

Contents

	Part One: Introduction	<i>page</i> 1
1	Functional programming	2
	1.1 Equations	3
	1.2 Types	4
	1.3 Higher order	6
	1.4 Laziness	8
	1.5 Haskell	11
2	Structure in programs	15
	2.1 Structured programming	15
	2.2 Design patterns	17
	2.3 Recursion patterns	19
3	The mathematics of program construction	24
	3.1 Programming as mathematics	24
	3.2 Science and engineering	26
	3.3 Formal methods	27
	3.4 Mathematical ergonomics	28
	Part Two: Fundamentals	32
4	Algebraic structures	33
	4.1 Orderings	33
	4.2 Monoids	35
	4.3 Monoid homomorphisms	37
	4.4 The Boom Hierarchy	38

4.5	Groups, semigroups, semirings	39
5	Category theory	45
5.1	Categories	46
5.2	Concrete categories	48
5.3	Constructions on categories	50
6	Functors and natural transformations	53
6.1	Functors	53
6.2	Datatypes	55
6.3	Binary functors	56
6.4	Natural transformations	57
6.5	Homfunctors	61
6.6	The Yoneda Lemma	62
7	Products and coproducts	67
7.1	Product bifunctor	67
7.2	Product map	69
7.3	Products in other categories	70
7.4	Coproduct	71
7.5	Duality	72
7.6	Exchange law	73
7.7	Distributivity	74
8	Adjunctions	78
8.1	Universal properties	78
8.2	Galois connections	79
8.3	Adjoint functors	81
8.4	Adjoints of the diagonal functor	83
8.5	Currying	85
8.6	Free and forgetful	86
9	Extreme solutions	89
9.1	Extreme objects	89
9.2	Algebras and coalgebras	91
9.3	Limits and colimits	95
9.4	Extreme (co-)algebras as (co-)limits	96
9.5	(Co-)limits as adjunctions	99
10	Monads	102
10.1	Closures	102
10.2	Monads from adjunctions	103
10.3	Adjunctions from monads	105

10.4	Free monads	107
10.5	Monadic functional programming	109
10.6	Comonads	111
11	Monoidal categories	117
11.1	Monoids in a monoidal category	119
11.2	Symmetry and strength	120
11.3	Applicative functors	124
	Part Three: Datatypes	128
12	Lists	129
12.1	Folds on lists	129
12.2	Left fold	132
12.3	Minimax	134
12.4	Duality theorems	136
12.5	When is a function a fold?	138
12.6	Scans	139
13	List homomorphisms	146
13.1	The homomorphism theorems	148
13.2	The Trick	149
13.3	Maximum segment sum	152
14	The theory of lists	159
14.1	Longest segment problems	160
14.2	Suffix closure	163
14.3	Prefix closure	164
14.4	Overlap closure	165
14.5	Growing windows	167
15	Data and codata	172
15.1	Algebraic datatypes as initial algebras	172
15.2	Properties of folds	175
15.3	Codatatypes	178
15.4	Implementation in Haskell	181
16	Primitive recursion	187
16.1	Paramorphisms	187
16.2	Predecessors	190
16.3	Apomorphisms	191
16.4	Grafting	193

17	Continuous algebras	198
	17.1 Pointed complete partial orders	199
	17.2 Datatypes in pcpo	202
	17.3 Products and coproducts	204
18	Hylomorphisms	209
	18.1 Least fixed points	209
	18.2 Envelopes	211
	18.3 Deforestation	213
	18.4 Divide-and-conquer	215
19	Recursive coalgebras	222
	19.1 A simple example	222
	19.2 Characterizing recursive coalgebras	224
	19.3 Structurally recursive coalgebras	225
	19.4 Computationally recursive coalgebras	227
	19.5 Corecursive algebras	229
	Part Four: Effects	233
20	Monadic programming	234
	20.1 Tracing	235
	20.2 Exceptions	238
	20.3 Multifunctions	239
	20.4 Updatable state	240
	20.5 Reading and writing	241
	20.6 The monad interface	242
	20.7 Monad comprehensions	244
	20.8 System interaction	246
21	Combining monads	250
	21.1 The same monad family	250
	21.2 Composing different monads	251
	21.3 Monad transformers	253
	21.4 Composing adjunctions	255
	21.5 Distributive laws	256
22	Algebraic theories	261
	22.1 Programming to an interface	261
	22.2 A counter example	262
	22.3 Nondeterminism	264

