

- [Wil92] Paul R. Wilson. Uniprocessor garbage collection techniques. In *International Workshop on Memory Management*, volume 637 of *Lecture Notes in Computer Science*, St. Malo, France, September 1992.
- [Win93] Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, 1993.
- [WS97] Mitchell Wand and Gregory T. Sullivan. Denotational semantics using an operationally-based term model. In *24th Symposium on the Principles of Programming Languages (POPL)*, pages 386–399, 1997.
- [You81] Richard M. Young. The machine inside the machine: Users' models of pocket calculators. *International Journal of Man-Machine Studies*, 15:51–85, 1981.

# Index

- , 771
- =
  - on domains, 792
  - on functions, 776
  - on sets, 771
  - on tuples, 773
- $\forall$ , 563, 586
- \*, 217
- \*cell\*, 335, 337
- \*failed\*, 376
- +, 217
  - denotational semantics, 254
  - operational semantics, 243
- , 217
- /, 217
  - denotational semantics, 254
  - operational semantics, 243
- :, 33, 34, 655
- :=, 329, 655, 658, 666
- <, 217
- <=, 217
- =
  - on types, 539
- =, 217
- >, 217
- >=, 217
- %, 217
- , 777
- ↦, 794
- let, 806
- , 777
- T, 167
- @ as infix *append*, 798
- !, 655
- ⊥, 144, 164, 167, 778
- . as infix *cons*, 798
- , 798
- ×, 773, 791–793
- \*, 796
- ∈, 770
- ∩, 771
- ∉, 770
- POSTFIX2, 52
- POSTTEXT, 98
- [], 796
- {}, 770
- ⊆, 771
- ⊂, 771
- ⊆, 551
- ↓, 771
- ↓, 773
- ∪, 771
- ^, 329
- ^, 655, 658, 659
- A-normal form, 767
- Abort, 352
- abort!, 352, 353
- Abstract data type, *see* Type
- Abstract interpretation, 527
- Abstract machine, 39
- Abstract syntax, 18–20
- Abstract syntax tree (AST), 18, 109
- Abstraction, 790, 803

- barrier, 790
  - data, *see* Data abstraction
  - lambda, 784
- Abstraction barrier, 599
- Abstraction violation, 601
- acquire!, 505
  - operational semantics, 507
- Action, 324
- Actor, 415
- ADA, 196, 314, 517, 748, 758
- add-first, 378
- Adequacy of denotational semantics, 149, 151
- after, 324
- Agenda, 495
- Algebra
  - semantic, 109
  - syntactic, 109
- Algebraic type schema, 665–668
- ALGOL 60, 259
- Alias, 363
- Aliasing, 654
- allocating, 345, 387, 431
- allocatingComp, 431
- allocatingComps, 431
- allocatingCopies, 440
- allocatingVals, 439
- Alpha-equivalence, 232
- Alpha-rename, 566, 629, 634
- Alternate, 18
- alts, 218
- and, 204
  - desugaring in FL, 209
- and?, 217
  - operational semantics, 243
- Answer domain, 382
- Answer domain, 109
- Answer of operational semantics, 40
- Anti-monotonic subtype, 554
- Anti-symmetric, 166, 774
- API, 599
- APL, 284, 517
- Appel, Andrew, 767, 768
- append, 797–798
- Applet, 400
- applet, 400, 401
- Application, 198, 204, 306, 802
  - desugaring in FL, 209
  - desugaring in HOOPLA, 307
  - of a function, 777
  - type of, 778
- Application programming interface (API), 599
- Applicative order reduction, 262
- arg-index, 219, 221
- arg?, 219, 221
- arithop, 34
- arithop-op, 219, 221
- arithop-rand1, 219, 221
- arithop-rand2, 219, 221
- arithop?, 219, 221
- Array, 418, 422
- assign, 342
- Assignment conversion, 356
- Association list, 418, 428
- Associative, 658
- AST (abstract syntax tree), 18
- Atomic, 504
- Axiom, 46–50
- axiom, 46
- Backtracking, 355, 367, 368
- BASIC, 270, 518
- begin, 306, 316, 327, 328, 337, 338, 341
  - denotational semantics, 316, 347
  - standard, 381
  - operational semantics, 316, 336
- begin-transaction!, 352
- Behavior, 43–44

