# Tiered Recursion and Strategies

Toshiyasu Arai* Georg Moser†

**Abstract**

The purpose of this paper is to stress the need for strategies in the framework of term-rewriting analysis of complexity classes. Independently our results can be conceived as a simplification and clarification of the results obtained in [6]. In particular, we introduce a new path order $\prec_{\text{POP}}$ for polynomial time computable functions, such that for a finite rewrite system $R$ compatible with $\prec_{\text{POP}}$, the *complexity* or *derivation length function* $\text{Dl}_R(n)$ is guaranteed to be bounded by a polynomial in $n$.

## 1 Introduction

The purpose of this paper is to stress the need for strategies in the framework of term-rewriting analysis of complexity classes. Our primary interest are term-rewriting characterisations of complexity classes, like the class of polytime computable functions $\mathcal{F}_{\text{PTIME}}$ or the class of functions, computable in polynomial space, $\mathcal{F}_{\text{PSPACE}}$. Rewrite systems are a powerful computational model and the (successful) analysis of a rewrite system $R$—corresponding to a complexity class $\mathcal{C}$—allows fruitful insight into the structure of $\mathcal{C}$. Compare, for example [6, 16, 5, 9, 8].

For *large* inductively defined classes $\mathcal{C}$ of recursive function, like the primitive recursive functions, or the multiple recursive functions, the *corresponding rewrite system* $R$ can be canonically defined, such that $R$ is feasible and directly allows a successful analysis of the function class $\mathcal{C}$. This canonical correspondence is lost, when studying low-complexity classes, like $\mathcal{F}_{\text{PTIME}}$. This was first observed by Beckmann and Weiermann, who realised that the seemingly natural term-rewriting analogue $R_B$ of a representation of the polytime functions is *non-feasible*, compare Section 3. To remedy this situation they introduced a clever term-rewriting characterisation of $\mathcal{F}_{\text{PTIME}}$, by defining a set $R_{B'}$ of feasible rewrite rules. Essentially, the new rewrite system $R_{B'}$ forces the rewrite relation $R_B$ into a specific *strategy*. For further details, see below.

The results presented below, may be conceived as a simplification and clarification of the results obtained in [6]. In particular, we introduce a new path order

*Graduate School of Science and Technology, Kobe University, arai@kurt.scitec.kobe-u.ac.jp

†Computational Logic, University of Innsbruck, georg.moser@uibk.ac.at

$\prec_{\text{POP}}$ for the polynomial time computable functions $\mathcal{F}_{\text{PTIME}}$, such that for a finite rewrite system $R$ compatible with $\prec_{\text{POP}}$, the *complexity* or *derivation length function* $\text{Dl}_R(n)$ is guaranteed to be bounded by a polynomial in $n$. The latter order is obtained by a stringent comparison of the needed concepts with earlier proof-theoretic considerations on termination orders, especially influential was the work by Buchholz [10]. Future work will deal with a comparison of $\prec_{\text{POP}}$ with other versions of syntactic restrictions of path orders, compare [12, 15].

## 2 Term-rewriting Characterisation

We start with a (well-known) connection between function classes, defined by (sets of) equations and term rewriting systems. Assume $\mathcal{C}$ denotes an inductively defined class of recursive number-theoretic functions and suppose each $f \in \mathcal{C}$ is defined via an equation of the form

$$f(\vec{x}) = t(\lambda\vec{y}.f(\vec{y}), \vec{x}) \ , \tag{1}$$

where $t$ may involve some previously defined functions from $\mathcal{C}$. The *corresponding rewrite system* $R$ is defined as follows. The underlying signature $\Sigma$ includes for each function in $\mathcal{C}$ a corresponding function symbol.[1] To represent natural numbers we assume that $\Sigma$ includes a constant 0 and a unary function symbol $S$. Then, *numerals* are defined as usual. (Later we will represent numerals in the form of dyadic strings.) For each function symbol $f \in \mathcal{C} - \{0, S\}$, the rule

$$f(\vec{x}) \to t(\lambda\vec{y}.f(\vec{y}), \vec{x}) \ .$$

is contained in $R$. In all non-pathological cases the corresponding rewrite system is terminating and ground-confluent.

Let $\mathcal{V}$ denote a countably infinite set of variables. As usual the term algebra over $\Sigma$ and $\mathcal{V}$ is denoted as $\mathcal{T}$, while the ground term algebra is written as $\mathcal{T}(\Sigma)$. The set of finite sequences of terms is denoted as $\mathcal{T}^*$. Every term $t \in \mathcal{T}$ is identified with the one element sequent $(t) \in \mathcal{T}^*$. Thus $\mathcal{T} \subseteq \mathcal{T}^*$.

The rewrite relation induced by a rewrite system $R$ is denoted as $\to_R$, while we write $\to_R^*$ to denote the transitive and reflexive closure of $\to_R$. (Compare [4] for background information on term rewriting.) Furthermore we use the following conventions. Let $k, l, k', l', r$ denote arbitrary natural numbers, respectively. The symbols $\overline{n}, \overline{m}$ are used to denote numerals. Terms are denoted by $r, s, t$, possibly extended by subscripts. We write $\mathbf{t}$, to denote sequences of terms $t_1, \ldots, t_k \in \mathcal{T}$ and $\mathbf{g}$ to denote sequences of function symbols $g_1, \ldots, g_k$, respectively. Further, we write $\mathbf{n}$ for a sequence of numerals $\overline{n}_1, \ldots, \overline{n}_k$. Sometimes it is useful to denote sequences of terms by lower-case Greek symbols.

We illustrate the approach with respect to the class of primitive recursive functions PRIM.

[1] To keep the notation simple, we denote the *function* $f \in \mathcal{C}$ and the *function symbol* $f \in \Sigma$ by the same symbol. No confusion will arise from this.

**Example 2.1.** The class of primitive recursive functions PRIM, is inductively defined by the equation schemata given in Table 1.

