On the Computational Content of Termination Proofs*

Georg Moser and Thomas Powell

Institute of Computer Science, University of Innsbruck, Austria
Email: {georg.moser,thomas.powell}@uibk.ac.at

Abstract. Given that a program has been shown to terminate using a particular proof, it is natural to ask what we can infer about its complexity. In this paper we outline a new approach to tackling this question in the context of term rewrite systems and recursive path orders. From an inductive proof that recursive path orders are well-founded, we extract an explicit realiser which bounds the derivational complexity of rewrite systems compatible with these orders. We demonstrate that by analysing our realiser we are able to derive, in a completely uniform manner, a number of results on the relationship between the strength of path orders and the bounds they induce on complexity.

1 Introduction

Proof theory emphasises *proofs* over *theorems*, as put most succinctly by Kreisel's famous question "What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?". One application of this quest in the context of program analysis is the link between termination and complexity. Is it possible to derive computational content from a given termination argument, so that we can automatically deduce bounds on the complexity of our programs?

We study this question in the abstract framework of term rewrite systems and recursive path orders, which we take to encompass multiset path orders, lexicographic path orders, and recursive path orders with status. Our main contribution is to analyse the proof that recursive path orders are well-founded and extract an explicit term in Gödel's system T which bounds the derivational complexity of rewrite systems reducing under these orders. Our framework is uniform in the sense that our term applies to any variant of recursive path order studied, by just adapting its parameters. We then demonstrate that a simple analysis of our term allows us to uniformly derive the well-known primitive recursive bounds on the derivational complexity of multiset path orders [1] (see Theorem 1) and the multiple recursive bounds on lexicographic path orders [2] (see Theorem 2).

The emphasis of this work is less on the technical results achieved, but in the method used to achieve them. Our re-derivation of the standard bounds for

* This work is supported by FWF (Austrian Science Fund) project P-25781.

multiset and lexicographic path orders contrasts greatly to the somewhat ad-hoc originally carried out by Hofbauer and Weiermann, and are much more closely related to the study of Buchholz [3], which forms the starting point of our work. However, whereas in [3] complexity bounds are obtained via a suitable formalisation of termination proofs in fragments of arithmetic and rely on Parson's fundamental work [4], our focus is on extracting an explicit subrecursive bound. Therefore, not only is our proof completely elementary and self-contained, but our concrete realising term is amenable to a much finer analysis of complexity for restricted path orders.

In addition to the aforementioned results, we obtain a novel derivational complexity analysis of recursive path orders with status [5], where we confirm that the induced complexity is multiple recursive, which follows as a corollary to our general boundedness result Theorem 2. This general bound is not surprising in the context of the well-known multiple-recursive bound on the lexicographic path orders and follows with relative ease from earlier work [6]. However, our smooth framework allows us to get rid of the technicalities involved in earlier work.

Throughout the history of term rewriting, a general link has been sought between the strength of a termination argument and the complexity of rewrite systems it admits. An early attempt at such a correspondence is the so-called *Cichon's principle*, which states that the derivational complexity function of a TRS $\mathcal{R}$ for which termination is provable using a termination order of order type α is eventually dominated by a function from the slow-growing hierarchy G_α along α, cf. [7]. Unfortunately, while this principle holds for the standard recursive path orders, it is not true in general, even for its relaxed version as proposed by Touzet [8] - see [9] for a proof. It is now accepted that the link between termination orders and complexity is dependent in a much more subtle way on the structure of the termination proof. Therefore, we believe that applying proof-theoretic techniques to analyse the computational meaning of path orders could provide some important insight into the relationship between termination and complexity. This approach has already been successfully pioneered in e.g. [10,11], and we hope that the work outlined here constitutes a first step towards a similarly successful program in the context of rewriting.

2 Recursive Path Orders

We assume familiarity with term rewriting [5,12], and recall only some basic notation. Let $\mathcal{V}$ denote a countable infinite set of variables, $\mathcal{F}$ a finite set of function symbols, and $\mathcal{T}(\mathcal{F},\mathcal{V})$ ($\mathcal{T}$ for short) the set of terms constructed from these. A *term rewrite system* (*TRS*) $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is a *finite* set of rewrite rules $l \to r$. For a given term t, $|t|$ denotes its size (the total number of variables and function symbols in t), $\mathsf{dp}(t)$ its depth (the maximal number of nesting function symbols) and $\mathsf{Var}(t)$ the set of variables in t. The rewrite relation is denoted as $\to_\mathcal{R}$ and we use the standard notations for its transitive and reflexive closure. The *derivation height* of a term s with respect to a well-founded, finitely branching

relation $\to$ is defined as: $\mathsf{dh}(s, \to) = \max\{n \mid \exists t\ s \to^n t\}$. The *derivational complexity function* $\mathsf{dc}_\mathcal{R}$ is defined as follows: $\mathsf{dc}_\mathcal{R}(n) := \max\{\mathsf{dh}(t, \to_\mathcal{R}) \mid |t| \leqslant n\}$.

