Theoretical Computer Science 585 (2015) 3–24

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A new order-theoretic characterisation of the polytime computable functions [☆]

Martin Avanzini [*], Naohi Eguchi, Georg Moser

Institute of Computer Science, University of Innsbruck, Austria

ARTICLE INFO

Article history:
Available online 9 March 2015

Keywords:
Term rewriting
Complexity analysis
Implicit computational complexity
Automation

ABSTRACT

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

1. Introduction

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

Thus we obtain a direct correspondence between a syntactic (and easily verifiable) condition of a program and the asymptotic worst-case complexity of the program. In this sense our work is closely related to similar studies in the field of *implicit computational complexity* (*ICC* for short). On the other hand the order sPOP* entails a new syntactic criteria to automatically establish polynomial runtime complexity of a given TRS. This criteria extends the state of the art in runtime complexity analysis as it is more precise or more efficient than related techniques. Note that the proposed syntactic method to analyse the (innermost) runtime complexity of rewrite systems is fully automatic. For any given TRS, compatibility with sPOP* can be efficiently checked by a machine. Should this check succeed, we get an asymptotic bound on the runtime

[☆] This work has been partially supported by the Austrian Science Fund, project number I-603-N18, by the John Templeton Foundation, grant number 20442, and the Japan Society for the Promotion of Science, project number 25·726.
[*] Corresponding author.
E-mail addresses: martin.avanzini@uibk.ac.at (M. Avanzini), naohi.eguchi@uibk.ac.at (N. Eguchi), georg.moser@uibk.ac.at (G. Moser).

http://dx.doi.org/10.1016/j.tcs.2015.03.003

complexity directly from the parameters of the order. It should perhaps be emphasised that compatibility of a TRS with sPOP* implies termination and thus our complexity analysis technique does not presuppose termination.

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

- We propose a new *recursion-theoretic characterisation* $\mathcal{B}_{\mathsf{wsc}}$ over binary words of the class FP. We establish that those $\mathcal{B}_{\mathsf{wsc}}$ functions that are definable with d nestings of predicative recursion can be computed by predicative recursive TRSs of degree d (cf. Theorem 4). Note that these functions belong to $\mathsf{DTIME}(n^d)$.
- We propose the new termination order sPOP*; sPOP* captures the recursion-theoretic principles of the class $\mathcal{B}_{\mathsf{wsc}}$. Thus we obtain a new *order-theoretic characterisation* of the class FP. Moreover, for any predicative recursive TRS of degree d its runtime complexity lies in $O(n^d)$ (cf. Theorem 1). Furthermore this bound is tight, that is, we provide a family of TRSs, delineated by sPOP*, whose runtime complexity is bounded from below by $\Omega(n^d)$, cf. Example 5.
- We extend upon sPOP* by proposing a generalisation, denoted $\mathsf{sPOP}^*_{\mathsf{PS}}$, admitting the same properties as above. This generalisations incorporates a more general recursion scheme that makes use of *parameter substitution* (cf. Theorem 2).
- We establish a novel, order-theoretic characterisation of $\mathsf{DTIME}(n^d)$. We show that $\mathsf{DTIME}(n^d)$ corresponds to the class of functions computable by *tail-recursive* predicative TRSs of degree d. This characterisation is based on the generalised small polynomial path order $\mathsf{sPOP}^*_{\mathsf{PS}}$ (cf. Theorem 6).
- sPOP* gives rise to a new syntactic method for *polynomial runtime complexity method*. This method is fully automatic. We have implemented the order sPOP* in the *Tyrolean Complexity Tool* TCT, version 2.0, an open source complexity analyser [2]. The experimental evidence obtained indicates the efficiency of the method and the obtained increase in precision.

1.1. Related work

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

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

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

In Bonfante et al. [10] restricted classes of polynomial interpretations are studied that can be employed to obtain polynomial upper bounds on the runtime complexity of TRSs. Polynomial interpretations are complemented with quasi-interpretations in [11], giving rise to alternative characterisations of complexity classes. None of the above results are applicable to relate the depth of recursion to the runtime complexity, in the sense mentioned above. Furthermore it is unknown how the body of work on quasi-interpretations can be employed in the context of runtime complexity analysis. We have also drawn motivation from Leivant's and Marion's characterisations of $\mathsf{DTIME}(n^d)$ [12,13], that provide related fine-grained classification of the polytime computable functions. Again, these results lack applicability in the context of runtime complexity analysis.

Polynomial complexity analysis is an active research area in rewriting. Starting from [14] interest in this field greatly increased over the last years, see for example [15–18] and [19] for an overview. This is partly due to the incorporation of a dedicated category for complexity into the annual termination competition (TERMCOMP).[1] However, it is worth emphasising that the most powerful techniques for runtime complexity analysis currently available, basically employ semantic considerations on the rewrite systems, which are notoriously inefficient.

We also want to mention ongoing approaches for the automated analysis of resource usage in programs. Notably, Hoffmann et al. [20] provide an automatic multivariate amortised cost analysis exploiting typing, which extends earlier results on amortised cost analysis. Finally Albert et al. [21] present an automated complexity tool for Java Bytecode programs, Alias et al. [22] give a complexity and termination analysis for flowchart programs, and Gulwani et al. [23] as well as Zuleger et al. [24] provide an automated complexity tool for C programs.

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