Lecture Notes in Computer Science 1467
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Chris Clack Kevin Hammond
Tony Davie (Eds.)

Implementation of Functional Languages

9th International Workshop, IFL'97
St. Andrews, Scotland, UK
September 10-12, 1997
Selected Papers

Springer

Volume Editors

Chris Clack
University College London, Department of Computer Science
Gower Street, London WC1E 6BT, UK
E-mail: clack@cs.ucl.ac.uk

Kevin Hammond
Tony Davie
University of St. Andrews, Division of Computer Science
North Haugh, St. Andrews, Scotland KY16 9SS, UK
E-mail: {kh,ad}@dcs.st-and.ac.uk

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Implementation of functional languages : 9th international workshop ; selected papers / IFL '97, St. Andrews, Scotland, UK, September 10 - 12, 1997. Chris Clack ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1998
(Lecture notes in computer science ; Vol. 1467)
ISBN 3-540-64849-6

CR Subject Classification (1991): D.3, D.1.1, F.3

ISSN 0302-9743
ISBN 3-540-64849-6 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10638546   06/3142 – 5 4 3 2 1 0   Printed on acid-free paper

Preface and Overview of Papers

This volume contains a selection of papers presented at the 1997 International Workshop on the Implementation of Functional Languages (IFL '97), held at St. Andrews in Scotland, September 10–12, 1997. This is the ninth in a series of workshops that were previously held in The Netherlands, Germany, Sweden, and the UK, and the second to be published in the Springer-Verlag series of Lecture Notes in Computer Science (selected papers from IFL '96 are published in LNCS Volume 1268).

The workshop has been growing over the years, and the 1997 meeting was perhaps the largest to date, attracting over 50 researchers from the international functional language community, the majority of whom presented papers at the workshop. We are pleased to be able to publish the selection of refereed and revised papers that appear herein.

While the original focus of the workshop was on parallel implementation, it has broadened over time to include compilation, type systems, language issues, benchmarking and profiling, parallelism, language issues, memory management, applications, and some theoretical work. It thus represents a cross-section of the active research community. The papers presented in this volume have been grouped under seven topic headings as follows:

Compilation. In keeping with the theme of the workshop, several papers deal with abstract machines and compilation techniques. Peyton Jones *et al.* identify problems with using C as a compiler target language and suggest an alternative language which they call `C--`; Holyer and Spiliopoulou present the Brisk Machine, a simplified STG machine that is specifically designed to support multiple paradigms such as computational mobility, dynamic loading and logic programming; Wakeling investigates how compilation to the Java Virtual Machine (JVM) can support the use of Haskell to program embedded processors and provides an interesting comparison between the G-machine and the JVM; Chitil demonstrates that the application of the compile-time optimisation technique of common subexpression elimination to a lazy functional language compiler can lead to an unexpected conclusion; and finally, Scholz presents a technique for producing efficient code from high-level array operations.

Types. Exploitation of type information for compilation continues to be an attractive and fertile area of research. Agat presents a typed intermediate language and a type and effect system which can be used in the register-allocation phase of a compiler; and Mogensen describes a refined program analysis which extends previous work to detect zero uses of an argument, thereby supporting finer compile-time optimisation: for example to avoid updates during garbage collection, or for a limited form of compile-time garbage collection.

Benchmarking and Profiling. In order to ensure acceptable performance from functional programs, it is important for programmers to understand how different data-structuring techniques perform and it is vital for systems developers to provide good benchmarking and profiling techniques. Erwig analyses benchmarking results for two different implementations of purely functional graph data structures tested against three patterns of graph utilisation; Moss and Runciman present Auburn, a system tool which uses the concept of a "datatype usage graph" (DUG) to synthesise code for benchmarking different implementations of an Abstract Data Type; and Sparud and Runciman describe how the traditionally resource-hungry technique of recording execution traces ("redex trails") can be managed parsimoniously — this is dramatically illustrated by their results from tracing a large computation.

Parallelism. Parallel implementation has always been a strong theme of the IFL workshops and this year is no exception. Loidl and Trinder analyse the use of "evaluation strategies" together with system tools such as the GranSim simulator, the instrumented GUM implementation and high-level profiling tools in order to parallelise three large programs; Loidl *et al.* provide an in-depth discussion of the parallelisation of LOLITA, which is claimed to be the largest existing non-strict functional program (at 47 000 lines); Junaidu *et al.* discuss their experience of using GranSim and GUM to parallelise Naira (a compiler for a parallel dialect of Haskell which has itself been parallelised); Chakravarty investigates the benefits of distinguishing remote *tasks* from local *threads*, and of generating tasks and threads lazily, in order to alleviate the problem of idle processors in a message-passing parallel system with high message latencies; Breitinger *et al.* present a distributed-memory parallel system that comprises Eden (which extends Haskell with a coordination language), PEARL (which extends the STG language with message-passing and processes) and DREAM (which extends the STG machine to provide a distributed run-time environment for Eden programs); and Serrarens illustrates how support for multicasting can improve the efficiency of data-parallel programs in Concurrent Clean.

Interaction. Managing events and other forms of user-interaction has long been a *bête-noir* of functional programming systems and at last support for such interaction is maturing. Karlsen and Westmeier provide a uniform framework for event handling that is independent of the source of the event; and Achten and Plasmeijer explain how the new Clean IO model (using new Clean features such as existential types and constructor classes) can support event-driven user interfaces with call-back functions, local state, and message-passing communication.

Language Design. Large-scale and real-world programming is also an important issue for sequential systems. Didrich *et al.* consider two issues related to programming in the large — modularisation and name spaces — and describe the design of the corresponding concepts to support large scale functional programming in the language Opal 2α; and Mohnen proposes the use of "extended context patterns" (together with additional typing rules and inference rules) in order to support the highly expressive language feature of context patterns (see LNCS 1268) whilst overcoming the efficiency problems associated with implementing the technique.

Garbage Collection. The final paper presented in this volume addresses the issue of memory management in the programming language Erlang. Given the unusual characteristic of Erlang that all heap pointers point from newer to older objects, Boortz and Sahlin present a compacting garbage collection algorithm which has zero memory overheads and takes linear time with respect to the size of the data area.

The papers published in this volume were selected using a rigorous *a posteriori* refereeing process from the 34 papers that were presented at the workshop. The reviewing was shared among the program committee, which comprised:

Warren Burton	Simon Fraser University	Canada
Chris Clack	University College London	UK
Tony Davie	University of St. Andrews	UK
Martin Erwig	Fern Universität Hagen	Germany
Pascal Fradet	IRISA/INRIA	France
Kevin Hammond	University of St. Andrews	UK
Pieter Hartel	University of Southampton	UK
Fritz Henglein	University of Copenhagen	Denmark
John Hughes	Chalmers University of Technology	Sweden
Werner Kluge	University of Kiel	Germany
Herbert Kuchen	Westfälische Wilhelms-Universität Münster	Germany
Bruce McKenzie	University of Canterbury	New Zealand
Erik Meijer	University of Utrecht	The Netherlands
John Peterson	Yale University	USA
Rinus Plasmeijer	University of Nijmegen	The Netherlands
Colin Runciman	University of York	UK

The overall balance of the papers is representative, both in scope and technical substance, of the contributions made to the St. Andrews 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 London, UK, September 9–11, 1998 (for more information see `http://www.cs.ucl.ac.uk/staff/ifl98/`).

April 1998 Chris Clack, Tony Davie, and Kevin Hammond

Table of Contents