

## CONTENTS

<i>Preface to the Second Edition</i>	xiii
<i>Preface</i>	xv
<b>1 Standard ML</b>	<b>1</b>
Functional Programming	2
1.1 Expressions versus commands	2
1.2 Expressions in procedural programming languages	3
1.3 Storage management	5
1.4 Elements of a functional language	5
1.5 The efficiency of functional programming	9
Standard ML	11
1.6 The evolution of Standard ML	11
1.7 The ML tradition of theorem proving	12
1.8 The new standard library	13
1.9 ML and the working programmer	15
<b>2 Names, Functions and Types</b>	<b>17</b>
Chapter outline	18
Value declarations	18
2.1 Naming constants	18
2.2 Declaring functions	19
2.3 Identifiers in Standard ML	21
Numbers, character strings and truth values	22
2.4 Arithmetic	22
2.5 Strings and characters	24
2.6 Truth values and conditional expressions	26
Pairs, tuples and records	27
2.7 Vectors: an example of pairing	28
2.8 Functions with multiple arguments and results	29
2.9 Records	32

vi *Contents*

2.10	Infix operators	36
	The evaluation of expressions	38
2.11	Evaluation in ML: call-by-value	39
2.12	Recursive functions under call-by-value	40
2.13	Call-by-need, or lazy evaluation	44
	Writing recursive functions	48
2.14	Raising to an integer power	48
2.15	Fibonacci numbers	49
2.16	Integer square roots	52
	Local declarations	53
2.17	Example: real square roots	54
2.18	Hiding declarations using <code>local</code>	55
2.19	Simultaneous declarations	56
	Introduction to modules	59
2.20	The complex numbers	59
2.21	Structures	60
2.22	Signatures	62
	Polymorphic type checking	63
2.23	Type inference	64
2.24	Polymorphic function declarations	65
	Summary of main points	67
<b>3</b>	<b>Lists</b>	<b>69</b>
	Chapter outline	69
	Introduction to lists	70
3.1	Building a list	70
3.2	Operating on a list	72
	Some fundamental list functions	74
3.3	Testing lists and taking them apart	74
3.4	List processing by numbers	76
3.5	Append and reverse	78
3.6	Lists of lists, lists of pairs	81
	Applications of lists	82
3.7	Making change	83
3.8	Binary arithmetic	85
3.9	Matrix transpose	87
3.10	Matrix multiplication	89
3.11	Gaussian elimination	90
3.12	Writing a number as the sum of two squares	93

<i>Contents</i>	vii
3.13 The problem of the next permutation	95
The equality test in polymorphic functions	96
3.14 Equality types	97
3.15 Polymorphic set operations	98
3.16 Association lists	101
3.17 Graph algorithms	102
Sorting: A case study	108
3.18 Random numbers	108
3.19 Insertion sort	109
3.20 Quick sort	110
3.21 Merge sort	111
Polynomial arithmetic	114
3.22 Representing abstract data	115
3.23 Representing polynomials	116
3.24 Polynomial addition and multiplication	117
3.25 The greatest common divisor	119
Summary of main points	121
<b>4 Trees and Concrete Data</b>	<b>123</b>
Chapter outline	123
The datatype declaration	124
4.1 The King and his subjects	124
4.2 Enumeration types	127
4.3 Polymorphic datatypes	128
4.4 Pattern-matching with <code>val</code> , <code>as</code> , <code>case</code>	130
Exceptions	134
4.5 Introduction to exceptions	134
4.6 Declaring exceptions	135
4.7 Raising exceptions	136
4.8 Handling exceptions	138
4.9 Objections to exceptions	140
Trees	141
4.10 A type for binary trees	142
4.11 Enumerating the contents of a tree	145
4.12 Building a tree from a list	146
4.13 A structure for binary trees	148
Tree-based data structures	148
4.14 Dictionaries	149
4.15 Functional and flexible arrays	154

viii *Contents*

4.16	Priority queues	159
	A tautology checker	164
4.17	Propositional Logic	164
4.18	Negation normal form	166
4.19	Conjunctive normal form	167
	Summary of main points	170
<b>5</b>	<b>Functions and Infinite Data</b>	<b>171</b>
	Chapter outline	171
	Functions as values	172
5.1	Anonymous functions with <i>fn</i> notation	172
5.2	Curried functions	173
5.3	Functions in data structures	176
5.4	Functions as arguments and results	177
	General-purpose functionals	179
5.5	Sections	179
5.6	Combinators	180
5.7	The list functionals <i>map</i> and <i>filter</i>	182
5.8	The list functionals <i>takewhile</i> and <i>dropwhile</i>	184
5.9	The list functionals <i>exists</i> and <i>all</i>	184
5.10	The list functionals <i>foldl</i> and <i>foldr</i>	185
5.11	More examples of recursive functionals	188
	Sequences, or infinite lists	191
5.12	A type of sequences	192
5.13	Elementary sequence processing	194
5.14	Elementary applications of sequences	197
5.15	Numerical computing	199
5.16	Interleaving and sequences of sequences	201
	Search strategies and infinite lists	204
5.17	Search strategies in ML	204
5.18	Generating palindromes	207
5.19	The Eight Queens problem	208
5.20	Iterative deepening	210
	Summary of main points	211
<b>6</b>	<b>Reasoning About Functional Programs</b>	<b>213</b>
	Chapter outline	213
	Some principles of mathematical proof	214
6.1	ML programs and mathematics	214

