

INDEX

- ! function, 314–318
- () constructor, 32
- * infix, 22, 23, 303
- + infix, 22, 23, 303
- infix, 22, 23, 303
- / infix, 23, 303
- :: constructor, 70–71, 77, 186
- := infix, 314–317
- < infix, 26
- <= infix, 27
- <> infix, 27, *see also* equality
- = infix, 27, 97, *see also* equality
- => keyword, 133, 138, 172, 463
- > infix, 27
- >= infix, 27
- @ infix, 78–80, 82, 186
 - eliminating, 80, 111, 146
 - for sequences, 195
 - proofs about, 226–229
- ^ infix, 25, 125
- ~ function, 23
- ~ function, 22, 23

- AAMP5 microprocessor, 256
- Aasa, Annika, 339
- abs function, 24
- abstract types, 97, 115, 263–269
 - examples of, 281–283, 327–334
 - for proof systems, 421
 - how to declare, 268
- abstraction over variables
 - in λ -calculus, 372–374, 377, 379
 - in logic, 407–409
- abstype declarations, 266–269, 281, 284, 460
 - repeated, 293
- Adams, Stephen, 154
- ALF system, 13
- all function, 184–185, 187

- amortized cost, 262
- andalso keyword, 27, 43, 462
- Andrews, Peter, 446
- app function, 320
- Appel, Andrew, 10, 15, 102, 351, 371
- append, *see* @
- applicative programming, *see* functional programming
- ARITH signature, 62–63, 86, 116
- arithmetic in ML, 14, 22–24, 137, 465
 - unlimited precision, 445–446
- Array structure, 335
- arrays
 - flexible, 154–159, 263, 300
 - functional, 336–339
 - mutable, 154, 335–336
- as keyword, 132, 156, 463–464
- assignment commands, *see* :=
- assignments, in logic, 398
- association lists, 101–102, 150, 287–288, 336
- atan function, 24
- AUTOMATH system, 395

- backtracking, *see* search, depth-first
- Backus, John, 6n, 9
- Beckert, B., 443
- before infix, 320
- Bevier, William R., 256
- Biagioni, Edoardo, 10
- binary arithmetic, 85–87
- Bind exception, 137, 138
- BinIO structure, 350
- Boehm, Hans, 446
- Bool structure, 340
- bool type, 26, 127
- boolean values, 26–27
 - in λ -calculus, 385
- Boyer/Moore theorem prover, *see* NQTHM

470 *Index*

- Braun, W., 155
 Bruijn, N. G. de, 395
 Burge, W. H., 371
- C, 15–16, 274
 call-by-name, 44–45, 194, 200n
 in λ -calculus, 375, 388–392, 395
 call-by-need, 8, 9, 45–48, 140, *see also*
 sequences
 call-by-value, 39–40, 43, 44, 136, 140
 in λ -calculus, 375, 389–393
 CAML, 12, 136
 Cardelli, Luca, 67
 Cartwright, Robert, 446
 case expressions, 133, 137–140, 173, 462
ceil function, 24
Char structure, 26, 341–342
char type, 25–26
Chr exception, 137
chr function, 25–26, 137
 Church, Alonzo, 47, 372
 Church-Rosser Theorem, 374
 Cohn, Avra, 256
 combinators, 180–182
 comments, 20, 467
 complex numbers, 59–61
 composition, of functions, *see o*
 computer algebra, 114
concat function, 74, 82
 for lists, 81, 187, 190
 for sequences, 437
 concatenation
 of lists, *see @*
 in λ -calculus, 388, 395
 of sequences, 195
 of strings, 25
 conditional expressions, 26–27, 43–44, 462
 and exceptions, 137, 140
 in λ -calculus, 385, 391
 type checking of, 64
 conjunctive normal form, 167–170, 240–242
 cons, *see ::* constructor
 Constable, Robert, 443
 constructive type theory, 13, 443
 constructors, 125, 130–132
 for lists, 70
 hiding, 159, 265–269
 control structures, 317–321
cos function, 24
- Cousineau, Guy, 12
- Damas, Luis, 67
 datatype bindings, 267, 461
 datatype declarations, 124–130, 460
 recursive, 142, 165, 192, 194, 233
 repeated, 128, 293
 with one constructor, 159, 261
 datatype specifications, 310
Date structure, 15
 Davis-Putnam procedure, 170, 446
 declarations, 18–22, 53–56, 460, *see also*
 each kind of declaration
 in a structure, 60
 of modules, 311–312, 457–458
 simultaneous, 56–58
 declarative programming, 10
depth function, 143, 189, 232
 dereferencing, *see !*
Dictionary functor, 281–283
DICTIONARY signature, 149–150, 266, 281,
 288
 Dijkstra, Edsger, 82, 93, 94, 159
 disjoint sum type, 129–130
 disjunctive normal form, 170
distrib function, *see* conjunctive normal
 form
Div exception, 137
div infix, 22, 49, 64
Domain exception, 137
 domain theory, 215, 216, 233, 247, 443
drop function, 78, 82, 111, 188, 424
 proofs about, 251–254
dropwhile function, 184
- efficiency, 9–10, 47
 of recursion, 42, 76–80
 Eight Queens problem, 208–211
Empty exception, 138
 environments, 21, 39, 62, 378, 393, 418
 eqtype specifications, 266, 269, 287–288,
 310, 459
EQUAL constructor, 127, 281, 289
 equality, 96–107
 and abstract types, 97, 267, 268
 and functions, 97, 234–236
 of references, 316, 332–334
 Euclid's Algorithm, *see* Greatest Common
 Divisor