Table 1: Primitive Recursion

| | |
|---|---|
| $O^n(\mathbf{x}) = 0$ , | **[zero]** |
| $S(x) = x + 1$ , | **[successor]** |
| $P(\mathbf{x}) = x_i$ , | **[projection]** |
| $\mathrm{SUB}[f, \mathbf{g}](\mathbf{x}) = f(\mathbf{g}(\mathbf{x}))$ , | **[composition]** |
| $\mathrm{PREC}[g, h](0, \mathbf{x}) = g(\mathbf{x})$ , | **[primitive recursion]** |
| $\mathrm{PREC}[g, h](S(y), \mathbf{x}) =$ | |
| $\quad h(y, \mathbf{x}, \mathrm{PREC}[g, h](y, \mathbf{x}))$ . | |

To define the *corresponding rewrite system* $R$, we assume a signature $\Sigma$, including for each function $f \in \mathrm{PRIM}$, the corresponding function symbol $f$. Further, we stipulate that $\{0, S\} \subset \Sigma$. Finally, the corresponding rewrite system $R$ is defined by schematic rules given in Table 2.

Table 2: Term-Rewriting Characterisation of Primitive Recursion

| | |
|---|---|
| $O^n(\mathbf{x}) \to 0$ , | **[zero]** |
| $P(\mathbf{x}) \to x_i$ , | **[projection]** |
| $\mathrm{SUB}[f, \mathbf{g}](\mathbf{x}) \to f(\mathbf{g}(\mathbf{x}))$ , | **[composition]** |
| $\mathrm{PREC}[g, h](0, \mathbf{x}) \to g(\mathbf{x})$ , | **[primitive recursion]** |
| $\mathrm{PREC}[g, h](S(y), \mathbf{x}) \to$ | |
| $\quad h(y, \mathbf{x}, \mathrm{PREC}[g, h](y, \mathbf{x}))$ . | |

Obviously the obtained rewrite system $R$ is infinite. Further, it is easy to see that $R$ is terminating and ground-confluent. E.g. termination can be seen by a suitable instance $\succ_{\mathrm{RPO}}$ of a recursive path order (RPO), see [4].

By conceiving function classes $\mathcal{C}$ as rewrite systems $R$, we turn our emphasis towards the *computation* of the functions in $\mathcal{C}$. This raises the questions of the *complexity* of the computation of a normal-form in $R$. In general, one expects that the complexity of the computation of $f \in \mathcal{C}$ is measured by a function in $\mathcal{C}$.

We introduce some further notation. Let $R$ denote a terminating rewrite system. We define the *norm* of a term $t$, denoted as $\mathrm{N}(t)$, as the size of $t$, i.e.

the number of symbols occurring in $t$.

**Definition 2.1.** The *derivation tree* $\mathcal{T}_R(t_1)$ of $t_1$, and the *derivation length function* of $f$ under $\to_R^*$ are defined as follows:

$$\begin{aligned} \mathcal{T}_R(t) &:= \{s : t \to_R^* s\} \\ \mathrm{Dl}_R(t) &:= \text{ the depth of the tree } \mathcal{T}_R(t) \\ \mathrm{DL}_{R,f}(\mathbf{t}) &:= \mathrm{Dl}_R(f(\mathbf{t})) \\ \mathrm{Dl}_R(n) &:= \max\{\mathrm{Dl}_R(t) : \mathrm{N}(t) \leq n\}. \end{aligned}$$

Let $f$ be a function in PRIM, then $\mathrm{DL}_{R,f}(\mathbf{n})$ bounds the maximal number of rewrite steps ending in the numeral $\overline{f(\mathbf{n})}$. In this sense $\mathrm{DL}_{R,f}(\mathbf{n})$ measures the *computational complexity* of $f(\mathbf{n})$.

With respect to the given example, we see that for each $f \in \mathcal{C}$, the length of the computation $\mathrm{DL}_{R,f}(\mathbf{n})$ is bounded by a primitive recursive function. In Hofbauer[14] it is shown that any *finite* rewrite rule reducing under RPO induces a primitive recursive derivation length function. Conversely it is easy to see that rewrite rules induced by primitive recursion reduce under RPO. We can apply this result to $R$, by noting that for each fixed term $f(\mathbf{n}) \in \mathcal{T}(\Sigma)$, its derivation tree is finite. Thus the rewrite system $R$ is *feasible*. The same holds for the class of *multiple recursive functions*, MREC, compare [17, 10, 3].

However, in the general case of an arbitrary function-class we cannot expect that $\mathrm{DL}_{R,f}$ can be bounded in a feasible way, even $\mathcal{C}$ is a rather small complexity class, such as the polytime functions. This is the topic of the following two sections.

# 3 Predicative Recursion for $\mathcal{F}_{\mathbf{PTIME}}$

It suffices to consider the polytime computable functions, i.e. those functions computable by a Turing machine $M$, such that $M$ runs in time $\leq p(n)$ for all inputs of length $n$, where $p$ denotes a polynomial. For our purposes, it is best to consider equivalent formulations of the class of polytime computable functions in terms of recursion schemes.

Recursion schemes such as *bounded recursion* due to Cobham [13] generate exactly the functions computable in polytime. In contrast to this, Bellantoni-Cook [7] introduce certain *unbounded* recursion schemes that distinguish between variables as to their position in a function.

$$f(\underbrace{x_1, \ldots, x_k}_{\text{normal}}; \underbrace{y_1, \ldots, y_l}_{\text{safe}}) \quad \in \mathcal{B}^{k,l} \ .$$

Variables $\mathbf{x}$ occurring to the left of the semi-colon are called *normal*, while variables $\mathbf{a}$ to the right are called *safe*. This separation between *normal* and *safe* variables gives rise to the following definition of the *predicative recursive functions*; for further details please see [7]. The idea of separating the arguments in two sets of normal and safe arguments is also known under the name *tired recursion*.

