CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

Contents

Preface **xiii**

Introduction **1**

I Basic Principles **7**

1 Expressions **13**
1.1 Expressions and Definitions . . . 14
1.1.1 Expressions . . . 14
1.1.2 Definitions . . . 15
1.1.3 Local Definitions . . . 15
1.2 Elementary Types . . . 16
1.2.1 Integers . . . 16
1.2.2 Floating-Point Numbers . . . 17
1.2.3 Floating Mode . . . 18
1.2.4 Booleans . . . 19
1.2.5 Character Strings . . . 19
1.2.6 Characters . . . 19
1.3 Cartesian Product . . . 20
1.4 Functions . . . 21
1.4.1 Defining Simple Functions . . . 21
1.4.2 Function Expressions . . . 22
1.4.3 Higher Order Function Expressions . . . 23
1.4.4 Recursive Functions . . . 24
1.4.5 Mutually Recursive Functions . . . 25
1.4.6 Functions Defined by Case . . . 25
1.4.7 The **function** Construction . . . 28
1.4.8 The **match with** Construction . . . 28
1.5 Polymorphism . . . 29
1.5.1 Type Variables . . . 29
1.5.2 Type Synthesis . . . 31
1.6 About Syntax . . . 32

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

1.6.1 Application Notation . . . 32
1.6.2 Syntactic Analysis of Caml Expressions . . . 33
1.6.3 Infix Operators Defined by the User . . . 34
1.7 Scope of Identifiers . . . 35
1.8 More about Function Types: Equivalent Function Spaces . . . 36
1.8.1 The Types of Predefined Operators . . . 37
1.9 Examples: Expressive Power . . . 37
1.9.1 Bounded Iterations . . . 37
1.9.2 Unbounded Iterations . . . 39
1.9.3 Binary Method . . . 39
1.9.4 Newton's Method . . . 40
1.9.5 About Iteration . . . 41
1.9.6 Summations . . . 42
1.10 Summary . . . 44
1.11 To Learn More . . . 44

2 Data Structures 47
2.1 Record or Named Cartesian Products . . . 47
2.2 Sums with Constructors . . . 49
2.2.1 Constant Constructors . . . 49
2.2.2 Constructors with Arguments . . . 50
2.2.3 Types with Only One Constructor and Abbreviations . . . 51
2.2.4 Recursive Types . . . 52
2.2.5 Polymorphic Types . . . 54
2.2.6 Lists . . . 55
2.2.7 Recursive Definitions of Values . . . 56
2.2.8 Abstract Types . . . 57
2.2.9 Concrete Syntax of Data Structures . . . 58
2.3 Lists . . . 59
2.3.1 General Functionals for Lists . . . 60
2.3.2 Partitioning and Sorting . . . 63
2.3.3 Representing Sets by Lists . . . 65
2.3.4 Searching in a List . . . 68
2.4 Summary . . . 70
2.5 To Learn More . . . 70

3 Semantics 73
3.1 Evaluation . . . 74
3.1.1 Evaluation as Rewriting Expressions . . . 74
3.1.2 Evaluation Strategies . . . 78
3.1.3 How to Deal with Recursion . . . 80
3.1.4 Another Way of Handling Recursion . . . 82

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

3.1.5 Behavior of Evaluation Processes 83
3.2 Defining Strategies 84
3.2.1 The Caml Strategy, or Evaluation by Value 85
3.2.2 Another Strategy: Delayed Evaluation 88
3.2.3 Extending Delayed Evaluation to Data Structures 89
3.3 Program Semantics 91
3.3.1 Rewrite Semantics 91
3.3.2 Denotational Semantics 92
3.4 Proving the Correctness of Programs 92
3.4.1 Equational Reasoning 92
3.4.2 Taking Functions as Values into Account 94
3.4.3 Reasoning by Case 94
3.4.4 Reasoning by Induction 94
3.4.5 Proof of Termination 95
3.4.6 First Example: Mirror Image of Trees 97
3.4.7 Second Example: Associativity of **append** 98
3.4.8 Example: Using Generalization 99
3.4.9 Example: Using Lemmas 100
3.4.10 Example: Proof under Hypothesis or Proof by Assumption 101
3.5 Typing Expressions 103
3.6 Summary 104
3.7 To Learn More 105

4 Imperative Aspects 107
4.1 Exceptions 107
4.1.1 Examples of Exceptions 108
4.1.2 Defining Exceptions 108
4.1.3 Creating an Exceptional Value 109
4.1.4 Filtering Exceptional Values 109
4.1.5 Using Exceptions 110
4.2 Input and Output 111
4.2.1 Sequence 111
4.2.2 The Type unit 112
4.2.3 I/O Functions 112
4.2.4 Example: Print Function 113
4.2.5 Example: Interaction 113
4.3 Character Streams 114
4.3.1 Constructing Streams 114
4.4 Modifiable Data Structures 118
4.4.1 Vectors 118
4.4.2 Records with Modifiable Fields 120
4.4.3 References 123

