

---

# Index

- ! (cut) operation (in Prolog), 662, 666, 669
- ! (escaped variable marker), 500, 509
- !! (read-only) operation, 206, 799
- " (double quote), 53, 821
- \$ (nesting marker), 53, 83, 355, 365
- ' (single quote), 35, 52, 821, 824
- ' (single quote) operation (in Lisp), 39
- \* (multiplication) operation, 54, 821
- \*/ (comment end), 841
- + (addition) operation, 54, 821
- (subtraction) operation, 54, 821
- . (field selector) operation, 54, 826
- . := (dictionary/array assignment) statement, 436, 437, 838
- . := (dictionary/array exchange) expression, 838
- / (floating division) operation, 54, 821
- /\* (comment start), 841
- :: (membership) constraint, 751
- := (state assignment) statement, 494, 497
- := (state exchange) expression, 497
- = (binding) operation, 44, 45, 47, 99
- =: (equality) constraint, 751
- =<: (less or equal) constraint, 753
- == (equality) comparison, 55
- =< (less or equal) comparison, 55
- ? (output argument), 57, 841
- @ (state access) operation, 494, 497
- # (tupling) constructor, 140, 826, 831
- % (comment to end of line), 841
- % (modulo) operation (in Java), 617
- & (inline character) operator, 820
- < (strictly less) comparison, 55, 830
- <= (optional method argument) operator, 500
- > (strictly greater) comparison, 55
- >= (greater or equal) comparison, 55
- \ (backslash), 821
- \ (backslash) notation (in Haskell), 311
- \= (inequality) comparison, 55
- \=: (inequality) constraint, 756
- \~ (tilde) minus sign, 820
- ` (backquote), 50, 821, 841
- ' (backquote) operation (in Lisp), 39
- | (list pairing) constructor, 52
- Abelson, Harold, 42
- absolute error, 120
- abstract data type (ADT), 195–210, 419
- abstract machine, 41, 56–78, 92–94, 239–241, 282–283, 348–349, 416–417
  - relation to semantic rules, 786
  - substitution-based, 126–127
- abstract syntax tree, 163
- abstraction, *xviii*, *see* control abstraction, *see* data abstraction
- active object, 556
- class, 492
- collection, 435
- collector, 189, 325, 435, 484
- connector, 326
- database, 654
- encapsulated search, 625
- hierarchy in object-oriented programming, 546
- iteration, 122

- life cycle, 40
- Linda (tuple space), 586
- linguistic, *see* linguistic abstraction
- list comprehension, 301
- lock, 582, 598
- loop, 181, 258
- mailbox, 391
- monitor, 592
- pipe of stream objects, 561
- port object, 350
- procedural, 178
- protector, 325
- replicator, 326
- software component, 220
- specialization hierarchy, xviii
- stream object, 265
- termination detection, 382
- transaction, 600
- tuple space, 586
- access (cell operation), 17, 414
- ACCLAIM project, xxvi
- accumulator, 138–141
  - breadth-first traversal, 156
  - declarative state, 408
  - depth-first traversal, 155
  - for** loop, 189
  - limit of declarative model, 315
  - loop abstraction, 185, 258
  - parser, 161
  - Prolog, 140
  - relation to difference lists, 142
  - relation to fold operation, 185
  - tree traversal, 192
- ACID properties, 600
- action (in GUI), 683
- active memory, 74
  - size, 172, 314
- ActiveX, 462
- actor model, 345
- adder
  - n*-bit, 341
  - full, 268
- address
  - IP, 207
  - URL, 211
- adjacency list, 464
- Adjoin operation, 161, 437, 438, 533, 550, 695, 826
- AdjoinAt operation, 378, 437, 438, 826
- adjunction (environment), 62
- ADT (abstract data type), 420
- advanced exercise, xxiv
- adversary, 208
- agent, 350
  - concurrent component, 362
  - message-passing approach, 576
- Alarm operation, 304, 393
- alarm clock, xv
- algebra, xxiii, 113
- algorithm
  - breadth-first traversal, 156
  - compression, 177
  - concurrent composition, 276
  - copying dual-space garbage collection, 78
  - Dekker’s algorithm for mutual exclusion, 570
  - depth-first traversal, 155
  - distributed, *see* distributed algorithm
  - elevator, 376
  - Flavius Josephus problem, 558
  - Floyd-Warshall, 470
  - garbage collection, 76
  - Hamming problem, 293
  - linear congruential generator, 474
  - mergesort, 137, 166
  - mergesort (generic), 181
  - Newton’s method for square roots, 119
  - nonalgorithmic programming, 622
  - parallel transitive closure, 469
  - Pascal’s triangle, 10
  - queue, 145
  - persistent, 297

- priority, 614
- quicksort, 232, 524
- random number generation, 472
- simulation, 476
- termination detection, 382
- thread scheduling, 240
- transaction manager, 605
- transitive closure, 464
- tree, 151
- tree-drawing, 158, 273
- unification, 101
- word frequency, 198
- alias (in QTK), 680
- aliasing, 418, 449
- allocation, 75
- always true (in temporal logic), 603
- Anderson, Ross J., 744, 843
- Andrews, Gregory, 582
- andthen** operator, 83
- Append operation, 829
- apple, 489
- Apple Corporation, xxvi
- applet, 718
- application
  - FlexClock example, 698
  - ping-pong example, 305
  - standalone, 222
    - Java, 555
  - video display, 321
- applicative order reduction, 330
- arbitrary precision arithmetic, 3
- arbitrary precision integer arithmetic, 821
- arch, xviii
- arithmetic, 54
- Arity** operation, 54, 826
- arity, 826
  - case** semantics, 67
  - method head, 499
  - Prolog, 661
  - record memory consumption, 174
  - record performance, 438
- Armstrong, Joe, 582
- Arnold, Ken, 537
- array, 436
  - extensible, 440
  - usage trade-offs, 438
- ASCII (American Standard Code for Information Interchange), 219, 458, 459, 715
- Ask** operation, 768
- ask operation, 782, 790
- assert/1** operation (in Prolog), 656, 662
- assertion, 444
- assignment
  - axiomatic semantics, 446
  - cell operation, 17, 414
  - in-place, 314
  - Java, 552
  - monotonic, 849
  - multiple, 849
  - single, 42, 849
  - to iteration variable, 189
- association, 531
- association list, 486
- asynchronous communication, 332
  - active object, 558
  - component interaction, 456
  - distributed object, 726
  - distributed port, 713
  - distributed semantics, 383
  - Erlang, 386
  - error reporting, 359
  - failure detection, 739
  - GUI, 685
  - message passing, 345
  - port, 347
  - receive, 332
  - relation with dataflow, 726
  - RMI, 356
  - send, 332, 719
  - slow network, 578
  - trade-off with fault detection, 745
- ATM (Asynchronous Transfer Mode), 387
- atom, 52, 824

- defining scope, 508
- predicate calculus, 633
- propositional logic, 632
- atomic action, 579–615
  - approaches to concurrency, 573
  - reasoning, 581
  - when to use, 576
- atomicity, 21, 600
- `AtomToString` operation, 824
- attribute
  - final (in Java), 544, 551
  - initialization, 498
  - object, 497
- availability, 711
- AXD 301 ATM switch, 387
- axiom
  - axiomatic semantics, 441
  - Horn clause (Prolog), 635
  - logic programming, 406, 634
  - predicate calculus, 633
- axiomatic semantics, 38, 440–450, 632
- Backus, John, 32
- backward chaining, 662
- `bagof/3` operation (in Prolog), 626, 670
- Bal, Henri E., 334
- basic constraint, 759
- batching
  - distributed stream, 719, 724
  - object invocations, 540, 565
- Baum, L. Frank, 1
- Bernstein, Philip A., 600
- billion dollar budget, 521
- binary integer, 819
- binding
  - basic and nonbasic, 788
  - dynamic, 506
  - static, 506
  - variable, 787
  - variable-variable, 47
- binomial theorem, 4
- bistable circuit (digital logic), 270
- blank space (in Oz), 841
- Blasband, Darius, 642
- block
  - file, 292
  - imperative language, 486
  - memory, 74, 76
  - Smalltalk, 543
- blocking operation, 239
- blue cut (in Prolog), 666, 668
- book
  - Component Software: Beyond Object-Oriented Programming*, 462
  - Concurrent Programming in Erlang*, 582
  - Concurrent Programming in Java*, 581
  - Concurrent Programming: Principles and Practice*, 582
  - Object-Oriented Software Construction*, 491
  - Software Fundamentals*, 462
  - Structure and Interpretation of Computer Programs*, xix
  - The Art of Prolog*, 666
  - The Mythical Man-Month*, 462
  - Transaction Processing: Concepts and Techniques*, 582
- boolean, 52
- Borges, Jorge Luis, 621, 707
- bottom-up software development, 451
- bound identifier occurrence, 64
- bounded buffer, 263
  - data-driven concurrent version, 263
  - lazy version, 290
  - monitor version, 593
- brand, 498
- Brand, Per, xxvi
- breadth-first traversal, 156
- `break` (in `for` loop), 190
- `break` statement, 486
- bridge building, xvi, xvii
- Brinch Hansen, Per, 592

- Brooks, Jr., Frederick P., 462  
Browser tool, *see* Mozart Programming System  
bundling, 420  
by-need execution, 281–284, 795  
    Multilisp, 337  
    Oz 3, 809  
    WaitNeeded operation, 795  
    WaitQuiet operation, 798  
ByNeed operation, 282  
    definition using WaitNeeded, 795  
byte (in Java), 553  
bytestring, 831  
  
calculator, 1  
calculus  
    analysis, 170  
    first-order predicate, 633  
    foundational, xvii  
     $\lambda$ , xvii, 41, 97, 331, 344, 792, 805, 811, 846  
     $\pi$ , xvii, 41, 54, 805  
calendar widget, 691–693  
call/1 operation (in Prolog), 661, 662  
call by . . ., *see* parameter passing  
call graph, 318  
capability, 208  
    declarative, 210  
    method label, 510  
    revocable, 434, 483  
Cardelli, Luca, 722  
cascading abort, 604  
**case** statement, 67, 790  
**catch** clause (in **try**), 94  
causality  
    concurrency, 238  
    message-passing events, 353  
**Ceil** operation, 822  
cell (explicit state), 409, 414–417, 794, 848  
cellular phone, xv, xviii  
chaining  
    backward, 662  
    forward, 674  
channel  
    asynchronous, 348, 385  
    component interface, 456  
    dataflow variable, 332  
    many-shot, 363  
    one-shot, 363, 368  
    port, 347  
    synchronous, 619  
character, 820  
    alphanumeric, 50, 841  
    Java, 553  
China, 707  
**choice** statement, 623, 772  
choice point, 623, 767  
    binary, 654, 762  
    constraint programming, 755  
    dynamic, 654  
    implementation, 772  
    memory leak, 668  
Choose operation, 654, 767  
chunk, 828  
    use in object reflection, 517  
Church-Rosser theorem, 331  
Churchill, Winston, 450  
Clarke, Arthur C.  
    second law, 104  
    third law, 314  
class, 413, 491, 496, 546  
    abstract, 523, 546  
    active object, 558  
    common limitations, 539  
    complete definition, 492  
    concrete, 523, 525, 546  
    delegation, 511  
    diagram, 528, 534  
    encapsulation control, 506  
    event manager example, 563  
    final, 492, 497, 543  
    forwarding, 511  
    generic, 524  
    higher-order programming, 525  
    implementation, 548  
    incremental definition, 502

