

## POLYNOMIAL PATH ORDERS

MARTIN AVANZINI AND GEORG MOSER

Institute of Computer Science, University of Innsbruck, Austria  
*e-mail address:* {martin.avanzini,georg.moser}@uibk.ac.at

---

**ABSTRACT.** This paper is concerned with the complexity analysis of constructor term rewrite systems and its ramification in implicit computational complexity. We introduce a path order with multiset status, the *polynomial path order* POP\*, that is applicable in two related, but distinct contexts. On the one hand POP\* induces polynomial innermost runtime complexity and hence may serve as a syntactic, and fully automatable, method to analyse the innermost runtime complexity of term rewrite systems. On the other hand POP\* provides an order-theoretic characterisation of the polytime computable functions: the polytime computable functions are exactly the functions computable by an orthogonal constructor TRS compatible with POP\*.

### 1. INTRODUCTION

In this paper we are concerned with the complexity analysis of constructor term rewrite systems. Since term rewrite systems (TRSs for short) underlie much of declarative programming, time complexity of functions defined by TRSs is of particular interest.

In rewriting two notions of complexity have been widely studied. Hofbauer and Lautemann proposed to assess the complexity of a given TRS as the maximal length of derivation sequences. More precisely the *derivational complexity function* relates the maximal length of a derivation with the size of the starting term [33]. As an alternative Hirokawa and the second author proposed to study the *runtime complexity function* [30], which forms a variation of the derivational complexity function. Instead of all possible derivations, one studies only derivations with starting terms whose arguments are constructor terms (aka *basic terms*), see also [21]. In the context of this paper, runtime complexity is the more natural notion. We emphasise that the runtime complexity of a rewrite system forms a *polynomially invariant* cost model [19], cf. Section 2.

To motivate our studies, we present a natural encoding of the well-known satisfiability problem SAT of propositional logic as a TRS. Given a propositional formula in conjunctive

---

*2012 ACM CCS:* [Theory of computation]: Computational complexity and cryptography—Complexity classes / Complexity theory and logic; Formal languages and automata theory—Formalisms—Rewrite systems; Logic—Logic and verification.

*Key words and phrases:* Term Rewriting, Complexity Analysis, Implicit Computational Complexity, Automation.

This work is partially supported by FWF (Austrian Science Fund) project I-603-N18.

normal form, the TRS  $\mathcal{R}_{\text{sat}}$  given below computes a satisfying assignment if it exists. Note that  $\mathcal{R}_{\text{sat}}$  is not confluent, i.e., the computation is performed nondeterministically. The rewrite system  $\mathcal{R}_{\text{sat}}$  thus encodes the function problem FSAT associated with the satisfiability problem. FSAT is complete for the class of *function problems over NP* (FNP for short). See Section 2 where FNP is formally defined. As corollary to the polynomial invariance of the runtime complexity of rewrite systems, we obtain that the runtime complexity of  $\mathcal{R}_{\text{sat}}$  is expected to be polynomial.

**Example 1.1.** Consider the following (non-confluent) TRS  $\mathcal{R}_{\text{sat}}$ :<sup>1</sup>

- |  |  |
|--|--|
| 1: $\text{negate}(+x) \rightarrow -x$  | 2: $\text{negate}(-x) \rightarrow +x$  |
| 3: $\text{eq}(c_0(x), c_1(y)) \rightarrow \text{ff}$                                 | 4: $\text{eq}(c_0(x), c_0(y)) \rightarrow \text{eq}(x, y)$   |
| 5: $\text{eq}(c_1(x), c_0(y)) \rightarrow \text{ff}$                                 | 6: $\text{eq}(c_1(x), c_1(y)) \rightarrow \text{eq}(x, y)$   |
| 7: $\text{eq}(-x, +y) \rightarrow \text{ff}$   | 8: $\text{eq}(-x, -y) \rightarrow \text{eq}(x, y)$   |
| 9: $\text{eq}(+x, -y) \rightarrow \text{ff}$   | 10: $\text{eq}(+x, -y) \rightarrow \text{eq}(x, y)$  |
| 11: $\text{eq}(\varepsilon, \varepsilon) \rightarrow \text{tt}$                      |  |
| 12: $\text{if}(\text{tt}, t, e) \rightarrow t$                                       | 13: $\text{if}(\text{ff}, t, e) \rightarrow e$   |
| 14: $\text{consistent}([]) \rightarrow \text{tt}$                                    | 15: $\text{consistent}(l : ls) \rightarrow \text{if}(\text{elem}(\text{negate}(l), ls), \text{ff}, \text{consistent}(ls))$ |
| 16: $\text{elem}(x, []) \rightarrow \text{ff}$                                       | 17: $\text{elem}(x, y : ys) \rightarrow \text{if}(\text{eq}(x, y), \text{tt}, \text{elem}(x, ys))$                         |
| 18: $\text{guess}([]) \rightarrow []$  | 19: $\text{guess}(c : cs) \rightarrow \text{choice}(c) : \text{guess}(cs)$   |
| 20: $\text{choice}(a : []) \rightarrow a$  | 21: $\text{choice}(a : b : bs) \rightarrow \text{choice}(b : bs)$  |
| 22: $\text{choice}(a : b : bs) \rightarrow a$  |  |
| 23: $\text{sat}(cs) \rightarrow \text{sat}'(\text{guess}(cs))$                       |  |
| 24: $\text{sat}'(as) \rightarrow \text{if}(\text{consistent}(as), as, \text{unsat})$ |  |

Atoms are encoded as binary strings (built from the constant  $\varepsilon$ , and unary constructors  $c_0$  and  $c_1$ ), the unary constructors  $(+)$  and  $(-)$  lift atoms to positive and negative literals respectively. The rules (1)–(11) define negation and equality on this representation of literals.

Clauses are lists of literals, clause sets are denoted by lists of clauses. Lists are constructed in the usual way using a constant  $[]$  and binary constructor  $(:)$ . Call a list of literals consistent, if an atom does not occur positively and negatively. This is formalised by rules (14)–(17). A clause set  $cs$  is satisfiable if and only if there exists a list of literals  $as$ , denoting a satisfying assignment, such that  $as$  is consistent and contains at least one literal from every clause  $c$ . The rules (18)–(22) are used to generate a candidate list  $as$  that contains for each clause one literal. Using these auxiliary rules, the algorithm is implemented by rules (23) and (24). Given a clause set  $cs$ , a candidate list  $as$  is guessed and its consistency is checked. If this check succeeds the list  $as$  is returned.

<sup>1</sup>This is a slight variant of Example TCT\_12/sat.xml in the current Termination Problem Database (TPDB) version 8.0.

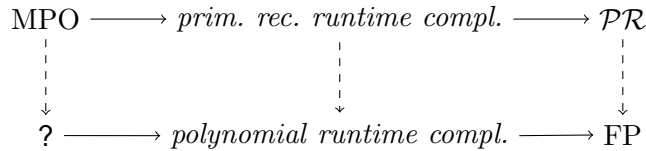


Figure 1: The Quest for “Polynomial Path Orders”

It is easy to see that  $\mathcal{R}_{\text{sat}}$  is terminating, for example this can be verified by showing compatibility with the multiset path order (MPO for short) [56]. It is a standard exercise in rewriting to assess the complexity of rewrite systems via an analysis of termination techniques and it is a well-known result that MPO induces primitive recursive derivational and runtime complexity [23, 32, 47]. Furthermore, MPO *characterises* the *primitive recursive functions* ( $\mathcal{PR}$  for short) [24]: any function computed by an MPO-terminating TRS is primitive recursive, vice versa, any primitive recursive function can be stated as an MPO-terminating TRS. However, from these results we can only conclude that the runtime complexity function of  $\mathcal{R}_{\text{sat}}$  is bounded by a primitive recursive function, which is hardly revealing. This motivates the quest for a “polynomial path order” depicted by the question mark in Figure 1.<sup>2</sup> Such an order should be a restriction of MPO, but *miniaturises* its properties: it would be expected that this order induces *polynomial* runtime complexity and provides a characterisation of the class FP of *functions computable in polynomial time*.

In this paper, we propose the *polynomial path order* ( $\text{POP}^*$  for short) as such a miniaturisation of MPO:  $\text{POP}^*$  induces polynomial runtime complexity (for innermost rewriting) and at the same time yields a characterisation of FP. In the design of  $\text{POP}^*$  we have striven for a *maximal* miniaturisation of MPO, so that these key features of  $\text{POP}^*$  remain intact. Alas, some of the essential properties of MPO cannot be preserved. First,  $\text{POP}^*$  can only analyse the *runtime complexity* of TRSs; the derivational complexity induced by  $\text{POP}^*$  is (at least) double-exponential (Example 3.13). Second, the restriction to *innermost* rewriting is essential (Example 3.14) and finally, our result only holds for *constructor* TRS (Example 3.15). More precisely, we establish the following results.

- (1)  $\text{POP}^*$  induces polynomial innermost runtime complexity on constructor TRSs. That is, the innermost runtime complexity function for a constructor TRS compatible with  $\text{POP}^*$  is polynomially bounded (Theorem 3.11).
- (2)  $\text{POP}^*$  captures exactly the class FP on orthogonal constructor TRSs. That is, any orthogonal constructor TRS compatible with  $\text{POP}^*$  computes a polytime function. On the other hand, any function in FP can be implemented by an orthogonal constructor TRS compatible with  $\text{POP}^*$  (Theorems 6.1 and 6.4).
- (3) We extend upon  $\text{POP}^*$  by proposing a generalisation  $\text{POP}_{\text{PS}}^*$ , admitting the same properties as outlined above, but that allows to handle more general recursion schemes that make use of parameter substitution (Theorem 7.5).
- (4) We have implemented the proposed technique in the *Tyrolean Complexity Tool* ( $\mathbb{TCT}$  for short) [11]. The experimental evidence obtained indicates the viability of the method.

By a comparison with the formal definition in Section 3 it is not difficult to verify that  $\mathcal{R}_{\text{sat}}$  is compatible with  $\text{POP}^*$  (cf. Example 3.8). This implies that the number of rewrite steps starting from  $\text{sat}(cs)$  is polynomially bounded in the size of the CNF  $cs$ . This can be

<sup>2</sup> Solid lines indicate a characterisation, whereas dashed lines indicate an inclusion relationship.

automatically verified by  $\mathsf{TCT}$  in a fraction of a second. Due to a suitable adaption of the polynomial invariance theorem [9] (cf. Proposition 2.4) we can thus *automatically* conclude that FSAT belongs to FNP.

The termination order  $\text{POP}^*$  gives a syntactic account of the principle of *predicative recursion* as proposed by Bellantoni and Cook [18]. Conclusively any TRS compatible with  $\text{POP}^*$  is called *predicative recursive*. Analogously  $\text{POP}^*$  can be conceived as syntactic account of Leivant’s notion of *tiered recurrence* [40–42], cf. Simmons [53]. We think that  $\text{POP}^*$  is not only of interest from the viewpoint of automated runtime complexity, but also from the viewpoint of *implicit computational complexity* (*ICC* for short) [15, 38]. In particular  $\text{POP}^*$  is applicable to verify closure properties of the class of polytime computable function. Through our extension  $\text{POP}_{\text{PS}}^*$ , we reobtain Bellantoni’s result that predicative recursion is closed under parameter substitution (cf. Section 7).

Preliminary versions of the presented results appeared in [5, 7, 12]. The order  $\text{POP}^*$  has been introduced in [5], extended to quasi-precedences in [12] and the extension  $\text{POP}_{\text{PS}}^*$  appeared in [7]. Apart from the correction of some shortcomings, we extend our earlier work in the following way: First, the presented definition of  $\text{POP}^*$  is more liberal and captures the underlying idea of predicative recursion more precisely, compare [5, Definition 4] and Definition 3.5 from Section 3. Second, our soundness result (cf. Theorem 6.1 from Section 6) is new and more general than similar results presented earlier. In particular it does no longer require typing of constructors nor the intermediate step of completely defined TRSs, cf. [5]. Third, the propositional encoding used in our automation of polynomial path orders (cf. Section 8) has been completely overhauled.

**1.1. Related Work.** Polynomial complexity analysis is an active research area in rewriting. Starting from [46] interest in *automated* polynomial complexity analysis greatly increased over the last years, see for example [30, 31, 45, 48, 57]. This is partly due to the incorporation of a dedicated category for complexity into the annual termination competition (TERMCOMP).<sup>3</sup>

There are several accounts of predicative analysis of recursion in the (ICC) literature. We mention Marion’s *light multiset path order* (*LMPO* for short) [43]. The path order LMPO provides an order-theoretic characterisation of the class FP and can be also consider as a miniaturisation of MPO of sorts: it is a restriction of MPO and yields an order-theoretic characterisation of a complexity class. On the other hand LMPO cannot be used to characterise the (innermost) runtime complexity of TRSs. This follows from Example 1.2 below. In particular, although  $\mathcal{R}_{\text{sat}}$  is compatible with LMPO, from this we can only conclude that FSAT is computable on a nondeterministic Turing machine in polynomial time. However, this follows by design as FSAT is complete for FNP.

**Example 1.2.** The TRS  $\mathcal{R}_{\text{bin}}$  is given by the following rules:

$$25: \text{bin}(x, 0) \rightarrow \text{s}(0) \quad 26: \text{bin}(0, \text{s}(y)) \rightarrow 0 \quad 27: \text{bin}(\text{s}(x), \text{s}(y)) \rightarrow +(\text{bin}(x, \text{s}(y)), \text{bin}(x, y)) .$$

For a precedence  $\succ$  that fulfils  $\text{bin} \succ \text{s}$  and  $\text{bin} \succ +$  we obtain that  $\mathcal{R}_{\text{bin}}$  is compatible with LMPO. However it is straightforward to verify that the family of terms  $\text{bin}(\text{s}^n(0), \text{s}^m(0))$  admits (innermost) derivations whose length grows exponentially in  $n$ . Still the underlying function can be proven polynomial, essentially relying on memoisation techniques [43].

<sup>3</sup><http://termcomp.uibk.ac.at/>.

On the other hand Arai and the second author introduced the *polynomial path order for FP* ( $POP_{FP}$  for short). In a similar way as LMPO,  $POP_{FP}$  characterises the class FP and in addition induces innermost polynomial runtime complexity. However in comparison to  $POP^*$ ,  $POP_{FP}$  severely lacks applicability as it requires a specific signature of the given rewrite system. For example  $POP_{FP}$  is not directly applicable to  $\mathcal{R}_{\text{sat}}$ : only a special transformation of the rewrite system  $\mathcal{R}_{\text{sat}}$  can be handled.

Furthermore, a strengthening of our first main theorem to runtime complexity can be obtained if one considers polynomial interpretations, where the interpretations of constructor symbols is restricted. Such restricted polynomial interpretations are called *additive* in [21]. Note that additive polynomial interpretations also characterise the functions computable in polytime [21]. Similarly, *quasi-interpretations* [22] provide an elegant way to characterise time complexity classes through a combination of syntactic (via restrictions of reduction orders) and semantic (via quasi-interpretations) considerations. To date it is unknown whether quasi-interpretations can be used to assess polynomial runtime complexity of TRSs. Unarguable these semantic techniques admit a better intensionality than the syntactic characterisation provided through the path order  $POP^*$ . But semantic methods are notoriously difficult to implement efficiently in an automated setting. In particular we are only aware of one accessible implementation of quasi-interpretations, our own [12]. Note that these semantic methods are not tailored for innermost rewriting, in particular Example 2.5 given below cannot be handled, while it can be easily handled by  $POP^*$ .

Although we consider here only time complexity, related work indicates that the overall approach is general enough to reason also about space complexity. For instance, the *Knuth-Bendix* order [14] can be miniaturised to characterise linear space [20]. Likewise, *super-interpretations* [44] provide a semantic technique capable of characterising polynomial space.

In [16], Beckmann and Weiermann have given a term rewriting characterisation of the principle of predicative recursion proposed by Bellantoni and Cook. Following ideas proposed by Cichon and Weiermann in [24], Beckmann and Weiermann thus reobtain Bellantoni’s result that predicative recursion is closed under parameter recursion.

We have extended our complexity analysis tool  $\mathsf{TCT}$  [11] with polynomial path orders. We briefly contrast this implementation to related tools for the static resource analysis of programs. Hoffmann et al. [34] provide an automatic multivariate amortised cost analysis exploiting typing, which extends earlier results on amortised cost analysis [55]. To indicate the applicability of our method we have employed a straightforward (and complexity preserving) transformation of the RAML programs considered in [34, 35] into TRSs. Equipped with  $POP^*$  our complexity analyser  $\mathsf{TCT}$  can handle all examples from [34]. Albert et al. [1] present an automated complexity tool for Java<sup>TM</sup> Bytecode programs, Alias et al. [2] give a complexity and termination analysis for flowchart programs, and Gulwani et al. [29] as well as Zuleger et al. [59] provide an automated complexity tool for C programs. Very recently Hofmann and Rodriguez proposed in [36] an automated resource analysis for object-oriented programs via an amortised cost analysis.

**1.2. Outline.** The remainder of this paper is organised as follows. In the next section we recall basic notions and starting points of this paper. In Section 3 we introduce polynomial path orders. In the subsequent Sections 4 and 5 we show that the innermost runtime complexity of predicative recursive TRSs is polynomially bounded. As essential tool for this we introduce an extended version of  $POP_{FP}$ . In Section 6 we present our ramification

of polynomial path orders in ICC. Parameter substitution is incorporated in Section 7. Our implementation is detailed in Section 8 and experimental evidence is provided in Section 9. Finally, we conclude and present future work in Section 10.

## 2. PRELIMINARIES

We denote by  $\mathbb{N}$  the set of natural numbers  $\{0, 1, 2, \dots\}$ . Let  $R$  be a binary relation. The transitive closure of  $R$  is denoted by  $R^+$  and its transitive and reflexive closure by  $R^*$ . For a binary relation  $R$ , we frequently write  $a R b$  instead of  $(a, b) \in R$ . Composition of binary relations  $R$  and  $S$  is denoted by  $R \cdot S$ , and defined in the usual way. For  $n \in \mathbb{N}$  we denote by  $R^n$  the  $n$ -fold composition of  $R$ . The binary relation  $R$  is *well-founded* if there exists no infinite chain  $a_0, a_1, \dots$  with  $a_i R a_{i+1}$  for all  $i \in \mathbb{N}$ . Moreover, we say that  $R$  is well-founded on a set  $A$  if there exists no such infinite chain with  $a_0 \in A$ . The relation  $R$  is *finitely branching* if for all elements  $a$ , the set  $\{b \mid a R b\}$  is finite.

A *proper order* is an irreflexive and transitive binary relation. A *preorder* is a reflexive and transitive binary relation. An *equivalence relation* is reflexive, symmetric and transitive.

A multiset is a collection in which elements are allowed to occur more than once. We denote by  $\mathcal{M}(A)$  the set of multisets over  $A$  and write  $\{\{a_1, \dots, a_n\}\}$  to denote multisets with elements  $a_1, \dots, a_n$ . We use  $m_1 \uplus m_2$  for the summation and  $m_1 \setminus m_2$  for difference on multisets  $m_1$  and  $m_2$ . The *multiset extension*  $R^{\text{mul}}$  of a relation  $R$  on  $A$  is the relation on  $\mathcal{M}(A)$  such that  $M_1 R^{\text{mul}} M_2$  if there exists multisets  $X, Y \in \mathcal{M}(A)$  satisfying

- (1)  $M_2 = (M_1 \setminus X) \uplus Y$ ,
- (2)  $\emptyset \neq X \subseteq M_1$  and
- (3) for all  $y \in Y$  there exists an element  $x \in X$  such that  $x R y$ .

In order to extend this definition to preorders and equivalences, we follow [28]. Let  $\sim$  denote an equivalence relation over the set  $A$  and let  $\succ = \succ \cup \sim$  be a binary relation over  $A$  so that  $\succ$  and  $\sim$  are *compatible* in the following sense:  $\sim \cdot \succ \cdot \sim \subseteq \succ$ . Let  $[a]_{\sim}$  denote the *equivalence class* of  $a \in A$  with respect to  $\sim$ . By the compatibility requirement, the extension  $\sqsupset$  of  $\succ$  to equivalence classes such that  $[a]_{\sim} \sqsupset [b]_{\sim}$  if and only if  $a \succ b$ , is well defined. We define the *strict multiset extension*  $\succ^{\text{mul}}$  of  $\succ$  as  $M_1 \succ^{\text{mul}} M_2$  if and only if  $[M_1]_{\sim} \sqsupset^{\text{mul}} [M_2]_{\sim}$ . Further, the *weak multiset extension*  $\succsim^{\text{mul}}$  of  $\succ$  is given by  $M_1 \succsim^{\text{mul}} M_2$  if and only if  $[M_1]_{\sim} \sqsupset^{\text{mul}} [M_2]_{\sim}$  or  $[M_1]_{\sim} = [M_2]_{\sim}$  holds. Note that if  $\succ$  is a preorder (on  $A$ ) then  $\succ^{\text{mul}}$  is a proper order and  $\succsim^{\text{mul}}$  a preorder on  $\mathcal{M}(A)$ , cf. [28]. Also  $\succ^{\text{mul}}$  is well-founded if  $\succ$  is well-founded.

**2.1. Complexity Theory.** Notations are taken from [49]. The *function problem*  $F_R$  associated with a binary relation  $R$  is defined as follows: given  $x$  find some  $y$  such that  $(x, y) \in R$  holds if  $y$  exists, otherwise return no. A binary relation  $R$  on words is called *polynomially balanced* if for all  $(x, y) \in R$ , the size of  $y$  is polynomially bounded in the size of  $x$ . The relation  $R$  is *polytime decidable* if  $(x, y) \in R$  is decided by a deterministic *Turing machine* (TM for short)  $M$  operating in polynomial time. The class NP is the class of languages  $L$  admitting polynomially balanced, polytime decidable relations  $R_L$  [49, Chapter 9]:  $L = \{x \mid (x, y) \in R_L \text{ for some } y\}$ . The class FNP is the class of function problems associated with the polynomially balanced and polytime decidable relations  $R_L$  as above. The class of *polytime computable functions* FP is the subclass resulting if we only consider function problems in FNP that can be solved in polynomial time [49, Chapter 10].

Recall that a function problem  $F$  *reduces* to a function problem  $G$  if there exist functions  $s$  and  $r$ , both computable in logarithmic space, such that for all  $x, y$  with  $F$  computing  $y$  on input  $x$ ,  $G$  computes on input  $s(x)$  the output  $z$  with  $r(z) = y$ . Note that both FNP and FP are closed under reductions. We note that nondeterministic Turing machines running in polynomial time compute function problems from FNP.

**Proposition 2.1.** Let  $N$  be a nondeterministic Turing machine that computes the function problem  $F$  in polynomial time. Then  $F \in \text{FNP}$ .

*Proof.* Define the following relation  $R$ :  $(x, y) \in R$  if and only if  $y$  is the encoding of an accepting computation of  $N$  on input  $x$ . For this encoding it is sufficient to encode a successful sequence of configurations. Since  $N$  operates in polynomial time, the length of any computation, and also the size of each configuration, is polynomially bounded. It follows that  $R$  is polynomially balanced. As it can be checked in linear time that  $y$  encodes an accepting run of  $N$  on input  $x$ ,  $R$  is polytime decidable. Hence the function problem  $F_R$  that computes an accepting run  $y$  of  $N$  on input  $x$  is in FNP. Finally notice that  $F$  reduces to  $F_R$ . To see this, employ following reduction: the function  $s$  is simply the identity function; the logspace computable function  $r$  extracts the result of  $N$  on input  $x$  from the accepting run  $y$  computed by  $F_R$  on input  $x$ . We conclude the lemma since FNP is closed under reductions.  $\square$

**2.2. Term Rewriting.** We assume at least nodding acquaintance with the basics of term rewriting [14]. We fix notions and notation that are used in the paper.

