A New Order-theoretic Characterisation of the Polytime Computable Functions[☆]

Martin Avanzini[a], Naohi Eguchi[a], Georg Moser[a]

[a] *Institute of Computer Science, University of Innsbruck, Austria*

Abstract

We propose a new order-theoretic characterisation of the class of polytime computable functions. To this avail we define the *small polynomial path order* (*sPOP** for short). This termination order entails a new syntactic method to analyse the innermost runtime complexity of term rewrite systems fully automatically: for any rewrite system compatible with sPOP* that employs recursion up to depth d, the (innermost) runtime complexity is polynomially bounded of degree d. This bound is tight. Thus we obtain a direct correspondence between a syntactic (and easily verifiable) condition of a program and the asymptotic worst-case complexity of the program.

Keywords: Term Rewriting, Complexity Analysis, Implicit Computational Complexity, Automation
2000 MSC: 68Q42,
2000 MSC: 03D15,
2000 MSC: 03B70

1. Introduction

In this paper we are concerned with the complexity analysis of term rewrite systems (TRSs for short). Based on a careful investigation into the principle of *predicative recursion* as proposed by Bellantoni and Cook [1] we introduce a new termination order, the *small polynomial path order* (*sPOP** for short). The order sPOP* provides a new characterisation of the class FP of polytime computable functions. Any function f computable by a TRS $\mathcal{R}$ compatible with sPOP* is polytime computable. On the other hand for any polytime computable function f, there exists a TRS $\mathcal{R}_f$ computing f such that $\mathcal{R}$ is compatible with sPOP*. Moreover sPOP* directly relates the depth of recursion of a given TRS to the polynomial degree of its runtime complexity. More precisely, we call a rewrite system $\mathcal{R}$ *predicative recursive of degree* d if $\mathcal{R}$ is compatible with sPOP* and the depth of recursion of all function symbols in $\mathcal{R}$ is bounded by d (see Section 3 for the formal definition). We establish that any predicative recursive rewrite system of degree d admits runtime complexity in $\mathsf{O}(n^d)$. Here n refers to the sum of the sizes of inputs. Furthermore we obtain a novel, order-theoretic characterisation of $\mathsf{DTIME}(n^d)$, the class of functions computed on register machines in $\mathsf{O}(n^d)$ steps.

Thus we obtain a direct correspondence between a syntactic (and easily verifiable) condition of a program and the asymptotic worst-case complexity of the program. In this sense our work is closely related to similar studies in the field of *implicit computational complexity* (*ICC* for short). On the other hand the order sPOP* entails a new syntactic criteria to automatically establish polynomial runtime complexity of a given TRS. This criteria extends the state of the art in runtime complexity analysis as it is more precise or more efficient than related techniques. Note that the proposed syntactic method to analyse the (innermost)

[☆]This work has been partially supported by the Austrian Science Fund, project number I-603-N18, by the John Templeton Foundation, and the Japan Society for the Promotion of Science.

Email addresses: martin.avanzini@uibk.ac.at (Martin Avanzini), naohi.eguchi@uibk.ac.at (Naohi Eguchi), georg.moser@uibk.ac.at (Georg Moser)

runtime complexity of rewrite systems is fully automatic. For any given TRS, compatibility with sPOP* can be efficiently checked by a machine. Should this check succeed, we get an asymptotic bound on the runtime complexity directly from the parameters of the order. It should perhaps be emphasised that compatibility of a TRS with sPOP* implies termination and thus our complexity analysis technique does not presuppose termination.

In sum, in this work we make the following contributions:

- We propose a new *recursion-theoretic characterisation* $\mathcal{B}_{\mathsf{wsc}}$ over binary words of the class FP. We establish that those $\mathcal{B}_{\mathsf{wsc}}$ functions that are definable with d nestings of predicative recursion can be computed by predicative recursive TRSs of degree d (cf. Theorem 4). Note that these functions belong to $\mathsf{DTIME}(n^d)$.

- We propose the new termination order sPOP*; sPOP* captures the recursion-theoretic principles of the class $\mathcal{B}_{\mathsf{wsc}}$. Thus we obtain a new *order-theoretic characterisation* of the class FP. Moreover, for any predicative recursive TRS of degree d its runtime complexity lies in $O(n^d)$ (cf. Theorem 1). Furthermore this bound is tight, that is, we provide a family of TRSs, delineated by sPOP*, whose runtime complexity is bounded from below by $\Omega(n^d)$, cf. Example 5.

- We extend upon sPOP* by proposing a generalisation, denoted sPOP$^*_{\mathrm{PS}}$, admitting the same properties as above. This generalisations incorporates a more general recursion scheme that makes use of *parameter substitution* (cf. Theorem 2).

- We establish a novel, order-theoretic characterisation of $\mathsf{DTIME}(n^d)$. We show that $\mathsf{DTIME}(n^d)$ corresponds to the class of functions computable by *tail-recursive* predicative TRSs of degree d. This characterisation is based on the generalised small polynomial path order sPOP$^*_{\mathrm{PS}}$ (cf. Theorem 6).

- sPOP* gives rise to a new syntactic method for *polynomial runtime complexity method*. This method is fully automatic. We have implemented the order sPOP* in the *Tyrolean Complexity Tool* TCT, version 2.0, an open source complexity analyser [2]. The experimental evidence obtained indicates the efficiency of the method and the obtained increase in precision.

1.1. Related Work

There are several accounts of predicative analysis of recursion in the (ICC) literature. We mention only those related works which are directly comparable to our work. See [3] for an overview on ICC.

The class $\mathcal{B}_{\mathsf{wsc}}$ is a syntactic restriction of the recursion-theoretic characterisation $\mathcal{N}$ of the class FEXP of *exponential time computable functions*, given by Arai and the second author in [4]. To account for the fact that FEXP is *not closed* under composition in general, the definition of $\mathcal{N}$ relies on a syntactically restricted form of composition. The same composition scheme allows a fine-grained control in our class $\mathcal{B}_{\mathsf{wsc}}$ through the degree of recursion. In [5] the authors use the class $\mathcal{N}$ as a sufficient basis for an order-theoretic account of FEXP, the *exponential path order* (*EPO** for short). Due to the close relationship of $\mathcal{B}_{\mathsf{wsc}}$ and $\mathcal{N}$, our order is both conceptually and technically close to EPO*.

Notably the clearest connection of our work is to Marion's *light multiset path order* (*LMPO* for short) [6] and the *polynomial path order* (*POP** for short) [7–9]. Both orders form a strict extension of sPOP*, but lack the precision of the latter. Although LMPO characterises FP, the runtime complexity of compatible TRSs is not polynomially bounded in general. POP* induces polynomial runtime complexities, but the obtained complexity certificate is usually very imprecise. In particular, due to the multiset status underlying POP*, for each $d \in \mathbb{N}$ one can form a TRS compatible with POP* that defines only a single function, but whose runtime is bounded from below by a polynomial of degree d, in the sizes of the inputs.

In Bonfante et. al. [10] restricted classes of polynomial interpretations are studied that can be employed to obtain polynomial upper bounds on the runtime complexity of TRSs. Polynomial interpretations are complemented with quasi-interpretations in [11], giving rise to alternative characterisations of complexity classes. None of the above results are applicable to relate the depth of recursion to the runtime complexity, in the sense mentioned above. Furthermore it is unknown how the body of work on quasi-interpretations can

be employed in the context of runtime complexity analysis. We have also drawn motivation from Leivant's and Marion's characterisations of $\mathsf{DTIME}(n^d)$ [12, 13], that provide related fine-grained classification of the polytime computable functions. Again, these results lack applicability in the context of runtime complexity analysis.

Polynomial complexity analysis is an active research area in rewriting. Starting from [14] interest in this field greatly increased over the last years, see for example [15–17] and [18] for an overview. This is partly due to the incorporation of a dedicated category for complexity into the annual termination competition (TERMCOMP).[1] However, it is worth emphasising that the most powerful techniques for runtime complexity analysis currently available, basically employ semantic considerations on the rewrite systems, which are notoriously inefficient.

We also want to mention ongoing approaches for the automated analysis of resource usage in programs. Notably, Hoffmann et al. [19] provide an automatic multivariate amortised cost analysis exploiting typing, which extends earlier results on amortised cost analysis. Finally Albert et al. [20] present an automated complexity tool for Java Bytecode programs, Alias et al. [21] give a complexity and termination analysis for flowchart programs, and Gulwani et al. [22] as well as Zuleger et al. [23] provide an automated complexity tool for C programs.

1.2. Outline

We present the main intuition behind sPOP* and provide an informal account of the obtained technical results.

The order sPOP* essentially embodies the predicative analysis of recursion set forth by Bellantoni and Cook [1]. In [1] a recursion-theoretic characterisation $\mathcal{B}$ of the class of polytime computable functions is proposed. This analysis is connected to the important principle of *tiering* introduced by Simmons [24] and Leivant [12, 25, 26]. The essential idea is that the arguments of a function are separated into *normal* and *safe* arguments (or correspondingly into arguments of different tiers). Building on this work we present a subclass $\mathcal{B}_{\mathsf{wsc}}$ of $\mathcal{B}$. Crucially the class $\mathcal{B}_{\mathsf{wsc}}$ admits only a weak form of composition. Inspired by a result of Handley and Wainer [27], we show that $\mathcal{B}_{\mathsf{wsc}}$ captures the polytime functions. This establishes our first main result.

We formulate the class $\mathcal{B}_{\mathsf{wsc}}$ over the set $\{0,1\}^*$ of binary words, the empty word is denoted by ϵ. Arguments of functions are partitioned into normal and safe ones. In notation, we write $f(t_1,\ldots,t_k\,;t_{k+1},\ldots,t_{k+l})$ where *normal* arguments are to the left, and *safe* arguments to the right of the semicolon. Abbreviate $\vec{x} = x_1,\ldots,x_k$ and $\vec{y} = y_1,\ldots,y_l$. The class $\mathcal{B}_{\mathsf{wsc}}$, depicted in Fig. 1, is the smallest class containing certain initial functions and closed under *safe recursion on notation* (**SRN**) and *weak safe composition* (**WSC**). By the weak form of composition only values are ever substituted into normal argument positions.

Suppose the definition of a TRS $\mathcal{R}$ is based on the equations in $\mathcal{B}_{\mathsf{wsc}}$. It is not difficult to deduce a precise bound on the runtime complexity of $\mathcal{R}$ by measuring the number of nested applications of safe recursion, the so called *depth of recursion*. In contrast Bellantoni and Cooks definition [1] of $\mathcal{B}$ is obtained from Fig. 1 by replacing weak safe composition with the more liberal scheme of *safe composition* (**SC**): $f(\vec{x}\,;\vec{y}) = h(\vec{i}(\vec{x}\,;)\,;\vec{j}(\vec{x}\,;\vec{y}))$. As soon as one of the functions $\vec{i}$ is size increasing, a tight correspondence between the runtime complexity and the depth of recursion is lost.

Our central observation is that from the function algebra $\mathcal{B}_{\mathsf{wsc}}$, one can distill a termination argument for the TRS $\mathcal{R}$. With sPOP*, this implicit termination argument is formalised as a termination order. In order to employ the separation of normal and safe arguments, we fix for each defined symbol in $\mathcal{R}$ a partitioning of argument positions into *normal* and *safe* positions. For constructors we fix (as in $\mathcal{B}_{\mathsf{wsc}}$) that all argument positions are safe. Moreover sPOP* restricts recursion to normal argument. Dual, only safe argument positions allow the substitution of recursive calls. Via the order constraints we can also guarantee that only normal arguments are substituted at normal argument positions. We emphasise that our notion of predicative recursive TRS is more liberal than the class $\mathcal{B}_{\mathsf{wsc}}$. Notably values are not restricted to words, but

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

Initial Functions	$S_i(;x) = xi$	$(i = 0, 1)$
	$P(;\epsilon) = \epsilon$	
	$P(;xi) = x$	$(i = 0, 1)$
	$I^{k,l}_j(\vec{x}\,;\vec{y}) = x_j$	$(j \in \{1, \dots, k\})$
	$I^{k,l}_j(\vec{x}\,;\vec{y}) = y_{j-k}$	$(j \in \{k+1, \dots, l+k\})$
	$C(;\epsilon, y, z_0, z_1) = y$	
	$C(;xi, y, z_0, z_1) = z_i$	$(i = 0, 1)$
	$O(\vec{x}\,;\vec{y}) = \epsilon$	
Weak Safe Composition	$f(\vec{x}\,;\vec{y}) = h(x_{i_1}, \dots, x_{i_n}; \vec{g}(\vec{x}\,;\vec{y}))$	
Safe Recursion on Notation	$f(\epsilon, \vec{x}\,;\vec{y}) = g(\vec{x}\,;\vec{y})$	
	$f(zi, \vec{x}\,;\vec{y}) = h_i(z, \vec{x}\,;\vec{y}, f(z, \vec{x}\,;\vec{y}))$	$(i = 0, 1)$

Figure 1: Defining initial functions and operations for $\mathcal{B}_{\mathsf{wsc}}$

can be formed from arbitrary constructors. We allow arbitrary deep right-hand sides, and implicit casting from normal to safe arguments. Still the main principle underlying $\mathcal{B}_{\mathsf{wsc}}$ remains reflected.

The remainder of the paper is organised as follows. After giving some preliminaries, Section 3 introduces the order sPOP*. Here we also prove correctness of sPOP* with respect to runtime complexity analysis. In Section 4 we incorporate parameter substitution into the order sPOP*. In Section 5 we then show that these orders are complete for FP, in particular we precisely relate sPOP* to the class $\mathcal{B}_{\mathsf{wsc}}$. In total we obtain an order-theoretic characterisation of FP. Exploiting the fine-grained control given by the degree of recursion, in Section 6 we provide an order-theoretic characterisation of $\mathsf{DTIME}(n^d)$. Finally in Section 7 and 8 we clarify the expressiveness of the established small polynomial path orders and conclude.

2. Preliminaries

We denote by $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, \dots\}$. For a finite *alphabet* $\mathbb{A}$ of *characters*, we denote by $\mathcal{W}(\mathbb{A})$ the set of *words over* $\mathbb{A}$, the empty word is denoted by ε. Let R be a binary relation. We denote by R^+ the transitive, by R^* the transitive and reflexive closure, and R^n denotes for $n \in \mathbb{N}$ the n-fold composition of R. We write $a\ R\ b$ for $(a, b) \in R$, the relation R is *well-founded* if there exists no infinite sequence $a_1\ R\ a_2\ R\ a_3\ R\ \dots$. The relation R is a *preorder* if it is transitive and reflexive, it is a *strict partial order* if it is irreflexive, antisymmetric and transitive, and R is an equivalence relation if it is reflexive, symmetric and transitive. Note that the transitive and reflexive closure of an order R (on a set S) gives always a preorder. Consider a preorder $\succcurlyeq$. Define $a \sim b$ if $a \succcurlyeq b$ and $b \succcurlyeq a$. Then this equivalence defines a partitioning of $\succcurlyeq$ into the equivalence $\sim$ and a strict partial order $\succ$.

2.1. Term Rewriting

We assume at least nodding acquaintance with the basics of term rewriting [28]. We fix a countably infinite set of *variables* $\mathcal{V}$ and a finite set of *function symbols* $\mathcal{F}$, the *signature*. For each $f \in \mathcal{F}$, the *arity* of f is fixed. The set of terms formed from $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is called *ground* if it contains no variables. The set of ground terms is indicated by $\mathcal{T}(\mathcal{F})$. The signature $\mathcal{F}$ contains a distinguished set of *constructors* $\mathcal{C} \subseteq \mathcal{F}$, elements of $\mathcal{T}(\mathcal{C}) \subseteq \mathcal{T}(\mathcal{F})$ are called *values*. Elements of $\mathcal{F}$ that are not constructors are called *defined symbols* and collected in $\mathcal{D}$. If not mentioned otherwise we denote by x, y, z variables, $f, g, h, \dots$ denote defined symbols. Terms are denoted by l, r or s, t, and values by u, v, w. All denotations are possibly followed by subscripts. We use the notation $\vec{s}$ to abbreviate a finite sequence of terms $s_1, \dots, s_n$.

The root symbol of term t is denoted as $\mathsf{rt}(t)$. The *size* of t is denoted by $|t|$ and refers to the number of occurrences of symbols t, the depth $\mathsf{dp}(t)$ is given recursively by $\mathsf{dp}(t) = 1$ if $t \in \mathcal{V}$, and $\mathsf{dp}(f(t_1, \dots, t_n)) = 1 + \max\{\mathsf{dp}(t_i) \mid i = 1, \dots, n\}$. Here we employ the convention that the maximum of an empty set is equal to 0. A *rewrite rule* is a pair (l, r) of terms, in notation $l \to r$, such that the *left-hand* side $l = f(l_1, \dots, l_n)$

is not a variable, the *root* f is defined, and all variables appearing in the *right-hand* r occur also in l. A *term rewrite system* (*TRS* for short) $\mathcal{R}$ is a set of rewrite rules.

We adopt *call-by-value* semantics and define the *rewrite relation* $\to_{\mathcal{R}}$ as follows.

$$(i)\ \frac{f(l_1,\dots,l_n) \to r \in \mathcal{R},\ \sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C})}{f(l_1\sigma,\dots,l_n\sigma) \to_{\mathcal{R}} r\sigma} \quad (ii)\ \frac{s \to_{\mathcal{R}} t}{f(\dots,s,\dots) \to_{\mathcal{R}} f(\dots,t,\dots)}\ .$$

If $s \to_{\mathcal{R}} t$ we say that s *reduces to* t in one step. For (i) we make various assumptions on $\mathcal{R}$: we suppose that there is exactly one *matching* rule $f(l_1,\dots,l_n) \to r \in \mathcal{R}$; the arguments l_i $(i = 1,\dots,n)$ contains no defined symbols; and variables occur only once in $f(l_1,\dots,l_n)$. That is, throughout this paper we fix $\mathcal{R}$ to denote a *completely defined*,[2] *orthogonal constructor* TRS [28]. Furthermore we are only concerned with *innermost* rewriting. Note that orthogonality enforces that our model of computation is deterministic.[3] If a term t has a normal form, then this term is unique and denoted by $t{\downarrow}$. For every n-ary defined symbol $f \in \mathcal{D}$, $\mathcal{R}$ defines a partial function $[\![f]\!] : \mathcal{T}(\mathcal{C})^n \to \mathcal{T}(\mathcal{C})$ where

$$[\![f]\!](u_1,\dots,u_n) := f(u_1,\dots,u_n){\downarrow} \quad \text{if } f(u_1,\dots,u_n){\downarrow} \in \mathcal{T}(\mathcal{C})\ ,$$

and $[\![f]\!](u_1,\dots,u_n)$ is undefined otherwise. Note that when $\mathcal{R}$ is *terminating*, i.e. when $\to_{\mathcal{R}}$ is well-founded, the function $[\![f]\!]$ is total.

