JFP **24** (2-3): 131–132, 2014.
doi:10.1017/S0956796814000124

Special Issue Dedicated to ICFP 2012 Editorial

SATNAM SINGH

Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
(*e-mail:* s.singh@acm.org)

ROBERT BRUCE FINDLER

Northwestern University, 2145 Sheridan Rd, Tech L359, Evanston, IL 60208, USA
(*e-mail:* robby@eecs.northwestern.edu)

The 17th ACM SIGPLAN International Conference on Functional Programming (ICFP) took place on September 10–12, 2012 in Copenhagen, Denmark. After the conference, the programme committee selected several outstanding papers and invited their authors to submit to this special issue of *Journal of Functional Programming*. Robby Findler and Satnam Singh acted as editors for these submissions. This issue includes the accepted papers, each of which provides substantial new material beyond the original conference version.

Dunfield presents the definition of a simply typed language with unrestricted intersection and union types in *Elaborating Intersection and Union Types*. These union and intersection types are versatile, allowing simple encodings of operator overloading, records, and dynamic typing. Dunfield's definition is phrased as an elaboration into ordinary λ-calculus terms, meaning that a language based on his calculus can be compiled via an ordinary ML compiler.

Endrullis, Hendriks, Bakhshi, and Rosu explore three different, commonly used models of streams in their paper *On the Complexity of Stream Equality*. They demonstrate that stream equality is not decidable. More surprisingly, the precise complexity of the equivalence problem depends on the stream model used, ranging from low levels of the arithmetic hierarchy through the entire analytical hierarchy. The paper also contains a careful introduction to the various stream models and the equality problem.

Johnson, Sergey, Earl, Might, and Van Horn show how to combine pushdown flow analysis and abstract garbage collection into a single analysis that is stronger than either individually in the eponymous paper *Introspective Pushdown Analysis*. The key difficulty in the combination is that pushdown flow analysis needs to restrict access to the stack (to preserve decidability) and abstract garbage collection needs access to the stack (to realize its gains in precision). To resolve this conflict, the authors introduce a variant of pushdown analysis that allows just enough stack inspection to facilitate abstract garbage collection, but not so much as to lose decidability.

Myreen and Owens make progress on the problem of adapting proof assistant systems into practical software development systems in *Proof-Producing Translation of Higher-Order Logic into Pure and Stateful ML*. Although one can write and verify an algorithm in a higher-order logic using a proof assistant such as Coq or HOL, the process of extracting

a program from the proof system by conversion into a regular programming language corresponds to a glaring leap of faith. This paper demonstrates how to increase the trustworthiness of such translation steps with an automated technique by proving correct the translation step with respect to the logical and operational semantics of the target language. The authors describe a system that implements a verified translation scheme from HOL4 to their subset of Standard ML called CakeML.

Dagand and McBride present a scheme for code reuse in dependently type programming systems through the use of functional ornaments in *Transporting Functions Across Ornaments*. Ornaments describe how one datatype can be enriched into another with the same structure. This allows a relationship to be established between functions on different datatypes, e.g. addition (on natural numbers) and concatenation (on lists). The definition of addition can then be lifted to lists by only providing the extra details necessary to add (concatenate) lists rather than numbers.

Wadler presents a foundation for concurrent programming in Propositions as Sessions that rests upon the Curry–Howard correspondence. The paper presents CP, a calculus inspired by work of Caires in which propositions of classical linear logic correspond to session types, as well as GV, a linear functional language with session types inspired by work of Gay and Vasconcelos. A translation from GV to CP is presented which formalises a connection between the standard presentation of session types and linear logic. Both CP and GV are free from races and deadlock.

We thank the authors and reviewers 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.

Satnam Singh and Robert Bruce Findler
Special Issue Editors