<i>Contents</i>	xix
22.4 Failure	266
22.5 Exceptions	268
22.6 State	270
22.7 Combining algebraic theories	272
23 Idioms	278
23.1 Strong, lax monoidal functors	278
23.2 Effects idiomatically	280
23.3 Idiomatic parsing	282
23.4 Free idioms	286
24 Idiomatic traversals	293
24.1 The traversal interface	293
24.2 The laws of traversal	295
24.3 Unlabelling a tree	296
24.4 Representing traversable datatypes	299
24.5 The batch idiom	301
24.6 Proof of the Representation Theorem	303
25 Staging	308
25.1 Day convolution	308
25.2 Fusing traversals	310
25.3 The repmin problem	311
25.4 Multiple phases	315
25.5 Sorting a tree	318
25.6 Breadth-first traversal	319
26 Comonads	323
26.1 The comonad interface	323
26.2 Stream functions	327
26.3 Materialising stream functions	329
26.4 The laws of stream functions	332
26.5 Cellular automata	335
26.6 The zipper	337
Part Five: Generics	345
27 Datatype-generic programming	346
27.1 Representing types	346
27.2 A universe of codes	347
27.3 Datatype-generic equality	348

27.4	Implicit type codes	350
27.5	Representing functors	351
27.6	Representing bifunctors	354
28	Derivatives	360
28.1	Derivatives of functors	361
28.2	Derivatives of bifunctors	364
28.3	Derivatives of fixpoints	367
28.4	Derivatives and zippers	369
28.5	Dissection	371
28.6	Tail-recursive traversal	373
29	Monadic recursion	377
29.1	Monadic map	378
29.2	Monadic fold	381
29.3	Monadic unfold	384
29.4	Distributors	386
29.5	Comonadic fold	388
29.6	Recursion schemes from comonads	391
30	Metamorphisms	400
30.1	An unfold after a fold	401
30.2	Streaming	402
30.3	Flushing	404
30.4	Infinite input	406
30.5	Total streaming	408
30.6	Beyond lists?	409
31	Accumulations	413
31.1	Labelled variants	414
31.2	Upwards accumulations	416
31.3	Downwards accumulations	418
31.4	Parallel prefix	422
32	Adjoint folds	428
32.1	Mutumorphisms revisited	429
32.2	Conjugates of adjunctions	432
32.3	Adjunctions as a unifying framework	433
32.4	Adjoint folds from currying	435
32.5	Adjoint unfolds	438
32.6	Recursion schemes from comonads	440

Part Six: Applications	445
33 Program fission	446
33.1 Counting words	447
33.2 Directionality	449
33.3 Extracting length	450
33.4 A different starting point	452
33.5 Nested loops	454
34 Sorting	459
34.1 Sorting by structure	460
34.2 Sorting networks	461
34.3 Sorting in strides	463
34.4 Sorting by primitive recursion	465
34.5 Sorting with hylomorphisms	467
34.6 Sorting without comparisons	468
35 Enumerating the rationals	477
35.1 Euclid and greatest common divisors	478
35.2 The Stern–Brocot tree	479
35.3 The Calkin–Wilf tree	482
35.4 Iterating through the rationals	483
36 Primes	487
36.1 Calculating the sieve	487
36.2 The genuine sieve	490
36.3 Productivity of the sieve	492
36.4 A more direct proof	495
37 Continued fractions	500
37.1 Converting continued fractions	501
37.2 Rational functions	503
37.3 Unary functions	504
37.4 Binary functions	507
38 The digits of π	512
38.1 Leibniz’s series for π	512
38.2 Conversion from a fixed base	513
38.3 A spigot algorithm	517
38.4 Lambert’s formula for π	519
38.5 Gosper’s formula for π	521
39 Arithmetic coding	525
39.1 Intervals	526

39.2	Models	527
39.3	Encoding	528
39.4	Decoding	529
39.5	Producing bits	531
39.6	Streaming encoding	533
39.7	Streaming decoding	533
39.8	Fixed-precision arithmetic	537
40	Asymmetric numeral systems	541
40.1	Arithmetic coding, revisited	541
40.2	From fractions to integers	545
40.3	Bounded precision	547
40.4	Trading in sequences	550
40.5	Streaming encoding	552
40.6	Streaming decoding	554
40.7	Summing up	554
41	Maximum segment sum revisited	559
41.1	Initial segments, datatype-generically	560
41.2	Horner's Rule, datatype-generically	562
41.3	Distributivity reconsidered	564
41.4	Reducing distributivity	565
	References	571
	Index	585
	Envoi	586
A	Solutions	587
B	Haskell	588