

Contents

- 1 Abstract Machines 1**
 - 1.1 The Concepts of Abstract Machine and of Interpreter 1
 - 1.1.1 The Interpreter 2
 - 1.1.2 An Example of an Abstract Machine: The Hardware Machine 5
 - 1.2 Implementation of a Language 9
 - 1.2.1 Implementation of an Abstract Machine 9
 - 1.2.2 Implementation: The Ideal Case 13
 - 1.2.3 Implementation: The Real Case and The Intermediate Machine 17
 - 1.3 Hierarchies of Abstract Machines 21
 - 1.4 Chapter Summary 24
 - 1.5 Bibliographic Notes 24
 - 1.6 Exercises 24
 - References 25
- 2 How to Describe a Programming Language 27**
 - 2.1 Levels of Description 27
 - 2.2 Grammar and Syntax 28
 - 2.2.1 Context-Free Grammars 30
 - 2.3 Contextual Syntactic Constraints 39
 - 2.4 Compilers 41
 - 2.5 Semantics 45
 - 2.6 Pragmatics 52
 - 2.7 Implementation 52
 - 2.8 Chapter Summary 53
 - 2.9 Bibliographical Notes 53
 - 2.10 Exercises 53
 - References 54

3	Foundations	57
3.1	The Halting Problem	57
3.2	Expressiveness of Programming Languages	59
3.3	Formalisms for Computability	60
3.4	There are More Functions than Algorithms	61
3.5	Chapter Summary	63
3.6	Bibliographical Notes	64
3.7	Exercises	64
	References	65
4	Names and The Environment	67
4.1	Names and Denotable Objects	67
4.1.1	Denotable Objects	69
4.2	Environments and Blocks	70
4.2.1	Blocks	71
4.2.2	Types of Environment	72
4.2.3	Operations on Environments	75
4.3	Scope Rules	77
4.3.1	Static Scope	78
4.3.2	Dynamic Scope	80
4.3.3	Some Scope Problems	82
4.4	Chapter Summary	85
4.5	Bibliographical Notes	86
4.6	Exercises	87
	References	90
5	Memory Management	91
5.1	Techniques for Memory Management	91
5.2	Static Memory Management	93
5.3	Dynamic Memory Management Using Stacks	93
5.3.1	Activation Records for In-line Blocks	96
5.3.2	Activation Records for Procedures	97
5.3.3	Stack Management	99
5.4	Dynamic Management Using a Heap	101
5.4.1	Fixed-Length Blocks	101
5.4.2	Variable-Length Blocks	103
5.5	Implementation of Scope Rules	105
5.5.1	Static Scope: The Static Chain	105
5.5.2	Static Scope: The Display	109
5.5.3	Dynamic Scope: Association Lists and CRT	111
5.6	Chapter Summary	115
5.7	Bibliographic Notes	116
5.8	Exercises	116
	References	118

6	Control Structure	119
6.1	Expressions	119
6.1.1	Expression Syntax	120
6.1.2	Semantics of Expressions	123
6.1.3	Evaluation of Expressions	125
6.2	The Concept of Command	129
6.2.1	The Variable	130
6.2.2	Assignment	131
6.3	Sequence Control Commands	136
6.3.1	Commands for Explicit Sequence Control	136
6.3.2	Conditional Commands	140
6.3.3	Iterative Commands	144
6.4	Structured Programming	150
6.5	Recursion	152
6.5.1	Tail Recursion	155
6.5.2	Recursion or Iteration?	159
6.6	Chapter Summary	160
6.7	Bibliographical Notes	161
6.8	Exercises	161
	References	163
7	Control Abstraction	165
7.1	Subprograms	166
7.1.1	Functional Abstraction	167
7.1.2	Parameter Passing	169
7.2	Higher-Order Functions	178
7.2.1	Functions as Parameters	179
7.2.2	Functions as Results	184
7.3	Exceptions	186
7.3.1	Implementing Exceptions	190
7.4	Chapter Summary	191
7.5	Bibliographical Notes	193
7.6	Exercises	194
	References	196
8	Structuring Data	197
8.1	Data Types	197
8.1.1	Types as Support for Conceptual Organisation	198
8.1.2	Types for Correctness	199
8.1.3	Types and Implementation	200
8.2	Type Systems	201
8.2.1	Static and Dynamic Checking	202
8.3	Scalar Types	203
8.3.1	Booleans	204
8.3.2	Characters	204

