Termination proofs and computational complexity classes

J.Y. Marion

Loria, Calligramme project,
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
Jean-Yves.Marion@loria.fr.

Abstract

Termination orderings on rewrite systems delineate classes of total functions such as the primitive recursive functions which are characterised by MPO [Hof92], multiple recursive functions [Pét66] which are characterised by LPO [Wei95]. Subrecursive hierarchy classifications has been extensively investigated by assigning ordinals to recursive definitions, see for example [BEW94,HW94]. The main issue is to express the computational complexity of a program from its termination proof. For this, we incorporate ideas involved in predicative analysis of recursive definitions which can be found in the works of [Sim88,BC92,Lei94] and also exploit more traditional methods from termination proofs. We consider here ways of specifying the computational complexity of algorithms which is an intensional property. On the other hand, the traditional meta-theory of program reasoning deals with the verification of extensional properties. The study of intensional properties of programs turns out to be much more challenging. For example, the class of all algorithms running, say, in polynomial time is not recursively enumerable with respect to an enumeration of partial recursive algorithms. But, imposing an intentional property on program constructions is a relevant question which is considered for example in Nuprl development as explained by Constable in [Con95]. One might wonder how relevant is a program whose proof termination is certified but runs in doubly exponential time.

We shall investigate the computational complexity of algorithms provided by rewrite systems with termination proofs based on polynomial and exponential interpretations. Termination proofs by polynomial interpretations were introduced by Lankford [Lan79]. Hofbauer and Lautemann [HL88] established that derivations admit a doubly exponential length. The background for our study really goes back to the work initiated by Cichon and Lescanne [CL92]. It was shown that a particularly important aspect was the interpretation of the constructors. More recently, we showed in [BCMT98] that the functions whose termination proofs involve a particular kind of successor interpretation, are exactly the functions computed in polynomial-time (PTIME), linear exponential time (ETIME $=$ DTIME$(2^{0(n)})$) and linear doubly exponential time (E$_2$TIME $=$ DTIME$(2^{2^{0(n)}})$).

While in [BCMT98] functions were just defined by a confluent system on strings, we now consider general rewrite systems with polynomial interpretations. Data structures are now lists, trees, ... So, we are somewhat closer

to real programming languages such as the ones designed in POLO [Gie95], ORME [Orm], LARCH [Lar]. Next, we present a notion of functions computed by non-confluent systems which appears in Krentel's work [Kre88] and which seems appropriate and robust as discussed by Grädel & Gurevich [GG95]. The whole purpose is to provide a basis whereby non-confluent rewrite systems are construed to non-deterministic computations. As in [BCMT98], we exhibit three kinds of polynomial interpretations of constructors. We analyse the computational complexity of algorithms with respect to the kind of interpretation given to constructors. Hence, we obtain three classes of functions which characterise exactly PTIME, ETIME, E_2TIME when systems are confluent. When systems are non-confluent, we capture the non-deterministic counterpart, that is the class of functions computable in non-deterministic polynomial time (NPTIME), non-deterministic exponential time (NETIME), and lastly non-deterministic doubly exponential time (NE_2TIME).

Machine independant characterisations of complexity classes were originated by Cobham [Cob62]. His approach is by means of 'bounded recursion on notation' in which rates of growth of functions are limited by functions already defined in the class. In contrast, in our work, polynomial interpretations just impose, via a reduction ordering, a local condition on each rewrite rule. In fact, the different kinds of polynomial interpretations of successors are somewhat akin to the notion of data tiering [BC92,Lei94]. Beckmann & Weiermann reformulate those ideas into a rewriting setting, see also [Cas96]. It is also worth mentioning the characterisation of PTIME functions over finite models in [Saz80,Gur83], since basically the same system is considered : the Herbrand-Gödel equations.

