcal{R}_4)$ of $\mathcal{R}_4$ are:

$$1\colon\ \mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{d}^\sharp(x)$$
$$2\colon\ \mathsf{e}^\sharp(\mathsf{S}(x),y) \to \mathsf{d}^\sharp(y) \qquad 3\colon\ \mathsf{e}^\sharp(\mathsf{S}(x),y) \to \mathsf{e}^\sharp(x,\mathsf{d}(y))$$
$$4\colon\ \mathsf{sexp}^\sharp(\mathsf{S}(x),\mathsf{e}(\mathsf{0},y)) \to \mathsf{e}^\sharp(y,\mathsf{S}(\mathsf{0})) \qquad 5\colon\ \mathsf{sexp}^\sharp(\mathsf{S}(x),\mathsf{e}(\mathsf{0},y)) \to \mathsf{sexp}^\sharp(x,\mathsf{e}(y,\mathsf{S}(\mathsf{0})))$$

We start with the dependency graph processor Φ^{DG}. The dependency graph of the initial DP problem $(\mathsf{DP}(\mathcal{R}_4),\mathcal{R}_4)$ contains three nontrivial SCCs $\{1\}$, $\{3\}$, and $\{5\}$, and two trivial SCCs $\{2\}$ and $\{4\}$. Finiteness of each of the nontrivial SCCs can be shown by the reduction pair processor Φ^{RP} employing the following linear polynomial algebra $\mathcal{A}$: $\mathsf{S}_{\mathcal{A}}(x) = x+1$, $\mathsf{0}_{\mathcal{A}} = 0$, $\mathsf{d}_{\mathcal{A}}(x) = 2x$, $\mathsf{e}_{\mathcal{A}}(x,y) = 0$, $\mathsf{sexp}_{\mathcal{A}}(x,y) = 0$, and $\mathsf{d}^\sharp_{\mathcal{A}}(x) = \mathsf{e}^\sharp_{\mathcal{A}}(x,y) = \mathsf{sexp}^\sharp_{\mathcal{A}}(x,y) = x$.

Figure 1 shows the proof tree PT of this termination proof, where we make use of a simplified notation for edge labels. The nodes at positions 11, 31, and 51 are leaves in this proof tree because they are labelled by the DP problem $(\varnothing, \mathcal{R}_4)$, which is trivially finite. The nodes at positions 2 and 4 are leaves because $\{4\}$ and $\{2\}$ are trivial SCCs of the dependency graph employed. The following derivation illustrates the importance of trivial SCCs in our analysis:

$$\mathsf{e}^\sharp(\mathsf{S}^n(0), \mathsf{S}(0)) \to^*_{\{3\}\cup\mathcal{R}_4} \mathsf{e}^\sharp(\mathsf{S}(0), \mathsf{S}^{2^n}(0)) \to_{\{2\}} \mathsf{d}^\sharp(\mathsf{S}^{2^n}(0)) \to^*_{\{1\}\cup\mathcal{R}_4} \mathsf{S}^{2^{n+1}}(0)$$

Observe that the step within the trivial SCC $\{2\}$ connects two subderivations using the (otherwise unconnected) SCCs $\{3\}$ and $\{1\}$, thus increasing the length of the total derivation. In order to capture this behaviour, we keep track of trivial SCCs in our proof trees.

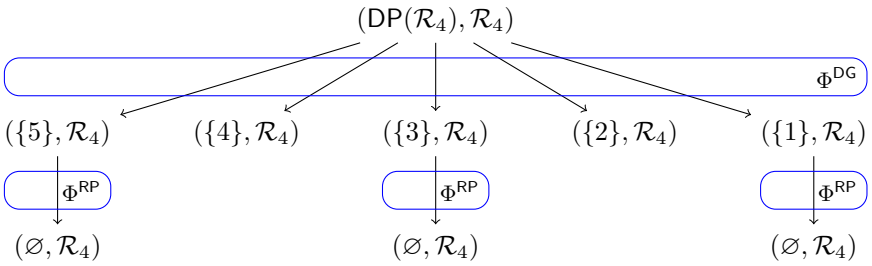

Figure 1 A proof tree of $\mathcal{R}_4$

For the remainder of this section, we assume that termination of $\mathcal{R}$ is shown Theorem 2.4 employing a proof tree PT. Further suppose that there exists a multiply recursive reduction pair function of $\mathcal{R}$, and fix such a function g. Let d be the depth of PT plus one. As stated in the proof plan, we now determine which part of the termination proof is active with respect to a given term. Intuitively, for many terms, only a part of PT is relevant for showing termination of that particular term. More specifically, for any term t, only a certain subset of the dependency pairs can be used for rewriting $t^\sharp$. Of these dependency pairs, we view the one occurring in the leftmost positions of PT (with respect to the order of PT) as the *current dependency pair*. We call the set of positions in which the current dependency pair occurs, the *current path of t in* PT.

▶ **Example 4.3** (continued from Example 4.2)**.** Consider the terms $t_1 = \mathsf{sexp}(\mathsf{S}(0), \mathsf{e}(0, \mathsf{S}(0)))$, $t_2 = \mathsf{sexp}(0, \mathsf{e}(\mathsf{S}(0), \mathsf{S}(0)))$, and $t_3 = \mathsf{e}(\mathsf{S}(0), \mathsf{S}(0))$. We obtain the following derivations: $t_1^\sharp \to_{\mathsf{DP}(\mathcal{R}_4)} t_2^\sharp$ and $t_1^\sharp \to_{\mathsf{DP}(\mathcal{R}_4)} t_3^\sharp$. Hence the term $t_1^\sharp$ is not a normal form with respect to $\{5\}/\mathcal{R}_4$ nor with $\{4\}/\mathcal{R}_4$. Similarly $t_3^\sharp$ is not a normal form with respect to $\{3\}/\mathcal{R}_4$ and $\{2\}/\mathcal{R}_4$. Therefore, the parts of PT highlighted in Figure 2 are particularly relevant for t_1 and t_3, respectively. The term $t_2^\sharp$ is a normal form with respect to $\mathsf{DP}(\mathcal{R}_4)/\mathcal{R}_4$, therefore no part of the proof tree is relevant to show termination for t_2.