- Observational equivalence and,
  - 83
  - of a concurrent program, 491
- Behavior of a concurrent program, 498
- Behavioral equivalence, 83
- Berra, Yogi, 489
- Bi-simulation, 511
- Big-step operational semantics
  - evaluation relation, 68
  - evaluation tree, 68
- Big-step operational semantics (SOS), 66
- Bijjective function, 780
- Binary relation, 774
- bind*, 250
- bind-exit*, 399
- Binding, 249
- Binding construct, 116, 138
- Binding occurrence, 226, 319
- binding-make*, 225
- binding-name*, 225
- binding-value*, 225
- Birrel, Andrew, 510
- Block structure, 283, 298
- Blocked thread, 493, 507
- Bool*, 770
- bool->int*, 465, 468–471
- bool=?*, 217
- boolean?*, 217
- BOS (big-step operational semantics), 66
- Bottom ( $\perp$ ), 164, 167, 778
- Bound identifier, 226
  - definition of, 228
- Bounded buffer, 508
- break*, 367, 388
- Brinch-Hansen, P., 510
- Bronte, Charlotte, 599
- C*, 315
- C*, 196, 197, 201, 203, 259, 296, 314, 315, 327, 356, 367, 420, 424, 427, 436, 440, 454, 456, 520, 523, 524, 556, 562, 637, 748, 763
- C++, 314, 420, 436, 453
- C#, 314
- Caffeine, 393
- call*, 198, 276
  - denotational semantics, 253, 358, 440
  - CBD, 276
  - CBL, 360
  - CBN, 268, 359
  - CBR, 360
  - CBV, 268, 359
  - dynamic scoping, 282
  - standard, 381
  - static scoping, 282
  - free and bound identifiers, 228
  - operational semantics, 241, 339
  - CBN, 260
  - CBV, 260
  - substitution in, 236
- Call-by-denotation (CBD), 275–278
- Call-by-eager (CBE), 502
- Call-by-lazy (CBL), *see* Call-by-need
- Call-by-name (CBN), 259, 277–279, 361
  - denotational semantics of, 267–269
  - operational semantics of, 259–266
- Call-by-need (CBL), 361
  - compared to Call-by-eager, 502
- call-by-need (CBL), 502
- Call-by-reference (CBR), 362
- Call-by-value (CBV), 259, 277–279, 359

- and recursive definitions, 272–274, 279
- denotational semantics of, 267–269
- operational semantics of, 259–266
- Call-by-value-copy (CBVC), 438
- Call-by-value-sharing (CBVS), 438
- `call-with-current-continuation`, 399, 401, 662, 663
- `call/cc`, *see* `call-with-current-continuation`
- Cantor, 772
- Capture, 258
  - of a variable, 233, 566
    - external, 234
    - in call-by-denotation, 276
    - internal, 233
- capturing-cont*, 387
- `car`, 216, 217
- Cardelli, Luca, 550
- Cardinality, 772
- Carrier, 628
- Cartesian product ( $\times$ ), 773, 791
- `catch`, 367, 399, 402
- CBD, *see* Call-by-denotation
- CBE, *see* Call-by-eager
- CBL, *see* Call-by-need
- CBN, *see* Call-by-name
- CBV, *see* Call-by-value
- CBVC, *see* Call-by-value-copy
- CBVS, *see* Call-by-value-sharing
- CCS, 510
- `cdr`, 216, 217
- `cell`, 327, 328, 335, 655
  - denotational semantics, 347
    - standard, 381
  - operational semantics, 336
- `cell-ref`, 327–329, 335, 337
  - denotational semantics, 347
    - standard, 382
  - operational semantics, 336
- `cell-set!`, 327–329, 335, 357
  - denotational semantics, 347
    - standard, 382
  - operational semantics, 336
- `cell=?`, 327, 328
  - denotational semantics, 347
  - operational semantics, 336
- `cell?`, 327, 328
  - denotational semantics, 347
  - operational semantics, 336
- Channel, 507–510
- `channel`, 507
- `channel`
  - operational semantics, 510
- `channel?`, 508
- check-boolean*, 344, 386
- check-location*, 344, 386
- check-procedure*, 344, 386
- check-quota*, 394
- `choose`, 494
- `choose`, 490, 494
- Church, 212
- Church list, 212
- Class, 302
- `class`, 306, 311
  - desugaring in HOOPLA, 307
- Clause, 795
- Closed expression, 226
- Closed procedure, *see* Closure
- Closure
  - of a relation, 775
  - reflexive transitive, 188
  - transitive, 775
- Closure conversion, 748
  - closure passing style, 750
  - code/env pairs, 749
  - defunctionalization, 758
  - lightweight, 758
  - selective, 756
- Closure passing style, 750

