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

Martin Avanzini[1], Naohi Eguchi[2], and Georg Moser[1]

[1] Institute of Computer Science, University of Innsbruck, Austria
{martin.avanzini,georg.moser}@uibk.ac.at
[2] Mathematical Institute, Tohoku University, Japan
eguchi@math.tohoku.ac.jp

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 upto depth d, the (innermost) runtime complexity is polynomially bounded of degree d. This bound is tight.

Keywords Term Rewriting, Implicit Computational Complexity, Runtime Complexity, Polynomial Time Functions

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*. Furthermore 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 $O(n^d)$.

[⋆] This work is partially supported by FWF (Austrian Science Fund) project I-608-N18 and by a grant of the University of Innsbruck. The second author is supported by a grant from the John Templeton Foundation for the project "Philosophical Frontiers in Reverse Mathematics".

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) 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 13). Note that these functions are computable on a register machine operating in time $\mathsf{O}(n^d)$.
– 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 3). 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 of sPOP*, admitting the same properties as above, that allows to handle more general recursion schemes that make use of *parameter substitution* (cf. Theorem 16).
– 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* $\mathsf{T_CT}$, version 1.9, an open source complexity analyser.[3] The experimental evidence obtained indicates the efficiency of the method and the obtained increase in precision.

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 [2] for an overview on ICC.

Notable the clearest connection of our work is to Marion's *light multiset path order* (*LMPO* for short) [3] and the *polynomial path order* (*POP** for short) [4,5,6]. Both orders form a strict extension of the here proposed order sPOP*, but lack the precision of the latter. Although LMPO characterises FP,

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

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 n^d.

In Bonfante et. al. [7] restricted classes of polynomial interpretations are studied that can be employed to obtain polynomial upper bounds on the runtime complexity of TRSs. None of the above results are applicable to relate the depth of recursion to the runtime complexity, in the sense mentioned above. We have also drawn motivation from [8] which provides a related fine-grained classification of the polytime computable functions, but which lacks applicability in the context of runtime complexity analysis.

Polynomial complexity analysis is an active research area in rewriting. Starting from [9] interest in this field greatly increased over the last years, see for example [10,11,12,13,14]. This is partly due to the incorporation of a dedicated category for complexity into the annual termination competition (TERMCOMP).[4] 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. [15] provide an automatic multivariate amortised cost analysis exploiting typing, which extends earlier results on amortised cost analysis. To indicate the applicability of our method we have employed a straightforward (and complexity preserving) transformation of the RAML programs considered in [15] into TRSs. Equipped with sPOP* our complexity analyser TCT can handle all examples from [15] and yields (asymptotically) optimal bounds. Finally Albert et al. [16] present an automated complexity tool for Java Bytecode programs, Alias et al. [17] give a complexity and termination analysis for flowchart programs, and Gulwani et al. [18] as well as Zuleger et al. [19] provide an automated complexity tool for C programs.

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

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 [20] and Leivant [21]. 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 [22], we show that $\mathcal{B}_{\mathsf{wsc}}$ captures the polytime functions. We formulate

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

the class $\mathcal{B}_{\mathsf{wsc}}$ over the set $\{0,1\}^*$ of binary words, where we write ϵ to denote the empty sequence and $S_i(;x)$ to denote the word xi.

The arguments of every function are partitioned into normal and safe ones. Notationally we write $f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$ where *normal* arguments are to the left, and *safe* arguments to the right of the semicolon. Abbreviate $\boldsymbol{x} = x_1,\dots,x_k$ and $\boldsymbol{y} = y_1,\dots,y_l$. 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.

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

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

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. 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**): $f(\boldsymbol{x}\,;\boldsymbol{y}) = h(\boldsymbol{i}(\boldsymbol{x}\,;)\,;\boldsymbol{j}(\boldsymbol{x}\,;\boldsymbol{y}))$. Hence in $\mathcal{B}$, normal, that is recursion, parameters can grow and consequently one cannot in general relate the number of nested applications of safe recursion to the runtime complexity of the defined function.

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 can be formed from an 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*. In Section 4 we prove correctness of sPOP* with respect to runtime complexity analysis. In Section 5 we show that the order is complete for FP, in particular we precisely relate sPOP* to the class $\mathcal{B}_{\mathsf{wsc}}$. In Section 6 we incorporate parameter substitution. Finally in Section 7 we conclude and provide ample experimental evidence. Due to space restrictions some proofs have been omitted. These can be found in the full version [23].

2 Preliminaries

We denote by $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, \dots\}$. For a binary relation R 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 of R. We write $a \mathrel{R} b$ for $(a, b) \in R$ and call R *well-founded* if there exists no infinite sequence $a_1 \mathrel{R} a_2 \mathrel{R} a_3 \mathrel{R} \dots$.

We assume at least nodding acquaintance with the basics of term rewriting [24]. We fix a countably infinite set of *variables* $\mathcal{V}$ and a finite set of *function symbols* $\mathcal{F}$, the *signature*. The set of terms formed from $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The signature $\mathcal{F}$ contains a distinguished set of *constructors* $\mathcal{C}$, elements of $\mathcal{T}(\mathcal{C}, \mathcal{V})$ are called *values*. Elements of $\mathcal{F}$ that are not constructors are called *defined symbols* and collected in $\mathcal{D}$. For a term t, the *size* of t is denoted by $|t|$ and refers to the number of symbols occurring in 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* $\xrightarrow{}_{\mathcal{R}}$ by

$$(i)\ \frac{f(l_1, \dots, l_n) \to r \in \mathcal{R},\ \sigma \colon \mathcal{V} \to \mathcal{T}(\mathcal{C}, \mathcal{V})}{f(l_1\sigma, \dots, l_n\sigma) \xrightarrow{}_{\mathcal{R}} r\sigma} \qquad (ii)\ \frac{s \xrightarrow{}_{\mathcal{R}} t}{f(\dots, s, \dots) \xrightarrow{}_{\mathcal{R}} f(\dots, t, \dots)}\ .$$