Throughout the paper, we fix a countably infinite set of *variables*  $\mathcal{V}$  and a finite *signature*  $\mathcal{F}$  of *function symbols*. The signature  $\mathcal{F}$  is partitioned into *defined symbols*  $\mathcal{D}$  and *constructors*  $\mathcal{C}$ . The set of *values*, *basic terms* and *terms* is defined according to the grammar

$$\begin{aligned} \text{(Values)} \quad & \mathcal{T}(\mathcal{C}, \mathcal{V}) \ni v := x \mid c(v_1, \dots, v_n) \\ \text{(Basic Terms)} \quad & \mathcal{T}_b(\mathcal{F}, \mathcal{V}) \ni s := x \mid f(v_1, \dots, v_n) \\ \text{(Terms)} \quad & \mathcal{T}(\mathcal{F}, \mathcal{V}) \ni t := x \mid c(t_1, \dots, t_n) \mid f(t_1, \dots, t_n). \end{aligned}$$

where  $x \in \mathcal{V}$ ,  $c \in \mathcal{C}$ , and  $f \in \mathcal{D}$ .

The *arity* of a function symbol  $f \in \mathcal{F}$  is denoted by  $\text{ar}(f)$ . We write  $s \triangleright t$  if  $t$  is a *subterm* of  $s$ , the strict part of  $\triangleright$  is denoted by  $\triangleright$ . The *size* of a term  $t$  is denoted by  $|t|$  and refers to the number of occurrences of variables and function symbols contained in  $t$ . We denote by  $\text{dp}(t)$  the *depth* of  $t$  which is defined as  $\text{dp}(t) = 1$  if  $t \in \mathcal{V}$  and  $\text{dp}(f(t_1, \dots, t_n)) = 1 + \max\{0\} \cup \{\text{dp}(t_i) \mid i = 1, \dots, n\}$ .

Let  $\succsim$  be a preorder on the signature  $\mathcal{F}$ , called *quasi-precedence* or simply *precedence*. We always write  $\succ$  for the induced proper order and  $\sim$  for the induced equivalence on  $\mathcal{F}$ . We lift the equivalence  $\sim$  to terms modulo argument permutation:  $s \sim t$  if either  $s = t$  or  $s = f(s_1, \dots, s_n)$  and  $t = g(t_1, \dots, t_n)$  where  $f \sim g$  and for some permutation  $\pi$ ,  $s_i \sim t_{\pi(i)}$  for all  $i \in \{1, \dots, n\}$ . Further we write  $s \triangleright_{\sim} t$  if  $t$  is a subterm of  $s$  modulo  $\sim$ , i.e.,  $s \triangleright \cdot \sim t$ . We denote by  $\mathcal{F}^{\prec f} := \{g \mid f \succ g\}$  the set of function symbols below  $f$  in the precedence  $\succsim$ . The *rank* of a function symbol is inductively defined by  $\text{rk}(f) = \max\{0\} \cup \{1 + \text{rk}(g) \mid f \succ g\}$ .

A *rewrite rule* is a pair  $(l, r)$  of terms, in notation  $l \rightarrow r$ , such that  $l$  is not a variable and all variables in  $r$  occur also in  $l$ . Here  $l$  is called the *left-hand*, and  $r$  the *right-hand side* of  $l \rightarrow r$ . A *term rewrite system* (TRS for short)  $\mathcal{R}$  over  $\mathcal{T}(\mathcal{F}, \mathcal{V})$  is a set of *rewrite rules*. In the following,  $\mathcal{R}$  always denotes a TRS. If not mentioned otherwise, we assume  $\mathcal{R}$  to be

*finite*. A relation on  $\mathcal{T}(\mathcal{F}, \mathcal{V})$  is a *rewrite relation* if it is closed under contexts and closed under substitutions. The smallest rewrite relation that contains  $\mathcal{R}$  is denoted by  $\rightarrow_{\mathcal{R}}$ .

A term  $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  is called a *normal form* if there is no  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  such that  $s \rightarrow_{\mathcal{R}} t$ . With  $\text{NF}(\mathcal{R})$  we denote the set of all normal forms of a TRS  $\mathcal{R}$ ; if the TRS is clear from context we simply write  $\text{NF}$ . Whenever  $t$  is a normal form of  $\mathcal{R}$  we write  $s \rightarrow_{\mathcal{R}}^! t$  for  $s \rightarrow_{\mathcal{R}}^* t$ . The *innermost rewrite relation*, denoted as  $\dot{\rightarrow}_{\mathcal{R}}$ , is the restriction of  $\rightarrow_{\mathcal{R}}$  where the arguments of the redex are in normal form. The TRS  $\mathcal{R}$  is *terminating* if no infinite rewrite sequence exists. The TRS  $\mathcal{R}$  has *unique normal forms* if for all  $s, t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  with  $s \rightarrow_{\mathcal{R}}^! t_1$  and  $s \rightarrow_{\mathcal{R}}^! t_2$  we have  $t_1 = t_2$ . The TRS  $\mathcal{R}$  is called *confluent* if for all  $s, t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  with  $s \rightarrow_{\mathcal{R}}^* t_1$  and  $s \rightarrow_{\mathcal{R}}^* t_2$  there exists a term  $u$  such that  $t_1 \rightarrow_{\mathcal{R}}^* u$  and  $t_2 \rightarrow_{\mathcal{R}}^* u$ . An *orthogonal* TRS is a *left-linear* and *non-overlapping* TRS. Here *left-linear* means that no variable occurs more than once in each left-hand side. A TRS is *overlapping*, if some pair of rules  $l_1 \rightarrow r_1$  and  $l_2 \rightarrow r_2$  in  $\mathcal{R}$ , renamed so that variables are disjoint, satisfies: (i) a subterm  $l'_1$  of  $l_1$  is unifiable with  $l_2$ , i.e.,  $l'_1\sigma = l_2\sigma$  for some substitution  $\sigma$ ; and (ii) if  $l'_1 = l_1$  then the rules  $l_1 \rightarrow r_1$  and  $l_2 \rightarrow r_2$  are not equal up to renaming of variables. Note that every orthogonal TRS is confluent [14]. A TRS  $\mathcal{R}$  is a *constructor* TRS if all left-hand sides are basic terms.

**2.3. Rewriting as Computational Model.** We fix *call-by-value* semantics and only consider *constructor* TRSs  $\mathcal{R}$ . Input and output are taken from the set of values  $\mathcal{T}(\mathcal{C}, \mathcal{V})$ , and defined symbols  $f \in \mathcal{D}$  denote computed functions. More precise, a (finite) *computation* of  $f \in \mathcal{D}$  on input  $v_1, \dots, v_n \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  is given by *innermost* reductions

$$f(v_1, \dots, v_n) = t_0 \dot{\rightarrow}_{\mathcal{R}} t_1 \dot{\rightarrow}_{\mathcal{R}} \dots \dot{\rightarrow}_{\mathcal{R}} t_\ell = w .$$

If the above computation ends in a value, i.e.,  $w \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ , we also say that  $f$  *computes* on input  $v_1, \dots, v_n$  in  $\ell$  steps the value  $w$ .

To account for nondeterministic computation, we capture the semantics of  $\mathcal{R}$  by assigning to each  $n$ -ary defined symbol  $f \in \mathcal{D}$  an  $n + 1$ -ary relation  $\llbracket f \rrbracket$  that relates input arguments  $v_1, \dots, v_n$  to computed values  $w$ . A *finite* set  $\mathcal{N}$  of *non-accepting patterns* is used to distinguish meaningful outputs  $w$  from outputs that should not be considered part of the computation [20]. A value  $w$  is *accepting* with respect to  $\mathcal{N}$  if no  $p \in \mathcal{N}$  and no substitution  $\sigma$  exists, such that  $p\sigma = w$  holds. A typical example of a value that should not be accepted is the constant *unsat* appearing in the TRS  $\mathcal{R}_{\text{sat}}$  from Example 1.1. Below function problem are extended to  $n + 1$ -ary relations in the obvious way.

**Definition 2.2.** Let  $\mathcal{R}$  be a TRS and let  $\mathcal{N}$  be a set of non-accepting patterns. For each  $n$ -ary symbol  $f \in \mathcal{D}$  the *relation*  $\llbracket f \rrbracket \subseteq \mathcal{T}(\mathcal{C}, \mathcal{V})^{n+1}$  defined by  $f$  in  $\mathcal{R}$  is given by

$$(v_1, \dots, v_n, w) \in \llbracket f \rrbracket \quad :\Leftrightarrow \quad f(v_1, \dots, v_n) \dot{\rightarrow}_{\mathcal{R}}^! w \text{ and } w \text{ is accepting with respect to } \mathcal{N} .$$

We say that  $\mathcal{R}$  *computes* the function problems associated with  $\llbracket f \rrbracket$ .

Note that if  $\mathcal{R}$  is confluent, then  $\llbracket f \rrbracket$  is in fact a (partial) function. Following [9, 30] we adopt an *unitary cost model* for rewriting, where each reduction step accounts for one time unit, cf. [25, 26]. Reductions are of course measured in the size of the input.

**Definition 2.3.** The *innermost runtime complexity function*  $\text{rc}_{\mathcal{R}} : \mathbb{N} \rightarrow \mathbb{N}$  relates sizes of basic terms  $f(v_1, \dots, v_n) \in \mathcal{T}_{\text{b}}(\mathcal{F}, \mathcal{V})$  to the maximal length of computation. Formally

$$\text{rc}_{\mathcal{R}}(n) := \max\{\ell \mid \exists s \in \mathcal{T}_{\text{b}}(\mathcal{F}, \mathcal{V}), |s| \leq n \text{ and } s = t_0 \dot{\rightarrow}_{\mathcal{R}} t_1 \dot{\rightarrow}_{\mathcal{R}} \dots \dot{\rightarrow}_{\mathcal{R}} t_\ell\} .$$



If clear from context, we sometime drop the qualifier 'innermost' and simply speak of *runtime complexity* (of the TRS  $\mathcal{R}$ ).

The restriction  $s \in \mathcal{T}_b(\mathcal{F}, \mathcal{V})$  accounts for the fact that computations start only from basic terms. We sometimes use  $\text{dh}_{\mathcal{R}}(s) := \max\{\ell \mid \exists t. s \xrightarrow{\ell}_{\mathcal{R}} t\}$  to refer to the (*innermost*) *derivation height* of a single term  $s$ . Note that the runtime complexity function is well-defined if  $\mathcal{R}$  is *terminating*, i.e.,  $\xrightarrow{\cdot}_{\mathcal{R}}$  is well-founded. Suppose  $\text{rc}_{\mathcal{R}}$  is asymptotically bounded from above by a linear, quadratic, . . . , polynomial function, we simply say that the runtime of  $\mathcal{R}$  is linear, quadratic, . . . , or respectively polynomial. If no confusion can arise, we drop the reference to the TRS  $\mathcal{R}$  and simply write  $\text{rc}$  instead of  $\text{rc}_{\mathcal{R}}$ .

By folklore it is known that rewriting can be implemented with only polynomial overhead if terms grow only polynomial during reductions. This implies that the functions computed by specific TRSs of polynomial runtime complexity can be implemented with polynomial time complexity on a Turing machine (or an alternative computation model). This observation can be significantly extended as it can be shown that the restriction on polynomial growth is not necessary.

In [8, 9] it is shown that the unitary cost model is reasonable for full rewriting. The deterministic case was established independently in [25, 26] using essentially the same approach. By Proposition 2.1 and a suitable adaption of Theorem 6.2 in [9] to innermost rewriting we obtain the following proposition.

**Proposition 2.4.** Let  $\mathcal{R}$  be a rewrite system with polynomial runtime. Then the function problems associated with  $\llbracket f \rrbracket$  defined by  $\mathcal{R}$  are contained in FNP. If  $\mathcal{R}$  is confluent, i.e., deterministic, then  $\llbracket f \rrbracket$  is a (partial) function contained in FP.

Our choice of adopting call-by-value semantics rests on the observation that the unitary cost model of unrestricted rewriting often overestimates the runtime complexity of computed functions. This has to do with the unnecessary duplication of redexes.

**Example 2.5.** Consider the constructor TRS  $\mathcal{R}_{\text{btree}}$  given by the following rules:

$$28: \text{btree}(0) \rightarrow \text{leaf} \quad 29: \text{dup}(t) \rightarrow \text{node}(t, t) \quad 30: \text{btree}(s(n)) \rightarrow \text{dup}(\text{btree}(n)) .$$

Then for  $n \in \mathbb{N}$ ,  $\text{btree}(s^n(0))$  computes a binary tree of height  $n$  in a linear number of steps. On the other hand,  $\mathcal{R}_{\text{btree}}$  gives also rise to a non-innermost reduction

$$\text{btree}(s^n(0)) \rightarrow_{\mathcal{R}} \text{dup}(\text{btree}(s^{n-1}(0))) \rightarrow_{\mathcal{R}} \text{node}(\text{btree}(s^{n-1}(0)), \text{btree}(s^{n-1}(0))) \rightarrow_{\mathcal{R}} \dots ,$$

obtained by preferring  $\text{dup}$  over  $\text{btree}$ . The length of the derivation is however exponential in  $n$ .

By Proposition 2.4 we obtain  $\llbracket \text{btree} \rrbracket \in \text{FP}$ . As indicated later, our analysis can automatically classify the function  $\llbracket \text{btree} \rrbracket$  as feasible.

### 3. THE POLYNOMIAL PATH ORDER

We arrive at the formal definition of the *polynomial path order* ( $\text{POP}^*$  for short). Variants of the definition presented here have been presented in earlier conference publications [5–7, 12].

As already mentioned in the introduction the multiset path order capture the primitive recursive functions. Hence reduction orders can entail implicit characterisations of complexity classes. This motivates the quest for a miniaturisation of MPO that precisely captures the class FP. Another motivation for the design of  $\text{POP}^*$  rests on the observation

that term-rewriting characterisations of complexity classes may facilitate the study of (low) complexity classes, cf. [16, 24]. Such applications imply the need to craft an order that induces polynomial innermost runtime complexity. POP\* meets these demands, by providing a syntactic account of the *predicative analysis* of recursion set forth by Bellantoni and Cook [18]. Analogously POP\* can be conceived as syntactic account of Leivant's notion of *tiered recurrence* [40, 42].

For each function  $f$ , the arguments to  $f$  are separated into *normal* and *safe* ones. To highlight this separation, we write  $f(\vec{x}; \vec{y})$  where arguments  $\vec{x}$  to the left of the semicolon are normal, the arguments  $\vec{y}$  to the right are safe. Bellantoni and Cook define a class  $\mathcal{B}$ , consisting of a small set of initial functions and that is closed under *safe composition* and *safe recursion on notation* (*safe recursion* for brevity).

The crucial ingredient in  $\mathcal{B}$  is that a new function  $f$  is defined via safe recursion by the equations:

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(2z + i, \vec{x}; \vec{y}) &= h_i(z, \vec{x}; \vec{y}, f(z, \vec{x}; \vec{y})) \quad i \in \{1, 2\}, \end{aligned} \tag{SRN}$$

for functions  $g, h_1$  and  $h_2$  already defined in  $\mathcal{B}$ . This definition corresponds to a recursion on the binary representation of numbers. Consequently the number of recursive calls is linear in the size of the recursive input. Unlike primitive recursive functions, the stepping functions  $h_i$  cannot perform recursion on the *impredicative* value  $f(z, \vec{x}; \vec{y})$ . This is a consequence of *data tiering*: recursion is performed on *normal* arguments only. Dual, the recursive call  $f(z, \vec{x}; \vec{y})$  is substituted into a *safe* argument position.

To maintain the separation, safe composition restricts the usual composition operator so that safe arguments are not substituted into normal argument position. Precisely, for functions  $h, \vec{r}$  and  $\vec{s}$  already defined in  $\mathcal{B}$ , a function  $f$  is defined by safe composition using the equation

$$f(\vec{x}; \vec{y}) = h(\vec{r}(\vec{x}); \vec{s}(\vec{x}; \vec{y})) . \tag{SC}$$

Crucially, the safe arguments  $\vec{y}$  are absent in normal arguments to  $h$ . The main result from [18] states that the class  $\mathcal{B}$  coincides with the class of polytime computable functions FP.

As a first step to capture the notion of predicative recursion in POP\*, we introduce the concept of *safe mappings* to handle the separation of argument positions.

**Definition 3.1.** A *safe mapping*  $\mathit{safe}$  is a function  $\mathit{safe} : \mathcal{F} \rightarrow 2^{\mathbb{N}}$  that associates with every  $n$ -ary function symbol  $f$  the set of *safe argument positions*  $\{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}$ . Argument positions included in  $\mathit{safe}(f)$  are called *safe*, those not included are called *normal* and collected in  $\mathit{norm}(f)$ . For  $n$ -ary constructors  $c$  we require that all argument positions are safe, i.e.,  $\mathit{safe}(c) = \{1, \dots, n\}$ .

We refine term equivalence so that the safe mapping is taken into account.

**Definition 3.2.** Let  $\succcurlyeq$  denote a precedence and  $\mathit{safe}$  a safe mapping. We define *safe equivalence*  $\simeq$  for terms  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  inductively as follows:  $s \simeq t$  if either  $s = t$  or  $s = f(s_1, \dots, s_n)$ ,  $t = g(t_1, \dots, t_n)$ ,  $f \sim g$  and there exists a permutation  $\pi$  such that for all  $i \in \{1, \dots, n\}$ ,  $s_i \simeq t_{\pi(i)}$  and  $i \in \mathit{safe}(f)$  if and only if  $\pi(i) \in \mathit{safe}(g)$ .

To avoid notational overhead, we fix a safe mapping  $\mathit{safe}$  and suppose that for each  $k + l$  ary function symbol  $f$ , the first  $k$  argument positions are normal, and the remaining argument positions are safe, i.e.,  $\mathit{safe}(f) = \{k + 1, \dots, k + l\}$ . This allows use to write

$f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$  as before. We require that the precedence  $\succ$  adheres the partitioning of  $\mathcal{F}$  into defined symbols and constructors in the following sense.

**Definition 3.3.** A precedence  $\succ$  is *admissible* (for POP\*) if  $f \sim g$  implies that either both  $f$  and  $g$  are defined symbols, or both are constructors.

In particular  $\tilde{\succ}$  preserves values, i.e., if  $s \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  and  $s \tilde{\succ} t$  then also  $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ . The following definition introduces an auxiliary order  $>_{\text{pop}}$ , the full order  $>_{\text{pop}^*}$  is then presented in Definition 3.5.

**Definition 3.4.** Let  $\succ$  denote a precedence. Consider terms  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  such that  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$ . Then  $s >_{\text{pop}} t$  if one of the following alternatives holds:

- (1)  $s_i \geq_{\text{pop}} t$  for some  $i \in \{1, \dots, k+l\}$  and, if  $f \in \mathcal{D}$  then  $i$  is a normal argument position ( $i \in \{1, \dots, k\}$ );
- (2)  $f \in \mathcal{D}$ ,  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \succ g$  and  $s >_{\text{pop}} t_i$  for all  $i = 1, \dots, m+n$ .

Here we set  $\geq_{\text{pop}} := >_{\text{pop}} \cup \tilde{\succ}$ .

Consider a function  $f$  defined by safe composition from  $r$  and  $s$ , cf. scheme (SC). The effect of this auxiliary order is to (properly) encompass safe composition in the full order  $>_{\text{pop}^*}$ . Note that the auxiliary order can orient  $f(\vec{x}; \vec{y}) >_{\text{pop}} r(\vec{x};)$  for defined symbols  $f$  and  $r$  with  $f \succ r$ . On the other hand,  $f(\vec{x}; \vec{y})$  and safe arguments  $y_i$  are incomparable, and thus the orientation of  $f(\vec{x}; \vec{y})$  and  $s(\vec{x}; \vec{y})$  fails, even if  $f \succ s$  is supposed.

**Definition 3.5.** Let  $\succ$  denote a precedence. Consider terms  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  such that  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$ . Then  $s >_{\text{pop}^*} t$  if one of the following alternatives holds:

- (1)  $s_i \geq_{\text{pop}^*} t$  for some  $i \in \{1, \dots, k+l\}$ , or
- (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 conditions hold:
  - $s >_{\text{pop}} t_j$  for all normal argument positions  $j = 1, \dots, m$ ;
  - $s >_{\text{pop}^*} t_j$  for all safe argument positions  $j = m+1, \dots, m+n$ ;
  - $t_j \notin \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$  for at most one safe argument position  $j \in \{m+1, \dots, m+n\}$ ;
- (3)  $f \in \mathcal{D}$ ,  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \sim g$  and the following conditions hold:
  - $\{\{s_1, \dots, s_k\}\} >_{\text{pop}^*}^{\text{mul}} \{\{t_1, \dots, t_m\}\}$ ;
  - $\{\{s_{k+1}, \dots, s_{k+l}\}\} \geq_{\text{pop}^*}^{\text{mul}} \{\{t_{m+1}, \dots, t_{m+n}\}\}$ .

Here  $\geq_{\text{pop}^*} := >_{\text{pop}^*} \cup \tilde{\succ}$ .

We use the notation  $>_{\text{pop}}^{(i)}$  and respectively  $>_{\text{pop}^*}^{(i)}$  to refer to the  $i^{\text{th}}$  case in Definition 3.4 respectively Definition 3.5. Note that POP\* is not a reduction order, as for example closure under contexts fails due to the conditions put upon  $>_{\text{pop}^*}^{(3)}$ . However POP\* is a restriction of MPO and thus a *termination order*.

**Remark.** The restrictions put upon  $>_{\text{pop}^*}^{(2)}$  amount to the fact that POP\* allows at most one recursive call per right-hand side.

**Remark.** The proposed constraints are weaker compared to the corresponding clause given in [5, Definition 4]. The early definition from [5, Definition 4], used the full order  $>_{\text{pop}^*}$  only on one argument of the right-hand side (the one that possibly holds the recursive call), the remaining arguments were all oriented with the auxiliary order  $>_{\text{pop}}$ .

The case  $>_{\text{pop}^*}^{(2)}$  accounts for definitions by safe composition (SC). The final restriction put onto  $>_{\text{pop}^*}^{(2)}$  is used to prevent multiple recursive calls as indicated in Example 1.2. The

case  $\succ_{\text{pop}^*}^{(3)}$  restricts the corresponding case in MPO by taking the separation of normal and safe argument positions into account. Note that here normal arguments need to decrease. This reflects that as in (SRN) recursion is performed on normal argument positions. We emphasise that one application of  $\succ_{\text{pop}^*}^{(2)}$  possibly followed by one application of  $\succ_{\text{pop}^*}^{(3)}$  orients the defining equations in (SRN), using the obvious precedence (using suitable term representation of natural numbers).

Our order-theoretic account of predicative recursion motivates following definition.

**Definition 3.6.** We call a constructor TRS  $\mathcal{R}$  *predicative recursive*, if  $\mathcal{R}$  is compatible with an instance  $\succ_{\text{pop}^*}$  of  $\text{POP}^*$  based on an admissible precedence.

Note that it can be determined in nondeterministic polynomial time that a constructor TRS is predicative recursive: simply guess a safe mapping and a precedence and apply the definition of  $\text{POP}^*$ . This is in contrast to semantic method (like additive polynomial interpretations) whose synthesis is undecidable in general.

We clarify Definition 3.5 on several examples. The first of these examples shows a typical application of predicative recursion.