The next definition formalises the relevant parts of a proof tree. As mentioned above it suffices to restrict the notion to a single path.

▶ **Definition 4.4.** The *current path* $\mathsf{PT}(t)$ of a term t in PT is defined as follows. If $t^\sharp \in \mathsf{NF}(\mathsf{DP}(\mathcal{R})/\mathcal{R})$, then $\mathsf{PT}(t)$ is the empty path, denoted as $()$. Otherwise, for each dependency pair $l \to r$ such that $t^\sharp \notin \mathsf{NF}(\{l \to r\}/\mathcal{R})$, consider the set of nodes whose label contains $l \to r$. By previous observations, each of these sets forms a path starting at the root node

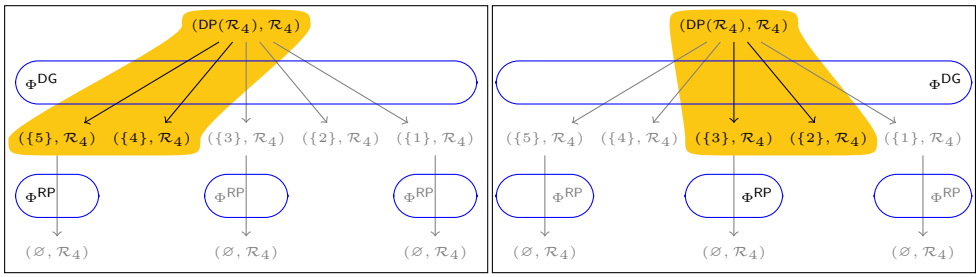

Figure 2 The relevant parts of the proof tree for two terms

of PT. The set of positions forming the leftmost of these paths is $\mathsf{PT}(t)$. We use $\mathsf{PT}_i(t)$ to project on single elements of $\mathsf{PT}(t) = (\alpha_1 = \epsilon, \alpha_2, \ldots, \alpha_n)$: if $i > n$, then $\mathsf{PT}_i(t) = \bot$, otherwise $\mathsf{PT}_i(t) = \alpha_i$.

▶ **Example 4.5** (continued from Example 4.3)**.** The current paths of t_1, t_2, and t_3 are the following: we have $\mathsf{PT}(t_1) = (\epsilon, 1)$, $\mathsf{PT}(t_2) = ()$, and $\mathsf{PT}(t_3) = (\epsilon, 3)$. For t_1, the projections on the single elements of the path are the following: $\mathsf{PT}_1(t_1) = \epsilon$, $\mathsf{PT}_2(t_1) = 1$, and $\mathsf{PT}_i(t_1) = \bot$ for $i > 2$.

Using the DP processors applied to the nodes of $\mathsf{PT}(t)$, we now define the complexity measure $\mathsf{norm}(t)$ assigned to t. For each DP processor, we use whatever value is naturally decreasing in the termination argument of that processor in order to get the associated bound. Given the reduction pair function g, $\mathsf{norm}(t)$ is easily computable.

▶ **Definition 4.6.** We define the mapping $\mathsf{norm}_i\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathbb{N} \cup \mathcal{T}(\mathcal{F},\mathcal{V}) \cup \{\bot\}$ for $i \in \mathbb{N}$ as follows: let $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ and $\alpha = \mathsf{PT}_i(t)$.

1) If $\alpha = \bot$, we set $\mathsf{norm}_i(t) = 0$ if $\mathsf{rt}(t)$ is a defined symbol, and $\mathsf{norm}_i(t) = \bot$ otherwise.
2) If $\alpha \neq \bot$ and $(\mathcal{P},\mathcal{R})$ denotes the node at position α in PT such that $(\mathcal{P},\mathcal{R})$ is a leaf, then either $\mathcal{P} = \varnothing$, or $\mathcal{P}$ is a trivial SCC of a dependency graph. In both cases, we set $\mathsf{norm}_i(t) = \mathsf{dh}(t^\sharp, \to_{\mathcal{P}/\mathcal{R}})$.
3) If $\alpha \neq \bot$ and $(\mathcal{P},\mathcal{R})$ denotes the node at position α in PT such that $(\mathcal{P},\mathcal{R})$ is an inner node, then suppose Φ labels each edge starting from $(\mathcal{P},\mathcal{R})$:
   - If Φ is Φ^{RP} with $\Phi((\mathcal{P},\mathcal{R})) = \{(\mathcal{Q},\mathcal{R})\}$, then we set $\mathsf{norm}_i(t) = \mathsf{dh}(t^\sharp, \to_{(\mathcal{P}\setminus\mathcal{Q})/(\mathcal{Q}\cup\mathcal{R})})$.
   - If Φ is Φ^{DG} using a dependency graph $\mathcal{G}$, then we set $\mathsf{norm}_i(t) = \mathsf{rk}(\mathcal{G}, t)$.
   - If Φ is Φ^{SC} using a simple projection π, then we set $\mathsf{norm}_i(t) = \pi(t^\sharp)$.

We extend the mappings norm_i to the *norm* of a term: $\mathsf{norm}(t) = (\mathsf{norm}_1(t), \ldots, \mathsf{norm}_d(t))$.

The central idea behind the complexity measures used in the mapping norm is that rewrite steps induce lexicographical decreases in the norm of the considered term.

▶ **Definition 4.7.** We define the following order $\sqsupset$ on $\mathbb{N} \cup \mathcal{T}(\mathcal{F},\mathcal{V}) \cup \{\bot\}$. We have $a \sqsupset b$ if and only if one of the following properties holds: (i) $a \in \mathbb{N}$, $b \in \mathbb{N}$, and $a > b$, (ii) $a \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $b \in \mathcal{T}(\mathcal{F},\mathcal{V})$, and $a(\to_{\mathcal{R}} \cup \rhd)^+ b$, and (iii) $a \in \mathcal{T}(\mathcal{F},\mathcal{V})$ and $b = 0$, or $a \in \mathcal{T}(\mathcal{F},\mathcal{V}) \cup \mathbb{N}$ and $b = \bot$.