Table 3: Predicative Recursive Functions

| | |
|---|---|
| $S_i^{0,1}(;a) = 2a + i$ | **[successors]** |
| $O^{k,l}(\mathbf{x};\mathbf{a}) = 0$ | **[zero]** |
| $U^{k,l}(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) = x_r$ | **[projection]** |
| $P^{0,1}(;0) = 0$ | **[predecessor]** |
| $P^{0,1}(;S_i(;a)) = a$ | |
| $C^{0,3}(;0,a_0,a_1) = a_0$ | **[conditional]** |
| $C^{0,3}(;S_i(;a),a_1,a_0) = a_{2-i}$ | |
| $\mathrm{SUB}^{k,l}[f,\mathbf{g},\mathbf{h}](\mathbf{x};\mathbf{a}) = f(\mathbf{g}(\mathbf{x};);\mathbf{h}(\mathbf{x};\mathbf{a}))$ | **[safe composition]** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](0,\mathbf{x};\mathbf{a}) = g(\mathbf{x};\mathbf{a})$ | **[predicative recursion** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](S_i(;b),\mathbf{x};\mathbf{a}) =$ | **on notation]** |
| $= h_1(b,\mathbf{x};\mathbf{a},\mathrm{PREC}[g,h_1,h_2](b,\mathbf{x};\mathbf{a}))$ . | |

We use the following notation: $i \in \{1,2\}$ and $r \in [1,k+l]$.

**Definition 3.1.** The defining equations for the predicative recursive functions for specific arities are presented in Table 3. Let $\mathsf{Q} \in \{S_i, O, U, P, \mathrm{SUB}, \mathrm{PREC}\}$. Then $\mathsf{Q}^{k,l} \in \mathcal{B}^{k,l}$. The superscripts denote the number of *normal* and *safe* arguments, respectively. As we have already done above, we occasionally drop these superscripts. The set of *predicative recursive functions* is defined as $\mathcal{B} = \bigcup_{k,l} \mathcal{B}^{k,l}$.

The polytime functions $\mathcal{F}_{\mathrm{PTIME}}$ can be defined as follows.

$$\mathcal{F}_{\mathrm{PTIME}} = \mathcal{B} = \bigcup_{k,l} \mathcal{B}^{k,l} \ .$$

Beckmann and Weiermann investigated the system of equations in the term rewriting framework, especially the rates of growth of the derivation lengths of corresponding rewrite systems. Based on the above equations, the rewrite system $R_B$ is defined by the schematic rules given in Table 4.

It is easy to see, that $R_B$ is terminating and ground-confluent, compare [6]. However, the following theorem shows that the rewrite system $R_B$ is *not* a feasible term-rewriting characterisation of the polytime functions, see [6] for a formal proof.

**Proposition 3.1** (Beckmann and Weiermann 6)**.** *For every $f \in \mathcal{B}$, the derivation length function* $\mathrm{DL}_{R_B,f}(\mathbf{m};\mathbf{n})$ *can be bounded* only *by an* exponential *of a monotone polynomial in the length of inputs* $\mathbf{m}$ and $\mathbf{n}$.

Table 4: A Non-Feasible Term-Rewriting Characterisation of the Predicative Recursive Functions

| | |
|---|---|
| $O^{k,l}(\mathbf{x};\mathbf{a}) \to 0$ , | **[zero]** |
| $U^{k,l}(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) \to x_r$ , | **[projection]** |
| $P^{0,1}(;0) \to 0$ , | **[predecessor]** |
| $P^{0,1}(;S_i(;a)) \to a$ , | |
| $C^{0,3}(;0,a_0,a_1) \to a_0$ , | **[conditional]** |
| $C^{0,3}(;S_i(;a),a_1,a_0) \to a_{2-i}$ , | |
| $\mathrm{SUB}^{k,l}[f,\mathbf{g},\mathbf{h}](\mathbf{x};\mathbf{a}) \to f(\mathbf{g}(\mathbf{x};);\mathbf{h}(\mathbf{x};\mathbf{a}))$ , | **[safe composition]** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](0,\mathbf{x};\mathbf{a}) \to g(\mathbf{x};\mathbf{a})$ , | **[predicative recursion** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](S_i(;b),\mathbf{x};\mathbf{a}) \to$ | **on notation]** |
| $\to h_1(b,\mathbf{x};\mathbf{a},\mathrm{PREC}[g,h_1,h_2](b,\mathbf{x};\mathbf{a}))$ . | |

We use the following notation: $i \in \{1,2\}$ and $r \in [1,k+l]$.

To prove this fact, Beckmann and Weiermann give an example of a function $f \in \mathcal{B}$ so that the derivation length function $\mathrm{DL}_{R_B,f}$ cannot be bounded by a polynomial. The central idea is to define a function $f \in \mathcal{B}$, by the use of predicative recursion and substitution, such that $f(\mathbf{m};)$ rewrites to terms that essentially code binary trees of height $m$. As the rewrite relation is unrestricted, an leftmost-innermost derivation starting from those terms has to be exponentially in the length of $m$, thus exponentially in the length of the input of the function $f$.

We conclude from this exponential lower-bound, that despite the careful distinction between *normal* and *safe* arguments, that makes sure that all *functions* in $\mathcal{B}$ are polytime computable, our chosen computation model—term rewriting—is yet too powerful.

The natural solution to this problem is to introduce a specific rewriting *strategy*, reflecting the different roles played by *normal* and *safe* argument. The strategy is remarkably simple. It suffices to make sure that rewrite rules for *safe composition* and *predicative recursion* must only be applied to terms, where all *safe* arguments are already in normal-form, i.e. are numerals. To reflect this strategy, Beckmann and Weiermann introduced the following clever term-rewriting characterisation of $\mathcal{B}$.

The rewrite system $R_{B'}$ is defined by the schematic rules given in Table 5.