**Example 3.7.** Consider the TRS  $\mathcal{R}_{\text{mul}}$  expressing multiplication in Peano arithmetic.

$$\begin{array}{ll} 31: & +(0; y) \rightarrow y \quad 32: & +(\text{s}(\cdot; x); y) \rightarrow \text{s}(\cdot; +(x; y)) \\ 33: & \times(0, y; \cdot) \rightarrow 0 \quad 34: & \times(\text{s}(\cdot; x), y; \cdot) \rightarrow +(y; \times(x, y; \cdot)) \end{array}$$

The TRS  $\mathcal{R}_{\text{mul}}$  is predicative recursive, using the precedence  $\times \succ + \succ \text{s}$  and the safe mapping as indicated in the rules: The rules (31) and (33) are oriented by  $\succ_{\text{pop}^*}^{(1)}$ . The rule (32) is oriented by  $\succ_{\text{pop}^*}^{(2)}$  using  $+ \succ \text{s}$  and  $+(\text{s}(\cdot; x); y) \succ_{\text{pop}^*}^{(3)} +(x; y)$ . Note that the latter inequality only holds as the first argument position of addition is normal. Similar, the final rule (34) is oriented by  $\succ_{\text{pop}^*}^{(2)}$ , employing  $\times \succ +$  together with  $\times(\text{s}(\cdot; x), y; \cdot) \succ_{\text{pop}^*}^{(1)} y$  and  $\times(\text{s}(\cdot; x), y; \cdot) \succ_{\text{pop}^*}^{(3)} \times(x, y; \cdot)$ . Note that the latter two inequalities require that the both argument positions of  $\times$  are normal, i.e., are used for recursion.

We re-consider the motivation Example 1.1 from the introduction.

**Example 3.8** (Example 1.1 continued). Consider the TRS  $\mathcal{R}_{\text{sat}}$  from the motivating Example 1.1, where the separation into normal and safe arguments for the defined function symbols is defined as follows:  $\text{nrm}(\text{if}) = \text{nrm}(\text{negate}) = \emptyset$ ,  $\text{nrm}(\text{eq}) = \{1\}$ . We set  $\text{nrm}(\text{elem}) = \{2\}$ , and for all remaining defined function symbols, we make all arguments normal, i.e.,  $\text{nrm}(\text{consistent}) = \text{nrm}(\text{sat}) = \text{nrm}(\text{sat}') = \text{nrm}(\text{guess}) = \text{nrm}(\text{choice}) = \{1\}$ .

Then  $\mathcal{R}_{\text{sat}} \subseteq \succ_{\text{pop}^*}$  for any admissible precedence satisfying the following constraints:  $\text{guess} \succ \text{choice}$ ,  $\text{consistent} \succ \text{if}$ ,  $\text{elem}$ ,  $\text{negate}$ ,  $\text{sat} \succ \text{sat}'$ ,  $\text{guess}$ , and  $\text{sat}' \succ \text{if}$ ,  $\text{consistent}$ ,  $\text{unsat}$ .

The next example is negative, in the sense that the considered TRSs admits polynomial runtime complexity, but fails to be compatible with  $\text{POP}^*$ .

**Example 3.9** (Example 3.7 continued). Consider the TRS  $\mathcal{R}_{\text{mul}}$  where the rule (34) is replaced by the rule

$$34a: \quad \times(\text{s}(\cdot; x), y; \cdot) \rightarrow +(\times(x, y; \cdot); y).$$

The resulting system has polynomial runtime complexity, which can be automatically verified with  $\mathbf{TCT}$ . However, the TRS does not follow the rigid scheme of predicative recursion. For this reason, it cannot be handled by  $\text{POP}^*$ . Technically, the terms  $\times(\text{s}(\cdot; x), y; \cdot)$  and  $\times(x, y; \cdot)$  are incomparable with respect to  $\succ_{\text{pop}}$  independent on the precedence, and consequently also orientation of left- and right-hand side with  $\succ_{\text{pop}^*}^{(2)}$  fails.

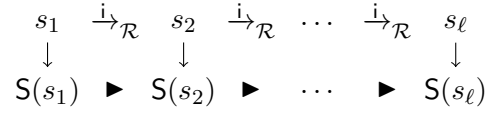


Figure 2: Predicative Embedding

The following examples clarifies the need for data tiering.

**Example 3.10** (Example 3.7 continued). Consider the extension of  $\mathcal{R}_{\text{mul}}$  by the two rules

$$35: \exp(0, y) \rightarrow s(; 0) \qquad 36: \exp(s(; x), y) \rightarrow \times(y, \exp(x, y);),$$

that express exponentiation  $y^x$  in an exponential number of steps. The definition of  $\exp$  disregards data tiering as imposed by predicative recursion. In particular, since  $\times$  admits no safe argument positions it cannot serve as a stepping function. Independent on the safe mapping for  $\exp$ , rule (36) cannot be oriented using polynomial path orders.

The following theorem constitutes the first main result of this paper.

**Theorem 3.11.** Let  $\mathcal{R}$  be a predicative recursive (constructor) TRS. Then the innermost derivation height of any basic term  $f(\vec{u}; \vec{v})$  is bounded by a polynomial in the maximal depth of normal arguments  $\vec{u}$ . The polynomial depends only on  $\mathcal{R}$  and the signature  $\mathcal{F}$ . In particular, the innermost runtime complexity of  $\mathcal{R}$  is polynomial.

The proof of Theorem 3.11 is involved and requires a variety of ingredients. We give a short outline. In Section 4, we define *predicative interpretations*  $S$  that flatten terms to *sequences of terms*, essentially separating safe from normal arguments. This allows us to analyse terms independent from safe arguments. Then we introduce an order  $\blacktriangleright$  on sequences of terms, that is simpler compared to  $>_{\text{pop}^*}$  and does not rely on the separation of argument positions. This *polynomial path order on sequences* extends the polynomial path order for FP, introduced in [4]. In Section 5 we establish a *predicative embedding* of derivations into  $\blacktriangleright$  as depicted in Figure 2.

In Theorem 4.17 we show that the length of  $\blacktriangleright$  descending sequences starting from basic terms can be bound appropriately. One may wonder whether a precise degree on the provided polynomial bound can be obtained by reasoning based on the depth of recursion and formation of composition rules. This is not the case as Lemma 3.12 clarifies, where we provide a family of TRS  $(\mathcal{R}_k)_{k \geq 1}$ , all with depth of recursion one and without the use of composition, such that  $\text{rc}_{\mathcal{R}_k}(n) = \Omega(n^k)$ . However, it is possible to design a restriction of POP\*, dubbed *small polynomial path order* in [13], which induces a precise degree on the provided polynomial bound.

**Lemma 3.12.** For every  $k \geq 1$  there exists a TRS  $\mathcal{R}_k$  over constructors  $s, 0$  defining a single defined symbol  $f_k$  such that  $\mathcal{R}_k \subseteq >_{\text{pop}^*}$  and  $\text{rc}_{\mathcal{R}_k}(n) = \Omega(n^k)$ .

*Proof.* Consider the following TRS  $\mathcal{R}_k$  that is compatible with  $>_{\text{pop}^*}$ :

$$\begin{aligned}
f_k(s(x_1), x_2, x_3, \dots, x_k;) &\rightarrow f_k(x_1, x_2, x_3, \dots, x_k;) \\
f_k(0, s(x_2), x_3, \dots, x_k;) &\rightarrow f_k(x_2, x_2, x_3, \dots, x_k;) \\
&\vdots \\
f_k(0, \dots, 0, s(x_k);) &\rightarrow f_k(x_k, \dots, x_k, x_k;)
\end{aligned}$$

We show the stronger claim that for all  $n \geq 1$  there exist constants  $c_k \in \mathbb{Q}$  such that

$$f_k(s^n(0), \dots, s^n(0)) \rightarrow_{\mathcal{R}_k}^{\geq c_k \cdot n^k} f_k(0, \dots, 0)$$

by induction on  $k$ . The base case is trivial, we consider the inductive step. Applying the induction hypothesis, by definition of  $\mathcal{R}_k$  and  $\mathcal{R}_{k+1}$  it is easy to see that for all  $n \geq 1$

$$\begin{aligned} f_{k+1}(s^n(0), \dots, s^n(0), s^n(0)) &\rightarrow_{\mathcal{R}_{k+1}}^{\geq c_k \cdot n^k} f_{k+1}(0, \dots, 0, s^n(0)) \\ &\rightarrow_{\mathcal{R}_{k+1}} f_{k+1}(s^{n-1}(0), \dots, s^{n-1}(0), s^{n-1}(0)) . \end{aligned}$$

Using this observation, a simple side induction on  $n$  reveals

$$f_{k+1}(s^n(0), \dots, s^n(0), s^n(0)) \rightarrow_{\mathcal{R}_{k+1}}^{\geq f(n)} f_{k+1}(0, \dots, 0, 0)$$

where  $f(n) = \sum_{i=1}^n c_k \cdot i^k \in \Omega(n^{k+1})$ . The exact overhead due to multiset comparisons is further investigated in Example 4.13.  $\square$

The next three examples stress that the restrictions to runtime complexity (Example 3.13), innermost reductions (Example 3.14), as well as constructor TRSs (Example 3.15), are all essential for the correctness of Theorem 3.11.

**Example 3.13.** Consider the TRS  $\mathcal{R}_{dc}$  given by the following rules, cf. [33, Example 1].

$$\begin{array}{ll} 37: +(0; y) \rightarrow y & 38: +(s(; x); y) \rightarrow s(+ (x; y)) \\ 39: d(0; ) \rightarrow 0 & 40: d(s(; x); ) \rightarrow s(s(; d(x; ))) \\ 41: q(0; ) \rightarrow 0 & 42: q(s(; x); ) \rightarrow +(s(; d(x; )); q(x; )) , \end{array}$$

where we suppose that  $0$  and  $s$  are the only constructors. As shown by Hofbauer and Lautemann,  $\mathcal{R}_{dc}$  admits at least double-exponentially *derivational complexity*; in particular it is easy to find a family of terms  $t_n$  such that  $\text{dh}(t_n) = 2^{2^{\Omega(n)}}$ .

On the other hand  $\mathcal{R}_{dc}$  is compatible with a polynomial path order  $>_{\text{pop}^*}$  as induced by a precedence  $\succ$  satisfying  $q \succ d \succ s \succ 0$  and the safe mapping as indicated in the rules. As a side-remark we emphasise that the orientability of rule (42) induces that  $\text{POP}^*$  properly extends the safe composition scheme (SC).

**Example 3.14** (Example 2.5 continued). Observe that  $\mathcal{R}_{\text{btree}} \subseteq >_{\text{pop}^*}$  with any admissible precedence satisfying  $\text{btree} \succ \text{dup}$  for the TRS  $\mathcal{R}_{\text{btree}}$  depicted in Example 2.5. We use  $\text{nrm}(\text{btree}) = \{1\}$  and  $\text{nrm}(\text{dup}) = \emptyset$  for the defined function symbols. Theorem 3.11 thus implies that the (innermost) runtime complexity of  $\mathcal{R}_{\text{btree}}$  is polynomial. On the other hand, we already observed that  $\mathcal{R}_{\text{btree}}$  admits exponentially long *outermost* reductions.

**Example 3.15.** Consider the TRS  $\mathcal{R}_{nc}$  given by the rules

$$\begin{array}{ll} 43: f(n; ) \rightarrow h(; gs(n; )) & 44: gs(0; ) \rightarrow 0 \\ 45: h(; g(; n)) \rightarrow c(; h(; n), h(; n)) & 46: gs(s(; n); ) \rightarrow g(; gs(n; )) \\ 47: g(; \perp) \rightarrow c(; h(; \perp), h(; \perp)) , \end{array}$$

where we suppose that the only constructors are  $\perp$ ,  $0$  and  $s$ . The rule (47) is used to define the symbol  $g$ , and to properly set up the precedence. The rules (44) and (46) are used to translate a tower  $s^n(; 0)$  to  $g^n(; 0)$ , using rule (43) we thus obtain a family of reductions

$$f(s^n(; 0); ) \rightarrow_{\mathcal{R}_{nc}} h(; gs(s^n(; 0); )) \rightarrow_{\mathcal{R}_{nc}}^* h(; g^n(; 0)) ,$$

for  $n \in \mathbb{N}$ . It is not difficult to see that the derivation height of the final term  $h(;g^n(;0))$  grows exponentially in  $n$  due to rule (45), and so the innermost runtime complexity of  $\mathcal{R}_{nc}$  is bounded by an exponential from below.

On the other hand,  $\mathcal{R}_{nc}$  is compatible with a polynomial path order  $>_{\text{pop}^*}$  as induced by a precedence  $\succ$  satisfying  $f \succ gs \succ g \succ h \succ c$  and the safe mapping as indicated in the rules. Observe that for rule (45) we exploit that  $g$  is defined, conclusively one can show  $g(;m) >_{\text{pop}^*} c(;h(;m), h(;m))$  and therefore  $h(;g(;m)) >_{\text{pop}^*}^{(1)} c(;h(;m), h(;m))$  holds. However, due to rule (47)  $\mathcal{R}_{nc}$  is not a constructor TRS, as demanded by Theorem 3.11.

Bellantoni and Cook's characterisation, as well as Leivant's work on tiered recurrence, was originally stated on word algebras. In quite recent work by Dal Lago et al., [39] the result by Leivant [40, 42] has been extended to arbitrary free algebras. In particular, in [39] a function  $f$  is defined by *general ramified recursion* as

$$f(c(x_1, \dots, x_n), \vec{y}) = h_c(x_1, \dots, x_n, \vec{y}, f(x_1, \vec{y}), \dots, f(x_n, \vec{y})) ,$$

for every  $n$ -ary constructor  $c$ , provided a data tiering principle is satisfied. Due to the linearity condition imposed on  $>_{\text{pop}^*}^{(2)}$ , such a recursion principle cannot be expressed in predicative recursive TRSs. Indeed, allowing this form of recursion would invalidate Theorem 3.11.

**Example 3.16** (Example 2.5 continued). Let  $\mathcal{R}_{gr}$  denote the extension of  $\mathcal{R}_{btree}$  from Example 2.5 by the rules

$$\begin{aligned} 48: & \quad f(n;) \rightarrow \text{traverse}(\text{btree}(n;);) \\ 49: & \quad \text{traverse}(\text{leaf};) \rightarrow \text{leaf} \\ 50: & \quad \text{traverse}(\text{node}(;x,y);) \rightarrow \text{node}(;\text{traverse}(x;), \text{traverse}(y;)) . \end{aligned}$$

The above definition of  $f$  is expressible in the system of [39], by simply conceiving the rewrite rules as defining equations. In particular it follows that  $\llbracket f \rrbracket$  is polytime computable. On the other hand, the runtime complexity of  $\mathcal{R}_{gr}$  is at least exponential as a derivation of  $f(s^n(0))$ ,  $\mathcal{R}_{gr}$  traverses every node of a binary tree of height  $n$ .

Finally we note that the order  $>_{\text{pop}^*}$  is *blind* on constructors, in particular  $>_{\text{pop}^*}$  collapses to the subterm relation (modulo equivalence) on values.

**Lemma 3.17.** Suppose the precedence underlying  $>_{\text{pop}^*}$  is admissible. If  $s >_{\text{pop}^*} t$  and  $s \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  then  $t$  is a safe subterm of  $s$  (modulo  $\approx$ ), in particular  $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  holds.

#### 4. THE POLYNOMIAL PATH ORDER ON SEQUENCES

Fix again a safe mapping  $\text{safe}$  on the signature  $\mathcal{F}$ . We now define the notion of *predicative interpretation* of a term  $t$ . Guided by the safe mapping, predicative interpretations map terms to sequences of terms. We then introduce the *polynomial path order on sequences*, intended to orient images of the predicative interpretation as outlined before.

To formalise sequences, 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  $\circ(t_1, \dots, t_n)$ , in particular if we write  $f(t_1, \dots, t_n)$  then  $f \neq \circ$ . Define the *normalised signature*  $\mathcal{F}_n$  as follows:

$$\mathcal{F}_n := \{f_n \mid f \in \mathcal{F}, \text{nrm}(f) = \{i_1, \dots, i_k\} \text{ and } \text{ar}(f_n) = k\} .$$

The predicative interpretation of a term  $f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$  results in a sequence  $[f_n(a_1, \dots, a_k)] \# a_{k+1} \# \dots \# a_{k+l}$ , where  $\#$  denotes *concatenation* of sequences and the sequences  $a_i$  are predicative interpretations of the corresponding arguments  $s_i$  ( $i = 1, \dots, k+l$ ). Note that in the interpretations, terms have sequences as arguments.

**Definition 4.1.** The set of *terms with sequence arguments*  $\mathcal{S}(\mathcal{F}, \mathcal{V}) \subseteq \mathcal{T}(\mathcal{F}_n \uplus \{\circ\}, \mathcal{V})$  and the set of *sequences*  $\mathcal{S}^*(\mathcal{F}, \mathcal{V}) \subseteq \mathcal{T}(\mathcal{F}_n \uplus \{\circ\}, \mathcal{V})$  is inductively defined as follows:

- (1)  $\mathcal{V} \subseteq \mathcal{S}(\mathcal{F}, \mathcal{V})$ , and
- (2) if  $a_1, \dots, a_n \in \mathcal{S}^*(\mathcal{F}, \mathcal{V})$  and  $f \in \mathcal{F}_n$  then  $f(a_1, \dots, a_n) \in \mathcal{S}(\mathcal{F}, \mathcal{V})$ , and
- (3) if  $t_1, \dots, t_n \in \mathcal{S}(\mathcal{F}, \mathcal{V})$  then  $[t_1 \ \dots \ t_n] \in \mathcal{S}^*(\mathcal{F}, \mathcal{V})$ .

We always write  $a, b, \dots$ , possibly extended by subscripts, for elements from  $\mathcal{S}(\mathcal{F}, \mathcal{V})$  and  $\mathcal{S}^*(\mathcal{F}, \mathcal{V})$ . The restriction of  $\mathcal{S}(\mathcal{F}, \mathcal{V})$  and  $\mathcal{S}^*(\mathcal{F}, \mathcal{V})$  to ground terms is denoted by  $\mathcal{S}(\mathcal{F})$  and  $\mathcal{S}^*(\mathcal{F})$  respectively. When no confusion can arise from this we call terms with sequence arguments simply terms. Further, we sometimes abuse set notation and write  $b \in [a_1 \ \dots \ a_n]$  if  $b = a_i$  for some  $i \in \{1, \dots, n\}$ . We denote by  $a \# b$  the *concatenation* of  $a \in \mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$  and  $b \in \mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$ . To avoid notational overhead we overload concatenation to both terms and sequences. Let  $\text{lift}(a) := [a]$  if  $a \in \mathcal{S}(\mathcal{F}, \mathcal{V})$  and  $\text{lift}(a) := a$  if  $a \in \mathcal{S}^*(\mathcal{F}, \mathcal{V})$ . We set  $a \# b := [a_1 \ \dots \ a_n \ b_1 \ \dots \ b_m]$  where  $\text{lift}(a) = [a_1 \ \dots \ a_n]$  and  $\text{lift}(b) = [b_1 \ \dots \ b_m]$ . As concatenation is associative we drop parenthesis at will. We define the *length* over  $\mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$  as  $\text{len}(a) := n$  where  $\text{lift}(a) = [a_1 \ \dots \ a_n]$ . The *sequence width wd* (or *width* for short) of an element  $a \in \mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$  is given recursively by

$$\text{wd}(a) := \begin{cases} 1 & \text{if } a \text{ is a variable,} \\ \max\{1, \text{wd}(a_1), \dots, \text{wd}(a_n)\} & \text{if } a = f(a_1, \dots, a_n) \text{ with } f \in \mathcal{F}_n, \text{ and} \\ \sum_{i=1}^n \text{wd}(a_i) & \text{if } a = [a_1 \ \dots \ a_n]. \end{cases}$$

In the following we tacitly employ  $\text{len}(a) \leq \text{wd}(a)$  and  $\text{wd}(a \# b) = \text{wd}(a) + \text{wd}(b)$  for all  $a, b \in \mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$ . We define the *norm* of  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  in correspondence to the depth of  $t$ , but disregard normal argument positions.

$$\text{nm}(t) := \begin{cases} 1 & t \text{ is a variable,} \\ 1 + \max\{0\} \cup \{\text{nm}(t_j) \mid j = k+1, \dots, k+l\} & t = f(t_1, \dots, t_k; t_{k+1}, \dots, t_{k+l}). \end{cases}$$

Note that since all argument positions of constructors are safe, the norm  $\text{nm}(\cdot)$  and depth  $\text{dp}(\cdot)$  coincide on values.

Predicative interpretations are given by two mappings  $\mathbf{S}$  and  $\mathbf{N}$ : the interpretation  $\mathbf{S}$  is applied on safe arguments and removes normal forms; the interpretation  $\mathbf{N}$  is applied to normal arguments and additionally encodes the norm of the given term as tally sequence. The latter allows us to track the maximal depth of normal forms erased by  $\mathbf{S}$ . Let  $\bullet \notin \mathcal{F}_n$  be a fresh constant. To encode natural numbers  $n \in \mathbb{N}$ , define its *tally sequence representation*  $\overline{n}$  as the sequence containing  $n$  occurrences of this fresh constant:  $\overline{0} = []$  and  $\overline{n+1} = \bullet \# \overline{n}$ .

**Definition 4.2.** A *predicative interpretation* for a TRS  $\mathcal{R}$  is a pair  $(\mathbf{S}_{\mathcal{R}}, \mathbf{N}_{\mathcal{R}})$  of mappings  $\mathbf{S}_{\mathcal{R}}, \mathbf{N}_{\mathcal{R}} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{S}^*(\mathcal{F} \cup \{\bullet\})$  defined as follows:

$$\begin{aligned} \mathbf{S}_{\mathcal{R}}(t) &:= \begin{cases} [] & \text{if } t \in \text{NF}(\mathcal{R}), \\ [f_n(\mathbf{N}_{\mathcal{R}}(t_1), \dots, \mathbf{N}_{\mathcal{R}}(t_k))] \# \mathbf{S}(t_{k+1}) \# \dots \# \mathbf{S}(t_{k+l}) & \text{otherwise where } (\star), \end{cases} \\ \mathbf{N}_{\mathcal{R}}(t) &:= \mathbf{S}_{\mathcal{R}}(t) \# \overline{\text{nm}(t)}. \end{aligned}$$



Here  $(\star)$  stands for  $t = f(t_1, \dots, t_k; t_{k+1}, \dots, t_{k+l})$ .

As the rewrite system  $\mathcal{R}$  is usually clear from the context, we drop the references to  $\mathcal{R}$  when unambiguous.

Below we introduce the order  $\blacktriangleright$  on sequences  $\mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$ . In the next section we then embed innermost  $\mathcal{R}$ -steps into this order, and use  $\blacktriangleright$  to estimate the length of reductions accordingly. For basic terms  $s = f(u_1, \dots, u_k; u_{k+1}, \dots, u_{k+l})$  we obtain

$$S(s) = [f_n(\mathbf{N}(u_1), \dots, \mathbf{N}(u_k))] \# S(u_{k+1}) \# \dots \# S(u_{k+l}) = [f_n(\overline{\mathbf{dp}(u_1)}, \dots, \overline{\mathbf{dp}(u_k)})].$$

Hence the obtained bound depends on depths of normal arguments only. To get the reader prepared for the definition of  $\blacktriangleright$ , we exemplify Definition 4.2 on a predicative recursive TRS.

