Small Polynomial Path Orders in TCT*

Martin Avanzini[1], Georg Moser[1], and Michael Schaper[1]

1 **Institute of Computer Science,**
University of Innsbruck, Austria
`{martin.avanzini,georg.moser,michael.schaper}@uibk.ac.at`

Abstract

1998 ACM Subject Classification F.2.2, F.4.1, F.4.2, D.2.4, D.2.8

Keywords and phrases Runtime Complexity, Polynomial Time Functions, Rewriting

1 Introduction

In [2] we propose the *small polynomial path order* (sPOP* for short). This order provides a characterisation of the class of *polytime computable function* via term rewrite systems (TRSs for short). Any polytime computable function is expressible as a constructor TRS which is compatible with (an instance of) sPOP*. On the other hand, any function defined by a constructor TRS compatible with sPOP* is polytime computable. This order has also ramifications in the *automated complexity analysis* of rewrite systems. The *innermost runtime complexity* of any constructor TRS $\mathcal{R}$ compatible with sPOP* lies in $O(n^d)$. Here $d \in \mathbb{N}$ refers to the maximal *depth of recursion* of defined symbols f in $\mathcal{R}$.

This work deals with the implementation of sPOP* in the *Tyrolean complexity tool*[1] (TCT for short). The order has been extended to relative rewriting, and takes also *usable arguments* [6] into account. As by-product, we obtain a form of *reduction pair* from sPOP*. Such reduction pairs can be used in the *dependency pair* analysis of Hirokawa and the second author [5] and Noschinski et al. [7]. For details and proofs we refer the reader to [1].

2 Small Polynomial Path Orders

We assume familiarity with rewriting [3]. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$, with *defined symbols* in $\mathcal{D}$. *Constructors* are denoted by $\mathcal{C} := \mathcal{F} \setminus \mathcal{D}$. Further, let $\mathcal{K} \subseteq \mathcal{D}$ denote a set of *recursive symbols*, and let $\gtrsim$ denote a (quasi)-precedence on $\mathcal{F}$. We denote by $>$ and $\sim$ the proper order and equivalence underlying $\gtrsim$. We call the precedence $\gtrsim$ *admissible* for sPOP* if it retains the partitioning of $\mathcal{F}$ in the following sense. If $f \sim g$ then $f \in \mathcal{C}$ implies $g \in \mathcal{C}$, likewise, $f \in \mathcal{K}$ implies $g \in \mathcal{K}$. Small polynomial path orders embody the principle of *predicative recursion* [4] on compatible TRSs. To this end, arguments of every function symbol are partitioned into normal and safe ones. Notationally we write $f(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ with *normal* arguments $t_1, \ldots, t_k$ separated from *safe* arguments $t_{k+1}, \ldots, t_{k+l}$ by a semicolon. For constructors, we fix that all argument positions are safe. We define the equivalence $\approx_\mathsf{s}$ on terms respecting this separation as follows: $s \approx_\mathsf{s} t$ holds if $s = t$ or $s = f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l})$ and $t = g(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ where $f \sim g$ and $s_i \approx_\mathsf{s} t_i$ holds for all $i = 1, \ldots, k + l$. We write $s \rhd^{\mathsf{n}}_{\approx} t$ if t is a subterm (modulo $\approx_\mathsf{s}$) of a normal argument of s.

* This work is partially supported by FWF (Austrian Science Fund) project I-608-N18.

[1] TCT is open source and available from `http://cl-informatik.uibk.ac.at/software/tct`.

The following definition introduces small polynomial path orders, also accounting for parameter substitution [2]. We denote by $\mathcal{T}(\mathcal{F}^{<f}, \mathcal{V})$ the set of terms built from variables and function symbols $\mathcal{F}^{<f} := \{g \in \mathcal{F} \mid f > g\}$.