If $s \xrightarrow{}_{\mathcal{R}} t$ 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 $f(l_1, \dots, l_n) \to r \in \mathcal{R}$; 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*,[5] *orthogonal constructor* TRS [24]. Furthermore we are only concerned with *innermost* rewriting. Note that orthogonality enforces that our model of computation is deterministic,[6] in particular when $\mathcal{R}$ is *terminating*, i.e. when $\xrightarrow{}_{\mathcal{R}}$ is well-founded, the semantics given as follows is well defined. For every n-ary defined symbol $f \in \mathcal{D}$, $\mathcal{R}$ defines a partial function $[\![f]\!] \colon \mathcal{T}(\mathcal{C}, \mathcal{V})^n \to \mathcal{T}(\mathcal{C}, \mathcal{V})$

[5] The restriction is not necessary, but simplifies our presentation, compare [6].

[6] As in [6] it is possible to adopt nondeterministic semantics, dropping orthogonality.

where

$$[\![f]\!](u_1,\dots,u_n) \coloneqq v \quad :\Leftrightarrow \quad \exists v. f(u_1,\dots,u_n) \to_{\mathcal{R}} \dots \to_{\mathcal{R}} v \text{ with } v \in \mathcal{T}(\mathcal{C},\mathcal{V})$$

and $[\![f]\!](u_1,\dots,u_n)$ is undefined otherwise.

Following [10] we adopt a unitary cost model. Reductions are of course measured in the size of terms. Let $\mathcal{T}_{\mathsf{b}}(\mathcal{F},\mathcal{V})$ denote the set of *basic* terms $f(u_1,\dots,u_n)$ where $f \in \mathcal{D}$ and $u_1,\dots,u_n \in \mathcal{T}(\mathcal{C},\mathcal{V})$. We define the *(innermost) runtime complexity function* $\mathrm{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N}$ as

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

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. In [25,26] it has been shown that the unitary cost model is reasonable: all functions $[\![f]\!]$ computed by $\mathcal{R}$ are computable on a conventional models of computation in time related polynomial 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 ($f \dashv_{\mathcal{R}} g$ for short), 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 $f \dashv_{\mathcal{R}}^{+} f$ holds, i.e., is defined based on itself. Noteworthy our notion also captures mutual recursion. Recursive functions are collected in $\mathcal{D}_{\mathsf{rec}}^{\succcurlyeq} \subseteq \mathcal{D}$. We denote by $\succcurlyeq$ least *preorder*, i.e., reflexive and transitive relation, on $\mathcal{F}$ containing $\dashv_{\mathcal{R}}$ and where constructors are equivalent, i.e., $c \succcurlyeq d$ for all constructors $c, d \in \mathcal{C}$. The preorder $\succcurlyeq$ is called the *precedence* of $\mathcal{R}$. We denote by $\succ$ and $\sim$ the usual separation of $\succcurlyeq$ into a proper order $\succ$ an an equivalence $\sim$. Kindly note that for $f \sim g$, if $g \in \mathcal{C}$ then also $f \in \mathcal{C}$; similar if $g \in \mathcal{D}_{\mathsf{rec}}^{\succcurlyeq}$ then also $f \in \mathcal{D}_{\mathsf{rec}}^{\succcurlyeq}$. The *depth of recursion* $\mathsf{rd}(f)$ of $f \in \mathcal{F}$ is defined as follows: let $d = \max\{\mathsf{rd}(g) \mid f \succ g\}$ be the maximal recursion depth of a function symbol g underlying the definition of f; then $\mathsf{rd}(f) \coloneqq 1 + d$ if f is recursive, otherwise $\mathsf{rd}(f) \coloneqq d$.

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

1: $+(\mathsf{0}\,;y) \to y$	3: $+(\mathsf{s}(x)\,;y) \to \mathsf{s}(+(x\,;y))$	5: $\mathsf{f}(x,y\,;) \to +(x\,;\times(y,y\,;))$
2: $\times(\mathsf{0},y\,;) \to \mathsf{0}$	4: $\times(\mathsf{s}(x),y\,;) \to +(y\,;\times(x,y\,;))$	

The TRS $\mathcal{R}_{\mathsf{arith}}$ follows along the line of $\mathcal{B}_{\mathsf{wsc}}$ from Figure 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 $\mathsf{f} \succ (\times) \succ (+) \succ \mathsf{S} \sim \mathsf{0}$ where addition $(+)$ and multiplication $(\times)$ are recursive, but f is not. Conclusively $\mathsf{rd}(+) = 1$, as f is not recursive we have $\mathsf{rd}(\mathsf{f}) = \mathsf{rd}(\times) = 2$.

3 The Small Polynomial Path Order

We arrive at the formal definition of sPOP*. Technically this order is a tamed recursive path order with product status, embodying *predicative analysis* of recursion set forth by Bellantoni and Cook [1]. We assume the arguments of defined symbol are separated into two kinds (by semicolon), normal and safe argument positions, cf. Fig. 1. For constructors we fix that all argument positions are safe. We denote by $>_{\mathsf{spop*}}$ the particular sPOP* based on the precedence $\succcurlyeq$ underlying the analysed TRS $\mathcal{R}$ and the aforementioned separation of argument positions.

