Lecture Notes in Computer Science 1608
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

S. Doaitse Swierstra Pedro R. Henriques
José N. Oliveira (Eds.)

Advanced Functional Programming

Third International School, AFP'98
Braga, Portugal, September 12-19, 1998
Revised Lectures

Volume Editors

S. Doaitse Swierstra
Utrecht University, Department of Computer Science
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
E-mail: doaitse@cs.uu.nl

Pedro R. Henriques
José N. Oliveira
University of Minho, Department of Informatics
Campus de Gualtar, 4709 Braga Codex, Portugal
E-mail: {prh,jno}@di.uminho.pt

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advanced functional programming : third international school ;
revised lectures / AFP'98, Braga, Portugal, September 12 - 19, 1998.
S. Doaitse Swierstra ... (ed.). - Berlin ; Heidelberg ; New York ;
Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo
: Springer, 1999
(Lecture notes in computer science ; Vol. 1608)
ISBN 3-540-66241-3

CR Subject Classification (1998): D.1.1, D.3.2, D.2.2, D.2.10

ISSN 0302-9743
ISBN 3-540-66241-3 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN: 10704973 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

In this volume you will find the lecture notes corresponding to the presentations given at the 3^{rd} summer school on Advanced Functional Programming, held in Braga, Portugal from September 12–19, 1998.

This school was preceded by earlier ones in Båstad (1995, Sweden, LNCS 925) and Olympia, WA (1996, USA, LNCS 1129). The goal of this series of schools is to bring recent developments in the area of functional programming to a large group of students. The notes are published in order to enable individuals, small study groups, and lecturers to become acquainted with recent work in the fast developing area of functional programming.

What made this school particularly interesting was the fact that all lectures introduced useful software, that was used by the students in the classes to get hands-on experience with the subjects taught. We urge readers of this volume to download the latest version of this software from the Internet and try to do the exercises from the text themselves; the proof of the program is in the typing.

The first lecture, on *Sorting Morphisms*, serves as a gentle introduction to the things to come. If you have always been afraid of the word "morphism", and you have been wondering what catamorphisms, anamorphisms, hylomorphims, and paramorphims were about, this is the paper to read first; you will discover that they are merely names for recursion patterns that occur over and over again when writing functional programs. The algorithms in the paper are all about sorting, and since you are likely to know those algorithms by heart already, seeing them structured and analyzed in a novel way should serve as a motivation to read on to the second lecture.

The second lecture, on *Generic Programming*, is almost a book in a book. The notes can be seen as the culminating point of the STOP-project, sponsored by the Dutch government at the end of the 80's and the beginning of the 90's. Its overall goal was the development of a calculational way of deriving programs. The project has provided deeper insight into real functional programming and into the theory behind many things commonly written by functional programmers. One of the main achievements of the project has been to make people aware of the fact that many algorithms can be described in a data-independent way. The PolyP system introduced in these notes is one of the translations to the Haskell-world of this theoretical underpinning.

The third lecture, on *Generic Program Transformation*, can also be seen as an application of the theory introduced in lecture two. Many efficiency-improving program transformations can be performed in a mechanical way, and these would not have been possible without insight into the correctness of such transformations gained in the lecture on Generic Programming.

The fourth lecture, on *Designing and Implementing Combinator Languages*, introduces an easy to write formalism for writing down the catamorphisms introduced in earlier chapters. It is shown how quite complicated catamorphisms, that at first sight seem rather forbidding by making extensive use of higher-order do-

mains, can actually be developed in a step-wise fashion, using an attribute grammar view; it is furthermore shown how to relate this way of programming with concepts from the object-oriented world thus making clear what the strengths and weaknesses of each world are.

The fifth lecture, titled *Using MetaML: A Staged Programming Language*, introduces the concept of partial evaluation. It serves as another instance of the quest for "the most generic of writing programs at the lowest cost". The staging techniques show how costs that were introduced by adding extra levels of abstraction, may be moved from run-time to compile-time.

It has been common knowledge to users of modern functional languages that the type system can be a great help in shortening programs and reducing errors. In the extreme one might see a type as a predicate capturing the properties of any expression with that type. In the sixth lecture on *Cayenne – Spice up your Programming with Dependent Types* it is shown in what direction functional languages are most likely to develop, and what may be expected of the new type systems to be introduced.

The last lecture, titled *Haskell as an Automation Controller*, shows that writing functional programs does not have to imply that one is bound to remain isolated from the rest of the world. Being able to communicate with software written by others in a uniform way, is probably one of the most interesting new developments in current computer science. It appears that the concept of a monad together with the Haskell typing rules, is quite adequate to describe the interface between Haskell programs and the outer world.

Finally we want to thank everyone who contributed to this school and made it such a successful event: sponsors, local system managers, local organizers, students, and last but not least the lecturers. We are convinced that everyone present at the school enjoyed this event as much as we did, and we all hope that you will feel some of the spirit of this event when studying these lecture notes.

March 1999

Doaitse Swierstra
Pedro Henriques
José Oliveira

Sponsorship

The school has received generous sponsorship from:
FCT - Fundação para a Ciência e Tecnologia, Ministério da Ciência e Tecnologia

Adega Cooperativa de Ponte de Lima
Agência Abreu
CGD - Caixa Geral de Depósitos
CIUM - Centro de Informática da Universidade do Minho
DI - Departamento de Informática da Universidade do Minho
GEPL - Grupo de Especificação e Processamento de Linguagens
LESI - Direcção de Curso de Engenharia de Sistemas e Informática
Enabler
Lactolima
Laticínios das Marinhas, Lda
Novabase Porto - Sistemas de Informação SA
Primavera Software
Projecto Camila - Grupo de Métodos Formais
Sidereus - Sistemas de Informação e Consultoria Informática Lda
SIBS - Sociedade Interbancária de Serviços
Vieira de Castro

Local Committee:

José Almeida, Minho
Luís Barbosa, Minho
José Barros, Minho
M. João Frade, Minho
Pedro Henriques, Minho
F. Mário Martins, Minho
F. Luis Neves, Minho
Carla Oliveira, Minho
Jorge Pinto, Lix
Jorge Rocha, Minho
Cesar Rodrigues, Minho
João Saraiva, Minho
M. João Varanda, Minho

Table of Contents