▶ **Definition 2.1.** Let $s = f(s_1, \dots, s_k; s_{k+1}, \dots, s_{k+l})$. Then $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$ if either

1) $s_i \gtrsim_{\mathsf{spop}^*_{\mathsf{ps}}} t$ for some argument s_i of s.
2) $f \in \mathcal{D}$, $t = g(t_1, \dots, t_m; t_{m+1}, \dots, t_{m+n})$ with $f > g$ and the following conditions hold: (i) $s \rhd^{\mathsf{n}}/_{\approx} t_j$ for all normal arguments t_j of t, (ii) $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ for all safe arguments t_j of t, and (iii) $t_j \notin \mathcal{T}(\mathcal{F}^{<f}, \mathcal{V})$ for at most one $j \in \{1, \dots, k + l\}$.
3) $f \in \mathcal{K}$, $t = g(t_1, \dots, t_k; t_{k+1}, \dots, t_{k+l})$ with $f \sim g$ and the following conditions hold: (i) $\langle s_1, \dots, s_k\rangle >^{\mathsf{prod}}_{\mathsf{spop}^*_{\mathsf{ps}}} \langle t_1, \dots, t_k\rangle$, (ii) $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ for all safe arguments t_j ($j = k+1, \dots, k+l$), and (iii) $t_j \in \mathcal{T}(\mathcal{F}^{<f}, \mathcal{V})$ for all $j = 1, \dots, k + l$.

Here $\gtrsim_{\mathsf{spop}^\star}$ denotes the extension of $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ by safe equivalence $\approx_{\mathsf{s}}$. Further, $>^{\mathsf{prod}}_{\mathsf{spop}^*_{\mathsf{ps}}}$ denotes the product extension of $>_{\mathsf{spop}^*_{\mathsf{ps}}}$: $\langle s_1, \dots, s_n\rangle >^{\mathsf{prod}}_{\mathsf{spop}^*_{\mathsf{ps}}} \langle t_1, \dots, t_n\rangle$ if $s_i \gtrsim_{\mathsf{spop}^\star} t_i$ for all $i = 1, \dots, n$, and $s_{i_0} >_{\mathsf{spop}^*_{\mathsf{ps}}} t_{i_0}$ for some $i_0 \in \{1, \dots, n\}$.

The *depth of recursion* $\mathsf{rd}_{\gtrsim,\mathcal{K}}(f)$ of $f \in \mathcal{F}$ is recursively defined by $\mathsf{rd}_{\gtrsim,\mathcal{K}}(f) := 1+d$ if $f \in \mathcal{K}$ and $\mathsf{rd}_{\gtrsim,\mathcal{K}}(f) := d$ if $f \notin \mathcal{K}$, where $d = \max\ \{0\} \cup \{\mathsf{rd}_{\gtrsim,\mathcal{K}}(g) \mid f > g\}$.

▶ Proposition 2.2 ([2]). Let $\mathcal{R}$ be a constructor TRS compatible with an instance $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ based on an admissible precedence $\gtrsim$ with recursive symbols $\mathcal{K}$. Then the innermost runtime complexity of $\mathcal{R}$ lies in $O(n^d)$, where $d = \max\{0\} \cup \{\mathsf{rd}_{\gtrsim,\mathcal{K}}(f) \mid f \in \mathcal{D}\}$.

3 Polynomial Path Orders as Complexity Processors

Our tool $\mathsf{T_CT}$ operates internally on *complexity problems* $\mathcal{P} = \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T}\rangle$, where $\mathcal{S}, \mathcal{W}, \mathcal{Q}$ are TRSs and $\mathcal{T}$ denotes a set of ground terms. The set $\mathcal{T}$ is called the set of *starting terms* of $\mathcal{P}$. Throughout the following, this complexity problem is kept fixed. The *complexity (function)* $\mathsf{cp}_{\mathcal{P}} : \mathbb{N} \to \mathbb{N}$ of $\mathcal{P}$ is defined as the partial function

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}) \mid \exists t \in \mathcal{T} \text{ and } |t| \leqslant n\}\ .$$