In [6] it is shown that for every $f \in \mathcal{B}$, $\mathrm{DL}_{R_{B'},f}(\mathbf{m};\mathbf{n})$ is bounded by a monotone polynomial in the length of inputs $\mathbf{m}$ and $\mathbf{n}$. This was shown by a number-theoretic interpretation of the rewrite system $R_{B'}$. We used the obser-

Table 5: A Feasible Term-Rewriting Characterisation of the Predicative Recursive Functions

| | |
|---|---|
| $O^{k,l}(\mathbf{x};\mathbf{a}) \to 0$ , | **[zero]** |
| $U^{k,l}(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) \to x_r$ , | **[projection]** |
| $P^{0,1}(;0) \to 0$ , | **[predecessor]** |
| $P^{0,1}(;S_i(;a)) \to a$ , | |
| $C^{0,3}(;0,a_0,a_1) \to a_0$ , | **[conditional]** |
| $C^{0,3}(;S_i(;a),a_1,a_0) \to a_{2-i}$ , | |
| $\mathrm{SUB}^{k,l}[f,\mathbf{g},\mathbf{h}](\mathbf{x};\mathbf{n}) \to f(\mathbf{g}(\mathbf{x};);\mathbf{h}(\mathbf{x};\mathbf{n}))$ , | **[safe composition]** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](0,\mathbf{x};\mathbf{n}) \to g(\mathbf{x};\mathbf{n})$ , | **[predicative recursion** |
| $\mathrm{PREC}^{k+1,l}[g,h_1,h_2](S_i(;b),\mathbf{x};\mathbf{n}) \to$ $\to h_i(b,\mathbf{x};\mathbf{n},\mathrm{PREC}[g,h_1,h_2](b,\mathbf{x};\mathbf{n}))$ . | **on notation]** |

We use the following notation: $i \in \{1,2\}$ and $r \in [1,k+l]$.

vation that the derivation trees $\mathcal{T}_{R_{B'}}(f(\mathbf{m};\mathbf{n}))$ are *isomorphic* no matter how the safe input numerals $\mathbf{n}$ vary, to extend this result, as follows.

**Theorem 3.1.** *For every $f \in \mathcal{B}$, $\mathcal{D}_{R_{B'},f}(\mathbf{m};\mathbf{n})$ is bounded by a monotone polynomial in the length of the normal inputs $\mathbf{m}$ only. Specifically for each $f$ we can find a number $\ell(f)$ so that $\mathcal{D}_{R_{B'},f}(\mathbf{m};\mathbf{n}) \leq (2+|\mathbf{m}|)^{\ell(f)}$.*

*Proof.* For a proof see [1]. This proof is direct and based on polynomial interpretations, below we give a different proof, explaining the effect of the chosen strategy more conclusively. □

The following result stresses the importance of the *strategy* to obtain a feasible term-rewriting characterisation of $\mathcal{F}_{\mathrm{PTIME}}$. We consider a variant of the rewrite system $R_{B'}$, denoted as $R_{\overline{B}}$. The rewrite system $R_{\overline{B}}$ is defined similarly as $R_{B'}$, but for the definition of the rules for *safe composition* and *predicative recursion*, whose definition is based on the original rewrite system $R_B$. The schematic rewrite rules for these cases are given in Table 6.

The rules in $R_{\overline{B}}$ cannot be extended to a rewrite relation as the rules are not closed under substitution. Abusing notation we still write $\to_{R_{\overline{B}}}$ to denote the least binary relation that contains $\to$ and is closed under contexts. Further, we use the same denotation for the derivation tree and the derivation length function of $R_{\overline{B}}$. Clearly, $R_{\overline{B}}$ coincides with $R_{B'}$ on ground terms.

Table 6: Making the Strategy explicit

$$\mathrm{SUB}[f,\mathbf{g},\mathbf{h}](\mathbf{x};\mathbf{s}) \to f(\mathbf{g}(\mathbf{x};);\mathbf{h}(\mathbf{x};\mathbf{s}))\ ,$$

where the $\mathbf{s}$ are $R_B$-irreducible and

$$\begin{array}{rcl} \mathrm{PREC}[g,h_1,h_2](O,\mathbf{x};\mathbf{s}) & \to & g(\mathbf{x};\mathbf{s})\ , \\ \mathrm{PREC}[g,h_1,h_2](S_i(;x),\mathbf{x};\mathbf{s}) & \to & h_i(x,\mathbf{x};\mathbf{s},\mathrm{PREC}[g,h_1,h_2](x,\mathbf{x};\mathbf{s}))\ , \end{array}$$

such that the $\mathbf{s}$ are $R_B$-irreducible.

**Theorem 3.2.** *For every $f \in \mathcal{B}$, $\mathrm{DL}_{R_{\overline{B}},f}(\mathbf{t};\mathbf{s})$ is bounded by a monotone polynomial in the length of inputs $\mathbf{t}$ and $\mathbf{s}$. Moreover the polynomial depends on the depths of $\mathcal{T}_{R_{\overline{B}}}(\mathbf{s})$ only linearly.*

# 4 Primitive Recursion and RPO

We return to the introductory example on primitive recursive functions. For our purpose it is best to define recursive path orders (RPOs) in the following way, compare [10]. It suffices to give the definition for the ground term algebra.

To simplify notation we write $\prec$ instead of $\prec_{\mathrm{RPO}}$. In the definition we make use of an auxiliary varyadic function symbol 'list' to denote sequences $(s_0,\ldots,s_n)$ of terms. Instead of $\mathrm{list}(s_0,\ldots,s_n)$ we write $(s_0,\ldots,s_n)$. Suppose $\Sigma = \{f_0,f_1,\ldots\}$, where $f_i$ is lower in the precedence than $f_j$, iff $i<j$ holds.

**Definition 4.1.** Inductive definition of $\prec$.

1. $\exists j \in \{1,\ldots,n\}\ [s \preceq t_j] \Rightarrow s \prec f(t_1,\ldots,t_n)$.
2. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,\forall i \in \{1,\ldots,m\}\ (s_i \prec t) \Rightarrow (s_1,\ldots,s_m) \prec t$.
3. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_q(s_1,\ldots,s_m)$ with $q<p\,\&\,\forall i \in \{1,\ldots,m\}\ (s_i \prec t) \Rightarrow s \prec t$.
4. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_p(s_1,\ldots,s_n)\,\&\,(s_1,\ldots,s_n) \prec (t_1,\ldots,t_n) \Rightarrow s \prec t$.
5. $\sigma = \sigma_0 * \cdots * \sigma_n\,\&\,\forall i \le n\ (\sigma_i \preceq t_i)\,\&\,\exists i \le n\ (\sigma_i \prec t_i) \Rightarrow \sigma \prec (t_0,\ldots,t_n)$ $(n \ge 1)$.