Following [29] we adopt a unitary cost model. Bounds are of course expressed with respect to the size of terms. Let $\mathcal{T}_{\mathsf{b}}(\mathcal{F})$ denote the set of *basic* (also called *constructor based*) terms $f(u_1,\dots,u_n)$ where $f \in \mathcal{D}$ and $u_1,\dots,u_n \in \mathcal{T}(\mathcal{C})$. We define the *(innermost) runtime complexity function* $\mathrm{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N}$ as

$$\mathrm{rc}_{\mathcal{R}}(n) := \max\{\ell \mid \exists s \in \mathcal{T}_{\mathsf{b}}(\mathcal{F}), |s| \leqslant n \text{ and } s = t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_\ell\}\ .$$

Hence $\mathrm{rc}_{\mathcal{R}}(n)$ maximises over the *derivation height* of terms s of size up to n, regarding only basic terms. The latter restriction accounts for the fact that computations start only from basic terms. The runtime complexity function is well-defined if $\mathcal{R}$ is terminating. If $\mathrm{rc}_{\mathcal{R}}$ is asymptotically bounded from above by a polynomial, we simply say that the runtime of $\mathcal{R}$ is polynomially bounded. This unitary cost model is reasonable:

Proposition 1 (Adequacy Theorem [30–32])**.** *All functions $[\![f]\!]$ computed by $\mathcal{R}$ are computable on a conventional models of computation, viz Turing machines, such that the time complexity on the latter is polynomially related to* $\mathrm{rc}_{\mathcal{R}}$*.*

In particular, if the runtime of $\mathcal{R}$ is polynomially bounded then $[\![f]\!]$ is polytime computable on a Turing machine for all $f \in \mathcal{D}$.

We say that a function symbol f is *defined based on* g, in notation $f \blacktriangleright_{\mathcal{R}} g$, if there exists a rewrite rule $f(l_1,\dots,l_n) \to r \in \mathcal{R}$ where g occurs in r. We call f *recursive* if $f \blacktriangleright^+_{\mathcal{R}} f$ holds, i.e. if f is defined based on itself. Recursive functions symbols are collected in $\mathcal{D}_{\mathsf{rec}} \subseteq \mathcal{D}$. Noteworthy our notion also captures mutual recursion. We denote by $\geqslant$ the least preorder on $\mathcal{F}$ containing $\blacktriangleright_{\mathcal{R}}$ and where constructors are equivalent, i.e. $c \geqslant d$ and $d \geqslant c$ for all constructors $c, d \in \mathcal{C}$. The preorder $\geqslant$ is called the *precedence* of $\mathcal{R}$. We denote by $>$ and $\sim$ the separation of $\geqslant$ into the strict partial order $>$ and the equivalence $\sim$. Note that for $f \sim g$, if $f \in \mathcal{C}$ then also $g \in \mathcal{C}$; similar $f \in \mathcal{D}_{\mathsf{rec}}$ implies $g \in \mathcal{D}_{\mathsf{rec}}$. The *rank* of $f \in \mathcal{F}$ with respect to $\geqslant$ is inductively defined by $\mathsf{rk}(f) = 1 + \max\{\mathsf{rk}(g) \mid f > g\}$. The *depth of recursion* $\mathsf{rd}(f)$ of $f \in \mathcal{F}$ is defined in correspondence to the rank, but only takes recursive symbols into account: let $d = \max\{\mathsf{rd}(g) \mid f > g\}$ be the maximal recursion depth of a function symbol g underlying the definition of f; then $\mathsf{rd}(f) := 1 + d$ if f is recursive, otherwise $\mathsf{rd}(f) := d$.

[2] The restriction is not necessary, but simplifies our presentation, compare [9].

[3] As in [9] it is possible to adopt nondeterministic semantics, dropping orthogonality.

Example 1. Consider following TRS $\mathcal{R}_{\mathsf{arith}}$, written in predicative notation.

$$\begin{array}{lll} 1\colon\ +(\mathsf{0}\,;y)\to y & 3\colon\ +(\mathsf{s}(\,;x)\,;y)\to \mathsf{s}(+(x\,;y)) & 5\colon\ \mathsf{f}(x,y\,;)\to +(x\,;\times(y,y\,;)) \\ 2\colon\ \times(\mathsf{0},y\,;)\to \mathsf{0} & 4\colon\ \times(\mathsf{s}(\,;x),y\,;)\to +(y\,;\times(x,y\,;))\,. & \end{array}$$

The TRS $\mathcal{R}_{\mathsf{arith}}$ follows along the line of $\mathcal{B}_{\mathsf{wsc}}$ from Fig. 1. The functions $[\![+]\!]$ and $[\![\times]\!]$ denote addition and multiplication on natural numbers, in particular $[\![\mathsf{f}]\!](\mathsf{s}^m(\mathsf{0}),\mathsf{s}^n(\mathsf{0}))=\mathsf{s}^r(\mathsf{0})$ where $r=m+n^2$. The precedence is given by $\mathsf{f}>\times>+>\mathsf{s}\sim\mathsf{0}$ where addition $(+)$ and multiplication $(\times)$ are recursive, but f is not recursive. We have $\mathsf{rd}(+)=1$ since addition is recursive, as f is not recursive but multiplication is recursive we have $\mathsf{rd}(\mathsf{f})=\mathsf{rd}(\times)=2$.

2.2. Register Machines

In this paper, we are considering *register machine* (*RM* for short) over words $\mathcal{W}(\mathbb{A})$ as initially proposed by Shepherdson and Sturgis in [33]. We employ following notations and conventions. A RM M consists of a *finite* set of *registers* that store words over $\mathcal{W}(\mathbb{A})$. Like values, i.e. constructor terms, words are denoted by u,v,w, and $\vec{u},\vec{v},\vec{w}$ denote sequences of words. No confusion can arise from this. For r a register, we use $\langle r\rangle$ to refer to the content of register r. The *control* of M consists of a finite sequences of (labeled) *instructions* $I_1;I_2;\ldots;I_l$ which are executed sequentially by M. Here an instruction can be one of the following:

(i) *Append instruction* $A^{(a)}(r)$: place $a\in\mathbb{A}$ on the left-hand end of $\langle r\rangle$;
(ii) *Delete instruction* $D(r)$: remove the left-most character from $\langle r\rangle$, if $\langle r\rangle\neq\varepsilon$;
(iii) *Conditional jump instruction* $J^{(a)}(r)[j]$: jump to instruction I_j, if the left-most character of $\langle r\rangle$ is $a\in\mathbb{A}$, otherwise proceed with the next instruction;
(iv) *Copy instruction* $C(r,r')$: overwrite $\langle r'\rangle$ by $\langle r\rangle$.

Our definition departs from [33] in following minor respects. Unlike in [33], we suppose that the set of registers is finite. This simplification does not impose any restrictions. Due to the absence of memory indirection instructions, only a fixed number of registers can be accessed by a machine M anyway. The instructions (i)–(iii) correspond to the *minimal* instruction set given in [33, Section 6], with the difference that in [33] the instruction (i) appends to the right. The additional copy instruction (iv) added from the extended instruction set of [33, Section 2] ensures that copying words has unitary cost. A configuration of the RM M is a tuple $\langle j,w_1,\ldots,w_m\rangle$ where $w_1,\ldots,w_m\in\mathcal{W}(\mathbb{A})$ are the content of the m registers and j ranges over the *labels* $1,\ldots,l$ of instructions $I_1,\ldots,I_l$ of M, and the dedicated *halting label* $l+1$. We denote by $\to_M$ the one-step transition relation obtained in the obvious way from our informal account of the instruction set (i)–(iv). For the halting label $l+1$ we set $\langle l+1,u_1,\ldots,u_m\rangle\to_M\langle l+1,u_1,\ldots,u_m\rangle$ for all words u_i $(i=1,\ldots,m)$. We say that the RM M *computes* the (partial) function $f_M\colon\mathcal{W}(\mathbb{A})^k\to\mathcal{W}(\mathbb{A})$ with $k\leqslant m$ defined as follows:

$$f_M(u_1,\ldots,u_k):=v_m\quad:\Leftrightarrow\quad\exists\ell.\langle 1,u_1,\ldots,u_k,\vec{\varepsilon}\rangle\to_M^{\ell}\langle l+1,v_1,\ldots,v_m\rangle\,.$$

We also say that on inputs $u_1,\ldots,u_k$ the computation halts in ℓ steps. Denote by $|u|$ the length, or *size*, of the word u. Extend this to $\vec{u}=u_1,\ldots,u_k$ so that $|\vec{u}|=\sum_{i=1}^k|u_i|$ denotes the sum of the sizes of $\vec{u}$. Let $d\in\mathbb{N}$. We denote by $\mathsf{DTIME}(n^d)$ the class of functions $f\colon\mathcal{W}(\mathbb{A})^k\to\mathcal{W}(\mathbb{A})$ computed by some RM M in the above way, where M halts on all inputs $\vec{u}$ in no more than $\mathsf{O}(|\vec{u}|^d)$ steps.

3. The Small Polynomial Path Order

We arrive at the formal definition of the *small polynomial path order* (*sPOP** for short). Conceptually this order is a tamed recursive path order with product status, embodying *predicative analysis* of recursion set forth by Bellantoni and Cook [1].

Throughout this section, fix a TRS $\mathcal{R}$. For each function symbol f, we assume an a priori separation of argument positions into *normal* and *safe* ones. Arguments under normal positions play the rôle of recursion parameters, whereas safe argument positions allow the substitution of recursive results, compare

the definition of $\mathcal{B}_{\mathsf{wsc}}$ drawn in Fig. 1 on page 4. For constructors c we fix that all argument positions are safe. As in Example 1, we indicate this separation directly in terms and write $f(\vec{s}\,;\vec{t})$ where the arguments $\vec{s}$ to the left of the semicolon are normal, the remaining arguments $\vec{t}$ are safe. This separation and the precedence $\geqslant$ underlying the analysed TRS $\mathcal{R}$ *induces* an *instance* of sPOP*, which is denoted by $>_{\mathsf{spop}*}$ below.

In order to define $>_{\mathsf{spop}*}$, we introduce some auxiliary relations. First of all, we lift equivalence $\sim$ underlying the precedence $\geqslant$ of $\mathcal{R}$ to terms, disregarding the order on arguments: s and t are *equivalent*, in notation $s \sim t$, if $s = t$, or $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_n)$ where $f \sim g$ and there exists a permutation π on argument positions $\{1, \ldots, n\}$ such that $s_i \sim t_{\pi(i)}$ for all $i = 1, \ldots, n$. *Safe equivalence* $\overset{\bullet}{\sim} \subseteq \sim$ takes also the separation of argument positions into account. In the definition of $s \overset{\bullet}{\sim} t$, we additionally require that i is a normal argument position of f if and only if $\pi(i)$ is normal argument position of g. We emphasise that $\sim$ (and consequently $\overset{\bullet}{\sim}$) preserves values: if $s \sim t$ and $s \in \mathcal{T}(\mathcal{C})$ then $t \in \mathcal{T}(\mathcal{C})$. We extend the subterm relation to term equivalence. Consider $s = f(s_1, \ldots, s_k\,; s_{k+1}, \ldots, s_{k+l})$. Define $s \trianglerighteq\!\!/_{\sim} t$ if either $s \sim t$ or $s \triangleright\!\!/_{\sim} t$, where $s \triangleright\!\!/_{\sim} t$ holds if $s_i \trianglerighteq\!\!/_{\sim} t$ for some argument s_i of s $(i = 1, \ldots k + l)$. We denote by $\triangleright^{\mathsf{n}}\!\!/_{\sim}$ the restriction of $\triangleright\!\!/_{\sim}$ where only normal arguments are considered: $s \triangleright^{\mathsf{n}}\!\!/_{\sim} t$ if $s_i \trianglerighteq\!\!/_{\sim} t$ for some *normal* argument position $i \in \{1, \ldots, k\}$.

Definition 1. Let s and t be terms such that $s = f(s_1, \ldots, s_k\,; s_{k+1}, \ldots, s_{k+l})$. Then $s >_{\mathsf{spop}*} t$ if one of the following alternatives holds.

1. $s_i \geqslant_{\mathsf{spop}*} t$ for some argument s_i of s $(i \in \{1, \ldots, k + l\})$.
2. $f \in \mathcal{D}$, $t = g(t_1, \ldots, t_m\,; t_{m+1}, \ldots, t_{m+n})$ where $f > g$ and the following holds:
   - $s \triangleright^{\mathsf{n}}\!\!/_{\sim} t_j$ for all normal arguments $t_1, \ldots, t_m$;
   - $s >_{\mathsf{spop}*} t_j$ for all safe arguments $t_{m+1}, \ldots, t_{m+n}$;
   - t contains at most one (recursive) function symbol h with $f \sim h$.
3. $f \in \mathcal{D}_{\mathsf{rec}}$, $t = g(t_1, \ldots, t_k\,; t_{k+1}, \ldots, t_{k+l})$ where $f \sim g$ and the following holds:
   - $\langle s_1, \ldots, s_k\rangle >_{\mathsf{spop}*} \langle t_{\pi(1)}, \ldots, t_{\pi(k)}\rangle$ for some permutation π on $\{1, \ldots, k\}$;
   - $\langle s_{k+1}, \ldots, s_{k+l}\rangle \geqslant_{\mathsf{spop}*} \langle t_{\tau(k+1)}, \ldots, t_{\tau(k+l)}\rangle$ for some permutation τ on $\{k + 1, \ldots, k + l\}$.

Here $s \geqslant_{\mathsf{spop}*} t$ denotes that either $s \overset{\bullet}{\sim} t$ or $s >_{\mathsf{spop}*} t$ holds. In the last clause we use $>_{\mathsf{spop}*}$ also for the extension of $>_{\mathsf{spop}*}$ to products: $\langle s_1, \ldots, s_n\rangle \geqslant_{\mathsf{spop}*} \langle t_1, \ldots, t_n\rangle$ means $s_i \geqslant_{\mathsf{spop}*} t_i$ for all $i = 1, \ldots, n$, and $\langle s_1, \ldots, s_n\rangle >_{\mathsf{spop}*} \langle t_1, \ldots, t_n\rangle$ indicates that additionally $s_{i_0} >_{\mathsf{spop}*} t_{i_0}$ holds for at least one $i_0 \in \{1, \ldots, n\}$.

Throughout the following, we write $s >^{\langle \mathsf{i}\rangle}_{\mathsf{spop}*} t$ if $s >_{\mathsf{spop}*} t$ follows from the i^{th} clause in Definition 1. A similar notation is employed for the consecutive introduced orders.

We say that the TRS $\mathcal{R}$ is *compatible* with $>_{\mathsf{spop}*}$ if all rules are *oriented* from left to right: $l >_{\mathsf{spop}*} r$ for all rules $l \to r \in \mathcal{R}$. As sPOP* forms a restriction of the recursive path order, compatibility with sPOP* implies termination [28].

Definition 2. We call the TRS $\mathcal{R}$ *predicative recursive (of degree d)* if $\mathcal{R}$ is compatible with an instance of sPOP* and the maximal recursion depth $\mathsf{rd}(f)$ of $f \in \mathcal{F}$ is d.

Consider the orientation of a rule $f(l_1, \ldots, l_n) \to r \in \mathcal{R}$. The case $>^{\langle 2\rangle}_{\mathsf{spop}*}$ is intended to capture functions f defined by weak safe composition (**WSC**), compare Fig. 1. In particular the use of $\triangleright^{\mathsf{n}}\!\!/_{\sim}$ allows only the substitution of normal arguments of f in normal argument positions of g. The last restriction put onto $>^{\langle 2\rangle}_{\mathsf{spop}*}$ is used to prohibit multiple recursive calls. If one drops this restriction, the TRS consisting of $\mathsf{f}(\mathsf{0}\,;) \to \mathsf{0}$ and $\mathsf{f}(\mathsf{s}(\,;x)\,;) \to \mathsf{c}(\,;\mathsf{f}(x\,;),\mathsf{f}(x\,;))$ is compatible with sPOP* but its runtime complexity can be only exponentially bounded. Finally, $>^{\langle 3\rangle}_{\mathsf{spop}*}$ accounts for recursive calls, in combination with $>^{\langle 2\rangle}_{\mathsf{spop}*}$ we capture safe recursion (**SRN**). The next theorem provides our second main result.

Theorem 1. *Suppose $\mathcal{R}$ is a predicative recursive TRS of degree d. Then the derivation height of any basic term $f(\vec{u}\,;\vec{v})$ is bounded by a polynomial of degree $\mathsf{rd}(f)$ in the sum of the depths of normal arguments $\vec{u}$. In particular, the runtime complexity function $\mathrm{rc}_{\mathcal{R}}$ is bounded by a polynomial of degree d.*

As a consequence of Theorem 1 and the adequacy theorem (c.f. Proposition 1), any predicative recursive (and orthogonal) TRS $\mathcal{R}$ computes a function from FP. We remark that Theorem 1 remains valid for the standard notion of innermost rewriting [28] on constructor TRSs. Neither orthogonality nor our fixed call-by-value reduction strategy is essential, compare [9].

We continue with an informal account of Definition 1 in our running example, the admittedly technical proof is shortly postponed.

Example 2 (Example 1 continued). We show that the TRS $\mathcal{R}_{\mathsf{arith}}$ depicted in Example 1 is predicative recursive. Recall that the precedence underlying $\mathcal{R}_{\mathsf{arith}}$ is given by $\mathsf{f} > \times > + > \mathsf{s} \sim \mathsf{0}$, and that $\mathcal{D}_{\mathsf{rec}} = \{\times, +\}$. The degree of recursion of $\mathcal{R}_{\mathsf{arith}}$ is thus two.

The case $>^{\langle 1\rangle}_{\mathsf{spop}*}$ is standard in recursive path orders and allows the treatment of projections as in rules 1 and 2. We have $+(\mathsf{0}\,;y) >^{\langle 1\rangle}_{\mathsf{spop}*} y$ using $y \overset{\mathsf{s}}{\sim} y$ and likewise $\times(\mathsf{0}, y\,;) >^{\langle 1\rangle}_{\mathsf{spop}*} \mathsf{0}$ using $\mathsf{0} \overset{\mathsf{s}}{\sim} \mathsf{0}$. Observe that the rule

$$5:\ \mathsf{f}(x, y\,;) \to +(x\,;\times(y, y\,;))\ ,$$

is oriented by $>^{\langle 2\rangle}_{\mathsf{spop}*}$ only: using $\mathsf{f} > \times$ and twice $\mathsf{f}(x, y\,;) \rhd^{\mathsf{n}}_{\sim} y$, i.e., y is a normal argument of $\mathsf{f}(x, y\,;)$, we have $\mathsf{f}(x, y\,;) >^{\langle 2\rangle}_{\mathsf{spop}*} \times(y, y\,;)$. Using that also $\mathsf{f} > +$ and $\mathsf{f}(x, y\,;) \rhd^{\mathsf{n}}_{\sim} x$ holds, another application of $>^{\langle 2\rangle}_{\mathsf{spop}*}$ orients rule 5.