- inheritance, 502
- inner (in Java), 540, 552
- introduction, 18
- member, 497
- metaclass, 547
- mixin, 566
- parameterized, 524
- patching, 521
- programming techniques, 518
- reflection of object state, 517
- structure view, 519
- substitution property, 518, 521, 523
- type view, 519
- clause
  - case** statement, 82
  - definite, 650
  - Erlang, 388
  - Horn, xxi, 635, 640, 650, 663
- clock
  - context-sensitive, 698
  - digital circuit, 271
  - digital logic, 267
  - synchronization, 308
- clone
  - array, 436
  - computation space, 769
  - dictionary, 437
  - object, 517
- clone** operation, 769
- closed distribution, 732
- closure, *see* procedure value
- cluster, 711
- code generator, 162
- coherence
  - between graphical views, 696
  - cache, 460, 733
  - network transparency, 720
- collector, 189, 325, 435, 484
- COM (Component Object Model), 462
- combinational logic, 267
- combinations, 4
- combinator, 278
- comics, 482
- comment (in Oz), 841
- Commit** operation, 769
- common self, 511
- compaction, 76
- comparison, 55
- compilation
  - separate, 105
  - standalone, 228
  - ping-pong example, 305
  - unit, 221, 412, 454, 816
- compiler, 162, 661
  - extensible, 542
  - Prolog, 664
  - smart-aleck, 314
  - unification operation, 99
- Compiler Panel tool, *see* Mozart Programming System
- compiler window (in OPI), 815
- complete list, 53, 829
- complete value, 46
- completeness
  - software development, 219
  - theorem prover, 634
  - Turing, 846
- complexity
  - amortized, 146, 174, 437
  - asymptotic, 167
  - banker's method, 175
  - big-oh notation, 167
  - introduction, 10–11
  - physicist's method, 175
  - space, 172
  - time, 11, 166
  - worst-case, 167, 172
- component, 112, 183, 220
  - abstraction, 458
  - avoiding dependencies, 459
  - diagram, 351
  - encapsulates a design decision, 458
  - functor, 412
  - further reading, 462
  - future role, 461

- graph, 461
  - implementation, 221
  - interface, 221
  - module, 412
  - software, 221
  - compositionality, 411, 620
    - class, 501
    - encapsulated search, 626
    - exception handling, 90
    - `Solve` operation, 626
  - compound
    - ADT, 435
    - component, 363
    - data structure, 19, 52
    - graphic figure, 535
    - nonterminal, 645
    - statement, 117
    - statement (in Java), 552
    - value, 43
  - computation, 61
    - blocked, 241
    - iterative, 118
    - recursive, 124
    - secure, 208
    - terminated, 241
  - computation model, xiii, 29
    - active object*, 556
    - constraint-based*, 627, 641, 758
    - data-driven concurrent*, 235
    - declarative concurrent with exceptions*, 326
    - declarative concurrent*, 235
    - declarative with exceptions*, 92
    - declarative*, 49
    - demand-driven concurrent with exceptions*, 328
    - demand-driven concurrent*, 235, 281
    - descriptive declarative*, 115
    - distributed*, 712
      - graph notation, 735
      - extended relational*, 660
      - general*, 782, 843
      - job-based concurrent*, 618
  - lazy concurrent*, 235, 281
  - maximally concurrent*, 577, 618, 766
  - constraint programming, 766
  - message-passing concurrent*, 347
  - nondeterministic concurrent*, 394, 641
  - object-based*, 538
  - order-determining concurrent*, 273
  - relational*, 623
  - secure declarative*, 203
  - shared-state concurrent*, 573
  - stateful concurrent*, 569, 575, 576, 807
  - stateful*, 413
  - strict functional*, 97
  - atomic action, 579
  - closed, 326
  - concurrent constraint, 808
  - concurrent logic programming, 287, 394, 811
  - design principles, xix, 843
  - determinate concurrent constraint
    - programming, 338
  - determinate parallel programming, 337
  - deterministic logic programming, 635
  - Erlang, 386
  - Haskell, 309
  - insecure, 325
  - Java, 551, 615
  - nondeterministic logic programming, 638
  - open, 326
  - partial failure, 326, 739
  - Prolog, 662
  - secure, 325
  - semantics, 779
  - using models together, xx, 324
- computation space, 654, 753, 761
  - blocked, 767
  - clone, 769
  - distributable, 768

- failed, 768
- merged, 768
- runnable, 767
- stable, 755, 766, 768
- succeeded, 768
- suspended, 768
- computer engineering, xxi
- computer science, xxi
- computing, *see* informatics
- conc** statement, 278
- concurrency, 322
  - competitive, 254
  - cooperative, 254
  - dataflow, 15
  - declarative, 233, 242, 804
  - difference with parallelism, 322
  - Erlang, 386
  - further reading, 581
  - importance for programming, xvi
  - interactive interface, 89
  - interleaving, 21
  - introduction, 14
  - Java, 615
  - monitor safety, 593
  - nondeterminism, 20
  - order-determining, 273
  - practical approaches, 573
  - queue, 583
  - rules of thumb, 576
  - teaching, xxii
  - transaction, 602
- concurrency control, 567, 602, 746
- concurrency-oriented programming (COP), xvi, xxii, 573
- concurrent composition, 276, 385
- condition variable, 599
- conditional critical section, 595
- configuration, 784
- confinement
  - failure in declarative model, 245
  - partial failure, 709
  - transaction, 601
- confluence, 331, 806
- connector, 326
- cons cell (list pair), 5, 52, 828
- consistency (in transaction), 600
- consistency protocol, 712
- constraint, 244
  - absolute distance, 774
  - basic, 759
  - equality, 751
  - finite domain, 760
  - inequality, 756
  - less or equal, 753
  - membership, 751
  - pairwise distinctness, 756
  - rational tree, 760
  - scalar product, 775
  - tree-drawing, 158
- constraint programming, 44, 274, 663
  - computation space, 627
  - partial information, 751
  - threads, 254, 577
- constructor, 554
- consumer, 257
- content edge, 733
- context, 91
- context-sensitive
  - grammar, 33
  - GUI design, 698
- continuation, 378
  - procedure, 359
  - record, 358
- continue** (in **for** loop), 190
- contract, 410, 520
- contrapositive law, 632
- control abstraction
  - break, 486
  - concurrent composition, 278
  - CSP communication, 619
  - Erlang mailbox, 402
  - higher-order programming, 177
  - iteration, 122
  - loops, 184
  - need for higher-order, 180
  - Smalltalk block, 543
  - try-finally**, 343
- convolution (symbolic), 232

- coordination model, 456, 586
- CORBA (Common Object Request Broker Architecture), 356, 462
- coroutine, 274, 456
  - relation to laziness, 285, 287, 574
- correctness, 410, 520, 632
  - introduction, 9–10
- Cray-1 computer, 175
- critical region, 582
- cryptarithmetic, 755, 776
- cryptography
  - unique name generation, 207
- CSP (Communicating Sequential Processes), 619
- curriculum (informatics), xxii
- currying, xxi, 194, 232
  - Haskell, 311
- cyclic structures, 102
- dangling reference, 65, 75, 180, 459, 552
- Danvy, Olivier, 232
- Darwin, Charles, 451
- data abstraction, 18, 419–435
  - abstract data type (ADT), 195–210
  - concurrent, 578
  - object, 420
  - object-oriented programming (OOP), 489
  - procedural (PDA), 420
- data structure
  - active, 77
  - class, 496
  - compound, 19, 52
  - cyclic, 101
  - dictionary, 196
  - difference list, 141
  - difference structure, 141
  - ephemeral, 146
  - external, 77
  - graph, 464
  - higher order, 183
- infinite, 11
- list, 52, 129
- long-lived, 78
- partial, 46
- persistent, 297
- protected, 203, 421
- queue, 145
- record, 52
- recursive, 130
- size, 173
- stack, 195
- store, 75
- tree, 150
- tuple, 52
- data type, *see* type
- data-driven execution, *see* eager execution
- database, 654
  - deductive, 655
  - distributed, 615
  - in-memory, 173
  - Mnesia (in Erlang), 387
  - persistence, 654
  - query, 655
  - relational, 655
  - shared-state concurrency, 576
  - transaction, 600
  - update, 655
- dataflow, 60
  - channel, 332
- declarative model, 16
- error, 89
- examples, 248
- I-structure, 337
- introduction, 15
- lazy execution, 285
- parallel transitive closure, 469
- rubber band, 251
- variable, 42, 47, 48
- Date, C.J., 655
- Dawkins, Richard, 410
- DBMS (Database Management System), 655

- DCOM (Distributed Component Object Model), 462
- deadlock, 605
  - avoidance, 605
  - callback, 357
  - concurrency control, 605
  - detection, 605
  - digital logic simulation, 270
  - lazy execution, 287
  - prevention, 605
  - resolution, 605
- deallocation, 75
- debugging, 516
  - dangling reference, 75, 180
  - declarative component, 313
  - distributed application, 745
  - erroneous suspension, 48, 90
  - inspecting object attributes, 501
- declarative, 111, 406
- declarative concurrency, *see* concurrency, declarative
- declarative program, 244
- declare** statement, 2, 87
  - syntax convention of book, xxix, 87
- define** clause (in functor), 221
- Definite Clause Grammar (DCG), 140, 650
- Delay operation, 304
- delay gate (digital logic), 270
- delay operation (in Multilisp), 337
- delay point, 580
- delegation, 512–515
  - otherwise method, 501
- Dell Corporation, xxvi, 201, 471
- demand-driven execution, *see* lazy execution
- De Morgan’s law, 632
- denotational semantics, 38
- dependency
  - component, 224
  - dataflow, 335
  - declarative component, 313, 323
  - grammar context, 33
- inheritance, 413
- order-determining concurrency, 273
- removal of sequential, 385
- sequential model, 411
- word frequency application, 225
- depth-first traversal, 155
- dereferencing, 45
- design by contract, 521
- design methodology
  - compositional, 461
  - concurrent program, 364
  - language, 40, 329, 543, 811, 850
  - large program, 450
  - noncompositional, 461
  - small program, 218
- design patterns, xxi, 413, 489, 534–537
  - Composite pattern, 535
  - declarative concurrency, 365
- destructive assignment, *see* state
- determinism (declarative programming), 111
- DHCP (Dynamic Host Connection Protocol), 207
- diagnostics, 621
- dialog model (in GUI), 695
- dictionary, 196, 437
  - declarative, 196
  - efficiency, 201
  - internal structure, 201
  - list-based implementation, 197
  - relation implementation, 659
  - relation to record, 438
  - relation to stream, 440
  - secure declarative, 207
  - standalone, 225
  - stateful, 198
  - tree-based implementation, 198
  - tuple space implementation, 588
  - usage trade-offs, 439
  - word frequency implementation, 471
- word-of-mouth simulation, 477