Here $\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}} := \xrightarrow{\mathcal{Q}}^*_{\mathcal{W}} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{S}} \cdot \xrightarrow{\mathcal{Q}}^*_{\mathcal{W}}$ denotes the *$\mathcal{Q}$-restricted rewrite relation of $\mathcal{S}$ relative to $\mathcal{W}$*, where $\xrightarrow{\mathcal{Q}}_{\mathcal{R}}$ is the restriction of $\to_{\mathcal{R}}$ where all proper subterms of the redex are in $\mathcal{Q}$ normal form. We call the complexity problem $\mathcal{P}$ a *runtime complexity problem* if all terms in $\mathcal{T}$ are *basic*, i.e., of the form $f(t_1, \dots, t_k)$ for $f \in \mathcal{D}$ and constructor terms $t_1, \dots, t_k$. It is called an *innermost complexity problem* if all normal forms of $\mathcal{Q}$ are normal forms of $\mathcal{S} \cup \mathcal{W}$.

A *(complexity) judgement* is a statement $\vdash \mathcal{P} \colon f$ where $\mathcal{P}$ is a complexity problem and $f : \mathbb{N} \to \mathbb{N}$. This judgement is *valid* if $\mathsf{cp}_{\mathcal{P}}$ is defined on all inputs, and $\mathsf{cp}_{\mathcal{P}} \in \mathcal{O}(f)$. A *complexity processor* is an inference rule

$$\frac{\vdash \mathcal{P}_1 \colon f_1 \quad \cdots \quad \vdash \mathcal{P}_n \colon f_n}{\vdash \mathcal{P} \colon f}\ .$$

This processor is *sound* if $\vdash \mathcal{P} \colon f$ is valid whenever the judgements $\vdash \mathcal{P}_1 \colon f_1, \dots, \vdash \mathcal{P}_n \colon f_n$ are valid. We follow the usual convention and annotate side conditions as premises to inference rules. An inference of $\vdash \mathcal{P} \colon f$ using sound processors is called a *complexity proof*. If this inference admits no assumptions, then the judgement $\vdash \mathcal{P} \colon f$ is valid.

In the following, we propose a complexity processors based on sPOP* that operates on innermost runtime complexity problems. In essence, this processor requires that $\mathcal{W} \subseteq \gtrsim_{\mathsf{spop}^*_{\mathsf{ps}}}$

and $\mathcal{S} \subseteq >_{\mathsf{spop}^*_{\mathsf{ps}}}$ holds, and that $\mathcal{W}$ and $\mathcal{S}$ are constructor TRSs. If these requirements are met, then the complexity of $\mathcal{P}$ lies in $\mathcal{O}(n^d)$ for $d \in \mathbb{N}$ the maximal depth of recursion as in Proposition 2.2. To weaken monotonicity requirements, we integrate *argument filterings* into the order. The argument filtering is constrained, so that in derivations of starting terms, no redex is removed. Compare [6], where *μ-monotone orders* are used in a similar spirit.

An argument filtering (for a signature $\mathcal{F}$) is a mapping π that assigns to every k-ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1, \dots, k\}$ or a (possibly empty) list $[i_1, \dots, i_l]$ of argument positions with $1 \leqslant i_1 < \cdots < i_l \leqslant k$. If $\pi(f)$ is a list we say that π is *non-collapsing* on f. Below π always denotes an argument filtering. For each $f \in \mathcal{F}$, let f_π denote a fresh function symbol associated with f. We define $\mathcal{F}_\pi := \{f_\pi \mid f \in \mathcal{F} \text{ and } \pi(f) = [i_1, \dots, i_l]\}$. The sets $\mathcal{D}_\pi$ and $\mathcal{C}_\pi$ denote the defined symbols and constructors in $\mathcal{F}_\pi$, as given by the restriction of $\mathcal{F}_\pi$ to symbols f_π associated with $f \in \mathcal{D}$ and $f \in \mathcal{C}$ respectively. We denote by π also its extension to terms: $\pi(t) := t$ if t is a variable, and for $t = f(t_1, \dots, t_k)$, $\pi(t) := \pi(t_i)$ if $\pi(f) = i$ and $f(\pi(t_{i_1}), \dots, \pi(t_{i_l}))$ if $\pi(f) = [i_1, \dots, i_l]$. For an order $\succ$ on terms over $\mathcal{F}_\pi$, we define $s \succ^\pi t$ if $\pi(s) \succ \pi(t)$ holds.