Well-founded path orders are a powerful method for proving the termination of rewrite systems, and recursive path orders are one of the best known of these. For an arbitrary relation $>$ defined on some set X, we let $>_{\mathsf{mul}}$ and $>_{\mathsf{lex}}$ denote respectively the multiset and lexicographic extensions of $>$ to finite tuples X^n. We write $<_{\mathsf{mul}}$ for the reverse of $>_{\mathsf{mul}}$, and analogously for all other annotated inequality symbols used below.

Definition 1 (Recursive path order). *Let $\succ$ be a well-founded precedence (i.e. a proper order) on a finite signature $\mathcal{F}$. The* recursive path order (RPO) *$\succ_{\mathsf{rpo}}$ with respect to some status function $\tau\colon \mathcal{F} \to \{\mathsf{mul}, \mathsf{lex}\}$ is defined recursively as follows: we say that $t = f(t_1, \ldots, t_n) \succ_{\mathsf{rpo}} s$ if one of the following holds:*

(a) $t_i \succeq_{\mathsf{rpo}} s$ for some $i = 1, \ldots, m$;
(b) $s = g(s_1, \ldots, s_m)$ with $f \succ g$ and $t \succ_{\mathsf{rpo}} s_i$ for all $i = 1, \ldots, n$;
(c) $s = f(s_1, \ldots, s_n)$, $t \succ_{\mathsf{rpo}} s_i$ for all $i = 1, \ldots, n$ and $(t_1, \ldots, t_n) \succ_{\mathsf{rpo}, \tau(f)} (s_1, \ldots, s_n)$.

Here $\succeq_{\mathsf{rpo}}$ denotes the reflexive closure of $\succ_{\mathsf{rpo}}$.

Recall that the *multiset path order* $\succ_{\mathsf{mpo}}$ is the instance of RPO for which $\tau(f) = \mathsf{mul}$ for all f, and analogously for the *lexicographic path-order*. We say that a TRS $\mathcal{R}$ is compatible with $\succ_{\mathsf{rpo}}$ for some suitable choice of $\succ$ and τ if $\mathcal{R} \subseteq \succ_{\mathsf{rpo}}$. It is easy (but tedious) to show that $\succ_{\mathsf{rpo}}$ is closed under both substitutions and contexts, and therefore from compatibility we obtain that $\to_\mathcal{R} \subseteq \succ_{\mathsf{rpo}}$. In the latter case, we say that $\mathcal{R}$ is *reducing* with respect to $\succ_{\mathsf{rpo}}$. It is well-known that $\succ_{\mathsf{rpo}}$ is well-founded, so any TRS compatible with $\succ_{\mathsf{rpo}}$ is terminating.

There are two well-known basic strategies to show that a RPO is well-founded. One can either appeal to some variant of the minimal-bad-sequence argument to show that there cannot exist an infinite descending chain of terms $t_0 \succ_{\mathsf{rpo}} t_1 \succ_{\mathsf{rpo}} \ldots$, either in the form of Kruskal's theorem or directly applied to path orders as in [13], or alternatively one can take what is essentially the contrapositive of this statement and proceed via a series of nested inductions on terms, as in e.g. [3,14]. This second approach is most amenable for the purposes of program extraction, so we sketch the inductive proof below.

Theorem 1. *$\succ_{\mathsf{rpo}}$ is well-founded.*

Proof. Let $t \in \mathrm{WF}$ denote that t is a well-founded term, i.e. there are no infinite descending sequences starting from t. We prove that

$$\forall f \in \mathcal{F} \underbrace{\forall t_1, \ldots, t_n \in \mathrm{WF}\ .\ f(t_1, \ldots, t_n) \in \mathrm{WF}}_{A(f)}\ . \tag{1}$$

Then, since we trivially have $x \in \mathrm{WF}$ for all variables x, we obtain $\forall t(t \in \mathrm{WF})$ from (1) by well-founded induction over the structure of terms. Therefore it remains to prove $\forall f A(f)$. Let us fix, for now, $f \in \mathcal{F}$ and $t_1, \ldots, t_n \in \mathrm{WF}$ and make the following assumptions:

