Dependency Graphs, Relative Rule Removal, the Subterm Criterion and Derivational Complexity*

Georg Moser
University of Innsbruck
Innsbruck, Austria
georg.moser@uibk.ac.at

Andreas Schnabl
University of Innsbruck
Innsbruck, Austria
andreas.schnabl@uibk.ac.at

Abstract

We study the derivational complexity induced by the dependency pair method, enhanced with the dependency graph refinement and the subterm criterion, allowing relative removal of rules, for base orders inducing linear derivational complexity. If relative rule removal is allowed, we get a multiply recursive upper bound, otherwise the bound is primitive recursive.

1 Introduction

We assume familiarity with term rewriting (see [3, 2, 16]) and the dependency pair method (cf. [1, 6, 7, 8, 17]). Let R be a finite TRS over the set of terms $\mathrm{T}(F,V)$ built over the signature F and variables V. Let $D \subseteq F$ denote the set of defined symbols. We use $\mathsf{DP}(R)$ and $\mathsf{DG}(R)$ to denote the set of dependency pairs of R and the dependency graph of R, respectively. We use DP problems as defined in [6, 17], and we write (P,R) to abbreviate DP problems of the shape $(P,\emptyset,R,\mathbf{m})$ (in the notation of [6, 17]).

In this extended abstract, we extend our recent results from [13, 14]. We investigate the derivational complexity of TRSs whose termination is shown by a restricted set of proof methods. Recall the definition of the derivation height of a terminating term t with respect to a finitely branching rewrite relation $\to$, and the derivational complexity function of a terminating TRS R:

$$\mathsf{dh}(t,\to) = \max\{n \mid \exists t\ s \to^n t\} \qquad \mathsf{dc}_R(n) = \max\{\mathsf{dh}(t,\to_R) \mid |t| \leqslant n\}$$

We show upper bounds on the derivational complexity of TRSs whose termination is proved by the following, well-known results:

Proposition 1 ([1])**.** *R is terminating iff for each strongly connected component (SCC) P of $\mathsf{DG}(R)$, the DP problem (P,R) is finite.*

Proposition 2 ([6])**.** *A DP problem (P,R) is finite iff there exist $P' \subset P$ and a reduction pair $(\gtrsim,\succ)$ such that $P \setminus P' \subseteq \succ$, $P' \cup R \subseteq \gtrsim$, and the DP problem (P',R) is finite.*

Proposition 3 ([8, 17])**.** *A DP problem (P,R) is finite if there exist $P' \subset P$ and a simple projection π such that $\pi(l) \rhd \pi(r)$ for each dependency pair $l \to r$ in $P \setminus P'$, $\pi(l) \trianglerighteq \pi(r)$ for each dependency pair $l \to r$ in P', and the DP problem (P',R) is finite.*

To motivate this study, we give two examples that provide a lower-bound on the derivational complexity induced by Propositions 1 and 2, or Propositions 1 and 3.

Example 4. Consider the following TRS R, taken from [11]:

$$\mathsf{i}(x) \circ (y \circ z) \to x \circ (\mathsf{i}(\mathsf{i}(y)) \circ z)$$
$$\mathsf{i}(x) \circ (y \circ (z \circ w)) \to x \circ (z \circ (y \circ w))$$

It is shown in [11] that dc_R is not primitive recursive. As already stated in [4], termination can be easily proved as follows: first, we consider the reduction pair induced by the polynomial algebra A with $\circ_A^\sharp(x,y) = y$, $\circ_A(x,y) = y+1$ and $\mathsf{i}_A(x) = 0$, which allows us to remove three of the five dependency pairs of R by Proposition 2. Next, we apply the reduction pair induced by the polynomial algebra B with $\circ_B^\sharp(x,y) = x$, $\circ_B(x,y) = 0$, and $\mathsf{i}_B(x) = x+1$ and conclude termination of R.

*Partly supported by FWF (Austrian Science Fund) project P20133-N15.

We write $\succ/\gtrsim$ to abbreviate the relation $\gtrsim^* \cdot \succ \cdot \gtrsim^*$ (compare [5]). Note that the reduction pairs employed in Example 4 induce linear derivational complexity, that is, the function $\mathsf{dh}(t^\sharp, \succ/\gtrsim)$ grows only linar in $|t|$.

