Contents

1 Computing **1**
- 1.1 Processes, Procedures, and Computers . . . 2
- 1.2 Measuring Computing Power . . . 3
  - 1.2.1 Information . . . 3
  - 1.2.2 Representing Data . . . 8
  - 1.2.3 Growth of Computing Power . . . 12
- 1.3 Science, Engineering, and the Liberal Arts . . . 13
- 1.4 Summary and Roadmap . . . 16

Part I: Defining Procedures

2 Language **19**
- 2.1 Surface Forms and Meanings . . . 19
- 2.2 Language Construction . . . 20
- 2.3 Recursive Transition Networks . . . 22
- 2.4 Replacement Grammars . . . 26
- 2.5 Summary . . . 32

3 Programming **35**
- 3.1 Problems with Natural Languages . . . 36
- 3.2 Programming Languages . . . 37
- 3.3 Scheme . . . 39
- 3.4 Expressions . . . 40
  - 3.4.1 Primitives . . . 40
  - 3.4.2 Application Expressions . . . 41
- 3.5 Definitions . . . 44
- 3.6 Procedures . . . 45
  - 3.6.1 Making Procedures . . . 45
  - 3.6.2 Substitution Model of Evaluation . . . 46
- 3.7 Decisions . . . 48
- 3.8 Evaluation Rules . . . 50
- 3.9 Summary . . . 52

4 Problems and Procedures **53**
- 4.1 Solving Problems . . . 53
- 4.2 Composing Procedures . . . 54
  - 4.2.1 Procedures as Inputs and Outputs . . . 55
- 4.3 Recursive Problem Solving . . . 56
- 4.4 Evaluating Recursive Applications . . . 64
- 4.5 Developing Complex Programs . . . 67
  - 4.5.1 Printing . . . 68

4.5.2 Tracing 69
4.6 Summary 73

5 Data **75**
5.1 Types 75
5.2 Pairs 77
5.2.1 Making Pairs 79
5.2.2 Triples to Octuples 80
5.3 Lists 81
5.4 List Procedures 83
5.4.1 Procedures that Examine Lists 83
5.4.2 Generic Accumulators 84
5.4.3 Procedures that Construct Lists 86
5.5 Lists of Lists 90
5.6 Data Abstraction 92
5.7 Summary of Part I 102

Part II: Analyzing Procedures

6 Machines **105**
6.1 History of Computing Machines 106
6.2 Mechanizing Logic 108
6.2.1 Implementing Logic 109
6.2.2 Composing Operations 111
6.2.3 Arithmetic 114
6.3 Modeling Computing 116
6.3.1 Turing Machines 118
6.4 Summary 123

7 Cost **125**
7.1 Empirical Measurements 125
7.2 Orders of Growth 129
7.2.1 Big *O* 130
7.2.2 Omega 133
7.2.3 Theta 134
7.3 Analyzing Procedures 136
7.3.1 Input Size 136
7.3.2 Running Time 137
7.3.3 Worst Case Input 138
7.4 Growth Rates 139
7.4.1 No Growth: Constant Time 139
7.4.2 Linear Growth 140
7.4.3 Quadratic Growth 145
7.4.4 Exponential Growth 147
7.4.5 Faster than Exponential Growth 149
7.4.6 Non-terminating Procedures 149
7.5 Summary 149

8 Sorting and Searching **153**
8.1 Sorting 153
8.1.1 Best-First Sort 153
8.1.2 Insertion Sort 157

8.1.3 Quicker Sorting 158
8.1.4 Binary Trees 161
8.1.5 Quicksort 166
8.2 Searching 167
8.2.1 Unstructured Search 168
8.2.2 Binary Search 168
8.2.3 Indexed Search 169
8.3 Summary 178

Part III: Improving Expressiveness

9 Mutation 179
9.1 Assignment 179
9.2 Impact of Mutation 181
9.2.1 Names, Places, Frames, and Environments 182
9.2.2 Evaluation Rules with State 183
9.3 Mutable Pairs and Lists 186
9.4 Imperative Programming 188
9.4.1 List Mutators 188
9.4.2 Imperative Control Structures 191
9.5 Summary 193