- Closures, 748
  - flat, 748–754
- CLU, 367, 402, 423, 424, 436, 438, 638, 648, 651
- Coalesced sum, 178
- cobegin, 501
- COBOL, 314
- Code bloat, 732
- Code component of a configuration, 39
- Code/env pairs, 749
- Coercion
  - implicit, 555
- colet, 501
- combine-env, 299
- comefrom, 663
- Command, 382
  - in POSTFIX, 6
- Comment, 770
- Commingling, unholy, 629
- Commit, 352
- commit!, 352
- commof, 597
- COMMON LISP, 196, 294, 315, 367, 399, 402, 464
- Communicating sequential processes (CSP), 510
- Communication, 503
- Commutative, 658
- Compilation, 305, 673
- Compile time, 513, 637
- Complete partial order (CPO), 174
- Component value, 270
- Composition
  - of functions ( $\circ$ ), 779
  - of Relations, 775
- Compound domain, 791
- Compound expression, 18
- Compound phrase, 25
- Compound syntactic domain, 23
- Computable function, 777
- Computation domain
  - in imperative languages, 343–346
- conceal, 297, 298, 428
  - denotational semantics, 300
- Concrete grammar, 21
- Concrete syntax, 20–21
- Concrete syntax tree (CST), 21
- Concurrency, 489–511
  - channel, 507–510
  - lock, 505–507
  - operational semantics, 495–498
- Concurrent, 490, 491
- CONCURRENT OBJECTS, 511
- cond, 204
  - desugaring in FL, 209
- Configuration
  - code component, 39
  - irreducible, 42
  - reducible, 42
  - state components, 39
- Configuration in SOS, 39
- Confluence, 75
  - one-step, 75
- Connection machine, 511
- cons, 216, 217
- cons, 797–798
- cons-stream, 434
- Consequent, 18
- Constant, 802
  - declaration, 357
- Constant function, 779
- Constraint
  - type, 584
- Constraint set, 665
- Constructor, 473
  - deconstructor for, 458
  - value, 458
- Constructor function, 803
- consume, 391, 392

