

# Contents

Contents	vii
Foreword	xi
Preface to the Second Edition	xv
Preface to the First Edition	xvii
Acknowledgments	xxi
<b>1 Building Abstractions with Procedures</b>	<b>1</b>
1.1 The Elements of Programming	4
1.1.1 Expressions	5
1.1.2 Naming and the Environment	7
1.1.3 Evaluating Combinations	9
1.1.4 Compound Procedures	11
1.1.5 The Substitution Model for Procedure Application	13
1.1.6 Conditional Expressions and Predicates	17
1.1.7 Example: Square Roots by Newton's Method	21
1.1.8 Procedures as Black-Box Abstractions	26
1.2 Procedures and the Processes They Generate	31
1.2.1 Linear Recursion and Iteration	32
1.2.2 Tree Recursion	37
1.2.3 Orders of Growth	42
1.2.4 Exponentiation	44
1.2.5 Greatest Common Divisors	48
1.2.6 Example: Testing for Primality	50
1.3 Formulating Abstractions with Higher-Order Procedures	56
1.3.1 Procedures as Arguments	57
1.3.2 Constructing Procedures Using Lambda	62
1.3.3 Procedures as General Methods	66
1.3.4 Procedures as Returned Values	72



<b>2</b>	<b>Building Abstractions with Data</b>	<b>79</b>
2.1	Introduction to Data Abstraction	83
2.1.1	Example: Arithmetic Operations for Rational Numbers	83
2.1.2	Abstraction Barriers	87
2.1.3	What Is Meant by Data?	90
2.1.4	Extended Exercise: Interval Arithmetic	93
2.2	Hierarchical Data and the Closure Property	97
2.2.1	Representing Sequences	99
2.2.2	Hierarchical Structures	107
2.2.3	Sequences as Conventional Interfaces	113
2.2.4	Example: A Picture Language	126
2.3	Symbolic Data	142
2.3.1	Quotation	142
2.3.2	Example: Symbolic Differentiation	145
2.3.3	Example: Representing Sets	151
2.3.4	Example: Huffman Encoding Trees	161
2.4	Multiple Representations for Abstract Data	169
2.4.1	Representations for Complex Numbers	171
2.4.2	Tagged data	175
2.4.3	Data-Directed Programming and Additivity	179
2.5	Systems with Generic Operations	187
2.5.1	Generic Arithmetic Operations	189
2.5.2	Combining Data of Different Types	193
2.5.3	Example: Symbolic Algebra	202
<b>3</b>	<b>Modularity, Objects, and State</b>	<b>217</b>
3.1	Assignment and Local State	218
3.1.1	Local State Variables	219
3.1.2	The Benefits of Introducing Assignment	225
3.1.3	The Costs of Introducing Assignment	229
3.2	The Environment Model of Evaluation	236
3.2.1	The Rules for Evaluation	238
3.2.2	Applying Simple Procedures	241
3.2.3	Frames as the Repository of Local State	244
3.2.4	Internal Definitions	249
3.3	Modeling with Mutable Data	251
3.3.1	Mutable List Structure	252



3.3.2	Representing Queues	261
3.3.3	Representing Tables	266
3.3.4	A Simulator for Digital Circuits	273
3.3.5	Propagation of Constraints	285
3.4	Concurrency: Time Is of the Essence	297
3.4.1	The Nature of Time in Concurrent Systems	298
3.4.2	Mechanisms for Controlling Concurrency	303
3.5	Streams	316
3.5.1	Streams Are Delayed Lists	317
3.5.2	Infinite Streams	326
3.5.3	Exploiting the Stream Paradigm	334
3.5.4	Streams and Delayed Evaluation	346
3.5.5	Modularity of Functional Programs and Modularity of Objects	352
<b>4</b>	<b>Metalinguistic Abstraction</b>	<b>359</b>
4.1	The Metacircular Evaluator	362
4.1.1	The Core of the Evaluator	364
4.1.2	Representing Expressions	368
4.1.3	Evaluator Data Structures	376
4.1.4	Running the Evaluator as a Program	381
4.1.5	Data as Programs	384
4.1.6	Internal Definitions	388
4.1.7	Separating Syntactic Analysis from Execution	393
4.2	Variations on a Scheme—Lazy Evaluation	398
4.2.1	Normal Order and Applicative Order	399
4.2.2	An Interpreter with Lazy Evaluation	401
4.2.3	Streams as Lazy Lists	409
4.3	Variations on a Scheme—Nondeterministic Computing	412
4.3.1	Amb and Search	414
4.3.2	Examples of Nondeterministic Programs	418
4.3.3	Implementing the Amb Evaluator	426
4.4	Logic Programming	438
4.4.1	Deductive Information Retrieval	441
4.4.2	How the Query System Works	453
4.4.3	Is Logic Programming Mathematical Logic?	462
4.4.4	Implementing the Query System	468



<b>5</b>	<b>Computing with Register Machines</b>	<b>491</b>
5.1	Designing Register Machines	492
5.1.1	A Language for Describing Register Machines	494
5.1.2	Abstraction in Machine Design	499
5.1.3	Subroutines	502
5.1.4	Using a Stack to Implement Recursion	506
5.1.5	Instruction Summary	512
5.2	A Register-Machine Simulator	513
5.2.1	The Machine Model	515
5.2.2	The Assembler	520
5.2.3	Generating Execution Procedures for Instructions	523
5.2.4	Monitoring Machine Performance	530
5.3	Storage Allocation and Garbage Collection	533
5.3.1	Memory as Vectors	534
5.3.2	Maintaining the Illusion of Infinite Memory	540
5.4	The Explicit-Control Evaluator	547
5.4.1	The Core of the Explicit-Control Evaluator	549
5.4.2	Sequence Evaluation and Tail Recursion	555
5.4.3	Conditionals, Assignments, and Definitions	558
5.4.4	Running the Evaluator	560
5.5	Compilation	566
5.5.1	Structure of the Compiler	569
5.5.2	Compiling Expressions	574
5.5.3	Compiling Combinations	581
5.5.4	Combining Instruction Sequences	587
5.5.5	An Example of Compiled Code	591
5.5.6	Lexical Addressing	600
5.5.7	Interfacing Compiled Code to the Evaluator	603
	References	611
	List of Exercises	619
	Index	621