The order $>_{\mathsf{spop*}}$ relies on some auxiliary relations. First of all, we lift equivalence $\sim$ underlying the precedence $\succcurlyeq$ to terms in the obvious way, but additionally disregarding the order on arguments: s and t are *equivalent*, in notation $s \sim t$, if $s = t$, or $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_n)$ where $f \sim g$ and $s_i \sim t_{\pi(i)}$ for all arguments and some permutation π. *Safe equivalence* $\overset{s}{\sim} \subseteq \sim$ takes also the separation of argument positions into account: 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 $\overset{s}{\sim}$) preserves values: if $s \sim t$ and $s \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ then $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$. We extend the (proper) subterm relation to term equivalence. Consider $s = f(s_1, \dots, s_k \,; s_{k+1}, \dots, s_{k+l})$. Then $s \rhd_{/\sim} t$ if $s_i \unrhd_{/\sim} t$ for some s_i $(i = 1, \dots k + l)$, where $\unrhd_{/\sim} = \sim \cup \rhd_{/\sim}$. Further $s \rhd^{\mathsf{n}}_{/\sim} t$ if $s_i \unrhd_{/\sim} t$ for some *normal* argument position $(i = 1, \dots, k)$.

Definition 2. *Let s and t be terms such that $s = f(s_1, \dots, s_k \,; s_{k+1}, \dots, s_{k+l})$. Then $s >_{\mathsf{spop*}} t$ if one of the following alternatives holds.*

1. *$s_i \geqslant_{\mathsf{spop*}} t$ for some argument s_i of s.*
2. *$f \in \mathcal{D}$, $t = g(t_1, \dots, t_m \,; t_{m+1}, \dots, t_{m+n})$ where $f \succ g$ and the following holds:*
   - *$s \rhd^{\mathsf{n}}_{/\sim} t_j$ for all normal arguments $t_1, \dots, t_m$;*
   - *$s >_{\mathsf{spop*}} t_j$ 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. *$f \in \mathcal{D}^{\succcurlyeq}_{\mathsf{rec}}$, $t = g(t_1, \dots, t_k \,; t_{k+1}, \dots, t_{k+l})$ where $f \sim g$ and the following holds:*
   - *$\langle s_1, \dots, s_k \rangle >_{\mathsf{spop*}} \langle t_{\pi(1)}, \dots, t_{\pi(k)} \rangle$ for some permutation π on $\{1, \dots, k\}$;*
   - *$\langle s_{k+1}, \dots, s_{k+l} \rangle \geqslant_{\mathsf{spop*}} \langle t_{\tau(k+1)}, \dots, t_{\tau(k+l)} \rangle$ for some permutation τ on $\{k + 1, \dots, k + l\}$.*

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

We say that the TRS $\mathcal{R}$ is *compatible* with $>_{\mathsf{spop*}}$ if 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, compatiblity with sPOP* implies termination. Furthermore 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.

We write $>^{\langle \mathsf{i} \rangle}_{\mathsf{spop*}}$ to refer to the i^{th} case in Definition 2. Consider the orientation of a rule $f(l_1, \dots, l_n) \to r \in \mathcal{R}$. The case $>^{\langle 2 \rangle}_{\mathsf{spop*}}$ is intended to capture functions f defined by weak safe composition (**WSC**), compare Fig. 1 on page 4. In particular the use of $\rhd^{\mathsf{n}}/_{\sim}$ 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. 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 main result.

Theorem 3. *Let $\mathcal{R}$ be a predicative recursive TRS of degree d. Then the innermost derivation height of any basic term $f(\boldsymbol{u}\,;\boldsymbol{v})$ is bounded by a polynomial of degree $\mathsf{rd}(f)$ in the sum of the depths of normal arguments $\boldsymbol{u}$. In particular, the innermost runtime complexity of $\mathcal{R}$ is bounded by a polynomial of degree d.*

The admittedly technical proof is postponed to the next section. We finish this section with an informal account of Definition 2 in our running example.

Example 4. 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 $\mathcal{D}^{\geqslant}_{\mathsf{rec}} = \{(\times),(+)\}$. The degree of recursion of $\mathcal{R}_{\mathsf{arith}}$ is thus 2.

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 $y \mathrel{\dot{\approx}} y$ and likewise $\times(\mathsf{0},y\,;) >^{\langle 1 \rangle}_{\mathsf{spop}*} \mathsf{0}$ using $\mathsf{0} \mathrel{\dot{\approx}} \mathsf{0}$. Observe that rule 5 defining f by composition is oriented by $>^{\langle 2 \rangle}_{\mathsf{spop}*}$ only: $\mathsf{f}(x,y\,;) >^{\langle 2 \rangle}_{\mathsf{spop}*} +(x\,;\times(y,y\,;))$ as $\mathsf{f} > +$, $\mathsf{f}(x,y\,;) \rhd^{\mathsf{n}}/_{\sim}\, x$, i.e., x occurs as a normal argument of f, and recursively $\mathsf{f}(x,y\,;) >^{\langle 2 \rangle}_{\mathsf{spop}*} \times(y,y\,;)$, using $\mathsf{f} > (\times)$ and $\mathsf{f}(x,y\,;) \rhd^{\mathsf{n}}/_{\sim}\, y$ (twice).

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 rule 4: $\times(\mathsf{s}(x),y\,;) >^{\langle 2 \rangle}_{\mathsf{spop}*} +(y\,;\times(x,y\,;))$ simplifies using $(\times) > (+)$ to $\times(\mathsf{s}(x),y\,;) \rhd^{\mathsf{n}}/_{\sim}\, y$ and $\times(\mathsf{s}(x),y\,;) >_{\mathsf{spop}*} \times(x,y\,;)$. As $(\times) \in \mathcal{D}^{\geqslant}_{\mathsf{rec}}$, using $>^{\langle 3 \rangle}_{\mathsf{spop}*}$ the constraint reduces to $\langle \mathsf{s}(x),y \rangle >_{\mathsf{spop}*} \langle x,y \rangle$ (which follows as $\mathsf{s}(x) >^{\langle 1 \rangle}_{\mathsf{spop}*} x$ and $y \mathrel{\dot{\approx}} y$) and $\langle\rangle \geqslant_{\mathsf{spop}*} \langle\rangle$. The careful reader might ask why both arguments position of $(\times)$ are normal. Clearly the former constraint dictates that the first position is normal. By similar reasoning the orientation $+(\mathsf{s}(x)\,;y) >_{\mathsf{spop}*} \mathsf{s}(+(x\,;y))$ of rule 3 dictates that the first argument position of $(+)$ is normal. As the second argument to multiplication is substituted into the normal argument position of addition, $\times(\mathsf{s}(x),y\,;) \rhd^{\mathsf{n}}/_{\sim}\, y$ correctly propagates that y is a recursion parameter. Reconsidering the orientation of rule 5 defining f, $\rhd^{\mathsf{n}}/_{\sim}$ propagates that f takes only normal arguments.