- evaluation, 38–48, 136
 lazy, *see* call-by-need
 strict, *see* call-by-value
- exception declarations, 135–136, 304, 460
- exception specifications, 310, 459
- exceptions, 134–141, 462
 and commands, 320–321
 eliminating, 151
 type checking of, 325
- exists* function, 184–185, 187
- exn* type, 135–139, 141, 177, 321
- exp* function, 24
- explode* function, 73
- expressions, 462–463
 in programming languages, 2–4
- fact* function, 40–42, 245
- facti* function, 40–42, 47
 proofs about, 214, 220–222, 245, 247
 type checking of, 64
- factorials, 40–43, 189, 317–319, *see also*
fact, *facti*
 in λ -calculus, 388–389, 393–395
- Fail* exception, 138
- false* constructor, 26, 127
- Fibonacci numbers, 49–51, 191, 222–223, 329–330
- filter* function, 182–183, 187, 209
 for sequences, 196, 206
- Fitzgerald, J. S., 256
- fixed point property, 389, 392
- FixedInt* structure, 14
- floor* function, 24
- fn* expressions, 172–174, 178, 323, 427–428, 462
 and delayed evaluation, 193, 202, 391
- foldl* function, 185–187
 proofs about, 236–237
- foldr* function, 185–187, 190, 211, 409
 proofs about, 237
- formulae, 398
 in ML, 408
- Fortran, 2, 7, 9, 127, 356
- FP, 9
- from* function, 193, 199
- Frost, R., 371
- fully-functorial programming, 294–299
- fun declarations, 19–20, 28–31, 125–127, 460
 of curried functions, 174, 182–183
 polymorphic, 325
- functional languages, 9
- functional programming, 1–11, 38, 58
 and imperative features, 327–330, 336–339
 applications of, 10
- functionals, 7–8, 179–190, 409, 426–428,
see also tacticals, tactics
 and parsing, 362–366
 proofs about, 233–237
- functions, 6
 as arguments, 177–178, 280
 as data, 176–177, 191–192
 curried, 173–178, 183–185, 209
 declaring, *see* fun declarations
 higher-order, *see* functionals
 iterative, 42–44, 49, 51, 76–78, 151, 186, 247
 recursive, 6, 40–44, 48–53, 175, 317
 with multiple arguments/results, 29–32, 82, 110
- functor declarations, 272, 275–277, 285–289, 312, 458
- functors, 271–299, 309, *see also*
 fully-functorial programming
- Gansner, Emden, 13
- garbage collection, 5–6, 130, 313
- Gaussian elimination, 90–92
- General* structure, 15
- Gerhart, Susan, 256
- Gordon, Michael J. C., 12, 440, 443
- Grant, P. W., 92
- graphs, 102–107, 278–280
- GREATER* constructor, 127
- Greatest Common Divisor, 3, 10, 48, 53, 248
 for polynomials, 120–121
- Greiner, John, 326
- Hal, 397, 407–443
- Halfant, Matthew, 200
- Hamming problem, 330
- handle* keyword, 138, 462
- HARP system, 443
- Harper, Robert, 326
- Haskell, 9, 10, 92, 102, 131n

472 *Index*

- hd* function, 74, 82, 138, 176
 - for sequences, 192, 327
- heaps, *see* priority queues
 - binomial, 164
- Hoare, C. A. R., 10, 15, 69, 110, 111
- HOL system, 443
- Holmström, Sören, 336
- Hoogerwoord, R., 159
- HOPE, 12
- HTML, 348–350
- Hudak, Paul, 9
- Huet, Gérard, 12, 421
- Hughes, John, 200, 336