Example 5. Consider the following family of TRSs, denoted as $R(k)$, parametrised by k ($k \geqslant 2$), which encode the k-ary Ackermann function Ack_k.

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

Termination of $R(k)$ can be shown by k applications of Proposition 3 with $\pi_i(\mathsf{Ack}_k^\sharp) = i$, where π_i is the simple projection used in the ith application of Proposition 3.

These examples show that Propositions 1, 2 and 3 admit much higher derivational complexities than the basic dependency pair method, which (as shown in [13]) only admits a triple exponential upper bound (for reduction pairs which induce linear derivational complexity).

Still, we can show that the derivational complexity of TRSs whose termination is proved using these propositions is bounded by a multiply recursive function. In the next section, we give a sketch of this proof for the case that only Propositions 1 and 2 are applied, where we assume that the reduction pairs employed induce at most multiple-recursive complexity. Note, however that our approach can be extended to termination proofs which additionally employ Proposition 3.

2 Proof of the Upper Bound

We start by ordering the (trivial and nontrivial) SCCs of $\mathsf{DG}(R)$ by assigning a *rank* to each of them. Let P, Q denote two SCCs. We call Q *reachable* from P if there exist nodes $u \in P$, $v \in Q$ and a path in $\mathsf{DG}(R)$ from u to v. For the remainder of this paper, let k be the number of SCCs in $\mathsf{DG}(R)$, A the maximum arity of any function symbol occurring in R, and $C := \max\{2\} \cup \{\mathsf{dp}(r) \mid l \to r \in R\}$. Consider the set of all bijective mappings from the set of SCCs of $\mathsf{DG}(R)$ to $\{1,\ldots,k\}$ with the property that $\mathsf{rk}(P) > \mathsf{rk}(Q)$ whenever Q is reachable from P in $\mathsf{DG}(R)$. We fix an arbitrary one of these mappings to be rk. We call $\mathsf{rk}(P)$ the *rank* of an SCC P. The rank of a position p in a term u such that $(u|_p)^\sharp \notin \mathsf{NF}(P/R)$ for some SCC P is defined by $\mathsf{rk}(u,p) := \max\{\mathsf{rk}(s \to t) \mid \exists\sigma\ (u|_p)^\sharp \to_R^* s\sigma\}$, where the rank of a dependency pair $s \to t$, denoted by $\mathsf{rk}(s \to t)$, is the rank of P such that $s \to t \in P$.

In the following we write P_i for the unique SCC with rank i. Let ℓ be a natural number (the maximum number of applications of Proposition 2 for any SCC), and split each SCC P_i of $\mathsf{DG}(R)$ into ℓ (possibly empty) disjoint parts $P_{i,1},\ldots,P_{i,\ell}$. The intuition behind this split is that $P_{i,j}$ contains the dependency pairs removed by the jth application of Proposition 2 within SCC P_i. Finally let g be a monotone function over $\mathbb{N}$ with $g(0) > 0$ and

$$\mathsf{dh}(t^\sharp, \to_{P_{i,j}/R \cup \bigcup_{j'=j+1}^{\ell} P_{i,j'}}) \leqslant g(|t|)\,,$$

where $1 \leqslant i \leqslant k$, $1 \leqslant j \leqslant \ell$. Note that the function g can be inferred from the used reduction pairs.

In the sequel of this section, we show that dc_R is multiply recursive whenever the function g is multiply recursive. To this end, we make use of a simulating TRS R' (depending on k, A, C, and the function g). The idea behind R' is that it can simulate any derivation of R, and can thus be employed to

show that dc_R is multiply recursive. We define the mapping $\mathsf{dh}^\sharp\colon \mathrm{T}(F,V) \to \mathbb{N} \times \mathbb{N}^k$ as follows:

$$\mathsf{dh}^\sharp(t,p) = \begin{cases} (i, \mathsf{dh}(t^\sharp, \to_{(P_{i,1}/R \cup \bigcup_{j=2}^{\ell} P_{i,j})}), \ldots, \mathsf{dh}(t^\sharp, \to_{(P_{i,\ell}/R)})) & \text{if } t^\sharp \notin \mathsf{NF}(P_i/R) \wedge \mathsf{rk}(t,p) = i\,, \\ (0,\ldots,0,1) & \text{if } t^\sharp \in \mathsf{NF}(\mathsf{DP}(R)/R) \wedge \mathsf{rt}(t|_p) \in D\,, \\ (0,\ldots,0) & \text{otherwise}\,. \end{cases}$$