We conclude that $\mathcal{R}_{\mathsf{arith}}$ is predicative recursive, with degree 2. By Theorem 3 runtime of $\mathcal{R}_{\mathsf{arith}}$ is thus bounded by a quadratic polynomial.

As a consequence of our main theorem, any predicative recursive (and orthogonal) TRS $\mathcal{R}$ of degree d computes a function from FP, compare [26]. These functions are even computable on a register machine operating in time $\mathsf{O}(n^d)$, provided $\mathcal{R}$ computes functions over a word algebra [23,27]. The latter restriction allows storing values in registers without significant encoding overhead. We emphasise also that the bound provided in Theorem 3 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 $\mathcal{R}_i$ ($i \in \mathbb{N}$) inductively as follows: $\mathcal{R}_0 := \{\mathsf{f}_0(x\,;) \to \mathsf{a}\}$ and $\mathcal{R}_{i+1}$ extends $\mathcal{R}_i$ by the rules

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

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.

4 Soundness

We now show that sPOP* is correct, i.e., we prove Theorem 3. Let $\mathcal{R}$ denote a predicative recursive TRS. Our proof makes use of a variety of ingredients. In Definition 7 we define *predicative interpretations* S that flatten terms to *sequences of terms*, separating safe from normal arguments. In Definition 8 we introduce a family of orders $(\blacktriangleright_\ell)_{\ell \in \mathbb{N}}$ on sequences of terms. The definition of $\blacktriangleright_\ell$ (for fixed ℓ) does not explicitly mention predicative notions and is conceptually simpler than $>_{\mathsf{spop*}}$. In Lemma 11 we show that predicative interpretations S embeds rewrite steps into $\blacktriangleright_\ell$:

$$\begin{tikzcd}
s \arrow[d] & \to_{\mathcal{R}} & s_1 \arrow[d] & \to_{\mathcal{R}} & \cdots & \to_{\mathcal{R}} & s_\ell \arrow[d] \\
\mathsf{S}(s) & \blacktriangleright_\ell & \mathsf{S}(s_1) & \blacktriangleright_\ell & \cdots & \blacktriangleright_\ell & \mathsf{S}(s_\ell)
\end{tikzcd}$$

Consequently the derivation height of s is bounded by the length of $\blacktriangleright_\ell$ descending sequences, which in turn can be bounded sufficiently whenever s is basic (cf. Theorem 10).

Consider a step $C[f(\boldsymbol{u}\sigma\,;\boldsymbol{v}\sigma)] \to_{\mathcal{R}} C[r\sigma] = t$. 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) $\mathcal{T}(\mathcal{C}, \mathcal{V}) \subseteq \mathcal{T}_{\mathsf{b}}^{\to}$, and (ii) if $f \in \mathcal{F}$, $\boldsymbol{v} \subseteq \mathcal{T}(\mathcal{C}, \mathcal{V})$ and $\boldsymbol{t} \subseteq \mathcal{T}_{\mathsf{b}}^{\to}$ then $f(\boldsymbol{v}\,;\boldsymbol{t}) \in \mathcal{T}_{\mathsf{b}}^{\to}$. This set is closed under rewriting.

Lemma 6. *Let $\mathcal{R}$ be a completely defined TRS compatible with $>_{\mathsf{spop*}}$. If $s \in \mathcal{T}_{\mathsf{b}}^{\to}$ and $s \to_{\mathcal{R}} t$ then $t \in \mathcal{T}_{\mathsf{b}}^{\to}$.*

Since $\mathcal{T}_{\mathsf{b}}^{\to}$ contains in particular all basic terms, it follows that the runtime complexity function $\mathrm{rc}_{\mathcal{R}}$ depends only on terms from $\mathcal{T}_{\mathsf{b}}^{\to}$. The *predicative interpretation* S maps terms from $\mathcal{T}_{\mathsf{b}}^{\to}$ to *sequences* of *normalised* terms by separating normal from safe arguments. We sometimes write f_{n} for the symbol f if it occurs in a normalised term. If f has k normal arguments, then f_{n} has arity k. To denote sequences of terms, we use a fresh variadic function symbol $\circ$. Here variadic means that the arity of $\circ$ is finite but arbitrary. We always write $[a_1 \cdots a_n]$ for $\circ(a_1,\dots,a_n)$, and if we write $f(a_1,\dots,a_n)$ then $f \neq \circ$. We denote by T^* the set of *sequences* $[t_1 \cdots t_n]$ of normalised terms $t_1,\dots,t_n$. We lift terms equivalence to sequences by disregarding order of elements: $[s_1 \cdots s_n] \sim [t_1 \cdots t_n]$ if $s_i \sim t_{\pi(i)}$ for all $i = 1,\dots,n$ and some permutation π on $\{1,\dots,n\}$. We define *concatenation* as $[s_1 \cdots s_n] \frown [t_1 \cdots t_n] \coloneqq [\, s_1 \cdots s_n \; t_1 \cdots t_m \,]$, and extend it to terms by identifying terms t with the singleton sequences $[\, t \,]$, for instance $s \frown t = [\, s \; t \,]$.

