Lecture Notes in Computer Science 2793
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Roland Backhouse Jeremy Gibbons (Eds.)

Generic Programming

Advanced Lectures

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Roland Backhouse
The University of Nottingham
School of Computer Science and Information Technology
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
E-mail: rcb@cs.nott.ac.uk

Jeremy Gibbons
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
E-mail: Jeremy.Gibbons@comlab.ox.ac.uk

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): D.3, D.1, D.2, F.3, E.1

ISSN 0302-9743
ISBN 3-540-20194-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 10931868 06/3142 5 4 3 2 1 0

Preface

Generic programming is about making programming more effective by making it more general. This volume is about a novel form of genericity in programs, based on parameterizing programs by the structure of the data they manipulate. The material is based on lectures presented at a summer school on Generic Programming held at the University of Oxford in August 2002.

The lectures by Hinze and Jeuring introduced Generic Haskell, an extension of the Haskell programming language that allows the programmer to define a function by induction on the structure of types. The implementation of Generic Haskell provided a valuable tool for students to experiment with applications of this form of datatype genericity. The lecture material in this volume is divided into two parts. The first part ("practice and theory") introduces Generic Haskell and the theory that underlies its design. The second part ("applications") discusses three advanced applications of Generic Haskell in some depth.

The value of generic programming is illusory unless the nature and extent of the genericity can be described clearly and precisely. The lectures by Backhouse and Crole delve deeper into the theoretical basis for datatype genericity. Backhouse reviews the notion of parametric polymorphism (a notion well known to functional programmers) and then shows how this notion is extended to higher-order notions of parametricity. These are used to characterize what it means for a value to be stored in a datatype. Also, transformations on data structures are given precise specifications in this way. Underlying this account are certain basic notions of category theory and allegory theory. Crole presents the category theory needed for a deeper understanding of mechanisms for defining datatypes.

The final chapter, by Fiadeiro, Lopes and Wermelinger applies the mathematical "technology" of parameterization to the larger-scale architectural structure of programs. The description of a system is split into components and their interactions; architectural connectors are parameterized by components, leading to an overall system structure consisting of components and connector instances establishing the interactions between the components.

Our thanks go to all those involved in making the school a success. We are grateful to the technical support staff of the Oxford University Computing Laboratory for providing computing facilities, to Yorck Hunke, David Lacey and Silvija Seres of OUCL for assistance during the school, and to St. Anne's College for an amenable environment for study. Thanks also go to Peter Buneman and Martin Odersky, who lectured at the school on *Semi-structured Data* and on *Object-Oriented and Functional Approaches to Compositional Programming*, respectively, but were unable to contribute to the proceedings.

June, 2003

Roland Backhouse
Jeremy Gibbons

Contributors

Roland Backhouse
School of Computer Science and Information Technology,
University of Nottingham, Nottingham, NG8 1BB, UK
rcb@cs.nott.ac.uk
http://www.cs.nott.ac.uk/~rcb/

Roy Crole
Department of Mathematics and Computer Science,
University of Leicester, University Road, Leicester, LE1 7RH, UK
roy.crole@mcs.le.ac.uk
http://www.mcs.le.ac.uk/~rcrole/

José Luiz Fiadeiro
Department of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, UK
jose@fiadeiro.org

Ralf Hinze
Institut für Informatik III, Universität Bonn, Römerstraße 164, 53117 Bonn, Germany
ralf@informatik.uni-bonn.de
http://www.informatik.uni-bonn.de/~ralf/

Paul Hoogendijk
Philips Research, Prof. Holstlaan 4, 5655 AA Eindhoven, The Netherlands
Paul.Hoogendijk@philips.com

Johan Jeuring
Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
and
Open University, Heerlen, The Netherlands
johanj@cs.uu.nl
http://www.cs.uu.nl/~johanj/

Antónia Lopes
Department of Informatics, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisboa, Portugal
mal@di.fc.ul.pt

Michel Wermelinger
Dept. of Informatics, Faculty of Sciences and Technology, New University of Lisbon, Quinta da Torre, 2829-516 Caparica, Portugal
mw@di.fct.unl.pt

Table of Contents