We define $\sqsupseteq$ to be the reflexive closure of $\sqsupset$. We write $\sqsupset^{\mathrm{lex}}$ and $\sqsupseteq^{\mathrm{lex}}$ for the lexicographic extensions of $\sqsupset$ and $\sqsupseteq$, respectively. Note that termination of $\mathcal{R}$ implies well-foundedness of $(\to_{\mathcal{R}} \cup \rhd)^+$, hence $\sqsupset$ is well-founded.

▶ **Lemma 4.8.** *Let s and t be terms such that $s \xrightarrow{\geq\epsilon}_{\mathcal{R}} t$. For all $1 \leqslant i < d$, if $\mathsf{PT}_i(s) = \mathsf{PT}_i(t)$ and $\mathsf{norm}_i(s) = \mathsf{norm}_i(t)$, then either $\mathsf{PT}_{i+1}(t) = \bot$, or $\mathsf{PT}_{i+1}(s) = \mathsf{PT}_{i+1}(t)$.*

Proof Sketch. Straightforward case distinction on the node at position $\mathsf{PT}_i(s)$. ◀

▶ **Lemma 4.9.** *For any terms s and t such that $s \xrightarrow{\geq\epsilon}_{\mathcal{R}} t$, we have $\mathsf{norm}(s) \sqsupseteq^{\mathrm{lex}} \mathsf{norm}(t)$.*

Proof. We can assume that $\mathsf{rt}(t)$ is a defined symbol. Otherwise, $\mathsf{norm}_i(t) = \bot$ for all $1 \leqslant i \leqslant d$, and hence $\mathsf{norm}(t) = (\bot, \ldots, \bot)$, so the lemma would be trivial. As $\mathsf{rt}(s) = \mathsf{rt}(t)$, $\mathsf{rt}(s)$ is also defined. Hence, $s^\sharp \to_{\mathcal{R}} t^\sharp$.

We now show the following by induction on $d-i$: if for all $1 \leqslant j < i$, $\mathsf{norm}_j(s) = \mathsf{norm}_j(t)$, then $(\mathsf{norm}_i(s), \ldots, \mathsf{norm}_d(s)) \sqsupseteq^{\mathrm{lex}} (\mathsf{norm}_i(t), \ldots, \mathsf{norm}_d(t))$. Clearly, this claim implies the lemma, so the remainder of this proof is devoted to it. Applying Lemma 4.8 $i-1$ times reveals that $\mathsf{PT}_i(t)$ is either $\bot$ or the same as $\mathsf{PT}_i(s)$. We perform case distinction on $\mathsf{PT}_i(t)$. We restrict our attention to the interesting case that $\mathsf{PT}_i(s) = \mathsf{PT}_i(t) = \alpha$, $\alpha \neq \bot$, and $(\mathcal{P}, \mathcal{R})$ denotes the node at α in PT such that $(\mathcal{P}, \mathcal{R})$ is an inner node.

Suppose Φ labels the edges starting from $(\mathcal{P}, \mathcal{R})$. If Φ is Φ^{RP} with $\Phi((\mathcal{P}, \mathcal{R})) = \{(\mathcal{Q}, \mathcal{R})\}$, then because of $s^\sharp \to_{\mathcal{R}} t^\sharp$, the inequality $\mathsf{dh}(s^\sharp, \to_{(\mathcal{P}\setminus\mathcal{Q})/(\mathcal{Q}\cup\mathcal{R})}) \geqslant \mathsf{dh}(t^\sharp, \to_{(\mathcal{P}\setminus\mathcal{Q})/(\mathcal{Q}\cup\mathcal{R})})$. Thus $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(t)$ holds. If Φ is Φ^{DG} using a dependency graph $\mathcal{G}$, then for each SCC $\mathcal{P}_j$ in $\mathcal{G}$, $s^\sharp$ can only be a normal form of $\mathcal{P}_j/\mathcal{R}$ if $t^\sharp$ is one, as well. Therefore, we have $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(t)$ in that case, too. If Φ is Φ^{SC} with simple projection π, then $\mathsf{norm}_i(s) = \pi(s^\sharp) \to_{\mathcal{R}}^{=} \pi(t^\sharp) = \mathsf{norm}_i(t)$, and hence $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(t)$.

So, regardless of the processor Φ, we have $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(t)$. If $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(t)$, then the claim trivially follows. On the other hand, if $\mathsf{norm}_i(s) = \mathsf{norm}_i(t)$, then the claim holds by induction hypothesis. ◀

The following lemma extends Lemma 4.9 to root steps $s \xrightarrow{\epsilon}_{\mathcal{R}} t$. However, in this case, we do not consider only the root position of t, but all positions that were "created" by the rewrite step. So essentially, we show that such a step causes a decrease in $\sqsupset^{\mathrm{lex}}$ from s to subterms of t. The restriction on positions p below takes care of the Dershowitz condition in the definition of dependency pairs and the substitution of the applied rewrite rule.

▶ **Lemma 4.10.** *For any terms s and t such that $s \xrightarrow{\epsilon}_{\mathcal{R}} t$, we have $\mathsf{norm}(s) \sqsupset^{\mathrm{lex}} \mathsf{norm}(t|_p)$ for all $p \in \mathsf{Pos}(t)$ such that $t|_p \not\trianglelefteq s$.*

Proof. For this proof, we fix p, and let $u = t|_p$. We can assume that $\mathsf{rt}(u)$ is a defined symbol. Otherwise, $\mathsf{norm}(u) = (\bot, \ldots, \bot)$, but $\mathsf{norm}(s) \sqsupseteq^{\mathrm{lex}} (0, \ldots, 0)$ (note that $\mathsf{rt}(s)$ is defined), so the lemma would be immediate. Hence, we have $s^\sharp \to_{\mathsf{DP}(\mathcal{R})} u^\sharp$ using some dependency pair $l \to r$. Let j be the greatest number between 1 and d such that $\mathsf{PT}_j(s) \neq \bot$, the node at $\mathsf{PT}_j(s)$ is $(\mathcal{Q}, \mathcal{R})$, and $\mathcal{Q}$ contains $l \to r$. Note that such a number exists: since $s^\sharp \to_{\mathsf{DP}(\mathcal{R})} u^\sharp$, we have $\mathsf{PT}_1(s) = \epsilon$, which denotes $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$, and $\mathsf{DP}(\mathcal{R})$ contains $l \to r$. Let $\alpha = \mathsf{PT}_j(s)$. We distinguish whether $\mathsf{PT}_j(u) = \alpha$. This determines whether the strict part of the lexicographic decrease must happen at index j or at an earlier index.