Definition 7. *We define the* predicative interpretation S *for all* $t \in \mathcal{T}_{\mathsf{b}}^{\rightarrow}$ *as follows. If* $t \in \mathcal{T}(\mathcal{C},\mathcal{V})$*, then* $\mathsf{S}(t) \coloneqq [\,]$*. Otherwise*

$$\mathsf{S}(f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})) \coloneqq [\, f_{\mathsf{n}}(t_1,\dots,t_k) \,] \frown \mathsf{S}(t_{k+1}) \frown \cdots \frown \mathsf{S}(t_{k+l}) \; .$$

We define the *small polynomial path order on sequences* T^*. As these serve a purely technical reason, it suffices to represent the order via finite approximations $\blacktriangleright_\ell$ (compare also [4]). The parameter $\ell \in \mathbb{N}$ controls the width of terms and sequences.

Definition 8. *Let* $\succcurlyeq$ *denote a precedence. For all* $\ell \geqslant 1$ *we define* $\blacktriangleright_\ell$ *on terms and sequences of terms inductively such that:*

1. $f(s_1,\dots,s_n) \blacktriangleright_\ell g(t_1,\dots,t_m)$ *if* $f \in \mathcal{D}$, $f \succ g$ *and the following conditions hold:*
   - $f(s_1,\dots,s_n) \rhd\!\!\!/_{\sim} t_j$ *for all* $j = 1,\dots,m$;
   - $m \leqslant \ell$.
2. $f(s_1,\dots,s_n) \blacktriangleright_\ell g(t_1,\dots,t_n)$ *if* $f \in \mathcal{D}_{\mathsf{rec}}^{\succcurlyeq}$, $f \sim g$ *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. $f(s_1,\dots,s_n) \blacktriangleright_\ell [t_1 \cdots t_m]$ *if the following conditions hold:*
   - $f(s_1,\dots,s_n) \blacktriangleright_\ell t_j$ *for all* $j = 1,\dots,m$;
   - *at most one element* t_j $(j \in \{1,\dots,m\})$ *contains a symbols* g *with* $f \sim g$;
   - $m \leqslant \ell$.
4. $[s_1 \cdots s_n] \blacktriangleright_\ell [t_1 \cdots t_m]$ *if there exists terms or sequences* b_i $(i = 1,\dots,n)$ *such that:*
   - $[t_1 \cdots t_m]$ *is equivalent to* $b_1 \frown \cdots \frown b_n$;
   - $s_i \mathrel{\underline{\blacktriangleright}}_\ell b_i$ *for all* $i = 1,\dots,n$;
   - $s_{i_0} \blacktriangleright_\ell b_{i_0}$ *for at least one* $i_0 \in \{1,\dots,n\}$.

Here $a \mathrel{\underline{\blacktriangleright}}_\ell b$ *denotes that either* $a \sim b$ *or* $a \blacktriangleright_\ell b$ *holds, and* $\rhd\!\!\!/_{\sim}$ *is also used for its extension to products:* $\langle s_1,\dots,s_n\rangle \rhd\!\!\!/_{\sim} \langle t_i,\dots,t_n\rangle$ *if* $s_i \unrhd\!\!\!/_{\sim} t_i$ *for all* $i = 1,\dots,n$*, and* $s_{i_0} \rhd\!\!\!/_{\sim} t_{i_0}$ *for at least one* $i_0 \in \{1,\dots,n\}$.

The next lemma collects some facts about the order $\blacktriangleright_\ell$:

Lemma 9. *For all* $\ell \geqslant 1$*, (i)* $\blacktriangleright_\ell \subseteq \blacktriangleright_{\ell+1}$*, (ii)* $\sim \cdot \blacktriangleright_\ell \cdot \sim \subseteq \blacktriangleright_\ell$*, and (iii)* $a \blacktriangleright_\ell b$ *implies* $a \frown c \blacktriangleright_\ell b \frown c$.

Let $\ell \geqslant 1$. The function $\mathsf{G}_\ell \colon \mathsf{T}^*(\mathcal{F}) \to \mathbb{N}$ measures the length of $\blacktriangleright_\ell$ descending sequence: $\mathsf{G}_\ell(a) \coloneqq \max\{m \mid a \blacktriangleright_\ell a_1 \blacktriangleright_\ell \cdots \blacktriangleright_\ell a_m\}$. Theorem 10 binds $\mathsf{G}_\ell(s)$ for (normalised) basic terms s sufficiently.

Theorem 10. *Let $f \in \mathcal{D}$. Then $\mathsf{G}_\ell(f_\mathsf{n}(u_1, \ldots, u_k)) \leqslant c \cdot n^{\mathsf{rd}(f)}$ for all values $u_1, \ldots, u_k$, where $n := \sum_{i=1}^{k} \mathsf{dp}(u_i)$. The constant $c \in \mathbb{N}$ depends only on f and ℓ.*

The remaining missing piece in our reasoning is to show that predicative interpretations *embed* innermost rewrite steps into $\blacktriangleright_\ell$, where ℓ depends only on the considered TRS $\mathcal{R}$.

Lemma 11. *Let $\mathcal{R}$ denote a predicative recursive TRS and let ℓ be the maximal size of a right-hand side in $\mathcal{R}$. If $s \in \mathcal{T}_\mathsf{b}^{\rightarrow}$ and $s \xrightarrow{i}_{\mathcal{R}} t$ then $\mathsf{S}(s) \blacktriangleright_\ell \mathsf{S}(t)$.*

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

Proof (of Theorem 3). Let $\mathcal{R}$ denote a predicative recursive TRS. We prove the existence of a constant $c \in \mathbb{N}$ such that for all values $\boldsymbol{u}, \boldsymbol{v}$, the derivation height of $f(\boldsymbol{u}\,;\boldsymbol{v})$ is bounded by $c \cdot n^{\mathsf{rd}(f)}$, where n is the sum of the depths of normal arguments $\boldsymbol{u}$.