- Consumer, 378, 390–392
- `consumer`, 392
- Consumer/producer coroutines, 378
- Contagious, 52
- Content of a mutable cell, 327
- Context, 64
  - POSTFIX command sequence, 84
  - POSTFIX evaluation, 66
  - POSTFIX program, 84
  - control, 366
  - evaluation, 64
  - hole, 64
  - naming, 365
  - observational equivalence and, 83
  - state, 365
- Context domain, 109, 122
- Continuation, 367, 378, 414, 458, 474, 475, 480
  - CPS conversion, 718–748
  - domain, 382
  - effects of, 662, 663
  - first-class, 398
  - multiple-value return, 369–371
  - normal, 367
  - procedural, 368–378
  - receiver, 369
- Continuation passing style, 379, 415
- Continuation passing style (CPS), 117
- `continue`, 367, 388
- Continuous function, 178
- Contract, 599
- Control context, 366
- Control point, 395
- Conway, Melvin, 415, 510
- Coroutine, 367, 378, 415, 510
  - producer/consumer, 378
- `coroutine`, 389, 390
- `count-from`, 378
- Countable, 772
- CPO, *see* Complete partial order
- CPS conversion, 718–748
- CSP, 510
- CST (concrete syntax tree), 21
- cummings, e e, 17
- Curried functions, 201, 781, 787
- Curried procedures, 635
- Curry, 598
- Curry, Haskell, 781
- Currying, 205
- `cwcc`, *see* `call-with-current-continuation`
- DAG, *see* Directed acyclic graph
- Data Abstr action
  - secure, 602
- Data abstraction, 599
  - abstraction barrier, 599
  - API, 599
  - client, 599
  - contract, 599
  - implementor, 599
  - interface, 599
  - invariant, 602
  - violation, 601
- Data declarations, 456–464
- Data dependency, 321
- Data type, *see* Type
- Deadlock, 494, 496, 506
- `decon`, 486
- Deconstructor, 458, 464
- deepCopying*, 440
- default-handlers*, 405
- `define`, 204, 255, 257, 306, 461, 640
- `define-constructor`, 485, 486
- `define-data`, 456, 461
  - desugaring, 461
- `define-datatype`, 640
- `define-desc`, 569, 573
- `defstruct` in COMMON LISP, 464
- Defunctionalization, 758
- delay, 434

- den-to-comp*, 252
- Denotable value, 249, 258, 270
- Denotational adequacy, 149, 151
- Denotational semantics, 13, 109
  - direc, 379
  - for FL, 248–255
  - not to be interpreted operationally, 274–275
  - standard, 379
- Denotational soundness, 145
- Denote, 109
- Dependent package, 629
- Dependent procedure, 631
- Dependent type, 629
  - static, 634
- depth\*sum<sub>1</sub>*, 369
- depth\*sum<sub>2</sub>*, 370
- depth\*sum<sub>3</sub>*, 370
- Dequeue, 507
- Dereferencing a variable, 358
- Derivation, 57
- derivative*, 289
- Description
  - equivalence, 574
- Destroying a thread, 501
- Desugaring, 195, 205
- Determinism, 72
- Deterministic, 489
- Deterministic transition relation, 42, 43, 50
- Diagonalization, 772
- Diamond property, *see* Confluence
- dict-adjoin-binding*, 225
- dict-bind*, 225
- dict-empty*, 225
- dict-empty?*, 225
- dict-first-binding*, 225
- dict-lookup*, 225
- dict-rest-bindings*, 225
- Difference
  - of environments, 428
- Difference of sets ( $-$ ), 771
- Dijkstra, E. W., 510
- Direct semantics, 366, 379
- Directed acyclic graph (DAG), 230
- Discrete partial order, 167
- Discriminant, 443, 465, 795
- Discriminated union, *see* Sum
- Disjoint sets, 772
- Disjoint union, 793
- dlambda*, 572
- do-while*, 385
- Domain, 769, 790–802
  - answer, 109
  - compound, 791
  - constructor, 790
  - context, 109, 122
  - definition, 792, 802
  - equality, 793
  - expression, 802
  - function, 798–802
  - lifted, 167
  - of denotable values, 249
  - primitive, 790, 802
  - product, 418
  - product ( $\times$ ), 791–793
  - reflexive, 192
  - sequence, 796–798
  - sum, 442, 793–796
  - syntactic, 23
  - tuple, 791
  - variable, 799
- Dorough, Bob, 417
- Dragging tail, 265
- dselect*, 640
- Dual, 553, 653
- dylambda*, 295
- DYLAN, 367, 399
- dylet*, 295
- DYNALEX, 295



- Dynamic environment, 289
- Dynamic property, 513
- Dynamic scope, 282
- Dynamic scoping, 281–293
- Dynamic semantics, 513
- Dynamic type, 517, *see* Type
- dyref*, 295
  