- difference list, 141  
difference structure, 141  
digital logic, 266  
    satisfiability problem, 176  
Dijkstra, Edsger Wybe, 441  
directed graph, 463  
disconnected operation, 741  
discriminant, 179  
dissentailment, 103–104, 781  
distributed algorithm  
    cached state, 733  
    garbage collection, 734  
    locking, 721  
    mobile state, 733  
    token passing, 721  
    unification, 733  
distributed lexical scoping, 722  
distributed system, 345, 707  
    closed, 714, 732  
    open, 508, 709, 711, 714  
    partial failure, 326  
    port semantics, 384  
    resource, 729  
Distribution Panel tool, *see* Mozart Programming System  
distribution strategy, 755, 761, 766  
**div** (integer division) operation, 54, 821  
divide-and-conquer, 137  
domain (in constraint programming), 760  
domain model (in GUI), 695  
domain of discourse, 633  
dotted pair, 5, 828  
Duchier, Denys, xxvi, 382  
Dudeney, Henry Ernest, 774  
durability, 600  
dynamic  
    binding, 506  
    linking, 222, 284  
    record creation, 165, 549, 695  
    scope, *see* scope, dynamic  
    typing, 104–106  
Dynamic HTML, 682  
eager execution  
    declarative model, 98  
    distributed producer/consumer, 719  
    introduction, 11  
    producer/consumer stream, 257  
    relation to synchronization, 334  
    strictness, 331  
eager failure detection, 739  
EBNF (Extended Backus-Naur Form), 32, 650, 651, 671  
Eco, Umberto, 90  
Einstein, Albert, 278  
Elcock, E. W., 621  
elephant, xiii  
elevator, *see* lift  
Emacs text editor, 815  
embedding, 183  
emulator window (in OPI), 815  
encapsulation, 18, 411  
    communication from inside encapsulated search, 673  
data abstraction, 420  
search, 625  
Encyclopaedia Britannica (11th edition), 489  
Ende, Michael, 345  
entailment, 99, 103–104, 781  
Enterprise Java Beans, 461  
entry point (in block), 486  
environment, 44, 61  
    adjunction, 62  
    calculating with, 62  
    contextual, 65  
    interactive interface, 87  
    restriction, 62  
ephemeral data structure, 146, 297  
equality  
    structure, 103, 418, 723  
    token, 418, 714, 723  
Ericsson (Telefonaktiebolaget LM Ericsson), 326, 386  
Erlang, 386–394  
error, 90

- absolute, 120
- “arity mismatch”, 500
- asynchronous reporting, 359
- compile-time detection, 105
- concurrent programming, 21
- confinement, 90
- dangling reference, 75
- domain, 96
- erroneous suspension, 48, 89, 449
- if** statement, 66
- incompatible binding, 44, 96
- infinite loop, 449
- memory leak, 75
- memory reclaiming, 75
- Mozart run-time, 815
- procedure application, 67
- relative, 120
- run-time detection, 105, 503
- starvation, 275
- type incorrect, 50, 96, 449, 450
- “uncaught exception”, 93, 801
- unification failure, 96
- variable declaration, xxix
- “variable not introduced”, 834
- “variable used before binding”, 48
- error logging, 563
- Escher, Maurits Cornelis, 679
- Euclid, 1
- event (in GUI), 682
- event handler, 563
- event manager
  - active objects, 563
  - adding functionality with inheritance, 564
- eventually true (in temporal logic), 603
- evolution
  - Darwinian, 451, 462
  - Internet, 412
  - software development, 451
- example programs (how to run), xxix
- exception, 90–96, 543, 848
  - distribution fault, 743
  - error, 96, 802
  - failure, 96, 801
  - system, 96
  - uncaught, 93, 801
- Exchange operation, 416
- exchange operation
  - on cells, 416
  - on object attributes, 497
- exclusive-or operation, 14
- execution state, 61
- exit point (in block), 486
- explicit state, *see* state
- Explorer tool, *see* Mozart Programming System
- exponential distribution, 475
- export** clause (in functor), 221
- expression, 81
  - andthen** operator, 83
  - basic operation, 54
  - basic value, 49
  - ill-formed, 92
  - $\lambda$  notation, 97, 311
  - nesting marker (\$), 83
  - normal order reduction, 330
  - orelse** operator, 83
  - procedure value, 65
  - receive** (Erlang), 391
- external resource, 77
- extreme programming, 452
- factorial function, 2–23, 124–127, 230, 618
- fail** statement, 623
- FailedValue operation, 328
- failure, 410
  - declarative concurrency, 245
  - detection, 709, 739
  - exception, 96
  - failed value, 328
  - finite, 662
  - heartbeat mechanism, 739
  - negation as failure, 662
  - partial, 326
  - software project, 521, 643

- transaction, 601
- unification, 102
- fairness
  - stream merger, 398
  - thread scheduling, 239, 240, 252, 333
- false**, 824
- fault, 601
  - confinement, 202, 601, 709, 745
- fault model, 739
- fault tolerance, 601
  - active, 743
  - Erlang, 387
    - lift control system, 377
  - FCFS (first-come, first-served), 366
  - FD.distance (propagator), 774
  - FD.distinct (propagator), 756, 774
  - FD.distribute (distribution strategy), 754, 766
  - FD.sumC (propagator), 775
- feature (record field name), 825
- feed a text (in OPI), 815
- FGCS (Fifth Generation Computer System), Japanese project, 397
- Fibonacci function, 249
- FIFO (first-in, first-out), 145, 232, 337, 366, 379, 456, 719, 724
  - Erlang, 386
  - port, 347
- file, 210
- Filter operation, 191, 829
- finalization, 78, 481
  - lazy execution, 481
- finally** clause (in **try**), 94
- finite domain, 760
- finite failure, 662
- finite state machine
  - event manager, 563
- finite tree, 150
- firewall, 326
- first-argument indexing, 656
- fixpoint, 766
- Flatten operation, 143, 232, 296
- Flavius Josephus problem, 558–561
- FlexClock utility, 698
- flip-flop, 270
- floating point
  - arithmetic, 821
  - number, 820
- FloatToInt operation, 822
- Floor operation, 822
- flow control, 98, 261
  - lazy execution, 261
  - thread priority, 265
- Floyd, Robert W., 441, 621
- fold operation, 185
  - FoldL**, 187, 465, 829
  - FoldR**, 187
  - stream objects, 258
- for** statement, 188
- for** loop, 447
- ForAll operation, 829
- formal language, 33
- forward chaining, 674
- forwarding, 512
- foundational calculus, xvii
- framework
  - computation model, 844
  - software reuse, 492
- France, 411
- free identifier occurrence, 58, 64, 633
- free memory, 75
- freeze/2** operation (in Prolog), 662
- French, Michael, 569
- fresh
  - name, 203
  - variable, 787
- full adder, 268
- fun** statement, 84
- function, 84
  - generating, 170
  - incremental, 296
  - introduction, 2
  - lazy, 797
  - monolithic, 296
  - monotonic, 849
  - partially applied, 194

- state transition, 351
- functional decomposition, 135, 542
- functional look, 196
- functor, 221, 454
  - distributed resource, 709, 730
  - interactive use, 816
  - main, 222
  - Oz 3, 809
- functor** statement, 223
- future, 336
- future** operation (in Multilisp), 336
- Gamma, Erich, 534
- garbage collection, 74, 75
  - copying dual-space, 78
  - distributed, 734
  - external references, 480
  - finalization, 481
  - generational, 78
  - lease-based, 734, 738
  - program pause, 76, 308
  - real-time, 76
  - root set, 76
  - weighted reference counting, 734
- Gardner, Martin, 774
- gate** statement, 272
- gate (digital logic), 267
- Gaussian distribution, 476, 477
- Gelernter, David, 586
- generalization, 568
- generate-and-test, 629, 758
- generating function, 170
- genericity, 180–182
  - object-oriented programming, 524
  - static and dynamic, 526
- global condition
  - ordered binary tree, 154
  - producer/consumer throughput, 261
  - reachability, 75
- GlobalStore, 743
- glue (in GUI), 687
- Glynn, Kevin, 308
- Gödel’s completeness theorem, 634
- Gödel’s incompleteness theorem, 634
- Gödel, Kurt, 634
- Goodman, Nathan, 600
- Gosling, James, 537
- grammar, 31–36
  - ambiguous, 34, 166
  - context-free, 33
  - context-sensitive, 33
  - definite clause (DCG), 140, 650
  - disambiguation, 34
  - EBNF (Extended Backus-Naur Form), 32, 671
  - left-recursive, 644
  - natural language, 643
  - nonterminal symbol, 32, 163
  - terminal symbol, 32, 163
  - unification, 649–650
- graph
  - bushy, 461
  - component, 461
  - distribution model, 735
  - Haskell expression, 310
  - hierarchical, 461
  - implementation, 464
  - inheritance, 502
  - nonlocal, 461
  - path, 463, 658
  - small world, 461
- Gray, Jim, 582, 600
- green cut (in Prolog), 669
- Gregorian calendar, 693
- Grolaux, Donatien, 689, 691
- grue cut (in Prolog), 669
- guard
  - ask operation, 782
  - concurrent logic programming, 397
  - CSP, 619
  - Erlang, 388
  - Haskell, 309
  - list comprehension, 302
  - monitor, 595
  - Prolog, 669
  - quiet (in Prolog), 669

- receive** expression (in Erlang), 391, 402
- guarded method, 595
- guardian, 481
- GUI (Graphical User Interface)
  - AWT (Abstract Window Toolkit)
    - package, 679
  - component-based programming, 461
  - design, 679
  - Haskell fudgets, 679
  - model-based, 695
  - Prototyper tool, 689
  - QTk, 213, 680, 729
    - interactive use, 214, 684
    - use in application, 225
  - read-only view, 695
  - Swing components, xxi, 679
  - text input/output, 213
  - Visual Studio, 679
- Hadzilacos, Vassos, 600
- halting problem, 209, 681
- Hamming problem, 293, 342
- Hamming, Richard, 293
- handler
  - event, 563
  - exception, 90, 91
  - finalization, 481
  - GUI design, 213, 683, 682–688
- HasFeature** operation, 826
- hash table, 438, 439
- Haskell, 308–313
- Helm, Richard, 534
- Herbert, Frank, 749
- heuristic
  - best first, 672
  - distribution strategy, 755, 766
- Hewitt, Carl, 345
- hexadecimal integer, 820
- higher-order programming, 113, 177–194
- introduction, 13
- iteration abstraction, 123
- library operations, 830
- relation to OOP, 538
- history
  - ADT language, 420
  - computer technology, 176
  - concurrent constraint programming, 663, 808
  - dangers of concurrency, 21
  - declarative concurrency, 337
  - functional programming, 96
  - incremental development, 451
  - object-oriented programming, 489
  - Oz, xxiv, 809
  - programming languages, 406
  - Prolog, 661
  - remote invocation, 709
- Hoare, Charles Antony Richard, 232, 441, 592
- Horn clause, xxi, 635, 640, 650, 663
- hot spot, 177
- HTML (Hypertext Markup Language), 115, 679
- Hudak, Paul, 96, 314
- IBM Corporation, 41
- IDE (Interactive Development Environment), xxix
- IDE (interactive development environment), 815
- identifier, 2, 44
  - bound occurrence, 64
  - escaped, 509
  - free occurrence, 58, 64, 633
- IEEE floating point standard, 544, 820
- if** statement, 66, 790
- IID (Iterative and Incremental) software development, 451
- ILOG Solver, 663
- impedance matching, 324
  - concurrency, 576
  - event manager example, 567
- imperative, 406
- implementation, 410