**Example 4.3.** Consider following TRS  $\mathcal{R}_f$ , where we suppose that besides  $f$ , also  $g$  and  $h$  are defined symbols:

$$51: f(0; y) \rightarrow y \qquad 52: f(s(x); y) \rightarrow g(h(x); f(x; y))$$

Consider a substitution  $\sigma : \mathcal{V} \rightarrow \mathbf{NF}$ . Using that  $S(v) = []$  and  $\mathbf{N}(v) = \overline{\mathbf{dp}(v)}$  for all normal forms  $v$ , the embedding  $S(l\sigma) \blacktriangleright S(r\sigma)$  of root steps ( $l \rightarrow r \in \mathcal{R}_f$ ) results in the following order constraints.

$$\begin{aligned} [f_n(\overline{1})] \blacktriangleright [] & \text{from rule 1} \\ [f_n(\overline{\mathbf{dp}(x\sigma) + 1})] \blacktriangleright [g_n(\mathbf{N}(h(x\sigma;))) f_n(\overline{\mathbf{dp}(x\sigma)})] & \text{from rule 2,} \end{aligned}$$

where  $\mathbf{N}(h(x\sigma;)) = [h_n(\mathbf{N}(x\sigma))] \# \overline{\mathbf{nm}(h(x\sigma;))} = [h_n(\overline{\mathbf{dp}(x\sigma)}) \bullet]$ . Consider now a step

$$s = f(s_1, \dots, s_i, \dots, s_{k+l}) \xrightarrow{i}_{\mathcal{R}_f} f(s_1, \dots, t_i, \dots, s_{k+l}) = t,$$

below the root, where  $s_i \xrightarrow{i}_{\mathcal{R}_f} t_i$ . Depending on the rewrite position  $i$ , which is either normal ( $i \in \{1, \dots, k\}$ ) or safe ( $i \in \{k+1, \dots, k+l\}$ ), the predicative embedding introduces one of the following two constraints:

$$\begin{aligned} S(s) &= [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(s_i), \dots, \mathbf{N}(s_k))] \# S(s_{k+1}) \# \dots \# S(s_{k+l}) & (a) \\ &\blacktriangleright [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(t_i), \dots, \mathbf{N}(s_k))] \# S(s_{k+1}) \# \dots \# S(s_{k+l}) = \mathbf{N}(t), \text{ or} \\ S(s) &= [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(s_k))] \# S(s_{k+1}) \# \dots \# S(s_i) \# \dots \# S(s_{k+l}) & (b) \\ &\blacktriangleright [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(s_k))] \# S(s_{k+1}) \# \dots \# S(t_i) \# \dots \# S(s_{k+l}) = S(t). \end{aligned}$$

To be able to deal with steps below normal argument positions as in (a), we also orient images of  $\mathbf{N}$ . This results additionally in following constraints:

$$\begin{aligned} S(f(0; y\sigma)) \# \overline{\mathbf{dp}(y\sigma) + 1} \blacktriangleright S(y\sigma) \# \overline{\mathbf{dp}(y\sigma)} & \text{from rule 1} \\ S(f(s(x\sigma); y\sigma)) \# \overline{\mathbf{dp}(y\sigma) + 1} \blacktriangleright S(g(h(x\sigma); f(x\sigma; y\sigma))) \# \overline{\mathbf{dp}(y\sigma) + 2} & \text{from rule 2.} \end{aligned}$$

The *polynomial path order on sequences* (POP for short), denoted by  $\blacktriangleright$ , constitutes a generalisation of the *path order for FP* as put forward in [4]. Whereas we previously used the notion of safe mapping to dictate predicative recursion on compatible TRSs, the order on sequences relies on the explicit separation of safe arguments as given by predicative interpretations. Following Buchholz [23], it suffices to present *finite approximations*  $\blacktriangleright_{k,l}$  of  $\blacktriangleright$ . The parameters  $k \in \mathbb{N}$  and  $l \in \mathbb{N}$  are used to controls the width and depth of right-hand sides. Fix a precedence  $\succcurlyeq$  on the normalised signature  $\mathcal{F}_n$ . We extend term equivalence with respect to  $\succcurlyeq$  to sequences by disregarding the order on elements.

**Definition 4.4.** We define  $a \sim b$  if  $a = b$  or there exists a permutation  $\pi$  such that  $a_i \sim b_{\pi(i)}$  for all  $i = 1, \dots, n$ , where either (i)  $a = [a_1 \cdots a_n]$ ,  $b = [b_1 \cdots b_n]$ , or (ii)  $a = f(a_1, \dots, a_n)$ ,  $b = g(b_1, \dots, b_n)$  and  $f \sim g$ .

In correspondence to  $\succ_{\text{pop}^*}$ , the order  $\blacktriangleright_{k,l}$  is based on an auxiliary order  $\succ_{k,l}$ .

**Definition 4.5.** Let  $k, l \geq 1$ . We define  $\succ_{k,l}$  with respect to the precedence  $\succcurlyeq$  inductively as follows:

- (1)  $f(a_1, \dots, a_n) \succ_{k,l} b$  if  $a_i \succcurlyeq_{k,l} b$  for some  $i \in \{1, \dots, n\}$ ;
- (2)  $f(a_1, \dots, a_n) \succ_{k,l} g(b_1, \dots, b_m)$  if  $f \succcurlyeq g$  and the following conditions are satisfied:
  - $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$  for all  $j = 1, \dots, m$ ;
  - $m \leq k$ ;
- (3)  $f(a_1, \dots, a_n) \succ_{k,l} [b_1 \cdots b_m]$  if the following conditions are satisfied:
  - $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$  for all  $j = 1, \dots, m$ ;
  - $m \leq \text{wd}(f(a_1, \dots, a_n)) + k$ ;
- (4)  $[a_1 \cdots a_n] \succ_{k,l} [b_1 \cdots b_m]$  if the following conditions are satisfied:
  - $[b_1 \cdots b_m] \sim c_1 \# \cdots \# c_n$ ;
  - $a_i \succcurlyeq_{k,l} c_i$  for all  $i = 1, \dots, n$ ;
  - $a_{i_0} \succ_{k,l} c_{i_0}$  for at least one  $i_0 \in \{1, \dots, n\}$ ;
  - $m \leq \text{wd}([a_1 \cdots a_n]) + k$ ;

Here  $\succcurlyeq_{k,l}$  denotes  $\succ_{k,l} \cup \sim$ . We write  $\succ_k$  to abbreviate  $\succ_{k,k}$ .

We stress that the definition lacks a case  $f(a_1, \dots, a_n) \succ_{k,l} g(b_1, \dots, b_m)$  where  $f \sim g$ . Still the order is sufficient to account for terms oriented by the auxiliary order  $\succ_{\text{pop}}$ .

**Example 4.6** (Example 4.3 continued). Reconsider rule (52) from the TRS  $\mathcal{R}_f$  given in Example 4.3, where in particular  $f(\mathbf{s}(x); y) \succ_{\text{pop}} h(x; \cdot)$ . We show below  $f_n(\overline{\text{dp}(x\sigma) + 1}) \succ_{1,4} N(h(\mathbf{s}(x\sigma); \cdot))$  for all substitutions  $\sigma : \mathcal{V} \rightarrow \text{NF}$ . First recall that by the overloading of concatenation, we can write

$$\bar{n} = [\bullet \cdots \bullet] = \bullet \# \cdots \# \bullet \# [] \# \cdots \# []$$

with  $n$  occurrences of  $\bullet$ , appending  $m$ -times the empty sequence  $[]$  for all  $n, m \in \mathbb{N}$ . Using that  $\bullet \sim \bullet$  and  $\bullet \succ_{k,l-1}^{(3)} []$  for  $l \geq 2$ , we can thus prove  $\overline{n+m} \succ_{k,l}^{(4)} \bar{n}$  whenever  $m \geq 1$ . Moreover we have

$$\begin{array}{lll} 1 : & \overline{\text{dp}(x\sigma) + 1} \succ_{1,2}^{(4)} \overline{\text{dp}(x\sigma)} & \text{as } \text{dp}(x\sigma) + 1 > \text{dp}(x\sigma) \\ 2 : & f_n(\overline{\text{dp}(x\sigma) + 1}) \succ_{1,3}^{(2)} h_n(\overline{\text{dp}(x\sigma)}) & \text{using } f_n \succcurlyeq h_n \text{ and } 1 \\ 3 : & f_n(\overline{\text{dp}(x\sigma) + 1}) \succ_{1,4}^{(3)} [h_n(\overline{\text{dp}(x\sigma)}) \bullet] & \text{by 2 and } f_n(\dots) \succ_{1,3}^{(2)} \bullet \\ & = N(h(x\sigma; \cdot)) . & \end{array}$$

We arrive at the definition of the full order  $\blacktriangleright_{k,l}$ .

**Definition 4.7.** Let  $k, l \geq 1$ . We define  $\blacktriangleright_{k,l}$  inductively as the least extension of  $\succ_{k,l}$  such that:

- (1)  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} b$  if  $a_i \blacktriangleright_{k,l} b$  for some  $i \in \{1, \dots, n\}$ ;
- (2)  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} g(b_1, \dots, b_m)$  if  $f \sim g$  and following conditions are satisfied:
  - $\{\{a_1, \dots, a_n\}\} \blacktriangleright_{k,l}^{\text{mul}} \{\{b_1, \dots, b_m\}\}$ ;
  - $m \leq k$ ;
- (3)  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} [b_1 \cdots b_m]$  and following conditions are satisfied:

- $f(a_1, \dots, a_n) \blacktriangleright_{k,l-1} b_{j_0}$  for at most one  $j_0 \in \{1, \dots, m\}$ ;
  - $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$  for all  $j \neq j_0$ ;
  - $m \leq \text{wd}(f(a_1, \dots, a_n)) + k$ ;
- (4)  $[a_1 \cdots a_n] \blacktriangleright_{k,l} [b_1 \cdots b_m]$  and following conditions are satisfied:
- $[b_1 \cdots b_m] \sim c_1 \# \cdots \# c_n$ ;
  - $a_i \blacktriangleright_{k,l} c_i$  for all  $i = 1, \dots, n$ ;
  - $a_{i_0} \blacktriangleright_{k,l} c_{i_0}$  for at least one  $i_0 \in \{1, \dots, n\}$ ;
  - $m \leq \text{wd}([a_1 \cdots a_n]) + k$ ;

Here  $\blacktriangleright_{k,l}$  denotes  $\blacktriangleright_{k,l} \cup \sim$ . We write  $\blacktriangleright_k$  to abbreviate  $\blacktriangleright_{k,k}$ .

The polynomial path order on sequences forms a restriction of the recursive path order with multiset status, where the variadic symbol  $\circ$  is implicitly ranked lowest in the precedence. As a consequence the order it is well-founded [28]. The use of the auxiliary order in  $\blacktriangleright_{k,l}^{(3)}$  accounts for our restriction that predicative recursive TRSs admit at most one recursive call per right-hand side. Observe  $t \blacktriangleright_k^{(3)} []$  for all terms  $t \notin \mathcal{V}$ , consequently  $[t_1 \cdots t_n] \blacktriangleright_k^{(4)} []$  if at least one term  $t_i$  is ground.

**Example 4.8** (Example 4.6 continued). We continue with the orientation of root steps from the TRS  $\mathcal{R}_f$  depicted in Example 4.3 for substitutions  $\sigma : \mathcal{V} \rightarrow \text{NF}$ . Consider the more involved case  $f(\mathbf{s}(x\sigma); y\sigma) \xrightarrow{i}_{\mathcal{R}_f} \mathbf{g}(\mathbf{h}(x\sigma); \mathbf{f}(x\sigma; y\sigma))$  due to rule (52). Note that in the orientation below we use  $\blacktriangleright_k^{(2)}$  to orient the recursive call (proof step 5), and  $\succ_k^{(2)}$  for the remaining elements (proof step 6).

$$\begin{array}{ll}
4 : & \overline{\text{dp}(x\sigma) + 1} \blacktriangleright_{1,2}^{(4)} \overline{\text{dp}(x\sigma)} & \text{as in Example 4.6} \\
5 : & \mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1}) \blacktriangleright_{1,3}^{(2)} \mathbf{f}_n(\overline{\text{dp}(x\sigma)}) & \text{using 4} \\
6 : & \mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1}) \succ_{1,5}^{(2)} \mathbf{g}_n(\mathbf{N}(\mathbf{h}(x\sigma; ))) & \text{using } \mathbf{f}_n \succ \mathbf{g}_n \text{ and 3} \\
7 : & \mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1}) \blacktriangleright_{1,6}^{(3)} [\mathbf{g}_n(\mathbf{N}(\mathbf{h}(x\sigma; ))) \mathbf{f}_n(\overline{\text{dp}(x\sigma)})] & \text{using 5 and 6} \\
& & = \mathbf{S}(\mathbf{g}(\mathbf{h}(x\sigma; ); \mathbf{f}(x\sigma; y\sigma))) \\
8 : & \mathbf{S}(\mathbf{f}(\mathbf{s}(x\sigma); y\sigma)) = [\mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1})] & \\
& & \blacktriangleright_{1,6}^{(3)} \mathbf{S}(\mathbf{g}(\mathbf{h}(x\sigma; ); \mathbf{f}(x\sigma; y\sigma))) & \text{using 7} \\
9 : & \mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1}) \blacktriangleright_{2,6}^{(3)} [\mathbf{g}_n(\mathbf{N}(\mathbf{h}(x\sigma; ))) \mathbf{f}_n(\overline{\text{dp}(x\sigma)}) \bullet] & \text{by 5, 6 and } \mathbf{f}_n(\dots) \blacktriangleright_{1,1} \bullet \\
10 : & \mathbf{N}(\mathbf{f}(\mathbf{s}(x\sigma); y\sigma)) = [\mathbf{f}_n(\overline{\text{dp}(x\sigma) + 1})] \# \overline{\text{dp}(y\sigma) + 1} & \\
& & \blacktriangleright_{2,6}^{(3)} \mathbf{S}(\mathbf{g}(\mathbf{h}(x\sigma; ); \mathbf{f}(x\sigma; y\sigma))) \# \overline{\text{dp}(y\sigma) + 1} & \text{using 9 and } \bullet \sim \bullet \\
& & = \mathbf{N}(\mathbf{g}(\mathbf{h}(x\sigma; ); \mathbf{f}(x\sigma; y\sigma))) .
\end{array}$$

For the last orientation we employ that the width of the left-hand side is at least  $\text{dp}(y\sigma) + 2$ , and the length of the right hand side is  $\text{dp}(y\sigma) + 4$ , as required we have

$$\text{len}(\mathbf{N}(\mathbf{g}(\mathbf{h}(x\sigma; ); \mathbf{f}(x\sigma; y\sigma)))) \leq \text{wd}(\mathbf{N}(\mathbf{f}(\mathbf{s}(x\sigma); y\sigma))) + 2 .$$

Observe that as in the above example, the parameter  $l$  in  $\blacktriangleright_{k,l}$  controls the depth of the proof tree of  $a \blacktriangleright_{k,l} b$ . Since leafs of such proof trees hold either due to case  $\blacktriangleright_k^{(1)}$  or the absence of arguments in the right-hand side, it follows that the depth of  $b$  is bounded linearly in  $l$  and the depth of  $a$ . From the example it should also be clear how the parameter

$k$  controls the length of right-hand sides, compare steps 9 and 10 where we had to increase the parameter  $k$ .

In the example we obtained that the predicative embedding of root steps  $l\sigma \xrightarrow{\mathcal{R}_f} r\sigma$  of predicative recursive TRS  $\mathcal{R}_f$  is possible for  $k = 2$ , independent on the considered substitution  $\sigma$ . The next lemma clarifies that such a global  $k$  can always be found, and depends on the right-hand sides only.

**Lemma 4.9.** Let  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l}) \in \mathcal{T}_b$ ,  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ , and  $\sigma$  be a normalising substitution. Then

- (1)  $\text{len}(\mathbf{S}(t\sigma)) \leq |t|$ ; and
- (2) if  $s >_{\text{pop}} t$  then  $\text{len}(\mathbf{N}(t\sigma)) \leq \max\{\text{nm}(s_1\sigma), \dots, \text{nm}(s_k\sigma)\} + 2 \cdot |t|$ ; and
- (3) if  $s >_{\text{pop}^*} t$  then  $\text{len}(\mathbf{N}(t\sigma)) \leq \max\{\text{nm}(s_1\sigma), \dots, \text{nm}(s_{k+l}\sigma)\} + 2 \cdot |t|$ .

As a consequence of the above lemma we obtain: if  $l\sigma \xrightarrow{\mathcal{R}} r\sigma$  is a root step of a predicative TRS  $\mathcal{R}$ , then  $\text{len}(\mathbf{Q}(r\sigma)) \leq \text{wd}(\mathbf{Q}(l\sigma)) + 2 \cdot |r|$  for  $\mathbf{Q} \in \{\mathbf{S}, \mathbf{N}\}$ . In the predicative embedding we instantiate  $k$  by twice the maximum size of right-hand sides of  $\mathcal{R}$ . The side-conditions imposed on  $\blacktriangleright_{k,l}$  allow us to estimate the length of right-hand sides based on the width of left-hand sides and the parameter  $k$ . This and other frequently used properties are collected in the next lemma, whose proof is not difficult.

**Lemma 4.10.** The following properties hold for all  $k \geq 1$  and  $a, b, c_1, c_2 \in \mathcal{S}(\mathcal{F}, \mathcal{V}) \cup \mathcal{S}^*(\mathcal{F}, \mathcal{V})$ .

- (1)  $>_l \subseteq \blacktriangleright_l \subseteq \blacktriangleright_k$  for all  $l \leq k$ ;
- (2)  $\sim \cdot \blacktriangleright_k \cdot \sim \subseteq \blacktriangleright_k$ ;
- (3)  $a \blacktriangleright_k b$  implies  $\text{len}(b) \leq \text{wd}(a) + k$ ;
- (4)  $a \blacktriangleright_k b$  implies  $c_1 \dot{+} a \dot{+} c_2 \blacktriangleright_k c_1 \dot{+} b \dot{+} c_2$ .

Following [4] we define a function  $\mathbf{G}_k$  that measures the  $\blacktriangleright_k$ -descending lengths on sequences. To simplify matters, we restrict the definition of  $\mathbf{G}_k$  to ground sequences. As images of predicative interpretations are always ground, this suffices for our purposes.

**Definition 4.11.** We define  $\mathbf{G}_k : \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F}) \rightarrow \mathbb{N}$  as

$$\mathbf{G}_k(a) := 1 + \max\{\mathbf{G}_k(b) \mid b \in \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F}) \text{ and } a \blacktriangleright_k b\}.$$

Note that due to Lemma 4.10 (2),  $\mathbf{G}_k(a) = \mathbf{G}_k(b)$  whenever  $a \sim b$ . The next lemma confirms that sequences act purely as containers.

**Lemma 4.12.** For  $[a_1 \ \dots \ a_n] \in \mathcal{S}^*(\mathcal{F})$  it holds that  $\mathbf{G}_k([a_1 \ \dots \ a_n]) = \sum_{i=1}^n \mathbf{G}_k(a_i)$ .

*Proof.* Let  $a = [a_1 \ \dots \ a_n] \in \mathcal{S}^*(\mathcal{F})$ . We first show  $\mathbf{G}_k(a) \geq \sum_{i=1}^n \mathbf{G}_k(a_i)$ . Let  $b, c \in \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$  and consider maximal sequences  $b \blacktriangleright_k b_1 \blacktriangleright_k \dots \blacktriangleright_k b_o$  and  $c \blacktriangleright_k c_1 \blacktriangleright_k \dots \blacktriangleright_k c_p$ . Using Lemma 4.10 (4) repeatedly we get

$$b \dot{+} c \blacktriangleright_k b_1 \dot{+} c \blacktriangleright_k \dots \blacktriangleright_k b_o \dot{+} c \blacktriangleright_k b_o \dot{+} c_1 \blacktriangleright_k \dots \blacktriangleright_k b_o \dot{+} c_p,$$

and thus  $\mathbf{G}_k(b \dot{+} c) \geq \mathbf{G}_k(b) + \mathbf{G}_k(c)$  holds for all  $b, c \in \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$ . We conclude  $\mathbf{G}_k(a) = \mathbf{G}_k(a_1 \dot{+} \dots \dot{+} a_n) \geq \sum_{i=1}^n \mathbf{G}_k(a_i)$  with a straight forward induction on  $n$ .

It remains to verify  $\mathbf{G}_k(a) \leq \sum_{i=1}^n \mathbf{G}_k(a_i)$ . For this we show that  $a \blacktriangleright_k b$  implies  $\mathbf{G}_k(b) < \sum_{i=1}^n \mathbf{G}_k(a_i)$  by induction on  $\mathbf{G}_k(a)$ . Consider the base case  $\mathbf{G}_k(a) = 0$ . Since  $a$  is ground it follows that  $a = []$ , the claim is trivially satisfied. For the inductive step  $\mathbf{G}_k(a) > 1$ , let  $a \blacktriangleright_k b$ . Since  $a$  is a sequence,  $a \blacktriangleright_k^{(4)} b$ . Hence  $b \sim b_1 \dot{+} \dots \dot{+} b_n$  where  $a_i \blacktriangleright_k b_i$  and thus  $\mathbf{G}_k(b_i) \leq \mathbf{G}_k(a_i)$  for all  $i = 1, \dots, n$ . Additionally  $a_{i_0} \blacktriangleright_k b_{i_0}$  and hence

$\mathbf{G}_k(b_{i_0}) < \mathbf{G}_k(a_{i_0})$  for at least one  $i_0 \in \{1, \dots, n\}$ . As in the first half of the proof, one verifies  $\mathbf{G}_k(b_i) \leq \mathbf{G}_k(b)$  for all  $i = 1, \dots, n$ . Note  $\mathbf{G}_k(b) < \mathbf{G}_k(a)$  as  $a \blacktriangleright_k b$ , hence induction hypothesis is applicable to  $b$  and all  $b_i$  ( $i = 1, \dots, n$ ). It follows that

$$\mathbf{G}_k(b) \leq \sum_{c \leq b} \mathbf{G}_k(c) \leq \sum_{i=1}^n \sum_{c \in b_i} \mathbf{G}_k(c) = \sum_{i=1}^n \mathbf{G}_k(b_i) < \sum_{i=1}^n \mathbf{G}_k(a_i).$$

This concludes the second part of the proof.  $\square$

The central theorem of this section, Theorem 4.17, states that  $\mathbf{G}_k(f(a_1, \dots, a_n))$  is polynomial in  $\sum_i^n \mathbf{G}_k(a_i)$ , where the polynomial bound depends only on  $k$  and the rank  $p$  of  $f$ . The proof of this is involved. To cope with the multiset comparison underlying  $\blacktriangleright_k^{(2)}$ , we introduce as a first step an *order-preserving* extension  $\mathbf{G}_k^n$  of  $\mathbf{G}_k$  to multisets of sequences, in the sense that  $\mathbf{G}_k^n(a_1, \dots, a_n) > \mathbf{G}_k^n(b_1, \dots, b_m)$  holds whenever  $\{\{a_1, \dots, a_n\} \blacktriangleright_k^{\text{mul}} \{b_1, \dots, b_m\}\}$  (provided  $k \geq m, n$ , cf. Lemma 4.15). As the next step toward our goal, we estimate  $\mathbf{G}_k(f(a_1, \dots, a_n))$  in terms of  $\mathbf{G}_k^n(a_1, \dots, a_n)$  whenever  $n \leq k$  and  $\text{rk}(f) \leq k$ . Technically we bind following functions by polynomials  $q_{k,p}$ . For all  $k, p \in \mathbb{N}$  with  $k \geq 1$  we define  $F_{k,p} : \mathbb{N} \rightarrow \mathbb{N}$  as

$$F_{k,p}(m) := \max\{\mathbf{G}_k(f(a_1, \dots, a_n)) \mid f(a_1, \dots, a_n) \in \mathcal{S}(\mathcal{F}), \text{rk}(f) \leq p, n \leq k \text{ and } \mathbf{G}_k^n(a_1, \dots, a_n) \leq m\}.$$

The definition of  $\mathbf{G}_k^n$  is defined in terms of an order-preserving homomorphism from  $\mathcal{M}(\mathbb{N})$  to  $\mathbb{N}$ . To illustrate the construction carried out below, consider the following example.

