TWO-LEVEL FUNCTIONAL LANGUAGES

Cambridge Tracts in Theoretical Computer Science

Titles in the series

1. G. Chaitin *Algorithmic Information Theory*
2. L.C. Paulson *Logic and Computation*
3. M. Spivey *Understanding Z*
4. G. Revesz *Lambda Calculus, Combinators and Functional Programming*
5. A. Ramsay *Formal Methods in Artificial Intelligence*
6. S. Vickers *Topology via Logic*
7. J-Y. Girard, Y. Lafont & P. Taylor *Proofs and Types*
8. J. Clifford *Formal Semantics & Pragmatics for Natural Language Processing*
9. M. Winslett *Updating Logical Databases*
10. K. McEvoy & J.V. Tucker (eds) *Theoretical Foundations of VLSI Design*
11. T.H. Tse *A Unifying Framework for Structured Analysis and Design Models*
12. G. Brewka *Nonmonotonic Reasoning*
15. S. Dasgupta *Design Theory and Computer Science*
17. J.C.M. Baeten (ed) *Applications of Process Algebra*
18. J.C.M. Baeten & W.P. Weijland *Process Algebra*
23. E.-R. Olderog *Nets, Terms and Formulas*
27. W.H. Hesselink *Programs, Recursion and Unbounded Choice*
29. P. Gärdenfors (ed) *Belief Revision*
30. M. Anthony & N. Biggs *Computational Learning Theory*
34. F. Nielson & H.R. Nielson *Two Level Functional Languages*

TWO-LEVEL FUNCTIONAL LANGUAGES

FLEMMING NIELSON & HANNE RIIS NIELSON
Department of Computer Science
Aarhus University, Denmark

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521403849

First published 1992
This digitally printed first paperback version 2005

A catalogue record for this publication is available from the British Library

ISBN-13 978-0-521-40384-9 hardback
ISBN-10 0-521-40384-7 hardback

ISBN-13 978-0-521-01847-0 paperback
ISBN-10 0-521-01847-1 paperback

Contents

List of Figures

List of Tables

Preface

The subject area of this book concerns the implementation of functional languages. The main perspective is that part of the implementation process amounts to

making computer science concepts explicit

in order to facilitate the application, and the development, of general frameworks for program analysis and code generation.

This is illustrated on a specimen functional language patterned after the λ-calculus:

- *Types* are made explicit in Chapter 2 by means of a Hindley/Milner/Damas type analysis.
- *Binding times* are made explicit in Chapter 3 using an approach inspired by the one for type analysis. The binding times of chief interest are *compile-time* and *run-time*.
- *Combinators* are made explicit in Chapter 4 but only for run-time computations whereas the compile-time computations retain their λ-calculus syntax.

The advantages of this approach are illustrated in the remainder of the book where the emphasis also shifts from a 'syntactic perspective' to a more 'semantic perspective':

- A notion of *parameterized semantics* is defined in Chapter 5 and this allows a wide variety of semantics to be given.
- It is illustrated for *code generation* in Chapter 6. Code is generated for a structured abstract machine and the correctness proof exploits Kripke-logical relations and layered predicates.
- It is illustrated for *abstract interpretation* in Chapter 7. We generalize Wadler's strictness analysis to general lists, show the correctness using logical relations, and illustrate the similarity between tensor products and Wadler's case analysis.

Finally, Chapter 8 discusses possible ways of extending the development. This includes the use of abstract interpretation to obtain an improved code generation that may still be proved correct. We also illustrate the role of the mixed λ-calculus and combinatory logic as a metalanguage for denotational semantics; this allows a systematic approach to compiler generation from semantic specifications.

Notes for the Reader

This book is intended for researchers and for students who already have some formal training. Much of the work reported here has been documented elsewhere in the scientific literature and we have therefore aimed at a style of exposition where we concentrate on the main insights and methods, including proofs and proof techniques, but where we feel free to refer to the literature for technically complex generalizations and details of tedious proofs. To facilitate this, we provide bibliographic notes covering variations of the technical development. Our notation is mostly standard but we find '$\hookrightarrow$' a more readable notation for 'partial functions' than '$\rightharpoonup$'.

Acknowledgements

The research reported here has been supported by The Danish Natural Sciences Research Council. The presentation has benefited from comments from our students and colleagues, in particular Torben Amtoft, Poul Christiansen, Fritz Henglein, Torben Lange, Jens Mikkelsen, Thorleif Nielson, Jens Palsberg, Hans J. Pedersen, Anders Pilegaard, Kirsten L. Solberg, Bettina B. Sørensen and Phil Wadler. Finally, David Tranah made a number of suggestions concerning how to improve the presentation.

Aarhus, January 1992

Flemming Nielson

Hanne Riis Nielson