- import** clause (in functor), 221
- inactive memory, 75
- incompleteness, 634
- inconsistency, 787
- incremental function, 295
- incremental software development, 451
- independence
  - components, 363, 455
  - compositional design, 461
  - concepts, xvii
  - concurrency, 14, 20, 89, 233
  - declarative loop, 189
  - declarative programming, 111
  - Erlang processes, 386
  - garbage collection, 76
  - message passing, 387
  - modularity, 319
  - multiple inheritance, 533
  - open programming, 105, 326, 714
  - polymorphism, 429
  - principle, 457
  - put and get (in queue), 381
  - reasoning with invariant assertions, 441
  - search and distribution strategies, 761
  - software deployment, 221
- India, xiii
- infinite precision arithmetic, 3
- infinity (floating point), 822
- informatics, xxi
  - curriculum, xxii
  - success of object-oriented programming, 489
  - usefulness of computation models, xiv
- Information Cities project, 412
- information systems, xxi
- inheritance, 19, 413, 420, 491
  - avoiding method conflicts, 511
  - cautionary tale, 521
  - design patterns, 534
  - directed and acyclic, 503
- event manager, 564
- factoring, 492
- generalization, 568
- graph, 502
- implementation, 550
- implementation-sharing problem, 533
- Java, 551
- multiple, 503, 527
  - rules of thumb, 533
- simple, 503
- single, 503, 531
- software reuse, 492
- static and dynamic binding, 504
- structure view, 519
- thread priority, 253
- type view, 518
- upward, 568
- inherited argument, 161
- Inject** operation, 772
- instantiation, 182–183, 411
- integer
  - arbitrary precision, 3, 821
  - arithmetic, 821
  - number, 819
  - tilde ~ as minus sign, 820
- interactive interface, 87
- interactive system
  - GUI design, 213
  - Mozart IDE, 815
  - pedagogical use, xix
  - reaction time guarantees, 174
- interface, 419
  - general concept, 221
  - inheritance, 492
  - Java, 551
  - Runnable** (in Java), 616
- interface builder, 679
- interleaving semantics, 237, 780
- Internet, 207, 345, 353, 707, 711, 717
  - simulation of Web use, 476
- interoperability, 106
- interpreter, 42
  - approach to define semantics, 41

- generic parser, 650  
GUI description, 703  
meta, 654, 676  
metacircular, 42  
original Erlang, 388  
Prolog arithmetic, 664  
QTk module, 703  
intractable problems, 176  
`IntToFloat` operation, 822  
invariant, 134, 382, 412, 441  
IP (Internet Protocol), 206  
`IsAtom` operation, 824  
`IsChar` operation, 823  
`IsDet` operation, 319, 321, 333, 394, 660, 803, 849  
`IsFailed` operation, 803  
`IsLock` operation, 583  
ISO 8859-1 character code, 820, 824  
isolation, 600  
`IsProcedure` function, 55  
`IsRecord` operation, 826  
I-structure, 336, 469, 811  
`IsTuple` operation, 827  
iterative software development, 451  
Janson, Sverker, xxvi, 663  
Japan, 397  
Java, 551–556, 615–617  
    monitor semantics, 592  
JavaBeans, 462  
Jefferson, Thomas, 9  
Johnson, Ralph, 534  
journaling, 532  
Kahn, Gilles, 337  
kernel language, *see* computation model  
kernel language approach, xvi, 36–42  
    choice of formalism, xix  
keywords (table of), 839  
Knuth, Donald Ervin, 170, 472  
Kowalski, Robert A., 406  
Kurzweil, Raymond, 176  
KWIC (keyword in context) index, 666  
`Label` operation, 54, 826  
label (record identification), 19, 52  
 $\lambda$  calculus, xvii, 41, 97, 331, 344, 792, 805, 811, 846  
LAN (local area network), 353, 717, 739  
language  
    *AKL*, xxvi, 661, 809  
    *Absys*, 406, 621  
    *Ada*, 432, 619  
    *Algol*, 406, 432, 489  
        declarative concurrent, 337  
        nondeterministic, 622  
    *Alice*, 106  
    *C++*, 43, 48, 75, 180, 334, 445, 486, 489, 504, 508–510, 535, 540, 545, 551, 663, 761  
    *C-Linda*, 586  
    *CLOS (Common Lisp Object System)*, 516  
    *CLU*, 420  
    *CSP (Communicating Sequential Processes)*, 619  
    *Clean*, 313  
    *Cobol*, 544  
    *Common Lisp*, 59, 190  
    *Concurrent Haskell*, xx  
    *Concurrent ML*, xx, 851  
    *Concurrent Prolog*, 397  
    *C*, 75, 179, 661  
    *Eiffel*, 519, 521, 850  
    *Erlang*, 75, 98, 326, 345, 386–394, 429, 456, 545, 563, 582, 807, 851  
    *E*, 208  
    *FCP (Flat Concurrent Prolog)*, 397, 807  
    *FP*, 329  
    *Flat GHC*, 397  
    *Fortran*, 406, 642  
        parsing problem, 642  
    *GHC (Guarded Horn Clauses)*, 397  
    *Haskell*, xiv, xvi, 43, 75, 98, 116,

- 137, 194, 272, 279, 286, 308–313, 329, 331, 334, 337, 342, 457, 545, 679, 807
- IC-Prolog*, 397
- Id*, 337, 811
- Java*, xiv, xvi, 41, 43, 48, 75, 180, 334, 356, 428, 430, 445, 462, 486, 489, 504, 508–510, 535, 540, 543–545, 551–556, 567, 581, 592, 615–617, 679, 807, 850
  - monitor, 592
- Leda*, xvii
- Linda* extension, 586, 619, 808
- Lisp*, xiv, 5, 39, 59, 75, 76, 129, 406, 544, 650, 664, 828
- ML*, *see Standard ML, Concurrent ML, Objective Caml*
- Mercury*, xiv, 116, 313, 663, 807
- Miranda*, 279, 342
- Multilisp*, 336, 811
- Objective Caml*, 543
- Obliq*, 722
- Oz 1, Oz 2, Oz 3*, 809
- Oz*, xix, xxvi, 313, 507, 545, 663, 807, 844, 851
- PL/I*, 642
  - parsing problem, 642
- Parlog*, 397
- Pascal*, 161, 179, 430, 807
- Prolog*, xiv, xvi, xxi, 5, 29, 48, 75, 116, 140, 142, 272, 287, 329, 334, 388, 397, 406, 545, 621–623, 635, 640, 642, 649, 654, 656–660, 660–671, 807, 851
  - pure, 640
  - SICStus, 190, 663
- Scheme*, xiv, xvi, xvii, xx, 29, 43, 59, 97, 98, 286, 545, 807
- Simula*, 406, 489
- Smalltalk*, xiv, 43, 75, 334, 489, 507, 509, 510, 516, 540, 543, 544, 850
- Standard ML*, xiv, xx, 29, 43, 97, 98, 116, 137, 194, 286, 313, 330, 807
- Visual Basic*, 461
- pH (parallel Haskell)*, 337, 811
- tcl/tk*, 679, 680, 703
- assembly, 209, 313, 406, 551, 622
- coordination, 586
- first-order, 177
- formal, 33
- higher-order, 177
- multiparadigm, xvii
- natural, xiii, 31, 38, 641
- nonstrict, 331
- popular computation models, 807
- practical, 31
- secure, 208
- specification, 116
- symbolic, 53, 545
- language design
  - abstraction life cycle, 40
  - declarative, 329
  - golden age, 406
  - layered, 850
  - lazy execution, 329
  - object properties, 543
  - trade-offs, 811
- Lao-tzu, 278
- last call optimization, 72
- latch (digital logic), 270
- late error detection (at run time), 503
- latency, 263
  - tolerance, 335
- LaTeX 2 $\varepsilon$  typesetting system, 459
- Latin-1, 458
- law
  - Clarke's second, 104
  - Clarke's third, 314
  - contrapositive, 632
  - De Morgan's, 632
  - Moore's, 176, 622
  - stack ADT, 195
- layered language design, 850
- lazy evaluation, 98
  - coroutining, 574

- explicit, 183
- Haskell, 310
- relation to call by need, 433, 485
- relation to nonstrict evaluation, 331
- schedule, 343
- strictness analysis, 289, 310, 342
- lazy execution, 278
  - bounded buffer, 263
  - flow control, 261
  - Hamming problem, 293
  - higher-order programming, 193
  - incremental, 295
  - introduction, 11
  - monolithic, 296, 342
  - needs finalization, 481
  - relation to synchronization, 334
- lazy failure detection, 739
- Lea, Doug, 581
- legal program, 31
- Length operation, 829
- lex/yacc parsing, 642
- lexical analyzer, 32
- lexical scope, *see* scope, lexical
- lexical syntax (of Oz), 839
- lexically scoped closure, *see* procedure value
- lexicographic order (of atoms), 55, 824
- Ley, Willy, 621
- library, 229
  - MOGUL (Mozart Global User Library), 222
  - Mozart Base and System modules, 229
  - Mozart Standard Library, 214, 225, 685, 690
  - universal, 621
- life cycle
  - abstraction, 40
  - memory block, 75
- LIFO (last-in, first-out), 491
- lift control system, 365
- lifting
  - booleans to streams, 271
  - serializability, 600
  - synchronization, 358
- lightweight transaction, 601
- Linda (tuple space), 586
- linguistic abstraction, 38–39, 124
  - case** (pattern matching), 790
  - class**, 548
  - conc** (concurrent composition), 278
  - delegation, 514
  - for** loop, 188, 447
  - fun** (function), 84
  - functor** (software component), 223
  - gate** (logic gate), 271
- local vs. global translation, 844
- macro (in Lisp), 39
- monitor, 593
- parameter passing, 434
- protected scope (Java), 567
- while** loop, 447
- linking, 211, 222, 224, 229, 455, 817
  - component, 223, 459
  - dynamic, 284
  - failure detection in Erlang, 387
- Linux operating system, xxvi, 201, 471, 499
- Liskov, Barbara, 420
- list, 52, 128, 828
  - circular, 829
  - complete, 53, 829
  - difference, 141
    - advantage, 145
  - flattening, 143
  - incomplete, 440
  - introduction, 4
  - nested, 135
  - partial, 440, 829
  - usage trade-offs, 439
- list comprehension, 301
- list pair, 52, 828
- literal, 824
- liveness, 602