**Example 4.13.** Consider multisets  $\mathcal{M}(\mathbb{N})$  of size  $k$ . Conceive such multisets  $\{\{m_1, \dots, m_k\}\}$  as natural numbers written in base- $c$  (with  $c > m_i$  for all  $i = 1, \dots, k$ ), where digits  $m_1, \dots, m_k$  are sorted from left to right in decreasing order. Then one can formulate chains  $M_1 >^{\text{mul}} M_2 >^{\text{mul}} \dots$  that can be understood as decreasing counters which however wrap from  $\{m_1, \dots, m_i + 1, 0, \dots, 0\}$  to  $\{m_1, \dots, m_i, m_i, \dots, m_i\}$ . Compare the TRS  $\mathcal{R}_k$  defined in Lemma 3.12 that models such counters. Using the correspondence, it is easy to prove that the length of a chain of this form starting from  $\{c-1, \dots, c-1\}$  is given by

$$\sum_{m_1=0}^{c-1} \sum_{m_2=0}^{m_1} \dots \sum_{m_k=0}^{m_{k-1}} m_k = \Omega(c^{k+1}).$$

The inclusion follows by  $k$ -times application of the Faulhaber's formula [37], which states that for all  $n, l \in \mathbb{N}$ ,  $\sum_{i=1}^n i^l = p_{l+1}(n)$  for some polynomial  $p_{l+1}$  of degree  $l+1$ .

The above example gives a polynomial lower bound on the number of  $>^{\text{mul}}$  descending sequences on multisets  $\mathcal{M}(\mathbb{N})$  of size  $k$ . We now prove that this lower bound also serves as an asymptotic upper bound, for all multisets of natural number of length *up to*  $k$ . For  $k \geq n \in \mathbb{N}$  and  $c \in \mathbb{N}$  we define the family of functions  $\mathbf{h}_{k,c}^n : \mathbb{N}^l \rightarrow \mathbb{N}$  such that

$$\mathbf{h}_{k,c}^n(m_1, \dots, m_n) = \sum_{i=1}^n \text{sort}^n(m_1, \dots, m_n, i) \cdot c^{(k-i)}.$$

Here  $\text{sort}^n(m_1, \dots, m_n, i)$  denote the  $i^{\text{th}}$  element of  $m_1, \dots, m_n$  sorted in descending order, i.e.,  $\text{sort}^n(m_1, \dots, m_n, i) := m_{\pi(i)}$  for  $i = 1, \dots, n$  and some permutation  $\pi$  such that  $m_{\pi(i)} \geq m_{\pi(i+1)}$  ( $i \in \{1, \dots, n-1\}$ ).

**Lemma 4.14.** Let  $k, n, n' \in \mathbb{N}$  such that  $k \geq n, n'$ . Then for all  $m_1, \dots, m_n \in \mathbb{N}$  and  $c > m_1, \dots, m_n$  we obtain:

- (1)  $\{\{m_1, \dots, m_n\} \succ^{\text{mul}} \{m'_1, \dots, m'_{n'}\}\}$  implies  $\mathfrak{h}_{k,c}^n(m_1, \dots, m_n) > \mathfrak{h}_{k,c}^{n'}(m'_1, \dots, m'_{n'})$ , and
- (2)  $c^k > \mathfrak{h}_{k,c}^n(m_1, \dots, m_n)$ .

Let  $k \in \mathbb{N}$  be fixed and let  $M \subseteq \mathcal{M}(\mathbb{N})$  collect all multisets of size up to  $k$ . By Lemma 4.14 the functions  $\mathfrak{h}_{k,\cdot}^l$  gives an order preserving homomorphism from  $(M, \succ^{\text{mul}})$  to  $(\mathbb{N}, >)$ . Furthermore this homomorphism is polynomially bounded in its elements. We extend this homomorphism to multisets  $\{\{a_1, \dots, a_n\}\}$  over  $\mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$ . Let  $k, n \in \mathbb{N}$  such that  $k \geq n$ . We define  $\mathfrak{G}_k^n : \mathcal{S}^*(\mathcal{F})^n \rightarrow \mathbb{N}$  as follows:  $\mathfrak{G}_k^n(a_1, \dots, a_n) := \mathfrak{h}_{k,c}^n(\mathfrak{G}_k(a_1), \dots, \mathfrak{G}_k(a_n))$ , where  $c = 1 + \max\{\mathfrak{G}_k(a_i) \mid i \in \{1, \dots, n\}\}$ . We obtain:

**Lemma 4.15.** Let  $a_1, \dots, a_n, b_1, \dots, b_m \in \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$  and let  $k \geq m, n$ . Then

$$\{\{a_1, \dots, a_n\} \blacktriangleright_k^{\text{mul}} \{b_1, \dots, b_m\}\} \implies \mathfrak{G}_k^n(a_1, \dots, a_n) > \mathfrak{G}_k^m(b_1, \dots, b_m).$$

In Theorem 4.17 below we prove  $\mathfrak{F}_{k,p}(m) \leq c \cdot (m+2)^d$  for some constants  $c, d \in \mathbb{N}$  depending only on  $k$  and  $p$ . Inevitably the proof of Theorem 4.17 is technical, the reader may skip the formal proof on the first read. In the proof of Theorem 4.17, we instantiate the constants  $c, d$  by parameters  $c_{k,p}, d_{k,p} \in \mathbb{N}$ , which are defined by recursion on  $p$  as follows:

$$d_{k,p} := \begin{cases} k+1 & \text{if } p=0, \\ (d_{k,p-1} \cdot k)^{k+1} + 1 & \text{otherwise;} \end{cases}$$

$$c_{k,p} := \begin{cases} k^k & \text{if } p=0, \\ (c_{k,p-1} \cdot k)^{\sum_{i=1}^k (k \cdot d_{k,p-1})^i} & \text{otherwise.} \end{cases}$$

The theorem is then proven by induction on  $p$  and  $m$ . Consider term  $f(a_1, \dots, a_n)$  with  $k \geq n$  and  $\mathfrak{G}_k^n(a_1, \dots, a_n) \leq m$ . At the heart of the proof, we show that  $c_{k,p} \cdot (m+2)^{d_{k,p}} > \mathfrak{G}_k(b)$  for arbitrary  $b$  with  $f(a_1, \dots, a_n) \blacktriangleright_k b$ . The most involved case is  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(3)} [b_1 \cdots b_o]$  where for all but one  $j \in \{1, \dots, o\}$  we have  $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$ . Here it is important to give a precise analysis of the order  $\succ_{k,l}$ , exploiting the parameters  $k$  and  $l$ . To this avail we define for  $l, k \geq 1$  and  $p \in \mathbb{N}$  a family of auxiliary functions  $g_{k,l,p} : \mathbb{N} \rightarrow \mathbb{N}$  by

$$g_{k,l,p}(m) := \begin{cases} k^l \cdot m^l & \text{if } p=0, \\ m & \text{if } p > 0 \text{ and } l=1, \\ c_{k,p-1} \cdot (m \cdot g_{k,l-1,p}(m))^{k \cdot d_{k,p-1}} & \text{otherwise.} \end{cases}$$

Having as premise the induction hypothesis of the main proof, the next lemma explains the rôle of  $\succ_{k,l}$ .

**Lemma 4.16.** Let  $f(a_1, \dots, a_n) \in \mathcal{S}(\mathcal{F})$ . Let  $k \geq n$  and  $m \geq \mathfrak{G}_k^n(a_1, \dots, a_n)$ . Suppose  $\mathfrak{F}_{k,p}(m') \leq c_{k,p}(m'+2)^{d_{k,p}}$  for all  $p < \text{rk}(f)$  and  $m'$ . Then for all  $b \in \mathcal{S}(\mathcal{F}) \cup \mathcal{S}^*(\mathcal{F})$ ,

$$f(a_1, \dots, a_n) \succ_{k,l} b \implies \mathfrak{G}_k(b) \leq g_{k,l,\text{rk}(f)}(m+2).$$

*Proof.* We prove lemma by induction on  $l$ . The base case  $l=1$  is easy to show, hence assume  $l > 1$ . Suppose  $f(a_1, \dots, a_n) \succ_{k,l} b$ , we continue by case analysis:

**Case**  $f(a_1, \dots, a_n) \succ_{k,l}^{(1)} b$ : Then  $a_i \succ_{k,l} b$  for some  $i \in \{1, \dots, n\}$ , and consequently  $\mathfrak{G}_k(b) \leq \mathfrak{G}_k(a_i)$ . Then by definition and assumption we even have  $\mathfrak{G}_k(a_i) \leq \mathfrak{G}_k^n(a_1, \dots, a_n) \leq m$ .

**Case  $f(a_1, \dots, a_n) \succ_{k,l}^{(2)} b$  where  $b = g(b_1, \dots, b_o)$ :** Then  $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$  for all  $j = 1, \dots, o$ . Set  $m' := \mathbf{G}_k^o(b_1, \dots, b_o)$ . We have

$$\begin{aligned} m' &< \max\{\mathbf{G}_k(b_j) + 1 \mid j \in \{1, \dots, o\}\}^k && \text{by definition and Lemma 4.14 (2)} \\ &\leq (g_{k,l-1,\text{rk}(f)}(m+2) + 1)^k && \text{applying induction hypothesis.} \end{aligned}$$

As in the considered case  $\text{rk}(g) < \text{rk}(f)$  holds, we have  $\mathbf{G}_k(b) \leq \mathbf{F}_{k,\text{rk}(g)}(m')$  and so by assumption and arithmetical reasoning we conclude

$$\begin{aligned} \mathbf{G}_k(b) &\leq c_{k,\text{rk}(g)} \cdot (m' + 2)^{d_{k,\text{rk}(f)}-1} \\ &< c_{k,\text{rk}(f)-1} \cdot ((g_{k,l-1,\text{rk}(f)}(m+2) + 1)^k + 2)^{d_{k,\text{rk}(f)}-1} && \text{substituting bound for } m' \\ &\leq c_{k,\text{rk}(f)-1} \cdot ((m+2) \cdot g_{k,l-1,\text{rk}(f)}(m+2))^{k \cdot d_{k,\text{rk}(f)}-1} \\ &= g_{k,l,\text{rk}(f)}(m+2) && \text{using } \text{rk}(f) > 0. \end{aligned}$$

**Case  $f(a_1, \dots, a_n) \succ_{k,l}^{(3)} b$  where  $b = [b_1 \dots b_o]$ :** Order constraints give  $o \leq \text{wd}(a) + k$  and  $f(a_1, \dots, a_n) \succ_{k,l-1} b_j$  ( $j = 1, \dots, o$ ). Exploiting that  $a_i$  is ground, a standard argument shows that  $\text{wd}(a_i) \leq \mathbf{G}_k(a_i)$ , and consequently  $\text{wd}(a_i) \leq m$ . Thus

$$o \leq \text{wd}(a) + k = \max\{1, \text{wd}(a_1), \dots, \text{wd}(a_n)\} + k \leq m + k \leq k \cdot (m + 1). \quad (4.1)$$

Since by Lemma 4.12 we have  $\mathbf{G}_k(b) = \sum_{i=1}^o \mathbf{G}_k(b_i)$ , using (4.1) we see

$$\begin{aligned} \mathbf{G}_k(b) &\leq k \cdot (m + 1) \cdot g_{k,l-1,\text{rk}(f)}(m + 2) && \text{by induction hypothesis} \\ &< g_{k,l,\text{rk}(f)}(m + 2) && \text{by case analysis.} \end{aligned}$$

□

**Theorem 4.17.** Let  $k \geq 1$  and  $p \in \mathbb{N}$ . There exists constants  $c, d \in \mathbb{N}$  (depending only on  $k$  and  $p$ ) such that for all  $m \in \mathbb{N}$  we have

$$\mathbf{F}_{k,p}(m) \leq c \cdot (m + 2)^d.$$

*Proof.* Fix  $f(a_1, \dots, a_n) \in \mathcal{S}(\mathcal{F})$  such that  $\text{rk}(f) = p$ ,  $k \geq n$  and  $\mathbf{G}_k^n(a_1, \dots, a_n) \leq m$ . To show the theorem, we prove that for all  $b$  with  $f(a_1, \dots, a_n) \blacktriangleright_k b$  we have  $\mathbf{G}_k(b) < c_{k,p} \cdot (m + 2)^{d_{k,p}}$  for constants  $c_{k,p}$  and  $d_{k,p}$  as defined on page 22. The proof is by induction on the lexicographic combination of  $p$  and  $m$ . The base case where  $p = 0$  and  $m = 0$  is easy to proof, we consider the inductive step. Consider the inductive step. By induction hypothesis we have

$$\mathbf{F}_{k,p'}(m') \leq c_{k,p} \cdot (m' + 2)^{d_{k,p'}} \quad \text{if } p' < p, \text{ or } p' = p \text{ and } m' < m.$$

For  $p' < p$  we will use the induction hypothesis as a premise to Lemma 4.16, for  $p' = p$  we use below the consequence

$$\mathbf{G}_k(g(b_1, \dots, b_o)) < c_{k,p} \cdot (m + 1)^{d_{k,p}} \quad \text{if } f \sim g, o \leq k \text{ and } \mathbf{G}_k^o(b_1, \dots, b_o) < m. \quad (4.2)$$

We analyse the cases  $p = 0$  and  $p > 0$  separately. In both cases we perform a side induction on  $l$ .

**Case  $p = 0$ :** By side induction on  $l$  we prove that  $\mathbf{G}_k(b) < k^k \cdot (m + 1)^{k+1} + k^l \cdot (m + 2)^l$  for all  $b$  with  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} b$ .

Note that if  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(1)} b$  holds, as in the proof of Lemma 4.16, we even have  $\mathbf{G}_k(b) \leq \mathbf{G}_k^n(a_1, \dots, a_n) \leq m$ .

Consider now  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(2)} b$  where  $g(b_1, \dots, b_o)$ . The ordering constraints give  $o \leq k$  and  $\{a_1, \dots, a_n\} \blacktriangleright_{k,l}^{\text{mul}} \{b_1, \dots, b_o\}$ , from Lemma 4.15 we thus get  $\mathbf{G}_k^o(b_1, \dots, b_o) < \mathbf{G}_k^n(a_1, \dots, a_n) \leq m$ . Since also  $f \sim g$  in this case we conclude as we even have

$$\begin{aligned} \mathbf{G}_k(g(b_1, \dots, b_o)) &< c_{k,0} \cdot (m+1)^{d_{k,0}} && \text{by main induction hypothesis} \\ &= k^k \cdot (m+1)^{k+1} && \text{by definition of } c_{k,0} \text{ and } d_{k,0}. \end{aligned}$$

Next consider  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(3)}$  where  $[b_1 \cdots b_o]$ . The order constraints give (i)  $a \blacktriangleright_{k,l-1} b_{j_0}$  for some  $j_0 \in \{1, \dots, o\}$ , (ii)  $a \triangleright_{k,l-1} b_j$  for all  $j \neq j_0$ , and (iii)  $o \leq \text{wd}(a) + k$ . We have

$$\begin{aligned} \mathbf{G}_k(b_{j_0}) &< k^k \cdot (m+1)^{k+1} + k^{l-1} \cdot (m+2)^{l-1} && \text{from (i), using SIH on } l \\ \mathbf{G}_k(b_j) &\leq k^{l-1} \cdot (m+2)^{l-1} \text{ for } j \neq j_0 && \text{from (ii), using Lemma 4.16} \\ o &\leq k \cdot (m+1) && \text{from (iii)}. \end{aligned}$$

For the last inequality, compare Equation (4.1) from Lemma 4.16. As  $\mathbf{G}_k(b) = \sum_{j=1}^o \mathbf{G}_k(b_j)$  by Lemma 4.12, substituting the above inequalities we get

$$\begin{aligned} \mathbf{G}_k(b) &< k^k \cdot (m+1)^{k+1} + k^{l-1} \cdot (m+2)^{l-1} && \text{bound on } \mathbf{G}_k(b_{j_0}) \\ &\quad + (k \cdot (m+1) - 1) \cdot k^{l-1} \cdot (m+2)^{l-1} && \text{bound on } o \text{ and } \mathbf{G}_k(b_j), j \neq j_0 \\ &< k^k \cdot (m+1)^{k+1} + k^l \cdot (m+2)^l. \end{aligned}$$

This concludes the final case of the side induction. Since  $\blacktriangleright_k = \blacktriangleright_{k,k}$  this preparatory step gives

$$\mathbf{G}_k(b) < k^k \cdot (m+1)^{k+1} + k^k \cdot (m+2)^k \leq k^k \cdot (m+2)^{k+1},$$

we conclude the case  $p = 0$ .

**Case  $p > 0$ :** We show first that for all  $k \geq l$ , if  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} b$  then

$$\mathbf{G}_k(b) \leq c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^{l+1}}. \quad (4.3)$$

The proof is by induction on  $l$ . Suppose  $f(a_1, \dots, a_n) \blacktriangleright_{k,l} b$ . The base case  $l = 1$  is trivial, so consider the inductive step  $l > 1$ . As in the case  $p = 0$ , if  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(1)} b$  then even  $\mathbf{G}_k(b) \leq m$ , and if  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(2)} b$  then even  $\mathbf{G}_k(b) \leq c_{k,p} \cdot (m+1)^{d_{k,p}}$ . Consider  $f(a_1, \dots, a_n) \blacktriangleright_{k,l}^{(3)} b$ . Then  $b = [b_1 \cdots b_o]$  with (i)  $a \blacktriangleright_{k,l-1} b_{j_0}$  for some  $j_0 \in \{1, \dots, o\}$ , (ii)  $a \triangleright_{k,l-1} b_j$  for all  $j \neq j_0$ , and (iii)  $o \leq \text{wd}(a) + k$ . A standard argument gives

$$g_{k,l,p}(n) \leq c_{k,p-1}^{\sum_{i=1}^{l-1} (k \cdot d_{k,p-1})^i} \cdot n^{\sum_{i=1}^l (k \cdot d_{k,p-1})^i},$$

for all  $n \in \mathbb{N}$ , thus

$$\begin{aligned} \mathbf{G}_k(b_{j_0}) &< c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^l} && \text{from (i), using SIH on } l \\ \mathbf{G}_k(b_j) &\leq c_{k,p-1}^{\sum_{i=1}^{l-2} (k \cdot d_{k,p-1})^i} \cdot (m+2)^{\sum_{i=1}^{l-1} (k \cdot d_{k,p-1})^i} \text{ for } j \neq j_0 && \text{from (ii), using Lemma 4.16} \\ o &\leq k \cdot (m+1) && \text{from (iii)}. \end{aligned}$$



Using Lemma 4.12 and substituting the above inequalities we get

$$\begin{aligned}
\mathbf{G}_k(b) &\leq c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^l} && \text{bound on } \mathbf{G}_k(b_{j_0}) \\
&\quad + k \cdot (m+1) \cdot c_{k,p-1}^{\sum_{i=1}^{l-2} (k \cdot d_{k,p-1})^i} \cdot (m+2)^{\sum_{i=1}^{l-1} (k \cdot d_{k,p-1})^i} && \text{bound on } \mathbf{G}_k(b_j), j \neq j_0 \\
&< c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^l} \\
&\quad + c_{k,p} \cdot (m+1) \cdot (m+2)^{\sum_{i=1}^{l-1} (k \cdot d_{k,p-1})^i} && \text{as } k \cdot c_{k,p-1}^{\sum_{i=1}^{l-2} (k \cdot d_{k,p-1})^i} < c_{k,p} \\
&\leq c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{\sum_{i=0}^l (k \cdot d_{k,p-1})^i} \\
&\leq c_{k,p} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^{l+1}},
\end{aligned}$$

as desired, we conclude Equation (4.3). From this preparatory step,  $\blacktriangleright_k = \blacktriangleright_{k,k}$  and  $(k \cdot d_{k,p-1})^{k+1} < (k \cdot d_{k,p-1})^{k+1} + 1 = d_{k,p}$  we finally get

$$\begin{aligned}
\mathbf{G}_k(b) &\leq c_{k,p+1} \cdot (m+1)^{d_{k,p}} + c_{k,p} \cdot (m+2)^{(k \cdot d_{k,p-1})^{k+1}} \\
&= c_{k,p} \cdot ((m+1)^{d_{k,p}} + (m+2)^{(k \cdot d_{k,p-1})^{k+1}}) < c_{k,p} \cdot (m+2)^{d_{k,p}},
\end{aligned}$$

and conclude also this case.  $\square$

As a consequence, the number of  $\blacktriangleright_k$ -descents on basic terms interpreted with predicative interpretation  $\mathbf{S}$  is polynomial in sum of depths of normal arguments.

**Corollary 4.18.** Let  $k \geq 1$  and consider  $f \in \mathcal{D}$  with  $m \leq k$  normal arguments. There exists a constant  $d \in \mathbb{N}$  depending only on  $k$  and the rank of  $f$  such that:

$$\mathbf{G}_k(\mathbf{S}(f(u_1, \dots, u_m; u_{m+1}, \dots, u_{m+n}))) = \mathcal{O}((\max_{i=1}^m \mathbf{dp}(u_i))^d)$$

for all  $u_1, \dots, u_{m+n} \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ .

*Proof.* Let  $s = f(u_1, \dots, u_m; u_{m+1}, \dots, u_{m+n})$  be as given by the corollary. Recall that since arguments of  $s$  are values, we have  $\mathbf{nm}(u_i) = \mathbf{dp}(u_i)$  as indicated on page 16, and further  $\mathbf{S}(u_i) = []$  holds for all  $i = 1, \dots, m+n$ . Thus

$$\mathbf{S}(s) = [f_n(\overline{\mathbf{dp}(u_1)}, \dots, \overline{\mathbf{dp}(u_m)})].$$

As  $\mathbf{G}_k(\bullet)$  is constant, say  $\mathbf{G}_k(\bullet) = c$ , by Lemma 4.12 we see that  $\mathbf{G}_k(\overline{\mathbf{dp}(u_i)}) = c \cdot \mathbf{dp}(u_i)$ . We conclude as

$$\begin{aligned}
\mathbf{G}_k(\mathbf{S}(s)) &= \mathbf{G}_k(f_n(\overline{\mathbf{dp}(u_1)}, \dots, \overline{\mathbf{dp}(u_m)})) && \text{by Lemma 4.12} \\
&\leq F_{k, \text{rk}(f)}(\mathbf{G}_k^l(\overline{\mathbf{dp}(u_1)}, \dots, \overline{\mathbf{dp}(u_m)})) && \text{by assumption } m \leq k \\
&\leq F_{k, \text{rk}(f)}\left(\left(1 + \max_{i=1}^m \mathbf{G}_k(\overline{\mathbf{dp}(u_i)})\right)^k\right) && \text{by Lemma 4.14 (2)} \\
&\leq F_{k, \text{rk}(f)}\left(\left(c \cdot (1 + \max_{i=1}^m \mathbf{dp}(u_i))\right)^k\right) && \text{using } \mathbf{G}_k(\overline{\mathbf{dp}(u_i)}) \leq c \cdot \mathbf{dp}(u_i) \\
&\in \mathcal{O}\left(\left(c \cdot (1 + \max_{i=1}^m \mathbf{dp}(u_i))\right)^{k+d_{k, \text{rk}(f)}}\right) && \text{by Theorem 4.17} \\
&= \mathcal{O}\left((\max_{i=1}^m \mathbf{dp}(u_i))^{k+d_{k, \text{rk}(f)}}\right). && \square
\end{aligned}$$

## 5. PREDICATIVE EMBEDDING

Fix a predicative recursive TRS  $\mathcal{R}$  and signature  $\mathcal{F}$ , and let  $>_{\text{pop}^*}$  be the polynomial path order underlying  $\mathcal{R}$  based on the (admissible) precedence  $\succ$ . We denote by  $\succ$  also the induced precedence on  $\mathcal{F}_n$  given by:  $f_n \sim g_n$  if  $f \sim g$  and  $f_n \succ g_n$  if  $f \succ g$ . Further, we set  $f \succ \bullet$  for all  $f_n \in \mathcal{F}_n$ . We denote by  $\blacktriangleright_\ell$  (and respectively  $\succ_\ell$ ) the approximation given in Definition 4.7 (respectively Definition 4.5) with underlying precedence  $\succ$ .

In this section, we establish the embedding of  $\dot{\mapsto}_{\mathcal{R}}$  into  $\blacktriangleright_\ell$  as outlined in the proof plan on page 13; in the sequel  $\ell$  is set to twice the maximum size of right-hand sides of  $\mathcal{R}$ . Lemma 5.2 below proves the embedding of root steps for the case  $l >_{\text{pop}^*} r$ . In Lemma 5.3 we then show that the embedding is closed under contexts. The next auxiliary lemma connects the auxiliary orders  $>_{\text{pop}}$  and  $>_{k,l}$  (compare Example 4.6).