Suppose $\mathsf{PT}_j(u) = \alpha$. Then we show that for all $1 \leqslant i \leqslant j$, $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$ holds, and $\mathsf{norm}_j(s) \sqsupset \mathsf{norm}_j(u)$. From these two properties, the lemma follows. In order to show them, we fix some $1 \leqslant i \leqslant j$. Let $\beta = \mathsf{PT}_i(s) = \mathsf{PT}_i(u)$.

1) If the node at position β is a leaf of PT, then $i = j$, and $\mathcal{Q}$ is a trivial SCC of a dependency graph. By assumption, $l \to r \in \mathcal{Q}$. Therefore, $\mathsf{dh}(s^\sharp, \to_{\mathcal{Q}/\mathcal{R}}) > \mathsf{dh}(u^\sharp, \to_{\mathcal{Q}/\mathcal{R}})$, and thus $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$.
2) If the node $(\mathcal{P}, \mathcal{R})$ at position β is an inner node of PT, let Φ be the label of each edge starting from $(\mathcal{P}, \mathcal{R})$. Obviously, $\mathcal{Q} \subseteq \mathcal{P}$, and therefore $l \to r \in \mathcal{P}$. For all possibilities of Φ, the semantics of Φ imply that $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$. Moreover, if $i = j$, then $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$ follows. In more detail: If Φ is Φ^{RP}, then let $\{(\mathcal{P}', \mathcal{R})\} = \Phi((\mathcal{P}, \mathcal{R}))$. Since $l \to r \in \mathcal{P}$, it follows that $\mathsf{dh}(s^\sharp, \to_{(\mathcal{P}\setminus\mathcal{P}')/(\mathcal{P}'\cup\mathcal{R})}) \geqslant \mathsf{dh}(u^\sharp, \to_{(\mathcal{P}\setminus\mathcal{P}')/(\mathcal{P}'\cup\mathcal{R})})$, and

thus $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$. If $i = j$, then by definition of j, $l \to r$ is contained in $\mathcal{P} \setminus \mathcal{P}'$. Therefore, $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$ in that case. If Φ is Φ^{DG} using a dependency graph $\mathcal{G}$, then by definition of SCCs in a dependency graph, $\mathsf{rk}(\mathcal{G}, s) \geqslant \mathsf{rk}(\mathcal{G}, l \to r) \geqslant \mathsf{rk}(\mathcal{G}, u)$, hence $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$. If $i = j$, then by definition of j, $\mathsf{rk}(\mathcal{G}, s) \neq \mathsf{rk}(\mathcal{G}, l \to r)$. Thus, $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$ in that case. If Φ is Φ^{SC} with $\Phi((\mathcal{P}, \mathcal{R})) = (\mathcal{P}', \mathcal{R})$ and simple projection π, then $\pi(s^\sharp) \trianglerighteq \pi(u^\sharp)$, and hence $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$. If $i = j$, then by definition of j, $l \to r \in \mathcal{P} \setminus \mathcal{P}'$, and hence $\pi(s^\sharp) \triangleright \pi(u^\sharp)$ and $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$ in that case.

In all cases, it follows that for all $1 \leqslant i \leqslant j$, $\mathsf{norm}_i(s) \sqsupseteq \mathsf{norm}_i(u)$ holds, and $\mathsf{norm}_j(s) \sqsupset \mathsf{norm}_j(u)$.

Now suppose $\mathsf{PT}_j(u) \neq \alpha$. Then let i be the greatest number between 1 and j such that $\mathsf{PT}_i(s) = \mathsf{PT}_i(u) = \beta$. As β is a prefix of α, the node $(\mathcal{P}, \mathcal{R})$ at β is an inner node of PT. Let Φ be the label of each edge starting from $(\mathcal{P}, \mathcal{R})$. Using the arguments from above, we see that $\mathsf{norm}_{i'}(s) \sqsupseteq \mathsf{norm}_{i'}(u)$ for all $1 \leqslant i' \leqslant i$. We now show that $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$ or $\mathsf{norm}_{i+1}(s) \sqsupset \mathsf{norm}_{i+1}(u)$ holds.

1) If Φ is Φ^{RP} or Φ^{SC}, then by our assumptions, $\mathsf{PT}_{i+1}(s) = \beta 1$. Since β has only one child in this case, this implies $\mathsf{PT}_{i+1}(u) = \bot$. Thus, $\mathsf{norm}_{i+1}(s) > 0 = \mathsf{norm}_{i+1}(u)$.

2) If Φ is Φ^{DG}, then $\mathsf{norm}_i(s) \neq \mathsf{norm}_i(u)$, since $\mathsf{PT}_{i+1}(s) \neq \mathsf{PT}_{i+1}(u)$ by assumption. Thus $\mathsf{norm}_i(s) \sqsupset \mathsf{norm}_i(u)$.

In both cases, it follows that $\mathsf{norm}(s) \sqsupset^{\mathrm{lex}} \mathsf{norm}(u)$, which is what we wanted to show. ◀

Up to now, we have shown norm decreases under rewriting. For rewrite steps whose redex position is at the root, this decrease is even strict. In order to turn this into an upper bound on derivational complexities, we still have to do some work: we also have to consider the norm of all proper subterms of a considered term, and the range of norm is not suitable for direct complexity measures yet. We now solve these problems by lifting the range of norm to the term level and simulating derivations of $\mathcal{R}$ at that level.