   $\sigma = \sigma_0 * \cdots * \sigma_n$ denotes the fact that the sequence $\sigma$ of terms is obtained from the concatenated $\sigma_0 * \cdots * \sigma_n$ by a permutation.

Note that in rule 2, $m=0$ is allowed, hence $() \prec t$ for any $t \in \mathcal{T}(\Sigma)$. Further, we write $s \succ t$ for $t \prec s$.

As Buchholz [10] observed, each rewrite system $R$ reducing under RPO is already reducing under a suitable approximation $\prec^k_{\mathrm{RPO}}$ of $\prec_{\mathrm{RPO}}$, where $k$ depends on $R$, only.

**Definition 4.2.** We set $\prec^k_{\mathrm{RPO}} \equiv \prec^k_k$ and define $\prec^l_k$ inductively.

1. $\exists j \in \{1,\ldots,n\}\ [s \preceq^l_k t_j] \Rightarrow s \prec^l_k f(t_1,\ldots,t_n)$. [subterm]
2. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,\forall i \in \{1,\ldots,m\}\ (s_i \prec^l_k t)\,\&\,m < k \Rightarrow (s_1,\ldots,s_m) \prec^l_k t$. [inaccessibility]
3. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_q(s_1,\ldots,s_m)$ with $q < p\,\&\,m < k\ \&\ \forall i \in \{1,\ldots,m\}\ (s_i \prec^l_k t) \Rightarrow s \prec^{l+1}_k t$. [inaccessibility]
4. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_p(s_1,\ldots,s_n)\,\&\,(s_1,\ldots,s_n) \prec^l_k (t_0,\ldots,t_n)\,\&\,n < k \Rightarrow s \prec^l_k t$. [monotonicity]
5. $\sigma = \sigma_0 * \cdots * \sigma_n = (s_0,\ldots,s_m)\,\&\,\forall i \le n\ (\sigma_i \preceq^l_k t_i)\,\&\,\exists i \le n\ (\sigma_i \prec^l_k t_i)\,\&\,m < k \Rightarrow \sigma \prec^l_k (t_0,\ldots,t_n)\,(n \ge 1)$. [multiset]

The next lemma follows easily.

**Lemma 4.1.** *Let* $\mathrm{dp}(t)$ *denote the depth of* $t$ *and* $\mathrm{wd}(t)$ *denote the maximal number of arguments of a function symbol, occurring in* $t$*. Then we have*

$$s \prec^l_k t \Rightarrow \mathrm{dp}(s) \le \mathrm{dp}(t) + l\,\&\,\mathrm{wd}(s) \le \max\{\mathrm{wd}(t), k\}\ .$$

For completeness, we restate the following theorem, due to Buchholz.

**Proposition 4.1** (Buchholz 10)**.** *If* $R$ *is a finite rewrite system reducing under* $\prec_{\mathrm{RPO}}$*, then* $\to_R$ *compatible with* $\prec^k_{\mathrm{RPO}}$*, where* $k := \max\{\mathrm{N}(r) \mid (l \to r) \in R\}$*. Consequently* $\mathrm{Dl}_R$ *is bounded by a primitive recursive function.*

# 5 A path ordering for the polynomial time computable functions

In order to define a *path order for* $\mathcal{F}_{\mathrm{PTIME}}(POP)$, that mimics the definition of RPO, we introduce two orders $\sqsubset$ and $\prec \equiv \prec_{\mathrm{POP}}$ as follows.

**Definition 5.1.** Inductive definition of $\sqsubseteq$.

1. $\exists j \in \{1,\ldots,n\}\ [s \sqsubseteq t_j] \Rightarrow s \sqsubset f(t_1,\ldots,t_n)$.
2. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_q(s_1,\ldots,s_m)$ with $q < p\,\&\,\forall i \in \{1,\ldots,m\}\ s_i \sqsubset t) \Rightarrow s \sqsubset t$.

Inductive definition of $\prec_{\mathrm{POP}}$. To simplifiy notation we write $\prec$ instead of $\prec_{\mathrm{POP}}$. The definition of $\prec_{\mathrm{POP}}$ is based on $\sqsubseteq$.

1. $s \sqsubset t \Rightarrow s \prec t$.

2. $\exists j \in \{1,\ldots,n\}\ [s \preceq t_j] \Rightarrow s \prec f(t_1,\ldots,t_n)$.

3. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,\exists i_0 \in \{1,\ldots,m\}\ [\forall i \neq i_0(1 \leq i \leq m)\ (s_i \sqsubset t)\,\&\,s_{i_0} \prec t] \Rightarrow (s_1,\ldots,s_m) \prec t$.

4. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_p(s_1,\ldots,s_n)\,\&\,(s_1,\ldots,s_n) \prec (t_1,\ldots,t_n) \Rightarrow s \prec t$.

5. $\sigma = \sigma_0 * \cdots * \sigma_n\,\&\,\forall i \leq n(\sigma_i \preceq t_i)\,\&\,\exists i \leq n(\sigma_i \prec t_i) \Rightarrow \sigma \prec (t_0,\ldots,t_n)\,(n \geq 1)$.

Note that in rule 3, $m = 0$ is allowed, hence $() \prec t$ for any $t \in \mathcal{T}(\Sigma)$. Further, we write $s \succ t$ for $t \prec s$.

The above given definition stems from a close inspection of the rewrite systems $R_{B'}$ and $R_{\overline{B}}$ in relation to the definition of $\prec_{\text{RPO}}$. The separation in two orders $\sqsubset$ and $\prec_{\text{POP}}$ is necessary to break the strength of the recursive path order $\prec_{\text{RPO}}$ that induced primitive recursive derivation lengths. Note that the following relations are missing:

- $t \equiv f_p(t_1,\ldots,t_n)\,\&\,\forall i(s_i \sqsubset t) \Rightarrow (s_0,\ldots,s_m) \sqsubset t$.
- $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_p(s_1,\ldots,s_n)\,\&\,(s_0,\ldots,s_n) \sqsubset (t_0,\ldots,t_n) \Rightarrow s \sqsubset t$.
- $\sigma = \sigma_0 * \cdots * \sigma_n\,\&\,\forall i \leq n(\sigma_i \sqsubseteq t_i)\,\&\,\exists i \leq n(\sigma_i \sqsubset t_i) \Rightarrow \sigma \sqsubset (t_1,\ldots,t_n)\,(n \geq 1)$.
- $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_q(s_1,\ldots,s_m)$ with $q < p\,\&\,\forall i \in \{1,\ldots,m\}\ (s_i \prec t) \Rightarrow s \prec t$.