- local** statement, 56, 63, 786
- lock, 579, 582–583
  - distributed, 721
  - get-release, 598
  - implementation, 590
  - introduction, 21
  - Java, 616
  - read, 620
  - simple, 591
  - thread-reentrant, 591
  - transaction, 602
  - write, 620
- lock** statement, 22, 583
- locus of control, 274
- logic
  - combinational, 267
  - gate, 267
  - predicate calculus, 633
  - propositional, 632
  - sequential, 269
  - temporal, 603
- logic program, 634
- logic programming, 44, 406, 632
  - difference list, 142
  - process model, 395, 807
  - unification, 101
- logical equivalence, 243, 785
  - configuration, 805
- logical formula, 633
- logical semantics, 38, 631–641
- logical sentence, 633
  - assertion, 441
  - invariant, 441
- Louis XIV, 405, 410
- Loyd, Sam, 774
- Lully, Raymond (Llull, Ramón), 621
- Lynch, Nancy, 353
- Mac OS X operating system, xxiv, xxvi, 254
- Macintosh computer, xxvi
- MacQueen, David, 337
- macro
  - alias (in `QTK`), 680
- Lisp, 39, 544
  - loop (in Common Lisp), 190
- Maher, Michael, 662, 808
- mailbox, 456
  - Erlang, 386
  - implementation, 391
- maintainability, 458
  - inheritance, 492
  - polymorphism, 425
- `MakeRecord` operation, 165, 549, 695, 826
- `MakeTuple` operation, 373, 827
- Manchester Mark I, 36
- Manna, Zohar, 441
- many-to-one communication, 351
- `Map` operation, 190, 466, 829
- Mariner I (software error), 643
- marshaling, 709
- master-slave, 732
- mathematical induction, 9
- matrix
  - graph representation, 464
  - list of lists implementation, 232
- `Max` operation, 194
- McCloud, Scott, 482
- `Member` operation, 829
- memoization, 417, 457
  - calendar example, 694
  - call by need, 434
  - declarative programming, 315
  - explicit state, 25, 694
  - unification, 102
- memory
  - address in abstract machine, 56
  - consumption, 172
  - content-addressable, 587
  - leak, 75
  - choice point, 668
  - Prolog, 668
  - life cycle, 75
- memory management, 72–78, 480–482
- C, 180
- garbage collection, 75

- Pascal, 180  
Merge operation, 771  
message, 499  
message-passing concurrency, *see object*, active, *see object, port*  
meta-interpreter, 654, 676  
meta-object protocol, *see protocol, meta-object*  
method  
    object, 19, 497  
    wrapping, 516  
methodology, *see software development*  
Meyer, Bertrand, 450, 491, 521, 527  
Microsoft Corporation, 462, 679  
middle-out software development, 451  
mind of programmer  
    capabilities (atoms vs. names), 510  
    difference list, 145  
    enforcing encapsulation, 420  
    language design trade-offs, 811  
    order-determining concurrency, 273  
    state (implicit vs. explicit), 408  
    use of constraints, 274  
minus sign (use of tilde ~), 820  
Mnesia (Erlang database), 387  
**mod** (integer modulo) operation, 54, 821  
model  
    computation, 29, *see computation model*  
    dialog (in GUI), 695  
    domain (in GUI), 695  
    GUI formalism, 695  
    logical semantics, 633  
    presentation (in GUI), 695  
    programming, 29  
model-based GUI design, 695  
modularity, xxi, 315, 409, 458  
    encapsulated search, 625  
    inadequacy of declarative model, 315  
reengineering, 522  
relation to concurrency, 239, 252, 319  
relation to explicit state, 315  
relational model, 660  
system decomposition, 457  
module, 183, 220, 454  
    **Array**, 436  
    **Atom**, 824  
    **Browser**, 224  
    **Char**, 821, 823  
    **Compiler**, 690, 815  
    **Connection**, 715, 732  
    **Dictionary**, 437  
    **Distribution** (supplement), 718  
    **FD**, 775  
    **Fault**, 739, 743  
    **File** (supplement), 211, 292, 564  
    **Finalize**, 481  
    **Float**, 821  
    **Int**, 821  
    **List**, 258, 385, 829  
    **Module**, 224, 413, 455, 730, 817  
    **MyList** (example), 222  
    **Number**, 14, 54, 182, 821  
    **OS**, 371, 609, 692, 699, 730  
    **ObjectSupport**, 517  
    **Open**, 564, 729  
    **Pickle**, 216, 223, 717  
    **Port**, 673  
    **Property**, 93, 253, 255  
    **QTk**, 213, 680, 729  
        interactive use, 214, 684  
        use in application, 225  
    **Record**, 826  
    **Remote**, 255, 732  
    **Search**, 776  
    **Space**, 654, 763  
    **String**, 824  
    **Thread**, 255, 276  
    **Time**, 304  
    **Tk**, 703  
    **Tuple**, 827  
    **Value**, 328, 830

- Base, 222, 229, 729
- compilation unit, 454
- dynamic linking, 285
  - failure, 329
- dynamic typing, 105
- Erlang, 389
- importing, 224
- interface, 455
- library, 229
- resource, 729, 746
- specification, 221
- System, 222, 229, 729
- MOGUL (Mozart Global User Library), 222
- monad, xxi, 309, 332
- monitor, 579, 592–600
  - condition variable, 598
  - guarded method, 595
  - implementation, 597
  - Java language, 616
  - Java semantics, 592
- monolithic
  - function, 296
  - stateful programming, 471
- monotonicity, 849
  - constraint programming, 766
  - dataflow variable, 336, 570
  - $need(x)$  predicate, 796
  - need property, 283
  - store, 781
  - thread reduction, 239, 781, 782
- Moore’s law, 176, 622
- Moore, Gordon, 176
- Morrison, J. Paul, 257
- Mozart Consortium, xxiv, xxvi, 106
- Mozart Programming System, 254
  - Base modules, 229, 729
  - Browser tool, 1, 88
    - displaying cyclic structures, 102
    - displaying strings, 716
    - WaitQuiet operation, 798
  - cheap concurrency, 252
  - command line interface, 817
  - Compiler Panel tool, 815
  - Distribution Panel tool, 815
  - Explorer tool, 757, 815
  - garbage collection, 78
  - GlobalStore, 743
  - interactive interface, 1, 87, 815
  - kernel languages, 844
  - library modules, 229
  - limited role of the compiler, 504
  - memory consumption, 174
  - Open Source license, xxiv
  - overview, xxiv
  - Panel tool, 815
  - performance, 201, 379
  - Prototyper tool, 689
  - separate compilation, 457
  - Standard Library, 214, 225, 230, 685, 690
  - System modules, 229, 729
  - thread scheduler, 253
  - uncaught exception, 93, 801
- multi-agent systems (MAS), xxiii, 345, 362, 576
- multimedia, 176
- multiprocessor
  - cluster, 711
  - shared-memory, 710
- multiset, 240
- mutable store (for cells), 794
- Myriorama, 273
- n-queens problem, 629
- name, 203, 714, 791, 824, 847
  - defining scope, 508
  - distributed computing, 711
  - fresh, 203
  - generation, 206
  - impure, 715
  - pure, 715
- name server, 403, 711
- natural design, 461, 569
- natural language, xiii, 31, 38, 641
- natural selection, 451, 462
- Naur, Peter, 32

- needed variable, 283, 796
- negation as failure, 662, 674
- nesting marker, 53, 83, 355, 365
- Netherlands, 819
- network awareness, 387, 723
- network transparency, 255, 387, 708
- neutral element, 187
- New operation, 495, 549
- NewActive operation, 558
- NewActiveExc operation, 562, 728
- NewArray operation, 436
- NewCell operation, 416
- NewDictionary operation, 437
- NewLock operation, 22, 583
- NewName operation, 203, 792, 824
- NewPort operation, 349
- NewSpace operation, 764
- NewStat operation, 725
  - resilient, 742
- Newton's method for square roots, 119
- Newton, Isaac, 278
- nil, 828
- node (in Erlang), 387
- noise (electronic), 473
- nonblocking operation, 333
  - `receive` (in Erlang), 394
  - delete (in queue), 598
  - read (in tuple space), 586
  - receive, 333
  - send, 333
  - stack, 578
- noncompositional design, 461
- nondeterminism
  - `choice` statement, 621, 623, 772
  - declarative concurrent model, 575
  - don't know, 324, 621
  - introduction, 20
  - limitation of declarative model, 319
  - logic programming, 638
  - observable, 20, 234, 319, 570, 573, 575, 576
- bounded buffer, 595
- Filter operation, 386
- lack of in Flavius Josephus problem, 561
- relation to exceptions, 327
- relation to coroutines, 275
- thread scheduler, 253
- nonstrict evaluation, 331
- Haskell, 310
- nonvar operation (in Prolog), 660, 662
- normal distribution, 476
- normal order reduction, 330
- Haskell, 309
- notify operation (in monitor), 592
- notifyAll operation (in monitor), 593
- NP-complete problems, 176
- NUL character, 841
- number, 52, 819
- O'Keefe, Richard, 111, 407, 668
- object, 413, 420, 542, 546
  - declarative, 423, 483, 568
  - introduction, 17
  - Java, 551, 807
  - passive, 556, 575, 576
  - strong, 542
- object code, 221
- object graph, 553
- object, active, 350, 556–567
  - comparison with passive object, 556
  - defining behavior with a class, 558
  - polymorphism, 429
- object, port, 346, 350, 413
  - approaches to concurrency, 573
  - Erlang, 563
  - further reading, 582
  - many-to-one communication, 351
  - polymorphism, 429
  - reactive, 351
  - reasoning, 352

