Preface and Overview of Papers

This volume contains a refereed selection of papers presented at the 1998 International Workshop on the Implementation of Functional Languages (IFL'98), held at University College, London, September 9–11, 1998. This was the tenth in a series of international workshops that have been held at locations in The Netherlands, Germany, Sweden, and the UK, and the third to be published in the Springer-Verlag series of Lecture Notes in Computer Science (selected papers from IFL'96 and IFL'97 are published in LNCS volumes 1268 and 1467 respectively).

The workshop has grown over the years, and the 1998 meeting attracted 64 researchers from the international functional language community, many of whom presented papers at the workshop. We are pleased that, after due revision and rigorous peer review, we are able to publish 15 of those papers in this volume.

The papers selected from the workshop cover a wide range of topics including work on parallel process co-ordination (Aßmann; Klusik, Ortega, and Peña), parallel profiling (Charles and Runciman; King, Hall, and Trinder), compilation (Grelck) and semantics of parallel systems (Hall, Baker-Finch, Trinder, and King), programming methodology (Koopman and Plasmeijer; Scholz), interrupt handling (Reid), type systems (McAdam; Pil), strictness analysis (Pape), concurrency and message passing (Holyer and Spiliopoulou; Serrarens and Plasmeijer) and inter-language working (Reinke).

Some of the work developed out of research reported at previous IFL workshops. In *Implementing Eden, or: Dreams Become Reality, pp 103–119*, Klusik, Ortega, and Peña show how expressions in the parallel functional language, EDEN, are compiled, work arising from their previous operational specification of the distributed abstract machine, DREAM (IFL'97). Grelck (*Shared Memory Multiprocessor Support for SAC, pp 38–53*) continues work on Single Assignment C (IFL'96), describing the compilation of SAC programs for multithreaded execution on multiprocessor systems. Scholz also describes work on SAC (*A Case Study: Effects of WITH-Loop-Folding on the NAS MG Benchmark in SAC, pp 216–228*) showing how the WITH-Loop-Folding optimization supports high-level array operations efficiently. Much work has been done in recent years on optimization of functional programming compilers. This paper shows that code generated from the SAC program not only reaches the execution time of code generated from FORTRAN, but even outperforms it by a significant amount. Aßmann (*Preliminary Performance Results for an Implementation of the Process Coordination Language K2, pp 1–19*) builds on earlier work on K2 (IFL'97), analysing the efficiency of a number of benchmarks. Pil (*Dynamic Types and Type Dependent Functions, pp 169–185*) describes work on type dependent functions which will be added to the Clean language using overloading. These will facilitate communication between independent (possibly persistent) functional programs using a minimum of dynamic typing. Two papers continue work on profiling parallel functional programs. King, Hall, and Trinder (*A Strategic Profiler for Glasgow Parallel Haskell, pp 88–102*) show how the GRANSIM-SP profiler attributes costs to the abstract evaluation strategies that created various threads. Charles and Runciman (*An Interactive Approach to Profiling Parallel Functional Programs, pp 20–37*) concentrate on the user interface to profiling tools, describing a system that interactively combines graphical information with a query interface. Finally, Koopman, and Plasmeijer (*Efficient Combinator Parsers, pp 120–136*) describe how elegant combinators for constructing parsers, which, however, have exponential time complexity, can be made more efficient using continuations.

Other papers represent work on new lines of research. Reinke (*Towards a Haskell/Java Connection, pp 200–215*) describes preliminary work on a system to support inter-language working between functional and object-oriented languages, a timely coverage of a vitally important issue. Reid (*Putting the Spine Back in the Spineless Tagless G-Machine: An Implementation of Resumable Black-Holes, pp 186–199*) shows how to tackle interrupts in lazy languages, an issue that has been problematic for some years. Serrarens and Plasmeijer (*Explicit Message Passing for Concurrent Clean, pp 229–245*) describe a language extension providing efficient flexible communication in a concurrent system. Holyer and Spiliopoulou also present concurrency research (*Concurrent Monadic Interfacing, pp 72–87*) in which the monadic style of interfacing is adapted to accommodate deterministic concurrency. Three papers describe more theoretical work, albeit for use in practical implementations. Pape (*Higher Order Demand Propagation, pp 153–168*) gives a new denotational semantics mapping function definitions into demand propagators, allowing the definition of a backward strictness analysis of the functions even when they are higher order and/or transmit across module boundaries. McAdam (*On the Unification of Substitutions in Type Inference, pp 137–152*) describes a polymorphic type inference algorithm which reduces the confusion which often arises when type errors are reported to users. Finally, Hall, Baker-Finch, Trinder, and King (*Towards an Operational Semantics for a Parallel Non-strict Functional Language, pp 54–71*) present a semantics for a simple parallel graph reduction system based on Launchbury's natural semantics for lazy evaluation.

The 15 papers published in this volume were selected using a rigorous *a-posteriori* refereeing process from the 39 papers that were presented at the workshop. The reviewing process was shared among the program committee, which comprised:

Chris Clack	University College London	UK
Tony Davie	University of St. Andrews	UK
John Glauert	University of East Anglia	UK
Kevin Hammond	University of St. Andrews	UK
Werner Kluge	University of Kiel	Germany
Pieter Koopman	University of Nijmegen	The Netherlands
Rita Loogen	University of Marburg	Germany
Greg Michaelson	Heriot-Watt University	UK
Markus Mohnen	RWTH Aachen	Germany
Marko van Eekelen	University of Nijmegen	The Netherlands

We would like to thank the many additional anonymous referees who provided us with timely, high-quality reports on which our decisions were based.

The overall balance of the papers is representative, both in scope and technical substance, of the contributions made to the London workshop as well as to those that preceded it. Publication in the LNCS series is not only intended to make these contributions more widely known in the computer science community but also to encourage researchers in the field to participate in future workshops, of which the next one will be held in Nijmegen, The Netherlands, September 7-10, 1999 (see `http://www.cs.kun.nl/~pieter/ifl99/index.htm`).

Significantly, this year the workshop attracted industrial sponsorship from both Andersen Consulting and Ericsson. This indicates the growing importance of functional programming languages and their implementation within two key commercial spheres (IT Consultancy and Telecommunications, respectively). We thank our sponsors for their generous contributions.

March 1999 Kevin Hammond, Tony Davie, and Chris Clack