*JFP* **23** (4): 355–356, 2013.  
doi:10.1017/S0956796813000129

# *Special Issue Dedicated to ICFP 2011 Editorial*

KENICHI ASAI

*Department of Information Science, Ochanomizu University, Tokyo, Japan*
(*e-mail:* asai@is.ocha.ac.jp)

BENJAMIN C. PIERCE

*Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA*
(*e-mail:* bcpierce@cis.upenn.edu)

The 16th ACM SIGPLAN International Conference on Functional Programming (ICFP) took place on September 19–21, 2011 in Tokyo, Japan. After the conference, the programme committee, chaired by Olivier Danvy, selected several outstanding papers and invited their authors to submit to this special issue of the *Journal of Functional Programming*. Kenichi Asai and Benjamin C. Pierce acted as editors for these submissions. This issue includes the three accepted articles, each of which provides substantial new material beyond the original conference version. The selected papers represent the importance of various aspects of proof techniques, from theory to practice, all of which aim at verifying realistic programs.

In *How to make ad hoc proof automation less ad hoc*, Gonthier *et al*. present a novel application of Coq's *canonical structures* to automate proofs. Their technique enables us to specify the customized behaviour of proof search concisely within Coq's type system, which has been previously done by tactics. The technique is simple and is illustrated with a number of examples. This pearl-like paper is another instance showing how one simple idea can be pushed through all the way to impact practical verification.

In *Secure distributed programming with value-dependent types*, Swamy *et al*. present F$^\star$, a dependently typed programming language with an advanced type system for secure distributed programming. While maintaining a logically consistent core, F$^\star$ includes arbitrary recursion for general programming as well as affine types for modular reasoning about effects. Despite the complexity of the type system, the paper clearly and elegantly explains it with many examples. F$^\star$ has been successfully used to program and verify a number of realistic examples, including new schemes for multi-party sessions, showing that F$^\star$ is near a sweet spot in dependently typed language design.

In *Modular verification of preemptive OS kernels*, Gotsman and Yang propose a logic for the verification of preemptive multiprocessor OS kernel code. One of the difficulties of verifying preemptive kernel code is the interaction between the scheduler and the kernel: at every program point, a process running in the kernel mode can be descheduled and rescheduled later possibly on a different CPU. Verifying all the possible interactions is simply unrealistic. Based on the concurrent separation logic, the paper presents a logic that can verify the scheduler and the rest of the kernel separately. While the reader may consider the topic outside the scope of functional programming, the program committee of

ICFP 2011 decided to accept this paper because it solves a difficult and important problem that is of interest to the functional programming community. In fact, the paper can be regarded as constructing a nice functional reasoning layer on top of low-level scheduler code.

We thank the authors and referees of these articles for their efforts producing and reviewing these articles within the strict time limits imposed by the special issue publication constraints. We are also grateful to Olivier Danvy for the initial leadership on this special issue and to Matthias Felleisen and David Tranah for general support and editorial advice.

Kenichi Asai and Benjamin C. Pierce
Special Issue Editors