For the rest of this section let A be the maximum arity of any function symbol occurring in $\mathcal{R}$, and $C := \max\{\mathsf{dp}(r) \mid l \to r \in \mathcal{R}\}$. Depending on PT, d, A, C, and g, we now define a TRS $\mathcal{S}$ which simulates $\mathcal{R}$ and is compatible with LPO. The simulating TRS $\mathcal{S}$ is based on a mapping tr (see Definition 4.13) such that $s \to_{\mathcal{R}} t$ implies $\mathsf{tr}(s) \to_{\mathcal{S}}^{+} \mathsf{tr}(t)$. Given a term t, tr employs the $d + A$-ary function symbol f. The first d arguments of f are used to represent $\mathsf{norm}(t)$; the last a arguments of f are used to represent $\mathsf{tr}(t')$ for each direct subterm t' of t. In the simulation, we often have to recalculate the first d arguments of each f. Due to the definition of norm, we know that for each term t and $1 \leqslant i \leqslant d$, either $\mathsf{norm}_i(t) \in \mathbb{N}$ and $\mathsf{norm}_i(t) \leqslant g(|t|)$, or $\mathsf{norm}_i(t) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and $\mathsf{norm}_i(t) \trianglelefteq t$, or $\mathsf{norm}_i(t) = \bot$. We use a unary function symbol choice such that $\mathsf{choice}(\mathsf{tr}(t))$ rewrites to the representations of $g(|t|)$, $\mathsf{tr}(t')$ for each subterm t' of t, and $\bot$. In particular we often we use terms of the shape $\mathsf{choice}(\mathsf{f}(\mathsf{0}, \dots, \mathsf{0}, x_1, \dots, x_A))$ in the definition of $\mathcal{S}$, so we use the abbreviation $N(x_1, \dots, x_A)$ for this.

The main tool for achieving the simulation of a root rewrite step $s \xrightarrow{\epsilon}_{\mathcal{R}} t$ are rules which build the new f symbols for the positions in t "created" by the step. These are at most A^{C+1} many new positions, and each proper subterm of s may be duplicated at most that many times. As a very simple example, if $d = 3$, $A = 1$, and $C = 1$, this behaviour is simulated

by rules of the following shape:

$$\begin{aligned}
\mathsf{f}(u_1, \mathsf{S}(u_2), u_3, x) &\to \mathsf{f}(u_1, u_2, N(x), \mathsf{f}(u_1, u_2, N(x), x)) \\
\mathsf{f}(u_1, \mathsf{f}(v_1, v_2, v_3, y), u_3, x) &\to \mathsf{f}(u_1, y, N(x), \mathsf{f}(u_1, y, N(x), x)) \\
\mathsf{f}(u_1, \mathsf{f}(v_1, v_2, v_3, y), u_3, x) &\to \mathsf{f}(u_1, \mathsf{0}, N(x), \mathsf{f}(u_1, \mathsf{0}, N(x), x)) \\
\mathsf{f}(u_1, \mathsf{0}, u_3, x) &\to \mathsf{f}(u_1, \bot, N(x), \mathsf{f}(u_1, \bot, N(x), x))
\end{aligned}$$

We use similar rules for decreases of u_1 or u_3 with respect to the ordering $\sqsupset$. In order to write down these rules concisely for arbitrary A and C, we make use of the following abbreviation M_i^k (for $i \in \{1, \dots, d\}$ and $k \in \mathbb{N}$):

$$\begin{aligned}
&M_i^0(u_1, \dots, u_i, x_1, \dots, x_A) = \mathsf{f}(u_1, \dots, u_i, \overline{N(x_1, \dots, x_A)}, x_1, \dots, x_A) \\
&M_i^{k+1}(u_1, \dots, u_i, x_1, \dots, x_A) \\
&\quad = \mathsf{f}(u_1, \dots, u_i, \overline{N(\overline{M_i^k(u_1, \dots, u_i, x_1, \dots, x_A)})}, \overline{M_i^k(u_1, \dots, u_i, x_1, \dots, x_A)})
\end{aligned}$$

Here u_i ($i \in \{1, \dots, i\}$) and x_j ($j \in \{1, \dots, A\}$) denote variables and $\bar{t}$ is an abbreviation of $t, \dots, t$, where the number of repetitions of t follows from the context.

Consider the reduction pair function g of $\mathcal{R}$. Since g is assumed to be a multiple recursive function, it is an easy exercise to define a TRS $\mathcal{S}'$ (employing the constructors S, $\mathsf{0}$) that computes the function g: one can simply define g using only initial functions, composition, primitive recursion, and k-ary Ackermann functions, and directly turn the resulting definition of g into rewrite rules. That is, there exists a TRS $\mathcal{S}'$ and a defined function symbol g such that $\mathsf{g}(\mathsf{S}^n(\mathsf{0})) \to^*_{\mathcal{S}'} \mathsf{S}^{g(n)}(\mathsf{0})$. Here we use $\mathsf{S}^n(\mathsf{0})$ to denote $\mathsf{S}(\dots(\mathsf{S}(\mathsf{0}))\dots)$, where S is repeated n times. Moreover, $\mathcal{S}'$ is compatible with a LPO such that the precedence $\succ$ of the LPO includes $\mathsf{g} \succ \mathsf{S} \succ \mathsf{0}$.