**Theorem 5.1.** *If a finite rewrite rule $R$ is reducing under POP, then the derivation length function* $\mathrm{Dl}_R(n)$ *is bounded by a monotone polynomial in $n$.*

To prove this theorem, we introduce suitable *approximations* $\prec_k$ of $\prec_{\text{POP}}$. Then we will follow the pattern of Buchholz's proof that the recursive path orders imply primitive recursive derivation lengths, see [10].

**Definition 5.2.** Inductive definition of $\sqsubset_k^l$; we write $\sqsubset_k$ to abbreviate $\sqsubset_k^k$.

1. $\exists j[s \sqsubseteq_k^l t_j] \Rightarrow s \sqsubset_k^l f(t_0,\ldots,t_n)$.
2. $t \equiv f_p(t_0,\ldots,t_n)\,\&\,s \equiv f_q(s_0,\ldots,s_m)$ with $q < p\,\&\,m < k\,\&\,\forall i(s_i \sqsubset_k^l t) \Rightarrow s \sqsubset_k^{l+1} t$.

Inductive definition of $\prec_k$. The definition of $\prec_k$ is based on $\sqsubseteq_k$.

1. $s \sqsubset_k t \Rightarrow s \prec_k t$.
2. $\exists j \in \{1,\ldots,n\}\ [s \preceq_k t_j] \Rightarrow s \prec_k f(t_1,\ldots,t_n)$.
3. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,\exists i_0 \in \{1,\ldots,m\}\ [\forall i \neq i_0(1 \leq i \leq m)\ (s_i \sqsubset_k t)\ \&\ s_{i_0} \prec_k t]\,\&\,m < k \Rightarrow (s_1,\ldots,s_m) \prec_k t$.

4. $t \equiv f_p(t_1,\ldots,t_n)\,\&\,s \equiv f_p(s_1,\ldots,s_n)\,\&\,(s_1,\ldots,s_n) \prec_k (t_1,\ldots,t_n)\,\&\,n < k \Rightarrow s \prec_k t$.

5. $\sigma = \sigma_0 * \cdots * \sigma_n = (s_0,\ldots,s_m)\,\&\,\forall i \leq n\ (\sigma_i \preceq_k t_i)\,\&\,\exists i \leq n\ (\sigma_i \prec_k t_i)\,\&\,m < k \Rightarrow \sigma \prec_k (t_0,\ldots,t_n)\,(n \geq 1)$.

We define $t \in \mathbf{t} = (t_0,\ldots,t_{n-1})$, if there exists $i < n$, such that $t = t_i$. Further

$$\begin{array}{lcl} D_k & := & \{(t_0,\ldots,t_l) \mid \forall j \leq l\ \forall s \prec_k t_j\,(s \in (t_0,\ldots,t_{j-1}))\ , \\ W_k & := & \{t \mid \exists d \in D_k\,\&\,t \in d\}\ . \end{array}$$

The elements of $D_k$ are called $k$-derivations. Finally, set

$$\mathrm{G}_k(\sigma) := \max\{n \in \mathbb{N} \mid \exists(\sigma_0,\ldots,\sigma_n)[\sigma_n \prec_k \cdots \prec_k \sigma_0 \equiv \sigma]\}\ .$$

and define

$$F_{k,p}(n) := \max\{\mathrm{G}_k(f_p\sigma)\colon \mathrm{G}_k(\sigma) \leq n\}\ .$$

It is easy to see that $\mathrm{G}_k(\sigma)$ is defined (written $\mathrm{G}_k(\sigma)\!\downarrow$), iff $\sigma \in W_k$. Our aim is to show this formally in a very weak fragment of arithmetic, the *bounded arithmetic* $S^1_2$, see [11]. We present the crucial lemmas and proof steps. Firstly the following lemma is proven by main induction on $k$ and side induction on $\sigma$.

**Lemma 5.1.** *Inductively we define $d_{k,0} := 2$ and $d_{k,p-1} := (d_{k,p})^k + 1$. There exists a constant (depending only on $k$ and $p$) such that*

$$F_{k,p}(n) \leq cn^{d_{k,p}} + c\ .$$

Now we indicate how to prove (inside the bounded arithmetic $S^1_2$) that, if $\mathrm{G}_k(\sigma)$ is defined, then $\mathrm{G}_k(f_p\sigma)$ is defined.

**Theorem 5.2.** *For each $k$ and $p$ we have:*

$$S^1_2 \vdash \forall\sigma[\mathrm{G}_k(\sigma)\!\downarrow \rightarrow \mathrm{G}_k(f_p\sigma)\!\downarrow]\ .$$

We assume some canonical arithmetisation of terms and identify each term with its *Gödel* number. This can be done in a way, such that the bounded theory $S^1_2$ is strong enough to speak about $\mathcal{T}$ and $\prec_k$. Based on this arithmetisation we formalise the proof of Lemma 5.1 in $S^1_2$. Using Theorem 5.2, we obtain the following crucial theorem.

**Theorem 5.3.** *For any* finite *signature and each $k$, we have:*

$$S^1_2 \vdash \forall\sigma[\mathrm{G}_k(\sigma)\!\downarrow]\ .$$

The proof proceeds by induction over the assertion $\sigma \in W_k$, that is equivalent to the statement $\mathrm{G}_k(\sigma)\!\downarrow$. The only problem in this is the unbounded quantifier in the definition of $W_k$. As we base our considerations on a *finite* signature however, we can use (the formalised version of) Lemma 5.1 to verify that such a bound exists and is expressible in $S^1_2$.