A map $\mu \colon \mathcal{F} \to \mathcal{P}(\mathbb{N})$ with $\mu(f) \subseteq \{1, \dots, k\}$ for every k-ary $f \in \mathcal{F}$ is called a *replacement map* on $\mathcal{F}$. The set $\mathcal{P}\mathrm{os}_\mu(t)$ of *μ-replacing positions* in a term t is given by $\mathcal{P}\mathrm{os}_\mu(t) := \varnothing$ if t is a variable, and $\mathcal{P}\mathrm{os}_\mu(t) := \{\epsilon\} \cup \{i{\cdot}p \mid i \in \mu(f) \text{ and } p \in \mathcal{P}\mathrm{os}_\mu(t_i)\}$ if $t = f(t_1, \dots, t_k)$. For a binary relation $\to$ on terms we denote by $\mathcal{T}_\mu(\to)$ the set of terms t where sub-terms at non-μ-replacing positions are in normal form: $t \in \mathcal{T}_\mu(\to)$ if for all positions p in t, if $p \notin \mathcal{P}\mathrm{os}_\mu(t)$ then $t|_p \to u$ does not hold for any term u. Let $\mathcal{R}$ denote a set of rewrite rules. A replacement map μ is called a *usable replacement map* for $\mathcal{R}$ in $\mathcal{P}$, if $\to^*_{\mathcal{S} \cup \mathcal{W}}(\mathcal{T}) \subseteq \mathcal{T}_\mu(\xrightarrow{\mathcal{Q}}_{\mathcal{R}})$. For a usable replacement map μ and argument filtering π, we say that π *agrees with* μ if for all function symbols f in the domain of μ, either (i) $\pi(f) = i$ and $\mu(f) \subseteq \{i\}$ or otherwise (ii) $\mu(f) \subseteq \pi(f)$ holds.

▶ **Theorem 3.1.** *Let $\mathcal{P} = \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$ be an innermost complexity problem, where $\mathcal{S}$ and $\mathcal{W}$ are constructor TRSs. Let π denote an argument filtering on the symbols in $\mathcal{P}$ that agrees with a usable replacement map for $\mathcal{S}$ in $\mathcal{P}$, and that is non-collapsing on defined symbols of $\mathcal{S}$. Let $\mathcal{K}_\pi \subseteq \mathcal{D}_\pi$ denote a set of recursive function symbols, and $\gtrsim$ an admissible precedence on $\mathcal{F}_\pi$. The following processor is sound, for $d := \max\{0\} \cup \{\mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(f_\pi) \mid f_\pi \in \mathcal{F}_\pi\}$.*

$$\frac{\mathcal{S} \subseteq >^\pi_{\mathsf{spop}^*_{\mathsf{ps}}} \quad \mathcal{W} \subseteq \gtrsim^\pi_{\mathsf{spop}^*_{\mathsf{ps}}}}{\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon n^d} \ .$$

We remark that the restriction that π is non-collapsing on defined symbols of $\mathcal{S}$ is essential, compare also [1]. In $\mathsf{T\mathcal{C}T}$, Theorem 3.1 is usually applied in combination with the *relative decomposition processor* [1], This processor allows the iterated combination of different techniques, by translating the judgement $\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon f$ into the two judgements $\vdash \langle \mathcal{S}_1/\mathcal{S}_2 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon f$ and $\vdash \langle \mathcal{S}_2/\mathcal{S}_1 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon f$, where $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$. Theorem 3.1 is tight, in the sense that for any $d \in \mathbb{N}$ one can find a complexity problem $\mathcal{P}$ that satisfies the pre-conditions, and whose complexity function lies in $\Omega(n^d)$ [2]. The next example illustrates the application of Theorem 3.1.