Finally, consider the recursive cases of addition (rule 3) and multiplication (rule 4). These can be oriented by a combination of $>^{\langle 2\rangle}_{\mathsf{spop}*}$ and $>^{\langle 3\rangle}_{\mathsf{spop}*}$. We exemplify this on the rule

$$4:\ \times(\mathsf{s}(\,;x), y\,;) \to +(y\,;\times(x, y\,;))\ .$$

Employing $\times > +$, case $>^{\langle 2\rangle}_{\mathsf{spop}*}$ is applicable. Thus orientation of this rule simplifies to $\times(\mathsf{s}(\,;x), y\,;) \rhd^{\mathsf{n}}_{\sim} y$ and $\times(\mathsf{s}(\,;x), y\,;) >_{\mathsf{spop}*} \times(x, y\,;)$. The former constraint is satisfied by definition. Since $\times$ is recursive, using $>^{\langle 3\rangle}_{\mathsf{spop}*}$ the latter constraint reduces to $\langle \mathsf{s}(\,;x), y\rangle >_{\mathsf{spop}*} \langle x, y\rangle$ and the trivial constraint $\langle\rangle \geqslant_{\mathsf{spop}*} \langle\rangle$. Clearly $\langle \mathsf{s}(\,;x), y\rangle >_{\mathsf{spop}*} \langle x, y\rangle$ holds as $\mathsf{s}(\,;x) >^{\langle 1\rangle}_{\mathsf{spop}*} x$ and $y \overset{\mathsf{s}}{\sim} y$. Hence we are done.

Note that any other partitioning of argument positions of multiplication invalidates the orientation of rule 4. The sub-constraint $\times(\mathsf{s}(\,;x), y\,;) >_{\mathsf{spop}*} \times(x, y\,;)$ requires that at least the first argument position of times is normal, the sub-constraint $\times(\mathsf{s}(\,;x), y\,;) \rhd^{\mathsf{n}}_{\sim} y$ enforces that also the second parameter is normal. The order thus determines that multiplication performs recursion on its first arguments, and that the second parameter should be considered normal since it is used as recursion parameter in addition. Reconsidering the orientation of rule 5, the use of $\rhd^{\mathsf{n}}_{\sim}$ propagates that f takes only normal arguments.

By Theorem 1 we obtain that addition admits linear, and multiplication as well as f admits quadratic runtime complexity. Overall the runtime complexity of $\mathcal{R}_{\mathsf{arith}}$ is quadratic.

The following examples clarifies the need for data tiering.

Example 3 (Example 2 continued). Consider the extension of $\mathcal{R}_{\mathsf{arith}}$ by the two rules

$$6 : \mathsf{exp}(\mathsf{0}, y) \to \mathsf{s}(\,;\mathsf{0}) \qquad 7 : \mathsf{exp}(\mathsf{s}(\,;x), y) \to \times(y, \mathsf{exp}(x, y)\,;)\ ,$$

that express exponentiation y^x in an exponential number of steps. The definition of exp is not predicative recursive, since the recursive result $\mathsf{exp}(x, y)$ is substituted as recursion parameter to multiplication. For this reason the orientation with $>_{\mathsf{spop}*}$ fails.

The next example is negative, in the sense that the considered TRSs admits polynomial runtime complexity, but fails to be compatible with sPOP*.

Example 4 (Example 3 continued). Consider now the TRS $\mathcal{R}_{\mathsf{arith}}$ where the rule 4 is replaced by the rule

$$4\mathrm{a}:\ \times(\mathsf{s}(\,;x), y\,;) \to +(\times(x, y\,;)\,;y)\ .$$

The resulting system admits polynomial runtime complexity. On the other hand, Theorem 1 is inapplicable since the system is not predicative recursive.

$$\begin{array}{ccccccc} s & \to_{\mathcal{R}} & s_1 & \to_{\mathcal{R}} & \cdots & \to_{\mathcal{R}} & s_\ell \\ \downarrow & & \downarrow & & & & \downarrow \\ \mathsf{S}(s) & \succcurlyeq_\ell & \mathsf{S}(s_1) & \succcurlyeq_\ell & \cdots & \succcurlyeq_\ell & \mathsf{S}(s_\ell) \end{array}$$

Figure 2: Predicative Embedding of $\to_{\mathcal{R}}$ into $\succcurlyeq_\ell$

We emphasise that the bound provided in Theorem 1 is tight, in the sense that for any d we can define a predicative TRS $\mathcal{R}_d$ of degree d admitting runtime complexity $\Omega(n^d)$.

Example 5. We define a family of TRSs $\mathcal{R}_d$ ($d \in \mathbb{N}$) inductively as follows: $\mathcal{R}_0 := \{\mathsf{f}_0(x\,;) \to \mathsf{a}\}$ and $\mathcal{R}_{d+1}$ extends $\mathcal{R}_d$ by the rules

$$\mathsf{f}_{d+1}(x\,;) \to \mathsf{g}_{d+1}(x, x\,;) \qquad \mathsf{g}_{d+1}(\mathsf{s}(\,;x), y\,;) \to \mathsf{b}(\,;\mathsf{f}_d(y\,;), \mathsf{g}_{d+1}(x, y\,;))\,.$$

Let $d \in \mathbb{N}$. It is easy to see that $\mathcal{R}_d$ is predicative recursive (with underlying precedence $\mathsf{f}_d > \mathsf{g}_d > \mathsf{f}_{d-1} > \mathsf{g}_{d-1} > \ldots > \mathsf{f}_0 > \mathsf{a} \sim \mathsf{b}$). As only g_i ($i = 1, \ldots, d$) are recursive, the recursion depth of $\mathcal{R}_d$ is d.

But also the runtime complexity of $\mathcal{R}_d$ is in $\Omega(n^d)$: For $d = 0$ this is immediate. Otherwise, consider the term $\mathsf{f}_{d+1}(\mathsf{s}^n(\,;\mathsf{a})\,;)$ ($n \in \mathbb{N}$) which reduces to $\mathsf{g}_{d+1}(\mathsf{s}^n(\,;\mathsf{a}), \mathsf{s}^n(\,;\mathsf{a})\,;)$ in one step. As the latter iterates $\mathsf{f}_d(\mathsf{s}^n(\mathsf{a}))$ for n times, the lower bound is established by inductive reasoning.

We now show that sPOP* is correct, i.e. we prove Theorem 1. Suppose $\mathcal{R}$ is a predicative recursive TRS of degree d. Our proof makes use of a variety of ingredients. In Definition 3 we define a *predicative interpretation* S that flatten terms to *sequences of terms*, separating safe from normal arguments. In Definition 4 we introduce a family of orders $(\succcurlyeq_\ell)_{\ell\in\mathbb{N}}$ on sequences of terms. The definition of $\succcurlyeq_\ell$ (for fixed ℓ) does not explicitly mention predicative notions and is conceptually simpler than $>_{\mathsf{spop}^*}$. In Lemma 4 we show that predicative interpretations S embeds rewrite steps into $\succcurlyeq_\ell$, as pictured in Fig. 2. Consequently the derivation height of s is bounded by the length of $\succcurlyeq_\ell$ descending sequences, which in turn can be bounded sufficiently whenever s is basic (cf. Lemma 7).

Consider a step $C[f(\vec{u}\sigma\,;\vec{v}\sigma)] \to_{\mathcal{R}} C[r\sigma] = t$. Due to the limitations imposed by $>_{\mathsf{spop}^*}$, it is not difficult to see that if $r\sigma$ is not a value itself, then at least all normal arguments are values. We capture this observation in the set $\mathcal{T}_{\mathsf{b}}^{\to}$, defined as the least set such that (i) $\mathcal{T}(\mathcal{C}) \subseteq \mathcal{T}_{\mathsf{b}}^{\to}$, and (ii) if $f \in \mathcal{F}$, $\vec{v} \subseteq \mathcal{T}(\mathcal{C})$ and $\vec{t} \subseteq \mathcal{T}_{\mathsf{b}}^{\to}$ then $f(\vec{v}\,;\vec{t}) \in \mathcal{T}_{\mathsf{b}}^{\to}$. This set is closed under rewriting.

Lemma 1. *Suppose $\mathcal{R}$ is compatible with $>_{\mathsf{spop}^*}$. If $s \in \mathcal{T}_{\mathsf{b}}^{\to}$ and $s \to_{\mathcal{R}} t$ then $t \in \mathcal{T}_{\mathsf{b}}^{\to}$.*

Proof. The lemma follows by a straightforward inductive argument on Definition 1. □

Observe that $\mathcal{T}_{\mathsf{b}}^{\to}$ includes all basic terms. For the runtime complexity analysis of $\mathcal{R}$, it thus suffices to consider reductions on $\mathcal{T}_{\mathsf{b}}^{\to}$ only.

3.1. Predicative Interpretation of Terms as Sequences

Predicative interpretations separate safe from normal arguments. To this avail, we define the *normalised signature* $\mathcal{F}_{\mathsf{n}}$ to contain all symbols from $\mathcal{F}$, with the sole difference that the arity of *defined symbols* f with k normal arguments is k in $\mathcal{F}_{\mathsf{n}}$. A term t is *normalised*, if $t \in \mathcal{T}(\mathcal{F}_{\mathsf{n}})$. Below we retain the separation into constructors, recursive and non-recursive symbols. As a consequence, the rank and recursion depth coincide with respect to both signatures, and also $\mathcal{T}(\mathcal{C}) \subseteq \mathcal{T}(\mathcal{F}_{\mathsf{n}})$. Terms $\mathcal{T}_{\mathsf{b}}(\mathcal{F}_{\mathsf{n}})$ are also called basic, these are obtained from $\mathcal{T}_{\mathsf{b}}(\mathcal{F})$ by dropping safe arguments.

To formalise sequences of (normalised) terms, we use an auxiliary variadic function symbol $\circ$. Here variadic means that the arity of $\circ$ is finite but arbitrary. We always write $[t_1\ \cdots\ t_n]$ for $\circ(t_1, \ldots, t_n)$, and if we write $f(t_1, \ldots, t_n)$ then $f \neq \circ$. We use $a, b, \ldots$ to denote terms *or* sequences of terms. In contrast, s, t, u, v, possibly followed by subscripts, denote terms which are not sequences. Abusing set-notation, we write $t \in [t_1\ \cdots\ t_n]$ if $t = t_i$ for some $i \in \{1, \ldots, n\}$. We lift terms equivalence to sequences by disregarding the order of elements: $[s_1\ \cdots\ s_n] \sim [t_1\ \cdots\ t_n]$ if $s_i \sim t_{\pi(i)}$ for all $i = 1, \ldots, n$ and some permutation π on

$\{1,\dots,n\}$. We denote by $a \frown b$ the *concatenation* of sequences. To avoid notational overhead we overload concatenation to both terms and sequences. For sequences a define $\mathrm{lift}(a) := a$, and for terms t define $\mathrm{lift}(t) := [\,t\,]$. We set $a \frown b := [\,s_1 \;\cdots\; s_m \;t_1 \;\cdots\; t_n\,]$ where $\mathrm{lift}(a) = [s_1 \;\cdots\; s_m]$ and $\mathrm{lift}(b) = [t_1 \;\cdots\; t_n]$.

Definition 3. We define the *predicative interpretation* S, mapping terms $t \in \mathcal{T}_{\mathsf{b}}^{\to}$ to sequences of normalised terms as follows:

$$\mathsf{S}(t) := \begin{cases} [\,] & \text{if } t \text{ is a value,} \\ [\,f(u_1,\dots,u_k)\,] \frown \mathsf{S}(t_{k+1}) \frown \cdots \frown \mathsf{S}(t_{k+l}) & \text{otherwise, where } t = f(u_1,\dots,u_k\,;t_{k+l},\dots,t_{k+1}). \end{cases}$$

Note that the predicative interpretation $\mathsf{S}(t)$ is a sequence of (normalised) basic terms for any term $t \in \mathcal{T}_{\mathsf{b}}^{\to}$. To get the reader prepared for the definition of the order $\succcurlyeq_\ell$ on sequences as defined below, we exemplify Definition 3 on a predicative recursive TRS.

Example 6. Consider following predicative recursive TRS $\mathcal{R}_{\mathsf{f}}$ where

$$1:\ \mathsf{f}(\mathsf{0}\,;y) \to y \qquad\qquad 2:\ \mathsf{f}(\mathsf{s}(x)\,;y) \to \mathsf{g}(x\,;\mathsf{f}(x\,;y))\ .$$

Consider a substitution $\sigma :\ \mathcal{V} \to \mathcal{T}(\mathcal{C})$. The embedding $\mathsf{S}(l\sigma) \succcurlyeq_\ell \mathsf{S}(r\sigma)$ of root steps ($l \to r \in \mathcal{R}_{\mathsf{f}}$) results in the following order constraints.

$$\begin{array}{rcrcll} \mathsf{S}(\mathsf{f}(\mathsf{0}\,;y\sigma)) & = & [\,\mathsf{f}(\mathsf{0})\,] \succcurlyeq_\ell [\,] & = & \mathsf{S}(y\sigma) & \text{by rule 1,} \\ \mathsf{S}(\mathsf{f}(\mathsf{s}(x\sigma)\,;y\sigma)) & = & [\,\mathsf{f}(\mathsf{s}(x\sigma))\,] \succcurlyeq_\ell \mathsf{g}(x\sigma) \frown \mathsf{f}(x\sigma) & = & \mathsf{S}(\mathsf{g}(x\sigma\,;\mathsf{f}(x\sigma\,;y\sigma))) & \text{by rule 2.} \end{array}$$

Kindly observe that in the first line we employed $\mathsf{S}(y\sigma) = [\,]$ because $y\sigma$ is a value. In the second line we tacitly employed the overloading of concatenation:

$$\mathsf{S}(\mathsf{g}(x\sigma\,;\mathsf{f}(x\sigma\,;y\sigma))) = [\,\mathsf{g}(x\sigma)\,] \frown \mathsf{S}(\mathsf{f}(x\sigma\,;y\sigma)) = [\,\mathsf{g}(x\sigma)\,] \frown [\,\mathsf{f}(x\sigma)\,] \frown [\,] = \mathsf{g}(x\sigma) \frown \mathsf{f}(x\sigma)\ .$$

Consider now a rewrite step $s \to_{\mathcal{R}} t$ below the root for $s \in \mathcal{T}_{\mathsf{b}}^{\to}$. As $s \in \mathcal{T}_{\mathsf{b}}^{\to}$, without loss of generality the rewrite step happens below a safe argument position. Hence

$$s = h(\vec{v}\,;s_1,\dots,s_i,\dots,s_l) \to_{\mathcal{R}} h(\vec{v}\,;s_1,\dots,t_i,\dots,s_l) = t$$

for some values $\vec{v}$, terms $s_1,\dots,s_l$ and $s_i \to_{\mathcal{R}} t_i$. To embed such rewrite steps we have to prove

$$[\,h(\vec{v})\,] \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(s_i) \frown \cdots \frown \mathsf{S}(s_l) \succcurlyeq_\ell [\,h(\vec{v})\,] \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(t_i) \frown \cdots \frown \mathsf{S}(s_l)\ .$$

We emphasise that for a root step $l\sigma \to_{\mathcal{R}} r\sigma$ of a predicative recursive TRS $\mathcal{R}$, the length of $\mathsf{S}(r\sigma)$ does not depend on σ, since images of σ are removed by the predicative interpretation. As a consequence, each step in an $\mathcal{R}$-derivation on $\mathcal{T}_{\mathsf{b}}^{\to}$ increases the length of predicative interpretations by a constant (depending on $\mathcal{R}$) only. Below, we bind this constant by the maximal size of a right-hand side in $\mathcal{R}$.

3.2. Small Polynomial Path Order on Sequences

We arrive at the definition of the order $\succcurlyeq_\ell$ on sequences. This order is used to orient images of the predicative interpretation S. The parameter $\ell \in \mathbb{N}$ in $\succcurlyeq_\ell$ controls the width of terms and sequences, and is crucial for the analysis of the length of $\succcurlyeq_\ell$-descending sequences carried out below.

Definition 4. Let $\geqslant$ denote a precedence. For all $\ell \geqslant 1$ we define $\succcurlyeq_\ell$ on terms and sequences of terms inductively such that:

1. $f(s_1,\dots,s_n) \succcurlyeq_\ell g(t_1,\dots,t_m)$ if $f \in \mathcal{D}$, $f > g$ and the following conditions hold:
   - $f(s_1,\dots,s_n) \not\sim t_j$ for all $j = 1,\dots,m$;
   - $m \leqslant \ell$.

2. $f(s_1,\dots,s_n) \succcurlyeq_\ell g(t_1,\dots,t_n)$ if $f \in \mathcal{D}_{\mathsf{rec}}$, $f \sim g$ and for some permutation π on $\{1,\dots,n\}$:
   - $\langle s_1,\dots,s_n\rangle \rhd\!\!\!/_\sim \langle t_{\pi(1)},\dots,t_{\pi(n)}\rangle$.
3. $f(s_1,\dots,s_n) \succcurlyeq_\ell [t_1 \ \cdots \ t_m]$ if the following conditions hold:
   - $f(s_1,\dots,s_n) \succcurlyeq_\ell t_j$ for all $j = 1,\dots,m$;
   - at most one element t_{j_0} ($j_0 \in \{1,\dots,m\}$) contains a symbols g with $f \sim g$;
   - $m \leqslant \ell$.
4. $[s_1 \ \cdots \ s_n] \succcurlyeq_\ell [t_1 \ \cdots \ t_m]$ if there exists terms *or* sequences b_i ($i = 1,\dots,n$) such that:
   - $[t_1 \ \cdots \ t_m] \sim b_1 \frown \cdots \frown b_n$;
   - $\langle s_1,\dots,s_n\rangle \succcurlyeq_\ell \langle b_1,\dots,b_n\rangle$.

We denote by $a \succcurlyeq_\ell b$ that either $a \sim b$ or $a \succcurlyeq_\ell b$ holds. We use $\rhd\!\!\!/_\sim$ and $\succcurlyeq_\ell$ also for their extension to products: $\langle s_1,\dots,s_n\rangle \rhd\!\!\!/_\sim \langle t_i,\dots,t_n\rangle$ if $s_i \trianglerighteq\!\!\!/_\sim t_i$ for all $i = 1,\dots,n$, and $s_{i_0} \rhd\!\!\!/_\sim t_{i_0}$ for at least one $i_0 \in \{1,\dots,n\}$; likewise $\langle s_1,\dots,s_n\rangle \succcurlyeq_\ell \langle t_i,\dots,t_n\rangle$ if $s_i \succcurlyeq_\ell t_i$ for all $i = 1,\dots,n$, and $s_{i_0} \succcurlyeq_\ell t_{i_0}$ for at least one $i_0 \in \{1,\dots,n\}$.

We point out that $\succcurlyeq_\ell$ misses the case: $f(s_1,\dots,s_n) \succcurlyeq_\ell t$ if $s_i \succcurlyeq_\ell t$ for some argument s_i. Since predicative interpretations remove values, the clause is not needed, compare the embedding of rule 1 given in Example 6. This case would invalidate the central Lemma 7 given below, which estimates the length of $\succcurlyeq_\ell$ descending sequences. Observe that on constructor based left-hand sides, the order constraints imposed by $\succcurlyeq_\ell^{\langle 1\rangle}$ and $\succcurlyeq_\ell^{\langle 2\rangle}$ translate to the order constraints imposed by $>_{\mathsf{spop}*}^{\langle 2\rangle}$ and $>_{\mathsf{spop}*}^{\langle 3\rangle}$ on *normal* arguments. The clauses $\succcurlyeq_\ell^{\langle 3\rangle}$ and $\succcurlyeq_\ell^{\langle 4\rangle}$ extend the order from terms to sequences. Noteworthy the second clause in $\succcurlyeq_\ell^{\langle 3\rangle}$ reflects that we do not allow multiple recursive calls, compare $>_{\mathsf{spop}*}^{\langle 2\rangle}$ and the definition of the predicative interpretation. We exercise Definition 4 on the constraints obtained in Example 6.