In analogy to the proof of Proposition 4.1 we employ the following characterisation of the provably total functions of $S^1_2$, due to Buss.

**Proposition 5.1** (Buss 11)**.** *Every provably total function of $S^1_2$ is polynomial time computable.*

Using this fact, we obtain the final step in our proof of Theorem 5.1.

**Theorem 5.4.** *For each $k$, $S^1_2$ proves the totality of the following function $F_k(n)$ for small numbers $n(= |x|$ for an $x)$:*

$$F_k(n) := \max\{\mathrm{G}_k(t) : \mathrm{N}(t) \leq n\}.$$

*Therefore $F_k(n)$ is bounded by a monotone polynomial of $n$.*

The first assertion of this theorem follows from Theorem 5.3. To see that this suffices to yield the existence of a monotone polynomial $p(n)$ that bounds $F_k(n)$ we apply Proposition 5.1 to Theorem 5.3. We obtain a polynomial time computable function $h(t)$ witnessing the existence of a $k$-derivation $d \in D_k$ for a term $t$. Finally observe that $\mathrm{G}_k(t)$ is bounded in the *length* of the derivation for $t$.

This concludes the proof-sketch for Theorem 5.1. In the next section we apply the POP $\prec_{\mathrm{POP}}$ to show the well-foundedness of the rewrite system $R_{B'}$.

Note that Theorem 5.1 allows an important strengthening of the aforementioned results on the term-rewriting characterisation of the polytime computable functions. Proposition 3.1, Theorem 3.1 and Theorem 3.2 classify the derivation length of the respective term-rewriting characterisation $R$ in terms of the derivation length function $\mathrm{DL}_{R,f}$, i.e. the (worst-case) complexity of the computation of a given function $f \in \mathcal{B}$ is estimated.

In contrast to this Theorem 5.1 classifies the derivation length of $R$—if $\to_{R_B}$ is compatible with $\prec_{\mathrm{POP}}$—in terms of the the derivation length function $\mathrm{Dl}_R$. We will remark on this as soon as we have verified that the rewrite relation $\to_{R_{B'}}$ is compatible with $\prec_{\mathrm{POP}}$. This is the task of the next section.

# 6 Predicative Recursion and POP

In the above section we have shown that if a finite rewrite system $R$ is reducing under $\prec_{\mathrm{POP}}$, then the derivation length function $\mathrm{Dl}_R(n)$ is bounded by a monotone polynomial in $n$. To reprove Theorem 3.1, it suffices to define a monotone interpretation over $\mathcal{B}$ into sequences of ground terms. To each ground term $t$ over $\mathcal{B}$ we associate a sequence $\mathrm{S}(t)$ of terms. In $\mathrm{S}(t)$ each function symbol $f \in \mathcal{B}^{k,l}$ is transformed to be a symbol of arity $k$.

**Definition 6.1.** Set $sn(\overline{n}) = |n|$ for numerals $\overline{n}$, $sn(f(\mathbf{t};\mathbf{s})) = \max_j sn(s_j)$ else. Then define

1. $\mathrm{S}(\overline{n}) := ()$(empty sequence) and $\mathrm{S}(S_i(;t)) := (S_i) * \mathrm{S}(t)$ for $t \not\equiv \overline{n}$.
2. For $f \not\equiv S_i$, $\mathrm{S}(f(\mathbf{t};\mathbf{s})) := (f(\mathrm{N}(t_0), \ldots, \mathrm{N}(t_n)), \mathrm{S}(s_0), \ldots, \mathrm{S}(s_m))$.

3. $\mathrm{N}(t) := \mathrm{S}(t) * sn'(t)$, where for a numeral $\overline{m} = sn(t)$, $m'$ denotes a sequence of $S_1, S_2$.

For example for $5 = S_1(;S_2(;0))$, we have $5' = (S_1, S_2)$. Using these definitions, we conclude the following theorem.

**Theorem 6.1.**

$$s \to_{R_{B'}} t \Rightarrow \mathrm{S}(t) \prec_{\mathrm{POP}} \mathrm{S}(s) \ .$$

*Proof.* We sketch the proof, by considering one of the rules in $R_{B'}$, namely the following:

$$\mathrm{PREC}[g, h_1, h_2](S_i(;t), \mathbf{t}; \mathbf{n}) \to h_i(t, \mathbf{t}; \mathbf{n}, \mathrm{PREC}[g, h_1, h_2](t, \mathbf{t}; \mathbf{n})) \ .$$

Let $\mathrm{lh}(f)$, $f \in B$ be defined as in [6]: (i) $\mathrm{lh}(f) := 1$, for $f \in \{S_i, O, U, P, \}$. (ii) $\mathrm{lh}(\mathrm{SUB}[f, \mathbf{g}, \mathbf{h}]) := 1 + \mathrm{lh}(f) + \mathrm{lh}(g_1) + \cdots + \mathrm{lh}(g_{k'}) + \mathrm{lh}(h_1) + \cdots + \mathrm{lh}(h_{l'})$. (iii) $\mathrm{lh}(\mathrm{PREC}[g, h_1, h_2]) := 1 + \mathrm{lh}(g) + \mathrm{lh}(h_1) + \mathrm{lh}(h_2)$.