▶ **Definition 4.11.** Consider the following (schematic) TRS $\mathcal{S}$, where $1 \leqslant i \leqslant d$ and $1 \leqslant j \leqslant A$. Here we use $\vec{x}$ as a shorthand for $x_1, \dots, x_A$.

$$\begin{aligned}
1_i&: && \mathsf{f}(u_1, \dots, u_{i-1}, \mathsf{S}(u_i), u_{i+1}, \dots, u_d, \vec{x}) \to M_i^C(u_1, \dots, u_i, \vec{x}) \\
2_{i,j}&: && \mathsf{f}(u_1, \dots, u_{i-1}, \mathsf{f}(v_1, \dots, v_d, \vec{y}), u_{i+1}, \dots, u_d, \vec{x}) \to M_i^C(u_1, \dots, u_{i-1}, y_j, \vec{x}) \\
3_i&: && \mathsf{f}(u_1, \dots, u_{i-1}, \mathsf{f}(v_1, \dots, v_d, \vec{y}), u_{i+1}, \dots, u_d, \vec{x}) \to M_i^C(u_1, \dots, u_{i-1}, \mathsf{0}, \vec{x}) \\
4_i&: && \mathsf{f}(u_1, \dots, u_{i-1}, \mathsf{0}, u_{i+1}, \dots, u_d, \vec{x}) \to M_i^C(u_1, \dots, u_{i-1}, \bot, \vec{x}) \\
5_j&: && \mathsf{size}(\mathsf{f}(u_1, \dots, u_d, \vec{x})) \to \times_A(\mathsf{size}(x_j)) \\
6&: && \mathsf{size}(\mathsf{c}) \to \mathsf{S}(\mathsf{0}) \\
7&: && \times_A(\mathsf{S}(x)) \to \mathsf{S}^A(\times_A(x)) \\
8&: && \times_A(\mathsf{0}) \to \mathsf{0} \\
9&: && \mathsf{f}(u_1, \dots, u_d, \vec{x}) \to \mathsf{c} \\
10_j&: && \mathsf{f}(u_1, \dots, u_d, \vec{x}) \to x_j \\
11&: && \mathsf{h}(x) \to \mathsf{f}(\overline{N(\overline{x})}, \overline{x}) \\
12&: && \mathsf{z} \to \mathsf{f}(\overline{N(\overline{\mathsf{c}})}, \overline{\mathsf{c}}) \\
13_j&: && \mathsf{choice}(\mathsf{f}(u_1, \dots, u_d, \vec{x})) \to x_j \\
14&: && \mathsf{choice}(x) \to \mathsf{g}(\mathsf{size}(x)) \\
15&: && \mathsf{choice}(x) \to \bot
\end{aligned}$$

These rules are augmented by $\mathcal{S}'$ defining the function symbol g. The signatures of $\mathcal{S}'$ and $\mathcal{S} \setminus \mathcal{S}'$ are disjoint with the exception of g and the constructors S and $\mathsf{0}$.

Note that $\mathcal{S}$ depends only on the constants d, A, C, and the reduction pair function g. The rules 1_i–4_i are the main rules for the simulation of the effects of a single rewrite step $s \xrightarrow{\epsilon}_{\mathcal{R}} t$ in $\mathcal{S}$. These rules employ that $\mathsf{norm}_i(s^\sharp) \sqsupset \mathsf{norm}_i((t|_p)^\sharp)$ for all $p \in \mathsf{Pos}(t)$ such that $t|_p \not\trianglelefteq s$, and $\mathsf{norm}_{i'}(s^\sharp) \sqsupseteq \mathsf{norm}_{i'}((t|_p)^\sharp)$ for all $1 \leqslant i' \leqslant i$. They are also responsible for creating the at most A^{C+1} many new positions and copies of each subterm of s in t. The rules 5_j–8 define a function symbol size, that is, $\mathsf{size}(s)$ reduces to a numeral $\mathsf{S}^n(\mathsf{0})$ such that $n \geqslant |s|$. The rules 9–10_j make sure that any superfluous positions and copies of subterms created by the rules of type 1_i–4_i can be deleted. The rules 11 and 12 guarantee that the simulating derivation can be started with a small enough initial term. The rules 13_j–15 define the function symbol choice introduced in the abbreviations M_i^C, and N. Loosely speaking, $\mathsf{choice}(t)$ is an upper bound of all $\mathsf{norm}_i(t)$ with respect to $\sqsupseteq$.

The next lemma essentially follows from Weiermann's result that LPO induces multiple recursive derivational complexity.

▶ **Lemma 4.12.** *The function $\mathsf{dc}_{\mathcal{S}}$ is multiply recursive.*

Proof. The TRS $\mathcal{S}'$ computing g can be shown terminating using an LPO such that the precedence $\succ$ of the LPO contains $\mathsf{g} \succ \mathsf{S} \succ \mathsf{0}$. It is easy to check that extending this precedence by

$$\mathsf{h}, \mathsf{z} \succ \mathsf{f} \succ \mathsf{choice} \succ \mathsf{g} \qquad \mathsf{size} \succ \mathsf{S} \succ \times, \mathsf{0}, \mathsf{c}, \bot\,,$$

makes the whole TRS $\mathcal{S}$ compatible with this LPO. By [25], termination of a finite TRS by an LPO implies that the derivational complexity of that TRS is multiple recursive. Note that the definition of multiple recursion used in this paper and the definition given in [25] coincide by [19]. Thus, $\mathsf{dc}_{\mathcal{S}}$ is a multiple recursive function. ◀

For the remainder of this section, let $\mathcal{H}$ denote the signature of $\mathcal{S}$. We now show that the TRS $\mathcal{S}$ indeed simulates $\mathcal{R}$ as requested. Since the proofs of the lemmas in this part of the paper are rather straightforward, but very technical. for them.

▶ **Definition 4.13.** The mapping $\mathsf{tr}\colon \mathcal{T}(\mathcal{F}) \to \mathcal{T}(\mathcal{H})$ is defined by the equation $\mathsf{tr}(t) = \mathsf{f}((\mathsf{norm}_1(t))^*, \dots, (\mathsf{norm}_d(t))^*, \mathsf{tr}(t_1), \dots, \mathsf{tr}(t_n), \mathsf{c}, \dots, \mathsf{c})$, where $t = f(t_1, \dots, t_n)$ and the operator $(\cdot)^*$ is defined for a term s as follows:

$$u^* := \begin{cases} \bot & \text{if } u = \bot \\ \mathsf{S}^u(\mathsf{0}) & \text{if } u \in \mathbb{N} \\ \mathsf{tr}(u) & \text{if } u \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \end{cases}$$

We define an equivalence $s \approx t$ on $\mathcal{T}(\mathcal{H})$. If $s = \mathsf{c}$, then $t = \mathsf{c}$. Otherwise if $s = \mathsf{f}(u_1, \dots, u_d, s_1, \dots, s_A)$, then $t = \mathsf{f}(v_1, \dots, v_d, t_1, \dots, t_A)$ such that $s_i \approx t_i$ for all $1 \leqslant i \leqslant A$.

