doi:10.1017/S0956796810000250

Special Issue Dedicated to ICFP 2008 Editorial

PETER THIEMANN
Institut für Informatik, Universität Freiburg
Georges-Köhler-Allee 079, 79110 Freiburg i. Br., Germany
(*e-mail:* thiemann@acm.org)

HENRIK NILSSON
School of Computer Science, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
(*e-mail:* nhn@cs.nott.ac.uk)

The 13th ACM SIGPLAN International Conference on Functional Programming (ICFP) was held in Victoria, British Columbia, Canada, in September 2008. Peter Thiemann chaired the program committee. After the conference, the authors of a selection of the presented papers were invited to submit extended versions of their work for this special issue of *Journal of Functional Programming* dedicated to ICFP 2008. All submitted papers were reviewed by at least three referees, including at least one expert, following the standard JFP procedures. In the end, four papers were accepted. These cover a broad range of topics and, taken together, we think they represent well the scope of ICFP 2008.

Obtaining a meaningful space profile for a functional program is hard. It is even harder for parallel functional programs. In their paper *Space profiling for parallel functional programs*, Daniel Spoonhower, Guy Blelloch, Bob Harper and Phillip Gibbons pick up this challenge and present a semantics-based space profiler for parallel functional programs on shared memory multiprocessors. The underlying cost semantics enables their profiler to obtain results that are independent of implementation details of, e.g. the run-time system. At the same time, the profiler reveals that the choice of a scheduling policy can make an asymptotic difference for space consumption. This paper provides valuable insights for anyone interested in the formalisation of space usage in (parallel) functional programs and in applying such formalisations in practical tools. It, thus, contributes to both theory and practise.

In his paper *Concrete stream calculus*, Ralf Hinze explores how the fact that suitably restricted equations on streams (infinite sequences of elements) having unique solutions can be exploited to develop a straightforward technique for proving properties about streams and stream programs. When applicable, this proof technique is an attractive alternative to coinduction as it boils down to equational reasoning, which likely is more familiar to many than the coinductive treatment. The development retains a practical focus by being carried out in Haskell. This paper is therefore a great starting point for people who want to learn about an elegant approach to stream programming and how to reason formally about such programs, as well as to those who have a general interest in reasoning about coinductive data types and corecursive programs.

A major hardware trend over the past decade has been that parallel processing capabilities, through multi-core processors, hyper threading and the like, have become commonplace even for low-end systems dedicated to irregular, everyday tasks, as opposed to high-end systems for heavy scientific computation or other specialist domains with a fairly regular problem structure. However, the design of languages suitable for fully reaping the benefits of this development is arguably lagging behind. In their paper *Implicitly-threaded parallelism in manticore*, Matthew Fluet, Mike Rainey, John Reppy and Adam Shaw discuss a new functional programming language that employs a number of novel language features aimed at filling this gap. The paper focuses on the implicitly threaded parallel aspects of the language and then in particular on those that clearly distinguish it from other parallel language designs: a novel parallel binding form, a non-deterministic parallel case and exceptions in the presence of data parallelism. Detailed examples illustrate the discussed language features, and the paper also covers the employed implementation strategies. This paper should thus appeal to a broad audience, from those who are interested in parallel programming in a functional setting to language implementers.

In the final paper, *NixOS: a purely functional linux distribution*, Eelco Dolstra, Andres Löh and Nicolas Pierron describe how functional programming principles can be applied to system configuration to address hard, real-world problems such as how to reliably upgrade systems, roll back changes easily, if needed and reproduce a configuration deterministically on another machine. The context of their work is NixOS, a non-trivial Linux distribution that uses a novel package manager to build the entire system configuration from a modular and purely functional specification, thus comprehensively demonstrating the practical utility and scalability of their approach.

We would like to thank the authors and the referees for their efforts in producing and reviewing these papers. In addition we would like to thank Matthias Felleisen and Xavier Leroy for the opportunity to publish the papers as a special issue of the *JFP*, and David Tranah for assistance, practical advice and patience.

Peter Thiemann and Henrik Nilsson
Special Issue Editors