Example 7 (Example 6 continued). We show that the order constraints drawn in Example 6 can be resolved for $\ell = 2$. Let $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C})$ be a substitution. Consider first the root step $\mathsf{f}(\mathsf{0}\,;y\sigma) \to_{\mathcal{R}} y\sigma$ due to rule 1. Exploiting the shape of σ, we have $\mathsf{S}(\mathsf{f}(\mathsf{0}\,;y\sigma)) = [\,\mathsf{f}(\mathsf{0})\,] \succcurlyeq_\ell^{\langle 4\rangle} [\,] = \mathsf{S}(y\sigma)$. For the root step $\mathsf{f}(\mathsf{s}(x\sigma)\,;y\sigma) \to_{\mathcal{R}} \mathsf{g}(x\sigma\,;\mathsf{f}(x\sigma\,;y\sigma))$ caused by rule 2 we have

$$
\begin{array}{lrll}
1: & \mathsf{s}(x\sigma) \rhd\!\!\!/_\sim x\sigma & & \\
2: & \mathsf{f}(\mathsf{s}(x\sigma)) \rhd\!\!\!/_\sim x\sigma & & \text{by 1,} \\
3: & \mathsf{f}(\mathsf{s}(x\sigma)) \succcurlyeq_2^{\langle 1\rangle} \mathsf{g}(x\sigma) & & \text{if } \mathsf{f} > \mathsf{g}\text{, using 2,} \\
4: & \mathsf{f}(\mathsf{s}(x\sigma)) \succcurlyeq_2^{\langle 2\rangle} \mathsf{f}(x\sigma) & & \text{by 1,} \\
5: & \mathsf{f}(\mathsf{s}(x\sigma)) \succcurlyeq_2^{\langle 3\rangle} \mathsf{g}(x\sigma)\frown\mathsf{f}(x\sigma) & & \text{using 3 and 4,} \\
6: & \mathsf{S}(\mathsf{f}(\mathsf{s}(x\sigma)\,;y\sigma)) \;=\; [\,\mathsf{f}(\mathsf{s}(x\sigma))\,] \succcurlyeq_2^{\langle 4\rangle} \mathsf{g}(x\sigma)\frown\mathsf{f}(x\sigma) & = \mathsf{g}(x\sigma\,;\mathsf{f}(x\sigma\,;y\sigma)) & \text{using 5.}
\end{array}
$$

Note that $\mathsf{g}(x\sigma)\frown\mathsf{f}(x\sigma) = [\,\mathsf{g}(x\sigma)\ \mathsf{f}(x\sigma)\,]$ and thus $\ell = 2$ is needed in the proof step 5.

The next lemma collects frequently used properties of $\succcurlyeq_\ell$.

Lemma 2. *For all $\ell \geqslant 1$ we have:*

- $\succcurlyeq_\ell \subseteq \succcurlyeq_{\ell+1}$,
- $\sim \cdot \succcurlyeq_\ell \cdot \sim \subseteq \succcurlyeq_\ell$, *and*
- $a \succcurlyeq_\ell b$ *implies* $a\frown c \succcurlyeq_\ell b\frown c$.

Proof. All but the third property follow directly from definition. Suppose $a \succcurlyeq_\ell b$ holds, and let $\mathrm{lift}(c) = [r_1 \ \cdots \ r_l]$. We show $a \frown c \succcurlyeq_\ell b \frown c$. First suppose $a = f(s_1, \ldots, s_n)$. Then we conclude

$$a \frown c = [\, f(s_1, \ldots, s_n) \ r_1 \ \cdots \ r_l \,] \succcurlyeq_\ell^{\langle 4 \rangle} b \frown r_1 \frown \cdots \frown r_l = b \frown c$$

employing the assumption $a \succcurlyeq_\ell b$ and $r_i \sim r_i$ for all $i = 1, \ldots, l$. Otherwise $a = [s_1 \ \cdots \ s_n]$, hence $a \succcurlyeq_\ell^{\langle 4 \rangle} b$ by assumption. Then $b \sim b_1 \frown \cdots \frown b_n$ with $s_i \succcurlyeq_\ell b_i$ for all $i = 1, \ldots, n$, where at least one orientation is strict. From this and again using $r_i \sim r_i$ $(i = 1, \ldots, l)$ we conclude

$$a \frown c = [\, s_1 \ \cdots \ s_n \ r_1 \ \cdots \ r_l \,] \succcurlyeq_\ell^{\langle 4 \rangle} b_1 \frown \cdots \frown b_n \frown r_1 \frown \cdots \frown r_l = b \frown c \,.$$

□

We emphasise that as a consequence of Lemma 2 we have that $c_1 \frown a \frown c_2 \succcurlyeq_\ell c_1 \frown b \frown c_2$ holds whenever $a \succcurlyeq_\ell b$ holds. The order constraints on sequences are defined so that sequences purely act as containers. More precise, every $\succcurlyeq_\ell$-descending sequence starting from $[s_1 \ \cdots \ s_n]$ can be seen as a combination of possibly interleaved, but otherwise independent $\succcurlyeq_\ell$-descending sequences starting from the elements s_i $(i = 1, \ldots, n)$.

3.3. Predicative Embedding

We now close the diagram outlined in Fig. 2 on Page 9, that is we prove the predicative embedding exemplified in Example 7 on the TRS $\mathcal{R}_f$. As a preparatory step, we consider root steps $l\sigma \to_\mathcal{R} r\sigma$ first. The complete embedding is then established in Lemma 4.

Lemma 3. *Consider a rewrite rule $l \to r \in \mathcal{R}$. Let $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C})$ be a substitution. If $l >_{\mathsf{spop}^*} r$ holds then $\mathsf{S}(l\sigma) \succcurlyeq_{|r|} \mathsf{S}(r\sigma)$.*

Proof. Let $l = f(l_1, \ldots, l_m \,;\, l_{m+1}, \ldots, l_{m+n})$. We first show

$$l >_{\mathsf{spop}^*} r \implies f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|} \mathsf{S}(t) \text{ for all } t \in \mathsf{S}(r\sigma) \,, \tag{$*$}$$

by induction on $|r|$. The non-trivial case is when $r\sigma$ is not a value, otherwise $\mathsf{S}(r\sigma) = [\,]$. Suppose thus $r = g(r_1, \ldots, r_{m'} \,;\, r_{m'+1}, \ldots, r_{m'+n'})$ where r is not a value. By definition

$$\mathsf{S}(r\sigma) = [\, g(r_1\sigma, \ldots, r_{m'}\sigma) \,] \frown \mathsf{S}(r_{m'+1}\sigma) \frown \cdots \frown \mathsf{S}(r_{m'+n'}\sigma) \,.$$

First consider the element $g(r_1\sigma, \ldots, r_{m'}\sigma) \in \mathsf{S}(r\sigma)$. We either have $l >_{\mathsf{spop}^*}^{\langle 2 \rangle} r$ or $l >_{\mathsf{spop}^*}^{\langle 3 \rangle} r$ by the assumption that r is not a value. In the case $l >_{\mathsf{spop}^*}^{\langle 2 \rangle} r$, we have $f > g$ and $l \rhd^{\mathbf{n}}\!/_\sim r_j$ for all normal arguments r_j $(j = 1, \ldots, m')$. The latter reveals that the instances $r_j\sigma$ are equivalent to proper subterms of the left-hand side $f(l_1\sigma, \ldots, l_m\sigma)$. Using this and that trivially $m' \leqslant |r|$ holds we conclude $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|}^{\langle 1 \rangle} g(r_1\sigma, \ldots, r_{m'}\sigma)$. In the remaining case $l >_{\mathsf{spop}^*}^{\langle 3 \rangle} r$, we have $m' = m$, $f \sim g$ where $f, g \in \mathcal{D}_{\mathsf{rec}}$ and moreover $\langle l_1, \ldots, l_m \rangle >_{\mathsf{spop}^*} \langle r_{\pi(1)}, \ldots, r_{\pi(m)} \rangle$ for some permutation π. By reasoning as above we see $\langle l_1\sigma, \ldots, l_m\sigma \rangle \rhd\!/_\sim \langle r_{\pi(1)}\sigma, \ldots, r_{\pi(m)}\sigma \rangle$ and conclude $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|}^{\langle 2 \rangle} g(r_1\sigma, \ldots, r_{m'}\sigma)$. Hence overall we obtain $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|} g(r_1\sigma, \ldots, r_{m'}\sigma)$.

Now consider the remaining elements $t \in \mathsf{S}(r\sigma)$, where $t \neq g(r_1\sigma, \ldots, r_{m'}\sigma)$. Then each t occurs in the interpretation of a safe argument of $r\sigma$, say $t \in \mathsf{S}(r_j\sigma)$ for some $j \in \{m'+1, \ldots, m'+n'\}$. One verifies that, $l >_{\mathsf{spop}^*} r_j$ holds: if $l >_{\mathsf{spop}^*}^{\langle 2 \rangle} r$ then by definition, otherwise $l >_{\mathsf{spop}^*}^{\langle 3 \rangle} r$ holds and we obtain $l >_{\mathsf{spop}^*}^{\langle 1 \rangle} r_j$. By induction hypothesis we have $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r_j|} t$. As $\succcurlyeq_{|r_j|} \subseteq \succcurlyeq_{|r|}$ we hence obtain $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|} t$ for all $t \in \mathsf{S}(r_j\sigma)$ and safe positions $j \in \{m'+1, \ldots, m'+n'\}$ of g. This concludes $(*)$.

We return to the proof of the lemma. A standard induction gives that the length of $\mathsf{S}(r\sigma)$ is bounded by $|r|$, compare the remark after Example 6. Using that σ maps to values, a second induction on $l >_{\mathsf{spop}^*} r$ gives that $\mathsf{S}(r\sigma)$ contains at most one (defined) function symbol g equivalent to f. Summing up, using $(*)$ we conclude $f(l_1\sigma, \ldots, l_m\sigma) \succcurlyeq_{|r|}^{\langle 3 \rangle} \mathsf{S}(r\sigma)$. Observe that by assumption the direct subterms of $l\sigma$ are values, and thus $\mathsf{S}(l\sigma) = [\, f(l_1\sigma, \ldots, l_m\sigma) \,]$ by definition. The lemma thus follows by one application of $\succcurlyeq_{|r|}^{\langle 4 \rangle}$. □

Lemma 4. *Let $\mathcal{R}$ denote a predicative recursive TRS and let $\ell := \max\{|r| \mid l \to r \in \mathcal{R}\}$. If $s \in \mathcal{T}_{\mathsf{b}}^{\to}$ and $s \to_{\mathcal{R}} t$ then $\mathsf{S}(s) \succcurlyeq_\ell \mathsf{S}(t)$.*

Proof. Let $s \in \mathcal{T}_{\mathsf{b}}^{\to}$ and consider a rewrite step $s \to_{\mathcal{R}} t$. We prove the lemma by induction on the rewrite position. In the base case we consider a root step $s = l\sigma \to_{\mathcal{R}} r\sigma = t$ for some rule $l \to r \in \mathcal{R}$. Since $\mathcal{R}$ is predicative recursive, $l >_{\mathsf{spop}^*} r$ holds. By Lemma 3 we have $\mathsf{S}(l\sigma) \succcurlyeq_{|r|} \mathsf{S}(r\sigma)$. Since $|r| \leqslant \ell$ the result follows.

For the inductive step, consider a rewrite step below the root. Since $s \in \mathcal{T}_{\mathsf{b}}^{\to}$ this step is of the form

$$s = f(\vec{v}; s_1, \dots, s_i, \dots, s_n) \to_{\mathcal{R}} f(\vec{v}; s_1, \dots, t_i, \dots, s_n) = t \,,$$

where $s_i \to_{\mathcal{R}} t_i$ for some $i \in \{1, \dots, n\}$. Wlog. we assume t is not a value. Using the induction hypothesis $\mathsf{S}(s_i) \succcurlyeq_\ell \mathsf{S}(t_i)$ and Lemma 2 we conclude

$$\mathsf{S}(s\sigma) = f(\vec{v}) \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(s_i) \frown \cdots \frown \mathsf{S}(s_n) \succcurlyeq_\ell f(\vec{v}) \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(t_i) \frown \cdots \frown \mathsf{S}(s_n) = \mathsf{S}(t\sigma) \,.$$

□

3.4. Binding the Length of $\succcurlyeq_\ell$-Descending Sequences

The following function G_ℓ relates a term, or sequence of terms, to the length of its longest $\succcurlyeq_\ell$-descending sequence.

Definition 5. For all $\ell \geqslant 1$, we define $\mathsf{G}_\ell(a) := \max\{m \mid a \succcurlyeq_\ell a_1 \succcurlyeq_\ell \cdots \succcurlyeq_\ell a_m\}$.

This function is well-defined, as $\succcurlyeq_\ell$ is well-founded. The latter can be seen as $\succcurlyeq_\ell$ forms a restriction of the multiset path order [34]. We remark that due to Lemma 2, $\mathsf{G}_\ell(a) = \mathsf{G}_\ell(b)$ whenever $a \sim b$. The following lemma confirms that sequences act as containers only.

Lemma 5. *For all sequences $[s_1 \ \cdots \ s_n]$, $\mathsf{G}_\ell([s_1 \ \cdots \ s_n]) = \sum_{i=1}^n \mathsf{G}_\ell(s_i)$.*

Proof. Let $a = [s_1 \ \cdots \ s_n]$ be a sequence and observe $\mathsf{G}_\ell(a_1 \frown a_2) \geqslant \mathsf{G}_\ell(a_1) + \mathsf{G}_\ell(a_2)$. This is a consequence of Lemma 2. Hence, in particular we obtain: $\mathsf{G}_\ell(a) = \mathsf{G}_\ell(s_1 \frown \cdots \frown s_n) \geqslant \sum_{i=1}^n \mathsf{G}_\ell(s_i)$.

To complete the proof, we proceed by induction on $\mathsf{G}_\ell(a)$. The base case $\mathsf{G}_\ell(a) = 0$ follows trivially. For the induction step, we show that $a \succcurlyeq_\ell b$ implies $\mathsf{G}_\ell(b) < \sum_{i=1}^n \mathsf{G}_\ell(s_i)$. From this, we obtain $\mathsf{G}_\ell([s_1 \ \cdots \ s_n]) \leqslant \sum_{i=1}^n \mathsf{G}_\ell(s_i)$, which together with the above observation yields $\mathsf{G}_\ell([s_1 \ \cdots \ s_n]) = \sum_{i=1}^n \mathsf{G}_\ell(s_i)$. Suppose $a \succcurlyeq_\ell b$. Then this is only possible due to $\succcurlyeq_\ell^{\langle 4 \rangle}$. Hence b is equivalent to $b_1 \frown \cdots \frown b_n$, where $s_i \succcurlyeq_\ell b_i$ for all $i = 1, \dots, n$ and $s_{i_0} \succcurlyeq_\ell b_{i_0}$ for at least one $i_0 \in \{1, \dots, n\}$. In particular, $\mathsf{G}_\ell(b_i) \leqslant \mathsf{G}_\ell(s_i)$ and $\mathsf{G}_\ell(b_{i_0}) < \mathsf{G}_\ell(s_{i_0})$. As we have $\mathsf{G}_\ell(b_i) \leqslant \mathsf{G}_\ell(b) < \mathsf{G}_\ell(a)$ for all $i = 1, \dots, n$, induction hypothesis is applicable to b and all b_i ($i \in \{1, \dots, n\}$). It follows that

$$\mathsf{G}_\ell(b) = \sum_{t \in b} \mathsf{G}_\ell(t) = \sum_{i=1}^n \sum_{t \in b_i} \mathsf{G}_\ell(t) = \sum_{i=1}^n \mathsf{G}_\ell(b_i) < \sum_{i=1}^n \mathsf{G}_\ell(s_i) \,.$$

□

We now approach Lemma 7, where we show that $\mathsf{G}_\ell(f(u_1, \dots, u_k)) \leqslant c \cdot (2 + m)^{\mathsf{rd}(f)}$ for some constant $c \in \mathbb{N}$ and $m = \sum_{i=1}^k \mathsf{dp}(u_i)$. The proof of Lemma 7 is slightly involved, and requires induction on the rank r of f and side induction on m. The constant c is defined in terms of $\mathsf{c}(r, d)$ for natural numbers $r, d \in \mathbb{N}$:

$$\mathsf{c}(r, d) := \begin{cases} 1 & \text{if } r = 1, \text{ and} \\ \mathsf{c}(r-1, d) \cdot \ell^{d+1} + 1 & \text{otherwise} \,. \end{cases}$$

Below the argument r will be instantiated by the rank, and d by the depth of recursion of a function symbol f. The next lemma is a technical lemma to ease the presentation of the proof of Lemma 7. The assumptions correspond exactly to the main induction hypothesis (IH) and side induction hypothesis (SIH) of Lemma 7.

Lemma 6. *Consider $f(u_1,\dots,u_k) \succcurlyeq_\ell g(v_1,\dots,v_l)$ and suppose that*

$$f > g \implies \mathsf{G}_\ell(g(v_1,\dots,v_l)) \leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot \big(2+\sum_{i=1}^{l}\mathsf{dp}(v_i)\big)^{\mathsf{rd}(g)}, \tag{IH}$$

$$f \sim g, \sum_{i=1}^{l}\mathsf{dp}(v_i) < \sum_{i=1}^{k}\mathsf{dp}(u_i) \implies \mathsf{G}_\ell(g(v_1,\dots,v_l)) \leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot \big(2+\sum_{i=1}^{l}\mathsf{dp}(v_i)\big)^{\mathsf{rd}(g)} . \tag{SIH}$$

Then

$$f(u_1,\dots,u_k) \succcurlyeq_\ell^{\langle 1\rangle} g(v_1,\dots,v_l) \implies \mathsf{G}_\ell(g(v_1,\dots,v_l)) \leqslant \mathsf{c}(\mathsf{rk}(f)-1,\mathsf{rd}(f)) \cdot \ell^{\mathsf{rd}(f)} \cdot \big(2+\sum_{i=1}^{k}\mathsf{dp}(u_i)\big)^{\mathsf{rd}(g)}, \tag{†}$$

$$f(u_1,\dots,u_k) \succcurlyeq_\ell^{\langle 2\rangle} g(v_1,\dots,v_l) \implies \mathsf{G}_\ell(g(v_1,\dots,v_l)) \leqslant \mathsf{c}(\mathsf{rk}(f),\mathsf{rd}(f)) \cdot \big(1+\sum_{i=1}^{k}\mathsf{dp}(u_i)\big)^{\mathsf{rd}(f)} . \tag{‡}$$

Proof. First consider the case $f(u_1,\dots,u_k) \succcurlyeq_\ell^{\langle 1\rangle} g(v_1,\dots,v_l)$. Then $f > g$ and so $\mathsf{rk}(f) > \mathsf{rk}(g)$ and $\mathsf{rd}(f) \geqslant \mathsf{rd}(g)$ hold. From the order constraints on arguments we can derive $\sum_{i=1}^{l}\mathsf{dp}(v_i) \leqslant l \cdot \sum_{i=1}^{k}\mathsf{dp}(u_i)$. Observe that the assumption gives also $l \leqslant \ell$. Summing up, simple arithmetical reasoning gives the implication (†) from (IH). Similar, when $f(u_1,\dots,u_k) \succcurlyeq_\ell^{\langle 2\rangle} g(v_1,\dots,v_l)$ holds we have $\mathsf{rk}(f) = \mathsf{rk}(g)$ and $\mathsf{rd}(f) = \mathsf{rd}(g)$. The order constraints on arguments give $\sum_{i=1}^{l}\mathsf{dp}(v_i) < \sum_{i=1}^{k}\mathsf{dp}(u_i)$. From this, the implication (‡) follows directly from (SIH). □

Lemma 7. *For all $f \in \mathcal{D}$, $\mathsf{G}_\ell(f(u_1,\dots,u_k)) \in \mathsf{O}\big((\sum_{i=1}^{k}\mathsf{dp}(u_i))^{\mathsf{rd}(f)}\big)$.*

Proof. Let ℓ be fixed. To show the theorem, we show

$$f(u_1,\dots,u_k) \succcurlyeq_\ell b \implies \mathsf{G}_\ell(b) < \mathsf{c}(\mathsf{rk}(f),\mathsf{rd}(f)) \cdot \big(2+\sum_{i=1}^{k}\mathsf{dp}(u_i)\big)^{\mathsf{rd}(f)} .$$

In proof we employ induction on $\mathsf{rk}(f)$ and side induction on $m := \sum_{i=1}^{k}\mathsf{dp}(u_i)$. Abbreviate $r := \mathsf{rk}(f)$ and $d := \mathsf{rd}(f)$. Assume that $f(u_1,\dots,u_k) \succcurlyeq_\ell b$ holds. We prove $\mathsf{G}_\ell(b) < \mathsf{c}(r,d) \cdot (2+m)^d$, where we show only the more involved inductive case $r > 1$. The base case $r = 1$ follows by similar reasoning. We analyse two cases.