Consider a derivation $f(\boldsymbol{u}\,;\boldsymbol{v}) \xrightarrow{i}_{\mathcal{R}} t_1 \xrightarrow{i}_{\mathcal{R}} \cdots \xrightarrow{i}_{\mathcal{R}} t_n$. Let $i \in \{0, \ldots, n-1\}$. By Lemma 6 it follows that $t_i \in \mathcal{T}_\mathsf{b}^{\rightarrow}$, and consequently $\mathsf{S}(t_i) \blacktriangleright_\ell \mathsf{S}(t_{i+i})$ due to Lemma 11. So in particular the length n is bounded by the length of $\blacktriangleright_\ell$ descending sequences starting from $\mathsf{S}(f(\boldsymbol{u}\,;\boldsymbol{v})) = [\,f_\mathsf{n}(\boldsymbol{u})\,]$. One verifies that $\mathsf{G}_\ell([\,f_\mathsf{n}(\boldsymbol{u})\,]) = \mathsf{G}_\ell(f_\mathsf{n}(\boldsymbol{u}))$. Thus Theorem 10 gives the constant $c \in \mathbb{N}$ as desired.

5 Completeness Results

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. Here $\mathcal{B}_\mathsf{wsc}^{k,l}$ refers to the subclass of $\mathcal{B}_\mathsf{wsc}$ with k normal and l safe argument positions.

Theorem 12. *Every polynomial time computable function belongs to $\bigcup_{k \in \mathbb{N}} \mathcal{B}_\mathsf{wsc}^{k,0}$.*

One can show this fact by following the proof of Theorem 3.7 in [22], where the unary variant of $\mathcal{B}_\mathsf{wsc}$ is defined and the inclusion corresponding to Theorem 12 is shown, cf. [23].

Theorem 13. *For any $\mathcal{B}_\mathsf{wsc}$-function f there exists an orthogonal TRS $\mathcal{R}_f$ that is predicative recursive of degree d, where d equals the maximal number of nested application of* (**SRN**) *in the definition of f.*

The completeness of sPOP* for the polytime computable functions is an immediate consequence of Theorem 12 and Theorem 13. The witnessing TRS $\mathcal{R}_f$ for $f \in \mathcal{B}_\mathsf{wsc}$ in Theorem 13 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 P, $\mathsf{I}^{k,l}_j$, C and $\mathsf{O}^{k,l}$ correspond respectively to the initial functions P, $I^{k,l}_j$, C and $O^{k,l}$ of $\mathcal{B}_{\mathsf{wsc}}$. The symbol $\mathsf{SUB}[\mathsf{h}, i_1, \dots, i_n, \mathbf{g}]$ is used to denote the function obtained by composing functions h and $\boldsymbol{g}$ 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. It is easy to see that $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is a constructor TRS. Further $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is orthogonal, thus any finite restriction $\mathcal{R}_f \subseteq \mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is confluent.

Proof (of Theorem 13). Let f be a function coming from $\mathcal{B}_{\mathsf{wsc}}$. 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 $\mathcal{B}_{\mathsf{wsc}}$.

The assertion of the theorem follows from Condition (1), (3)) and (5). 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 $\mathcal{R}_g, \mathcal{R}_{h_0}, \mathcal{R}_{h_1}$ and witnessing 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 $\mathsf{f} :\equiv \mathsf{SRN}[\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1]$. 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:

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

Define the precedence $\succcurlyeq_f$ extending $\succcurlyeq_g \cup \succcurlyeq_{h_0} \cup \succcurlyeq_{h_1}$ by $\mathsf{f} \sim \mathsf{f}$ and $\mathsf{f} \succ \mathsf{g}'$ for any $\mathsf{g}' \in \{\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1\}$. Note that the union $\succcurlyeq_g \cup \succcurlyeq_{h_0} \cup \succcurlyeq_{h_1}$ is still a precedence. This can be seen as follows. Assume that both $\mathsf{f}_0 \succ \mathsf{f}_1$ and $\mathsf{f}_1 \succ \mathsf{f}_0$ hold for some symbols f_0 and f_1. Then by definition $\mathsf{f}_0 \equiv \mathcal{O}_0[\cdots \mathsf{f}_1 \cdots]$ and $\mathsf{f}_1 \equiv \mathcal{O}_1[\cdots \mathsf{f}_0 \cdots]$ for some operations $\mathcal{O}_0, \mathcal{O}_1 \in \{\mathsf{SUB}, \mathsf{SRN}\}$. This means that the function corresponding to f_0 is defined by either (**WSC**) or (**SRN**) via the function corresponding to f_1 and vice versa, but these contradict. 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, $\mathsf{f}(\epsilon, \boldsymbol{x}; \boldsymbol{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{g}(\boldsymbol{x}; \boldsymbol{y})$ 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, $\mathsf{f}(\mathsf{S}_0(;z), \boldsymbol{x}\,; \boldsymbol{y}) >^{\langle 3 \rangle}_{\mathsf{spop}*} \mathsf{f}(z, \boldsymbol{x}\,; \boldsymbol{y})$ holds.

This together with the definition of the precedence $\geqslant_f$ allows us to conclude $\mathsf{f}(\mathsf{S}_0(;z), \boldsymbol{x} ; \boldsymbol{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{h}_0(z, \boldsymbol{x} ; \boldsymbol{y}, \mathsf{f}(z, \boldsymbol{x} ; \boldsymbol{y}))$.

Consider Condition (4). For each $g' \in \{g, h_0, h_1\}$, g' is maximal in the precedence $\geqslant_{g'}$ by induction hypothesis for g'. Hence by the definition of $\geqslant_f$, f is maximal in $\geqslant_f$.