▶ **Example 3.2.** Consider the innermost complexity problem $\mathcal{P}^\sharp_{\mathsf{log}} = \langle \mathcal{S}^\sharp_{\mathsf{log}}/\mathcal{W}_{\mathsf{log}}, \mathcal{S}^\sharp_{\mathsf{log}} \cup \mathcal{W}_{\mathsf{log}}, \mathcal{T}^\sharp_{\mathsf{log}} \rangle$ where the TRS $\mathcal{S}^\sharp_{\mathsf{log}}$ consisting of the rewrite rules

$$\mathsf{half}^\sharp(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{half}^\sharp(x) \qquad \mathsf{log}^\sharp(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{log}^\sharp(\mathsf{s}(\mathsf{half}(x))) \ ,$$

the TRS $\mathcal{W}_{\mathsf{log}}$ consists of the rules

$$\mathsf{half}(0) \to 0 \qquad \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{half}(x)) \ ,$$

and $\mathcal{T}^\sharp$ consists of the basic terms $f(\mathsf{s}^n(0))$ for $n \in \mathbb{N}$ and $f \in \{\mathsf{half}^\sharp, \mathsf{log}^\sharp\}$. Observe that the rules in $\mathcal{S}^\sharp_{\mathsf{log}}$ can only be applied on root positions in derivations starting from $\mathcal{T}^\sharp_{\mathsf{log}}$. It follows that the map $\mu_\varnothing$, which maps any function symbol f in $\mathcal{P}^\sharp_{\mathsf{log}}$ to $\varnothing$, is a usable replacement map for $\mathcal{S}^\sharp_{\mathsf{log}}$ in $\mathcal{P}^\sharp_{\mathsf{log}}$. Consider the argument filtering π with $\pi(\mathsf{half}) = 1$ and $\pi(f) = [1]$ for $f \neq \mathsf{half}$. Note that π trivially agrees with $\mu_\varnothing$. Using $\mathcal{K}_\pi := \{\mathsf{half}^\sharp, \mathsf{log}^\sharp\}$ and the empty precedence one can show $\mathcal{S}^\sharp_{\mathsf{log}} \subseteq >^\pi_{\mathsf{spop}^*_{\mathsf{ps}}}$ and $\mathcal{W}_{\mathsf{log}} \subseteq \gtrsim^\pi_{\mathsf{spop}^*_{\mathsf{ps}}}$. Trivially $\mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(\mathsf{s}_\pi) = \mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(0_\pi) = 0$, as neither $\mathsf{half}^\sharp_\pi > \mathsf{log}^\sharp_\pi$ nor $\mathsf{log}^\sharp_\pi > \mathsf{half}^\sharp_\pi$ holds, we see that $\mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(\mathsf{half}^\sharp_\pi) = \mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(\mathsf{log}^\sharp_\pi) = 1$. By Theorem 3.1, the complexity of $\mathcal{P}^\sharp_{\mathsf{log}}$ is bounded by a linear function.