Note that, if $\mathsf{dh}^\sharp(t,p) = (i,i_1,\ldots,i_\ell)$ with $i > 0$, then $\max\{i_1,\ldots,i_\ell\} > 0$, as well. We give some intuition for this definition. Consider an arbitrary SCC P_i, a term t and a position p in t such that $\mathsf{rk}(t|_p) = i$. Suppose that $t|_p$ is not in normal form with respect to P_i/R. Then for once, we need to estimate $\mathsf{dh}(t^\sharp, \to_{(P_{i,j}/R \cup \bigcup_{j'=j+1}^{\ell} P_{i,j'})})$ for all $j \in \{1,\ldots,\ell\}$. On the other hand the case that $t|_p \in \mathsf{NF}(\mathsf{DP}(R)/R)$ has to be accounted for. For any rewrite step applied to t with redex position p, and any position p' "created" by that step we have $\mathsf{dh}^\sharp(t,p) >^{\mathrm{lex}} \mathsf{dh}^\sharp(t,p')$ in both cases.

The simulating TRS R' is based on a mapping tr such that $s \to_R t$ implies $\mathsf{tr}(s) \to_{R'}^+ \mathsf{tr}(t)$. Given a term t, tr essentially assigns to each position p of t an $A+\ell$-ary function symbol f_i. The index i of f_i and the first ℓ arguments are used to represent $\mathsf{dh}^\sharp(t,p)$.

Definition 6. *Let $t = f(t_1,\ldots,t_n)$ and $(i,i_1,\ldots,i_\ell) = \mathsf{dh}^\sharp(t^\sharp)$. Then the mapping $\mathsf{tr} : \mathrm{T}(R) \to \mathrm{T}(R')$ is defined as follows: $\mathsf{tr}(t) = \mathsf{f}_i(\mathsf{S}^{i_1}(0),\ldots,\mathsf{S}^{i_\ell}(0),\mathsf{tr}(t_1),\ldots,\mathsf{tr}(t_n),\mathsf{b},\ldots,\mathsf{b})$.*

Here we write $\mathsf{S}^n(0)$ as abbrevation for $\mathsf{S}(\cdots(\mathsf{S}(0)))$ (n times S). Note that if f is a constant, that is, $t = f$, then $\mathsf{tr}(t) = \mathsf{f}_i(\mathsf{S}^{i_1}(0),\ldots,\mathsf{S}^{i_\ell}(0),\mathsf{b},\ldots,\mathsf{b})$. To simplify the presentation, we compress sequences of b to $\overline{\mathsf{b}}$. Similarly, we use $\overline{x}$ for sequences of x.

The main tool for achieving the simulation of a rewrite step $s \to_R t$ are rules which create the progenies (as defined in [13]) of the redex position p of the step. The set of progenies of p contains at most A^{C+1} many elements, and each proper subterm of $t|_p$ may be duplicated at most that many times. In order to write down the right hand sides of these rules concisely for arbitrary A and C, we make use of some abbreviations. We define the shorthand $\vec{x}$ for $(x_1,\ldots,x_A)$. Moreover, we define $M_i^C(n_1,\ldots,n_\ell,\vec{x})$ as follows:

$$\begin{aligned} M_i^0(n_1,\ldots,n_\ell,\vec{x}) &:= \mathsf{f}_i(n_1,\ldots,n_\ell,\vec{x}) \\ M_i^{j+1}(n_1,\ldots,n_\ell,\vec{x}) &:= \mathsf{f}_i(n_1,\ldots,n_\ell,M_i^j(n_1,\ldots,n_\ell,\vec{x}),\ldots,M_i^j(n_1,\ldots,n_\ell,\vec{x})) \end{aligned}$$

We also define following abbreviation $X(\vec{x})$: $X(\vec{x}) = \mathsf{g}(\mathsf{size}(0,\ldots,0,\vec{x}))$.