It remains to show Condition (5). Since f is a recursive symbol $\mathsf{rd}(\mathsf{f}) = 1 + \max\{\mathsf{rd}(\mathsf{g}), \mathsf{rd}(\mathsf{h}_0), \mathsf{rd}(\mathsf{h}_1)\}$. 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 $\mathcal{B}_{\mathsf{wsc}}$.

6 A Non-Trivial Closure Property of the Polytime Functions

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 $\boldsymbol{p}$ by

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

We introduce *small polynomial path order with parameter substitution* (*sPOP*$^*_{PS}$ for short), that extends clause $>^{\langle 3 \rangle}_{\mathsf{spop}*}$ to account for the schema ($\mathbf{SRN_{PS}}$).

Definition 14. *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.*
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:*
   - *$s \rhd^{\mathsf{n}}_{/\sim} t_j$ 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. *$f \in \mathcal{D}^{\geqslant}_{\mathsf{rec}}$, $t = g(t_1, \dots, t_k ; t_{k+1}, \dots, t_{k+l})$ 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 $s \overset{s}{\sim} t$ 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 sPOP$^*_{\mathrm{PS}}$ in the obvious way. Parameter substitution extends the analytical power of sPOP* significantly. In particular, sPOP* can handle tail recursion as in the following example.

Example 15. The TRS $\mathcal{R}_{\mathsf{rev}}$ consists of the three rules

$$\mathsf{rev}(xs\,;) \to \mathsf{rev}_{\mathsf{tl}}(xs\,;[\,]) \quad \mathsf{rev}_{\mathsf{tl}}([\,]\,;ys) \to ys \quad \mathsf{rev}_{\mathsf{tl}}(x:xs\,;ys) \to \mathsf{rev}_{\mathsf{tl}}(xs\,;x:ys)$$

reverses lists formed from the constructors $[\,]$ and $(:)$. Then $\mathcal{R}_{\mathsf{rev}}$ is compatible with $>_{\mathsf{spop}^*_{\mathsf{ps}}}$, but due to the last rule not with $>_{\mathsf{spop}*}$.

Still sPOP$^*_{\mathrm{PS}}$ induces polynomially bounded runtime complexity in the sense of Theorem 3. As a consequence of the next theorem, the runtime of $\mathcal{R}_{\mathsf{rev}}$ is inferred to be linear.

Theorem 16. *Let $\mathcal{R}$ be a predicative recursive TRS of degree d (with respect to Definition 14). Then the innermost derivation height of any basic term $f(\boldsymbol{u}\,;\boldsymbol{v})$ is bounded by a polynomial of degree $\mathsf{rd}(f)$ in the sum of the depths of normal arguments $\boldsymbol{u}$. In particular, the innermost runtime complexity of $\mathcal{R}$ is bounded by a polynomial of degree d.*

Corollary 17. *The class $\mathcal{B}_{\mathsf{wsc}}$ is closed under safe recursion on notation with parameter substitution. More precisely, for any functions $g, h_0, h_1, \boldsymbol{p} \in \mathcal{B}_{\mathsf{wsc}}$, there exists a unique polytime computable function f such that $f(\epsilon, \boldsymbol{x}\,;\boldsymbol{y}) = g(\boldsymbol{x}\,;\boldsymbol{y})$ and $f(zi, \boldsymbol{x}\,;\boldsymbol{y}) = h_i(z, \boldsymbol{x}\,;\boldsymbol{y}, f(z, \boldsymbol{x}, \boldsymbol{p}(z, \boldsymbol{x}\,;\boldsymbol{y})))$ for each $i = 0, 1$.*

Furthermore sPOP$^*_{\mathrm{PS}}$ is complete for the polytime computable functions. To state a stronger completeness result, we extend the class $\mathcal{B}_{\mathsf{wsc}}$ to a class $\mathcal{B}_{\mathsf{wsc+ps}}$ that is also closed under the scheme $(\mathbf{SRN_{PS}})$. Then sPOP$^*_{\mathrm{PS}}$ is complete for $\mathcal{B}_{\mathsf{wsc+ps}}$ in the sense of Theorem 13.

Theorem 18. *For any $\mathcal{B}_{\mathsf{wsc+ps}}$-function f there exists a confluent TRS $\mathcal{R}_f$ that is predicative recursive of degree d (with respect to Definition 14), where d equals the maximal number of nested application of $(\mathbf{SRN_{PS}})$ in the definition of f.*

7 Conclusion

We propose a new order, the small polynomial path order sPOP*. 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 $O(n^d)$. This termination order induces 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.

Moreover, sPOP* gives rise to a new, fully automatic, syntactic method for polynomial runtime complexity analysis. We performed experiments on the relative power of sPOP* (respectively sPOP$^*_{\mathrm{PS}}$) with respect to LMPO [3] and POP* [5]. We selected two test-suites: test-suite TC constitutes of 597 terminating constructor TRSs and test-suite TCO, containing 290 examples, resulting from restricting test-suite TC to orthogonal systems.[7] On the larger benchmark TC, LMPO proves an exponential bound on a subset of 57 examples. For

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

four examples this bound is indeed tight, out of the remaining POP* can verify polynomially bounded runtime complexity on 43 examples. 39 examples of these can also be handled with sPOP* and for all these examples the runtime complexity is at most cubic. Thus sPOP* brings about a significant increase in precision, which accompanied with only minor decrease in power. This assessment remains true, if we consider the smaller benchmark set TCO. sPOP$^*_{\mathrm{PS}}$ increases the analytic power of POP* on test-suite TC from 39 to 54 examples, from the 15 new examples 13 cannot be handled by any other technique.[8]

To test the applicability of sPOP* in the context of program analysis, we have employed a straightforward (and complexity preserving) transformation of RAML programs considered in [15] into TRSs. In Table 2 we present the performance of TCT on this test-suite. Equipped with sPOP$^*_{\mathrm{PS}}$ our complexity analyser TCT can handle all examples in [15] and all but 5 of the RAML test-suite [28]. This is a noteworthy performance as the transformation from RAML programs to TRSs used amounts to a straightforward (almost naive) program transformation.Furthermore the dedicated prototype crucially exploits the fact that RAML programs are typed. On the other hand TCT works on standard, that is untyped, TRSs.