4 Polynomial Path Orders and Dependency Pairs

In TCT, a *dependency pair* problem (*DP* problem for short) is a complexity problem whose strict and weak component contains also *dependency pairs*. Unlike for termination analysis, we allow *compound symbols* in right hand sides of dependency pairs. The purpose of these symbols is to group function calls. The example considered above is a DP problem that was generated by TCT on AG01/#3.7 from the *termination problem data base*[2] (*TPDB* for short). For each k-ary $f \in \mathcal{D}$, let $f^\sharp$ denote a fresh function symbol also of arity k, the *dependency pair symbol* (of f). The least extension of $\mathcal{F}$ to all dependency pair symbols is denoted by $\mathcal{F}^\sharp$. We define $t^\sharp := f^\sharp(t_1, \dots, t_k)$ if $t = f(t_1, \dots, t_k)$ and $f \in \mathcal{D}$, and $t^\sharp := t$ otherwise. For a set of terms T, we denote by $T^\sharp$ the set of *marked terms* $T^\sharp := \{t^\sharp \mid t \in T\}$. Consider the infinite signature $\mathcal{C}\mathsf{om}$ that contains for each $i \in \mathbb{N}$ a fresh constructor symbol $\mathsf{c}_i \in \mathcal{C}\mathsf{om}$ of arity i. Symbols in $\mathcal{C}\mathsf{om}$ are called *compound symbols*. We denote by $\textsc{com}(t)$ the term t, and overload this notation to sequences of terms such that $\textsc{com}(t_1, \dots, t_k) = \mathsf{c}_k(t_1, \dots, t_k)$ for $k \neq 1$. A *dependency pair* (*DP* for short) is a rewrite rule $l^\sharp \to \textsc{com}(r_1^\sharp, \dots, r_k^\sharp)$ where $l, r_1, \dots, r_k \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Let $\mathcal{S}$ and $\mathcal{W}$ be two TRSs over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and let $\mathcal{S}^\sharp$ and $\mathcal{W}^\sharp$ be two sets of dependency pairs. A *dependency pair complexity problem*, or simply *DP problem*, is a runtime complexity problem $\mathcal{P}^\sharp = \langle \mathcal{S}^\sharp \cup \mathcal{S} / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{Q}, \mathcal{T}^\sharp \rangle$ over marked basic terms $\mathcal{T}^\sharp$. We keep the convention that $\mathcal{S}^\sharp$ and $\mathcal{W}^\sharp$ denote dependency pairs. Our notion of a DP problem is general enough to capture images of the transformations proposed in the literature [5, 7] for polynomial complexity analysis, compare [1]. In the following, we suppose $\mathcal{S} = \varnothing$, i.e., the complexity function of $\mathcal{P}^\sharp$ accounts for dependency pairs only. We emphasise that for innermost runtime complexity analysis, TCT always constructs a DP problem of this shape, by either applying the *weightgap condition* [5] or using *dependency tuples* [7] only.

As a consequence of the following simple observation, the argument filtering employed in Theorem 3.1 has to fulfil, besides the non-collapsing condition on defined symbols of $\mathcal{S}^\sharp$, only mild conditions on compound symbols.

▶ **Lemma 4.1.** *Let $\mathcal{P}^\sharp = \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{Q}, \mathcal{T}^\sharp \rangle$ be a DP problem. Suppose μ denotes a usable replacement map for dependency pairs $\mathcal{S}^\sharp$ in $\mathcal{P}^\sharp$. Then $\mu_{\textsc{com}}$ is a usable replacement map for $\mathcal{S}^\sharp$ in $\mathcal{P}^\sharp$. Here $\mu_{\textsc{com}}$ denotes the restriction of μ to compound symbols in the following sense: $\mu_{\textsc{com}}(\mathsf{c}_n) := \mu(\mathsf{c}_n)$ for all $\mathsf{c}_n \in \mathcal{C}\mathsf{om}$, and otherwise $\mu_{\textsc{com}}(f) := \varnothing$ for $f \in \mathcal{F}^\sharp$.*

For DP problems, one can remove the non-collapsing condition on the employed argument filtering. The inferred complexity bound is less fine grained however.

▶ **Theorem 4.2.** *Let $\mathcal{P}^\sharp = \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{Q}, \mathcal{T}^\sharp \rangle$ be an innermost DP problem, where $\mathcal{S}^\sharp$, $\mathcal{W}^\sharp$ and $\mathcal{W}$ are constructor TRSs. Let μ denote a usable replacement map for $\mathcal{S}^\sharp \cup \mathcal{W}^\sharp$ in $\mathcal{P}^\sharp$, Let*

[2] See `http://termination-portal.org/wiki/Termination_Competition`.