▶ **Lemma 4.14.** *For all ground terms s with $t \approx \mathsf{tr}(s)$, $\mathsf{size}(t) \to_{\mathcal{S}}^+ \mathsf{S}^n(\mathsf{0})$ where $n \geqslant |s|$.*

Proof Sketch. The proof uses induction on $|s|$. The main ingredient of the inductive argument is straightforward application of the rules 5_j–8 of $\mathcal{S}$. ◀

▶ **Lemma 4.15.** *The following properties of $\mathcal{S}$ hold:*

1) *If $s = \mathsf{f}(u_1^*, \dots, u_d^*, \vec{s})$, $t = \mathsf{tr}(t') = \mathsf{f}(v_1^*, \dots, v_d^*, \vec{s})$, and $(u_1, \dots, u_d) \sqsupset^{\mathrm{lex}} (v_1, \dots, v_d)$, then $s \to_{\mathcal{S}}^+ t$.*
2) *For any ground terms $s = \mathsf{tr}(s')$ and $t = \mathsf{tr}(t')$, $s' \xrightarrow{\epsilon}_{\mathcal{R}} t'$ implies $s \to_{\mathcal{S}}^+ t$.*

3) *If* $a \to_{\mathcal{R}} b$ *and* $\mathsf{tr}(a) \to_{\mathcal{S}}^{+} \mathsf{tr}(b)$*, then for any* n*-ary function symbol* $f \in \mathcal{F}$*, we have* $s = \mathsf{tr}(f(t_1, \ldots, a, \ldots, t_n)) \to_{\mathcal{S}}^{+} \mathsf{tr}(f(t_1, \ldots, b, \ldots, t_n))$.

Proof Sketch. The proof uses mutual induction on $\mathsf{dh}(s, \to_{\mathcal{S}} \cup \rhd)$. Note that by Lemma 4.12, $\mathcal{S}$ terminates, and hence $\to_{\mathcal{S}} \cup \rhd$ is well-founded. The following are the central parts of the inductive arguments for the three properties of the lemma:

1) Property 1 states that all lexicographic decreases in the norm of a term occurring during a simulation can indeed be handled by $\mathcal{S}$. The main ingredient for this part of the proof is the application of the rules 1_i–4_i of $\mathcal{S}$.
2) Property 2 states that the simulation of a root rewrite step can indeed be done within $\mathcal{S}$. The outline of this part of the proof is the following: The first step of the simulation is an application of one of the rules 1_i–4_i. This yields a rewrite step of the shape

   $$s \to_{\mathcal{S}} M_i^C((\mathsf{norm}_1(s'))^*, \ldots, (\mathsf{norm}_{i-1}(s'))^*, v_i'^*, s_1, \ldots, s_n)$$

   such that $v_i' \sqsupseteq \mathsf{norm}_i(s')$. The proof then proceeds to show by side induction on $\mathsf{dp}(u)$ the following for subterms u of t', which concludes Property 2 of the lemma:

   $$M_i^{\mathsf{dp}(u)}((\mathsf{norm}_1(s'))^*, \ldots, (\mathsf{norm}_{i-1}(s'))^*, v_i'^*, s_1, \ldots, s_n) \to_{\mathcal{S}}^{*} \mathsf{tr}(u) \ .$$

   The most important argument within the side induction is the application of Lemma 4.10.
3) Property 3 essentially states that Property 2 is closed under contexts. The main ingredient for this part of the proof is the application of Lemma 4.9.

◀

The next lemma is an easy consequence of Lemma 4.15(2) and (3).

▶ **Lemma 4.16.** *For any ground terms* s *and* t*,* $s \to_{\mathcal{R}} t$ *implies* $\mathsf{tr}(s) \to_{\mathcal{S}}^{+} \mathsf{tr}(t)$.

Lemma 4.16 yields that the length of any derivation in $\mathcal{R}$ can be estimated by the maximal derivation height with respect to $\mathcal{S}$. To extend Lemma 4.16 so that the derivational complexity function $\mathsf{dc}_{\mathcal{R}}$ can be measured via the function $\mathsf{dc}_{\mathcal{S}}$ we make use of the following lemma.

▶ **Lemma 4.17.** *For any ground term* t*, we have* $\mathsf{h}^{\mathsf{dp}(t)}(\mathsf{z}) \to_{\mathcal{S}}^{+} \mathsf{tr}(t)$.

Proof Sketch. The proof uses induction on $\mathsf{dp}(t)$. The main ingredients of the inductive argument are rules 11–12 of $\mathcal{S}$ and $\mathsf{choice}(t')$ essentially being an upper bound for all $\mathsf{norm}_i(t')$. ◀

We recall our main result, Theorem 3.5.

▶ **Theorem** (Main Theorem)**.** *Let* $\mathcal{R}$ *be a TRS whose termination is shown via Theorem 2.4 and let the reduction pair function* g *of* $\mathcal{R}$ *be multiple recursive. Then the derivational complexity function* $\mathsf{dc}_{\mathcal{R}}$ *with respect to* $\mathcal{R}$ *is bounded by a multiple recursive function. Furthermore this upper bound is tight.*

Proof. Let $\mathcal{S}$ be the simulating TRS for $\mathcal{R}$, as defined over the course of this section. Due to Lemma 4.12, $\mathsf{dc}_{\mathcal{S}}$ is multiply recursive. Let t be a term. Without loss of generality, we assume that t is ground. Due to Lemmas 4.16 and 4.17, we have the following inequalities:

$$\mathsf{dh}(t, \to_{\mathcal{R}}) \leqslant \mathsf{dh}(\mathsf{tr}(t), \to_{\mathcal{S}}) \leqslant \mathsf{dh}(\mathsf{h}^{\mathsf{dp}(t)}(\mathsf{z}), \to_{\mathcal{S}}) \ .$$

