Editorial TFP 2009/2010

Viktória Zsók · Rex Page · Julia Lawall

Published online: 8 February 2014
© Springer Science+Business Media New York 2014

The Symposium on Trends in Functional Programming has a 25-year history as an annual forum for discussing emerging aspects of methods, tools, and theoretical frameworks supporting the development of declarative software. This volume contains major revisions of papers selected from the post-symposium proceedings of the 2009 symposium in Komárno, Slovakia and the 2010 symposium in Norman, Oklahoma. Out of 50 papers presented at these symposia, the program chairs selected 10 papers, and invited their authors to submit extended versions. The resulting articles went through the normal journal reviewing process, with each article being reviewed by three reviewers. Based on the reviews six were selected for inclusion in the special issue.

Parallel computing is becoming increasingly important, due to the prevalence of multi-cores, and it has long been recognized that pure functional languages, due to their lack of side effects, are potentially very useful for parallel computing. The paper *Comparing Low-Pain and No-Pain Multicore Haskell*s by Mustafa Aswad, Phil Trinder, Abdallah Al Zain, Greg Michaelson and Jost Berthold assesses the state of the art in parallel Haskell. This paper evaluates various parallel Haskell implementations, characterized as low-pain (some parallel annotations required) and no-pain (no parallel annotations required), in terms of performance and ease of programming. This systematic study forms a reference point for the state of the art as well as creating a foundation for future evaluations.

The paper *Graph-based Communication in Eden* by Thomas Horstmeyer and Rita Loogen also addresses ease of programming parallel systems, this time in the context of distributed

V. Zsók
Eötvös Loránd University, Budapest, Hungary
e-mail: zsv@inf.elte.hu

R. Page (✉)
University of Oklahoma, Norman, OK, USA
e-mail: page@ou.edu

J. Lawall
INRIA/LIP6/UPMC/Sorbonne University, Paris, France
e-mail: julia.lawall@lip6.fr

computing. Horstmeyer and Loogen propose the language Grace, which is an extension of the parallel version of Haskell Eden. Grace makes it possible to create a complex network topology in terms of the explicit declaration of nodes and edges, rather than by passing channels within the implementation code as requires in Eden. The paper presents some interesting type issues that arise in this context, and a detailed performance evaluation.

The design of an appropriate type system is an important part of the design of many functional languages. *Types and Associated Type Families for Hardware Simulation and Synthesis* by Andy Gill, Tristan Bull, Andrew Farmer, Garrin Kimmell, and Ed Komp provides a summary and rationale for a new implementation of Kansas Lava, which embeds in Haskell a language for specifying combinatorial and sequential circuits. It uses types to express certain details of circuit behavior. The paper presents a monad for interfacing to VHDL, discusses a representation of unknown values in circuits, and includes an example protocol for enabled signals.

John Capper and Henrik Nilsson describe a type system for checking properties of dynamic systems of equations in their paper *Structural Types for Systems of Equations*. It uses a formalism based on simply typed lambda-calculus as term language and provides a modular mechanism for checking the balance between the number of variables and the number of equations, which can sometimes detect under-defined or over-defined systems. Certain properties of the type system are discussed, including sketches of proofs of some of those properties, and many aspects of the work are formalized in Agda code. The paper provides an accessible entry point for researchers who are not specialists in type theory.

The remaining papers consider graph-based programming languages. Nets in Motion (NiMo) is a graphical programming language in which networks evolve visibly with the computational state, providing mechanisms for execution tracing and debugging. NiMo incorporates graph reductions, types, and other features similar to Haskell, into a consistent, graphical syntax. *Graphical and Incremental Type Inference, a Graph Transformation Approach*, a paper by Silvia Clerici, Cristina Zoltan, and Guillermo Prestigiacomo, focuses on incremental type inference, using graph unification and a compositional calculus for typing nets. Some of the ideas are relevant in both a graphical form, as presented in the paper, and in conventional text-based languages.

Finally, the paper *On Graph Rewriting, Reduction, and Evaluation in the Presence of Cycles* by Ian Zerny considers language graphs from a theoretical point of view. The paper mechanically interconnects the graph rewriting systems of Barendregt and the graph reduction machines of Turner for locally-recursive subgraphs in the context of directed acyclic graphs. The latest reduction semantics are applied to graphs and terms. The derivation techniques are presented as implementations in Standard ML.

The editors thank the authors for their efforts in revising their TFP papers for this volume and thank the reviewers who provided helpful, detailed comments to aid the authors in the revision process.