If $b = [t_1 \ \cdots \ t_l]$ is a sequence, then by assumption $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 3\rangle} b$. Thus $l \leqslant \ell$ and $f(v_1,\dots,v_k) \succcurlyeq_\ell t_j$, i.e. either $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 1\rangle} t_j$ or $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 2\rangle} t_j$ holds for all $j = 1,\dots,l$. Due to the second condition imposed on $\succcurlyeq_\ell^{\langle 3\rangle}$, we even have $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 1\rangle} t_j$ for all but one $j \neq j_0 \in \{1,\dots,l\}$. Suppose first that f is recursive. Then $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 1\rangle} t_j$ implies $d > \mathsf{rd}(\mathsf{rt}(t_j)) \geqslant 0$. Using induction and side induction hypothesis to satisfy the assumptions of Lemma 6, we obtain

$$\begin{aligned} \mathsf{G}_\ell(t_j) &\leqslant \mathsf{c}(r-1,d) \cdot \ell^d \cdot (2+m)^{d-1} && \text{(for all } j \neq j_0\text{)} \\ \mathsf{G}_\ell(t_{j_0}) &\leqslant \max\{\mathsf{c}(r-1,d) \cdot \ell^d \cdot (2+m)^{d-1}, \mathsf{c}(r,d) \cdot (1+m)^d\} . \end{aligned}$$

Here the second inequality is obtained by combining the conclusions of the two implications provided by Lemma 6. We conclude the case as follows.

$$\begin{aligned} \mathsf{G}_\ell(b) &= \sum_{j=1}^{l}\mathsf{G}_\ell(t_j) && \text{by Lemma 5,} \\ &\leqslant \mathsf{c}(r,d) \cdot (1+m)^d + l \cdot \big(\mathsf{c}(r-1,d) \cdot \ell^d \cdot (2+m)^{d-1}\big) && \text{above consequences of Lemma 6,} \\ &< \mathsf{c}(r,d) \cdot (1+m)^d + \mathsf{c}(r,d) \cdot (2+m)^{d-1} && \text{using } l \leqslant \ell \text{ and unfolding } \mathsf{c}(r,d), \\ &\leqslant \mathsf{c}(r,d) \cdot (2+m)^d . \end{aligned}$$

Suppose now that f is not recursive. Then also $f(v_1,\dots,v_k) \succcurlyeq_\ell^{\langle 1\rangle} t_{j_0}$. Employing that $f > \mathsf{rt}(t_j)$ implies $d \geqslant \mathsf{rd}(\mathsf{rt}(t_j))$, using Lemma 5 and Lemma 6 we see

$$\mathsf{G}_\ell(b) = \sum_{j=1}^{l} \mathsf{G}_\ell(t_j) \leqslant l \cdot (\mathsf{c}(r-1,d) \cdot \ell^d \cdot (2+m)^d) < \mathsf{c}(r,d) \cdot (2+m)^d \,.$$

This finishes the cases when b is a sequence.

Finally, when $b = g(t_1,\dots,t_l)$ is a term we conclude directly by Lemma 6, using $\mathsf{c}(r-1,d) \cdot \ell^d < \mathsf{c}(r,d)$ similar to above. □

Putting things together, we arrive at the proof of our first theorem.

Proof of Theorem 1. Let $\mathcal{R}$ denote a predicative recursive TRS. We prove the existence of a constant $c \in \mathbb{N}$ such that for all values $\vec{u}, \vec{v}$ the derivation height of $f(\vec{u}\,;\vec{v})$ is bounded by $c \cdot n^{\mathsf{rd}(f)}$, where n is the sum of the depths of normal arguments $\vec{u}$.

Consider a derivation $f(\vec{u}\,;\vec{v}) \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_n$. Let $i \in \{0,\dots,n-1\}$. By Lemma 1 it follows that $t_i \in \mathcal{T}_{\mathsf{b}}^{\to}$, and consequently $\mathsf{S}(t_i) \succcurlyeq_\ell \mathsf{S}(t_{i+1})$ due to Lemma 4. So in particular the length n is bounded by the length of $\succcurlyeq_\ell$ descending sequences starting from $\mathsf{S}(f(\vec{u}\,;\vec{v})) = [\,f(\vec{u})\,]$. By Lemma 5, $\mathsf{G}_\ell([\,f(\vec{u})\,]) = \mathsf{G}_\ell(f(\vec{u}))$. Thus Lemma 7 gives the constant $c \in \mathbb{N}$ as desired. □

4. Parameter Substitution

Bellantoni already observed that his definition of FP is closed under safe recursion on notation with *parameter substitution*. Here a function f is defined from functions g, h_0, h_1 and $\vec{p}$ by

$$\begin{array}{rl} f(\epsilon,\vec{x}\,;\vec{y}) & = g(\vec{x}\,;\vec{y}) \\ f(zi,\vec{x}\,;\vec{y}) & = h_i(z,\vec{x}\,;\vec{y}, f(z,\vec{x}\,;\vec{p}(z,\vec{x}\,;\vec{y}))) \qquad (i=0,1)\,. \end{array} \tag{$\mathbf{SRN_{PS}}$}$$

We introduce the *small polynomial path order with parameter substitution* ($sPOP^*_{PS}$ for short), where clause $>^{\langle 3\rangle}_{\mathsf{spop}*}$ is extended to account for the schema ($\mathbf{SRN_{PS}}$). Theorem 1 remains valid under this extension.

Definition 6. Let s and t be terms such that $s = f(s_1,\dots,s_k\,;s_{k+1},\dots,s_{k+l})$. Then $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$ if one of the following alternatives holds.

1. $s_i \geqslant_{\mathsf{spop}^*_{\mathsf{ps}}} t$ for some argument s_i of s ($i \in \{1,\dots,k+l\}$).
2. $f \in \mathcal{D}$, $t = g(t_1,\dots,t_m\,;t_{m+1},\dots,t_{m+n})$ where $f > g$ and the following holds:
   - $s \rhd\!\!\!\!/_{\sim} t_j$ for all normal arguments $t_1,\dots,t_m$;
   - $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ for all safe arguments $t_{m+1},\dots,t_{m+n}$;
   - t contains at most one (recursive) function symbols g with $f \sim g$.
3. $f \in \mathcal{D}_{\mathsf{rec}}$, $t = g(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$ where $f \sim g$ and the following holds:
   - $\langle s_1,\dots,s_k\rangle >_{\mathsf{spop}^*_{\mathsf{ps}}} \langle t_{\pi(1)},\dots,t_{\pi(k)}\rangle$ for some permutation π on $\{1,\dots,k\}$;
   - $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ for all safe arguments t_j;
   - arguments $t_1,\dots,t_{k+l}$ contain no (recursive) symbols g with $f \sim g$.

Here $s \geqslant_{\mathsf{spop}^*_{\mathsf{ps}}} t$ denotes that either $s \lightsquigarrow t$ or $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$. In the last clause, we use $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ also for the product extension of $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ (modulo permutation).

We adapt the notion of predicative recursive TRS of degree d to $\mathrm{sPOP}^*_{\mathrm{PS}}$ in the obvious way. Note that $>_{\mathsf{spop}*} \subseteq >_{\mathsf{spop}^*_{\mathsf{ps}}}$ does *not* hold in general, due to the third constraint put onto $>^{\langle 3\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. Still, the above order extends the analytic strength of small polynomial path orders.

Lemma 8. *If a TRS $\mathcal{R}$ is compatible with $>_{\mathsf{spop}*}$ then $\mathcal{R}$ is also compatible with $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ using the same precedence and separation of argument positions.*

Proof. Consider the orientation $l >_{\mathsf{spop}*} r$ of a rule $l \to r \in \mathcal{R}$. To prove the lemma, we show that $l >_{\mathsf{spop}^*_{\mathsf{ps}}} r$ holds by replacing every application of $>^{\langle \mathrm{i} \rangle}_{\mathsf{spop}*}$ by $>^{\langle \mathrm{i} \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. We prove this claim by induction on $>_{\mathsf{spop}*}$. We consider the only non-trivial case where $s >^{\langle 3 \rangle}_{\mathsf{spop}*} t$ appears in the proof of $l >_{\mathsf{spop}^*_{\mathsf{ps}}} r$. Compare the case $>^{\langle 3 \rangle}_{\mathsf{spop}*}$ with the new case $>^{\langle 3 \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. Using the induction hypothesis, the order constraints on normal arguments are immediately satisfied. Now fix a safe argument t_j of t. From $s >^{\langle 3 \rangle}_{\mathsf{spop}*} t$ we obtain a safe argument s_i of s with $s_i >_{\mathsf{spop}*} t_j$. Hence $s_i >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ holds by induction hypothesis. Thus $s >^{\langle 1 \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ holds as desired. Observe that the safe arguments s_i of s are proper subterms of the left-hand side l, hence the terms s_i contain no defined symbols. Since $>_{\mathsf{spop}*}$ collapses to the subterm relation on constructor terms, it follows that the safe argument t_j of t are constructor terms too. From this we see that the final constraint of $>^{\langle 3 \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ is satisfied. □

Parameter substitution extends the analytic power of sPOP* significantly. Noteworthy, sPOP$^*_{\mathrm{PS}}$ can deal with *tail-recursive* rewrite systems.

Example 8. The TRS $\mathcal{R}_{\mathsf{rev}}$ consisting of the three rules

$$\mathsf{rev}(xs\,;) \to \mathsf{rev}_{\mathsf{tl}}(xs\,;\mathsf{nil}) \qquad \mathsf{rev}_{\mathsf{tl}}(\mathsf{nil}\,;ys) \to ys \qquad \mathsf{rev}_{\mathsf{tl}}(\mathsf{cons}(\,;x,xs)\,;ys) \to \mathsf{rev}_{\mathsf{tl}}(xs\,;\mathsf{cons}(\,;x,ys))\;,$$

which reverses lists formed from the constructors nil and cons. Define the separation of argument positions as indicated in the rules. The underlying precedence is given as $\mathsf{rev} > \mathsf{rev}_{\mathsf{tl}} > \mathsf{cons}$. Since $\mathsf{rev}_{\mathsf{tl}}$ is the only recursive symbol, the degree of recursion of $\mathcal{R}_{\mathsf{rev}}$ is one.

Notice that orientation of the final rule with the induced sPOP* reduces to the unsatisfiable constraint $ys >_{\mathsf{spop}*} \mathsf{cons}(\,;x,ys)$. In contrast, orientation with the induced POP$^*_{\mathrm{PS}}$ reduces to the constraint $\mathsf{rev}_{\mathsf{tl}}(\mathsf{cons}(\,;x,xs)\,;ys) >_{\mathsf{spop}^*_{\mathsf{ps}}} \mathsf{cons}(\,;x,ys)$, which can be resolved by one application of $>^{\langle 2 \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ followed by three applications of $>^{\langle 1 \rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$.

As a consequence of the next theorem, the runtime of $\mathcal{R}_{\mathsf{rev}}$ is inferred to be linear.

Theorem 2. *Let $\mathcal{R}$ be a predicative recursive TRS of degree d (with respect to Definition 6). Then the derivation height of any basic term $f(\vec{u}\,;\vec{v})$ is bounded by a polynomial of degree $\mathsf{rd}(f)$ in the sum of the depths of normal arguments $\vec{u}$. In particular, the runtime complexity function $\mathrm{rc}_{\mathcal{R}}$ is bounded by a polynomial of degree d.*

Proof. We observe that under the new definition all proofs, in particular the predicative embedding shown in Section 3.3, go through unchanged. □

5. Predicative Recursive Rewrite Systems Compute all Polytime Functions

In this section we show that sPOP* is complete for FP. Indeed, we can even show a stronger result. Let f be a function from $\mathcal{B}_{\mathsf{wsc}}$ that makes only use of d nestings of safe recursion on notation, then there exists a predicative recursive TRS $\mathcal{R}_f$ of degree d that computes the function f.

By definition $\mathcal{B}_{\mathsf{wsc}} \subseteq \mathcal{B}$ for Bellantoni and Cooks predicative recursive characterisation $\mathcal{B}$ of FP given in [1]. Concerning the converse inclusion, the following theorem states that the class $\mathcal{B}_{\mathsf{wsc}}$ is large enough to capture *all* the polytime computable functions.

Theorem 3. *Every polynomial time computable function belongs to $\mathcal{B}_{\mathsf{wsc}}$.*

One can show this fact by following the proof of Theorem 3.7 in [27], where the unary variant of $\mathcal{B}_{\mathsf{wsc}}$ is defined and the inclusion corresponding to Theorem 3 is shown.The completeness of sPOP* for the polytime computable functions is an immediate consequence of Theorem 3 and the following result.

Theorem 4. *For any $\mathcal{B}_{\mathsf{wsc}}$-function f there exists a predicative recursive TRS $\mathcal{R}$ computing f and of degree d, where d equals the maximal number of nested application of* (**SRN**) *in the definition of f.*

Proof. Let f be a function coming from $\mathcal{B}_{\mathsf{wsc}}$. A witnessing TRS $\mathcal{R}$ is obtained via a term rewriting characterisation of the class $\mathcal{B}_{\mathsf{wsc}}$ depicted in Fig. 1 on page 4. The term rewriting characterisation expresses the definition of $\mathcal{B}_{\mathsf{wsc}}$ as an *infinite* TRS $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$. We define a one-to-one correspondence between functions from $\mathcal{B}_{\mathsf{wsc}}$ and the set of function symbols for $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ as follows. Constructor symbols ϵ, s_0 and s_1 are used to denote binary words. The function symbols S_i, P, $\mathsf{I}_j^{k,l}$, C and $\mathsf{O}^{k,l}$ correspond respectively to the initial functions S_i, P, $I_j^{k,l}$, C and $O^{k,l}$ of $\mathcal{B}_{\mathsf{wsc}}$. The symbol $\mathsf{SUB}[\mathsf{h}, i_1, \ldots, i_n, \vec{\mathsf{g}}]$ is used to denote the function obtained by composing functions h and $\vec{g}$ according to the schema of (**WSC**). Finally, the function symbol $\mathsf{SRN}[\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1]$ corresponds to the function defined by safe recursion on notation from g, h_0 and h_1 in accordance to the schema (**SRN**). With this correspondence, $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is obtained by orienting the equations in Fig. 1 from left to right.

By induction according to the definition of f in $\mathcal{B}_{\mathsf{wsc}}$ we show the existence of a TRS $\mathcal{R}_f$ and a precedence $\succcurlyeq_f$ such that:

1. $\mathcal{R}_f$ is a finite restriction of $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$,
2. $\mathcal{R}_f$ contains the rule(s) defining the function symbol f corresponding to f,
3. $\mathcal{R}_f$ is compatible with $>_{\mathsf{spop}*}$ induced by $\succcurlyeq_f$,
4. f is maximal in the precedence $\succcurlyeq_f$ underlying $\mathcal{R}_f$, and
5. the depth of recursion $\mathsf{rd}(\mathsf{f})$ equals the maximal number of nested application of (**SRN**) in the definition of f in $\mathcal{B}_{\mathsf{wsc}}$.

It can be seen from Condition (1), (3) and (5) that the theorem is witnessed by $\mathcal{R}_f$. To exemplify the construction we consider the step case that f is defined from some functions $g, h_0, h_1 \in \mathcal{B}_{\mathsf{wsc}}$ by the schema (**SRN**). By induction hypothesis we can find witnessing TRSs $\mathcal{R}_g, \mathcal{R}_{h_0}, \mathcal{R}_{h_1}$ with underlying precedences $\succcurlyeq_g, \succcurlyeq_{h_0}, \succcurlyeq_{h_1}$ respectively for g, h_0, h_1. Extend the set of function symbols by a new recursive symbol $\mathsf{f} := \mathsf{SRN}[\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1]$. Let $\mathcal{R}_f$ be the TRS consisting of $\mathcal{R}_g$, $\mathcal{R}_{h_0}$, $\mathcal{R}_{h_1}$ and the following three rules:

$$\mathsf{f}(\epsilon, \vec{x}\,; \vec{y}) \to \mathsf{g}(\vec{x}\,; \vec{y}) \qquad\qquad \mathsf{f}(\mathsf{s}_i(; x), \vec{x}\,; \vec{y}) \to \mathsf{h}_i(z, \vec{x}\,; \vec{y}, \mathsf{f}(z, \vec{x}\,; \vec{y})) \quad (i = 0, 1)\ .$$

It is not difficult to see that the precedence $\succcurlyeq_f$ of $\mathcal{R}_f$ extends the precedences $\succcurlyeq_g$, $\succcurlyeq_{h_0}$ and $\succcurlyeq_{h_1}$ by $\mathsf{f} \sim \mathsf{f}$ and $\mathsf{f} > \mathsf{g}'$ for $\mathsf{g}' \in \{\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1\}$.

Let $>_{\mathsf{spop}*}$ be the sPOP* induced by $\succcurlyeq_f$. Then it is easy to check that $\mathcal{R}_f$ enjoys Condition (1) and (2). In order to show Condition (3), it suffices to orient the three new rules by $>_{\mathsf{spop}*}$. For the first rule, $\mathsf{f}(\epsilon, \vec{x}; \vec{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{g}(\vec{x}; \vec{y})$ holds by the definition of $\succcurlyeq_f$. For the remaining two rules we only orient the case $i = 0$. Since f is a recursive symbol and $\mathsf{s}_0(; z) >^{\langle 1 \rangle}_{\mathsf{spop}*} z$ holds, $\mathsf{f}(\mathsf{s}_0(; z), \vec{x}\,; \vec{y}) >^{\langle 3 \rangle}_{\mathsf{spop}*} \mathsf{f}(z, \vec{x}\,; \vec{y})$ holds. This together with the definition of the precedence $\succcurlyeq_f$ allows us to conclude $\mathsf{f}(\mathsf{s}_0(; z), \vec{x}\,; \vec{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{h}_0(z, \vec{x}\,; \vec{y}, \mathsf{f}(z, \vec{x}\,; \vec{y}))$.

