A New Term Rewriting Characterisation of ETIME functions

Martin Avanzini and Naohi Eguchi*

Institute of Computer Science, University of Innsbruck, Austria
{martin.avanzini,naohi.eguchi}@uibk.ac.at

Abstract. Adopting former term rewriting characterisations of polytime and exponential-time computable functions, we introduce a new reduction order, the *Path Order for ETIME* (POE* for short), that is sound and complete for ETIME computable functions. The proposed reduction order for ETIME makes contrasts to those related complexity classes clear.

1 Introduction

Function-algebraic approaches to computational complexity classes without explicit bounding constraints have been developed, providing successful characterisations of various complexity classes of functions as the smallest classes containing certain initial functions closed under specific operations. Initially, S. Bellantoni and S. Cook introduced a restrictive form of primitive recursion known as *safe recursion* [7], or independently D. Leivant introduced *tiered recursion* [10], characterising the class of polynomial-time computable functions. The idea of safe recursion is to separate the arguments of every function into two kinds (by semicolon) so that the number of recursive calls is measured only by an argument occurring left to semicolon whereas recursion terms are substituted only for arguments occurring right:

$$\begin{aligned} f(\varepsilon, \vec{y}\,; \vec{z}) &= g(\vec{y}\,; \vec{z}) \\ f(x \cdot i, \vec{y}\,; \vec{z}) &= h(x, \vec{y}\,; \vec{z}, f(x, \vec{y}\,; \vec{z})) \qquad (i = 0, 1) \end{aligned} \tag{\textbf{Safe Recursion}}$$

In contrast to classical approaches based on bounded recursion, the function-algebraic characterisation by safe recursion enables us to define every polytime function by a purely equational system, or in other word by a term rewrite system. Improving the function-algebraic characterisation by S. Bellantoni and S. Cook, together with G. Moser the authors introduced the *(small) polynomial path order* (sPOP*) [3] that constitutes an *order-theoretic* characterisation of the polytime functions. In the present work, we introduce a syntactic extension of sPOP*, the *Path Order for ETIME* (POE* for short). This order characterises the class *FETIME* of *ETIME* computable functions, i.e., functions computable in deterministic time $2^{\mathsf{O}(n)}$.

* The first author is supported by the FWF (Austrian Science Fund) project I-608-N18. The second author is supported by JSPS posdoctoral fellowships for young scientists.

2 Function-algebraic Backgrounds

Various function-algebraic characterisations of the ETIME functions are known, e.g. [11,8]. It is also known that extension of safe recursion to (multiple) nested recursion, called *safe nested recursion*, captures the class of exponential-time computable functions [1]. Improving the function-algebraic characterisation by safe nested recursion, the authors together with G. Moser have introduced an order, the *Exponential Path Order* (EPO*), that is sound and complete for the exponential-time functions. The order proposed here is a syntactic restriction of EPO*.

It turns out that the following form of safe nested recursion with single recursion arguments is sound for ETIME functions.

$$\begin{aligned} f(\varepsilon, \vec{y}\,; \vec{z}) &= g(\vec{y}\,; \vec{z}) \\ f(x \cdot i, \vec{y}\,; \vec{z}) &= h(x, \vec{y}\,; f(x, \vec{y}\,; \vec{h'}(x, \vec{y}\,; \vec{z}, f(x, \vec{y}\,; \vec{z})))) \qquad (i = 0, 1) \end{aligned}$$

The definition of POE* essentially encodes this recursion scheme. In contrast to related work, this scheme does neither rely on *bounded functions* [11] and allows the definition of functions that grow faster than a linear polynomial [8].

3 The Path Order for ETIME (POE*)

We assume at least nodding acquaintance with the basics of term rewriting [6]. For an order $>$, we denote by $>^{\mathsf{prod}}$ the *product extension* of $>$ defined by $\langle a_1, \ldots, a_k\rangle >^{\mathsf{prod}} \langle b_1, \ldots, b_k\rangle$ if $a_i = b_i$ or $a_i > b_i$ for all $i = 1, \ldots, k$, and there exists at least one $j \in \{1, \ldots, k\}$ such that $a_j > b_j$ holds.

