CONTENTS

Preface xv

Chapter 1  Strings, Languages, and Compilers 1
1.1  Introduction 1
1.2  Basic Language Concepts 1
1.3  Basic Compiler Concepts 3
1.4  Basic Set Theory 4
1.5  Null String 6
1.6  Concatenation 7
1.7  Exponent Notation 7
1.8  Star Operator 8
1.9  Concatenation of Sets of Strings 9
1.10  Plus Operator 11
1.11  Question Mark Operator 11
1.12  Shorthand Notation for a Set Containing a Single String 12
1.13  Operator Precedence 12
1.14  Regular Expressions 13
1.15  Limitations of Regular Expressions 15
           Problems 16

Chapter 2  Context-Free Grammars, Part 1 19
2.1  Introduction 19
2.2  What is a Context-Free Grammar? 20
2.3  Derivations Using a Context-Free Grammar 21
2.4  Language Defined by a Context-Free Grammar 23
2.5  Different Ways of Representing Context-Free Grammars 25
2.6  Some Simple Grammars 26
2.7  Techniques for Generating Languages with Context-Free Grammars 29
2.8  Regular and Right Linear Grammars 35
### CONTENTS

#### 2.9 Counting with Regular Grammars
- Page 37

#### 2.0 Grammars for Lists
- Page 39

#### 2.10 An Important Language that is Not Context Free
- Page 44
  - Problems 45

#### Chapter 3  Context-Free Grammars, Part 2
- Page 49
  - 3.1 Introduction 49
  - 3.2 Parse Trees 49
  - 3.3 Leftmost and Rightmost Derivations 51
  - 3.4 Substitution 52
  - 3.5 Ambiguous Grammars 54
  - 3.6 Determining Nullable Nonterminals 59
  - 3.7 Eliminating Lambda Productions 60
  - 3.8 Eliminating Unit Productions 64
  - 3.9 Eliminating Useless Nonterminals 66
  - 3.10 Recursion Conversions 71
  - 3.11 Adding the Null String to a Language
    - Problems 76
    - 77

#### Chapter 4  Context-Free Grammars, Part 3
- Page 83
  - 4.1 Introduction 83
  - 4.2 Grammars for Arithmetic Expressions 83
  - 4.3 Specifying Associativity and Precedence in Grammars 90
  - 4.4 Backus-Naur Form 92
  - 4.5 Syntax Diagrams 94
  - 4.6 Abstract Syntax Trees and Three-Address Code 96
  - 4.7 Noncontracting Grammars 97
  - 4.8 Essentially Noncontracting Grammars 97
  - 4.9 Converting a Context-Free Grammar to an Essentially Noncontracting Grammar 98
  - 4.10 Pumping Property of Context-Free Languages
    - Problems 101
    - 104

#### Chapter 5  Chomsky's Hierarchy
- Page 107
  - 5.1 Introduction 107
  - 5.2 Context-Sensitive Productions 107
  - 5.3 Context-Sensitive Grammars 110
  - 5.4 Unrestricted Grammars 111
    - Problems 112

#### Chapter 6  Top-Down Parsing
- Page 115
  - 6.1 Introduction 115
  - 6.2 Top-Down Construction of a Parse Tree 115
  - 6.3 Parses that Fail 117
  - 6.4 A Bad Grammar for Top-Down Parsing 118
  - 6.5 Deterministic Parsers 119
  - 6.6 A Parser that Uses a Stack 120
  - 6.7 Table Representation of a Stack Parser 124
  - 6.8 Handling Productions with Nonleading Terminal 126
Chapter 7  LL(1) Grammars