- identifiers, 21–22, 61, 465–467
- i f* expressions, *see* conditional expressions
- ignore* function, 319
- imperative programming, 2–5, 79, 108
 - in ML, 313–340, 344–356
- ImperativeIO* functor, 350
- implode* function, 73
- include* specifications, 307–308, 310, 459
- induction
 - on natural numbers, 216–224, 244–245
 - on size, 238–245
 - structural, 224–233, 245
 - well-founded, 238, 242–247
- infix* declarations, 460
- infix* operators, 36–38, 283, 303, 363, 462
 - parsing, 364–366, 412–414
- input/output, 8, 340–356
- instances
 - of polymorphic types, 65, 176
 - of signatures, 264
 - of terms/formulae, 379, 416–420
- Int* structure, 24, 340
- int* type, 22–24
- inter* function, 98, 183
- interleave* function, 195, 202
- IntInf* structure, 14
- io* exception, 344
- Isabelle system, 246, 421, 440, 443
- it* value, 19, 50, 174
- iterates* function, 196, 198, 331

- keywords, of Standard ML, 21

- Lakatos, Imre, 256
- λ -calculus, 182, 372–396
- LAMBDA system, 13
- Landin, Peter, 12
- Launchbury, J., 371
- Lazy ML, 9
- LCF system, 11–13, 421, 440, 443
- leanTAP system, 443
- left-recursive rules, 361, 381, 412
- length* function, 76–77, 82, 229
- Leroy, Xavier, 136
- LESS* constructor, 127
- let* expressions, 53–55, 135–137, 300–301, 318
 - and polymorphism, 324
- Letz, R., 443
- lexical analysis, 358–360, 368, 412
- library, ix, 13–15, 127, 319
 - arithmetic and, 24, 303
 - arrays and, 335
 - characters and, 26
 - functionals and, 187
 - input/output and, 350
 - lists and, 82, 335
 - strings and, 26
- Lisp, 7, 9, 75, 234
- List* structure, 82, 138
- list* type, 70, 144
- ListPair* structure, 82, 187
- lists, 6, 69–122, 141, 336
 - doubly linked, *see* ring buffers
 - functionals for, 182–188, 195
 - in λ -calculus, 387–388
 - in other languages, 71
 - induction on, 225
 - lazy, *see* sequences
- ln* function, 24
- local declarations, 55–56, 300–301, 457, 460
- logic
 - first-order, 398–407
 - notation, 215
 - propositional, 164–170, 399–400
- LOLITA, 11

- Ménessier-Morain, Valérie, 446
- MacQueen, David, 12, 299
- Magnusson, Lena, 13
- making change, 83–84, 139, 197–198
- map* function, 182–183, 187–188, 190, 209
 - for pairs of lists, 187

- for sequences, 195
- x sequences, 199
- proofs about, 235–236
- Match* exception, 73, 137
- Math* structure, 24, 137
- matrix operations, 87–93, 183, 275–280
- Mauny, Michel, 136
- max* function, 24
- maxl* function, 72–73, 238
- mem* function, 97–99, 185
- merging, 111, 117–118
- meta-variables, *see* variables, in unification
- Miller, Keith, 108
- Miller, Steven, 256
- Milner, Robin, 11–12, 66, 421
- min* function, 24
- Miranda, 9
- ML, 1
 - and verification, 214–215
 - as meta-language, 12–13, 430–431
 - compilers for, xii–xiii
 - evolution of, ix, 11–12, 28
 - Standard, 10–16
- ML-Yacc, 371
- mod* infix, 22, 49
- modules, 12, 16, 59–63, 257–312, *see also*
 - functors, signatures, structures
 - reference guide to, 308–312
- multisets, 251–254, 399

- names, *see* identifiers
- natural numbers, 216–219, 224
 - in λ -calculus, 386–387, 393–395
- negation normal form, 166–167, 238–240
- newmem* function, 98, 187
- Newton-Raphson method, *see* square roots, real
- nil* constructor, 70–71
- Nipkow, Tobias, 371
- nlength* function, 76, 226–227, 229, 233, 238
- NONE* constructor, 128
- nonfix* declarations, 38, 460
- Nordström, Bengt, 13
- normal forms
 - in λ -calculus, 374–375, 392
- not* function, 27, 127
- NQTHM system, 246
- nrev* function, 79–80, 227–233, 237
- nth* function, 138, 424