- sharing one thread, 378
- when to use, 576
- object, stream, 256, 265–266, 411, 413, 574
  - comparison with port object, 351
  - Flavius Josephus problem, 561
  - higher-order iteration, 258
  - polymorphism, 429
  - producer/consumer, 257
  - transducer, 259
- Ockham, William of, 29
- octal integer, 819
- Okasaki, Chris, 175, 330
- OLE (Object Linking and Embedding), 462
- OMG (Object Management Group), 462
- open distribution, 709, 711, 714
- open program, 202
- Open Source software, xxiv
- operating system, 208
  - Linux, xxvi, 201, 471, 499
  - Mac OS X, xxiv, xxvi, 254
  - Solaris, xxvi, xxix
  - Unix, xxiv, 254, 305, 459, 553, 642
  - VM (Virtual Machine), 41
  - Windows, xxiv, 254
- operation (in data abstraction), 420
- operational semantics, 38, 60, 635, 779–811
- operator
  - associativity, 34, 837
  - binary, 34, 836
  - infix, 52, 83, 185, 793, 826, 828, 830, 836
  - mixfix, 826, 837
  - postfix, 837
  - precedence, 34, 836
  - prefix, 836
  - ternary, 838
  - unary, 836
- OPI (Oz programming interface), 815
- optimistic scheduling, 603
- optimization, 177
  - avoid during development, 452
  - combinatoric, 661
  - compiler, 162
  - eager execution, 302
  - early error detection, 504
  - first-argument indexing, 659
  - memoization, 694
  - monitor performance, 597
  - object system, 545
  - relational programming, 622
  - short-circuit protocol, 559
  - standard computer, 314
  - thread priorities, 265
- order-independence, 49
  - unification, 110, 232
- orelse** operator, 83
- otherwise method, 500
- OTP (Ericsson Open Telecom Platform), 386
- overloading, 313, 429
- overriding relation, 502
- Oz, Wizard of, 1
- ozc** command, 228, 817
- pairwise distinct, 756, 774, 782
- palindrome, 628
- palindrome product problem, 628, 757
- Panangaden, Prakash, 338
- Panel tool, *see* Mozart Programming System
- Papert, Seymour, xiii, 233
- paradigm, xiii, xvi, *see* computation model
  - declarative, 29
  - school of thought, xvi
- paragraph (in OPI), 815
- parallelism, 237, 322
  - difference with concurrency, 322
  - importance of nonstrictness, 331
  - importance of worst-case complexity, 174
- parameter passing, 430–434

- call by name, 432
  - exercise, 484
- call by need, 433
  - exercise, 485
  - lazy evaluation, 433
- call by reference, 57, 430
  - Java, 555
- call by value, 431
  - Java, 555
- call by value-result, 431
- call by variable, 430
- parity, 14
- Parnas, David Lorge, xxii, 462
- parser, 32, 161–166
  - generic, 650
  - gump tool, 39
  - natural language, 641
- partial failure, 326
- partial list, 440, 829
- partial order, 238
- partial termination, 243, 338, 804
- partial value, 46
- Pascal’s triangle, 4
- Pascal, Blaise, 5
- pass by ..., *see* parameter passing
- pattern matching
  - case** statement, 6, 67
  - function (in Erlang), 388
  - Haskell, 309
  - receive** expression (in Erlang), 391
  - reduction rule semantics, 784
- PDA (procedural data abstraction), 420
- pencil, xviii
- Pentium III processor, 201, 471
- performance
  - cluster computing, 711
  - competitive concurrency, 254
  - constraint programming, 758
  - Cray-1 supercomputer, 175
  - declarative programming, 313
  - dictionary, 201
  - distributed stream, 724
- lazy language, 289, 342
- measuring, 167
- memoization, 25, 315
- mobile object, 724
- monitor, 597
- Mozart Programming System, 201, 379
- personal computer, 175
- price of concurrency, 339
- record field access, 438
- role of optimization, 177, 265, 302
- role of parallelism, 237, 322
- transitive closure, 471
- word-of-mouth simulation, 486
- permanent failure, 739
- permutations, 2
- persistence
  - data structure, 149, 297
  - database, 654
  - Erlang, 387
  - transaction, 600
- personal computer, 3, 252, 254, 289, 304
  - low-cost, 74, 175
- pessimistic scheduling, 603
- Phidani Software, 642
- $\pi$  calculus, xvii, 41, 54, 805
- pipeline, 259
- pixel, 556
- placeholder
  - dataflow variable, 86
  - future (in Multilisp), 336
  - GUI design, 686, 703
- planning
  - flight, 671
  - WARPLAN, 621
- Plotkin, Gordon, 779
- point
  - resting, 338
  - synchronization, 333
  - two-dimensional space, 554
- pointer, 76
  - content edge, 733

- dependency, 459
- dynamic typing, 106
- garbage collection, 76
- memory block, 480
- resource, 480
- state, 733
- POLA (Principle of Least Authority), 209
- polymorphism, 18, 106, 425, 462, 493
  - active objects, 429
  - ad-hoc, 429
  - apportioning responsibility, 425
  - example, 530
  - Haskell, 312
  - object-oriented programming, 490
  - port objects, 429
  - stream objects, 429
  - universal, 429
- Ponsard, Christophe, 545
- port (explicit state), 349, 719, 848
  - communication from inside encapsulated search, 673
  - distributed semantics, 383
- `Port.sendRecv` operation, 673
- portal, 476
- postcondition, 441, 521
- potential function, 175
- precondition, 441, 521
- predicate calculus, 633
- preemption, 252
- preprocessor, 318
  - DCG (in Prolog), 649
  - design patterns, 536
  - extended DCG (in Prolog), 140
  - fallacy of, 318
- presentation model (in GUI), 695
- principle
  - abstraction, 410
  - avoid changing interfaces, 458
  - avoid premature optimization, 177, 452
  - balance planning and refactoring, 452
  - centralized first, distributed later, 745
  - class is final by default, 492
  - compartmentalize responsibility, 425, 451
  - concentrate explicit state, 412
  - creative extension, xiv, 844
  - decisions at right level, 460
  - declarative concurrency, 242, 281
  - document component interfaces, 451
  - documented violations, 460
  - eager default, lazy declared, 330
  - encapsulate design decisions, 458
  - enriching control (in logic programming), 640
  - error confinement, 90
  - “everything should be an object”, 542
  - exploit data abstraction uniformity, 543
  - form mirrors content, 544
  - freely exchange knowledge, 451
  - function structure follows type structure, 135
  - functional abstraction, 4
  - last call optimization, 72
  - layered language design, 850
  - least authority (POLA), 209
  - least expressiveness, 323
  - least privilege, 209
  - minimize dependencies, 387, 459
  - minimize indirections, 459
  - model independence, 457
  - more is not better or worse, just different, xx
  - Mozart design rules, xxvi
  - natural selection, 451, 462
  - need to know, 209
  - objects over ADTs, 490
  - pay only on use, 620
  - predictable dependencies, 460
  - run time is all there is, 504, 690
  - separation of concerns, 567
  - stateful component with declaration, 451

- tive behavior, 417
- substitution property, 518, 521, 523
- syntax stability, 643
- system decomposition, 210
- type first, 137
- use data abstraction everywhere, 489, 543
- working software keeps working, 59, 459, 722
- private scope, 507, 508
  - C++ and Java sense, 508
  - Smalltalk and Oz sense, 507
- probability
  - exponential distribution, 475
  - Gaussian distribution, 476
  - normal distribution, 476
  - uniform distribution, 474
  - unique name generation, 207
- problem
  - cryptarithmetic, 755, 776
  - digital logic satisfiability, 176
  - Flavius Josephus, 558–561
  - flight planning, 671
  - grocery puzzle, 774
  - halting, 209, 681
  - Hamming, 293, 342
  - intractable, 176
  - library scheduler, 672
  - making change, 775
  - n-queens, 629
  - NP-complete, 176
  - palindrome product, 628, 757
  - Send-More-Money, 755, 776
  - undecidable, 209
  - zebra puzzle, 774
- proc** statement, 65, 792
- procedure
  - as component, 412
  - basic operations, 55
  - external reference, 65
  - importance, 54
  - order, 177
  - tail-recursive, 72
- procedure value (closure), 65, 178, 792
  - anonymous, 53
  - common limitation, 179, 552
  - distributed lexical scoping, 722
  - encoding as an object, 540
  - higher-order programming, 177
  - relation to inner class, 552
- process
  - concurrent calculus, 54
  - concurrent program design, 364
  - CSP, 619
  - distributed system, 707
  - Erlang, 350, 386, 389
  - large program design, 450
  - operating system, 255
  - producer and consumer, 724
  - run-time error, 96
  - small program design, 218
- processor, 237
  - cluster computing, 711
  - dataflow machine, 337, 469
  - parallel functional programming, 331
  - shared-memory multiprocessor, 710
- producer, 257
- profiling, 177, 452
- program design, *see* software development
- program point, 444, 606
- programming, xv, 1
  - accumulator, 139
  - component-based, 412
  - concurrent, 573
  - constraint, 44, 274, 577, 663
  - data-centered, 576
  - declarative, 29, 406
    - descriptive, 115
    - need for algorithms, 116
    - programmable, 115
  - Erlang, 388
  - flow-based, 257
  - functional, 406

- future developments, 461
- good style, xxi
- Haskell, 309
- higher-order, 113, 123, 177–194
  - introduction, 13
  - relation to object-oriented, 538
- imperative, 29, 406
- Java, 552, 615
- kernel language approach, xvi
- logic, 44, 101, 142, 406, 632
- multi-agent, 412, 576
- multiparadigm, xiv, xxvi
  - event manager, 566
  - nonalgorithmic, 622
  - object-based, 19, 538
  - object-oriented (OOP), 19, 413, 489
  - open, 105, 202
  - paradigm, xiii, xvi, 29, *see computation model*
- Prolog, 663
- real-time, 304
- relational, 621
- stateful, 29
- stateless, 29
- synchronous, 266
- programming model, xiii, 29
- Prolog, 660–671
  - Aquarius, 140, 661
  - Parma, 661
  - SICStus, 190, 663
    - state threads package, 190
  - proof engineering, 117
  - proof rule, 444
  - propagate-and-search, 629, 750
  - propagator, 752, 760
  - property
    - liveness, 602
    - object, 497
    - safety, 602
  - propositional logic, 632
  - protected scope, 508
    - C++ sense, 509
    - Java sense, 567
- protection boundary, 202
- protector, 325
- protocol, 353
  - by-need, 282
  - communication, 715
  - consistency, 712
  - DHCP (Dynamic Host Connection Protocol), 207
  - distributed binding, 733
  - distributed unification, 733
  - eager copy, 734
  - eager immediate copy, 734
  - interaction (in GUI), 682
  - invalidation, 733
  - IP (Internet Protocol), 206
  - lazy copy, 733
  - meta-object, 516
  - mobile state, 733
  - negotiation, 376
  - short-circuit, 559
  - stationary state, 733
  - TCP (Transmission Control Protocol), 712, 740
  - timer, 368
- Prototyper tool, 689
- pseudorandom numbers, 473
- Psilion Series 3 palmtop computer, 378
- public scope, 507
- pure object-oriented language, 543
- QTk, 213, 680, 729
  - interactive use, 214, 684
  - Prototyper, 689
  - use in application, 225
- quadratic equation, 179
- quantifier, 441, 445, 633, 645
  - existential (in Prolog), 671
- quantum (in thread scheduling), 252
- query
  - database, 655
  - logic programming, 634
- queue, 145
  - amortized ephemeral, 147
  - amortized persistent, 298