Finally, we are in the position to present the rules of the simulating TRS R'.

Definition 7. *Consider the following (schematic) TRS R', where $i \in \{0,\ldots,k\}$, $i' \in \{1,\ldots,k\}$, $j \in \{1,\ldots,\ell\}$, and $j' \in \{1,\ldots,A\}$.*

$$\begin{aligned}
1_{i,j}\colon &\quad \mathsf{f}_i(n_1,\ldots,n_{j-1},\mathsf{S}(n_j),n_{j+1},\ldots,n_\ell,\vec{x}) \to M_i^C(n_1,\ldots,n_j,X(\vec{x}),\ldots,X(\vec{x}),\vec{x}) \\
2_{i'}\colon &\quad \mathsf{f}_{i'}(n_1,\ldots,n_\ell,\vec{x}) \to f_{i'-1}(X(\vec{x}),\ldots,X(\vec{x}),\vec{x}) \\
3_{i,j'}\colon &\quad \mathsf{size}(\mathsf{f}_i(n_1,\ldots,n_\ell,\vec{x})) \to \mathsf{d}_A(\mathsf{size}(x_{j'})) \\
4\colon &\quad \mathsf{size}(\mathsf{b}) \to \mathsf{S}(0) \\
5\colon &\quad \mathsf{d}_A(\mathsf{S}(x)) \to \mathsf{S}^A(\mathsf{d}_A(x))) \\
6\colon &\quad \mathsf{d}_A(0) \to 0 \\
7\colon &\quad \mathsf{f}_0(n_1,\ldots,n_\ell,\vec{x}) \to \mathsf{b} \\
8_{i,j'}\colon &\quad \mathsf{f}_i(n_1,\ldots,n_\ell,\vec{x}) \to x_{j'} \\
9\colon &\quad \mathsf{f}(x) \to \mathsf{f}_k(X(\overline{x}),\ldots,X(\overline{x}),\overline{x}) \\
10\colon &\quad \mathsf{z} \to \mathsf{f}_k(X(\overline{\mathsf{b}}),\ldots,X(\overline{\mathsf{b}}),\overline{\mathsf{b}})
\end{aligned}$$

These rules are augmented by rules defining the function symbol g*, that is, we choose some finite TRS* R'' *(employing disjoint defined function symbols with the exception of* g*) such that* S *and* $\mathsf{0}$ *are constructor symbols, and the unique normal form of* $\mathsf{g}(\mathsf{S}^n(\mathsf{0}))$ *is* $\mathsf{S}^{g(n)}(\mathsf{0})$ *(with respect to* R''*).*

Recall that it is not difficult to define a suitable TRS R'', whenever g is computable. Moreover, whenever g is a primitive recursive function, it is possible to give such a TRS R'', whose termination can be shown by the lexicographic path order (LPO).

We motivate the rules of R'. The rules $1_{i,j}$ are the main rules for the simulation of the effects of a single step $s \to_R t$ in R'. One rewrite step in s with redex position p creates at most A^{C+1} many new positions p' with $\mathsf{dh}^\sharp(t,p) >^{\text{lex}} \mathsf{dh}^\sharp(t,p')$, and the subterms of $t|_p$ are duplicated at most A^{C+1} times. Observe that a decrease in the jth argument of f_i may "reset" the value of arguments $j+1$ to ℓ, which explains the occurrence of the values $X(\vec{x})$ on the right hand side of these rules. The rules $2_{i'}$ simulate that each of the new positions q created by $s \to t$ might be of rank j ($i' > j$). The rules $3_{i,j'}$–6 define the function symbol size, that is, $\mathsf{size}(s)$ reduces to a numeral $\mathsf{S}^l(\mathsf{0})$ such that $l \geqslant |s|$. The rules 7–$8_{i,j'}$ make sure that any superfluous positions and copies of subterms created by the rules of type $1_{i,j}$ can be deleted. Finally, the rules 9 and 10 guarantee that the simulating derivation can be started with a small enough initial term (see also the second part of Lemma 8 below). Recall that the function $\mathsf{dc}_{R'}$ bounds the relationship between $|t|$ and $\mathsf{dh}(t,\to_{R'})$, and in general it is not the case that $|\mathsf{tr}(s)| \leqslant |s|$, hence we need the second part of Lemma 8.