- null* function, 74, 82
 - for sequences, 327
- Nuprl system, 443

- o* infix, 180–181, 187, 234–237, 370
- O’Keefe, Richard, 112–113
- occurs check, 416–417
- Odersky, M., 102
- Okasaki, Chris, 159, 164
- op* keyword, 37–38, 176–177, 180–181, 186, 465
- open declarations, 299–305, 363, 460
- Oppacher, F., 443
- Oppen, Derek, 354
- option* type, 128
- ord* function, 25–26
- ORDER* signature, 280–284, 286
- order* type, 127, 281
- ordered binary decision diagrams, 170, 446
- ordered* predicate, 249
- orElse* keyword, 27, 43, 462
- OS* structure, 145
- Otter system, 443
- Overflow* exception, 137
- overloading, 23–24, 27, 64, 67, 102

- pairs and tuples, 27–38, 136
 - in λ -calculus, 386
- palindromes, 207–208, 211
- parameters, in logic, 406–407, 416–417, 428–433, 435
- Park, Stephen, 108
- parsing, 360–372, 381–382, 411–414
 - LR, 371
- Pascal, 4, 7, 108, 175
 - pointers in, 316, 321, 332
- patterns, 130–133, 182–183, 463–464
 - and *fn* notation, 172
 - for datatypes, 125–127
 - for lists, 72–73
 - for pairs and tuples, 28–32
 - for records, 34–35, 338
 - for trees, 143
 - in *val* declarations, 31–32, 131–132, 138
 - layered, *see* *as* keyword
 - non-exhaustive, 73–75, 83–85, 137
 - overlapping, 126, 168
 - wildcard, 74, 126, 131
- Paulson, Lawrence C., 246, 440, 443

474 *Index*

- PDP-8, 446
- Pelletier, F., 441
- permutations, 95–96, 251
- Peyton Jones, Simon L., 10
- polymorphism, *see* types
- polynomials, 114–121, 446
- Posegga, J., 443
- powers, 48–49, 223–224
- powerset, 99–100
- pretty printing, 351–356, 369–371, 382–384, 414
- Pretty* structure, 355
- prime numbers, 94, 199, 219
- PrimIO* functor, 351
- print* function, 345, 348
- priority queues, 159–164, 281, 283–284, 290–293
- PriorityQueue* functor, 284, 296
- product types, 27–38
- products, Cartesian, 100, 187, 203
- Prolog, 71, 417, 443
- proof, 399–407
 - automated, 440–443
 - constructive, 219
 - states, 420–424
- prop* type, 165
- Quaife, A., 443
- quantifiers, 215–216, 403–407, 428–430
 - and induction, 221–224, 227–228, 245
- queues, 206, 258–274
 - mutable, 334
 - priority, *see* priority queues
- raise* keyword, 136, 462
- random numbers, 108–109, 198–199
- Reade, Chris, 154
- real* function, 24
- REAL* signature, 303
- Real* structure, 24, 340
- real* type, 22–24, 303
- Real32* structure, 14
- Real64* structure, 14
- rec* keyword, 460
- records, 32–35, 338
 - selecting fields of, 34–35, 464, 467
- recursion, *see* datatype declarations; functions
 - in λ -calculus, 388–395
 - infinite, 363
 - mutual, 57
 - well-founded, 245–246
- ref* constructor, 314, 325
- ref* type, 314, 321
- references, 314–334
 - and polymorphism, 321–326
 - cyclic, 316, 329, 331
- referential transparency, 3–4
- reflect* function, 144, 189, 230–233
- repeat* function
 - for parsing, 362–363, 382
 - for powers of a function, 188–189, 202
 - tactical, 436–437, 439–440
- Reppy, John, 13
- rev* function, 79–80, 82
- revAppend* function, 80, 227–228
- reversing a list, 79–80, 186, 324–325. *see also nrev, rev, revAppend*
- ring buffers, 331–334
- scanning, *see* lexical analysis
- Scheme, 9
- Scott, Dana, 13, 375n
- search, 204–211
 - best-first, 160, 206
 - breadth-first, 104, 204–208, 210–211
 - depth-first, 103–105, 204–210, 437–443. *see also* making change
 - iterative deepening, 210–211, 439, 441
- sections, 179–181
- Sedgewick, Robert, 105, 108
- semirings, 280
- sequences, 191–211, 366–367, 423
 - and numerical analysis, 199–201
 - infinite, 8, 195, 226
 - in λ -calculus, 388, 389, 395
 - using references, 327–330
- sequents, 398–407, 428
 - basic, 399, 423–424
- set operations, 98–100, 187
- SETHEO system, 443
- sharing
 - constraints, 290–294, 306–310, 459
 - of references, 323
- side effects, 3, 323
- sign* function, 24
- signature constraints, 264–266, 269, 275, 299, 311–312