**Lemma 5.1.** Suppose  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l}) \in \mathcal{T}_b(\mathcal{F}, \mathcal{V})$ ,  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  and  $\sigma : \mathcal{V} \rightarrow \text{NF}$ . Then for predicative interpretation  $\mathbf{Q} \in \{\mathbf{S}, \mathbf{N}\}$  we have

$$s >_{\text{pop}} t \implies f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \succ_{2 \cdot |t|} \mathbf{Q}(t\sigma).$$

*Proof.* The proof proceeds by induction on the definition of  $>_{\text{pop}}$  and makes use of Lemmas 3.17, 4.9 and 4.10.  $\square$

**Lemma 5.2.** Suppose  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l}) \in \mathcal{T}_b(\mathcal{F}, \mathcal{V})$ ,  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  and  $\sigma : \mathcal{V} \rightarrow \text{NF}$ . Then for predicative interpretation  $\mathbf{Q} \in \{\mathbf{S}, \mathbf{N}\}$  we have

$$s >_{\text{pop}^*} t \implies \mathbf{Q}(s\sigma) \blacktriangleright_{2 \cdot |t|} \mathbf{Q}(t\sigma).$$

*Proof.* Let  $s, t, \sigma$  be as given in the lemma. We prove the stronger assertions

- (1)  $f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \blacktriangleright_{2 \cdot |t|} \mathbf{S}(t\sigma)$ ,
- (2)  $f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \succ_{2 \cdot |t|} \mathbf{S}(t\sigma)$  if  $t \in \mathcal{T}(\mathcal{F}^{<f}, \mathcal{V})$ , and
- (3)  $f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \# \overline{\text{nm}(s\sigma)} \blacktriangleright_{2 \cdot |t|} \mathbf{N}(t\sigma)$ .

As  $\mathbf{S}(s) = [f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma))]$ , Property 1 and Lemma 4.10 (3) yield  $\mathbf{S}(s\sigma) \blacktriangleright_{2 \cdot |t|} \mathbf{S}(s\sigma)$ . Furthermore  $\mathbf{N}(s) = \mathbf{S}(f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma))) \# \overline{\text{nm}(s\sigma)}$ . Hence Property 3 immediately yields  $\mathbf{N}(s\sigma) \blacktriangleright_{2 \cdot |t|} \mathbf{N}(s\sigma)$ .

We continue with the proof of the assertions by induction on  $>_{\text{pop}^*}$  and set  $u := f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma))$ .

**Case  $s \stackrel{(1)}{>}_{\text{pop}^*} t$ :** Due to Lemma 3.17 (employing  $>_{\text{pop}} \subseteq >_{\text{pop}^*}$ ) we obtain that  $t\sigma$  is a safe subterm of  $s_i\sigma$  and  $t\sigma \in \text{NF}$ . The latter implies  $\mathbf{S}(t\sigma) = []$  and thus Properties 1 and 2 follow. For Property 3, observe that  $\text{len}(\overline{\text{nm}(t\sigma)}) = \text{nm}(t\sigma) \leq \text{nm}(s_i\sigma) \leq \text{wd}(u \# \overline{\text{nm}(s\sigma)})$ . Here the last inequality follows by a simple case distinction on  $i$ . From this and  $u \stackrel{(2)}{\succ}_{2 \cdot |t| - 1} \bullet$  we get

$$u \# \overline{\text{nm}(s\sigma)} \blacktriangleright_{2 \cdot |t|}^{(4)} \overline{\text{nm}(t\sigma)} = \mathbf{N}(t\sigma).$$

**Case  $s \stackrel{(2)}{>}_{\text{pop}^*} t$ :** The assumption gives  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \succ g$  and further  $s >_{\text{pop}} t_i$  for all normal argument positions  $i = 1, \dots, m$ , and  $s >_{\text{pop}^*} t_i$  for all safe argument positions  $i = m+1, \dots, m+n$ , of  $g$ . Additionally  $t_{i_0} \notin \mathcal{T}(\mathcal{F}^{<f}, \mathcal{V})$  for at most one argument position  $i_0$ . Set  $v := g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma))$  and let  $\mathbf{S}(t_i\sigma) = [v_{i,1} \cdots v_{i,j_i}]$  for all safe argument positions  $i = m+1, \dots, m+n$ . Hence, we obtain:

$$\mathbf{S}(t\sigma) = [g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)) \ v_{m+1,1} \cdots v_{m+1,j_{m+1}} \cdots v_{m+n,1} \cdots v_{m+n,j_{m+n}}].$$

Applying Lemma 5.1 on all normal arguments of  $t$ , we see

$$u \triangleright_{2 \cdot |t| - 1} g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)) = v, \quad (5.1)$$

from the assumptions  $f_n \succ g_n$  and  $s \succ_{\text{pop}} t_i$  for all  $i = 1, \dots, m$ . Since  $s \succ_{\text{pop}^*} t_{i_0}$  by assumption, induction hypothesis on  $i_0$  gives  $u \blacktriangleright_{2 \cdot |t_{i_0}|} \mathbf{S}(t_{i_0}\sigma) = [v_{i_0,1}, \dots, v_{i_0,j_{i_0}}]$ . We obtain:

$$u \blacktriangleright_{2 \cdot |t| - 1} v_{i_0, j_0} \quad \text{for some } j_0 \in \{1, \dots, j_{i_0}\} \quad (5.2)$$

$$v \triangleright_{2 \cdot |t| - 1} v_{i_0, j} \quad \text{for all } j = 1, \dots, j_{i_0}, j \neq j_0. \quad (5.3)$$

Induction hypothesis on safe argument positions  $i$  gives:

$$u \triangleright_{2 \cdot |t| - 1} v_{i, j} \quad \text{for all } i = m + 1, \dots, m + n, i \neq i_0 \text{ and } j = 1, \dots, j_i. \quad (5.4)$$

Due to Lemma 4.9 (1),  $\text{len}(\mathbf{S}(t\sigma)) \leq |t|$ . Hence property 1 follows by  $\blacktriangleright_{2 \cdot |t|}^{(3)}$  using equations (5.1)–(5.4). Likewise, Property 3 follows by an additional use of  $u \triangleright_{2 \cdot |t| - 1}^{(2)}$  and

$$\begin{aligned} \text{len}(\mathbf{N}(t\sigma)) &\leq 2 \cdot |t| + \max\{\text{nm}(s_1\sigma), \dots, \text{nm}(s_{k+l}\sigma)\} \\ &\leq 2 \cdot |t| + \text{wd}(f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \# \overline{\text{nm}(s\sigma)}). \end{aligned}$$

Here the first inequality follows by Lemma 4.9 (3). For Property 2 we proceed as above, but strengthen inequality (5.2) to  $u \triangleright_{2 \cdot |t| - 1} v_{i_0, j_0}$ .

**Case  $s \succ_{\text{pop}^*}^{(3)} t$ :** Then  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \sim g$ . Further, the assumption gives  $\{\{s_1, \dots, s_k\}\} \succ_{\text{pop}^*}^{\text{mul}} \{\{t_1, \dots, t_m\}\}$  and  $\{\{s_{k+1}, \dots, s_{k+l}\}\} \geq_{\text{pop}^*}^{\text{mul}} \{\{t_{m+1}, \dots, t_{m+n}\}\}$ . Hence  $t \notin \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$  and Property 2 is vacuously satisfied. We prove Properties 1 and 3. Using  $s_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  for all normal argument positions  $i = 1, \dots, m$  and employing Lemma 3.17 we see that  $\{\{s_1, \dots, s_k\}\} \succ_{\text{pop}^*}^{\text{mul}} \{\{t_1, \dots, t_m\}\}$  implies

$$\{\{\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)\}\} \blacktriangleright_{2 \cdot |t| - 1}^{\text{mul}} \{\{\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)\}\}.$$

Hence due to  $f_n \sim g_n$  and  $m \leq |t| \leq 2 \cdot |t| - 1$  we obtain:

$$f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \blacktriangleright_{2 \cdot |t| - 1}^{(2)} g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)). \quad (5.5)$$

Assumption  $\{\{s_{k+1}, \dots, s_{k+l}\}\} \geq_{\text{pop}^*}^{\text{mul}} \{\{t_{m+1}, \dots, t_{m+n}\}\}$  together with  $s_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$  for all  $i = k + 1, \dots, k + l$  gives  $t_j \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ . As a consequence we have  $\mathbf{S}(t_j\sigma) = []$  for all  $j = m + 1, \dots, m + n$  and we obtain:

$$f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \blacktriangleright_{2 \cdot |t|}^{(3)} [g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma))] = \mathbf{S}(t\sigma)$$

which concludes the argument for property 1. For property 3, we see that the order constraints on safe arguments imply  $\text{nm}(s\sigma) \geq \text{nm}(t\sigma)$ . Thus  $\overline{\text{nm}(s\sigma)} \blacktriangleright_{2 \cdot |t|} \overline{\text{nm}(t\sigma)}$ , using this and Equation (5.5) we obtain

$$f_n(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \# \overline{\text{nm}(s\sigma)} \blacktriangleright_{2 \cdot |t|}^{(4)} g_n(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)) \# \overline{\text{nm}(t\sigma)} = \mathbf{N}(t\sigma),$$

by Lemma 4.10 (1) and Lemma 4.10 (4).  $\square$

**Lemma 5.3.** Let  $\ell \geq \max\{\text{ar}(f_n) \mid f_n \in \mathcal{F}_n\}$  and  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ . Then for  $Q \in \{\mathbf{N}, \mathbf{S}\}$ ,

$$Q(s) \blacktriangleright_\ell Q(t) \implies Q(C[s]) \blacktriangleright_\ell Q(C[t]) .$$

*Proof.* We proceed by induction on the context  $C$ . It suffices to consider the inductive step. Consider terms  $s = f(s_1, \dots, s_i, \dots, s_{k+l})$  and  $t = f(s_1, \dots, t_i, \dots, s_{k+l})$ . We restrict our attention the predicative interpretation  $\mathbf{N}$  and show  $\mathbf{N}(f(s_1, \dots, s_i, \dots, s_{k+l})) \blacktriangleright_\ell \mathbf{N}(f(s_1, \dots, t_i, \dots, s_{k+l}))$ , whenever  $\mathbf{N}(s_i) \blacktriangleright_\ell \mathbf{N}(t_i)$ .

Recall that  $\mathbf{N}(s) = \mathbf{S}(s) \# \overline{\text{nm}(s)}$  and  $\mathbf{N}(t) = \mathbf{S}(t) \# \overline{\text{nm}(t)}$ . If  $\text{nm}(s) \geq \text{nm}(t)$  then  $\mathbf{N}(s) \blacktriangleright_\ell \mathbf{N}(t)$  follows from  $\mathbf{S}(s) \blacktriangleright_\ell \mathbf{S}(t)$  and Lemma 4.10 (4). Hence suppose  $\text{nm}(s) < \text{nm}(t)$ . We consider only the case where  $t \notin \mathbf{NF}$ . The assumption  $\text{nm}(s) < \text{nm}(t)$  implies that  $i$  is a safe argument position of  $f$ . Hence we obtain:

$$\mathbf{N}(s) = [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(s_k))] \# \mathbf{S}(s_{k+1}) \# \dots \# \mathbf{S}(s_i) \# \dots \# \mathbf{S}(s_{k+l}) \# \overline{\text{nm}(s)}$$

and

$$\mathbf{N}(t) = [f_n(\mathbf{N}(s_1), \dots, \mathbf{N}(s_k))] \# \mathbf{S}(s_{k+1}) \# \dots \# \mathbf{S}(t_i) \# \dots \# \mathbf{S}(s_{k+l}) \# \overline{\text{nm}(t)} .$$

By definition we have  $\text{nm}(s_i) < \text{nm}(s)$ . This together with  $\text{nm}(s) < \text{nm}(t)$  yields  $\text{nm}(t) = \text{nm}(t_i) + 1$  by the shape of  $s$  and  $t$ . Using Lemma 4.10 (4) and the assumption  $\mathbf{N}(s_i) \blacktriangleright_\ell \mathbf{N}(t_i)$  we obtain:

$$\mathbf{S}(s_i) \# \overline{\text{nm}(s_i)} \# \bullet \blacktriangleright_\ell \mathbf{S}(t_i) \# \overline{\text{nm}(t_i)} \# \bullet .$$

From this we have  $\mathbf{S}(s_i) \# \overline{\text{nm}(s)} \blacktriangleright_\ell \mathbf{S}(t_i) \# \overline{\text{nm}(t)}$  and thus due to Lemma 4.10 (2) and Lemma 4.10 (4) we obtain  $\mathbf{N}(s) \blacktriangleright_\ell \mathbf{N}(t)$ .  $\square$

We have established our first main result.

*Proof of Theorem 3.11.* Let  $\mathcal{R}$  be a predicative recursive TRS and fix an arbitrary basic term  $s = f(u_1, \dots, u_m; u_{m+1}, \dots, u_{m+n})$ . Set the parameter  $\ell$  as follows:

$$\ell := \max\{\text{ar}(f_n) \mid f_n \in \mathcal{F}_n\} \cup \{2 \cdot |r| \mid l \rightarrow r \in \mathcal{R}\} .$$

As  $\mathcal{F}_n$  and  $\mathcal{R}$  are finite,  $\ell$  is well-defined. Consider a maximal  $\mathcal{R}$ -derivation

$$s \xrightarrow{i}_{\mathcal{R}} s_1 \xrightarrow{i}_{\mathcal{R}} s_2 \xrightarrow{i}_{\mathcal{R}} \dots \xrightarrow{i}_{\mathcal{R}} s_k ,$$

starting from an arbitrary term  $s$ , that is,  $k = \text{dh}_{\mathcal{R}}(s)$ . Using Lemma 5.2 together with Lemma 5.3  $k$ -times we get

$$\mathbf{S}(s) \blacktriangleright_\ell \mathbf{S}(s_1) \blacktriangleright_\ell \mathbf{S}(s_2) \blacktriangleright_\ell \dots \blacktriangleright_\ell \mathbf{S}(s_k) .$$

As a consequence, we have  $k \leq G_\ell(\mathbf{S}(s))$  by definition of  $G_\ell$  and thus:

$$\text{dh}(s, \xrightarrow{i}_{\mathcal{R}}) \leq G_\ell(\mathbf{S}(s)) = O\left(\left(\max_{i=1}^m \text{dp}(u_i)\right)^d\right) ,$$

where the asymptotic estimation follows by Corollary 4.18. Note that the degree  $d$  depends only on  $\ell$ .  $\square$

## 6. AN ORDER-THEORETIC CHARACTERISATION OF THE POLYTIME FUNCTIONS

We now present the application of polynomial path orders in the context of *implicit computational complexity*. As by-product of Proposition 2.4 and Theorem 3.11 we immediately obtain that  $\text{POP}^*$  is *sound* for FNP respectively FP.

**Theorem 6.1.** Let  $\mathcal{R}$  be a predicative recursive (constructor) TRS. For every relation  $\llbracket f \rrbracket$  defined by  $\mathcal{R}$ , the function problem  $F_f$  associated with  $\llbracket f \rrbracket$  is in FNP. Moreover, if  $\mathcal{R}$  is confluent then  $\llbracket f \rrbracket \in \text{FP}$ .

Although it is decidable whether a TRS  $\mathcal{R}$  is predicative recursive (we present a sound and complete automation in Section 9), confluence is undecidable in general. To get a decidable result for FP, one can replace confluence by an decidable criteria, for instance orthogonality.

We will now also establish that  $\text{POP}^*$  is *complete* for FP, that is, every function  $f \in \text{FP}$  is computed by some confluent (even orthogonal) predicative recursive TRS. For this we employ Beckmann and Weiermann's *term rewriting characterisation* of the Bellantoni and Cook's class  $\mathcal{B}$ .

**Definition 6.2.** [16, Definition 2.2] For each  $k, l \in \mathbb{N}$  the set of function symbols  $\mathcal{F}_{\mathcal{B}}^{k,l}$  with  $k$  normal and  $l$  safe argument positions is the least set of function symbols such that

- (1)  $\epsilon \in \mathcal{F}_{\mathcal{B}}^{0,0}$ ,  $S_1, S_2 \in \mathcal{F}_{\mathcal{B}}^{0,1}$ ,  $P \in \mathcal{F}_{\mathcal{B}}^{0,1}$ ,  $C \in \mathcal{F}_{\mathcal{B}}^{0,4}$  and  $I_j^{k,l}, O^{k,l} \in \mathcal{F}_{\mathcal{B}}^{k,l}$ , where  $j = 1, \dots, k+l$ ;
- (2) if  $\vec{r} = r_1, \dots, r_m \in \mathcal{F}_{\mathcal{B}}^{k,0}$ ,  $\vec{s} = s_1, \dots, s_n \in \mathcal{F}_{\mathcal{B}}^{k,l}$  and  $h \in \mathcal{F}_{\mathcal{B}}^{m,n}$  then  $\text{SC}[h, \vec{r}, \vec{s}] \in \mathcal{F}_{\mathcal{B}}^{k,l}$ ;
- (3) if  $g \in \mathcal{F}_{\mathcal{B}}^{k,l}$  and  $h_1, h_2 \in \mathcal{F}_{\mathcal{B}}^{k+1,l+1}$  then  $\text{SRN}[g, h_1, h_2] \in \mathcal{F}_{\mathcal{B}}^{k+1,l}$ ;

The *predicative signature* is given by  $\mathcal{F}_{\mathcal{B}} := \bigcup_{k,l \in \mathbb{N}} \mathcal{F}_{\mathcal{B}}^{k,l}$ . Only the constant  $\epsilon$  and *dyadic successors*  $S_1, S_2$ , which serve the purpose of encoding natural numbers in binary, are constructors. The remaining symbols from  $\mathcal{F}_{\mathcal{B}}$  are defined symbols.

In Figure 3 we recall from [16, Definition 2.7] the (infinite) schema of rewrite rules  $R_{\mathcal{B}}$  that form a term rewriting characterisation of the class  $\mathcal{B}$ . Here we let  $k, l$  range over  $\mathbb{N}$  and set  $\vec{x} := x_1, \dots, x_k$  and  $\vec{y} := y_1, \dots, y_l$  for  $k$  respectively  $l$  distinct variables.

**Remark.** We emphasise that the system  $R_{\mathcal{B}}$  is called *infeasible* in [16]. Indeed  $R_{\mathcal{B}}$  admits an exponential lower bound on the derivation height if one considers full rewriting. This is induced by duplicating redexes as explained already in Example 2.5 on page 9. However, this should rather be understood as a miss-configuration of the evaluation strategy, rather than a defect of the rewrite system. Indeed, in our completeness argument below, we exploit that  $R_{\mathcal{B}}$  is predicative recursive, thus the *innermost* runtime complexity is polynomial, as expected.

We emphasise that the above rules are all orthogonal and the following proposition verifies that  $R_{\mathcal{B}}$  generates only polytime computable functions.

**Proposition 6.3.** [16, Lemma 5.2] Let  $f \in \text{FP}$ . There exists a finite restriction  $\mathcal{R}_f \subsetneq R_{\mathcal{B}}$  such that  $\mathcal{R}_f$  computes  $f$ .

We arrive at our completeness result.

**Theorem 6.4.** For every  $f \in \text{FP}$  there exists a finite, orthogonal, and predicative recursive (constructor) TRS  $\mathcal{R}_f$  that computes  $f$ .

### Initial Functions

$$\begin{aligned}
& P(; \epsilon) \rightarrow \epsilon \\
& P(; S_i(; x)) \rightarrow x && \text{for } i = 1, 2 \\
& I_j^{k,l}(\vec{x}; \vec{y}) \rightarrow x_j && \text{for all } j = 1, \dots, k \\
& I_j^{k,l}(\vec{x}; \vec{y}) \rightarrow y_{j-k} && \text{for all } j = k + 1, \dots, l + k \\
& C(; \epsilon, y, z_1, z_2) \rightarrow y \\
& C(; S_i(; x), y, z_1, z_2) \rightarrow z_i && \text{for } i = 1, 2 \\
& O(\vec{x}; \vec{y}) \rightarrow \epsilon
\end{aligned}$$

### Safe Composition (SC)

$$SC[h, \vec{r}, \vec{s}](\vec{x}; \vec{y}) \rightarrow h(\vec{r}(\vec{x}; ); \vec{s}(\vec{x}; \vec{y}))$$

### Safe Recursion on Notation (SRN)

$$\begin{aligned}
& SRN[g, h_1, h_2](\epsilon, \vec{x}; \vec{y}) \rightarrow g(\vec{x}; \vec{y}) \\
& SRN[g, h_1, h_2](S_i(; z), \vec{x}; \vec{y}) \rightarrow h_i(z, \vec{x}; \vec{y}, SRN[g, h_1, h_2](z, \vec{x}; \vec{y})) && \text{for } i = 1, 2
\end{aligned}$$

Figure 3: Term Rewriting Characterisation of the Class  $\mathcal{B}$

*Proof.* Take the finite TRS  $\mathcal{R}_f \subsetneq R_{\mathcal{B}}$  from Proposition 6.3 that computes  $f$ . Obviously  $\mathcal{R}_f$  is orthogonal hence confluent.

It remains to verify that  $\mathcal{R}_f$  is compatible with some instance  $>_{\text{pop}^*}$ . To define  $>_{\text{pop}^*}$  we use the separation of normal from safe argument positions as indicated in the rules. To define the precedence underlying  $>_{\text{pop}^*}$ , we define a mapping  $\text{lh}$  from the signature of  $\mathcal{F}_{\mathcal{B}}$  into the natural numbers as follows:

- $\text{lh}(f) := 0$  if  $f$  is one of  $\epsilon, S_0, S_1, C, P, I_j^{k,l}$  or  $O^{k,l}$ ;
- $\text{lh}(SC[h, \vec{r}, \vec{s}]) := 1 + \text{lh}(h) + \sum_{r \in \vec{r}} \text{lh}(r) + \sum_{s \in \vec{s}} \text{lh}(s)$ ;
- $\text{lh}(SRN[g, h_1, h_2]) := 1 + \text{lh}(g) + \text{lh}(h_1) + \text{lh}(h_2)$ .

Finally for each pair of function symbol  $f$  and  $g$  occurring in  $\mathcal{R}_f$ , we set  $f \succ g$  if  $\text{lh}(f) > \text{lh}(g)$ . Then  $\succ$  defines an admissible precedence.

It is straight forward to verify that  $\mathcal{R}_f \subseteq >_{\text{pop}^*}$  where  $>_{\text{pop}^*}$  is based on the precedence  $\succ$  and the safe mapping as indicated in Definition 6.2.  $\square$

Observe that compatibility of  $R_{\mathcal{B}}$  with  $\text{POP}^*$  together with Theorem 3.11 yields a strengthened version of Theorem 4.3 in [16], as due to our result the innermost derivation height is polynomially bounded in the depth of the normal arguments only. The latter result can be obtained directly, by a simplification of the semantic argument given in [16, Section 4], see [3].

By Theorem 6.1 and Theorem 6.4 we obtain a precise characterisation of the class of polytime computable functions and thus arrive at the second main result of the paper.

**Corollary 6.5.** The following class of functions are equivalent:

- (1) The class of functions computed by confluent predicative recursive (constructor) TRSs.
- (2) The class of polytime computable functions FP.

We note that it is not decidable whether a rewrite system is confluent. However, to get a decidable characterisation we could replace confluence by orthogonality, compare Theorem 6.4.

