

---

# Contents

<b>1. The Extensible Language</b>	1	4.4. Search	48
1.1. Design by Evolution	1	4.5. Mapping	53
1.2. Programming Bottom-Up	3	4.6. I/O	56
1.3. Extensible Software	5	4.7. Symbols and Strings	57
1.4. Extending Lisp	6	4.8. Density	59
1.5. Why Lisp (or When)	8		
<b>2. Functions</b>	9	<b>5. Returning Functions</b>	61
2.1. Functions as Data	9	5.1. Common Lisp Evolves	61
2.2. Defining Functions	10	5.2. Orthogonality	63
2.3. Functional Arguments	13	5.3. Memoizing	65
2.4. Functions as Properties	15	5.4. Composing Functions	66
2.5. Scope	16	5.5. Recursion on Cdrs	68
2.6. Closures	17	5.6. Recursion on Subtrees	70
2.7. Local Functions	21	5.7. When to Build Functions	75
2.8. Tail-Recursion	22		
2.9. Compilation	24	<b>6. Functions as Representation</b>	76
2.10. Functions from Lists	27	6.1. Networks	76
		6.2. Compiling Networks	79
<b>3. Functional Programming</b>	28	6.3. Looking Forward	81
3.1. Functional Design	28		
3.2. Imperative Outside-In	33	<b>7. Macros</b>	82
3.3. Functional Interfaces	35	7.1. How Macros Work	82
3.4. Interactive Programming	37	7.2. Backquote	84
		7.3. Defining Simple Macros	88
<b>4. Utility Functions</b>	40	7.4. Testing Macroexpansion	91
4.1. Birth of a Utility	40	7.5. Destructuring in Parameter Lists	93
4.2. Invest in Abstraction	43	7.6. A Model of Macros	95
4.3. Operations on Lists	44	7.7. Macros as Programs	96

- 7.8. Macro Style 99
- 7.9. Dependence on Macros 101
- 7.10. Macros from Functions 102
- 7.11. Symbol Macros 105
- 8. When to Use Macros 106**
  - 8.1. When Nothing Else Will Do 106
  - 8.2. Macro or Function? 109
  - 8.3. Applications for Macros 111
- 9. Variable Capture 118**
  - 9.1. Macro Argument Capture 118
  - 9.2. Free Symbol Capture 119
  - 9.3. When Capture Occurs 121
  - 9.4. Avoiding Capture with Better Names 125
  - 9.5. Avoiding Capture by Prior Evaluation 125
  - 9.6. Avoiding Capture with Gensyms 128
  - 9.7. Avoiding Capture with Packages 130
  - 9.8. Capture in Other Name-Spaces 130
  - 9.9. Why Bother? 132
- 10. Other Macro Pitfalls 133**
  - 10.1. Number of Evaluations 133
  - 10.2. Order of Evaluation 135
  - 10.3. Non-functional Expanders 136
  - 10.4. Recursion 139
- 11. Classic Macros 143**
  - 11.1. Creating Context 143
  - 11.2. The `with-` Macro 147
  - 11.3. Conditional Evaluation 150
  - 11.4. Iteration 154
  - 11.5. Iteration with Multiple Values 158
  - 11.6. Need for Macros 161
- 12. Generalized Variables 165**
  - 12.1. The Concept 165
  - 12.2. The Multiple Evaluation Problem 167
  - 12.3. New Utilities 169
  - 12.4. More Complex Utilities 171
  - 12.5. Defining Inversions 178
- 13. Computation at Compile-Time 181**
  - 13.1. New Utilities 181
  - 13.2. Example: Bezier Curves 185
  - 13.3. Applications 186
- 14. Anaphoric Macros 189**
  - 14.1. Anaphoric Variants 189
  - 14.2. Failure 195
  - 14.3. Referential Transparency 198
- 15. Macros Returning Functions 201**
  - 15.1. Building Functions 201
  - 15.2. Recursion on Cdrs 204
  - 15.3. Recursion on Subtrees 208
  - 15.4. Lazy Evaluation 211
- 16. Macro-Defining Macros 213**
  - 16.1. Abbreviations 213
  - 16.2. Properties 216
  - 16.3. Anaphoric Macros 218
- 17. Read-Macros 224**
  - 17.1. Macro Characters 224
  - 17.2. Dispatching Macro Characters 226
  - 17.3. Delimiters 227
  - 17.4. When What Happens 229
- 18. Destructuring 230**
  - 18.1. Destructuring on Lists 230
  - 18.2. Other Structures 231
  - 18.3. Reference 236
  - 18.4. Matching 238

- 19. A Query Compiler** 246
  - 19.1. The Database 247
  - 19.2. Pattern-Matching Queries 248
  - 19.3. A Query Interpreter 250
  - 19.4. Restrictions on Binding 252
  - 19.5. A Query Compiler 254
- 20. Continuations** 258
  - 20.1. Scheme Continuations 258
  - 20.2. Continuation-Passing  
Macros 266
  - 20.3. Code-Walkers and CPS  
Conversion 272
- 21. Multiple Processes** 275
  - 21.1. The Process Abstraction 275
  - 21.2. Implementation 277
  - 21.3. The Less-than-Rapid  
Prototype 284
- 22. Nondeterminism** 286
  - 22.1. The Concept 286
  - 22.2. Search 290
  - 22.3. Scheme Implementation 292
  - 22.4. Common Lisp  
Implementation 294
  - 22.5. Cuts 298
  - 22.6. True Nondeterminism 302
- 23. Parsing with ATNs** 305
  - 23.1. Background 305
  - 23.2. The Formalism 306
  - 23.3. Nondeterminism 308
  - 23.4. An ATN Compiler 309
  - 23.5. A Sample ATN 314
- 24. Prolog** 321
  - 24.1. Concepts 321
  - 24.2. An Interpreter 323
  - 24.3. Rules 329
  - 24.4. The Need for  
Nondeterminism 333
  - 24.5. New Implementation 334
  - 24.6. Adding Prolog Features 337
- 24.7. Examples 344
- 24.8. The Senses of Compile 346
- 25. Object-Oriented Lisp** 348
  - 25.1. Plus ça Change 348
  - 25.2. Objects in Plain Lisp 349
  - 25.3. Classes and Instances 364
  - 25.4. Methods 368
  - 25.5. Auxiliary Methods and  
Combination 374
  - 25.6. CLOS and Lisp 377
  - 25.7. When to Object 379