JFP **22** (4–5): 379–381, 2012. © Cambridge University Press 2012
doi:10.1017/S0956796812000287

Editorial

Special issue dedicated to ICFP 2010

UMUT A. ACAR
Programming Languages and Systems Group, Max Planck Institute for Software Systems, Germany
(*e-mail:* umut@mpi-sws.org)

JAMES CHENEY
Laboratory for Foundations of Computer Science, University of Edinburgh, Edinburgh, UK
(*e-mail:* jcheney@inf.ed.ac.uk)

STEPHANIE WEIRICH
School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA 19104, USA
(*e-mail:* sweirich@cis.upenn.edu)

The 15th ACM SIGPLAN International Conference on Functional Programming (ICFP) took place on September 27–29, 2010 in Baltimore, Maryland. After the conference, the programme committee, chaired by Stephanie Weirich, selected several outstanding papers and invited their authors to submit to this special issue of *Journal of Functional Programming*. Umut A. Acar and James Cheney acted as editors for these submissions. This issue includes the seven accepted papers, each of which provides substantial new material beyond the original conference version. The selected papers reflect a consensus by the program committee that ICFP 2010 had a number of strong papers that link core functional programming ideas with other areas, such as multicore, embedded systems, and data compression.

In "Lazy tree splitting," Bergstrom *et al.* adapt a technique for parallelisation called *lazy binary splitting* to a functional, nested data parallel language called Parallel ML. A key advantage of this approach is that performance is robust, in that good performance does not depend on tuning parameters such as block size. This represents a significant contribution to the use of functional programming languages for programming multicore architectures.

In "Fortifying macros," Culpepper describes a new macro system for Scheme-like languages. The approach allows the programmer to specify macros that generate syntactic transformations and error-detecting codes, which help to discover and report programming mistakes. In practice, the new system can reduce the need for ad hoc validation and error-reporting code, improving the conciseness, robustness, and maintainability of macros.

In "The impact of higher-order state and control effects on local relational reasoning," Dreyer, Neis, and Birkedal investigate the tradeoffs between the power of the programming language and the properties that can be proved using local relational reasoning. Their goal is to understand when two programs are equivalent, to be used, for example, to reason about the correctness of program transformation.

This paper applies the modern technique of step-indexed Kripke logical relations to this old problem of reasoning about program equivalence in the presence of expressive language constructs, such as recursive types, abstract types, general references, and continuation objects. It also discusses how reasoning ability changes under restrictions to the language, giving example properties that are true only if the language lacks higher-order state or control effects such as throwing a continuation. Although the mathematical machinery necessary for this work is highly technical, much of the presentation is given in an informal, pedagogic style based on clean "visual" proof sketches.

In "Every bit counts: the binary representation of types, data and programs," Kennedy and Vytiniotis propose question–answer games for binary encoding and decoding of typed data and typed programs. This paper takes critical advantage of types and functional programming to construct codes for λ-calculi, and proves that codes derived from games never encode two distinct values using the same code. The authors formalize relevant interesting properties of their proposed framework and prove them using the Coq proof assistant. This paper originally appeared in ICFP 2010 as a pearl but has been extended significantly to include a full formalization in Coq.

In "The Reduceron reconfigured and reevaluated," Naylor and Runciman present an approach to compilation from a lazy functional language to the Reduceron, a Field-Programmable Gate Array (FPGA) implementation of a processor specifically designed to implement graph reduction. A particular highlight of this paper is its detailed comparison with other approaches to functional language implementation on special-purpose hardware, including a direct comparison that shows that the Reduceron (running at 96 MHz) offers an order of magnitude speedup compared to compilation to a conventional, pipelined RISC architecture (running on the same FPGA at 210 MHz), and is only four times slower than conventional compiled code running at 3 GHz. These results are timely in that future developments in computer architecture may rely upon increasing performance per clock cycle in order to conserve energy.

In "A unified treatment of syntax with binders," Poulliard and Pottier present a new technique for programming with syntax involving bound names in Agda or other dependently typed languages. The paper combines ideas from their ICFP 2010 paper and Poulliard's subsequent ICFP 2011 paper, which improved and extended the first paper. A central idea is to index the types of terms with *worlds*, which allows both nominal and de Bruijn implementations while facilitating a proof of correctness for each approach using logical relations involving renamings.

In "Systematic abstraction of abstract machines," Van Horn and Might propose a methodology for transforming abstract machines into sound abstract interpretations, making it possible to systematically develop static analyses for programming languages. The process maps the abstract machine into a non-deterministic fine-state transition systems by a series of refactorings that thread operations such as variable look-up and continuation-retrieval through a finite store. The papers shows

the effectiveness of the techniques by considering several abstract machines for higher-order programming languages.

We thank the authors and referees of these papers for their efforts producing and reviewing these papers within the strict time limits imposed by the special issue publication constraints. We also gratefully acknowledge the support of the JFP editors-in-chief and editorial office.