Lemma 8. *For any ground terms* s *and* t*,* $s \to_R t$ *implies* $tr(s) \to_{R'}^+ tr(t)$*. Moreover, for any ground term* t*, we have* $\mathsf{f}^{|t|-1}(\mathsf{z}) \to_{R'}^+ tr(t)$*.*

Lemma 8 yields that the length of any derivation in R can be estimated by the maximal derivation length with respect to R'. It remains to verify whether R' has multiply recursive derivational complexity. Whenever g is a multiply recursive function (hence also whenever g is a linear function), it can be encoded by a set of LPO-orientable rules, and then R' can be oriented by LPO. Then by [19] $\mathsf{dc}_{R'}$ is bounded by a multiply recursive function.

Theorem 9. *Let* R *be a TRS whose termination is provable by Propositions 1 and 2. Moreover, assume that for any reduction pair* $(\gtrsim, \succ)$ *used in any application of Proposition 2, there exists a multiply recursive function* g *such that* $\mathsf{dh}(t, \succ/\gtrsim) \leqslant g(|t|)$ *for all terms* t*. Then* dc_R *is bounded by a multiply recursive function.*

For the special case that $\ell = 1$ (which means that in each SCC of $\mathsf{DG}(R)$, exactly one application of Proposition 2 is needed, which removes all dependency pairs in that SCC at once), the *general ramified lexicographic path order* (GRLPO), a restricted version of LPO which has been introduced by Weiermann in [18], can be used instead of LPO in the above argument. Then by [18], $\mathsf{dc}_{R'}$ is bounded by a primitive recursive function.

With some modifications, the considerations made in this paper can be extended to additionally allow the use of Proposition 3 in the termination proof of R. Our main result remains the same for this setting:

Theorem 10. *Let* R *be a TRS whose termination is provable by Propositions 1, 2 and 3. Moreover, assume that for any reduction pair* $(\gtrsim, \succ)$ *used in any application of Proposition 2, there exists a multiply recursive function* g *such that* $\mathsf{dh}(t, \succ/\gtrsim) \leqslant g(|t|)$ *for all terms* t*. Then* dc_R *is bounded by a multiply recursive function.*

This concludes that termination by the restricted version of the DP framework outlined by Propositions 1, 2 and 3 induces multiply recursive derivational complexity. This constitutes a first, but important, step towards the analysis of the (derivational) complexity induced by the DP Framework. In this context

it seems important to emphasise that currently no reduction pairs are known which induce non-multiply recursive complexities for finite TRSs.

This leads us to the conjecture that for termination proofs (within the DP framework) based on any of the currently known DP processors the induced derivational complexity of the initial TRS will be multiple-recursive. As a corollary to the conjecture we would obtain a proof that none of the techniques underlying current termination provers is in theory powerful enough to prove termination of Dershowitz's system `TRS/D33-33`, aka the Hydra battle rewrite system (see [3, 12]).

Future work will concentrate on this conjecture as well as an analysis of the DP framework from the point of view of runtime complexity analysis (see [9, 10, 15]).

References

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

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

[3] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 245–319. Elsevier Science, 1990.

[4] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(3):195–220, 2008.

[5] A. Geser. *Relative Termination*. PhD thesis, Universität Passau, 1990.

[6] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. 11th LPAR*, volume 3452 of *LNAI*, pages 301–331, 2005.

[7] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.

[8] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474–511, 2007.

[9] N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. 4th IJCAR*, volume 5195 of *LNCS*, pages 364–379, 2008.

[10] Nao Hirokawa and Georg Moser. Complexity, graphs, and the dependency pair method. In *Proc. of 15th LPAR*, volume 5330 of *LNCS*, pages 652–666, 2008.

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

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

[13] 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.

[14] G. Moser and A. Schnabl. The derivational complexity induced by the dependency pair method, 2010. Draft. Available at `http://cl-informatik.uibk.ac.at/users/aschnabl`.

[15] Georg Moser. Proof theory at work: Complexity analysis of term rewrite systems. *CoRR*, abs/0907.5527, 2009.

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

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

[18] A. Weiermann. A termination ordering for primitive recursive schemata, 1995. Preprint.

[19] A. 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.