(A) $\forall g \prec fA(g)$
(B) $\forall(s_1,\ldots,s_n) \prec_{\mathsf{rpo},\tau(f)} (t_1,\ldots,t_n)(s_1,\ldots,s_n \in \mathrm{WF} \to f(s_1,\ldots,s_n) \in \mathrm{WF})$,

where we note that $\prec_{\mathsf{rpo},\tau(f)}$ is only ever applied to tuples of well-founded terms. We prove that $f(t_1,\ldots,t_n) \in \mathrm{WF}$ is well-founded by showing that

$$\forall s(\underbrace{s \prec_{\mathsf{rpo}} f(t_1,\ldots,t_n) \to s \in \mathrm{WF}}_{B(s)}) ,$$

using induction over $\blacktriangleleft$, where $\blacktriangleleft$ denotes the immediate subterm relation. Let us fix s and assume that $\forall s' \blacktriangleleft s\ B(s')$. Then if $t = f(t_1,\ldots,t_n) \succ_{\mathsf{rpo}} s$ there are three possibilities:

(a) $t_i \succeq_{\mathsf{rpo}} s$ for some $i = 1,\ldots,n$, in which case $s \in \mathrm{WF}$ by assumption that $t_i \in \mathrm{WF}$;
(b) $s = g(s_1,\ldots,s_m)$ with $f \succ g$ and $t \succ_{\mathsf{rpo}} s_i$ for all i. Then by our induction hypothesis we must have $s_1,\ldots,s_m \in \mathrm{WF}$, and therefore by assumption (A) we have $g(s_1,\ldots,s_m) \in \mathrm{WF}$ too;
(c) $s = f(s_1,\ldots,s_n)$ with $t \succ_{\mathsf{rpo}} s_i$ for all i and $(t_1,\ldots,t_n) \succ_{\mathsf{rpo},\tau(f)} (s_1,\ldots,s_n)$. Then again $s_1,\ldots,s_n \in \mathrm{WF}$, and this time by assumption (B) we have $f(s_1,\ldots,s_n) \in \mathrm{WF}$.

This establishes $B(s)$, and thus by $\blacktriangleleft$-induction we obtain $\forall s B(s)$ and hence well-foundedness of $f(t_1,\ldots,t_n)$. We now carry out two further inductions to eliminate the assumptions (A) and (B) in turn. First, from (A) $\to$ (B) $\to$ $(t_1,\ldots,t_n \in \mathrm{WF} \to f(t_1,\ldots,t_n) \in \mathrm{WF})$ and well-founded induction over $(\mathrm{WF}, \succ_{\mathsf{rpo},\tau(f)})$ we obtain (A) $\to A(f)$, and this yields $\forall f A(f)$ by induction on $(\mathcal{F},\succ)$, and we're done. □

2.1 A Finitary Formulation of Theorem 1

In general, we know that an arbitrary rewrite system $\mathcal{R}$ compatible with some $\succ_{\mathsf{rpo}}$ must terminate by well-foundedness of $\succ_{\mathsf{rpo}}$. However, for a fixed rewrite system, the full strength of Theorem 1 is never used, since unlike $\succ_{\mathsf{rpo}}$ the rewrite relation $\to_{\mathcal{R}}$ is only finitely branching. Rather, $\mathcal{R}$ will always lie in some *finitary approximation* of $\succ_{\mathsf{rpo}}$, where the size of this approximation will depend in some suitable sense on the 'size' of $\mathcal{R}$. Thus, in order to successfully analyse the complexity of the termination proof, we are not interested in analysing the well-foundedness of $\succ_{\mathsf{rpo}}$ itself, but only these finitary approximations to it.

A precise characterisation of the approximation of $\succ_{\mathsf{rpo}}$ needed to prove well-foundedness of a given TRS is established by Buchholz in [3], and we reformulate his idea below in a slightly simplified way (the simplification being possible because here we do not consider varyadic function symbols).

In what follows, we use the abbreviation $\mathsf{RPO}(>,t,s)$ for the statement that one of the conditions (a)-(c) in Definition 1 holds for s, t and $>$. Thus one defines $\succ_{\mathsf{rpo}}$ by $f(t_1,\ldots,t_n) \succ_{\mathsf{rpo}} s$ iff $\mathsf{RPO}(\succ_{\mathsf{rpo}}, f(t_1,\ldots,t_n), s)$.