7.1 Introduction 137
7.2 FIRST Set of the Right Side of a Production 137
7.3 Determining Operation Sequences 140
7.4 Determining Selection Sets of Lambda Productions 142
7.5 Whatever-Follows-Left-Follows-Rightmost Rule 145
7.6 Selection Sets for Productions with Nullable Right Sides 147
7.7 Selection Sets Containing End-of-Input Symbol 149
7.8 A Stack Parser for a Grammar with Lambda Productions 152
7.9 Converting a Non-LL(1) Grammar to an LL(1) Grammar 153
7.10 Parsing with an Ambiguous Grammar 160
7.11 Computing FIRST and FOLLOW Sets 163
Problems 165

Chapter 8  Table-Driven Stack Parser

8.1 Introduction 171
8.2 Unifying the Operations of a Stack Parser 172
8.3 Implementing a Table-Driven Stack Parser 175
8.4 Improving Our Table-Driven Stack Parser 180
8.5 Parsers that are Not Deterministic—A Digression on Theory 181
Problems 183

Chapter 9  Recursive-Descent Parsing

9.1 Introduction 185
9.2 Simple Recursive-Descent Parser 185
9.3 Handling Lambda Productions 192
9.4 A Common Error 197
9.5 Java Code for Productions 198
9.6 Left Factoring in a Recursive-Descent Parser 199
9.7 Eliminating Tail Recursion 204
9.8 Translating the Star, Plus, and Question Mark Operators 208
9.9 Doing Things Backward 210
Problems 211

Chapter 10  Recursive-Descent Translation

10.1 Introduction 215
10.2 A Simple Translation Grammar 215
10.3 Converting a Translation Grammar to Java Code 217
10.4 Specifications for a Translation Grammar 218
10.5 Passing Information During a Parse 231
10.6 L-Attributed Grammars 236
10.7 New Token Manager 238
10.8 Solving the Token Lookahead Problem 241
10.9 Code for the New Token Manager 241
10.10 Translation Grammar for Prefix Expression Compiler 253
Chapter 11 Assembly Language

11.1 Introduction 265
11.2 Structure of the J1 Computer 265
11.3 Machine Language Instructions 266
11.4 Assembly Language Instructions 268
11.5 Pushing Characters 269
11.6 aout Instruction 270
11.7 Using Labels 270
11.8 Using the Assembler 272
11.9 stay Instruction 275
11.10 Compiling an Assignment Statement 277
11.11 Compiling print and println 280
11.12 Outputting Strings 281
11.13 Inputting Decimal Numbers 283
11.14 Entry Directive 284
11.15 More Assembly Language 285
Problems 285

Chapter 12 S1—A Simple Compiler

12.1 Introduction 289
12.2 The Source Language 289
12.3 Grammar for Source Language 290
12.4 The Target Language 291
12.5 Symbol Table 292
12.6 Code Generator 293
12.7 Token Class 293
12.8 Writing the Translation Grammar 294
12.9 Implementing the S1 Compiler 299
12.10 Trying Out S1 315
12.11 Advice on Extending the S1 Compiler 318
12.12 Specifications for S2 320
Problems 324

Chapter 13 JavaCC

13.1 Introduction 331
13.2 JavaCC Extended Regular Expressions 333
13.3 JavaCC Input File 337
13.4 Specifying Actions for Regular Expressions 344
13.5 JavaCC Input File for S1j 348
13.6 Files Produced by JavaCC 355
13.7 Using the Star and Plus Operators 359
13.8 Choice Points and the Lookahead Directive 362
13.9 JavaCC's Choice Algorithm 367
13.10 Syntactic and Semantic Lookahead 371
13.11 Using JavaCC to Create a Token Manager Only 372
13.12 Using the Token Chain 373
13.13 Suppressing Warning Messages 377
Problems 387
## Chapter 14  Building on S2

14.1 Introduction 383
14.2 Extending println and print 383
14.3 Cascaded Assignment Statement 388
14.4 Unary Plus and Minus 313
14.5 readInt Statement 393
14.6 Controlling the Token Trace from the Command Line 395
14.7 Specifications for S3 Problems 396

