J. Functional Programming **11** (1): 1–2, January 2001. Printed in the United Kingdom

Special Issue on Functional Programming and Computational Complexity

Editorial

This issue of the *Journal of Functional Programming* is dedicated to work presented at the Workshop on Implicit Computational Complexity in Programming Languages, affiliated with the 1998 meeting of the International Conference on Functional Programming in Baltimore.

Several machine-independent approaches to computational complexity have been developed in recent years; they establish a correspondence linking computational complexity to conceptual and structural measures of complexity of declarative programs and of formulas, proofs and models of formal theories. Examples include descriptive complexity of finite models, restrictions on induction in arithmetic and related first order theories, complexity of set-existence principles in higher order logic, and specifications in linear logic. We refer to these approaches collectively as Implicit Computational Complexity. This line of research provides a framework for a streamlined incorporation of computational complexity into areas such as formal methods in software development, programming language theory, and database theory.

A fruitful thread in implicit computational complexity is based on exploring the computational complexity consequences of introducing various syntactic control mechanisms in functional programming, including restrictions (akin to static typing) on scoping, data re-use (via linear modalities), and iteration (via ramification of data). These forms of control, separately and in combination, can certify bounds on the time and space resources used by programs. In fact, all results in this area establish that each restriction considered yields precisely a major computational complexity class. The complexity classes thus obtained range from very restricted ones, such as NC and Alternating logarithmic time, through the central classes Poly-Time and Poly-Space, to broad classes such as the Elementary and the Primitive Recursive functions.

Considerable effort has been invested in recent years to relax as much as possible the structural restrictions considered, allowing for more flexible programming and proof styles, while still guaranteeing the same resource bounds. Notably, more flexible control forms have been developed for certifying that functional programs execute in Poly-Time.

The 1998 workshop covered both the theoretical foundations of the field and steps toward using its results in various implemented systems, for example in controlling the computational complexity of programs extracted from constructive proofs. The five papers included in this issue nicely represent this dual concern of theory and practice. As they are going to print, we should note that the field of Implicit Computational Complexity continues to thrive: successful workshops dedicated to it were affiliated with both the LICS'99 and LICS'00 conferences. Special issues, of Information and Computation dedicated to the former, and of Theoretical Computer Science to the latter, are in preparation.

edited by
Daniel Leivant
Dept. of Computer Science
Indiana University

and Bob Constable
Dept. of Computer Science
Cornell University