Definition 2. *The approximation $\succ_k$ of $\succ_{\mathsf{rpo}}$ is recursively defined as follows: we have $t = f(t_1, \ldots, t_n) \succ_k s$ iff*

$$\mathsf{RPO}(\succ_k, f(t_1, \ldots, t_n), s) \wedge \mathsf{dp}(s) \leqslant k + \mathsf{dp}(t) \wedge \mathsf{Var}(s) \subseteq \mathsf{Var}(t) \ ,$$

where $\mathsf{dp}(t)$ denotes the depth of t.

We call $\succ_k$ finitary because by definition for each t there are only finitely many s for which $t \succ_k s$. The proof of the next theorem can essentially be read-off from Buchholz's proof in [3]. However, our later proof extraction is based on the simplified proof given here.

Theorem 2 (Buchholz [3]). *Any $\mathcal{R}$ compatible with an $\succ_{\mathsf{rpo}}$ is contained in $\succ_k$ for some k depending on $\mathcal{R}$.*

Proof. It is first shown that

(i) $t \succ_{\mathsf{rpo}} s$ implies $t\sigma \succ_{\mathsf{dp}(s)} s\sigma$ for any substitution σ,
(ii) $t_j \succ_k s$ implies $f(t_1, \ldots, t_n) \succ_k f(t_1, \ldots, t_{j-1}, s, t_{j+1}, \ldots, t_n)$ for any f.

Property (i) is most easily established as in [3]; as usual $t \succ_{\mathsf{rpo}} s$ implies that $\mathsf{Var}(s\sigma) \subseteq \mathsf{Var}(t\sigma)$. Furthermore, induction on $\succ_{\mathsf{rpo}}$ yields that $t \succ_{\mathsf{rpo}} s$ implies $\mathsf{dp}(s\sigma) \leqslant \mathsf{dp}(s) + \mathsf{dp}(t\sigma)$. Yet another induction over $\succ_{\mathsf{rpo}}$ derives (i), and property (ii) is similarly straightforward. Now, for a given $\mathcal{R}$ let $k := \max\{\mathsf{dp}(r) : l \to r \in \mathcal{R}\}$. It is then clear that if $\mathcal{R}$ is compatible with $\succ_{\mathsf{rpo}}$ then $t \to_{\mathcal{R}} s$ implies $t \succ_k s$, since by (i) we have $l\sigma \succ_k r\sigma$ for all rules $l \to r$, and therefore $C[l\sigma] \succ_k C[r\sigma]$ by induction on (ii). □

3 Bounding the Derivational Complexity of $\mathcal{R}$

We now construct a term which forms a recursive analogue to Theorem 1, but takes into account the fact that we only need to consider finitary approximations. Let $\mathcal{R}$, a RPO $\succ_{\mathsf{rpo}}$ compatible with $\mathcal{R}$ and a suitable approximation $\succ_k$ of $\succ_{\mathsf{rpo}}$ be fixed for the remainder of the paper.

3.1 Term and Tree Encodings, Gödel's system T

We assume that the terms $\mathcal{T}$ of our rewrite system can be primitive recursively encoded into $\mathbb{N}$, and write $s <_{\mathcal{T}} t$ to denote that the code of s is less than the code of t. Let $\mathcal{T}^\star$ denote the set of all finite trees of terms. For $T \in \mathcal{T}^\star$ we write $\mathsf{rt}(T)$ to denote the root of T, and write $S \subset T$ if S is an immediate subtree of T. Again, we assume that $\mathcal{T}^\star$ has been primitive recursively encoded into $\mathbb{N}$.

In what follows we work in the standard language of system T in all finite types ρ: terms are built from the usual arithmetic constants, λ-abstraction and application, and Gödel's primitive recursor $\mathsf{R}^h_\rho(n) = hn(\lambda m < n \,.\, \mathsf{R}^h_\rho(m))$ whose output can have arbitrary type ρ. It is clear that recursion over the decidable relations $\blacktriangleleft$ and $\subset$ can be defined in terms of the recursor of base type $\mathbb{N}$, since

without loss of generality we can assume that if $s \blacktriangleleft t$ then $s <_{\mathcal{T}} t$, and similarly for $\subset$. Therefore terms build up from these forms of recursion are primitive recursive in the usual sense. On the other hand, given a well-founded lifting $\subset_*$ of $\subset$ to tuples $(\mathcal{T}^\star)^n$ (where in the sequel $\subset_*$ will be either the multiset or lexicographic lifting) we let $\mathsf{TR}_{\subset_*}$ denote the transfinite recursor over $\subset_*$ of base output type. We leave open for now how this can be formally defined within system T as this will depend on the lifting.