## Chapter 15  Compiling Control Structures

15.1 Introduction 399
15.2 while Statement 399
15.3 if Statement 403
15.4 do-while Statement 407
15.5 Range Checking of Numerical Constants 408
15.6 Handling Backslash-Quote in a String 410
15.7 Handling Backslash-Quote with JavaCC 411
15.8 Universal Blocks in JavaCC 416
15.9 Handling Strings that Span Lines 418
15.10 Handling Strings that Span Lines Using JavaCC 419
15.11 SPECIAL_TOKEN Block in JavaCC 422
15.12 Error Recovery 424
15.13 Error Recovery in JavaCC 429
15.14 Specifications for S4 Problems 430

## Chapter 16  Compiling Programs in Functional Form

16.1 Introduction 435
16.2 Separate Assembly and Linking 435
16.3 Calling and Returning from Functions 439
16.4 Source Language for S5 443
16.5 Symbol Table for S5 445
16.6 Code Generator for S5 446
16.7 Translation Grammar for S5 447
16.8 Linking with a Library 457
16.9 Specifications for S5 458
16.10 Extending S5 Problems 461

## Chapter 17  Finite Automata

17.1 Introduction 465
17.2 Deterministic Finite Automata 466
17.3 Converting a DFA to a Regular Expression 468
17.4 Java Code for a DFA 472
17.5 Nondeterministic Finite Automata 474
17.6 Using an NFA as an Algorithm 476
17.7 Converting an NFA to a DFA with the Subset Algorithm 478
17.8 Converting a DFA to a Regular Grammar 479
17.9 Converting a Regular Grammar to an NFA 482
17.10 Converting a Regular Expression to an NFA 484
17.11 Finding the Minimal DFA 488
17.12 Pumping Property of Regular Languages 493
    Problems 495

Chapter 18  Capstone Project: Implementing Grep Using Compiler 499
            Technology

18.1 Introduction 499
18.2 Regular Expressions for Our Grep Program 501
18.3 Token Manager for Regular Expression 501
18.4 Grammar for Regular Expressions 503
18.5 Target Language for Our Regular Expression Compiler 503
18.6 Using an NFA for Pattern Matching 508
    Problems 513

Chapter 19  Compiling to a Register-Oriented Architecture 515

19.1 Introduction 515
19.2 Using the Register Instruction Set 516
19.3 Modifications to the Symbol Table for R1 517
19.4 Parser and Code Generator for R1 518
    Problems 526

Chapter 20  Optimization 529

20.1 Introduction 529
20.2 Using the 1dc Instruction 531
20.3 Reusing Temporary Variables 532
20.4 Constant Folding 535
20.5 Register Allocation 537
20.6 Peephole Optimization 540
    Problems 543

Chapter 21  Interpreters 547

21.1 Introduction 547
21.2 Converting S1 to II 549
21.3 Interpreting Statements that Transfer Control 552
21.4 Implementing the Compiler-Interpreter C11 553
21.5 Advantages of Interpreters 558
    Problems 559

Chapter 22  Bottom-Up Parsing 561

22.1 Introduction 561
22.2 Principles of Bottom-Up Parsing 561
22.3 Parsing with Right- versus Left-Recursive Grammars 565
22.4 Bottom-Up Parsing with an Ambiguous Grammar 566
22.5 Do-Not-Reduce Rule 569
22.6 SLR(1) Parsing 570
22.7 Shift/Reduce Conflicts 577
22.8 Reduce/Reduce Conflicts 579
22.9 LR(1) Parsing 579
    Problems 584
Chapter 23  yacc

23.1  Introduction  587
23.2  yacc Input and Output Files  587
23.3  A Simple yacc-Generated Parser  588
23.4  Passing Values Using the Value Stack  596
23.5  Using yacc With an Ambiguous Grammar  602
23.6  Passing Values down the Parse Tree  604
23.7  Implementing S1y  606
23.8  jflex  612
        Problems  618

Appendix A  Stack Instruction Set  621

Appendix B  Register Instruction Set  625

References  629

Index  631