We define the precedence $\succ$ compatible with lh, i.e. $f \succ g$ if $\mathrm{lh}(f) > \mathrm{lh}(g)$. By definition we obtain the following sequence of comparisons:

$$\begin{aligned}
&\mathrm{S}(\mathrm{PREC}[g, h_1, h_2](S_i(;t), \mathbf{t}; \mathbf{n})) = \\
&= (\mathrm{PREC}(\mathrm{N}(S_i(;t)), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)), \mathrm{S}(n_1), \ldots, \mathrm{S}(n_l)) = \\
&= (\mathrm{PREC}(S_i * \mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)) \qquad (\star) \\
&\succ_{\mathrm{POP}} (h_i(\mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)), \mathrm{PREC}(\mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k))) = \\
&= (h_i(\mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)), \mathrm{S}(n_1), \ldots, \mathrm{S}(n_l), \\
&\quad \mathrm{PREC}(\mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k))) = \\
&= \mathrm{S}(h_i(t, \mathbf{t}; \mathbf{n}, \mathrm{PREC}[g, h_1, h_2](t, \mathbf{t}; \mathbf{n}))) \ .
\end{aligned}$$

To see that $(\star)$ holds, we proceed as follows. For all $i \in [1, k]$ we have

$$\mathrm{PREC}(S_i * \mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)) \succ_{\mathrm{POP}} \mathrm{N}(t_i) \ ,$$

due to Definition 5.1.1. Further, by Definition 5.1.4, we conclude

$$\mathrm{PREC}(S_i * \mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)) \succ_{\mathrm{POP}} \mathrm{PREC}(\mathrm{N}(t), \mathrm{N}(t_1), \ldots, \mathrm{N}(t_k)) \ .$$

Thus $(\star)$ follows if we apply Definition 5.1.3 together with Definition 5.1.5. In rule 5.1.3 we employ the precedence, that implies $\mathrm{PREC}[g, h_1, h_2] \succ h_i$, for $i \in \{1, 2\}$. □

The above theorem verifies our claims that the POP is indeed compatible with $R_{B'}$. Further we obtain the following corollary.

**Corollary 6.1.** *The derivation length function* $\mathrm{Dl}_{R_{B'}}$ *is bounded by a monotone polynomial in* $n$.

At the end of Section 5 it was mentioned that in Theorem 5.1 the dependence on a specific function $f \in \mathcal{B}$ could be removed and the derivation length is measured in the term-complexity of the starting term. Due to this we can conclude the above corollary. The latter result does not follow from the former results. Proposition 3.1, Theorem 3.1 and Theorem 3.2. Clearly it is possible to use Proposition 3.1 or Theorem 3.1 to define, for a given term $t$ with term-complexity $n$ a polynomial in $n$ that bounds the derivation length function $\mathrm{Dl}_{R_{B'}}$. However the thus obtained polynomial will depend on the term-complexity $n$. Contrary the monotone polynomial referred to in Corollary 6.1, does not depend on $n$.

# 7 Conclusion

The purpose of this paper is to stress the need for strategies in the framework of term-rewriting analysis of complexity classes. Independently our results can be conceived as a simplification and clarification of the results obtained in [6]. In particular, we introduce a new polynomial path order $\prec_{\mathrm{POP}}$ for polynomial time functions, such that for a finite rewrite system $R$ compatible with $\prec_{\mathrm{POP}}$, the *complexity* or *derivation length function* $\mathrm{Dl}_R(n)$ is guaranteed to be bounded by a polynomial in $n$.

The definition of the polynomial path order stems from a close inspection of the rewrite systems $R_{B'}$ and $R_{\overline{B}}$ in relation to the definition of recursive path orders. We believe that this path orders brings out clearly the ideas underlying the clever term-rewriting characterisation of $\mathcal{F}_{\mathrm{PTIME}}$ originally given by Beckmann and Weiermann in [6].

To show that for a finite rewrite system $R$ compatible with $\prec_{\mathrm{POP}}$, the *complexity* or *derivation length function* $\mathrm{Dl}_R(n)$ is guaranteed to be bounded by a polynomial in $n$, we used Buchholz's proof that the recursive path orders induce primitive recursive derivation length as pattern, see [10]. However, instead of the fragment of arithmetic $I\Sigma_1$, we employed the bounded arithmetic $S^1_2$. In this way we could use the fact that this weak arithmetic exactly captures the polynomial time computable functions as provably recursive functions.

# Note Added in Print

We would like to thank Arnold Beckmann who uncovered an embarrassing error in this paper.

Arnold pointed out to us that the function $\mathrm{G}_k$ is not necessarily polytime computable, and thus is not representable inside of $S^1_2$, contrary to what we claim in the proof sketch of Theorem 5.3. Thus Theorem 5.3 is wrong, which invalidates Theorem 5.4. In follow-up work, we established a corrected variant of this theorem, cf. [2].

# References


[1] T. Arai and G. Moser. A note on a term rewriting characterization of PTIME. In *Proc. 7th WST*, pages 10–13. number AIB-2004-07 of Aachener Informatik-Berichte, 2004. Extended abstract.

[2] T. Arai and G. Moser. Proofs of termination of rewrite systems for polytime functions. In *Proc. 25th FSTTCS*, volume 3821 of *LNCS*, pages 529–540, 2005.

[3] T. Arai. Some results on cut-elimination, provable well-orderings, induction, and reflection. *APAL*, pages 93–184, 1998. Original title: 'From the Attic'.

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

[5] M. Barra. Complexity bounds on ARSs characterizing complexity classes. Master's thesis, University of Oslo, Department of Mathematics, 2004.

[6] A. Beckmann and A. Weiermann. A term rewriting characterization of the polytime functions and related complexity classes. *Arch. Math. Log.*, 36:11–30, 1996.

[7] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complex.*, 2(2):97–110, 1992.

[8] G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretations and small space bounds. In *Proc. 16th RTA*, volume 3467 of *LNCS*, pages 150–164, 2005.

[9] G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretations: A way to control resources. *TCS*, 2005. to appear.

[10] W. Buchholz. Proof-theoretical analysis of termination proofs. *APAL*, 75:57–65, 1995.

[11] S.-R. Buss. *Bounded Arithmetic.* PhD thesis, Princeton University, 1985.

[12] E.-A. Cichon and J.-Y. Marion. Light LPO. Technical report 99-R-138, 1999.

[13] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science, proceedings of the second International Congress*, Jerusalem, 1964, 1965. North-Holland.

[14] D. Hofbauer. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *TCS*, 105:129–140, 1992.

[15] J.-Y. Marion. Analysing the implicit complexity of programs. *IC*, 183:2–18, 2003.

[16] I. Oitavem. A term rewriting characterization of the functions computable in polynomal space. *Arch. Math. Log.*, 41:35–47, 2002.

[17] A. Weiermann. Termination proofs for term rewriting systems with lexicographic path ordering imply multiply recursive derivation lengths. *TCS*, 139:355–362, 1995.