3.2 Computing Derivation Trees

Let t be a term and let $\Phi_k(t)$ denote the finite tree T with root t, whose branches $(t, t_1, \ldots, t_n)$ are precisely $\succ_k$-derivations $t \succ_k t_1 \succ_k \ldots \succ_k t_n$ from t; terms will be denoted by lower-case letters and derivation trees by upper-case letters. Finiteness of $\Phi_k(t)$ follows since $\succ_k$ is well-founded and finitely branching. We now show how $\Phi_k(t)$ can be computed. Let $t = f(t_1, \ldots, t_n)$ for some $f \in \mathcal{F}$ and terms $t_1, \ldots, t_n$, and suppose that for each $g \prec f$ we have a function $F_g \colon (\mathcal{T}^\star)^m \to \mathcal{T}^\star$ where $m = \mathsf{ar}(g)$ that satisfies

$$F_g(\Phi_k(s_1), \ldots, \Phi_k(s_m)) = \Phi_k(g(s_1, \ldots, s_m)) \text{ for all } s_1, \ldots, s_m \,. \qquad \text{(A)}$$

Suppose, in addition, that we have a function $G_{t_1,\ldots,t_n} \colon (\mathcal{T}^\star)^n \to \mathcal{T}^\star$ satisfying

$$G_{t_1,\ldots,t_n}(\Phi_k(s_1), \ldots, \Phi_k(s_n)) = \Phi_k(f(s_1, \ldots, s_n)) \text{ for all } \boldsymbol{s} \prec_{k,\tau(f)} \boldsymbol{t} \,. \qquad \text{(B)}$$

Here $\prec_{k,\tau(f)}$ abbreviates the $\tau(f)$-extension of the approximation $\prec_k$ and $\boldsymbol{t} = t_1, \ldots, t_n$.

Lemma 1. *Given $T_1, \ldots, T_n \in \mathcal{T}^\star$ define the function $H^{F_{g\prec f}, G_{\boldsymbol{t}}, T_1, \ldots, T_n} \colon \mathcal{T} \to \mathcal{T}^\star$ (where $F_{g\prec f}, G_{\boldsymbol{t}}, T_1, \ldots, T_n$ are treated as parameters) by subterm recursion as follows (suppressing parameters):*

$$H(s) := \begin{cases} T_i[s] & \text{for the least } i \text{ such that } s \text{ is equal to either } \mathsf{rt}(T_i) \text{ or some child of } \mathsf{rt}(T_i) \text{ in } T_i, \text{ if such an } i \text{ exists} \\ F_g(H(s_1), \ldots, H(s_m)) & \text{if } s = g(s_1, \ldots, s_m) \text{ for } g \prec f \\ G_{\boldsymbol{t}}(H(s_1), \ldots, H(s_n)) & \text{if } s = f(s_1, \ldots, s_n) \\ [] & \text{otherwise ,} \end{cases}$$

where $[]$ denotes the empty tree and $T[s]$ the subtree of T with root s. Then

$$H^{F_{g\prec f}, G_{\boldsymbol{t}}, \Phi_k(t_1), \ldots, \Phi_k(t_n)}(s) = \Phi_k(s) \,,$$

for all $s \prec_k t$, assuming (A) *and* (B).

Proof. By induction on $\blacktriangleleft$. If $s \prec_k t$ there are three possibilities. First, if $s \preceq_k t_i$ for some i then either $s = t_i$ or s is a child of t_i in $\Phi_k(t_i)$, and so $H(s) =$

$\Phi_k(t_i)[s] = \Phi_k(s)$. Otherwise, suppose $s = g(s_1, \ldots, s_m)$ for $g \prec f$ and $s_i \prec_k t$ for all i, by the induction hypothesis we obtain $H(s_i) = \Phi_k(s_i)$ and therefore

$$H(s) = F_g(\Phi_k(s_1), \ldots, \Phi_k(s_m)) = \Phi_k(g(s_1, \ldots, s_m)) \ ,$$

by assumption (A). Similarly, if $s = f(s_1, \ldots, s_n)$, where for all i, $s_i \prec_k t$ and $(s_1, \ldots, s_n) \prec_{k,\tau(f)} (t_1, \ldots, t_n)$, then the induction hypothesis together with (B) yields

$$H(s) = G_{\boldsymbol{t}}(\Phi_k(s_1), \ldots, \Phi_k(s_n)) = \Phi_k(f(s_1, \ldots, s_n)) \ .$$

From this the lemma follows. □

Let t be a term and let $(S_i)_{i \in I}$ be a finite collection of trees. Then the tree $t * \prod_{i \in I} S_i$ is the finite tree with root t and immediate subtrees S_i.