8.3.3	Integers	205
8.3.4	Reals	205
8.3.5	Fixed Point	205
8.3.6	Complex	206
8.3.7	Void	207
8.3.8	Enumerations	207
8.3.9	Intervals	208
8.3.10	Ordered Types	209
8.4	Composite Types	209
8.4.1	Records	209
8.4.2	Variant Records and Unions	211
8.4.3	Arrays	216
8.4.4	Sets	221
8.4.5	Pointers	222
8.4.6	Recursive Types	227
8.4.7	Functions	229
8.5	Equivalence	230
8.5.1	Equivalence by Name	231
8.5.2	Structural Equivalence	232
8.6	Compatibility and Conversion	234
8.7	Polymorphism	237
8.7.1	Overloading	238
8.7.2	Universal Parametric Polymorphism	239
8.7.3	Subtype Universal Polymorphism	241
8.7.4	Remarks on the Implementation	242
8.8	Type Checking and Inference	244
8.9	Safety: An Evaluation	246
8.10	Avoiding Dangling References	247
8.10.1	Tombstone	248
8.10.2	Locks and Keys	249
8.11	Garbage Collection	250
8.11.1	Reference Counting	251
8.11.2	Mark and Sweep	253
8.11.3	Interlude: Pointer Reversal	254
8.11.4	Mark and Compact	255
8.11.5	Copy	255
8.12	Chapter Summary	258
8.13	Bibliographic Notes	259
8.14	Exercises	259
	References	262
9	Data Abstraction	265
9.1	Abstract Data Types	265
9.2	Information Hiding	268
9.2.1	Representation Independence	271

9.3	Modules	271
9.4	Chapter Summary	272
9.5	Bibliographical Notes	275
9.6	Exercises	275
	References	276
10	The Object-Oriented Paradigm	277
10.1	The Limits of Abstract Data Types	277
10.1.1	A First Review	281
10.2	Fundamental Concepts	281
10.2.1	Objects	282
10.2.2	Classes	283
10.2.3	Encapsulation	287
10.2.4	Subtypes	287
10.2.5	Inheritance	292
10.2.6	Dynamic Method Lookup	297
10.3	Implementation Aspects	301
10.3.1	Single Inheritance	303
10.3.2	The Problem of Fragile Base Class	305
10.3.3	Dynamic Method Dispatch in the JVM	306
10.3.4	Multiple Inheritance	309
10.4	Polymorphism and Generics	314
10.4.1	Subtype Polymorphism	315
10.4.2	Generics in Java	317
10.4.3	Implementation of Generics in Java	321
10.4.4	Generics, Arrays and Subtype Hierarchy	323
10.4.5	Covariant and Contravariant Overriding	325
10.5	Chapter Summary	328
10.6	Bibliographical Notes	328
10.7	Exercises	329
	References	331
11	The Functional Paradigm	333
11.1	Computations without State	333
11.1.1	Expressions and Functions	335
11.1.2	Computation as Reduction	337
11.1.3	The Fundamental Ingredients	338
11.2	Evaluation	339
11.2.1	Values	340
11.2.2	Capture-Free Substitution	340
11.2.3	Evaluation Strategies	341
11.2.4	Comparison of the Strategies	343
11.3	Programming in a Functional Language	345
11.3.1	Local Environment	345
11.3.2	Interactiveness	346

11.3.3	Types	346
11.3.4	Pattern Matching	347
11.3.5	Infinite Objects	349
11.3.6	Imperative Aspects	350
11.4	Implementation: The SECD Machine	353
11.5	The Functional Paradigm: An Assessment	355
11.6	Fundamentals: The λ -calculus	358
11.7	Chapter Summary	364
11.8	Bibliographical Note	365
11.9	Exercises	365
	References	366
12	The Logic Programming Paradigm	369
12.1	Deduction as Computation	369
12.1.1	An Example	371
12.2	Syntax	374
12.2.1	The Language of First-Order Logic	374
12.2.2	Logic Programs	376
12.3	Theory of Unification	377
12.3.1	The Logic Variable	377
12.3.2	Substitution	379
12.3.3	Most General Unifier	381
12.3.4	A Unification Algorithm	383
12.4	The Computational Model	387
12.4.1	The Herbrand Universe	387
12.4.2	Declarative and Procedural Interpretation	388
12.4.3	Procedure Calls	389
12.4.4	Control: Non-determinism	392
12.4.5	Some Examples	395
12.5	Extensions	398
12.5.1	Prolog	398
12.5.2	Logic Programming and Databases	403
12.5.3	Logic Programming with Constraints	404
12.6	Advantages and Disadvantages of the Logic Paradigm	406
12.7	Chapter Summary	408
12.8	Bibliographical Notes	409
12.9	Exercises	409
	References	411
13	A Short Historical Perspective	413
13.1	Beginnings	413
13.2	Factors in the Development of Languages	415
13.3	1950s and 60s	417
13.4	The 1970s	421
13.5	The 1980s	425
13.6	1990s	428

13.7 Chapter Summary 430

13.8 Bibliographical Notes 431

 References 431

Index 433



<http://www.springer.com/978-1-84882-913-8>

Programming Languages: Principles and Paradigms

Gabbrielli, M.; Martini, S.

2010, XX, 440 p., Softcover

ISBN: 978-1-84882-913-8