Lambda-Calculus and Combinators, an Introduction

Combinatory logic and λ-calculus were originally devised in the 1920s for investigating the foundations of mathematics using the basic concept of 'operation' instead of 'set'. They have since evolved into important tools for the development and study of programming languages.

The authors' previous book *Introduction to Combinators and λ-Calculus* served as the main reference for introductory courses on λ-calculus for over twenty years: this long-awaited new version offers the same authoritative exposition and has been thoroughly revised to give a fully up-to-date account of the subject.

The grammar and basic properties of both combinatory logic and λ-calculus are discussed, followed by an introduction to type-theory. Typed and untyped versions of the systems, and their differences, are covered. λ-calculus models, which lie behind much of the semantics of programming languages, are also explained in depth.

The treatment is as non-technical as possible, with the main ideas emphasized and illustrated by examples. Many exercises are included, from routine to advanced, with solutions to most of them at the end of the book.

Review of *Introduction to Combinators and λ-Calculus:*

'This book is very interesting and well written, and is highly recommended to everyone who wants to approach combinatory logic and λ-calculus (logicians or computer scientists).' *Journal of Symbolic Logic*

'The best general book on λ-calculus (typed or untyped) and the theory of combinators.' Gérard Huet, *INRIA*

Lambda-Calculus and Combinators, an Introduction

J. ROGER HINDLEY

Department of Mathematics,
Swansea University, Wales, UK

JONATHAN P. SELDIN

Department of Mathematics and Computer Science,
University of Lethbridge, Alberta, Canada

CAMBRIDGE
UNIVERSITY PRESS

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

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

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

www.cambridge.org
Information on this title: www.cambridge.org/9780521898850

First published 2008

Printed in the United Kingdom at the University Press, Cambridge

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

Library of Congress Cataloguing in Publication data
Hindley, J. Roger.
Lambda-calculus and combinators : an introduction / J. Roger Hindley,
Jonathan P. Seldin.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-521-89885-0 (hardback)
1. Lambda-calculus. 2. Combinatory logic. I. Seldin, J. P. II. Title.
QA9.5.H565 2008
511.3′5–dc22
2008006276

ISBN 978-0-521-89885-0 hardback

To Carol, Goldie
and
Julie

Contents

Preface

The λ-calculus and combinatory logic are two systems of logic which can also serve as abstract programming languages. They both aim to describe some very general properties of programs that can modify other programs, in an abstract setting not cluttered by details. In some ways they are rivals, in others they support each other.

The λ-calculus was invented around 1930 by an American logician Alonzo Church, as part of a comprehensive logical system which included higher-order operators (operators which act on other operators). In fact the language of λ-calculus, or some other essentially equivalent notation, is a key part of most higher-order languages, whether for logic or for computer programming. Indeed, the first uncomputable problems to be discovered were originally described, not in terms of idealized computers such as Turing machines, but in λ-calculus.

Combinatory logic has the same aims as λ-calculus, and can express the same computational concepts, but its grammar is much simpler. Its basic idea is due to two people: Moses Schönfinkel, who first thought of it in 1920, and Haskell Curry, who independently re-discovered it seven years later and turned it into a workable technique.

The purpose of this book is to introduce the reader to the basic methods and results in both fields.

The reader is assumed to have no previous knowledge of these fields, but to know a little about propositional and predicate logic and recursive functions, and to have some experience with mathematical induction.

Exercises are included, and answers to most of them (those marked *) are given in an appendix at the end of the book. In the early chapters there are also some extra exercises without answers, to give more routine practice if needed.

References for further reading are included at the ends of appropriate chapters, for the reader who wishes to go deeper.

Some chapters on special topics are included after the initial basic chapters. However, no attempt has been made to cover all aspects of λ-calculus and combinatory logic; this book is only an introduction, not a complete survey.

This book is essentially an updated and re-written version of [HS86].[1] It assumes less background knowledge. Those parts of [HS86] which are still relevant have been retained here with only minor changes, but other parts have been re-written considerably. Many errors have been corrected. More exercises have been added, with more detailed answers, and the references have been brought up to date. Some technical details have been moved from the main text to a new appendix.

The three chapters on types in [HS86] have been extensively re-written here. Two of the more specialized chapters in [HS86], on higher-order logic and the consistency of arithmetic, have been dropped.

Acknowledgements The authors are very grateful to all those who have made comments on [HS86] which have been very useful in preparing the present book, especially Gérard Berry, Steven Blake, Naim Çağman, Thierry Coquand, Clemens Grabmayer, Yexuan Gui, Benedetto Intrigila, Kenichi Noguchi, Hiroakira Ono, Gabriele Ricci, Vincenzo Scianni, John Shepherdson, Lewis Stiller, John Stone, Masako Takahashi, Peter Trigg and Pawel Urzyczyn.

Their comments have led to many improvements and the correction of several errors. (Any errors that remain are entirely the authors' responsibility, however.)

On the production side, the authors wish to thank Larissa Kowbuz for her very accurate typing of an early draft, the designers of LaTeX and its associated software for the actual type-setting, and M. Tatsuta for the use of his macro 'proof.sty'. We are also very grateful to Chris Whyley for technical help, and to Cambridge University Press, especially David Tranah, for their expert input.

On the financial side we thank the National Sciences and Engineering Research Council of Canada for a grant which enabled a consultation visit by Hindley to Seldin in 2006, and the Department of Mathematics and Computer Science of the University of Lethbridge for helpful hospitality during that visit.

[1] Which itself was developed from [HLS72] which was written with Bruce Lercher.

Last but of course not least, the authors are very grateful to their wives, Carol Hindley and Goldie Morgentaler, for much encouragement and patient tolerance during this book's preparation.

Notes on the text This book uses a notation for functions and relations that is fairly standard in logic: *ordered pairs* are denoted by '$\langle\ ,\ \rangle$', a *binary relation* is regarded as a set of ordered pairs, and a *function* as a binary relation such that no two of its pairs have the same first member. If any further background material is needed, it can be found in textbooks on predicate logic; for example [Dal97], [Coh87], [End00], [Men97] or [Rau06].

The words '*function*', '*mapping*' and '*map*' are used interchangeably, as usual in mathematics. But the word '*operator*' is reserved for a slightly different concept of function. This is explained in Chapter 3, though in earlier chapters the reader should think of 'operator' and 'function' as meaning the same.

As usual in mathematics, the *domain* of a function ϕ is the set of all objects a such that $\phi(a)$ is defined, and the *range* of ϕ is the set of all objects b such that $(\exists a)(b = \phi(a))$. If A and B are sets, a *function from A to B* is a function whose domain is A and whose range is a subset of B.

Finally, a note about 'we' in this book: 'we' will almost always mean the reader and the authors together, not the authors alone.