Lemma 2. *Define the function* $K^{F_{g \prec f}, G_{\boldsymbol{t}}} \colon (\mathcal{T}^\star)^n \to \mathcal{T}^\star$ *as follows:*

$$K^{F_{g \prec f}, G_{\boldsymbol{t}}}(T_1, \ldots, T_n) := t' * \prod_{s \prec_k t'} H^{F_{g \prec f}, G_{\boldsymbol{t}}, T_1, \ldots, T_n}(s) \ ,$$

where $t' = f(\mathsf{rt}(T_1), \ldots, \mathsf{rt}(T_n))$. *Then assuming* (A) *and* (B) *we have*

$$K^{F_{g \prec f}, G_{\boldsymbol{t}}}(\Phi_k(t_1), \ldots, \Phi_k(t_n)) = \Phi_k(t) \ .$$

Proof. First, we remark that $K^{F_{g \prec f}, G_{\boldsymbol{t}}}$ is primitive recursive in $F_{g \prec f}$ and $G_{\boldsymbol{t}}$, since $s \prec_k t$ is a primitive recursive predicate, and $\prod_{s \prec_k t}$ a finite search bounded by some primitive recursive term, by definition of $\prec_k$. Now, due to Lemma 1 we have

$$\begin{aligned} K^{F_{g \prec f}, G_{\boldsymbol{t}}}(\Phi_k(t_1), \ldots, \Phi_k(t_n)) &= t * \prod_{s \prec_k t} H^{F_{g \prec f}, G_{\boldsymbol{t}}, \Phi_k(t_1), \ldots, \Phi_k(t_n)}(s) \\ &= t * \prod_{s \prec_k t} \Phi_k(s) \ , \end{aligned}$$

and this is just the tree whose branches are $\succ_k$-derivations $t \succ_k s \succ_k \ldots \succ_k s_n$, which is exactly $\Phi_k(t)$. □

Lemma 3. *Define function* $F_f^{F_{g \prec f}} \colon (\mathcal{T}^\star)^n \to \mathcal{T}^\star$ *using the transfinite recursor over* $\subset_{\tau(f)}$ *as follows*

$$F_f^{F_{g \prec f}}(T_1, \ldots, T_n) := K^{F_{g \prec f}, F_f \restriction_{(S_1, \ldots, S_n) \subset_{\tau(f)} (T_1, \ldots, T_n)}}(T_1, \ldots, T_n) \ .$$

Then assuming (A)*, for all* $t_1, \ldots, t_n$ *we have*

$$F_f^{F_{g \prec f}}(\Phi_k(t_1), \ldots, \Phi_k(t_n)) = \Phi_k(f(t_1, \ldots, t_n)) \ .$$

Proof. By induction on $\prec_{k,\tau(f)}$; suppose for all $(s_1, \ldots, s_n) \prec_{k,\tau(f)} (t_1, \ldots, t_n)$ the lemma is true. Then (B) holds for $G_{t_1, \ldots, t_n} = F_f \restriction_{(S_1, \ldots, S_n) \subset_{\tau(f)} (\Phi(t_1), \ldots, \Phi(t_n))}$ and by Lemma 2 we have

$$F_f(\Phi_k(t_1), \ldots, \Phi_k(t_n)) = \Phi_k(f(t_1, \ldots, t_n)) \ .$$

This completes the induction step, so the result holds for all arguments $t_1, \ldots, t_n$. □

Lemma 4. *For each $f \in \mathcal{F}$ there exists a function $F_f \colon (\mathcal{T}^\star)^n \to \mathcal{T}^\star$ for $n = \mathsf{ar}(f)$ satisfying*

$$F_f(\Phi_k(t_1), \ldots, \Phi_k(t_n)) = \Phi_k(f(t_1, \ldots, t_n)) \ .$$

Proof. By Lemma 3 we construct F_f in terms of F_g for $g \prec f$ assuming (A), and since $\prec$ is well-founded this construction is well-defined and correct for all f. □

Theorem 3. *There exists a function $F \colon \mathcal{T} \to \mathcal{T}^\star$ primitive recursive in $\mathsf{TR}_{\subset_{\tau(f)}}$ for $f \in \mathcal{F}$ such that*

$$F(t) = \Phi_k(t) \ ,$$

for all terms t.

