

Contents

Foreword	xiii
Foreword to Structure and Interpretation of Computer Programs, 1984	xvii
Preface	xxi
Prefaces to Structure and Interpretation of Computer Programs, 1996 & 1984	xxiii
Acknowledgments	xxvii
1 Building Abstractions with Functions	1
1.1 The Elements of Programming	3
1.1.1 Expressions	3
1.1.2 Naming and the Environment	5
1.1.3 Evaluating Operator Combinations	6
1.1.4 Compound Functions	8
1.1.5 The Substitution Model for Function Application	11
1.1.6 Conditional Expressions and Predicates	13
1.1.7 Example: Square Roots by Newton's Method	18
1.1.8 Functions as Black-Box Abstractions	21
1.2 Functions and the Processes They Generate	26
1.2.1 Linear Recursion and Iteration	27
1.2.2 Tree Recursion	32
1.2.3 Orders of Growth	36
1.2.4 Exponentiation	38
1.2.5 Greatest Common Divisors	41
1.2.6 Example: Testing for Primality	43
1.3 Formulating Abstractions with Higher-Order Functions	48
1.3.1 Functions as Arguments	49
1.3.2 Constructing Functions using Lambda Expressions	53
1.3.3 Functions as General Methods	58
1.3.4 Functions as Returned Values	63
2 Building Abstractions with Data	69
2.1 Introduction to Data Abstraction	72
2.1.1 Example: Arithmetic Operations for Rational Numbers	72
2.1.2 Abstraction Barriers	76
2.1.3 What Is Meant by Data?	78
2.1.4 Extended Exercise: Interval Arithmetic	81

2.2	Hierarchical Data and the Closure Property	84
2.2.1	Representing Sequences	85
2.2.2	Hierarchical Structures	93
2.2.3	Sequences as Conventional Interfaces	98
2.2.4	Example: A Picture Language	110
2.3	Symbolic Data	124
2.3.1	Strings	124
2.3.2	Example: Symbolic Differentiation	126
2.3.3	Example: Representing Sets	131
2.3.4	Example: Huffman Encoding Trees	140
2.4	Multiple Representations for Abstract Data	147
2.4.1	Representations for Complex Numbers	149
2.4.2	Tagged data	152
2.4.3	Data-Directed Programming and Additivity	156
2.5	Systems with Generic Operations	163
2.5.1	Generic Arithmetic Operations	164
2.5.2	Combining Data of Different Types	169
2.5.3	Example: Symbolic Algebra	176
3	Modularity, Objects, and State	189
3.1	Assignment and Local State	190
3.1.1	Local State Variables	190
3.1.2	The Benefits of Introducing Assignment	197
3.1.3	The Costs of Introducing Assignment	200
3.2	The Environment Model of Evaluation	206
3.2.1	The Rules for Evaluation	207
3.2.2	Applying Simple Functions	210
3.2.3	Frames as the Repository of Local State	213
3.2.4	Internal Declarations	218
3.3	Modeling with Mutable Data	222
3.3.1	Mutable List Structure	222
3.3.2	Representing Queues	231
3.3.3	Representing Tables	235
3.3.4	A Simulator for Digital Circuits	241
3.3.5	Propagation of Constraints	252
3.4	Concurrency: Time Is of the Essence	263
3.4.1	The Nature of Time in Concurrent Systems	264
3.4.2	Mechanisms for Controlling Concurrency	268
3.5	Streams	280
3.5.1	Streams Are Delayed Lists	281
3.5.2	Infinite Streams	288
3.5.3	Exploiting the Stream Paradigm	295
3.5.4	Streams and Delayed Evaluation	305
3.5.5	Modularity of Functional Programs and Modularity of Objects	311

4	Metalinguistic Abstraction	317
4.1	The Metacircular Evaluator	319
4.1.1	The Core of the Evaluator	321
4.1.2	Representing Components	328
4.1.3	Evaluator Data Structures	339
4.1.4	Running the Evaluator as a Program	344
4.1.5	Data as Programs	348
4.1.6	Internal Declarations	351
4.1.7	Separating Syntactic Analysis from Execution	355
4.2	Lazy Evaluation	360
4.2.1	Normal Order and Applicative Order	361
4.2.2	An Interpreter with Lazy Evaluation	362
4.2.3	Streams as Lazy Lists	370
4.3	Nondeterministic Computing	373
4.3.1	Search and <code>amb</code>	374
4.3.2	Examples of Nondeterministic Programs	378
4.3.3	Implementing the <code>amb</code> Evaluator	386
4.4	Logic Programming	398
4.4.1	Deductive Information Retrieval	401
4.4.2	How the Query System Works	411
4.4.3	Is Logic Programming Mathematical Logic?	419
4.4.4	Implementing the Query System	424
5	Computing with Register Machines	449
5.1	Designing Register Machines	450
5.1.1	A Language for Describing Register Machines	452
5.1.2	Abstraction in Machine Design	456
5.1.3	Subroutines	457
5.1.4	Using a Stack to Implement Recursion	462
5.1.5	Instruction Summary	468
5.2	A Register-Machine Simulator	468
5.2.1	The Machine Model	470
5.2.2	The Assembler	474
5.2.3	Instructions and Their Execution Functions	477
5.2.4	Monitoring Machine Performance	484
5.3	Storage Allocation and Garbage Collection	487
5.3.1	Memory as Vectors	488
5.3.2	Maintaining the Illusion of Infinite Memory	493
5.4	The Explicit-Control Evaluator	499
5.4.1	The Dispatcher and Basic Evaluation	500
5.4.2	Evaluating Function Applications	504
5.4.3	Blocks, Assignments, and Declarations	512
5.4.4	Running the Evaluator	513

5.5	Compilation	519
5.5.1	Structure of the Compiler	522
5.5.2	Compiling Components	527
5.5.3	Compiling Applications and Return Statements	535
5.5.4	Combining Instruction Sequences	543
5.5.5	An Example of Compiled Code	546
5.5.6	Lexical Addressing	554
5.5.7	Interfacing Compiled Code to the Evaluator	557
References		565
Index		571
List of Exercises		607