

Contents

Preface	vii
Acknowledgments	ix
0 Introduction	xv
1 A First Look at LISP	1
1.1 The origins of LISP.	1
1.2 A glimpse into the structure of LISP.	4
1.2.1 Data.	5
1.2.2 Functions.	6
1.2.3 Expressions in the language.	8
1.2.4 QUOTE.	9
1.2.5 Logical terms.	11
1.2.6 Branching in computations.	12
1.2.7 Lambda functions.	14
1.2.8 Assignments.	16
1.3 Additional reading.	19
2 Elementary Programming	20
2.1 Programming straightforward computations.	20
2.2 Recursive and iterative computations.	25
2.2.1 A general view of recursion.	25
2.2.2 Programming recursive computations.	27
2.2.3 Rules for recursive programming.	33
2.2.4 Iteration.	35
2.3 External files.	39

2.4	Searching.	43
2.4.1	Is searching a sign of ignorance?	43
2.4.2	Kinds of search.	43
2.5	Extra variables and functions, for efficiency.	50
2.6	Other list functions.	54
2.6.1	MEMBER and more.	54
2.6.2	Functions for association lists.	56
2.6.3	Example: translation between Inuit and English.	57
2.7	The fixed-point style of computation.	61
2.8	Input and output.	64
2.9	The procedural and functional styles of computation.	68
2.10	Additional reading	70
3	Deeper into Essential Structure	71
3.1	LISP's data.	71
3.1.1	S-expressions.	72
3.1.2	Representations of data.	74
3.1.3	Atoms.	76
3.1.4	Data types.	81
3.2	Another look at familiar functions	82
3.2.1	CAR, CDR, and CONS: an alternate semantics.	82
3.2.2	EQ and EQUAL.	83
3.2.3	Options for LAMBDA functions.	85
3.2.4	SETF.	87
3.3	LISP syntax in detail.	88
3.3.1	Terms.	89
3.3.2	Variables and forms.	89
3.3.3	Syntax puzzles.	91
3.4	Self-processing.	92
3.5	Bindings, scopes, and environments.	96
3.6	Additional reading.	103
4	Computational Philosophy	104
4.1	Models of computation.	104
4.1.1	Mathematical formalisms for computation.	105

4.1.2	Church's thesis and computational completeness.	107
4.2	Pure LISP.	110
4.3	Types of recursion.	114
4.3.1	Fat recursion and tail recursion.	114
4.3.2	Compound recursion.	115
4.3.3	Monotonic and nonmonotonic recursion.	116
4.4	The limits of LISP: an unsolvable problem.	118
4.5	The folklore of fixed-point computation.	122
4.6	Additional Reading.	125
5	LISP Functions for Powerful Programming.	126
5.1	Debugging tools.	126
5.1.1	The listener, access to local bindings, and BACKTRACE.	127
5.1.2	STEP and TRACE.	129
5.1.3	Other tools for debugging.	131
5.1.4	Application: debugging a LISP text (optional).	134
5.2	Applicative operators.	137
5.2.1	APPLY, FUNCALL, MAPCAR, and MAPLIST.	138
5.2.2	Application: symbolic differentiation.	141
5.3	Macros.	145
5.3.1	The basic idea.	145
5.3.2	Applications: INFIX and EVAL&TIME.	148
5.4	Structures, vectors, and arrays.	151
5.4.1	Vectors and arrays.	151
5.4.2	Structures.	153
5.4.3	Application: working with organic molecules.	156
5.5	Function closures.	163
5.5.1	Creating and using closures.	163
5.5.2	Application: COMPOSE and Currying CONS.	166
5.6	COERCion.	168
5.7	Surgical operations.	170
6	Interpreters: from Algebra to LISP	176
6.1	Algebras and interpreting: an abstract view.	176

6.1.1	Algebras of objects and algebras of terms.	177
6.1.2	Interpretation and computation.	179
6.2	LISP interpreters.	183
6.2.1	A sketch of the usual LISP interpreter.	184
6.2.2	A call-by-name interpreter for LISP.	190
6.2.3	Put the interpreter inside, keep the user out.	194
6.3	Compiled LISP.	198
7	Mathematical Foundations of LISP	200
7.1	The lambda calculus defined.	200
7.2	Straightforward computation in the lambda calculus.	205
7.3	Fixed-points for object functions.	208
7.4	Recursive functions in the lambda calculus.	210
7.5	Pure LISP in the lambda calculus.	213
7.6	Additional reading.	215
8	Automatic Reasoning, Algebraic Intelligence	216
8.1	Logics.	217
8.1.1	Propositional logic.	218
8.1.2	Fundamental theorems for the propositional calculus.	223
8.1.3	Other approaches to automatic reasoning in PC.	225
8.2	Predicate logic and unification.	229
8.3	PROLOG.	233
8.4	Speculation on algebraic intelligence.	236
8.5	Additional reading.	239
9	Bibliography	240
10	Answers to Selected Exercises	246
11	Index	269