We briefly summarise the main results

n-ary constructor interpretation	Confluent	Non-Confluent
Kind 0 : $\sum_{i=1}^{n} X_i + \gamma$	PTIME	NPTIME
Kind 1 : $\sum_{i=1}^{n} \alpha_i X_i + \gamma$	ETIME	NETIME
Kind 2 : $\sum_{i=1}^{n} \alpha_i X_i^{\beta_i} + \gamma$	E_2TIME	NE_2TIME
Kind 3 : $\sum_{i=1}^{n} 2^{X_i} + \gamma$	DTIME($\exp_\infty$(n))	DTIME($\exp_\infty$(n))

where $\exp_\infty(0) = 2$ and $\exp_\infty(n+1) = 2^{\exp_\infty(n)}$

References

[BC92] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2:97–110, 1992.

[BCMT98] G. Bonfante, A. Cichon, J.Y Marion, and H. Touzet. Complexity classes and rewrite systems with polynomial interpretation. In *CSL*, 1998. Technical Report LORIA 98-R-060, `http://www.loria.fr/~marionjy`.

[BEW94] W. Buchholz, E.A.Cichon, and A. Weiermann. A uniform approach to fundamental sequences and hierarchies. *Mathematical Logic Quarterly*, 40, 1994.

[Cas96] V-H Caseiro. An equational characterization of the poly-time functions on any constructor data strucure. Technical Report 226, University of Oslo, Dept. of informatics, December 1996. `http://www.ifi.uio.no/~ftp/publications`.

[CL92] E.A Cichon and P. Lescanne. Polynomial interpretations and the complexity of algorithms. In *CADE'11*, pages 139–147, 1992.

[Cob62] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1962.

[Con95] R. Constable. Expressing computationnal complexity in constructive type theory. In D. Leivant, editor, *LCC'94*, number 960 in LNCS, pages 131–144, 1995.

[GG95] Erich Grädel and Yuri Gurevich. Tailoring recursion for complexity. *J. Symbolic Logic*, 60(3):952–969, Sept. 1995.

[Gie95] J. Giesl. Generating polynomial orderings for termination proofs. In *RTA*, number 914 in Lecture Notes in Computer Science, pages 427–431, 1995.

[Gur83] Y. Gurevich. Algebras of feasible functions. In *Twenty Fourth Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press, 1983.

[HL88] D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *RTA*, number 355 in Lecture Notes in Computer Science, 1988.

[Hof92] D. Hofbauer. Termination proofs with multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140, 1992.

[HW94] W.G. Handley and S.S. Wainer. Equational derivation vs. computation. *Annals of Pure and Applied Logic*, 70:17–49, 1994.

[Kre88] M. Krentel. The complexity of optimization problems. *Journal of computer and system sciences*, 36:490–519, 1988.

[Lan79] D.S. Lankford. On proving term rewriting systems are noetherien. Technical Report MTP-3, louisiana Technical University, 1979.

[Lar] *Larch*. `http://www.sds.lcs.mit.edu/spd/larch/index.html`.

[Lei94] Daniel Leivant. Predicative recurrence and computational complexity I: Word recurrence and poly-time. In Peter Clote and Jeffery Remmel, editors, *Feasible Mathematrics II*. Birkhäuser, 1994.

[Orm] *Orme*. `http://www.ens-lyon.fr/~plescann/publications.html`.

[Pét66] Rósza Péter. *Rekursive Funktionen*. Akadémiai Kiadó, Budapest, 1966. English translation: *Recursive Functions*, Academic Press, New York, 1967.

[Saz80] Vladimir Sazonov. Polynomial computability and recursivity in finite domains. *Electronische Informationsverarbeitung und Kybernetik*, 7:319–323, 1980.

[Sim88] Harold Simmons. The realm of primitive recursion. *Archive for Mathematical Logic*, 27:177–, 1988.

[Wei95] A. Weiermann. Termination proofs by lexicographic path orderings yield multiply recursive derivation lengths. *Theoretical Computer Science*, 139:335–362, 1995.