π denote an argument filtering on the symbols in $\mathcal{P}$ that agrees with a usable replacement map for all dependency pairs in $\mathcal{P}^\sharp$. Let $\mathcal{K}_\pi \subseteq \mathcal{D}^\sharp_\pi$ denote a set of recursive function symbols, and $\gtrsim$ an admissible precedence where $c_\pi \sim d_\pi$ only holds for non-compound symbols $c, d \notin \mathsf{Com}$. The following processor is sound, for $d := \max\{0\} \cup \{\mathsf{rd}_{\gtrsim,\mathcal{K}_\pi}(f_\pi) \mid f_\pi \in \mathcal{F}^\sharp_\pi\}$.

$$\frac{\mathcal{S}^\sharp \subseteq >^{\pi}_{\mathsf{spop}^*_{\mathsf{ps}}} \quad \mathcal{W}^\sharp \cup \mathcal{W} \subseteq \gtrsim^{\pi}_{\mathsf{spop}^*_{\mathsf{ps}}}}{\vdash \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{Q}, \mathcal{T}^\sharp \rangle : n^{\max(1, 2\cdot d)}} \ .$$

We remark that the pre-conditions of the theorem are essential, and the estimated complexity is asymptotically optimal in general [1].

5 Conclusion

In this work we have outlined the implementation of sPOP* in TCT. We conclude with an empirical evaluation of this method. In Table 1 we contrast sPOP* to *matrix interpretations* (MI for short). Here the subscript DP denotes that the input is first transformed into a DP problem and syntactically simplified, compare [1, Section 14.5]. As testbed we used the 757 well-formed constructor TRSs from the TPDB 8.0.[3]

bound	sPOP*	sPOP$^\star_{\mathsf{DP}}$	MI$_{\mathsf{DP}}$
$\mathcal{O}(1)$	4\0.17	20\0.28	20\0.27
$\mathcal{O}(n^1)$	20\0.17	72\0.31	98\0.48
$\mathcal{O}(n^2)$	23\0.19	11\0.44	17\4.67
$\mathcal{O}(n^3)$	6\0.23	3\0.60	8\14.7
total	54\0.19	106\0.32	143\1.55
maybe	703\0.34	652\1.20	613\18.3

Table 1 Number of oriented problems and average execution time (secs.)

Comparing sPOP* and sPOP$^\star_{\mathsf{DP}}$ we see a significant increase in precision and power. This can be attributed to the relaxed conditions on the employed argument filtering. On the testbed, sPOP$^\star_{\mathsf{DP}}$ cannot cope in power with MI$_{\mathsf{DP}}$, but the average execution time of sPOP$^\star_{\mathsf{DP}}$ is significantly lower. Worthy of note, sPOP$^\star_{\mathsf{DP}}$ and MI$_{\mathsf{DP}}$ are incomparable. Their combination can handle 149 examples.

References

1. M. Avanzini. *Verifying Polytime Computability Automatically.* PhD thesis, University of Innsbruck, Institute for Computer Science, 2013. Submitted. Available at `http://cl-informatik.uibk.ac.at/~zini/publications/`.
2. M. Avanzini, N. Eguchi, and G. Moser. A New Order-theoretic Characterisation of the Polytime Computable Functions. In *Proc. of 10th APLAS*, volume 7705 of *LNCS*, pages 280–295, 2012.
3. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge UP, 1998.
4. S. Bellantoni and S. Cook. A new Recursion-Theoretic Characterization of the Polytime Functions. *Computational Complexity*, 2(2):97–110, 1992.
5. N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. In *Proc. of 4th IJCAR*, volume 5195 of *LNAI*, pages 364–380, 2008.
6. N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. 2012. To appear.
7. L. Noschinski, F. Emmes, and J. Giesl. A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems. In *Proc. of 23rd CADE*, volume 6803 of *LNAI*, pages 422–438. Springer, 2011.

[3] See `http://cl-informatik.uibk.ac.at/software/tct/experiments/wst2013` for full experimental evidence and explanation on the setup.