Consider Condition (4). For each function $g' \in \{g, h_0, h_1\}$ from $\mathcal{B}_{\mathsf{wsc}}$, the corresponding function symbol g' is maximal in the precedence $\succcurlyeq_{g'}$ by induction hypothesis for g'. Hence by the definition of $\succcurlyeq_f$, f is maximal in $\succcurlyeq_f$.

It remains to show Condition (5). Notice that $\mathsf{rd}(\mathsf{f}) = 1 + \max\{\mathsf{rd}(\mathsf{g}), \mathsf{rd}(\mathsf{h}_0), \mathsf{rd}(\mathsf{h}_1)\}$, since f is a recursive symbol. Without loss of generality let us suppose $\mathsf{rd}(\mathsf{g}) = \max\{\mathsf{rd}(\mathsf{g}), \mathsf{rd}(\mathsf{h}_0), \mathsf{rd}(\mathsf{h}_1)\}$. Then by induction hypothesis for g, $\mathsf{rd}(\mathsf{g})$ equals the maximal number of nested application of (**SRN**) in the definition of g in $\mathcal{B}_{\mathsf{wsc}}$. Hence $\mathsf{rd}(\mathsf{f}) = 1 + \mathsf{rd}(\mathsf{g})$ equals the one in the definition of f in $\mathcal{B}_{\mathsf{wsc}}$. □

We obtain that predicative recursive TRSs give a *sound* and *complete* characterisation of the polytime computable functions.

Theorem 5. *The following classes of functions are equivalent:*

1. *The class of functions computed by predicative recursive TRSs.*
2. *The class of functions computed by predicative recursive TRSs allowing parameter substitution.*
3. *The class* FP *of functions computable in polynomial time.*

Proof. Let PR_1 and PR_2 refer to the classes defined in clauses (1) and (2) respectively. We have

$$\mathsf{PR}_1 \overset{\text{(Def.)}}{\subseteq} \mathsf{PR}_2 \overset{\text{(Thm. 2)}}{\subseteq} \mathsf{FP} \overset{\text{(Thm. 3)}}{\subseteq} \mathcal{B}_{\mathsf{wsc}} \overset{\text{(Thm. 4)}}{\subseteq} \mathsf{PR}_1 \,.$$

For the second inclusion we tacitly employed the adequacy theorem. □

We remark that from our standing restriction on TRSs, orthogonality is essentially used to ensure that semantics of TRSs are well-defined. Orthogonality could be replaced by the less restrictive, although undecidable, notion of confluence.

As a corollary to Theorem 5 we obtain that the class FP, viz $\mathcal{B}_{\mathsf{wsc}}$, is closed under parameter substitution.

Corollary 1. *For any functions $g, h_0, h_1, \vec{p} \in \mathcal{B}_{\mathsf{wsc}}$, there exists a unique polytime computable function f such that $f(\epsilon, \vec{x}\,; \vec{y}) = g(\vec{x}\,; \vec{y})$ and $f(zi, \vec{x}\,; \vec{y}) = h_i(z, \vec{x}\,; \vec{y}, f(z, \vec{x}\,; \vec{p}(z, \vec{x}\,; \vec{y})))$ for each $i = 0, 1$.*

6. Predicative Recursion precisely captures Register Machine Computations

Exploiting the fine-grained control given by the degree of recursion, we now provide an order-theoretic characterisation of $\mathsf{DTIME}(n^d)$ via *predicative tail-recursive* TRSs.

Definition 7. A TRS $\mathcal{R}$ is *tail-recursive* if for every rule $f(\vec{v}) \to r \in \mathcal{R}$, if g with $g \sim f$ occurs in r then it occurs at the root position in r. The TRS $\mathcal{R}$ is *predicative tail-recursive* (of degree d), if it is tail-recursive and predicative recursive (of degree d), with respect to Definition 6.

For instance, the TRS $\mathcal{R}_{\mathsf{rev}}$ from Example 8 is a predicative tail-recursive TRS. The restriction to tail-recursion is unarguably severe. Still, predicative tail-recursive TRSs of degree d can compute polynomials $c \cdot n^d + e$ (in unary notation) for all $c, e \in \mathbb{N}$.

Example 9. Let $p(n) := c \cdot n^d + e$ denote a polynomial with constants $c, d, e \in \mathbb{N}$. The TRS $\mathcal{R}_p$ is given by the following rules.

$$\begin{aligned}
\mathsf{p}_0(x, y\,; z) &\to \mathsf{s}^c(\,; z) && \\
\mathsf{p}_r(0, y\,; z) &\to z && \text{for } r = 1, \dots, d, \\
\mathsf{p}_r(\mathsf{s}(\,; x), y\,; z) &\to \mathsf{p}_r(x, y\,; \mathsf{p}_{r-1}(y, y\,; z))) && \text{for } r = 1, \dots, d, \\
\mathsf{p}(x\,;) &\to \mathsf{p}_d(x, x\,; \mathsf{s}^e(\,; 0)) &&
\end{aligned}$$

This TRS is tail-recursive, moreover it is predicative recursive with recursive symbols $\mathsf{p}_1, \dots, \mathsf{p}_d$ and precedence $\mathsf{p}_d > \cdots \mathsf{p}_1 > \mathsf{p}_0 > \mathsf{s} \sim 0$. In total, $\mathcal{R}_p$ is thus predicative tail-recursive, of degree d.

Let $\ulcorner n \urcorner = \mathsf{s}^n(\,; 0)$ denote the denotation of $n \in \mathbb{N}$ as value with constructors s and 0. One verifies that for $u, v, w \in \mathbb{N}$, $\mathsf{p}_r(\ulcorner u \urcorner, \ulcorner v \urcorner, \ulcorner w \urcorner)$ reduces to the value $\ulcorner c \cdot u \cdot v^{r-1} + w \urcorner$, for $r = 1, \dots, d$. Thus $[\![\mathsf{p}]\!](\ulcorner n \urcorner) = \ulcorner c \cdot n \cdot n^{d-1} + e \urcorner = \ulcorner c \cdot n^d + e \urcorner$ for all $n \in \mathbb{N}$.

6.1. Predicative Tail-Recursive TRSs of Degree d are Complete for $\mathsf{DTIME}(n^d)$

Fix a register machine M that computes a function $f : \mathcal{W}(\mathbb{A})^k \to \mathcal{W}(\mathbb{A})$ in time $\mathsf{O}(n^d)$. We show that this function is computable by a predicative tail-recursive TRS of degree d. Let $\mathcal{C}_{\mathbb{A}}$ denote the set of constructors that contains a symbol ϵ, and for each $a \in \mathbb{A}$ an unary symbol a. Then the word $w = a_1, \dots, a_l \in \mathcal{W}(\mathbb{A})$ can be represented as value $\mathsf{a_1}(\cdots(\mathsf{a_l}(\epsilon))\cdots)$ over $\mathcal{C}_{\mathbb{A}}$. Having this correspondence in mind, we confuse words with such values below. Furthermore, we suppose for each instruction label $j = 1, \dots, l+1$ of M a designated constant j used to denote this label. The following lemma shows that the one-step transition relation of M is expressible by a predicative TRS $\mathcal{R}_0^M$ of degree 0.

Lemma 9. *Let M be a RM with m registers. There exists a predicative tail-recursive TRS $\mathcal{R}_0^M$ of degree 0 defining the symbols $\mathsf{M}_0, \mathsf{M}_1, \dots, \mathsf{M}_m$, such that*

$$\mathsf{M}_0(\,; \mathsf{j}, u_1, \dots, v_m) \to_{\mathcal{R}_0^M} \mathsf{j}' \quad \textit{and} \quad \mathsf{M}_i(\,; \mathsf{j}, u_1, \dots, u_m) \to_{\mathcal{R}_0^M} v_i \quad (i = 1, \dots, m)$$

iff $\langle j, u_1, \dots, u_m \rangle \to_M \langle j', v_1, \dots, v_m \rangle$.

Proof. Suppose $\langle j, u_1, \dots, u_m\rangle \to_M \langle j', v_1, \dots, v_m\rangle$. For the definition of M_0, note that $j' \neq j+1$ only if the j^{th} instruction in the control of M is a jump instruction. In this case, j' can be determined by the left-most character of one of the values u_i ($i \in \{1, \dots, m\}$). And so j' can be computed in one step using pattern-matching on the inputs $\mathsf{j}, u_1, \dots, u_m$ only. Similarly, for the definition of M_i ($i = 1, \dots, m$), the word v_i is either $\mathsf{a}(u_i)$, ϵ, one of $u_1, \dots, u_m$ or the direct subterm of u_i. Again the precise shape can be determined purely by pattern matching on the inputs $\mathsf{j}, u_1, \dots, u_m$. □

Lemma 10. *Let $f \in \mathsf{DTIME}(n^d)$. Then f is computed by a predicative tail-recursive TRS $\mathcal{R}_f$ of degree d.*

Proof. Suppose the function $f\colon \mathcal{W}(\mathbb{A})^k \to \mathcal{W}(\mathbb{A})$ is computed by a RM M in time $p(n) \in \mathsf{O}(n^d)$. Let $c, e \in \mathbb{N}$ denote constants such that $p(n) \leqslant c \cdot |n|^d + e$ for all $n \in \mathbb{N}$. The construction of $\mathcal{R}_f$ is an adaption of the TRS $\mathcal{R}_p$ given in Example 9, using the TRS $\mathcal{R}_0^M$ provided in Lemma 9 to simulate one step of the RM M. Let $m \geqslant k$ be the numbers of registers of M. For function symbols $\vec{\mathsf{M}} := \mathsf{M}_0, \dots, \mathsf{M}_m$ as provided in Lemma 9, let $\vec{\mathsf{M}}^{\langle \ell\rangle}(\,;\vec{t})$ be the *ℓ-fold parallel composition* of $\vec{\mathsf{M}}$ on terms $\vec{t}$, given by $\vec{\mathsf{M}}^{\langle 0\rangle}(\,;\vec{t}) := \vec{t}$ and

$$\vec{\mathsf{M}}^{\langle \ell+1\rangle}(\,;\vec{t}) := \mathsf{M}_0(\,;\vec{\mathsf{M}}^{\langle \ell\rangle}(\,;\vec{t})), \dots, \mathsf{M}_m(\,;\vec{\mathsf{M}}^{\langle \ell\rangle}(\,;\vec{t}))\ .$$

Observe that iterated application of Lemma 9 yields:

$$\vec{\mathsf{M}}_j^{\langle \ell\rangle}(\,;\mathsf{l}, u_1, \dots, u_m) \to^*_{\mathcal{R}_0^M} v_j \iff (l, u_1, \dots, u_m) \to^\ell_M (l', v_1, \dots, v_m) \quad \text{for all } \ell \geqslant 1 \text{ and } j = 1, \dots, m. \tag{1}$$

For each $r = 1, \dots, d$ and $i = 0, \dots, m$, let $\mathsf{f}_{r,i}$ be fresh a function symbol with $2 \cdot k$ normal and m safe argument positions. Let $\vec{x} := x_1, \dots, x_k$, $\vec{y} := y_1, \dots, y_k$ and $\vec{z} := z_0, \dots, z_m$ denote pairwise distinct variables. The TRS $\mathcal{R}_f$ extends $\mathcal{R}_0^M$ by the following rules.

$$\begin{aligned}
\mathsf{f}_{0,i}(\vec{x}, \vec{y}\,;\vec{z}) &\to \vec{\mathsf{M}}_i^{\langle c\rangle}(\,;\vec{z})\\
\mathsf{f}_{r,i}(\vec{\epsilon}, \vec{y}\,;\vec{z}) &\to z_i\\
\mathsf{f}_{r,i}(\vec{\epsilon}, \mathsf{a}(x_j), \dots, x_k, \vec{y}\,;\vec{z}) &\to \mathsf{f}_{r,i}(\vec{\epsilon}, x_j, \dots, x_k, \vec{y}\,;\mathsf{f}_{r-1,0}(\vec{y}, \vec{y}\,;\vec{z}), \dots, \mathsf{f}_{r-1,m}(\vec{y}, \vec{y}\,;\vec{z}))\\
\mathsf{f}(\vec{x}\,;) &\to \mathsf{f}_{d,m}(\vec{x}, \vec{x}\,;\vec{\mathsf{M}}^{\langle e\rangle}(\,;\mathsf{1}, \vec{x}, \vec{\epsilon}))\ .
\end{aligned}$$

Here the index r ranges over $1, \dots, d$, the index i ranges over $0, \dots, k$ and $a \in \mathbb{A}$. Let $\vec{u}$ and $\vec{v}$ be vectors of words of length k. Observe that

$$\mathsf{f}_{r,i}(\vec{u}, \vec{v}\,;w_1, \dots, w_k, \vec{\epsilon}) \to^*_{\mathcal{R}_f} \vec{\mathsf{M}}_i^{\langle c\cdot|\vec{u}|\cdot|\vec{v}|^{r-1}\rangle}(\,;w_1, \dots, w_k, \vec{\epsilon})\!\downarrow\ .$$

This derivation can be shown by induction on r and $|\vec{u}|$, in correspondence to Example 9. For words $\vec{w} = w_1, \dots, w_k$, this thus yields

$$\mathsf{f}(\vec{w}\,;) \to^*_{\mathcal{R}_f} \vec{\mathsf{M}}_m^{\langle c\cdot|\vec{w}|\cdot|\vec{w}|^{d-1}\rangle}(\,;\vec{\mathsf{M}}_m^{\langle e\rangle}(\,;\mathsf{1}, \vec{w}, \vec{\varepsilon}))\!\downarrow = \vec{\mathsf{M}}_m^{\langle c\cdot|\vec{w}|^d+e\rangle}(\,;\mathsf{1}, \vec{w}, \vec{\varepsilon})\!\downarrow\ . \tag{2}$$

Putting the derivations (1) and (2) together, and using that RM M runs in time $p(n) \leqslant c \cdot |\vec{w}|^d + e$ on input $w_1, \dots, w_k$, we conclude that $[\![\mathsf{f}]\!](w_1, \dots, w_k) = f(w_1, \dots, w_k)$ holds.

Observe that the precedence $\geqslant$ of $\mathcal{R}_f$ on defined symbols is given by

$$\mathsf{f} > \mathsf{f}_{d,0}, \dots, \mathsf{f}_{d,m} > \dots > \mathsf{f}_{0,0}, \dots, \mathsf{f}_{0,m} > \mathsf{M}_0, \dots, \mathsf{M}_m\ ,$$

where only the symbols $\mathsf{f}_{r,i}$ for $r > 0$ are recursive in $\mathcal{R}_f$. In particular, the recursion depth of $\mathcal{R}_f$ is thus d. It is also not difficult to see that $\mathcal{R}_f$ is predicative recursive. As $\mathcal{R}_f$ is tail-recursive, the lemma follows. □

6.2. Predicative Tail-Recursive TRSs of Degree d are Sound for $\mathsf{DTIME}(n^d)$

We now show the converse of Lemma 10. Fix a predicative tail-recursive TRS $\mathcal{R}$ of degree d. Call a function symbol *monadic* if its arity is at most one. Suppose all constructors of $\mathcal{R}$ are monadic and consider a defined symbol f in $\mathcal{R}$. We show that the function $[\![f]\!]$ computed by $\mathcal{R}$ can be implemented on a RM M_f, operating in time $\mathsf{O}(n^d)$. The restriction to monadic constructors allows us to identify values of $\mathcal{R}$ with words over the alphabet $\mathbb{A}_{\mathcal{C}}$, which contains for every constructor $\mathsf{a_i} \in \mathcal{C}$ a distinct letter a_i. We use the word $c_1, \dots, c_l$ to denote the value $\mathsf{c_1}(\mathsf{c_2}(\dots \mathsf{c_{l-1}}(\mathsf{c_l}) \dots))$. Having this correspondence in mind, we again confuse words with values.

To ease presentation, we first consider the sub-case where $\mathcal{R}$ is *simple*. Here a rule $f(\vec{u}\,;\vec{v}) \to r$ is called *simple* if r is a constructor term or $r = g(\vec{w}\,; h_1(\vec{u}\,;\vec{v}), \dots, h_k(\vec{u}\,;\vec{v}))$ where g is either a defined or a constructor symbol and $h_1, \dots, h_k \in \mathcal{D}$. Furthermore $\mathcal{R}$ is called *simple* if all its rules are simple.

Lemma 11. *If $\mathcal{R}$ is simple, then $[\![f]\!] \in \mathsf{DTIME}(n^{\mathsf{rd}(f)})$ for every defined symbol f from $\mathcal{R}$.*

Proof. For each defined symbol f in $\mathcal{R}$ with k normal and l safe arguments, we define a corresponding RM M_f with input registers $\vec{x}_f = x^f_1, \dots, x^f_{k+l}$ and output variable z^f. On input $\vec{u} = u_1, \dots, u_k$ and $\vec{v} = v_1, \dots, v_l$ the RMs M_f run in time $\mathsf{O}(|\vec{u}|^{\mathsf{rd}(f)})$. To simplify the presentation, we first suppose that the precedence of $\mathcal{R}$ is strict on defined symbols, i.e. $f \sim g$ for $f, g \in \mathcal{D}$ implies $f = g$. The construction is by induction on the rank p of f, the bound is proven by induction on p and side induction on $|u|$. Suppose the input registers $\vec{x}_f$ hold the values $\vec{u}, \vec{v}$.

First observe that M_f is able to determine in constant time (depending only on $\mathcal{R}$) the (unique) rewrite rule applicable to $f(\vec{u}\,;\vec{v})$. Since there are only a constant number of rules in $\mathcal{R}$, it suffice to realise that the time required for pattern matching depends only on $\mathcal{R}$. To this end, suppose we want to match $f(\vec{u}\,;\vec{v})$ against the left hand-side $f(\vec{l_n}\,;\vec{l_s}) \to r \in \mathcal{R}$. Due to linearity, M_f can match the arguments $\vec{u}, \vec{v}$ against $\vec{l_n}, \vec{l_s}$ individually. For this, the RM M_f just has to copy sequentially each argument $w \in \vec{u}, \vec{v}$ to a temporary register, w_i can then be matched against the corresponding argument $l_i \in \vec{l_n}, \vec{l_s}$ using a constant number of jump and delete instructions.

