COMPILING WITH CONTINUATIONS

COMPILING WITH CONTINUATIONS

ANDREW W. APPEL
Department of Computer Science, Princeton University

CAMBRIDGE UNIVERSITY PRESS
Cambridge
New York *Port Chester* *Melbourne* *Sydney*

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/9780521416955

First published 1992
This digitally printed first paperback version (with corrections) 2006

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

Library of Congress Cataloguing in Publication data

Appel, Andrew W., 1960-
Compiling with continuations / Andrew W. Appel.
p. cm.
Includes bibliographical references.
ISBN 0-521-41695-7
1. Compilers (Computer programs) 2. Standard ML of New Jersey.
I. Title.
QA76.76.C65A67 1992
005.4′53–dc20 91-26651
CIP

ISBN-13 978-0-521-41695-5 hardback
ISBN-10 0-521-41695-7 hardback

ISBN-13 978-0-521-03311-4 paperback
ISBN-10 0-521-03311-X paperback

CONTENTS

ACKNOWLEDGMENTS

The compiler described in this book—**Standard ML of New Jersey**—is the work of many people. **David B. MacQueen** and I began the work in 1986; we intended to spend about one year making a Standard ML front end as a tool for further research. David did most of the work on the type checker and module system; we worked together on the parser, abstract syntax, static environment mechanism, and semantic analysis. My interest has been in dynamic semantics, intermediate representations, optimization, code generation, and runtime system.

Of course, we've ended up spending more than five years on the implementation, as the scope of project has become much more ambitious: a complete, robust, efficient, portable programming environment for Standard ML. We could not have done it without the help we've received from many talented people. In alphabetical order:

Bruce F. Duba helped improve the pattern-match compiler, the CPS constant-folding phase, the in-line expansion phase, the spill phase, and numerous other parts of the compiler. He also helped to design the *call with current continuation* mechanism.

Adam T. Dingle implemented the Emacs mode for the debugger.

Lal George taught the code generator about floating-point registers and made floating-point performance respectable. He also fixed several difficult bugs introduced by me and others.

Trevor Jim helped design the CPS representation, and implemented the match compiler and the closure-conversion phase, the original library of floating-point functions, and the original assembly-language implementation of external primitive functions.

James S. Mattson implemented the first version of the lexical-analyzer generator used in constructing the compiler.

James W. O'Toole implemented the NS32032 code generator.

Norman Ramsey implemented the MIPS code generator.

John H. Reppy contributed many improvements and rewrites of the runtime system. He implemented the signal-handling mechanism, improved the *call with current continuation* mechanism, designed the current mechanism for calls to C functions, implemented the sophisticated new garbage collector, and generally made the runtime system more robust. He also implemented the SPARC code generator.

Nick Rothwell helped implement the separate compilation mechanism.

Zhong Shao implemented common-subexpression elimination, as well as the callee-save convention that uses multiple-register continuations for faster procedure calls.

David R. Tarditi improved the lexical-analyzer generator and implemented the parser generator used to build parts of the front end; he helped in implementing the type-reconstruction algorithm used by the debugger; and he implemented the ML-to-C translator with **Anurag Acharya** and **Peter Lee**.

Mads Tofte helped implement the separate compilation mechanism.

Andrew P. Tolmach implemented the SML/NJ debugger. He also rewrote static environments (symbol tables) in a more functional style.

Peter Weinberger implemented the first version of the copying garbage collector.

Finally, I would like to thank those who gave me helpful comments on early drafts of this book: Ron Cytron, Mary Fernandez, Lal George, Peter Lee, Greg Morrisett, Zhong Shao, David Tarditi, and Andrew Tolmach.