We fix a countably infinite set of *variables* $\mathcal{V}$ and a finite set of *function symbols* $\mathcal{F}$, the *signature*. The set of terms formed from $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The signature $\mathcal{F}$ contains a distinguished set of *constructors* $\mathcal{C}$, elements of $\mathcal{T}(\mathcal{C})$ are called *values*. Elements of $\mathcal{F}$ that are not constructors are called *defined symbols* and collected in $\mathcal{D}$. We use always v to denote values, and arbitrary terms are denoted by l, r and $s, t, \ldots$, possibly followed by subscripts. A *substitution* σ is a finite mapping from variables to terms, its homomorphic extension to terms is also denoted by σ and we write $t\sigma$ instead of $\sigma(t)$.

A *term rewrite system* (*TRS* for short) $\mathcal{R}$ (over $\mathcal{F}$) is a finite set of rewrite rules $f(l_1, \ldots, l_n) \to r$, where all variable in the term r occur in the term $f(l_1, \ldots, l_n)$ and $f \in \mathcal{D}$. Adopting *call-by-value* semantics, we define the *rewrite relation* $\to_{\mathcal{R}}$ by

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

Throughout the present notes we only consider *completely defined*,[1] *orthogonal constructor* TRSs [6], that is, for each application of (i) there is exactly

[1] The restriction is not necessary, but simplifies our presentation.

one matching rule $f(l_1,\dots,l_n) \to r \in \mathcal{R}$; the terms l_i $(i = 1,\dots,n)$ contains no defined symbols and variables occur only once in $f(l_1,\dots,l_n)$.

For each defined symbol f of arity k, $\mathcal{R}$ defines a function $[\![f]\!] : \mathcal{T}(\mathcal{C})^k \to \mathcal{T}(\mathcal{C})$ by $[\![f]\!](v_1,\dots,v_k) := v$ iff $f(v_1,\dots,v_k) \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} v$. These functions are well-defined if $\mathcal{R}$ *terminating*, i.e., when $\to_{\mathcal{R}}$ is well-founded.

For a term t, the *size* of t is denoted as $|t|$ referring to the number of symbols occurring in t. For a complexity measure for TRSs, the *(innermost) runtime complexity function* $\mathrm{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N}$ is defined by

$$\mathrm{rc}_{\mathcal{R}}(n) := \max\{\ell \mid \exists s = f(v_1,\dots,v_n), |s| \leqslant n \text{ and } s = t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_\ell\}\,,$$

which is well-defined for terminating TRSs $\mathcal{R}$. The runtime-complexity function constitutes an *invariant cost-model* for rewrite systems: the functions $[\![f]\!]$ $(f \in \mathcal{D})$ can be computed within polynomial overhead on conventional models of computation, e.g., on Turing machines [9,5].

Let $>$ denote a strict order on $\mathcal{F}$, the *precedence*. We assume that the argument positions of every function symbol are separated into two kinds. The separation is denoted by semicolon as $f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$, where $t_1,\dots,t_k$ are called *normal* arguments whereas $t_{k+1},\dots,t_{k+l}$ are called *safe* ones. For constructors $\mathcal{C}$, we suppose that all symbols are safe. We write $s \rhd^{\mathrm{n}} t$ if t is a *sub-term of a normal argument* of s, i.e., $s = f(s_1,\dots,s_k\,;s_{k+1},\dots,s_{k+l})$ and t occurs in a term s_i for $i \in \{1,\dots,k\}$. The following definition introduces the order POE^*.

Definition 1. *Let $>$ be a precedence and let $s = f(s_1,\dots,s_k\,;s_{k+1},\dots,s_{k+l})$. Then $s >_{\mathsf{poe}^*} t$ if one of the following alternatives holds.*

1. *$s_i \geqslant_{\mathsf{poe}*} t$ for some $i \in \{1,\dots,k+l\}$.*
2. *$f \in \mathcal{D}$ and $t = g(t_1,\dots,t_m\,;t_{m+1},\dots,t_{m+n})$ with $f > g$ and:*
   - *$s \rhd^{n} t_j$ for all $j \in \{1,\dots,m\}$;*
   - *$s >_{\mathsf{poe}^*} t_j$ for all $j \in \{m+1,\dots,m+n\}$;*
3. *$f \in \mathcal{D}$ and $t = f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$ and:*
   - *$\langle s_1,\dots,s_k\rangle >^{\mathsf{prod}}_{\mathsf{poe}^*} \langle t_1,\dots,t_k\rangle$*
   - *$s >_{\mathsf{poe}^*} t_j$ for all $j \in \{k+1,\dots,k+l\}$.*

We call a TRS $\mathcal{R}$ *POE*-compatible* if $\mathcal{R} \subseteq >_{\mathsf{poe}^*}$ holds for some $>_{\mathsf{poe}^*}$.