Once the applicable rewrite rule has been identified, the RM M_f can proceed according to its right-hand side as follows. If $f(\vec{u}\,;\vec{v})$ rewrites in one step to a value, say w, then $w = C[x\sigma]$ for some constructor context C and substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C})$. Then some input register $x_i \in \vec{x}_f$ holds the word $C'[x\sigma]$. Notice that the contexts C and C' depend only on the applied rewrite rule. Hence M_f can provide the result w in register z_f in constant time. Thus suppose $f(\vec{u}\,;\vec{v})$ does not rewrite to a value in one step. Since $\mathcal{R}$ is simple

$$f(\vec{u}\,;\vec{v}) \to_{\mathcal{R}} g(w_1, \dots, w_m\,; h_1(\vec{u}\,;\vec{v}), \dots, h_n(\vec{u}\,;\vec{v})) \text{ where } h_1, \dots, h_n \in \mathcal{D}\,.$$

As $\mathcal{R}$ is predicative recursive, $f > h_j$ holds for all $j = 1, \dots, n$. Furthermore, either $f > g$ or $f = g$ holds by our assumption that $\geqslant$ is strict on defined symbols. In both cases, order constraints on normal arguments give $f(\vec{u}\,;\vec{v}) \rhd^{\mathsf{n}}\!/_{\sim} w_i$ $(i = 1, \dots, m)$, i.e. some input register holds a superterm of w_i. The RM M_f can prepare the arguments w_i in dedicated registers x^g_i for all $i = 1, \dots, m$ in constant time. By induction hypothesis, there exist RMs M_{h_j} $(j = 1, \dots, n)$ that on input registers $\vec{x}_{h_j}$ initialised by $\vec{u}, \vec{v}$, compute the value $[\![h_j]\!](\vec{u}\,;\vec{v})$ in time $\mathsf{O}(|\vec{u}|^{\mathsf{rd}(h_j)})$. The RM M_f can use these machines as sub-procedures (cf. [33]) in order to compute $[\![h_j]\!](\vec{u}\,;\vec{v})$ $(j = 1, \dots, n)$. Overall this requires at most $\mathsf{O}(|\vec{u}|^d)$ steps, where $d := \max\{\mathsf{rd}(h_1), \dots, \mathsf{rd}(h_n)\} \leqslant \mathsf{rd}(f)$ is the maximal recursion depth of the defined symbols h_i. The interesting case is now when $g \in \mathcal{D}$. We analyse the cases $f > g$ and $f = g$ independently.

If $f > g$ then as before we can use a machine M_g given by induction hypothesis that computes $[\![g]\!](\vec{w}\,;[\![h_1]\!](\vec{u}\,;\vec{v}), \dots, [\![h_n]\!](\vec{u}\,;\vec{v})) = [\![f]\!](\vec{u}\,;\vec{v})$ where $\vec{w} := w_1, \dots, w_m$ in time $\mathsf{O}(|\vec{w}|^{\mathsf{rd}(g)})$, from the already initialised input registers $\vec{x}^g$. As a consequence of the order constraints on $\mathcal{R}$ we see $|w_i| \leqslant \max\{|u_j| \mid u_j \in \vec{u}\}$ for all $i = 1, \dots, m$. Thus $|w| \leqslant m \cdot |\vec{u}|$, and hence overall the procedure takes time $\mathsf{O}(|\vec{u}|^d) + \mathsf{O}(|\vec{w}|^{\mathsf{rd}(g)}) \subseteq \mathsf{O}(|\vec{u}|^{\mathsf{rd}(f)})$. For the inclusion we employ $d \leqslant \mathsf{rd}(f)$ and $\mathsf{rd}(g) \leqslant \mathsf{rd}(f)$ as given by the assumptions.

Otherwise $f = g$, hence f is recursive. Recall that $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ collapses to the subterm relation (modulo equivalence) on values. From the order constraint on normal arguments $\langle \vec{u} \rangle >_{\mathsf{spop}^*_{\mathsf{ps}}} \langle \vec{w} \rangle$ it is thus not difficult to derive $|\vec{w}| < |\vec{u}|$. Recall $d = \max\{\mathsf{rd}(h_1), \dots, \mathsf{rd}(h_n)\} < \mathsf{rd}(f)$ since f is recursive. Thus it follows

that $|\vec{u}|^d + |\vec{w}|^{\mathsf{rd}(f)} \leqslant |\vec{u}|^{\mathsf{rd}(f)} + 1$. Using the side induction hypothesis we conclude that M_f operates in time $\mathsf{O}(|\vec{u}|^d) + \mathsf{O}(|\vec{w}|^{\mathsf{rd}(f)}) = \mathsf{O}(|\vec{u}|^{\mathsf{rd}(f)})$ overall. We conclude this final case.

To lift the assumption on the precedence, suppose $\{f_1, \dots, f_\ell\}$ is the set of all function symbols equivalent to $f \in \mathcal{D}$, i.e., $f_1, \dots, f_\ell$ are defined by mutual recursion. Since this class is finite, one can store i (for $i = 1, \dots, \ell$) in a dedicated register of M_f, say r. Although more tedious, it is not difficult to see that the above construction can then be altered, so that M_f computes $f_{\langle r \rangle}(\vec{u}\,;\vec{v})$ on input $\vec{u}, \vec{v}$. □

We now remove the restriction that $\mathcal{R}$ is simple. For that we define the relation $\rightsquigarrow$ on TRSs as follows. Let $h_1, \dots, h_k$ be fresh symbols not appearing in $\mathcal{R}$. Then

$$\mathcal{R} \uplus \{f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;t_1,\dots,t_k)\} \rightsquigarrow \mathcal{R} \cup \{f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;h_1(\vec{u}_f\,;\vec{v}_f),\dots,h_k(\vec{u}_f\,;\vec{v}_f))\} \\ \cup \{h_i(\vec{u}_f\,;\vec{v}_f) \to t_i \mid i = 1,\dots,k\}\,,$$

provided the *transformed rule* $f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;t_1,\dots,t_k)$ is not already simple. The relation $\rightsquigarrow$ enjoys following properties.

Lemma 12.

1. *The relation $\rightsquigarrow$ is well-founded.*
2. *If $\mathcal{R} \rightsquigarrow \mathcal{S}$ then $\to_{\mathcal{R}} \subseteq \to_{\mathcal{S}}^{+}$.*
3. *Let $\mathcal{R}$ be a predicative tail-recursive TRS of degree d that uses only monadic constructors. If $\mathcal{R} \rightsquigarrow \mathcal{S}$ then $\mathcal{S}$ enjoys the same properties.*

Proof. Let $\|\mathcal{R}\| := \sum_{r\in\mathbb{R}}|r|$, where $\mathbb{R} = \{r \mid l \to r \in \mathcal{R} \text{ is not a simple rule}\}$. Then an infinite chain $\mathcal{R}_1 \rightsquigarrow \mathcal{R}_2 \rightsquigarrow \cdots$ translates into an infinite descend $\|\mathcal{R}_1\| > \|\mathcal{R}_2\| > \cdots$. Hence property 1 follows.

Suppose now $\mathcal{R} \rightsquigarrow \mathcal{S}$. For property 2, consider a rewrite step $C[f(\vec{u}_f\sigma\,;\vec{v}_f\sigma)] \to_{\mathcal{R}} C[g(\vec{u}_g\sigma\,;t_1\sigma,\dots,t_k\sigma)]$ using a transformed rule $f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;t_1,\dots,t_k) \in \mathcal{R}$. Then

$$C[f(\vec{u}_f\sigma\,;\vec{v}_f\sigma)] \to_{\mathcal{S}} C[g(\vec{u}_g\sigma\,;h_1(\vec{u}_f\sigma\,;\vec{v}_f\sigma),\dots,h_k(\vec{u}_f\sigma\,;\vec{v}_f\sigma))] \to_{\mathcal{S}}^{k} C[g(\vec{u}_g\sigma\,;t_1\sigma,\dots,t_k\sigma)]\,,$$

simulates the considered step. So clearly $\to_{\mathcal{R}} \subseteq \to_{\mathcal{S}}^{+}$ follows.

Finally consider property 3, and consider a TRS $\mathcal{S}$ such that $\mathcal{R} \rightsquigarrow \mathcal{S}$. Let $f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;t_1,\dots,t_k)$, denote the rule which is replaced by

$$f(\vec{u}_f\,;\vec{v}_f) \to g(\vec{u}_g\,;h_1(\vec{u}_f\,;\vec{v}_f),\dots,h_k(\vec{u}_f\,;\vec{v}_f)) \quad \text{and} \quad h_i(\vec{u}_f\,;\vec{v}_f) \to t_i \ (i = 1,\dots,k).$$

Let $\geqslant$ denote the precedence underlying $\mathcal{R}$, and $\sqsupseteq$ the precedence underlying the simplified TRS $\mathcal{S}$. Notice that $\sqsupseteq$ is an extension of $\geqslant$, which collapses to $\geqslant$ on the signature $\mathcal{F}$ of $\mathcal{R}$. As the freshly introduced symbols h_i are not recursive, the recursion depth of every symbol $h \in \mathcal{F}$ is preserved by the transformation.

It is obvious that when $\mathcal{R}$ is tail-recursive, so is $\mathcal{S}$. It thus remains to verify that $\mathcal{S}$ is oriented by the order $\sqsupseteq_{\mathsf{spop}^*_{\mathsf{ps}}}$. Since $\geqslant \subseteq \sqsupseteq$, and as a consequence $>_{\mathsf{spop}^*_{\mathsf{ps}}} \subseteq \sqsupseteq_{\mathsf{spop}^*_{\mathsf{ps}}}$, it suffices to show that the orientation $f(\vec{u}_f\,;\vec{v}_f) >_{\mathsf{spop}^*_{\mathsf{ps}}} g(\vec{u}_g\,;t_1,\dots,t_k)$ of the replaced rule implies

$$f(\vec{u}_f\,;\vec{v}_f) \sqsupseteq_{\mathsf{spop}^*_{\mathsf{ps}}} g(\vec{u}_g\,;h_1(\vec{u}_f\,;\vec{v}_f),\dots,h_k(\vec{u}_f\,;\vec{v}_f)) \tag{3}$$

$$h_i(\vec{u}_f\,;\vec{v}_f) \sqsupseteq_{\mathsf{spop}^*_{\mathsf{ps}}} t_i \qquad (i = 1,\dots,k)\,. \tag{4}$$

We perform case analysis on the assumption.

Suppose first $f(\vec{u}_f\,;\vec{v}_f) >^{\langle 1\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} g(\vec{u}_g\,;t_1,\dots,t_k)$ holds. Note that by the shape of left-hand sides in $\mathcal{R}$ and definition of the precedence, g is a constructor in the considered case. In particular g admits only safe argument positions. Thus $f(\vec{u}_f\,;\vec{v}_f) \sqsupseteq^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} g(\vec{u}_g\,;h_1(\vec{u}_f\,;\vec{v}_f),\dots,h_k(\vec{u}_f\,;\vec{v}_f))$ holds using $f(\vec{u}_f\,;\vec{v}_f) \sqsupseteq^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} h_i(\vec{u}_f\,;\vec{v}_f)$ $(i = 1,\dots,k)$. This concludes (3). The assumption give also $f(\vec{u}_f\,;\vec{v}_f) \rhd\!\!\!/_{\sim} t_i$ for all $i = 1,\dots,k$, thus $h_i(\vec{u}_f\,;\vec{v}_f) \sqsupseteq^{\langle 1\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} t_i$ holds and we conclude (4).

Finally suppose that $f(\vec{u}_f\,;\vec{v}_f) >_{\mathsf{spop}^*_{\mathsf{ps}}} g(\vec{u}_g\,;t_1,\dots,t_k)$ follows by $>^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ either or $>^{\langle 3\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. Using the order constraint $f(\vec{u}_f\,;\vec{v}_f) >_{\mathsf{spop}^*_{\mathsf{ps}}} h_i(\vec{u}_f\,;\vec{v}_f)$ for all $i = 1,\dots,k$, we see that (3) follows by $\sqsupseteq^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ or $\sqsupseteq^{\langle 3\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$

respectively. For equation (4), observe that since $\mathcal{R}$ is tail-recursive, the assumption gives $f(\vec{u}_f\,;\vec{v}_f) >_{\mathsf{spop}^*_{\mathsf{ps}}} t_i$ $(i = 1,\dots,k)$ using only applications of $>^{\langle 1\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ and $>^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. Repeating these proofs, but employing $h_i \sqsupset g$ instead of $f > g$ yields a proof of (4). □

Lemma 13. *Let $\mathcal{R}$ be a predicative tail-recursive TRS of degree d, and suppose all constructors are monadic. Then $[\![f]\!] \in \mathsf{DTIME}(n^{\mathsf{rd}(f)})$ for every defined symbol f from $\mathcal{R}$.*

Proof. Let $\mathcal{S}$ be a $\rightsquigarrow$-normal form of our analysed TRS $\mathcal{R}$. Then $\mathcal{S}$ is simple as otherwise $\rightsquigarrow$-reducible. Using the assumptions on $\mathcal{R}$, Lemma 12 yields that $\mathcal{S}$ satisfies the preconditions of Lemma 11. Moreover, it shows that $\mathcal{S}$ computes all functions computed by $\mathcal{R}$. We conclude by Lemma 11. □

6.3. Predicative Tail-Recursive TRSs of Degree d Characterise $\mathsf{DTIME}(n^d)$

By Lemma 10 and Lemma 13, we obtain following correspondence.

Theorem 6. *For each $d \in \mathbb{N}$, the following classes of functions are equivalent:*

1. *The class of functions computed by predicative tail-recursive TRSs of degree d, using only monadic constructors.*
2. *The class $\mathsf{DTIME}(n^d)$ of functions computed by register machines operating in time $\mathsf{O}(n^d)$.*

This theorem is closely connected to the recursion-theoretic characterisation of the polytime computable functions provided by Leivant [12], and the one of Marion [13]. Leivant uses *ramified recurrence schemes* to impose data tiering on functions defined by recursive means. Restricted to word algebras and two tiers, a function f in Leivant's class belongs to $\mathsf{DTIME}(n^d)$, where d corresponds to the number of nested recursive definitions in f. Vice verse, any function in $\mathsf{DTIME}(n^d)$ is expressible in Leivant's class using two tiers, and maximal d nested recursive definitions. Hence there is a precise correspondence between the functions f defined in Leivant's class based on d nested recursive definitions, and the functions definable by predicative recursive TRS of degree d. Syntactically, the restriction to two tiers in Leivant's class results in a composition scheme conceptually similar to weak safe composition. Substitution is only allowed on arguments not used for recursion.

Our result as well as Leivant's characterisation, relies on recursion schemes that go beyond primitive recursion with data tiering. Leivant allows recursive definitions by *simultaneous recursion*. We note that in our context, simultaneous recursion cannot be permitted. In general, such an extension would invalidate Theorem 1 and Theorem 2 respectively. Instead, we resort to parameter substitution, which is essential for our completeness result. Still simultaneous recursion can be reduced to primitive recursion, preserving the data tiering principle underlying [12]. However, this program transformation relies on a form of tupling, and does not preserve the number of nestings of recursive definitions. In our context, parameter substitution can be eliminated in recursive definitions in a similar spirit, at the expense of the depth of recursion.

Restoring to *strict ramified recurrence schemes*, Marion [13] provides a fine-grained characterisation of $\mathsf{DTIME}(n^d)$ in the spirit of Leivant's characterisation and our result above. The underlying strict ramification principle requires that each recursive definition increases the tier of an input. As a consequence, the exponent d is reflected in the maximal level of an input tier. Crucial here again is the restriction to a composition scheme akin to our weak form of composition.

In Marion's characterisation, functions can return multiple values. As a consequence, the simulation of register machine computations requires neither simultaneous recursion nor similar concepts. It is not difficult to show that a modification of our computational model, which accounts for multi-valued functions, allows the completeness result given in Lemma 10 even if we disallow parameter substitution. We feel however that such a modification, tailored specifically to register machines, introduces a rather ad-hoc flavor to our formulation of computation by TRSs.

7. Examples and Experimental Evaluation

We briefly contrast the orders sPOP^* and $\mathrm{sPOP}^*_{\mathrm{PS}}$ to its predecessor POP^* [9], Marion's LMPO [6], as well as interpretation methods found in state-of-the-art complexity provers. Furthermore, we present experimental results.

Lightweight Multiset Path Order and Polynomial Path Orders. The order sPOP* forms a restriction of POP* and LMPO, whereas the latter two orders are incomparable in general. In contrast to the family of polynomial path orders, LMPO allows multiple recursive calls in right-hand sides. As clarified in the next example, extending our methods would invalidate the corresponding main results (Theorem 1 and Theorem 2 respectively).

Example 10. The TRS $\mathcal{R}_{\mathsf{bin}}$ is given by the following rules:

$$\mathsf{bin}(x,\mathsf{0}) \to \mathsf{s}(\mathsf{0}) \qquad \mathsf{bin}(\mathsf{0},\mathsf{s}(y)) \to \mathsf{0} \qquad \mathsf{bin}(\mathsf{s}(x),\mathsf{s}(y)) \to +(\mathsf{bin}(x,\mathsf{s}(y)),\mathsf{bin}(\mathsf{s}(x),y)) \;.$$

For a precedence $\geqslant$ that fulfils $\mathsf{bin} > \mathsf{s}$ and $\mathsf{bin} > +$ we obtain that $\mathcal{R}_{\mathsf{bin}}$ is compatible with LMPO. This TRS can however neither be handled by sPOP* nor sPOP$^*_{\mathrm{PS}}$. It is straightforward to verify that the family of terms $\mathsf{bin}(\mathsf{s}^n(\mathsf{0}),\mathsf{s}^m(\mathsf{0}))$ admits derivations whose length grows exponentially in n. Still the underlying function can be proven polynomial, essentially relying on memoisation techniques [6].

On the other hand, POP* integrates a multiset status. In contrast, both LMPO and sPOP* are restricted to product status.

Example 11. Consider the following one-ruled TRS $\mathcal{R}_{\mathsf{levy}}$ originally stemming from Jean-Jaques Lévy:[4]

$$\mathsf{f}(\mathsf{g}(\,;x),y\,;y) \to \mathsf{g}(\,;\mathsf{f}(x,x\,;y)) \;.$$

Polynomial runtime complexity of this system can be shown by POP*. The system is neither compatible with an instance of LMPO nor sPOP*, because the product of arguments to f cannot be ordered.

However, the system becomes orientable with an instance of sPOP$^*_{\mathrm{PS}}$, if we make also the second argument of f safe. Observe that $\mathsf{f}(\mathsf{g}(\,;x)\,;y,y) >^{\langle 3\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} \mathsf{f}(x\,;x,y)$ holds, using $\langle \mathsf{g}(\,;x)\rangle >_{\mathsf{spop}^*_{\mathsf{ps}}} \langle x\rangle$, $\mathsf{f}(\mathsf{g}(\,;x)\,;y,y) >^{\langle 1\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} x$ as well as $\mathsf{f}(\mathsf{g}(\,;x)\,;y,y) >^{\langle 1\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}} y$. From this, one application of $>^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$ orients the rule. Since f is the only recursive symbol, Theorem 2 shows that the runtime complexity of $\mathcal{R}_{\mathsf{levy}}$ is at most linear.

Even though sPOP* forms a restriction of POP* and LMPO, its extension by parameter substitution is incomparable to LMPO and POP*. Consider the following TRS.

Example 12. The TRS $\mathcal{R}_{\mathsf{add}}$ consists of the following rules.[5]

$$+(\mathsf{0}\,;y) \to y \qquad +(\mathsf{s}(\,;x)\,;y) \to \mathsf{s}(\,;+(x\,;y)) \qquad +(\mathsf{s}(\,;x)\,;y) \to +(x\,;\mathsf{s}(\,;y))$$

Due to the last rule, the TRS $\mathcal{R}_{\mathsf{add}}$ is neither compatible with sPOP*, POP* nor LMPO. The system is however compatible with the instance $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ of sPOP$^*_{\mathrm{PS}}$ as induced by the precedence underlying $\mathcal{R}_{\mathsf{add}}$ and separation of argument positions indicated in the rules. The degree of recursion of $\mathcal{R}_{\mathsf{add}}$ is one. The runtime complexity of $\mathcal{R}_{\mathsf{add}}$ is inferred to be linear by Theorem 2.