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

4.4.4 Circular Lists . . . 124
4.4.5 Doubly Linked Lists . . . 127
4.4.6 Comparing with Pascal, C, and Lisp . . . 128
4.5 Semantics of Destructive Operations . . . 129
4.6 Summary . . . 131
4.7 To Learn More . . . 131

II Applications 133

5 Formal Terms, Pattern Matching, Unification 135
5.1 Trees . . . 135
5.1.1 A Few Useful Functions . . . 136
5.1.2 Functionals for Trees . . . 137
5.1.3 Occurrences in a Tree . . . 140
5.1.4 Trees and Abstract Syntax . . . 140
5.2 Terms with Variables . . . 143
5.2.1 Introducing Variables . . . 143
5.2.2 Substitutions . . . 144
5.2.3 Filtering and Pattern Matching . . . 146
5.2.4 Unification . . . 147
5.3 Application: Type Synthesis . . . 148
5.3.1 Constraints on Types . . . 149
5.3.2 Generating Constraints . . . 149
5.3.3 Constraint Resolution . . . 153
5.4 Summary . . . 154
5.5 To Learn More . . . 154

6 Balanced Trees 157
6.1 Binary Trees . . . 159
6.2 Tree Traversals and Morphisms . . . 161
6.3 Order and Pre-Order Relations . . . 163
6.4 Binary Search Trees . . . 164
6.4.1 Searching for an Element . . . 166
6.4.2 Adding an Element . . . 167
6.4.3 Adding an Element to the Root . . . 168
6.4.4 Removing an Element . . . 170
6.5 Balanced Trees . . . 173
6.5.1 Rotations . . . 174
6.5.2 AVL Trees . . . 176
6.5.3 Adding Elements to an AVL Tree . . . 179
6.5.4 Removing Elements from an AVL Tree . . . 184

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

6.6 Dictionaries . . . 187
6.7 Ordered Sets . . . 189
6.8 Functional Queues . . . 192
6.9 Summary . . . 194
6.10 To Learn More . . . 194

7 Graphs and Problem Solving 195
7.1 Algorithms to Explore Graphs . . . 197
7.1.1 Naive Breadth-First Search . . . 199
7.1.2 Naive Depth-First Search . . . 202
7.1.3 Optimal Breadth-First Search . . . 202
7.1.4 Variations and Improvements . . . 203
7.1.5 Optimal Depth-First Search . . . 204
7.1.6 Exhaustive Search . . . 205
7.1.7 Computing Connected Components . . . 206
7.2 The Red Donkey . . . 206
7.3 The Game of Solitaire . . . 216
7.4 Summary . . . 229
7.5 To Learn More . . . 230

8 Syntactic Analysis 231
8.1 Regular Expressions . . . 232
8.2 Context-Free Grammar . . . 233
8.2.1 Types and Values . . . 235
8.2.2 Ambiguities . . . 236
8.3 Recognizers . . . 237
8.3.1 Recognizer Constructors . . . 238
8.3.2 Limitations . . . 240
8.3.3 Recognizers for Regular Expressions . . . 242
8.3.4 Recognizers for Grammars . . . 243
8.3.5 Various Derived Forms . . . 245
8.4 Analysis = Recognition + Values . . . 246
8.4.1 Constructors for Analyzers . . . 246
8.4.2 A Lexical Analyzer . . . 249
8.4.3 Analyzer for Nested Parentheses . . . 251
8.4.4 Derived Forms and Higher Order Analyzers . . . 252
8.4.5 An Analyzer for Arithmetic Expressions . . . 255
8.4.6 Predictive Analysis . . . 256
8.5 Streams and Pattern Matching . . . 258
8.5.1 Streams . . . 258
8.5.2 Stream Pattern Matching . . . 259
8.5.3 Lexical Analysis by Stream Pattern Matching . . . 261

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

8.5.4 Higher Order Analyzers . . . 263
8.5.5 Syntactic Analysis by Stream Pattern Matching . . . 265
8.6 Compiling Regular Expressions . . . 265
8.6.1 Finite Automata . . . 266
8.6.2 Abstract Syntax Trees for Regular Expressions . . . 267
8.6.3 Computing Null, First, and Last . . . 269
8.6.4 Computing follow . . . 271
8.6.5 Representing Automata . . . 273
8.6.6 Computing Automata . . . 273
8.6.7 Interpreting a Deterministic Finite Automaton . . . 276
8.7 Summary . . . 278
8.8 To Learn More . . . 278
8.8.1 Comparing with Other Methods of Syntactic Analysis . . . 279
8.8.2 Further References . . . 280