Example 1. The standard addition $(x, y) \mapsto x + y$ (in unary notation) is defined by a TRS $\mathcal{R}_{\mathsf{add}}$ consisting of the following two rules.

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

Define a precedence by $\mathsf{add} > \mathsf{s}$ and an argument separation as indicated in the rules. Then it can be seen that $\mathsf{add}(\mathsf{0}\,;y) >_{\mathsf{poe}^*} y$ and $\mathsf{add}(\mathsf{s}(\,;x)\,;y) >_{\mathsf{poe}^*} \mathsf{s}(\,;\mathsf{add}(x\,;y))$ hold for the order $>_{\mathsf{poe}^*}$ induced by the precedence $>$.

Example 2. Consider the TRS $\mathcal{R}_{\mathsf{exp}}$ consisting of the following two rules.

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

The TRS $\mathcal{R}_{\mathsf{exp}}$ computes the exponential $2^x + y$ (in unary notation) and is compatible with the order $>_{\mathsf{poe}^*}$ using the precedence $\mathsf{exp} > \mathsf{s}$.

Example 3. A factorial function of the form $y \cdot x! + z$ is defined by a TRS $\mathcal{R}_{\mathsf{fac}}$ consisting of $\mathcal{R}_{\mathsf{add}}$ and additionally of the following three rules.

$$3: \quad \mathsf{fac}(\mathsf{0}, y\,;z) \to \mathsf{add}(y\,;z) \qquad\qquad 4: \mathsf{fac}(\mathsf{s}(\,;x), \mathsf{0}\,;z) \to z$$
$$5: \mathsf{fac}(\mathsf{s}(\,;x), \mathsf{s}(\,;y)\,;z) \to \mathsf{fac}(\mathsf{s}(\,;x), y\,;\mathsf{fac}(x, \mathsf{s}(\,;x)\,;z))$$

The TRS $\mathcal{R}_{\mathsf{fac}}$ is not compatible with any POE*. Here, rule 5 is not orientable since the element-wise comparison of $\langle \mathsf{s}(\,;x), \mathsf{s}(\,;y)\rangle$ and $\langle x, \mathsf{s}(\,;x)\rangle$ fails.

By the following Theorem, it is not surprising that $\mathcal{R}_{\mathsf{fac}}$ is not POE*-compatible.

Theorem 1 (Soundness of POE*for FETIME). *Every function defined by a POE*-compatible rewrite system is in FETIME.*

This theorem follows from the following key lemma whose proof is involved and hence kindly referred to a technical report [4].

Lemma 1. *For any POE*-compatible rewrite system $\mathcal{R}$, $\mathrm{rc}_{\mathcal{R}}(n) \in 2^{\mathsf{O}(n)}$ holds.*

Hence, our technique has also applications in the runtime-complexity analysis of TRSs. We emphasise that a compatible order $>_{\mathsf{poe}^*}$ can be computed in non-deterministic polynomial time in the size of the input TRS $\mathcal{R}$ (i.e, number of symbols in $\mathcal{R}$), simply by guessing a suitable precedence and separation of argument positions and checking the constraints imposed by Definition 1. We conjecture, that a compatible $>_{\mathsf{poe}^*}$ can even be computed in polynomial time.

Although the inverse of Lemma 1 is in general not true, the order is also extensionally complete for FETIME.

Theorem 2 (Completeness of POE* for FETIME). *Every ETIME function can be defined by a POE*-compatible rewrite system.*

Proof (Sketch). Consider words formed from dyadic successors $\mathsf{0}$ and $\mathsf{1}$ together with a constant ϵ, denoting the empty word. The following rewrite rules

$$\mathsf{f}_1(\epsilon\,;u) \to \mathsf{step}(\,;u) \qquad \mathsf{f}_1(\mathsf{i}(\,;x)\,;u) \to \mathsf{f}_1(x\,;\mathsf{f}_0(x\,;u)) \quad (\text{for } \mathsf{i} \in \{\mathsf{0},\mathsf{1}\})\,,$$

define a function $[\![f]\!]_1(w\,;c) = [\![\mathsf{step}]\!]^{2^{|w|}}(\,;c)$, i.e., $2^{|w|}$-fold iteration of $[\![\mathsf{step}]\!]$. Here, we suppose that $|w|$ counts the number of occurrences of $\mathsf{0}$ and $\mathsf{1}$ in w. Next consider the following rewrite rules.

