From Bounded Arithmetic to Memory Management: Use of Type Theory to Capture Complexity Classes and Space Behaviour

Martin Hofmann

Laboratory for the Foundations of Computer Science
Division of Informatics, University of Edinburgh

Bounded arithmetic [3] is a subsystem of Peano arithmetic defining exactly the polynomial time functions. As Gödel's system T corresponds to Peano arithmetic Cook and Urquhart's system PV_ω [4] corresponds to bounded arithmetic. It is a type system with the property that all definable functions are polynomial time computable.

PV_ω as a programming language for polynomial time is, however, unsatisfactory in several ways. Firstly, it requires to maintain explicit size bounds on intermediate results and secondly, many obviously polynomial time algorithms do not fit into the type system. The attempt to alleviate these restrictions has lead to a sequence of new type systems capturing various complexity classes (PTIME, PSPACE, EXPTIME, LINSPACE) without explicit reference to bounds. Among them are Cook-Bellantoni's [2] and Bellantoni-Niggl-Schwichtenberg's systems of safe recursion [1], tiered systems by Leivant and Marion [12,11], subsystems of Girard's linear logic [6,5], and various systems by myself [9,7,8].

The most recent work [10] has shown that one of these systems can be adapted to allow for explicit memory management including in-place update while still maintaining a functional semantics.

The talk will give a bird's eye overview of the above-mentioned calculi and then discuss in some more detail the recent applications to memory management. This will include recent yet unpublished results about the expressive power of higher-order linear functions and general recursion in the context of [10]. These results suggests that the expressive power equals $\bigcup_c \mathrm{DTIME}(2^{n^c})$.

References

1. S. Bellantoni, K.-H. Niggl, and H. Schwichtenberg. Ramification, Modality, and Linearity in Higher Type Recursion. *Annals of Pure and Applied Logic*, 2000. to appear.
2. Stephen Bellantoni and Stephen Cook. New recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.
3. Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.
4. S. Cook and A. Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63:103–200, 1993.
5. J.-Y. Girard. Light Linear Logic. *Information and Computation*, 143, 1998.
6. J.-Y. Girard, A. Scedrov, and P. Scott. Bounded linear logic. *Theoretical Computer Science*, 97(1):1–66, 1992.

S. Abramsky (Ed.): TLCA 2001, LNCS 2044, pp. 2–3, 2001.

7. Martin Hofmann. Linear types and non size-increasing polynomial time computation. To appear in Theoretical Computer Science. See www.dcs.ed.ac.uk/home/papers/icc.ps.gz for a draft. An extended abstract has appeared under the same title in Proc. Symp. Logic in Comp. Sci. (LICS) 1999, Trento, 2000.
8. Martin Hofmann. Programming languages capturing complexity classes. *SIGACT News Logic Column*, 9, 2000. 12 pp.
9. Martin Hofmann. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 104:113–166, 2000.
10. Martin Hofmann. A type system for bounded space and functional in-place update. *Nordic Journal of Computing*, 2001. To appear, see www.dcs.ed.ac.uk/home/mxh/papers/nordic.ps.gz for a draft. An extended abstract has appeared in *Programming Languages and Systems*, G. Smolka, ed., Springer LNCS, 2000.
11. D. Leivant and J.-Y. Marion. Predicative Functional Recurrence and Poly-Space. In *Springer LNCS 1214: Proc. CAAP*, 1997.
12. Daniel Leivant. Stratified Functional Programs and Computational Complexity. In *Proc. 20th IEEE Symp. on Principles of Programming Languages*, 1993.