We remark that POP* can be extended by parameter substitution [9]. Unless $\mathsf{PSPACE} = \mathsf{P}$, this extension does however not carry over to LMPO, without sacrificing polytime computability. For instance, the natural extension of LMPO by parameter substitution can handle Example 36 from [11]. This example encodes the PSPACE complete problem of quantifier elimination on quantified Boolean formulas.

Polynomial and Matrix Interpretations. Small polynomial path orders are in general incomparable to interpretation methods, notably *matrix* [36] and *polynomial interpretations* [37]. These are the most frequently used base techniques in complexity tools nowadays. A *polynomial interpretation* is an $\mathcal{F}$-algebra [28] $\mathcal{A}$ with carrier $\mathbb{N}$, where interpretations $f_{\mathcal{A}} : \mathbb{N}^k \to \mathbb{N}$ (for every k-ary function symbol f) are monotone polynomials. A TRS $\mathcal{R}$ is *compatible* with a polynomial interpretation, aka *polynomially terminating*, if for every rule $l \to r \in \mathcal{R}$, under any assignment the left-hand side l is interpreted in $\mathcal{A}$ larger than the interpretation of the

[4]This is example 2.59 in Steinbach and Kühler's collection of TRSs [35].
[5]This is example 2.09 in Steinbach and Kühler's collection of TRSs [35].

right-hand side r. We say that a polynomial interpretation $\mathcal{A}$ *induces* polynomial runtime complexity if the interpretation of every basic term is bounded by a polynomial in the size of s. For such an interpretation $\mathcal{A}$ compatible with $\mathcal{R}$, the runtime complexity of $\mathcal{R}$ is bounded by a polynomial. *Additive polynomial interpretations* [10], where all *constructors* c are interpreted by *additive polynomials* $c_{\mathcal{A}}(x_1, \dots, x_k) = \delta + \sum_{i=1}^{k} x_i$ ($\delta \in \mathbb{N}$) induce polynomial runtime complexity.

Not every polynomially terminating TRS is predicative recursive, even if only additive interpretations are employed. Vice verse, not every predicative recursive TRS is polynomially terminating so that the underlying interpretation induces polynomial runtime complexity. This is clarified in the next two examples.

Example 13. The one-ruled TRS $\{\mathsf{f}(\mathsf{c}(x)) \to \mathsf{f}(\mathsf{d}(x))\}$ is polynomially terminating, using interpretations $\mathsf{c}_{\mathcal{A}}(x) = x + 1$ and $\mathsf{f}_{\mathcal{A}}(x) = \mathsf{d}_{\mathcal{A}}(x) = x$, but it is not compatible with any of the above mentioned restrictions of recursive path orders.

Example 14. Consider the predicative tail-recursive TRS $\mathcal{R}_{\mathsf{btree}}$, which consists of the following two rewrite rules:

$$\mathsf{f}(0\,;y) \to y \qquad \mathsf{f}(\mathsf{s}(\,;x)\,;y) \to \mathsf{f}(x\,;\mathsf{c}(\,;y,y))\;.$$

Suppose this rewrite system is compatible with a polynomial interpretation $\mathcal{A}$. Consider the reduction of a basic term $s_n := \mathsf{f}(\mathsf{s}^n(\,;0)\,;\mathsf{s}(\,;0))$ for $n \in \mathbb{N}$. This yields a binary tree v_n of height n, with leafs $\mathsf{s}(\,;0)$. Observe that by monotonicity, $\mathsf{c}_{\mathcal{A}}(x, y) \geqslant x + y$ holds. Note that orientation requires that $\mathsf{s}_{\mathcal{A}}(x) > x$. As a consequence, the interpretation of terms v_n grows exponentially in n. As by compatibility the interpretations of terms necessarily decrease during reduction, it follows that $\mathcal{A}$ does not induce polynomial runtime complexity.

A *matrix interpretation* $\mathcal{A}$ is similar to a polynomial interpretation, but the underlying carrier of the $\mathcal{F}$-algebra is $\mathbb{N}^d$ ($d \geqslant 1$), and interpretation functions are of shape $f_{\mathcal{A}}(\vec{x}_1, \dots, \vec{x}_k) = F_1 \cdot \vec{x}_1 + \cdots + F_k \cdot \vec{x}_k + f$. Here F_i ($i = 1, \dots, k$) denote matrices of size $d \times d$, and f is vector over $\mathbb{N}$. The notions of compatibility and induced polynomial complexity carry over naturally from polynomial interpretations. As for (additive) polynomial interpretations it can be shown that matrix interpretations are incompatible to small path orders. This is clarified in the next example.

Example 15 (Continued from Examples 2 and 13). Reconsider the predicative recursive TRS $\mathcal{R}_{\mathsf{arith}}$ from Example 1. This system cannot be shown compatible with matrix interpretations. Intuitively this holds due to the linear form of matrix interpretations. The interpretation of a basic term $\times(x, y\,;)$ has to be a non-linear expression in both x and y.

Vice verse, the (additive) polynomial interpretation given in Example 13 turns naturally into a matrix interpretation compatible with the one-ruled TRS depicted in Example 13. On the other hand, this TRS is not predicative recursive.

Our final example shows that even in cases where semantic methods apply, order-based techniques might deduce a tighter bound.

Example 16 (Continued from Example 11). While the TRS $\mathcal{R}_{\mathsf{levy}}$ given in Example 11 can be handled with semantic methods, the polynomial interpretations can only verify a quadratic upper bound. To the contrary, $\mathrm{sPOP}^*_{\mathrm{PS}}$ can verify the (non-optimal) linear bound.

Experimental Assessment. The small polynomial path order sPOP^* gives rise to a new, fully automatic, syntactic method for polynomial runtime complexity analysis. We have implemented this technique in our complexity tool TCT [2]. In particular the complexity proofs above have been obtained automatically with TCT.

In order to further test the viability of small polynomial path orders, we performed experiments on the relative power of sPOP^* (respectively $\mathrm{sPOP}^*_{\mathrm{PS}}$) with respect to LMPO [6], POP^* [8] and interpretations [10,

		LMPO	POP*	sPOP*	sPOP$^*_{\mathsf{PS}}$	SEM	SEM+sPOP$^*_{\mathsf{PS}}$
TC	$\mathsf{O}(1)$	—	—	9/0.13	9/0.13	—	3/0.12
	$\mathsf{O}(n)$	—	—	23/0.16	37/0.21	83/0.73	89/0.70
	$\mathsf{O}(n^2)$	—	—	6/0.22	7/0.23	20/2.17	17/1.84
	$\mathsf{O}(n^3)$	—	—	1/0.58	1/0.62	—	1/6.66
	$\bigcup_{k\in\mathbb{N}} \mathsf{O}(n^k)$	—	43/0.12	—	—	—	—
	Compatible	54/0.14	43/0.12	39/0.17	54/0.21	103/1.01	110/0.91
	Incompatible	543/0.25	554/0.25	558/0.24	543/0.25	25/4.48	25/4.54
	Timeout	—	—	—	—	469/10.0	462/10.0
TCO	$\mathsf{O}(1)$	—	—	5/0.12	5/0.12	—	3/0.12
	$\mathsf{O}(n)$	—	—	14/0.15	19/0.18	44/0.84	45/0.78
	$\mathsf{O}(n^2)$	—	—	4/0.20	4/0.21	13/2.04	11/1.94
	$\bigcup_{k\in\mathbb{N}} \mathsf{O}(n^k)$	—	24/0.11	—	—	—	—
	Compatible	29/0.13	24/0.11	23/0.15	54/0.17	57/1.11	59/0.96
	Incompatible	261/0.13	266/0.17	267/0.17	702/0.17	8/4.36	8/4.29
	Timeout	—	—	—	—	225/10.0	223/10.0

Table 1: Number of oriented problems and average execution times (secs.) on data-sets **TC** and **TCO**.

36] suited to polynomial complexity analysis. Experiments were conducted with TCT version 2.0,[6] on a machine with 8 Dual-Core Opteron™ 885 processors (2.6GHz). We abort TCT if a complexity certificate could not be found within 10 seconds. We selected two data-sets: data-set **TC** constitutes of 597 terminating constructor TRSs and data-set **TCO**, containing 290 examples, resulting from restricting test-suite **TC** to orthogonal systems.[7]

Table 1 summarises the results obtained on data-sets **TC** and **TCO**.[8] On the larger benchmark **TC**, the total of 39 examples drawn in column sPOP* are necessarily a subset of the 54 examples compatible with LMPO, and also the 43 examples compatible with POP*. Note that LMPO induces only exponential bounded runtime complexity. On three examples, including the TRS $\mathcal{R}_{\mathsf{bin}}$ depicted in Example 10, this bound is indeed tight. Whereas POP* can only give an unspecified polynomial bound, sPOP* assesses the complexity of compatible systems between constant and cubic. Thus sPOP* brings about a significant increase in precision, accompanied with only minor decrease in power. This assessment remains true, if we consider the smaller benchmark set **TCO**. Parameter substitution increases the analytic power of POP* on test-suite **TC** from 39 to 54 examples. From the 15 new examples, 13 examples are neither compatible with LMPO nor POP*.

The last two columns in Table 1 indicate the strength of semantic techniques and their combination with sPOP* (column SEM+sPOP$^*_{\mathsf{PS}}$). In column SEM we employed matrix interpretations [36] (dimension 1 to 3) as well as additive polynomial interpretations [10] (degree 2 and 3). Here we make use of the modular combination technique proposed by Zankl and Korp [38] to combine the interpretation techniques. Coefficients, respectively entries in coefficients, range up to 7. To ensure that matrix interpretations induce polynomial runtime complexity, we resort to the non-trivial criteria found in [16]. Column SEM+sPOP$^*_{\mathsf{PS}}$ corresponds to column SEM, where sPOP$^*_{\mathrm{PS}}$ is additionally integrated.

[6] Available from `http://cl-informatik.uibk.ac.at/software/tct/`.

[7] The test-suites are taken from the Termination Problem Database (TPDB), version 8.0; `http://termcomp.uibk.ac.at`.

[8] Full experimental evidence is provided under `http://cl-informatik.uibk.ac.at/software/tct/experiments/spopstar-ICC`.

It is immediate that syntactic techniques alone cannot compete with the expressive power of interpretations. If we consider the total number of compatible systems only, semantic techniques are roughly twice as powerful as the strongest syntactic technique ($\text{sPOP}^*_{\text{PS}}$). Still, the syntactic techniques proposed in this work provide a fruitful addition to the interpretation method. Contrasting columns SEM and $\mathsf{SEM{+}sPOP^*_{PS}}$, not only the total number of certified systems, but also the precision of the obtained certificates, is increased by the addition of $\text{sPOP}^*_{\text{PS}}$. Note also the slight decrease in execution time.

8. Conclusion

We propose a new order, the small polynomial path order sPOP^*, together with its extension $\text{sPOP}^*_{\text{PS}}$ to *parameter substitution*. Based on sPOP^*, we delineate a class of rewrite systems, dubbed systems of predicative recursion of degree d, such that for rewrite systems in this class we obtain that the runtime complexity lies in $O(n^d)$. Exploiting the control given by the degree of recursion, we establish a novel characterisation of the functions computable in time $\mathsf{O}(n^d)$, on *register machines* via the small polynomial path order $\text{sPOP}^*_{\text{PS}}$.

Thus small polynomial path orders induce a new order-theoretic characterisation of the class of polytime computable functions. This order-theoretic characterisation enables a fine-grained control of the complexity of functions in relation to the number of nested applications of recursion. One the other hand, small polynomial path orders provide a novel *syntactic*, and very fast, criteria to automatically establish polynomial runtime complexity of a given TRS. The latter criteria extends the state of the art in runtime complexity analysis as it can be more precise or more efficient than previously known techniques.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments that greatly helped in improving the presentation.

References

[1] S. Bellantoni, S. Cook, A new Recursion-Theoretic Characterization of the Polytime Functions, Computational Complexity 2 (2) (1992) 97–110.

[2] M. Avanzini, G. Moser, Tyrolean Complexity Tool: Features and Usage, in: Proc. of 24^{th} RTA, Vol. 21 of Leibniz International Proceedings in Informatics, 2013, pp. 71–80.

[3] P. Baillot, J.-Y. Marion, S. R. D. Rocca, Guest Editorial: Special Issue on Implicit Computational Complexity, ACM Transactions on Computational Logic 10 (4).

[4] T. Arai, N. Eguchi, A New Function Algebra of EXPTIME Functions by Safe Nested Recursion, ACM Transactions on Computational Logic 10 (4) (2009) 24:1–24:19.

[5] M. Avanzini, N. Eguchi, G. Moser, A Path Order for Rewrite Systems that Compute Exponential Time Functions, in: Proc. of 22^{nd} RTA, Vol. 10 of Leibniz International Proceedings in Informatics, 2011, pp. 123–138.

[6] J.-Y. Marion, Analysing the Implicit Complexity of Programs, Information and Computation 183 (2003) 2–18.

[7] T. Arai, G. Moser, Proofs of Termination of Rewrite Systems for Polytime Functions, in: Proc. of 15^{th} FSTTCS, Vol. 3821 of Lecture Notes in Computer Science, 2005, pp. 529–540.

[8] M. Avanzini, G. Moser, Complexity Analysis by Rewriting, in: Proc. of 9^{th} FLOPS, Vol. 4989 of Lecture Notes in Computer Science, 2008, pp. 130–146.

[9] M. Avanzini, G. Moser, Polynomial Path Orders, Logical Methods in Computer Science 9 (4).

[10] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, Algorithms with Polynomial Interpretation Termination Proof, Journal of Functional Programming 11 (1) (2001) 33–53.

[11] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, Quasi-interpretations: A Way to Control Resources, TCS 412 (25) (2011) 2776–2796.

[12] D. Leivant, Ramified Recurrence and Computational Complexity I: Word Recurrence and Poly-time, in: Feasible Mathematics II, Progress in Computer Science and Applied Logic, Vol. 13, Birkhäuser Boston, 1995, pp. 320–343.

[13] J.-Y. Marion, On Tiered Small Jump Operators, Logical Methods in Computer Science 5 (1) (2009) 1–19.

[14] G. Moser, A. Schnabl, Proving Quadratic Derivational Complexities Using Context Dependent Interpretations, in: Proc. 19^{th} RTA, Vol. 5117 of Lecture Notes in Computer Science, 2008, pp. 276–290.

[15] N. Hirokawa, G. Moser, Automated Complexity Analysis Based on the Dependency Pair Method, CoRR abs/1102.3129, submitted.

[16] A. Middeldorp, G. Moser, F. Neurauter, J. Waldmann, H. Zankl, Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems, in: Proc. of 4^{th} CAI, Vol. 6472 of Lecture Notes in Computer Science, 2011, pp. 1–20.
[17] M. Avanzini, G. Moser, A Combination Framework for Complexity, in: Proc. of 24^{th} RTA, Vol. 21 of Leibniz International Proceedings in Informatics, 2013, pp. 55–70.
[18] G. Moser, Proof Theory at Work: Complexity Analysis of Term Rewrite Systems, CoRR abs/0907.5527, Habilitation Thesis.
[19] J. Hoffmann, K. Aehlig, M. Hofmann, Multivariate Amortized Resource Analysis, Transactions on Programming Languages and Systems 34 (3) (2012) 14.
[20] E. Albert, P. Arenas, S. Genaim, M. Gómez-Zamalloa, G. Puebla, D. Ramírez, G. Román, D. Zanardini, Termination and Cost Analysis with COSTA and its User Interfaces, Electronic Notes in Theoretical Computer Science 258 (1) (2009) 109–121.
[21] C. Alias, A. Darte, P. Feautrier, L. Gonnord, Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs, in: Proc. 17^{th} SAS, Vol. 6337 of Lecture Notes in Computer Science, 2010, pp. 117–133.
[22] S. Gulwani, K. Mehra, T. Chilimbi, SPEED: Precise and Efficient Static Estimation of Program Computational Complexity, in: Proc. of 36th POPL, ACM, 2009, pp. 127–139.
[23] F. Zuleger, S. Gulwani, M. Sinn, H. Veith, Bound Analysis of Imperative Programs with the Size-Change Abstraction, in: Proc. of 18^{th} SAS, Vol. 6887 of Lecture Notes in Computer Science, 2011, pp. 280–297.
[24] H. Simmons, The Realm of Primitive Recursion, Archive for Mathematical Logic 27 (1988) 177–188.
[25] D. Leivant, A Foundational Delineation of Computational Feasiblity, in: Proc. of 6th LICS, IEEE Computer Society, 1991, pp. 2–11.
[26] D. Leivant, Stratified Functional Programs and Computational Complexity, in: Proc. of 20^{th} POPL, 1993, pp. 325–333.
[27] W. G. Handley, S. S. Wainer, Complexity of Primitive Recursion, in: Computational Logic, NATO ASI Series F: Computer and Systems Science, Vol. 165, 1999, pp. 273–300.
[28] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
[29] N. Hirokawa, G. Moser, Automated Complexity Analysis Based on the Dependency Pair Method, in: Proc. of 4^{th} IJCAR, Vol. 5195 of Lecture Notes in Artificial Intelligence, 2008, pp. 364–380.
[30] U. Dal Lago, S. Martini, On Constructor Rewrite Systems and the Lambda-Calculus, in: Proc. of 36^{th} ICALP, Vol. 5556 of Lecture Notes in Computer Science, 2009, pp. 163–174.
[31] M. Avanzini, G. Moser, Complexity Analysis by Graph Rewriting, in: Proc. of 10^{th} FLOPS, Vol. 6009 of Lecture Notes in Computer Science, 2010, pp. 257–271.
[32] M. Avanzini, G. Moser, Closing the Gap Between Runtime Complexity and Polytime Computability, in: Proc. of 21^{st} RTA, Vol. 6 of Leibniz International Proceedings in Informatics, 2010, pp. 33–48.
[33] J. C. Shepherdson, H. E. Sturgis, Computability of Recursive Functions, Journal of the Association for Computing Machinery 10 (1963) 217–255.
[34] M. F. Ferreira, Termination of term rewriting. well-foundedness, totality and transformations, Ph.D. thesis, University of Utrecht, Faculty for Computer Science (1995).
[35] J. Steinbach, U. Kühler, Check your ordering - termination proofs and open problems, Tech. Rep. SEKI-Report SR-90-25, University of Kaiserslautern (1990).
[36] J. Endrullis, J. Waldmann, H. Zantema, Matrix Interpretations for Proving Termination of Term Rewriting, Journal of Automated Reasoning 40 (3) (2008) 195–220.
[37] D. Lankford, On Proving Term Rewriting Systems are Noetherian, Tech. Rep. MTP-3, Louisiana Technical University (1979).
[38] H. Zankl, M. Korp, Modular Complexity Analysis via Relative Complexity, in: Proc. of 21^{st} RTA, Vol. 6 of Leibniz International Proceedings in Informatics, 2010, pp. 385–400.