$$\mathsf{f}_2(\epsilon, y\,;u) \to \mathsf{f}_1(y\,;u) \qquad \mathsf{f}_2(\mathsf{i}(\,;x), y\,;u) \to \mathsf{f}_2(x, y\,;\mathsf{f}_2(x, y\,;u)) \quad (\text{for } \mathsf{i} \in \{\mathsf{0},\mathsf{1}\})\,.$$

Then $[\![\mathsf{f}_2]\!](w, w\,;u) = [\![\mathsf{step}]\!]^{2^{2\cdot|w|}}(\,;u)$. This construction can be extended to k functions such that $[\![\mathsf{f}_k]\!](w, \ldots, w\,;u) = [\![\mathsf{step}]\!]^{2^{k\cdot|w|}}(\,;u)$. Note that all rules can be oriented by $>_{\mathsf{poe}^*}$, given by the precedence $\mathsf{f}_k > \cdots > \mathsf{f}_1 > \mathsf{step}$.

Using this construction, it is possible to simulate ETIME Turing machines running in time $2^{\mathsf{O}(n)}$ by a POE*-compatible TRS, essentially by substituting the transition function for step. Note that step can be defined by pattern matching only, in particular it is easy to define step such that the underlying rewrite rules are POE*-compatible.

Corollary 1. *The class FETIME coincides with the class of functions computed by POE*-compatible rewrite systems.*

4 Conclusion

Adopting former works [2,3], we introduced a reduction order, the Path Order for ETIME (POE*), that is sound and complete for FETIME. The path order POE* is a strictly intermediate order between the (small) path order for polytime (sPOP*) and exponential path order (EPO*).

These orders differ only in constraints imposed on recursive definitions: POE* extends sPOP* by allowing nested recursive calls, as in the TRS $\mathcal{R}_{\mathsf{exp}}$; the order EPO* permits additionally recursion along lexicographic descending arguments, as in rule 5 of the TRS $\mathcal{R}_{\mathsf{fac}}$. Consequently, from our three examples only the TRS $\mathcal{R}_{\mathsf{add}}$ is compatible with sPOP*, whereas $\mathcal{R}_{\mathsf{add}}$ and $\mathcal{R}_{\mathsf{exp}}$ is compatible with POE* and EPO* can even handle $\mathcal{R}_{\mathsf{fac}}$.

This contrast clarifies the relationship FP $\subseteq$ FETIME $\subseteq$ FEXP for the class FP of polytime and the class FEXP of exponential time computable functions.

References

1. T. Arai and N. Eguchi. A New Function Algebra of EXPTIME Functions by Safe Nested Recursion. *ACM TCL*, 10(4), 2009. Article No. 24, 19 pages.
2. M. Avanzini, N. Eguchi, and G. Moser. A Path Order for Rewrite Systems that Compute Exponential Time Functions. In *Proc. of the 22nd RTA*, volume 10 of *LIPIcs*, pages 123–138, 2011.
3. M. Avanzini, N. Eguchi, and G. Moser. A New Order-theoretic Characterisation of the Polytime Computable Functions. In *Proc. of the 10th APLAS*, volume 7705 of *LNCS*, pages 280–295, 2012.
4. M. Avanzini and N. Eguchi. A New Term-rewriting Characterisation of ETIME Functions. *Technical report*. NE will make this report available at arXiv before submission.
5. M. Avanzini and G. Moser. Closing the Gap Between Runtime Complexity and Polytime Computability. In *Proc. of the 21st RTA* , volume 6 of *LIPIcs*, pages 33–48, 2010.
6. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambredge University Press, 1998.
7. S. Bellantoni and S. A. Cook. A New Recursion-theoretic Characterization of the Polytime Functions. *Computational Complexity*, 2(2):97–110, 1992.
8. P. Clote. A Safe Recursion Scheme for Exponential Time. In *Proc. of the 4th LFCS*, volume 1234 of *LNCS*, pages 44–52, 1997.
9. U. Dal Lago and S. Martini. On Constructor Rewrite Systems and the Lambda-Calculus. In *Proc. of the 36th ICALP*, volume 5556 of *LNCS*, pages 163–174, 2009.
10. D. Leivant. Ramified Recurrence and Computational Complexity I: Word Recurrence and Poly-time. In *Feasible Mathematics II, Progress in Computer Science and Applied Logic*, volume 13, pages 320–343. Birkhäuser Boston, 1995.
11. B. Monien. A Recursive and Grammatical Characterization of Exponential Time Languages. *Theoretical Computer Science*, 3:61–74, 1977.