Complexity Analysis in Presence of Control Operators and Higher-Order Functions.

Ugo Dal Lago Giulio Pellitta

Abstract

A polarized version of Girard, Scedrov and Scott's Bounded Linear Logic is introduced and its normalization properties studied. Following Laurent [7], the logic naturally gives rise to a type system for the $\lambda\mu$-calculus, whose derivations reveal bounds on the time complexity of the underlying term.

1 Introduction

Among non-functional properties of programs, bounds on the amount of resources (like computation time and space) programs need when executed are particularly significant. The problem of deriving such bounds is indeed crucial in safety-critical systems, but is undecidable when considering non-trivial programming languages.

A different approach consists in analysing the *abstract* complexity of programs. E.g., one can take the number of instructions executed by the program as a measure of its execution time. One advantage of this analysis is the independence from the specific hardware platform executing the program, which only needs to be analysed once. Properties of programs written in higher-order functional languages are for various reasons well-suited to be verified by way of type systems. This includes not only safety properties (e.g. well-typed programs do not go wrong), but more complex ones, including resource bounds [6, 1, 2].

In this work, we delineate a methodology for complexity analysis of higher-order programs *with control operators*. The latter are constructs which are available in most concrete functional programming languages (including Scheme and OCaml), and allow control to flow in non-standard ways. We introduce a type system for de Groote's $\lambda\mu$-calculus [4] derived from Bounded Linear Logic [5]. We prove it to be sound: typable programs can indeed be reduced in a number of steps lesser or equal to a (polynomial) bound which can be read from the underlying type derivation. A similar result can be given when the cost model is the one induced by an abstract machine. To the authors' knowledge, this is the first example of a complexity analysis methodology coping well not only with higher-order functions, but also with control operators.

References

[1] P. Baillot and K. Terui. Light types for polynomial time computation in lambda-calculus. *Information and Computation*, 207(1):41–62, 2009.

[2] U. Dal Lago and M. Gaboardi. Linear dependent types and relative completeness. *Logical Methods in Computer Science*, 8(4), 2012.

[3] U. Dal Lago and G. Pellitta. Complexity analysis in presence of control operators and higher-order functions. In *LPAR-19, Proceedings*, pages 258–273. Springer, 2013.

[4] P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In *Logic Programming and Automated Reasoning*, pages 31–43. Springer, 1994.

[5] J.-Y. Girard, A. Scedrov, and P. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theoretical Computer Science*, 97(1):1–66, 1992.

[6] S. Jost, K. Hammond, H.-W. Loidl, and M. Hofmann. Static determination of quantitative resource usage for higher-order programs. In *POPL*, Madrid, Spain, 2010. ACM Press.

[7] O. Laurent. Polarized proof-nets and $\lambda\mu$-calculus. *Theoretical Computer Science*, 290(1):161–188, 2003.