9 Geometry and Drawings 281
9.1 Meet MLgraph . . . 282
9.1.1 Geometric Plane . . . 282
9.1.2 Geometric Elements . . . 282
9.1.3 Constructing Images . . . 285
9.2 Drawing Trees . . . 289
9.2.1 Drawing Principles . . . 289
9.2.2 Computing the Reduction Coefficients . . . 293
9.3 Tiling . . . 298
9.3.1 The Method Used . . . 301
9.3.2 Generating Transformations . . . 304
9.3.3 Colored Tilings . . . 305
9.3.4 Handling Transformations Formally . . . 309
9.4 Tiling a Hyperbolic Plane . . . 318
9.4.1 Complex Arithmetic . . . 319
9.4.2 Hyperbolic Isometries . . . 320
9.4.3 Escher's Fish . . . 322
9.4.4 The Isometry Group . . . 324
9.4.5 Escher's Circle Limit III . . . 328
9.5 Summary . . . 332
9.6 To Learn More . . . 335
9.6.1 About PostScript . . . 335
9.6.2 About Drawing Trees . . . 335
9.6.3 About Tilings . . . 336

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

10 Exact Arithmetic **337**
10.1 Representing Integers by Lists . . . 338
10.1.1 Choosing a Base . . . 338
10.1.2 Operations with the Least Significant Digit at the Head . . 338
10.1.3 Comparisons as Operations . . . 340
10.1.4 Division as an Operation . . . 342
10.1.5 A Type for Natural Numbers (Integers for Counting) . . . 346
10.1.6 Exponentiation . . . 346
10.1.7 Computing Square Roots . . . 347
10.1.8 Reading and Printing Natural Numbers . . . 348
10.2 Other Representations of Natural Numbers . . . 349
10.2.1 Doubly Linked Circular Lists . . . 349
10.2.2 Arrays . . . 353
10.3 Signed Integers, Both Negative and Positive . . . 353
10.4 Rational Numbers . . . 354
10.5 An Application: Computing π . . . 356
10.6 Summary . . . 359
10.7 To Learn More . . . 360

III Implementation **361**

11 Evaluation **363**
11.1 Evaluation with the Help of Environments . . . 363
11.2 Writing an Evaluator in Caml . . . 366
11.3 Evaluation by Necessity = Lazy Evaluation = Delayed Evaluation 369
11.4 Summary . . . 375
11.5 To Learn More . . . 375

12 Compilation **377**
12.1 A Simplified Model of Computer Memory . . . 377
12.2 Implementation of Data Structures . . . 378
12.2.1 Coding Data . . . 380
12.2.2 CONS as an Operation . . . 381
12.2.3 Sharing Data . . . 381
12.2.4 Representing Other Types of Data . . . 384
12.2.5 Recovering Allocated Memory . . . 384
12.3 Producing Code . . . 386
12.3.1 List of Instructions . . . 386
12.3.2 Principle behind Compilation . . . 388
12.3.3 About the Compilation Scheme . . . 392
12.4 Implementation . . . 392

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
0521571839 - The Functional Approach to Programming
Guy Cousineau and Michel Mauny
Table of Contents
More information

12.4.1 Code Simulation . . . 393
12.4.2 Producing Code . . . 394
12.5 Examples . . . 397
12.6 Summary . . . 399
12.7 To Learn More . . . 399

13 Type Synthesis **401**
13.1 Type Rules . . . 401
13.1.1 Polymorphism by Substitution . . . 402
13.1.2 Parametric Polymorphism . . . 403
13.1.3 Generalizing Type Schemes . . . 403
13.1.4 Instantiation . . . 404
13.1.5 Type Rules with Parametric Polymorphism . . . 404
13.2 Writing a Program to Synthesize Types . . . 406
13.2.1 Generalizing a Type . . . 407
13.2.2 Instantiation . . . 408
13.2.3 A Few Utilities . . . 408
13.2.4 Main Function . . . 409
13.2.5 Examples . . . 412
13.3 Synthesizing Types by Destructive Unification . . . 413
13.3.1 Destructive Unification . . . 413
13.3.2 Instantiation and Generalization . . . 416
13.3.3 The Algorithm for Synthesis . . . 417
13.4 Summary . . . 418
13.5 To Learn More . . . 419

Quick Reference to the Syntax of Caml Light **421**

How to Get Caml, MLgraph, and the Examples **425**

References **427**

Index **431**

Index of Types and Functions **443**

© Cambridge University Press www.cambridge.org