10 Objects 195
10.1 Packaging Procedures and State 196
10.1.1 Encapsulation 196
10.1.2 Messages 197
10.1.3 Object Terminology 199
10.2 Inheritance 200
10.2.1 Implementing Subclasses 202
10.2.2 Overriding Methods 204
10.3 Object-Oriented Programming 207
10.4 Summary 209

11 Interpreters 211
11.1 Python 212
11.1.1 Python Programs 213
11.1.2 Data Types 216
11.1.3 Applications and Invocations 219
11.1.4 Control Statements 219
11.2 Parser 221
11.3 Evaluator 223
11.3.1 Primitives 223
11.3.2 If Expressions 225
11.3.3 Definitions and Names 226
11.3.4 Procedures 227
11.3.5 Application 228
11.3.6 Finishing the Interpreter 229
11.4 Lazy Evaluation 229
11.4.1 Lazy Interpreter 230
11.4.2 Lazy Programming 232

11.5 Summary 235

Part IV: The Limits of Computing

12 Computability **237**
12.1 Mechanizing Reasoning 237
12.1.1 Gödel's Incompleteness Theorem 240
12.2 The Halting Problem 241
12.3 Universality 244
12.4 Proving Non-Computability 245
12.5 Summary 251

Indexes **253**
Index 253
People 256

List of Explorations

1.1 Guessing Numbers 7
1.2 Twenty Questions 8
2.1 Power of Language Systems 29
4.1 Square Roots 62
4.2 Recipes for π 69
4.3 Recursive Definitions and Games 71
5.1 Pascal's Triangle 91
5.2 Pegboard Puzzle 93
7.1 Multiplying Like Rabbits 127
8.1 Searching the Web 177
12.1 Virus Detection 246
12.2 Busy Beavers 249

List of Figures

1.1 Using three bits to distinguish eight possible values. 6

2.1 Simple recursive transition network. 22
2.2 RTN with a cycle. 23
2.3 Recursive transition network with subnetworks. 24
2.4 Alternate *Noun* subnetwork. 24
2.5 RTN generating "Alice runs". 25
2.6 System power relationships. 30
2.7 Converting the *Number* productions to an RTN. 31
2.8 Converting the *MoreDigits* productions to an RTN. 31
2.9 Converting the *Digit* productions to an RTN. 32

3.1 Running a Scheme program. 39

4.1 A procedure maps inputs to an output. 54
4.2 Composition. 54
4.3 Circular Composition. 57
4.4 Recursive Composition. 58
4.5 Cornering the Queen. 72

5.1 Pegboard Puzzle. 93

6.1 Computing *and* with wine. 110
6.2 Computing logical *or* and *not* with wine 111
6.3 Computing *and3* by composing two *and* functions. 112
6.4 Turing Machine model. 119
6.5 Rules for checking balanced parentheses Turing Machine. 121
6.6 Checking parentheses Turing Machine. 121

7.1 Evaluation of *fibo* procedure. 128
7.2 Visualization of the sets $O(f)$, $\Omega(f)$, and $\Theta(f)$. 130
7.3 Orders of Growth. 131

8.1 Unbalanced trees. 165

9.1 Sample environments. 182
9.2 Environment created to evaluate (*bigger* 3 4). 184
9.3 Environment after evaluating (**define** *inc* (*make-adder* 1)). 185
9.4 Environment for evaluating the body of (*inc* 149). 186
9.5 Mutable pair created by evaluating (*set-mcdr! pair pair*). 187
9.6 MutableList created by evaluating (*mlist* 1 2 3). 187

10.1 Environment produced by evaluating: 197
10.2 Inheritance hierarchy. 201
10.3 Counter class hierarchy. 206

12.1 Incomplete and inconsistent axiomatic systems. 239
12.2 Universal Turing Machine. 245
12.3 Two-state Busy Beaver Machine. 249