- signature declarations, 62–63, 311, 457
- signatures, 62–63, 263–271, 285, 309–310, 458
 - closed, 297–299
 - empty, 288
- sin* function, 24
- Sisal, 9
- Size* exception, 138
- size* function, 25
 - for trees, 143, 189, 232
- Skolem functions, 417
- Sleator, Daniel, 262
- Smith, M. H., 11
- software development, 15–16, 213, 285
- SOME* constructor, 128
- sorting, 108–114, 160–162, 177–178
 - topological, 105–107
 - verification of, 249–254
- space, 5, 42, 130
- specifications
 - executable, 10, 229
 - in signatures, 264, 286, 309–310, 459, *see also* sharing constraints
 - of programs, 220, 248–249, 251, 255–256
- sqrt* function, 24
- square roots
 - integer, 52–53
 - real, 54–55, 199–201, *see also sqrt* function
- Srivias, Mandayam, 256
- static binding, 21
- Stickel, Mark, 170, 446
- str* function, 26
- StreamIO* functor, 350
- streams, input/output, 344–347
- String* structure, 26, 341–342
- string* type, 26
- StringCvt* structure, 341
- strings, 24–26, 73–74, 184, 340–343, 465–466
- structure declarations, 60–63, 311, 457
- structure specifications, 283–284, 310, 459
- structures, 60–62, 308, 311, 458
 - empty, 288
 - in logic, 398
- Substring* structure, 341
- subs* infix, 99, 115
- Subscript* exception, 82, 137
- substitution
 - in λ -calculus, 373–379
 - in logic, 403–404, 408
- Substring* structure, 26, 348
- substrings, 26, 341–342, 348–350
- Suen, E., 443
- sum of two squares, 93–94
- Sussman, Gerald, 200
- tacticals, 13, 436–440
- tactics, 13, 420–435, 440–443
- take* function, 77, 82, 111, 424
 - for sequences, 193, 328
 - proofs about, 251–254
- takewhile* function, 184
- Tarditi, David, 371
- Tarjan, Robert, 262
- tautology checking, 164–170, 238–242, 354
- terms
 - in logic, 398, 408
 - of λ -calculus, 372–373, 378–379
- theorem proving, 11–13, 397–443, 446
- Time* structure, 15
- Timer* structure, 15
- tl* function, 75, 82, 138
 - for sequences, 192, 327
- Tofte, Mads, 12, 325
- TREE* signature, 295, 297
- Tree* structure, 148, 297, 304–305
- tree* type, 142, 148, 304
- trees, 6, 141–164, 189
 - balanced, 143–147, 150–154
 - binary search, 150–154, 176–177, 281–283
 - induction on, 229
 - traversing, 145–147, 189, 231–233
- true* constructor, 26, 127
- trunc* function, 24
- Turner, David A., 9, 47, 100, 182
- type abbreviations, *see* type declarations
 - in signatures, 307
- type constraints, 23–24, 29–30, 35, 324, 334, 462
- type constructors, 368
- type declarations, 29, 35, 460
- type specifications, 269, 309, 459
- type variables, 65–67, 368, 466
 - equality, 97–99
- types, 63–67, 199, 464

476 *Index*

- dynamic, 136, 177
- equality, 97, 102
- parsing and displaying, 367–371
- polymorphic, 128–130, 321–326
- restrictiveness of, 71, 124, 325

- unification, 405–407, 416–420
 - efficient, 420
 - higher-order, 421
 - tactic for, 423–426, 435
- union* function, 98, 115
- unit* type, 32, 129–130, 192, 321
- universe, 398
- unzip* function, 81–82
- Uribe, T., 170
- user interfaces, 431

- v*-arrays, 336–339
- val* declarations, 18–19, 31–32, 172–174, 428, 460
 - polymorphic, 178, 323–324
- val* specifications, 309, 459
- validity, 398–399
- variables, 21
 - in λ -calculus, 372–373, 375–377, 379
 - in logic, 398, 408
 - in unification, 405–407, 416–417, 428–430, 434–435
 - type, *see* type variables
- Vector* structure, 335
- vectors, 27–32, 37, 89
- verification, 13, 213–256
 - limitations of, 223, 254–256

- Wegner, Peter, 67
- well-founded relations, 242–244, 246
- where type qualification, 310
- while expressions, 318, 462
- withtype keyword, 460
- Word8* structure, 14
- Wright, Andrew, 326

- Y* combinator, 389, 391–392

- Zhang, Hantao, 170
- zip* function, 81, 82