Contents

Foreword

Preface to the Second Edition

Preface to the First Edition

Acknowledgments

1 Building Abstractions with Procedures
- 1.1 The Elements of Programming
  - 1.1.1 Expressions
  - 1.1.2 Naming and the Environment
  - 1.1.3 Evaluating Combinations
  - 1.1.4 Compound Procedures
  - 1.1.5 The Substitution Model for Procedure Application
  - 1.1.6 Conditional Expressions and Predicates
  - 1.1.7 Example: Square Roots by Newton's Method
  - 1.1.8 Procedures as Black-Box Abstractions
- 1.2 Procedures and the Processes They Generate
  - 1.2.1 Linear Recursion and Iteration
  - 1.2.2 Tree Recursion
  - 1.2.3 Orders of Growth
  - 1.2.4 Exponentiation
  - 1.2.5 Greatest Common Divisors
  - 1.2.6 Example: Testing for Primality
- 1.3 Formulating Abstractions with Higher-Order Procedures
  - 1.3.1 Procedures as Arguments
  - 1.3.2 Constructing Procedures Using `Lambda`
  - 1.3.3 Procedures as General Methods
  - 1.3.4 Procedures as Returned Values

2 Building Abstractions with Data
- 2.1 Introduction to Data Abstraction
  - 2.1.1 Example: Arithmetic Operations for Rational Numbers
  - 2.1.2 Abstraction Barriers
  - 2.1.3 What Is Meant by Data?
  - 2.1.4 Extended Exercise: Interval Arithmetic
- 2.2 Hierarchical Data and the Closure Property
  - 2.2.1 Representing Sequences
  - 2.2.2 Hierarchical Structures
  - 2.2.3 Sequences as Conventional Interfaces
  - 2.2.4 Example: A Picture Language
- 2.3 Symbolic Data
  - 2.3.1 Quotation
  - 2.3.2 Example: Symbolic Differentiation
  - 2.3.3 Example: Representing Sets
  - 2.3.4 Example: Huffman Encoding Trees

- 2.4 Multiple Representations for Abstract Data
  - 2.4.1 Representations for Complex Numbers
  - 2.4.2 Tagged data
  - 2.4.3 Data-Directed Programming and Additivity
- 2.5 Systems with Generic Operations
  - 2.5.1 Generic Arithmetic Operations
  - 2.5.2 Combining Data of Different Types
  - 2.5.3 Example: Symbolic Algebra

3 Modularity, Objects, and State

- 3.1 Assignment and Local State
  - 3.1.1 Local State Variables
  - 3.1.2 The Benefits of Introducing Assignment
  - 3.1.3 The Costs of Introducing Assignment
- 3.2 The Environment Model of Evaluation
  - 3.2.1 The Rules for Evaluation
  - 3.2.2 Applying Simple Procedures
  - 3.2.3 Frames as the Repository of Local State
  - 3.2.4 Internal Definitions
- 3.3 Modeling with Mutable Data
  - 3.3.1 Mutable List Structure
  - 3.3.2 Representing Queues
  - 3.3.3 Representing Tables
  - 3.3.4 A Simulator for Digital Circuits
  - 3.3.5 Propagation of Constraints
- 3.4 Concurrency: Time Is of the Essence
  - 3.4.1 The Nature of Time in Concurrent Systems
  - 3.4.2 Mechanisms for Controlling Concurrency
- 3.5 Streams
  - 3.5.1 Streams Are Delayed Lists
  - 3.5.2 Infinite Streams
  - 3.5.3 Exploiting the Stream Paradigm
  - 3.5.4 Streams and Delayed Evaluation
  - 3.5.5 Modularity of Functional Programs and Modularity of Objects

4 Metalinguistic Abstraction

- 4.1 The Metacircular Evaluator
  - 4.1.1 The Core of the Evaluator
  - 4.1.2 Representing Expressions
  - 4.1.3 Evaluator Data Structures
  - 4.1.4 Running the Evaluator as a Program
  - 4.1.5 Data as Programs
  - 4.1.6 Internal Definitions
  - 4.1.7 Separating Syntactic Analysis from Execution
- 4.2 Variations on a Scheme -- Lazy Evaluation
  - 4.2.1 Normal Order and Applicative Order
  - 4.2.2 An Interpreter with Lazy Evaluation
  - 4.2.3 Streams as Lazy Lists
- 4.3 Variations on a Scheme -- Nondeterministic Computing
  - 4.3.1 Amb and Search
  - 4.3.2 Examples of Nondeterministic Programs
  - 4.3.3 Implementing the `Amb` Evaluator
- 4.4 Logic Programming
  - 4.4.1 Deductive Information Retrieval
  - 4.4.2 How the Query System Works
  - 4.4.3 Is Logic Programming Mathematical Logic?
  - 4.4.4 Implementing the Query System

5 Computing with Register Machines
- 5.1 Designing Register Machines
  - 5.1.1 A Language for Describing Register Machines
  - 5.1.2 Abstraction in Machine Design
  - 5.1.3 Subroutines
  - 5.1.4 Using a Stack to Implement Recursion
  - 5.1.5 Instruction Summary
- 5.2 A Register-Machine Simulator
  - 5.2.1 The Machine Model
  - 5.2.2 The Assembler
  - 5.2.3 Generating Execution Procedures for Instructions
  - 5.2.4 Monitoring Machine Performance
- 5.3 Storage Allocation and Garbage Collection
  - 5.3.1 Memory as Vectors
  - 5.3.2 Maintaining the Illusion of Infinite Memory
- 5.4 The Explicit-Control Evaluator
  - 5.4.1 The Core of the Explicit-Control Evaluator
  - 5.4.2 Sequence Evaluation and Tail Recursion
  - 5.4.3 Conditionals, Assignments, and Definitions
  - 5.4.4 Running the Evaluator
- 5.5 Compilation
  - 5.5.1 Structure of the Compiler
  - 5.5.2 Compiling Expressions
  - 5.5.3 Compiling Combinations
  - 5.5.4 Combining Instruction Sequences
  - 5.5.5 An Example of Compiled Code
  - 5.5.6 Lexical Addressing
  - 5.5.7 Interfacing Compiled Code to the Evaluator

References

List of Exercises

Index

[Go to first, previous, next page; contents; index]