Note that $|\mathsf{h}^{\mathsf{dp}(t)}(\mathsf{z})| \leqslant |t|$. Hence for all $n \in \mathbb{N}$: $\mathsf{dc}_{\mathcal{R}}(n) \leqslant \mathsf{dc}_{\mathcal{S}}(n)$. Thus, $\mathsf{dc}_{\mathcal{R}}$ is multiply recursive. Tightness of the bound follows by Lemma 3.3: for any multiply recursive function f, there exists a k such that $\mathsf{dc}_{\mathcal{R}_3^k}$ dominates f, and $\mathsf{dc}_{\mathcal{R}_3^k}$ terminates by Theorem 2.4. Moreover, the proof tree induced by the termination proof shown in Example 3.2 admits a constant reduction pair function. ◀

5 Conclusion and Future Work

In this paper we established that the derivational complexity of any TRS whose termination can be shown within the DP framework is bounded by a multiple recursive function, whenever the set of processors used is suitably restricted.

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. [3], Cichon and Lescanne [4], Hirokawa and the first author [11]. In [11] 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 [3], where it is augmented by an *average case* analysis. Finally [4] studies the complexity of the functions *computed* by a given TRS. This latter notion is extensively studied within *implicit computational complexity theory* (see [2] for an overview).

While we have presented our results for derivational complexity, it is easy to see that the same result holds for runtime complexity (as defined in [11]) and also for the complexity of the functions computed by the TRS (as suggested in [4]). For the former it suffices to observe that runtime complexity is a restriction of derivational complexity and that Example 3.2 provides a non-primitive recursive lower bound also in the context of runtime complexity. With respect to the second claim it suffices to observe that any function computed by a TRS that admits at most multiple recursive runtime complexity is computable on a Turing machine in multiple recursive time (compare also [1]). We assume that a similar result holds for the more involved notion proposed in [3]. However, this requires further work.

Thus our results indicate that the DP framework may induce multiple recursive complexity. This constitutes a first, but important, step towards the analysis of the complexity induced by the DP framework *in general*. Note that for all termination technique whose complexity has been analysed a multiple recursive upper bound exists. This leads us to the following conjecture.

▶ **Conjecture.** *Let $\mathcal{R}$ be a TRS whose termination can be proved with the DP framework using any DP processors, whose induced complexity does not exceed the class of multiple recursive functions. Then the derivational complexity with respect to $\mathcal{R}$ is multiple recursive.*

Should this conjecture be true, then for instance, none of the existing automated termination techniques would in theory be powerful enough to prove termination of Dershowitz's system `TRS/D33-33`, aka the Hydra battle rewrite system, whose complexity is not a provable recursive function of Peano Arithmetic (see [6, 14]). Hence far beyond multiple-recursion.

References

1 M. Avanzini and G. Moser. Closing the gap between runtime complexity and polytime computability. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 33–48, 2010.

2 P. Baillot, J.-Y. Marion, and S. R. D. Rocca. Guest editorial: Special issue on implicit computational complexity. *ACM Trans. Comput. Log.*, 10(4), 2009.

3 C. Choppy, S. Kaplan, and M. Soria. Complexity analysis of term-rewriting systems. *TCS*, 67(2–3):261–282, 1989.

4 E.-A. Cichon and P. Lescanne. Polynomial interpretations and the complexity of algorithms. In *Proc. 11th CADE*, volume 607 of *LNCS*, pages 139–147, 1992.

5 T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

6 N. Dershowitz and G. Moser. The Hydra battle revisited. In *Rewriting, Computation and Proof*, volume 4600 of *LNCS*, pages 1–27, 2007.

7 J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *JAR*, 40(3):195–220, 2008.

8 A. Geser. *Relative Termination*. PhD thesis, Universität Passau, 1990.

9 J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *JAR*, 37(3):155–203, 2006.

10 N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *IC*, 205:474–511, 2007.

11 N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. 4th IJCAR*, volume 5195 of *LNAI*, pages 364–380, 2008.

12 D. Hofbauer. *Termination Proofs and Derivation Lengths in Term Rewriting Systems*. PhD thesis, Technische Universität Berlin, 1992.

13 D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. 3rd RTA*, volume 355 of *LNCS*, pages 167–177, 1989.

14 G. Moser. The Hydra Battle and Cichon's Principle. *AAECC*, 20(2):133–158, 2009.

15 G. Moser and A. Schnabl. The derivational complexity induced by the dependency pair method. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 255–269, 2009.

16 G. Moser and A. Schnabl. The derivational complexity induced by the dependency pair method. *LMCS*, 2011. To appear. Available at `http://arxiv.org/abs/0904.0570`.

17 G. Moser and A. Schnabl. Termination proofs in the dependency pair framework may induce multiple recursive derivational complexity. *CoRR*, abs/1103.5082, 2011.

18 F. Neurauter, H. Zankl, and A. Middeldorp. Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In *Proc. 17th LPAR*, volume 6397 of *LNCS (ARCoSS)*, pages 550–564, 2010.

19 R. Péter. Zusammenhang der mehrfachen und transfiniten Rekursionen. *JSL*, 15(4):248–272, 1950.

20 R. Péter. *Recursive Functions*. Academic Press, 1967.

21 H. E. Rose. *Subrecursion - function and hierarchies*. Clarendon Press, 1984.

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

23 R. Thiemann. *The DP Framework for Proving Termination of Term Rewriting*. PhD thesis, University of Aachen, 2007.

24 J. Waldmann. Polynomially bounded matrix interpretations. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 357–372, 2010.

25 A. Weiermann. Termination proofs for term rewriting systems with lexicographic path orderings imply multiply recursive derivation lengths. *TCS*, 139(1,2):355–362, 1995.

26 H. Zankl and M. Korp. Modular complexity analysis via relative complexity. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 385–400, 2010.