## 7. A NON-TRIVIAL CLOSURE PROPERTY OF THE POLYTIME COMPUTABLE FUNCTIONS

Bellantoni [17] already observed that the class  $\mathcal{B}$  is closed under *predicative recursion on notation with parameter substitution* (scheme (SRN<sub>PS</sub>)). Essentially this recursion scheme allows substitution on *safe* argument positions. More precise, a new function  $f$  is defined by the equations

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(2z + i, \vec{x}; \vec{y}) &= h_i(z, \vec{x}; \vec{y}, f(z, \vec{x}; \vec{p}(\vec{x}; \vec{y}))) \quad i \in \{1, 2\}. \end{aligned} \tag{SRN<sub>PS</sub>}$$

Bellantoni's result has been reobtained by Beckmann and Weiermann [16, Corollary 5.4] employing a similar term rewriting characterisation. In this section, we introduce the *polynomial path order with parameter substitution* (POP<sub>PS</sub><sup>\*</sup> for short). POP<sub>PS</sub><sup>\*</sup> provides an order-theoretic characterisation of predicative recursion with parameter substitution, that again precisely captures the class FP. Furthermore POP<sub>PS</sub><sup>\*</sup> induces polynomial innermost runtime complexity. As a consequence, we obtain yet another proof of Bellantoni's result.

The next definition introduces POP<sub>PS</sub><sup>\*</sup>. It is a variant of POP<sup>\*</sup>, where clause  $>_{\text{pop}^*}^{(3)}$  has been modified and allows computation at safe argument positions.

**Definition 7.1.** Let  $\succcurlyeq$  denote a precedence. Consider terms  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  such that  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$ . Then  $s >_{\text{pop}_{\text{ps}}^*} t$  if one of the following alternatives holds:

- (1)  $s_i \geq_{\text{pop}_{\text{ps}}^*} t$  for some  $i \in \{1, \dots, k+l\}$ , or
- (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 conditions hold:
  - $s >_{\text{pop}} t_j$  for all normal argument positions  $j = 1, \dots, m$ ;
  - $s >_{\text{pop}_{\text{ps}}^*} t_j$  for all safe argument positions  $j = m+1, \dots, m+n$ ;
  - $t_j \notin \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$  for at most one safe argument position  $j \in \{m+1, \dots, m+n\}$ ;
- (3)  $f \in \mathcal{D}$ ,  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \sim g$  and the following conditions hold:
  - $\{\{s_1, \dots, s_k\}\} >_{\text{pop}_{\text{ps}}^*}^{\text{mul}} \{\{t_1, \dots, t_m\}\}$ ;
  - $s >_{\text{pop}_{\text{ps}}^*} t_j$  and  $t_j \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$  for all safe argument positions  $j = m+1, \dots, m+n$ .

Here  $\geq_{\text{pop}_{\text{ps}}^*} := >_{\text{pop}_{\text{ps}}^*} \cup \dot{\sim}$ .

The next lemma shows that POP<sub>PS</sub><sup>\*</sup> extends the analytic power of POP<sup>\*</sup>.

**Lemma 7.2.** For any underlying admissible precedence  $\succcurlyeq$ ,  $>_{\text{pop}^*} \subseteq >_{\text{pop}_{\text{ps}}^*}$ .

Note that POP<sub>PS</sub><sup>\*</sup> is strictly more powerful than POP<sup>\*</sup>, as witnessed by the following example.

**Example 7.3.** Consider the constructor TRS  $\mathcal{R}_{\text{rev}}$  defining the reversal of lists in a tail recursive fashion:

$$\begin{aligned} 53: \text{rev}_{\text{tl}}([\ ]; ys) &\rightarrow ys & 54: \text{rev}_{\text{tl}}(\text{cons}(x, xs); ys) &\rightarrow \text{rev}_{\text{tl}}(xs; \text{cons}(x, ys)) \\ 55: \text{rev}(xs; ) &\rightarrow \text{rev}_{\text{tl}}(xs; \text{nil}) . \end{aligned}$$

It is not difficult to see that  $\mathcal{R}_{\text{rev}}$  is compatible with POP<sub>PS</sub><sup>\*</sup>, if we use the precedence  $\text{rev} \succ \text{rev}_{\text{tl}} \succ \text{nil} \sim \text{cons}$ . Note that orientation of rule (54) breaks down to  $\text{cons}(x, xs) >_{\text{pop}_{\text{ps}}^*} xs$

and  $\text{rev}_{\text{tl}}(\text{cons}(x, xs); ys) >_{\text{pop}_{\text{ps}}^*} \text{cons}(x, ys)$ . On the other hand,  $>_{\text{pop}^*}$  fails as the corresponding clause  $>_{\text{pop}^*}^{(3)}$  requires  $ys \geq_{\text{pop}^*} \text{cons}(x, ys)$ .

Due to Lemma 7.2,  $\text{POP}_{\text{PS}}^*$  is complete for the class of polytime computable functions. To show that it is sound, we prove that  $\text{POP}_{\text{PS}}^*$  induces polynomially bounded runtime complexity in the sense of Theorem 3.11. The crucial observation is that the embedding of  $\xrightarrow{\mathcal{R}}$  into  $\blacktriangleright_k$  does not break if we relax compatibility constraints to  $\mathcal{R} \subseteq >_{\text{pop}_{\text{ps}}^*}$ .

**Lemma 7.4.** Suppose  $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l}) \in \mathcal{T}_{\text{b}}$ ,  $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$  and  $\sigma : \mathcal{V} \rightarrow \mathcal{T}(\mathcal{C}, \mathcal{V})$ . Then for predicative interpretation  $\mathbf{Q} \in \{\mathbf{S}, \mathbf{N}\}$  we have

$$s >_{\text{pop}_{\text{ps}}^*} t \implies \mathbf{Q}(s\sigma) \blacktriangleright_{2 \cdot |t|} \mathbf{Q}(t\sigma).$$

*Proof.* First one verifies that Lemma 4.9 holds even if we replace  $>_{\text{pop}^*}$  by  $>_{\text{pop}_{\text{ps}}^*}$ . In particular, the assumptions give

$$\text{len}(\mathbf{N}(t\sigma)) \leq 2 \cdot |t| + \max\{\text{nm}(s_1\sigma), \dots, \text{nm}(s_{k+l}\sigma)\}. \quad (7.1)$$

The proof follows the pattern of the proof of Lemma 5.2, i.e., we proceed by induction on  $>_{\text{pop}_{\text{ps}}^*}$ .

We cover only the new case  $s >_{\text{pop}_{\text{ps}}^*}^{(3)} t$ . Let  $s, t$ , and  $\sigma$  be as given in the lemma. Then  $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$  where  $f \sim g$ . Further, the assumption gives  $\{\{s_1, \dots, s_k\}\} >_{\text{pop}^*}^{\text{mul}} \{\{t_1, \dots, t_m\}\}$ . As  $t \notin \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$  it suffices to verify Property 1 and Property 3 from Lemma 5.2. As before, we obtain:

$$f_{\text{n}}(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \blacktriangleright_{2 \cdot |t| - 1}^{(2)} g_{\text{n}}(\mathbf{N}(t_1\sigma), \dots, \mathbf{N}(t_m\sigma)). \quad (7.2)$$

By assumption  $s >_{\text{pop}_{\text{ps}}^*} t_j$  and  $t_j \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$ , induction hypothesis gives

$$f_{\text{n}}(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \geq_{2 \cdot |t| - 1} \mathbf{S}(t_j\sigma). \quad (7.3)$$

As  $\text{len}(\mathbf{S}(t\sigma)) \leq |t|$  by Lemma 4.9 (1), we obtain  $f_{\text{n}}(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \blacktriangleright_{2 \cdot |t|}^{(3)} \mathbf{S}(t\sigma)$  from equations (7.2) and (7.3). Likewise, from this assertion 3 follows by  $\blacktriangleright_k^{(4)}$  using additionally  $f_{\text{n}}(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \geq_{2 \cdot |t| - 1}^{(2)} \bullet$  and

$$\begin{aligned} \text{len}(\mathbf{N}(t\sigma)) &\leq 2 \cdot |t| + \max\{\text{nm}(s_1\sigma), \dots, \text{nm}(s_{k+l}\sigma)\} && \text{by Equation (7.1)} \\ &\leq 2 \cdot |t| + \text{wd}(f_{\text{n}}(\mathbf{N}(s_1\sigma), \dots, \mathbf{N}(s_k\sigma)) \overline{+ \text{nm}(s\sigma)}). && \square \end{aligned}$$

Following the pattern of the proof of Theorem 3.11, replacing the use of Lemma 5.2 by Lemma 7.4 we obtain:

**Theorem 7.5.** Let  $\mathcal{R}$  be a constructor TRS compatible with an instance of  $\text{POP}_{\text{PS}}^*$ . Then the innermost derivation height of any basic term  $f(\vec{u}; \vec{v})$  is bounded by a polynomial in the maximal depth of normal arguments  $\vec{u}$ . The polynomial depends only on  $\mathcal{R}$  and the signature  $\mathcal{F}$ .

As a corollary we get the following variant of Theorem 5.3 in [16].

**Corollary 7.6.** Let  $\mathcal{R}$  be the rewrite system based on the defining equations from Figure 3 and the Schema ( $\text{SRN}_{\text{PS}}$ ). Then the innermost derivation height of any basic term  $f(\vec{u}; \vec{v})$  is bounded by a polynomial in the maximal depth of normal arguments  $\vec{u}$ .

*Proof.* By construction there exists an instance  $>_{\text{pop}_{\text{ps}}^*}$  such that  $\mathcal{R} \subseteq >_{\text{pop}_{\text{ps}}^*}$ . Thus by the theorem, the result follows.  $\square$



Applying Proposition 2.4 yields that predicative recursion is closed under parameter substitution. We can even show a stronger result from Theorem 7.5.

**Corollary 7.7.** Let  $\mathcal{R}$  be a constructor TRS compatible with an instance of  $\text{POP}_{\text{PS}}^*$ . For every relation  $\llbracket f \rrbracket$  defined by  $\mathcal{R}$ , the function problem  $F_f$  associated with  $\llbracket f \rrbracket$  is in FNP. Moreover, if  $\mathcal{R}$  is confluent then  $\llbracket f \rrbracket \in \text{FP}$ .

By Lemma 7.2, parameter substitution extends the power of  $\text{POP}^*$ , together with Theorem 6.4 that shows completeness of  $\text{POP}^*$ , this shows completeness of  $\text{POP}_{\text{PS}}^*$ . We obtain our third result.<sup>4</sup>

**Corollary 7.8.** The following class of functions are equivalent:

- (1) The class of functions computed by confluent constructor TRS compatible with an instance of  $\text{POP}_{\text{PS}}^*$ .
- (2) The class of polytime computable functions FP.

## 8. AUTOMATION OF POLYNOMIAL PATH ORDERS

In this section we present an automation of polynomial path orders, for brevity we restrict our efforts to the order  $>_{\text{pop}^*}$ . Consider a constructor TRS  $\mathcal{R}$ . Checking whether  $\mathcal{R}$  is predicative recursive is equivalent to guessing a precedence  $\succ$  and partitioning of argument positions so that  $\mathcal{R} \subseteq >_{\text{pop}^*}$  holds for the induces order  $>_{\text{pop}^*}$ . As standard for recursive path orders [51, 58], this search can be automated by encoding the constraints imposed by Definition 3.5 into *propositional logic*. To simplify the presentation, we extend the language of propositional logic with truth-constants  $\top$  and  $\perp$  in the obvious way. In the constraint presented below we employ the following atoms.

**8.1. Propositional Atoms.** To encode the separation of normal from safe arguments, we introduce for  $f \in \mathcal{D}$  and  $i = 1, \dots, \text{ar}(f)$  the atoms  $\text{safe}_{f,i}$  so that  $\text{safe}_{f,i}$  represents the assertion that the  $i^{\text{th}}$  argument position of  $f$  is safe. Further we set  $\text{safe}_{f,i} := \top$  for  $n$ -ary  $f \in \mathcal{C}$  and  $i = 1, \dots, n$ , reflecting that argument positions of constructors are always safe.

Since  $\text{POP}^*$  is blind on constructors, predicative recursive TRSs are even compatible with  $>_{\text{pop}^*}$  as induced by an admissible precedence where constructors are equivalent and minimal. For each pair of symbols  $f, g \in \mathcal{D}$ , we introduce propositional atoms  $\succ_{f,g}$  and  $\sim_{f,g}$  so that  $\succ_{f,g}$  represents the assertion  $f \succ g$ , and likewise  $\sim_{f,g}$  represents the assertion  $f \sim g$ . Overall we define for function symbols  $f$  and  $g$  the propositional formulas

$$\lceil f \succ g \rceil := \begin{cases} \top & \text{if } f \in \mathcal{D} \text{ and } g \in \mathcal{C}, \\ \perp & \text{if } f \in \mathcal{C} \text{ and } g \in \mathcal{C}, \\ \succ_{f,g} & \text{otherwise.} \end{cases} \quad \lceil f \sim g \rceil := \begin{cases} \top & \text{if } f \in \mathcal{C} \text{ and } g \in \mathcal{C}, \\ \perp & \text{if } f \in \mathcal{D} \text{ and } g \in \mathcal{D}, \\ \sim_{f,g} & \text{otherwise.} \end{cases}$$

To ensure that the variables  $\succ_{f,g}$  and respectively  $\sim_{f,g}$  encode a preorder on  $\mathcal{D}$  we encode an order preserving homomorphism into the natural order  $>$  on  $\mathbb{N}$ . To this extent, to each  $f \in \mathcal{D}$  we associate a natural number  $\text{rk}_f$  encoded as binary string with  $\lceil \log_2(|\mathcal{D}|) \rceil$  bits. It is straight forward to define Boolean formulas  $\lceil \text{rk}_f > \text{rk}_g \rceil$  (respectively  $\lceil \text{rk}_f = \text{rk}_g \rceil$ )

<sup>4</sup>Again confluence can be replaced by orthogonality, as in Corollary 6.5.

that are satisfiable iff the binary numbers  $\text{rk}_f$  and  $\text{rk}_g$  are decreasing (respectively equal) in the natural order. Using these we set

$$\text{valid-precedence}(\mathcal{D}) := \bigwedge_{f,g \in \mathcal{D}} (\ulcorner f \succ g \urcorner \rightarrow \ulcorner \text{rk}_f > \text{rk}_g \urcorner) \wedge \bigwedge_{f,g \in \mathcal{D}} (\ulcorner f \sim g \urcorner \rightarrow \ulcorner \text{rk}_f = \text{rk}_g \urcorner).$$

We say that a propositional assignment  $\mu$  *induces* the precedence  $\succcurlyeq$  if  $\mu$  satisfies  $\ulcorner f \succ g \urcorner$  when  $f \succ g$  and  $\ulcorner f \sim g \urcorner$  when  $f \sim g$ . The next lemma verifies that **valid-precedence** serves our needs.

**Lemma 8.1.** For any assignment  $\mu$  that satisfies **valid-precedence**( $\mathcal{D}$ ),  $\mu$  induces an admissible precedence on  $\mathcal{F}$ . Vice versa, for any admissible precedence  $\succcurlyeq$  on  $\mathcal{F}$ , any valuation  $\mu$ , satisfying  $\mu(\ulcorner f \succ g \urcorner)$  iff  $f \succ g$  and  $\mu(\ulcorner f \sim g \urcorner)$  iff  $f \sim g$ , also satisfies the formula **valid-precedence**( $\mathcal{D}$ ).

**8.2. Order Constraints.** For concrete pairs of terms  $s = f(s_1, \dots, s_n)$  and  $t$ , we define

$$\ulcorner s >_{\text{pop}^*} t \urcorner := \ulcorner s >_{\text{pop}^*}^{(1)} t \urcorner \vee \ulcorner s >_{\text{pop}^*}^{(2)} t \urcorner \vee \ulcorner s >_{\text{pop}^*}^{(3)} t \urcorner$$

which enforces the orientation  $f(s_1, \dots, s_n) >_{\text{pop}^*} t$  using propositional formulations of the three clauses in Definition 3.5. To complete the definition for arbitrary left-hand sides, we set  $\ulcorner x >_{\text{pop}^*} t \urcorner := \perp$  for all  $x \in \mathcal{V}$ . Further, weak orientation is given by

$$\ulcorner s \geq_{\text{pop}^*} t \urcorner := \ulcorner s >_{\text{pop}^*} t \urcorner \vee \ulcorner s \lesssim t \urcorner,$$

where the constraint  $\ulcorner s \lesssim t \urcorner$  refers to a formulation of Definition 3.2 in propositional logic, defined as follows. For  $s = t$  we simply set  $\ulcorner s \lesssim t \urcorner := \top$ . Consider the case  $s = f(s_1, \dots, s_n)$  and  $t = g(t_1, \dots, t_n)$ . Then  $s \lesssim t$  if  $f \sim g$  and moreover  $s_i \lesssim t_{\pi(i)}$  for all  $i = 1, \dots, n$  and some permutation  $\pi$  on argument positions that takes the separation of normal and safe positions into account. To encode  $\pi(i) = j$ , we use fresh atoms  $\pi_{i,j}$  for  $i, j = 1, \dots, n$ . The propositional formula **permutation**( $\pi, n$ ) :=  $\bigwedge_{i=1}^n \mathbf{one}(\pi_{i,1}, \dots, \pi_{i,n})$  is used to assert that the atoms  $\pi_{i,j}$  reflect a permutation on  $\{1, \dots, n\}$ . Here **one**( $\pi_{i,1}, \dots, \pi_{i,n}$ ) expresses that exactly one of its arguments evaluates to  $\top$ . We set

$$\ulcorner s \lesssim t \urcorner := \ulcorner f \sim g \urcorner \wedge \mathbf{permutation}(\pi, n) \wedge \left( \bigwedge_{j=1}^n \pi_{i,j} \rightarrow \ulcorner s_i \lesssim t_j \urcorner \wedge (\mathbf{safe}_{f,i} \leftrightarrow \mathbf{safe}_{g,j}) \right).$$

To complete the definition, we set  $\ulcorner s \lesssim t \urcorner = \perp$  for the remaining cases.

**Lemma 8.2.** Suppose the assignment  $\mu$  induces an admissible precedence  $\succcurlyeq$  and  $\mu$  satisfies  $\ulcorner s \lesssim t \urcorner$ . Then  $s \lesssim t$  with respect to the precedence  $\succcurlyeq$ . Vice versa, if  $s \lesssim t$  then  $\ulcorner s \lesssim t \urcorner$  is satisfiable by assignments  $\mu$  that induce the precedence underlying  $\lesssim$ .

We now define the encoding for the different cases underlying the definition of  $>_{\text{pop}^*}$ . Assuming that  $\ulcorner s_i \geq_{\text{pop}^*} t \urcorner$  enforces  $s_i >_{\text{pop}^*} t$  clause  $>_{\text{pop}^*}^{(1)}$  is expressible as

$$\ulcorner f(s_1, \dots, s_n) >_{\text{pop}^*}^{(1)} t \urcorner := \bigvee_{i=1}^n \ulcorner s_i \geq_{\text{pop}^*} t \urcorner.$$

in propositional logic. For clause  $>_{\text{pop}^*}^{(2)}$  we use propositional atoms  $\alpha_i$  ( $i = 1, \dots, m$ ) to mark the unique argument position of  $t = g(t_1, \dots, t_m)$  that allows  $t_i \notin \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V})$ . The

propositional formula **zero-or-one** $(\alpha_1, \dots, \alpha_m)$  expresses that zero or one  $\alpha_i$  evaluates to  $\top$ . Further, we introduce the auxiliary constraint

$$\lceil g(t_1, \dots, t_m) \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V}) \rceil := \lceil f \succ g \rceil \wedge \bigwedge_{j=1}^m \lceil t_j \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V}) \rceil.$$

and  $\lceil x \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V}) \rceil := \top$  for  $x \in \mathcal{V}$ . Using these, clause  $\succ_{\text{pop}^*}^{(2)}$  becomes expressible as

$$\begin{aligned} \lceil f(s_1, \dots, s_n) \succ_{\text{pop}^*}^{(2)} g(t_1, \dots, t_m) \rceil &:= \lceil f \in \mathcal{D} \rceil \wedge \lceil f \succ g \rceil \\ &\wedge \bigwedge_{j=1}^m (\text{safe}_{g,j} \rightarrow \lceil s \succ_{\text{pop}^*} t_j \rceil) \wedge \bigwedge_{j=1}^m (\neg \text{safe}_{g,j} \rightarrow \lceil s \succ_{\text{pop}} t_j \rceil) \\ &\wedge \text{zero-or-one}(\alpha_1, \dots, \alpha_m) \wedge \bigwedge_{j=1}^m (\neg \alpha_j \rightarrow \lceil t_j \in \mathcal{T}(\mathcal{F}^{\prec f}, \mathcal{V}) \rceil). \end{aligned}$$

Here  $\lceil f \in \mathcal{D} \rceil = \top$  if  $f \in \mathcal{D}$  and otherwise  $\lceil f \in \mathcal{D} \rceil = \perp$ . The propositional formula  $\lceil s \succ_{\text{pop}} t \rceil$  expresses the orientation with the  $\succ_{\text{pop}}$  and is given by

$$\lceil f(s_1, \dots, s_n) \succ_{\text{pop}} t \rceil := \lceil f(s_1, \dots, s_n) \succ_{\text{pop}}^{(1)} t \rceil \vee \lceil f(s_1, \dots, s_n) \succ_{\text{pop}}^{(2)} t \rceil$$

and otherwise  $\lceil x \succ_{\text{pop}} t \rceil = \perp$ , where

$$\begin{aligned} \lceil f(s_1, \dots, s_n) \succ_{\text{pop}}^{(1)} t \rceil &:= \bigvee_{i=1}^n ((\lceil s_i \succ_{\text{pop}} t \rceil \vee \lceil s_i \simeq t \rceil) \wedge (\lceil f \in \mathcal{D} \rceil \rightarrow \neg \text{safe}_{f,i})) \\ \lceil f(s_1, \dots, s_n) \succ_{\text{pop}}^{(2)} t \rceil &:= \begin{cases} \lceil f \in \mathcal{D} \rceil \wedge \lceil f \succ g \rceil & \text{if } t = g(t_1, \dots, t_m) \\ \lceil f \in \mathcal{D} \rceil \wedge \bigwedge_{j=1}^m \lceil f(s_1, \dots, s_n) \succ_{\text{pop}} t_j \rceil & \\ \perp & \text{if } t \in \mathcal{V}. \end{cases} \end{aligned}$$

This concludes the propositional formulation of clause  $\succ_{\text{pop}^*}^{(2)}$ .

The main challenge in formulating clause  $\succ_{\text{pop}^*}^{(3)}$  is to deal with the encoding of multiset-comparisons. We proceed as in [52] and encode the underlying *multiset cover*.

**Definition 8.3.** Let  $\succ_{\text{mul}}$  denote the multiset extension of a binary relation  $\succ = \succ \uplus \sim$ . Then a pair of mapping  $(\gamma, \varepsilon)$  where  $\gamma: \{1, \dots, m\} \rightarrow \{1, \dots, n\}$  and  $\varepsilon: \{1, \dots, n\} \rightarrow \{\top, \perp\}$  is a multiset cover on multisets  $\{\{a_1, \dots, a_n\}\}$  and  $\{\{b_1, \dots, b_m\}\}$  if the following holds for all  $j \in \{1, \dots, m\}$ :

- (1) if  $\gamma(j) = i$  then  $a_i \succ b_j$ , in this case we say that  $a_i$  covers  $b_j$ ;
- (2) if  $\varepsilon(j) = \top$  then  $s_{\tau(j)} \sim t_j$  and  $\tau$  is injective on  $\{j\}$ , i.e.,  $a_{\tau(j)}$  covers only  $b_j$ .

The multiset cover  $(\gamma, \varepsilon)$  is said to be *strict* if at least one cover is strict, i.e.,  $\varepsilon(j) = \perp$  for some  $j \in \{1, \dots, m\}$ .

It is straight forward to verify that multiset covers characterise the multiset extension of  $\succ$  in the following sense.