Proof. Define F using subterm recursion as

$$F(t) := \begin{cases} [x] & \text{if } t = x \\ F_f(F(t_1), \ldots, F(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \ . \end{cases}$$

Then theorem follows by Lemma 4 and induction over the structure of t. □

3.3 Derivational Complexity

Let $|\cdot| \colon \mathcal{T}^\star \to \mathbb{N}$ denote the recursive function which returns the length of the longest branch of trees in $\mathcal{T}^\star$.

Theorem 4. *Suppose that the TRS $\mathcal{R}$ is compatible with RPO for some suitable status function τ. Then its derivational complexity is bounded by a function primitive recursive in $\mathsf{TR}_{\subset_{\tau(f)}}$ for $f \in \mathcal{F}$.*

Proof. By Theorem 2, if $\mathcal{R}$ is compatible with RPO then it is compatible with $\succ_k$ for some k, and so by Theorem 3 $\to_{\mathcal{R}}$ derivations from t are contained in the tree $F(t)$, where F is primitive recursive in the $\mathsf{TR}_{\subset_{\tau(f)}}$. In particular, $\mathsf{dh}(t, \to_{\mathcal{R}}) \leqslant |F(t)|$, and therefore

$$\mathsf{dc}_{\mathcal{R}}(n) \leqslant \max_{|t| \leqslant n} |F(t)|$$

which is primitive recursive in F since we can bound the search $|t| \leq n$ because we only need to search over a finite number of variables. □

We can now re-derive, in a completely uniform way, some of the well-known complexity results concerning recursive path orders. To do this, first let $\mathsf{TR}_{\mathsf{mul}(n)}$ denote multiset recursion of lowest type over tuples $(x_1, \ldots, x_n) \colon \mathbb{N}^n$ of size n, and $\mathsf{TR}_{\mathsf{lex}(n)}$ lexicographic recursion. Then we have the following.

Lemma 5. *(a) $\mathsf{TR}_{\mathsf{mul}(n)}$ is definable from the Gödel recursor of lowest type; (b) $\mathsf{TR}_{\mathsf{lex}(n)}$ is definable from the Gödel type 1 recursor.*

Proof. Part (a) is straightforward, as one can easily find a (primitive recursive) encoding of $\mathbb{N}^n$ into $\mathbb{N}$ that preserves the multiset order.

For (b) we use induction on n. It's clear that $\mathsf{TR}_{\mathsf{lex}(1)}$ is just primitive recursion in the usual sense. Now, assuming that $\mathsf{TR}_{\mathsf{lex}(n-1)}$ has been defined, use this to construct the functional $h^H \colon \mathbb{N} \to (\mathbb{N} \to \mathbb{N}^{n-1} \to \mathbb{N}) \to \mathbb{N}^{n-1} \to \mathbb{N}$, parametrised by $H \colon \mathbb{N} \to \mathbb{N}^{n-1} \to (\mathbb{N} \to \mathbb{N}^{n-1} \to \mathbb{N}) \to \mathbb{N}$, and defined by

$$h^H x F \boldsymbol{x} := H x \boldsymbol{x} \left(\lambda y, \boldsymbol{y} \,.\, \begin{cases} h^H x F \boldsymbol{y} & \text{if } y = x \wedge \boldsymbol{y} <_{\mathsf{lex}(n-1)} \boldsymbol{x} \\ F y \boldsymbol{y} & \text{if } y < x \end{cases} \right) .$$

Now by unwinding definitions we see that the term $\mathsf{R}_1^h \colon \mathbb{N} \to \mathbb{N}^{n-1} \to \mathbb{N}$ satisfies

$$\mathsf{R}^h(x_1)(x_2, \ldots, x_n) = H(x_1)(x_2, \ldots, x_n)(\lambda \boldsymbol{y} <_{\mathsf{lex}(n)} \boldsymbol{x} \,.\, \mathsf{R}^h(y_1)(y_2, \ldots, y_n)) \,,$$

where now $\boldsymbol{x} = (x_1, \ldots, x_n)$. But this is just recursion over $\mathsf{lex}(n)$. □

Corollary 1 (Hofbauer [1]). *If $\mathcal{R}$ is compatible with MPO then $\mathcal{R}$ has primitive recursive derivational complexity.*