<i>Contents</i>	ix
6.2 Mathematical induction and complete induction	216
6.3 Simple examples of program verification	220
Structural induction	224
6.4 Structural induction on lists	225
6.5 Structural induction on trees	229
6.6 Function values and functionals	233
A general induction principle	237
6.7 Computing normal forms	238
6.8 Well-founded induction and recursion	242
6.9 Recursive program schemes	246
Specification and verification	248
6.10 An ordering predicate	249
6.11 Expressing rearrangement through multisets	251
6.12 The significance of verification	254
Summary of main points	256
<b>7 Abstract Types and Functors</b>	<b>257</b>
Chapter outline	258
Three representations of queues	258
7.1 Representing queues as lists	259
7.2 Representing queues as a new datatype	260
7.3 Representing queues as pairs of lists	261
Signatures and abstraction	263
7.4 The intended signature for queues	263
7.5 Signature constraints	264
7.6 The <code>abstype</code> declaration	266
7.7 Inferred signatures for structures	269
Functors	271
7.8 Testing the queue structures	272
7.9 Generic matrix arithmetic	275
7.10 Generic dictionaries and priority queues	280
Building large systems using modules	285
7.11 Functors with multiple arguments	285
7.12 Sharing constraints	290
7.13 Fully-functorial programming	294
7.14 The <code>open</code> declaration	299
7.15 Signatures and substructures	305
Reference guide to modules	308
7.16 The syntax of signatures and structures	309

x	<i>Contents</i>	
	7.17 The syntax of module declarations	311
	Summary of main points	312
<b>8</b>	<b>Imperative Programming in ML</b>	<b>313</b>
	Chapter outline	313
	Reference types	314
	8.1 References and their operations	314
	8.2 Control structures	317
	8.3 Polymorphic references	321
	References in data structures	326
	8.4 Sequences, or lazy lists	327
	8.5 Ring buffers	331
	8.6 Mutable and functional arrays	335
	Input and output	340
	8.7 String processing	340
	8.8 Text input/output	344
	8.9 Text processing examples	346
	8.10 A pretty printer	351
	Summary of main points	356
<b>9</b>	<b>Writing Interpreters for the <math>\lambda</math>-Calculus</b>	<b>357</b>
	Chapter outline	357
	A functional parser	357
	9.1 Scanning, or lexical analysis	358
	9.2 A toolkit for top-down parsing	360
	9.3 The ML code of the parser	363
	9.4 Example: parsing and displaying types	367
	Introducing the $\lambda$ -calculus	372
	9.5 $\lambda$ -terms and $\lambda$ -reductions	372
	9.6 Preventing variable capture in substitution	375
	Representing $\lambda$ -terms in ML	378
	9.7 The fundamental operations	378
	9.8 Parsing $\lambda$ -terms	381
	9.9 Displaying $\lambda$ -terms	382
	The $\lambda$ -calculus as a programming language	384
	9.10 Data structures in the $\lambda$ -calculus	385
	9.11 Recursive definitions in the $\lambda$ -calculus	388
	9.12 The evaluation of $\lambda$ -terms	389
	9.13 Demonstrating the evaluators	393

<i>Contents</i>	xi
Summary of main points	396
<b>10 A Tactical Theorem Prover</b>	<b>397</b>
Chapter outline	397
A sequent calculus for first-order logic	398
10.1 The sequent calculus for propositional logic	399
10.2 Proving theorems in the sequent calculus	400
10.3 Sequent rules for the quantifiers	403
10.4 Theorem proving with quantifiers	404
Processing terms and formulæ in ML	407
10.5 Representing terms and formulæ	407
10.6 Parsing and displaying formulæ	411
10.7 Unification	416
Tactics and the proof state	420
10.8 The proof state	420
10.9 The ML signature	421
10.10 Tactics for basic sequents	424
10.11 The propositional tactics	426
10.12 The quantifier tactics	428
Searching for proofs	430
10.13 Commands for transforming proof states	430
10.14 Two sample proofs using tactics	433
10.15 Tacticals	436
10.16 Automatic tactics for first-order logic	440
Summary of main points	444
<i>Project Suggestions</i>	445
<i>Bibliography</i>	449
<i>Syntax Charts</i>	457
<i>Index</i>	469