**Lemma 8.4.** We have  $\{\{a_1, \dots, a_n\}\} \succ^{\text{mul}} \{\{b_1, \dots, b_m\}\}$  if and only if there exists a multiset cover  $(\gamma, \varepsilon)$  on  $\{\{a_1, \dots, a_n\}\}$  and  $\{\{b_1, \dots, b_m\}\}$ . Moreover,  $\{\{a_1, \dots, a_n\}\} \succ^{\text{mul}} \{\{b_1, \dots, b_m\}\}$  if and only if the cover is strict.

Consider the orientation  $f(s_1, \dots, s_n) \succ_{\text{pop}^*}^{(3)} g(t_1, \dots, t_m)$ . Then normal arguments are strictly, and safe arguments weakly decreasing with respect to the multiset-extension of  $\succ_{\text{pop}^*}$ . Since the partitioning of normal and safe argument is not fixed, in the encoding of  $\succ_{\text{pop}^*}^{(3)}$  we formalise a multiset-comparison on *all* arguments, where the underlying multiset-cover  $(\gamma, \varepsilon)$  will be restricted so that if  $s_i$  covers  $t_j$ , i.e.,  $\gamma(i) = j$ , then both  $s_i$  and  $t_j$  are safe or respectively normal. To this extend, for a specific multiset cover  $(\gamma, \varepsilon)$  we introduce variables  $\gamma_{i,j}$  and  $\varepsilon_i$ , where  $\gamma_{i,j} = \top$  represents  $\gamma(j) = i$  and  $\varepsilon_i = \top$  denotes  $\varepsilon(i) = \top$  ( $1 \leq i \leq n, 1 \leq j \leq m$ ). We set

$$\begin{aligned} \lceil f(s_1, \dots, s_n) \succ_{\text{pop}^*}^{(3)} g(t_1, \dots, t_m) \rceil &:= \lceil f \in \mathcal{D} \rceil \wedge \lceil f \succ g \rceil \\ &\wedge \bigwedge_{i=1}^n \bigwedge_{j=1}^m \left( \gamma_{i,j} \rightarrow (\varepsilon_i \rightarrow \lceil s_i \approx t_j \rceil) \wedge (\neg \varepsilon_i \rightarrow \lceil s_i \succ_{\text{pop}^*} t_j \rceil) \wedge (\text{safe}_{f,i} \leftrightarrow \text{safe}_{g,j}) \right) \\ &\wedge \bigwedge_{j=1}^m \text{one}(\gamma_{1,j}, \dots, \gamma_{n,j}) \wedge \bigwedge_{i=1}^n (\varepsilon_i \rightarrow \text{one}(\gamma_{i,1}, \dots, \gamma_{i,m})) \wedge \bigvee_{i=1}^n (\neg \text{safe}_{f,i} \wedge \neg \varepsilon_i). \end{aligned}$$

Here the first line establishes the Condition 8.3(1), where  $\text{safe}_{f,i} \leftrightarrow \text{safe}_{g,j}$  additionally enforces the separation of normal from safe arguments. The final line formalises that  $\gamma$  maps  $\{1, \dots, m\}$  to  $\{1, \dots, n\}$ , Condition 8.3(2) as well as the strictness condition on normal arguments. This completes the encoding of  $\succ_{\text{pop}^*}$ .

**Lemma 8.5.** Suppose  $\mu$  induces an admissible precedence  $\succcurlyeq$  and satisfies  $\lceil s \succ_{\text{pop}^*} t \rceil$ . Then  $s \succ_{\text{pop}^*} t$  with respect to the precedence  $\succcurlyeq$ . Vice versa, if  $s \succ_{\text{pop}^*} t$  then  $\lceil s \succ_{\text{pop}^*} t \rceil$  is satisfiable assignments  $\mu$  that induce the precedence underlying  $\succ_{\text{pop}^*}$ .

Putting the constraints together we get the following theorem, which witnesses the fourth main result of this paper.

**Theorem 8.6.** Let  $\mathcal{R}$  be a constructor TRS. The propositional formula

$$\text{predicative-recursive}(\mathcal{R}) := \text{valid-precedence}(\mathcal{D}) \wedge \bigwedge_{l \rightarrow r \in \mathcal{R}} \lceil l \succ_{\text{pop}^*} r \rceil,$$

is satisfiable if and only if  $\mathcal{R}$  is predicative recursive.

We have implemented this reduction to SAT in our complexity analyser **TCF**. As underlying SAT-solver we employ the open source solver **MiniSat** [27].

**8.3. Efficiency Considerations.** The SAT-solver **MiniSat** requires its input in CNF. For a concise translation of  $\text{predicative-recursive}(\mathcal{R})$  to CNF we use the approach of Plaisted and Greenbaum [50] that gives an equisatisfiable CNF linear in size. Our implementation also eliminates redundancies resulting from multiple comparisons of the same pair of term  $s, t$  by replacing subformulas  $\lceil s \succ_{\text{pop}^*} t \rceil$  with unique propositional atoms  $\delta_{s,t}$ . Since  $\lceil s \succ_{\text{pop}^*} t \rceil$  occurs only in positive contexts, it suffices to add  $\delta_{s,t} \rightarrow \lceil s \succ_{\text{pop}^*} t \rceil$ , resulting in an equisatisfiable formula. Also during construction of  $\text{predicative-recursive}(\mathcal{R})$  our implementation performs immediate simplifications under Boolean laws.

## 9. EXPERIMENTAL ASSESSMENT

In this section we present an empirical evaluation of polynomial path orders. We selected two testbeds: Testbed **TC** constitutes of 597 terminating constructor TRSs, obtained by restricting the innermost runtime complexity problemset from the *Termination Problem Database*<sup>5</sup> (*TPDB* for short), version 8.0, to known to be terminating constructor TRSs. Termination is checked against the data available from the termination competition. Testbed **TCO**, containing 290 examples, results from restricting Testbed **TC** to orthogonal systems. Unarguably the TPDB is an imperfect choice as examples were collected primarily to assess the strength of termination provers, but it is at the moment the only extensive source of TRSs.

Experiments were conducted with **TCT** version 1.9.1,<sup>6</sup> on a laptop with 4Gb of RAM and Intel<sup>®</sup> Core<sup>™</sup> i7-2620M CPU (2.7GHz, quad-core). We assess the strength of POP\* and POP\*<sub>PS</sub> in comparison to its predecessors MPO and LMPO. The implementation of MPO, LMPO and POP\*<sub>PS</sub> follows the line of polynomial path orders as explained in Section 8.<sup>7</sup> We contrast these syntactic techniques to *interpretations* as implemented in our complexity tool **TCT**. The last column show result of constructor restricted matrix interpretations [45] (dimension 1 and 3) as well as polynomial interpretations [21] (degree 2 and 3), run in parallel on the quad-core processor. We employ interpretations in their default configuration of **TCT**, noteworthy coefficients (respectively entries in coefficients) range between 0 and 7, and we also make use of the *usable argument positions* criterion [31] that weakens monotonicity constraints. Table 1 shows totals on systems that can respectively cannot be handled.<sup>8</sup> To the right of each entry we annotate the average execution time, in seconds.

		MPO	LMPO	POP*	POP* <sub>PS</sub>	interpretations
<b>TC</b>	compatible	76\0.33	57\0.20	43\0.18	56\0.19	139\2.77
	incompatible	521\0.58	540\0.47	554\0.42	541\0.43	272\6.47
	timeout	—	—	—	—	186\25.0
<b>TCO</b>	compatible	40\0.29	29\0.16	24\0.14	29\0.15	75\2.81
	incompatible	250\0.33	261\0.27	266\0.26	261\0.27	133\6.12
	timeout	—	—	—	—	82\25.0

Table 1: Empirical Evaluation, comparing syntactic to semantic techniques.

It is immediate that syntactic techniques cannot compete with the expressive power of interpretations. In Testbed **TC** there are in fact only three examples compatible with POP\*<sub>PS</sub> where **TCT** could not find interpretations. There are additionally four examples compatible with LMPO but not so with interpretations, including the TRS  $\mathcal{R}_{\text{bin}}$  from Example 1.2. All but one (noteworthy the merge-sort algorithm from Steinbach and Kühlers collection [54, Example 2.43]) of these do in fact admit exponential runtime complexity, thus a priori they are not compatible to the restricted interpretations.

<sup>5</sup>The TPDB is available online <http://termcomp.uibk.ac.at/>.

<sup>6</sup>Available from <http://cl-informatik.uibk.ac.at/software/tct/>.

<sup>7</sup>As far as we know our implementation of LMPO in **TCT** is the only implementation currently available.

<sup>8</sup>Full evidence available at <http://cl-informatik.uibk.ac.at/software/tct/experiments/popstar>.

We emphasise that parameter substitution significantly increases the strength of  $\text{POP}^*$ , 13 examples are provable by  $\text{POP}_{\text{PS}}^*$  but neither by  $\text{POP}^*$  nor LMPO. LMPO could benefit from parameter substitution, we conjecture that the resulting order is still sound for FP.

In sum on Testbed **TCO**, containing only orthogonal TRSs, in total 75 systems (26% of the testbed) can be verified to encode polytime computable functions, 35 (12% of the testbed) can be verified polytime computable by only syntactic techniques. It should be noted that not all examples appearing in our collection encode polytime computable functions, the total amount of such systems is unknown.

It seems that Table 1 clearly shows the weakness of polynomial path orders (even with parameter substitution) for automated polynomial runtime complexity. However, remark the average execution times provided.  $\text{POP}_{\text{PS}}^*$  succeeds on average 14 times faster than polynomial and matrix interpretations. Here the difficulty of implementing interpretations efficiently is also reflected in the total number of timeouts. Furthermore note that a competitive complexity analyser cannot be based on direct techniques alone. Instead, our complexity analyser **TC** recursively decomposes complexity problems using various complexity preserving transformation techniques [10], discarding those problems that can be handled by basic techniques as contrasted in Table 1. Certificates are only obtained if finally all subproblems can be discarded, above all it is crucial that subproblems can be discarded quickly. Due to the efficiency of syntactic methods, these can be safely preposed to semantic techniques, thus speeding up the overall procedure.

## 10. CONCLUSION AND FUTURE WORK

This paper is concerned with the complexity analysis of constructor term rewrite systems and its ramification in implicit computational complexity.

We have proposed a path order with multiset status, the polynomial path order  $\text{POP}^*$ . The order  $\text{POP}^*$  is a syntactical restriction of multiset path orders, with the distinctive feature that the innermost runtime complexity of compatible TRSs lies in  $O(n^d)$  for some  $d$ . Based on  $\text{POP}^*$ , we delineate a class of rewrite systems, dubbed systems of predicative recursion, so that the class of functions computed by these systems corresponds to FP, the class of polytime computable functions. We have shown that an extension of  $\text{POP}^*$ , the order  $\text{POP}_{\text{PS}}^*$  that also accounts for parameter substitution, increases the intensionality of  $\text{POP}^*$ .

From the viewpoint of implicit computational complexity we have provided new implicit characterisations of the class of polytime functions. More precisely,  $\text{POP}^*$  and  $\text{POP}_{\text{PS}}^*$  are sound for the class of function problems FNP and are readily applicable to obtain exact characterisations of the polytime computable functions. As an easy corollary, we have given an alternative proof of Bellantoni's result that the polytime computable functions are closed under parameter substitution.

From the viewpoint of (automated) runtime complexity analysis we have proposed two new syntactic techniques to establish polynomial innermost runtime complexity. In contrast to semantic techniques polynomial path orders are partly lacking in intensionality but greatly surpluses in verification time. Note that in our complexity prover **TC**, we do not intend to replace semantic techniques, but rather prepose them by  $\text{POP}_{\text{PS}}^*$ , in order to improve **TC** both in analytic power and speed.

In runtime complexity analysis one is in particular interested in obtaining asymptotically tight bounds. Although we could estimate the degree of the witnessing bounding

function for  $\text{POP}^*$  and  $\text{POP}_{\text{PS}}^*$ , such a bound would be a gross overestimation. This is partly due to the underlying multiset extension.

Very recently, together with Eguchi we have proposed a simplification of the polynomial path orders studied here: the *small polynomial path orders* (*sPOP*<sup>\*</sup> for short). This termination order entails a finer control on the runtime complexity: for any rewrite system compatible with *sPOP*<sup>\*</sup> that employs recursion upto depth  $d$ , the innermost runtime complexity is polynomially bounded of degree  $d$ . This bound is tight, see [13]. This becomes possible, as the underlying scheme of safe composition is restricted to so-called *weak safe composition*.

#### ACKNOWLEDGEMENT

We are in particular thankful to Nao Hirokawa for fruitful discussions. Furthermore the second author would like to thank Toshiyasu Arai for having introduced him to the topic of predicative recursion. Finally, we are indebted to the anonymous reviewers for their constructive criticism.

#### REFERENCES

- [1] E. Albert, P. Arenas, S. Genaim, M. Gómez-Zamalloa, G. Puebla, D. Ramírez, G. Román, and D. Zanardini. Termination and Cost Analysis with COSTA and its User Interfaces. *Electronic Notes in Theoretical Computer Science*, 258(1):109–121, 2009.
- [2] C. Alias, A. Darte, P. Feautrier, and L. Gonnord. Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs. In *Proc. of 17<sup>th</sup> SAS*, volume 6337 of *Lecture Notes in Computer Science*, pages 117–133, 2010.
- [3] T. Arai and G. Moser. A Note on a Term Rewriting Characterization of PTIME. In *Proc. of 7<sup>th</sup> WST*, pages 10–13. number AIB-2004-07 of Aachener Informatik-Berichte, 2004. Extended abstract.
- [4] T. Arai and G. Moser. Proofs of Termination of Rewrite Systems for Polytime Functions. In *Proc. of 25<sup>th</sup> FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 529–540. Springer Verlag, 2005.
- [5] M. Avanzini and G. Moser. Complexity Analysis by Rewriting. In *Proc. of 9<sup>th</sup> FLOPS*, volume 4989 of *Lecture Notes in Computer Science*, pages 130–146. Springer Verlag, 2008.
- [6] M. Avanzini and G. Moser. Dependency Pairs and Polynomial Path Orders. In *Proc. of 20<sup>th</sup> RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 48–62. Springer Verlag, 2009.
- [7] M. Avanzini and G. Moser. Polynomial Path Orders and the Rules of Predicative Recursion with Parameter Substitution. In *Proc. of 10<sup>th</sup> WST*, pages 16–20, 2009.
- [8] M. Avanzini and G. Moser. Complexity Analysis by Graph Rewriting. In *Proc. of 10<sup>th</sup> FLOPS*, volume 6009 of *Lecture Notes in Computer Science*, pages 257–271. Springer Verlag, 2010.
- [9] M. Avanzini and G. Moser. Closing the Gap Between Runtime Complexity and Polytime Computability. In *Proc. of 21<sup>st</sup> RTA*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 33–48, 2010.

- [10] M. Avanzini and G. Moser. A Combination Framework for Complexity. In *Proc. 24th RTA*, volume 21, pages 55–70. Leibniz International Proceedings in Informatics, 2013.
- [11] M. Avanzini and G. Moser. Tyrolean Complexity Tool: Features and usage. In *Proc. 24th RTA*, Leibniz International Proceedings in Informatics, pages 71–80, 2013. 21.
- [12] M. Avanzini, G. Moser, and A. Schnabl. Automated Implicit Computational Complexity Analysis (System Description). In *Proc. of 4<sup>th</sup> IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 132–139. Springer Verlag, 2008.
- [13] M. Avanzini, N. Eguchi, and G. Moser. A New Order-theoretic Characterisation of the Polytime Computable Functions. In *Proc. of 10<sup>th</sup> APLAS*, volume 7705 of *Lecture Notes in Computer Science*, pages 280–295, 2012.
- [14] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [15] P. Baillot, J.-Y. Marion, and S. Ronchi Della Rocca. Guest Editorial: Special Issue on Implicit Computational Complexity. *ACM Transactions on Computational Logic*, 10(4), 2009.
- [16] A. Beckmann and A. Weiermann. A Term Rewriting Characterization Of the Polytime Functions and Related Complexity Classes. *Archive for Mathematical Logic*, 36:11–30, 1996.
- [17] S. Bellantoni. *Predicative Recursion and Computational Complexity*. PhD thesis, University of Toronto, Faculty for Computer Science, 1992.
- [18] S. Bellantoni and S. Cook. A new Recursion-Theoretic Characterization of the Polytime Functions. *Computational Complexity*, 2(2):97–110, 1992.
- [19] P. Van Emde Boas. Machine Models and Simulation. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 1–66. The MIT Press, 1990.
- [20] G. Bonfante and G. Moser. Characterising Space Complexity Classes via Knuth-Bendix Orders. In *Proc. of 17<sup>th</sup> LPAR*, volume 6397 of *Lecture Notes in Computer Science*, pages 142–156, 2010.
- [21] G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with Polynomial Interpretation Termination Proof. *Journal of Functional Programming*, 11(1):33–53, 2001.
- [22] G. Bonfante, J.-Y. Marion, and J.-Y. Moyén. Quasi-interpretations: A Way to Control Resources. *Theoretical Computer Science*, 412(25), 2011.
- [23] W. Buchholz. Proof-theoretical Analysis of Termination Proofs. *Annals of Pure and Applied Logic*, 75:57–65, 1995.
- [24] E. A. Cichon and A. Weiermann. Term Rewriting Theory for the Primitive Recursive Functions. *Annals of Pure and Applied Logic*, 83(3):199–223, 1997.
- [25] U. Dal Lago and S. Martini. On Constructor Rewrite Systems and the Lambda-Calculus. In *Proc. of 36<sup>th</sup> ICALP*, volume 5556 of *Lecture Notes in Computer Science*, pages 163–174. Springer Verlag, 2009.
- [26] U. Dal Lago and S. Martini. Derivational Complexity is an Invariant Cost Model. In *Proc. of 1<sup>st</sup> FOPARA*, 2009.
- [27] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In *Proc. of 6<sup>th</sup> SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer Verlag, 2003.
- [28] M. C. F. Ferreira. *Termination of Term Rewriting*. PhD thesis, University of Utrecht, November 1995. Well-foundedness, Totality and Transformations.



- [29] S. Gulwani, K.K. Mehra, and T.M. Chilimbi. SPEED: Precise and Efficient Static Estimation of Program Computational Complexity. In *Proc. of 36<sup>th</sup> POPL*, pages 127–139. Association for Computing Machinery, 2009.
- [30] N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. In *Proc. of 4<sup>th</sup> IJCAR*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 364–380. Springer Verlag, 2008.
- [31] N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. *CoRR*, abs/1102.3129, 2011. submitted.
- [32] D. Hofbauer. Termination Proofs by Multiset Path Orderings Imply Primitive Recursive Derivation Lengths. *Theoretical Computer Science*, 105:129–140, 1992.
- [33] D. Hofbauer and C. Lautemann. Termination Proofs and the Length of Derivations. In *Proc. of 3<sup>rd</sup> RTA*, volume 355 of *Lecture Notes in Computer Science*, pages 167–177. Springer Verlag, 1989.
- [34] J. Hoffmann, K. Aehlig, and M. Hofmann. Multivariate Amortized Resource Analysis. In *Proc. of 38<sup>th</sup> POPL*, pages 357–370. Association for Computing Machinery, 2011.
- [35] J. Hoffmann, K. Aehlig, and M. Hofmann. Resource Aware ML. In *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 781–786, 2012.
- [36] M. Hofmann and D. Rodriguez. Automatic Type Inference for Amortised Heap-Space Analysis. In *Proc. of 22<sup>nd</sup> ESOP*, volume 7792 of *Lecture Notes in Computer Science*, pages 593–613, 2013.
- [37] D. E. Knuth. Johann Faulhaber and Sums of Powers. *MC*, 203:277–294, 1993.
- [38] U. Dal Lago. A Short Introduction to Implicit Computational Complexity. In *Lectures on Logic and Computation - ESSLLI 2010 Copenhagen, Denmark, August 2010, ESSLLI 2011, Ljubljana, Slovenia, August 2011, Selected Lecture Notes*, volume 7388 of *Lecture Notes in Computer Science*, pages 89–109, 2011.
- [39] U. Dal Lago, S. Martini, and M. Zorzi. General Ramified Recurrence is Sound for Polynomial Time. In *Proc. of DICE2010*, volume 23 of *Electronic Proceedings in Theoretical Computer Science*, pages 47–62, 2010.
- [40] D. Leivant. Subrecursion and lambda representation over free algebras (preliminary summary). In *Feasible mathematics (Ithaca, NY, 1989)*, Progr. Comput. Sci. Appl. Logic, pages 281–291. Birkhäuser Boston, 1990.
- [41] D. Leivant. A Foundational Delineation of Computational Feasibility. In *Proc. of 6<sup>th</sup> LICS*, pages 2–11. IEEE Computer Society, 1991.
- [42] D. Leivant. Stratified Functional Programs and Computational Complexity. In *Proc. of 20<sup>th</sup> POPL*, pages 325–333. ACM Press, 1993.
- [43] J.-Y. Marion. Analysing the Implicit Complexity of Programs. *Information and Computation*, 183:2–18, 2003.
- [44] J.-Y. Marion and R. Péchoux. Sup-interpretations, a Semantic Method for Static Analysis of Program Resources. *ACM Trans. Comput. Log.*, 10(4), 2009.
- [45] A. Middeldorp, G. Moser, F. Neuraüter, J. Waldmann, and H. Zankl. Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems. In *Proc. of 4<sup>th</sup> CAI*, volume 6472 of *Lecture Notes in Computer Science*, pages 1–20. Springer Verlag, 2011.
- [46] G. Moser and A. Schnabl. Proving quadratic derivational complexities using context dependent interpretations. In *Proc. of 19<sup>th</sup> RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 276–290, 2008.

- [47] G. Moser and A. Weiermann. Relating derivation lengths with the slow-growing hierarchy directly. In *Proc. of 14<sup>th</sup> RTA*, volume 2706 of *Lecture Notes in Computer Science*, pages 296–310, 2003.
- [48] L. Noschinski, F. Emmes, and J. Giesl. A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems. In *Proc. of 23<sup>rd</sup> CADE*, Lecture Notes in Computer Science, pages 422–438. Springer Verlag, 2011.
- [49] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, second edition, 1995.
- [50] D. A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.
- [51] P. Schneider-Kamp, C. Fuhs, R. Thiemann, J. Giesl, E. Annov, M. Codish, A. Middeldorp, and H. Zankl. Implementing RPO and POLO Using SAT. In *DDP*, number 07401 in Leibniz International Proceedings in Informatics. Dagstuhl, 2007.
- [52] P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, and J. Giesl. Proving Termination Using Recursive Path Orders and SAT Solving. In *Proc. of 6<sup>th</sup> FroCoS*, volume 4720 of *Lecture Notes in Computer Science*, pages 267–282. Springer Verlag, 2007.
- [53] H. Simmons. The Realm of Primitive Recursion. *Applied Mathematicas Letters*, 27: 177–188, 1988.
- [54] J. Steinbach and U. Kühler. Check your Ordering - Termination Proofs and Open Problems. Technical Report SEKI-Report SR-90-25, University of Kaiserslautern, 1990.
- [55] R.E. Tarjan. Amortized Computational Complexity. *SIAM J. Alg. Disc. Meth*, 6(2): 306–318, 1985.
- [56] TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
- [57] H. Zankl and M. Korp. Modular Complexity Analysis via Relative Complexity. In *Proc. of 21<sup>st</sup> RTA*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 385–400, 2010.
- [58] H. Zankl and A. Middeldorp. Satisfying KBO Constraints. In *Proc. of the 18<sup>th</sup> RTA*, volume 4533 of *Lecture Notes in Computer Science*, pages 389–403. Springer Verlag, 2007.
- [59] F. Zuleger, S. Gulwani, M. Sinn, and H. Veith. Bound Analysis of Imperative Programs with the Size-Change Abstraction. In *Proc. of 18<sup>th</sup> SAS*, volume 6887 of *Lecture Notes in Computer Science*, pages 280–297. Springer Verlag, 2011.