- breadth-first traversal, 156
- concurrent, 379, 583
- nonblocking delete, 598
- priority, 605, 614
- worst-case ephemeral, 147
- worst-case persistent, 299
- race condition, 20, 234
- raise** statement, 93, 801
- random number generation, 472
- rational tree, 760
- Raymond, Eric, 462
- reachable memory, 74
- ReadOnly** operation, 799
- real-time computing
  - garbage collection, 76
  - hard, 174, 253, 254, 304
  - soft, 304
- reasoning
  - algebraic, 111, 116, 323
  - atomic action, 581
  - causal, 353, 575
  - lift control system, 375
  - logical, xix, 111, 632
  - message-passing concurrent model, 352
  - shared-shate concurrent model, 324
  - stateful model, 324, 440
- receive
  - asynchronous, 332
  - nonblocking, 333
  - synchronous, 332
- receive** expression (in Erlang), 391
- recomputation, 761, 776
- record, 19, 52, 825
  - adjoin, 827
  - basic operations, 54, 826
  - dynamic creation, 165, 549, 695
  - importance, 53
  - type, 438
  - usage trade-offs, 438
- recurrence equation, 167
- recursion, 3, 113, 124
  - direct, 113
  - indirect, 113
  - mutual, 110
  - polymorphic, 309
  - programming with, 127
  - Prototyper tool, 690
  - tail recursion optimization, 72
- red cut (in Prolog), 670
- Red Hat Corporation, xxvi, 201, 471
- reduction order, 330–332
  - applicative, 330
  - normal, 330
- reduction rule, 784
- reengineering, 522
- refactoring, 452
- reference, 714
- referential transparency, 113
- reflection, 515
- region (in OPI), 815
- register
  - abstract machine, 56
  - forwarding, 621
  - memory management, 74
- registration
  - action procedures (in GUI), 683
  - display refresh (FlexClock), 700
  - distributed binding, 737
  - finalization, 481
- relation, 655
- relative error, 120
- reliability, 711
- rendezvous, 619
- replicator, 326
- research project, xxiv
- resolution
  - deadlock, 605
  - logic programming, 635, 640, 662
  - SLDNF, 662
  - video display, 321
- resource
  - distributed component, 746
  - distributed system, 729
  - external, 77, 480
  - file descriptor, 293

- localized, 709
- producer/consumer pipeline, 261
- use of laziness, 289
- responsibility
  - atomicity and consistency (in transaction), 600
  - compartmentalize (in a team), 451
  - coroutine (avoiding starvation), 275
  - design by contract, 521
  - distributed memory management, 738
  - dynamic typing, 493
  - failure confinement, 245
  - memory management, 76
  - role of polymorphism, 425
  - type inference, 137
- resting point, 338
- restriction (environment), 62
- retract/1** operation (in Prolog), 662
- return** (in **for** loop), 190
- Reuter, Andreas, 582, 600
- Reynolds, John C., 419
- right, *see* name
- Rinard, Martin C., 338
- RISC (Reduced Instruction Set Computer) microprocessor, 621
- RMI (remote method invocation), 354, 709, 724, 725
- root variable, 763
- Round operation, 822
- RPC (remote procedure call), 354, 709
- rubber band, 251
- runic inscription, 779
- Runnable** interface (in Java), 616
- s-expression, 650
- Sacks, Oliver, 405
- safety, 602
- Saint-Exupéry, Antoine de, 111
- Santayana, George, 694
- Saraswat, Vijay A., 338, 662, 808
- scalability
  - compilation, 458
  - multiprocessor, 710
  - program development, 105
  - weighted reference counting, 737
- scalar product (constraint), 775
- scheduler
  - Delay** operation, 304
  - deterministic, 253
  - lift control system, 366
  - nondeterministic, 253
  - randomness, 473
  - resource allocation, 672
  - round-robin, 252, 256
  - thread, 239, 252
  - transaction, 603
- Schulte, Christian, xxvi
- science, xv, xviii
- scientific method, xvii
- scope, 56, 507
  - attribute, 510
  - dynamic, 59
  - lexical, 57, 59, 64, 508, 539
    - absence in Prolog, 661
    - distributed, 722
    - hiding, 221, 411, 423, 442, 483, 495, 549
    - substitution, 803
  - private, 507, 508
  - protected, 508
  - public, 507
  - static, *see* lexical
  - user-defined, 508
- search
  - aggregate, 670
  - all-solutions, 626
  - binary, 151
  - branch-and-bound, 772
  - breadth-first, 644
  - communication from inside encapsulated search, 673
  - constraint programming, 274
  - contribution of AKL, 809
  - danger, 639

- database query, 657
- depth-first, 622, 644
- deterministic, 621
- encapsulated, 625
- generate-and-test, 629, 758
- iterative deepening, 644
- linear, 197
- logic programming, 661
- n-queens problem, 629
- one-solution, 626
- overuse, xxi
- propagate-and-search, 629, 750
- pruning, 662
- relational computation model, 623
- relational programming, 621
- saturation, 772
- search space, 622
- search strategy, 761
- search tree, 624
- security
  - abstract data type, 201–210
  - application, 744
  - atom vs. name, 508
  - capability, 208
  - data abstraction, 419–435
  - distributed resources, 731
  - distributed system, 743
  - engineering, 744
  - hardware, 744
  - human society, 208
  - implementation, 744
  - kernel language concepts, 847
  - language, 208, 744
  - linguistic abstraction, 39
  - mechanism, 208
  - open distribution, 711
  - policy, 208
  - right, 791, 847
  - static typing, 106
  - threat model, 744
- self**
  - clone, 517
  - delegation, 511
- dynamic binding, 505
- forwarding, 511
- Java, 553
- `this` notation, 551
- self** (in Erlang), 390
- semantic stack, 62
  - Runnable, 62
  - suspended, 62
  - terminated, 62
- semantics, 31
  - abstract machine, 56–78, 92–94, 239–241, 282–283, 348–349, 416–417
  - axiomatic, 38, 440–450, 632
  - by-need trigger, 282
  - cell, 416
  - common abstractions, 808
  - denotational, 38
  - exceptions, 92
  - interleaving, 780
  - kernel language, *see abstract machine*
  - kernel language approach, 38
  - logical, 38, 631–641
  - monitor (in Java), 592
  - operational, 38, 60, 635, 779–811
  - port, 348, 383
  - secure types, 203
  - semantic statement, 61
  - SOS (structural operational semantics), 779
  - thread, 239
- Send operation, 349
  - slot-reserving semantics, 384
- send
  - asynchronous, 332
  - latency, 263
  - nonblocking, 333
  - synchronous, 332
- Send-More-Money problem, 755, 776
- separation of concerns, 567
- sequential logic, 269
- serializability, 600
- serialization, 709

- serializer, 325
- set comprehension, 301
- setof/3** operation (in Prolog), 626, 666, 670
- Shakespeare, William, 815
- shared-state concurrency, *see* atomic action, *see* lock, *see* monitor, *see* transaction
- sharing, 418
  - Browser tool, 102, 829
  - distributed state, 720
  - distributed value, 716
  - thread, 378
- short-circuit
  - concurrent composition, 277
  - Flavius Josephus problem, 559
  - transitive closure, 464
- Show operation, 340
- side effect, 411
  - declarative, 288
- signal** operation (in monitor), 592
- signature (of procedure), 129
- simulation
  - components, 412
  - digital logic, 266–272
  - inadequacy of declarative model, 173
  - Internet, 412
  - multi-agent, 412
  - slow network, 578
  - small world, 486
  - word-of-mouth, 476
- Simurgh, 707
- single-assignment store, 42–49, 60, 781
  - importance, 43
- singularity, 176
- sink (consumer), 259
- 64-bit address, 78
- 64-bit word, 74, 175, 820
- skip** statement, 62, 785
- SLDNF resolution, 662
- small world
  - graph, 461
- simulation, 486
- Smolka, Gert, xxvi
- snapshot (of state), 437, 718
- software design, *see* design methodology, *see* language design
- software development, 218, 450
  - bottom-up, 451
  - compositional, 453
  - concurrent components, 362
  - distributed programming, 745
  - evolutionary, 451
  - extreme programming, 452
  - framework, 492
  - IID (Iterative and Incremental), 451
  - importance of names, 508
  - in the large, 450
  - in the small, 218
  - incremental, 451
  - interactive interface, 87
  - iterative, 451
  - middle-out, 451
  - stepwise refinement, 465, 604
  - test-driven, 452
  - top-down, 8, 451
- software engineering, 450
  - component as unit of deployment, 221
  - concurrency, 233
  - distributed lexical scoping, 722
  - further reading, 462
  - informatics curriculum, xxii
  - lexical scoping, 59
- software rot, 459
- Solaris operating system, xxvi, xxix
- Solve** operation, 626, 773
- SolveAll** operation, 626
- SolveOne** operation, 626
- Sort** operation, 670, 829
- SOS (structural operational semantics), 779
- source (producer), 259
- source code, 221
  - interactive, 815

- million line, xvi, 36, 387, 457, 458  
nonexistent, 492  
preprocessor input, 318  
reengineering, 522  
set of functors, 285  
textual scope, 64  
variable name, 44
- space, *see* computation space, *see also* memory  
space leak, *see* memory leak  
specification, 410  
    component, 461  
specification language, 116  
speculative execution (in nonstrict language), 331
- stack  
    declarative object, 423  
    depth-first traversal, 156  
    memory management, 74  
    open declarative, 195, 421  
    proving it correct, 442  
    secure declarative bundled, 423  
    secure declarative unbundled, 205, 422  
    secure stateful bundled, 424  
    secure stateful unbundled, 424  
    semantic, 61  
    stateful concurrent, 578  
standalone application, 222  
    **declare** not allowed, 87  
    Java, 555  
    uncaught exception, 93
- starvation, 275  
    wait set implementation, 597
- state  
    cell (mutable variable), 414  
    declarative, 408  
    explicit, 16, 409  
    implicit, 408  
    interaction with call by name, 485  
    lazy execution, 481  
    lazy language, 331  
    memory management, 77
- modularity property, 315  
nonstrict language, 331  
port (communication channel), 347  
reasoning with, 38, 440  
revocable capability, 434  
threading, 139  
transformation, 133
- state transition diagram, 353, 368  
component design, 364  
floor component, 369  
lift component, 371  
lift controller component, 369  
transaction, 607
- stateless (declarative programming), 111
- statement  
    **case**, 67, 790  
    **catch** (clause in **try**), 94  
    **choice**, 623, 772  
    **conc**, 278  
    **declare**, 2, 87  
    **fail**, 623  
    **finally** (clause in **try**), 94  
    **for**, 188  
    **fun**, 84  
    **functor**, 223  
    **gate**, 272  
    **if**, 66, 790  
    **local**, 56, 63, 786  
    **lock**, 22, 583  
    **proc**, 65, 792  
    **raise**, 93, 801  
    **skip**, 62, 785  
    **thread**, 241, 785  
    **try**, 92, 799  
    break, 486  
    compound, 117  
    compound (in Java), 552  
    declarative kernel language, 49  
    interactive, 87  
    procedure application, 66  
    sequential composition, 63, 785  
    suspendable, 65

