Modern Compiler Implementation in Java, Second Edition

by Andrew W. Appel and Jens Palsberg ISBN:052182060x

Cambridge University Press © 2002 (501 pages)

This textbook describes all phases of a compiler, and thorough coverage of current techniques in code generation and register allocation, and the compilation of functional and object-oriented languages.

Back Cover

This textbook describes all phases of a compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as the compilation of functional and object-oriented languages, which is missing from most books. The most accepted and successful techniques are described concisely, rather than as an exhaustive catalog of every possible variant. Detailed descriptions of the interfaces between modules of a compiler are illustrated with actual Java classes.

The first part of the book, Fundamentals of Compilation, is suitable for a one-semester first course in compiler design. The second part, Advanced Topics, which includes the compilation of object-oriented and functional languages, garbage collection, loop optimization, SSA form, instruction scheduling, and optimization for cache-memory hierarchies, can be used for a second-semester or graduate course.

This new edition has been rewritten extensively to include more discussion of Java and object-oriented programming concepts, such as visitor patterns. A unique feature in the newly redesigned compiler project in Java for a subset of Java itself. The project includes both front-end and back-end phases, so that students can build a complete working compiler in one semester.

About the Authors

Andrew W. Appel is Professor of Computer Science at Princeton University. He has done research and published papers on compilers, functional programming languages, runtime systems and garbage collection, type systems, and computer security; he is also the author of the book *Compiling with Continuations*. He is a designer and founder of the Standard ML of New Jersey project. In 1998, Appel was elected a Fellow of the Association for Computing Machinery for “significant research contributions in the area of programming languages and compilers” and for his work as editor-in-chief (1993-7) of the *ACM Transactions on Programming Languages and Systems*, the leading journal in the field of compilers and programming languages.

Hens Palsberg is Associate Professor of Computer Science at Purdue University. His research interests are programming languages, compilers, software engineering, and information security. He has authored more than 50 technical papers in these areas and a book with Michael Schwartzbach, *Object-Oriented Type Systems*. In 1998, he received the National Science Foundation Faculty Early Career Development Award, and in 1999, the Purdue

Modern Compiler Implementation in Java, Second Edition

Andrew W. Appel Princeton University
Jens Palsberg Purdue University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First edition published 1998
Second edition published 2002

Typefaces Times, Courier, and Optima *System* LATEX[AU]

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

Library of Congress Cataloging in Publication data

Appel, Andrew W., 1960-
Modern compiler implementation in Java /
Andrew W. Appel with Jens Palsberg.-
[2nd ed.]
p. cm.
Includes bibliographical references and index.

0-521-82060-X

1. Java (Computer program language) 2. Compilers (Computer programs) I. Palsberg, Jens. II. Title.
QA76.73.J38 A65 2002

005.4′53-dc21

There are many different derivations of the same sentence. A *leftmost* derivation is one in which the leftmost nonterminal symbol is always the one expanded; in a *rightmost* derivation, the rightmost nonterminal is always the next to be expanded.

Derivation 3.2 is neither leftmost nor rightmost; a leftmost derivation for this sentence would begin,

- *S*
- *S* ; *S*
- id := *E* ; *S*
- id := num ; *S*
- id := num ; id := *E*
- id := num ; id := *E* + *E*
- ⋮

PARSE TREES

A *parse tree* is made by connecting each symbol in a derivation to the one from which it was derived, as shown in Figure 3.3. Two different derivations can have the same parse tree.

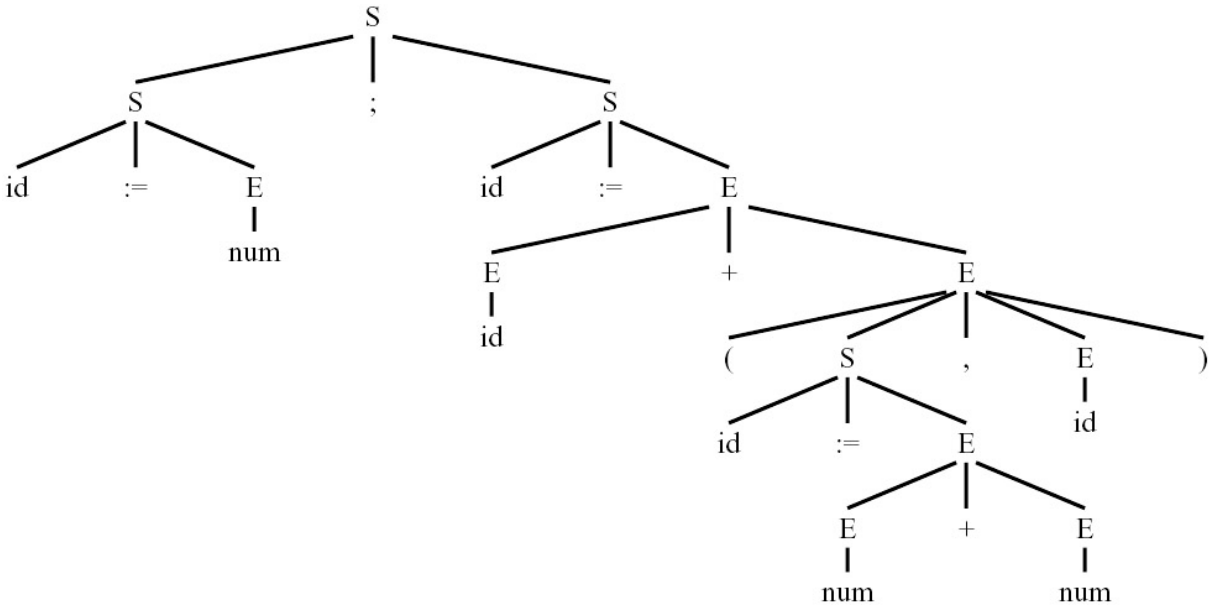

Figure 3.3: Parse tree.

AMBIGUOUS GRAMMARS

A grammar is *ambiguous* if it can derive a sentence with two different parse trees. Grammar 3.1 is ambiguous, since the sentence `id := id+id+id` has two parse trees (Figure 3.4).