	TCT	TCT/sPOP$^*_{\mathrm{PS}}$
O(n)	3\ 2.84	3\ 3.65
O(n^2)	14\ 13.78	15\ 14.70
O(n^3)	15\ 38.80	16\ 39.68
unknown	6\ 36.41	5\ 41.94

Fig. 2. Empirical Evaluation on translated RAML sources

References

1. Bellantoni, S., Cook, S.: A new Recursion-Theoretic Characterization of the Polytime Functions. CC **2**(2) (1992) 97–110
2. Baillot, P., Marion, J.Y., Rocca, S.R.D.: Guest Editorial: Special Issue on Implicit Computational Complexity. TOCL **10**(4) (2009)
3. Marion, J.Y.: Analysing the Implicit Complexity of Programs. IC **183** (2003) 2–18
4. Arai, T., Moser, G.: Proofs of Termination of Rewrite Systems for Polytime Functions. In: Proc. of 15th FSTTCS. Volume 3821 of LNCS. (2005) 529–540
5. Avanzini, M., Moser, G.: Complexity Analysis by Rewriting. In: Proc. of 9th FLOPS. Volume 4989 of LNCS. (2008) 130–146
6. Avanzini, M., , Moser, G.: Polynomial Path Orders: A Maximal Model. CoRR **cs/CC/1209.3793** (2012) Available at `http://www.arxiv.org/`.
7. Bonfante, G., Cichon, A., Marion, J.Y., Touzet, H.: Algorithms with Polynomial Interpretation Termination Proof. JFP **11**(1) (2001) 33–53
8. Marion, J.Y.: On Tiered Small Jump Operators. LMCS **5**(1) (2009)
9. Moser, G., Schnabl, A.: Proving Quadratic Derivational Complexities Using Context Dependent Interpretations. In: Proc. 19th RTA. Volume 5117 of LNCS. (2008) 276–290

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

10. Hirokawa, N., Moser, G.: Automated Complexity Analysis Based on the Dependency Pair Method. In: Proc. of 4th IJCAR. Volume 5195 of LNAI. (2008) 364–380
11. Zankl, H., Korp, M.: Modular Complexity Analysis via Relative Complexity. In: Proc. of 21st RTA. Volume 6 of LIPIcs. (2010) 385–400
12. Noschinski, L., Emmes, F., Giesl, J.: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems. In: Proc. of 23rd CADE. Volume 6803 of LNAI. (2011) 422–438
13. Hirokawa, N., Moser, G.: Automated Complexity Analysis Based on the Dependency Pair Method. CoRR **abs/1102.3129** (2011) submitted.
14. Middeldorp, A., Moser, G., Neurauter, F., Waldmann, J., Zankl, H.: Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems. In: Proc. of 4th CAI. Volume 6472 of LNCS. (2011) 1–20
15. Hoffmann, J., Aehlig, K., Hofmann, M.: Multivariate Amortized Resource Analysis. In: Proc. of 38th POPL, ACM (2011) 357–370
16. Albert, E., Arenas, P., Genaim, S., Gómez-Zamalloa, M., Puebla, G., Ramírez, D., Román, G., Zanardini, D.: Termination and Cost Analysis with COSTA and its User Interfaces. ENTCS **258**(1) (2009) 109–121
17. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs. In: Proc. 17th SAS. Volume 6337 of LNCS. (2010) 117–133
18. Gulwani, S., Mehra, K., Chilimbi, T.: SPEED: Precise and Efficient Static Estimation of Program Computational Complexity. In: Proc. of 36th POPL, ACM (2009) 127–139
19. Zuleger, F., Gulwani, S., Sinn, M., Veith, H.: Bound Analysis of Imperative Programs with the Size-Change Abstraction. In: Proc. of 18th SAS. Volume 6887 of LNCS. (2011) 280–297
20. Simmons, H.: The Realm of Primitive Recursion. AML **27** (1988) 177–188
21. Leivant, D.: A Foundational Delineation of Computational Feasiblity. In: Proc. of 6th LICS, IEEE Computer Society (1991) 2–11
22. Handley, W.G., Wainer, S.S.: Complexity of Primitive Recursion. In: Computational Logic, NATO ASI Series F: Computer and Systems Science. Volume 165. (1999) 273–300
23. Avanzini, M., Eguchi, N., Moser, G.: A New Order-theoretic Characterisation of the Polytime Computable Functions. CoRR **cs/CC/1201.2553** (2012) Available at `http://www.arxiv.org/`.
24. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
25. Dal Lago, U., Martini, S.: On Constructor Rewrite Systems and the Lambda-Calculus. In: Proc. of 36th ICALP. Volume 5556 of LNCS. (2009) 163–174
26. Avanzini, M., Moser, G.: Closing the Gap Between Runtime Complexity and Polytime Computability. In: Proc. of 21st RTA. Volume 6 of LIPIcs. (2010) 33–48
27. Avanzini, M., Eguchi, N., Moser, G.: On a Correspondence between Predicative Recursion and Register Machines. In: Proc. of 12th WST. (2012) 15–19, available at `http://cl-informatik.uibk.ac.at/users/georg/events/wst2012/`.
28. Hoffmann, J., Aehlig, K., Hofmann, M.: Resource aware ml. In: CAV. Volume 7358 of LNCS. (2012) 781–786