- value creation, 63
- variable-variable binding, 63
- static
  - binding, 506
  - linking, 222
  - scope, *see* scope, lexical
  - typing, 51, 104–106
- `stdin` (standard input), 229, 553
- `stdout` (standard output), 553
- Steiner, Jennifer G., 334
- Stirling’s formula for factorial, 618
- storage manager, 325
- store, 781
  - equivalence, 785
  - mutable (for cells), 416
  - mutable (for ports), 348
  - need, 780, 795
  - predicate, 781
  - read-only, 206, 798
  - single-assignment, 42–49, 60, 99, 235, 781
  - trigger, 282, 795
  - value, 43
- stream, 795
  - deterministic, 257
  - Java, 553
  - merger, 395
  - producer/consumer, 257
  - usage trade-offs, 439
- strict ..., *see* eager ...
- strict two-phase locking, 604
- strictness analysis, 289, 310, 342
- string, 53, 830
  - virtual, 211, 831
- `StringToAtom` operation, 824
- structure
  - compiler, 162
  - compositional, 461
  - difference, 141
  - distribution, 255
  - effect of concurrency, 252
  - grammar, 32
  - hierarchical, 453
  - interpreter, 653
- noncompositional, 461
- program, 219, 220
- structure equality, 103, 418, 723
- substitution, 126, 803
- substitution property, 518, 521, 523
- subtype
  - basic types, 52
  - class hierarchy, 518
- Sun Microsystems, xxvi, 462
- superclass, 503, 513, 556
- supercomputer, 175
- supply-driven execution, *see* eager execution
- suspension
  - `Delay` operation, 305
  - due to program error, 48, 89
  - thread, 239, 276
- Sussman, Gerald Jay, 42
- Sussman, Julie, 42
- Symbian Ltd., 378
- symbolic link, 459
- synchronization, 333–337
  - clock, 308
  - dataflow, 790
- `synchronized` keyword, 593, 616
- synchronous communication, 332
  - active object variant, 562
  - component interaction, 456
  - CSP, 619
  - dependency, 387
  - error reporting, 360
  - failure detection, 400, 739
  - fault confinement, 745
  - receive, 332
  - send, 332
- synchronous programming, 266
- syntactic sugar, 40, 79–84
  - dynamic record creation, 165
  - `local` statement, 40
  - state transition diagram, 369
- syntax, 31
  - convention for examples, xxix
  - language, 31
  - nestable constructs (in Oz), 833

- nestable declarations (in Oz), 833
- Oz language, 833
- Oz lexical, 839
- Prolog, 663
- term (in Oz), 833
- synthesized argument, 161
- system exception, 96
- Szyperski, Clemens, 462
- tail call optimization, 72
- Tanenbaum, Andrew S., 334
- task (in concurrency), 780
- tautology, 632
- TCP (Transmission Control Protocol), 712, 740
- technology, xv
  - dangers of concurrency, 21
  - history of computing, 176
  - magic, 314
  - molecular computing, 176
  - Prolog implementation, 661
  - reengineering, 522
  - singularity, 176
  - software component, 462
  - synchronous digital, 267
  - transition to 64-bit, 78
- Tel, Gerard, 353
- tell operation, 782, 787
- temporal logic, 603
- temporary failure, 739
- term
  - Erlang, 391
  - Oz, 833
  - Prolog, 664
- termination
  - detection, 276, 382
  - ping-pong example, 305
  - failure in declarative program, 245
  - partial, 243, 338, 804
  - proof, 449
  - total, 804
- test-driven development, 452
- testing
  - declarative programs, 111, 407
  - dynamic typing, 105
  - programming in the small, 219
  - stateful programs, 407
- text file, 210
- Thalys high-speed train, 382
- theorem
  - binomial, 4
  - Church-Rosser, 331
  - Gödel's completeness, 634
  - Gödel's incompleteness, 634
  - halting problem, 681
- theorem prover, 117, 634, 662
- Therac-25 scandal, 21
- thinking machine, 621
- third-party independence, 335
- 32-bit address, 78
- 32-bit word, 74, 174
- this, *see self*
- Thompson, D'Arcy Wentworth, 405
- thread, 846
  - declarative model, 233
  - hanging, 399
  - interactive interface, 89
  - introduction, 15
  - Java, 615
  - monotonicity property, 239, 781, 782
  - priority, 253
  - ready, 239
  - runnable, 239
  - suspended, 239
  - synchronization, 333
- thread statement, 241, 785
- Thread class (in Java), 616
- throughput, 263
- thunk, 432
- ticket, 480, 714
  - Connection module, 715
- ticking, 307
- time complexity, 11
- time slice, 252–254
  - duration, 254

- time-lease mechanism, 480, 734, 738
- time-out, 740
  - Erlang, 391–394
  - system design, 460
- timer protocol, 368
- timestamp, 207, 602
- timing measurement
  - active object, 379
  - memory consumption, 173
  - palindrome product (constraint version), 758
  - palindrome product (naive version), 629
  - transitive closure, 471
  - word frequency, 201
- token equality, 418, 714, 723
- token passing, 579, 588, 591, 721
- token syntax (of Oz), 833
- tokenizer, 32, 162
- top-down software development, 8, 451
- total termination, 804
- trade-off
  - asynchronous communication vs. fault confinement, 745
  - compilation time vs. execution efficiency, 457
  - compositional vs. noncompositional design, 461
  - dynamic vs. static scoping, 58
  - dynamic vs. static typing, 104
  - explicit state vs. implicit state, 315, 409
  - expressiveness vs. execution efficiency, 116
  - expressiveness vs. manipulability, 681
  - functional decomposition vs. type decomposition, 542
  - helper procedure placement, 120
  - indexed collections, 435
  - inheritance vs. component composition, 462, 492
  - kernel language design, 844
  - language design, 811
  - lazy vs. eager execution, 329
  - memory use vs. execution speed, 177
  - names vs. atoms, 510
  - nonstrictness vs. explicit state, 331, 344
  - objects vs. ADTs, 428
  - optimistic vs. pessimistic scheduling, 603
  - planning vs. refactoring, 452
  - simple semantics vs. efficient implementation, 788
  - single vs. multiple inheritance, 532
  - specification language vs. programming language, 116
  - testing declarative vs. stateful programs, 407
  - type view vs. structure view, 520
- transaction, 580, 600–615
  - bank example, 202
  - database, 655
  - distributed, 615
  - fault-tolerant store, 743
  - incarnation, 607
  - logging example, 505
  - nested, 615
  - restart, 606
  - tell operation, 788, 811
  - two-phase locking, 604
- transaction manager, 610
- transaction processing monitor, 611
- transducer, 259
- transition, 368
- transitive closure, 463–471
  - declarative algorithm, 465, 468
  - Floyd-Warshall algorithm, 470
  - stateful algorithm, 468
- translator, *see* compiler
- tree, 150
  - balanced, 150
  - depth, 151
  - drawing, 158

- finite, 150
- Haskell expression, 310
- node depth, 156
- ordered binary, 151
- parse, 32, 162, 643
- rational, 760
- search, 624
- stream merge, 398
- syntax, 162
- ternary, 150
- traversal, 155
- trigger, 281–285, 847
  - activation, 282
  - by-need, 575
  - definition using `WaitNeeded`, 795
  - implicit, 282
  - programmed, 193, 262, 282
- true**, 824
- try** statement, 92, 799
- tuple, 52, 826
  - usage trade-offs, 438
- tuple space, 456, 586–590
- Turing machine, 41, 115, 681, 846
- Turing, Alan, 115
- two-phase locking, 604
- type, 50, 195
  - abstract, 195
  - ADT, 195
  - atomic, 824
  - basic, 52
  - bundled, 420
  - class, 313
  - concurrent data abstraction, 578
  - declarative, 420
  - descriptive, 129
  - dynamic, 51, 104–106, 709
  - hierarchy, 52
  - inference, 98, 137, 309
  - open, 420
  - PDA (object), 420
  - polymorphic, 312
  - secure, 201–210, 419–435
  - signature, 129, 309
  - stateful, 420
- stateless, 420
- static, 51, 104–106
- strong, 104, 308
- unbundled, 420
- weak, 104
- type decomposition, 522, 542
- UML (Unified Modeling Language), 489, 528
- undecidable problem, 209
- Unicode, 458, 459, 820
  - Java, 553
- unification, 98–104, 846
  - algorithm, 101
  - distributed, 733
  - equivalence set of variables, 101
  - examples, 99
- unification grammar, 649–650
- uniform distribution, 474
- unit**, 824
- Unix operating system, xxiv, 254, 305, 459, 553
  - lex/yacc parsing tools, 642
  - pipe, xvi, 257
- unmarshaling, 709
- URL (Uniform Resource Locator), 211, 715
  - limited lifetime, 460
- value, 43
  - compatible, 46
  - complete, 46
  - compound, 43
  - failed, 328, 802, 848
  - in data abstraction, 420
  - partial, 46
- value store, 43
- var** operation (in Prolog), 662
- variable, 2
  - binding, 44, 45, 787
  - dataflow, 42, 48, 846
  - declarative, 42
  - determined, 66, 101, 206, 283, 333, 660, 791

- domain, 760
- equivalence set, 101
- escaped, 500, 509
- final (in Java), 551
- free, 58, 633
- fresh, 651, 787
- global, 87
- identifier, 2, 44
- instance, 497
- interactive, 87
- local, 180
- logic, 101
- mutable, *see* cell
- needed, 283, 796
- read-only, 206, 209, 348, 380, 798, 847
- root, 763
- single-assignment, 336
- special (in Common Lisp), 59
- unification, 99
- Virding, Robert, 582
- virtual machine, xvii, 41, 676
- virtual string, 211, 831
- visibility
  - component-based programming, 364
  - computation space, 764
  - horizontal, 509
  - vertical, 507
- Visual Basic, 461
- Vlissides, John, 534
- VLSI-BAM microprocessor, 621
- VM (Virtual Machine) operating system, 41
- Vossen, Gottfried, 600
  
- Wait** operation (in dataflow), 791
- wait** operation (in monitor), 592
- wait point, 579
- wait-for graph, 605
- WaitNeeded** operation, 797, 847
- WaitQuiet** operation, 798, 803
- WaitStable** operation, 767
  
- WaitTwo** operation, 319, 320, 393, 396, 399, 641
- WAN (wide area network), 711, 739
- Warren, David H.D., 621
- watcher, 739, 742
- Web services, 345, 356
- Weikum, Gerhard, 600
- wheel, xviii
- while** loop, 447
- white space, *see* blank space
- widget (in GUI), 213, 681, 682
- Width** operation, 826
- Wikström, Claes, 582
- Wilf, Herbert S., 170
- Williams, Mike, 582
- window (in GUI), 682
- Windows operating system, xxiv, 254
- wire, 363
  - many-shot, 363
  - one-shot, 363, 368
- Wood, David, 378
- word
  - frequency, 198
  - memory, 74
  - specification, 219
- word-of-mouth simulation, 476
- wrapping and unwrapping, 204
- WWW (World Wide Web), 211
  
- XML (Extensible Markup Language), 115
  
- zombie, 559