Proof. This follows from Lemma 5 (a) and the observation that $\mathsf{TR}_{\subset_{\mathsf{mul}}}$ is definable from $\mathsf{TR}_{\mathsf{mul}}$ since $(\mathcal{T}^\star, \subset)$ can be recursively encoded in $(\mathbb{N}, <)$. □

Corollary 2 (Weiermann [2]). *If $\mathcal{R}$ is compatible with RPO then $\mathcal{R}$ has multiply recursive derivational complexity.*

Proof. This follows analogously to Corollary 1, this time using Lemma 5 (b). The fact that type one functions definable from the Gödel level 1 recursor are multiply recursive is folklore, see e.g. [15]. □

4 Conclusion

The most important feature of our work is not the rederivation of known complexity bounds, but in the manner in which we were able to do this. By constructing a concrete realising term F as a computational analogue to Theorem 1 which computes finitary $\succ_k$-derivation trees, we provided a bridge which relates the proof-theoretic complexity of well-founded recursive path orders to the derivational complexity of rewrite systems compatible with these orders.

A crucial point that we want to explore in future work is that our realising term is uniformly dependent on the parameters of the recursive path order used to prove termination, along with the size of the rewrite system, and any restriction in these parameters will cause a corresponding restriction in the complexity of F. Therefore a further, more detailed analysis of the structure of the realiser should enable us to obtain more refined complexity bounds.

For example, it follows from Weiermann's original derivational complexity analysis of the lexicographic path order that the induced multiple recursive bound allows parametrisation in the maximal arity of the function symbols, cf. [2], see also [16, Chapter 8]. Similar results follow from Hofbauer's analysis

of the multiset path order, cf. [1]. We expect that these and similar finer characterisations of the derivational complexity induced by specific parameters of the recursive path orders can be obtained with relative ease in our context. More generally, we hope to extend these results and in particular derive new criteria on path orders which guarantee feasible complexity of rewrite systems.

As another example, one could study restricted variants of the lexicographic lifting on tuples which do not require type 1 recursion to define the corresponding recursor, giving us strengthenings of the multiset path order which allow us to prove interesting closure properties for the primitive recursive functions, an idea initiated by Cichon and Weiermann in [17].

References

1. Hofbauer, D.: Termination proofs by multiset path orderings imply primitive recursive derivation lengths. TCS **105** (1992) 129–140
2. Weiermann, A.: Termination proofs for term rewriting systems with lexicographic path ordering imply multiply recursive derivation lengths. TCS **139** (1995) 355–362
3. Buchholz, W.: Proof-Theoretic Analysis of Termination Proofs. APAL **75** (1995) 57–65
4. Parsons, C.: On a number theoretic choice schema and its relation to induction. In: Proc. Intuitionism and Proof Theory. (1970) 459–473
5. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
6. Moser, G., Weiermann, A.: Relating derivation lengths with the slow-growing hierarchy directly. In: Proc. 14th RTA. Volume 2706 of LNCS. (2003) 296–310
7. Cichon, E.A.: Termination orderings and complexity characterisations. In: Proof Theory, Cambridge University Press (1992) 171–193
8. Touzet, H.: Encoding the Hydra battle as a rewrite system. In: Proc. 23rd MFCS. Volume 1450 of LNCS. (1998) 267–276
9. Moser, G.: KBOs, Ordinals, Subrecursive Hierarchies and All That. JLC (2015) to appear.
10. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. In: Proc. 26th LICS, IEEE (2011) 269–278
11. Berardi, S., Oliva, P., Steila, S.: Proving Termination with Transition Invariants of Height ω. In: Proc. 15th ICTCS. (2014) 237–240
12. TeReSe: Term Rewriting Systems. Volume 55 of Cambridge Tracks in Theoretical Computer Science. Cambridge University Press (2003)
13. Ferreira, M.C.F., Zantema, H.: Well-Foundedness of Term Orderings. In: Proc. 4th CTRS. Volume 968 of LNCS. (1995) 106–123
14. Goubault-Larrecq, J.: Well-Founded Recursive Relations. In: Proc. 15th CSL. Volume 2142 of LNCS. (2001) 484–498
15. Weiermann, A.: How is it that infinitary methods can be applied to finitary mathematics? Gödel's T: A case study. JSL **63** (1998) 1348–1370
16. T-Arai: Some results on cut-elimination, provable well-orderings, induction, and reflection. APAL (1998) 93–184
17. Cichon, E.A., Weiermann, A.: Term Rewriting Theory for the Primitive Recursive Functions. APAL **83** (1997) 199–223