- Eager evaluation, 502–503
- ecase*, 446
- Effect, 258, 315, 349–350, 653–668
  - comefrom*, 663
  - goto*, 664
  - control, 663–664
  - erasure, *see* Effect, masking
  - init*, 655
  - latent, 654
  - masking, 660–662, 664
  - polymorphic, 656
  - reconstruction, 665–668
  - region, 654
  - store, 655
  - system, 653–668
- Effect system, 653
- Either, 446
- EL, 18–31, 55, 56, 60, 66, 72, 73, 76, 82, 90–92, 107, 110, 120–122, 124, 128, 135, 139–141, 143, 145, 149–151, 196, 199, 219, 221, 461
  - deterministic behavior, 73–76
- elect*, 393
- Element expression, 802
- Element of, 770
- ELM, 60, 66, 67, 69, 70, 76, 92, 118–122, 124, 149, 219, 221, 450, 452–455, 460, 467, 468
- elm-eval*, 219, 221, 467
- ELMM, 56–60, 64–66, 68–70, 73–76, 91, 111–119, 122, 123, 149, 151
  - 149, 151
- else*, 447
- Emerson, Ralph Waldo, 195
- Empty set, 770
- empty-env*, 250
- empty-store*, 342
- empty-tenv*, 450
- empty?*, 797–798
- Energy, 77–78
- Enqueue, 507
- ensure-assigned*, 380
- ensure-bound*, 380
- Environment, 248, 249, 748
  - as a model for product data, 418
  - call-time, 285
  - diagram, 285
  - dynamic, 285, 289
  - lexical, 289
  - type, 528–529, 591, 592
  - value, 528
- Environment conversion, *see* Closure
  - conversion
- equal?*, 216, 217
- Equality
  - on domains, 793
  - on functions, 776
  - on sets, 771
- Equational proof, 113
- Equational reasoning, 111
- Equi-recursive type equality, 548
- Equivalence
  - of descriptions, 574
- Equivalence class, 774
- Equivalence relation, 232, 539, 774
- Eratosthenes
  - sieve of, 433
- ERLANG, 197
- err-to-comp*, 252, 344, 386, 405
- Error, 44–45, 51–52

- error, 198, 247, 255, 344, 384, 561, 566
  - denotational semantics, 253
  - standard, 381
- error-comp*, 252, 344
- error-cont*, 380
- Escape procedure, 399
- Eta rule, 247
- Evaluation context, 64
- Evaluation relation, 68
- Evaluation tree of a big-step operational semantics, 68
- except when**, 402
- Exception, 367, 399, 402–414
  - handle, 402
  - handling, 402
  - raise, 402
  - resumption semantics, 402
  - signal, 402
  - termination semantics, 402
- exec**, 33, 34
- Existential package, 609
- Existential type, 608–619
  - export restriction, 613, 616
  - import restriction, 612, 616
- Expansive, 649
- Explicit type, *see* Type
- Export restriction, 613, 616
- Exported names, 281
- Exports, 637
- expr-to-comp*, 344, 386
- Expression, 382
  - Impure, 657
  - language, 383
  - Pure, 657
- Expressive, 378
- Expressive power, 515, 516, 562
- Expressive type system, 519
- extend-env*, 250
- extend-env\**, 299
- extend-handlers*, 405
- extend-tenv*, 450
- extend-tenv\**, 450
- extending-handlers*, 405
- Extensionality, 113
- External variable capture, 234
- extract-value*, 273, 347, 385
- fail**, 223
- failed?**, 223
- Failure, 376
- Failure continuation, 458, 474, 475
- Failure thunk, 469
- false**, 216, 217
- FDV, 667
- Felleisen, Matthias, 104
- fetch*, 342
- fetching*, 345, 387, 431
- FF, 53–54
- Fibonacci number, 94
- Final configuration of an operational semantics, 40, 42
- finally**, 409
- First-class procedure, 197
- First-class procedures, 122
- first-fresh*, 342
- First-In/First-Out (FIFO), 507
- Fixed point, 159, 182, 272, 343
  - iterative technique for finding, 160–165
  - least, 164
  - theorem of least, 182
- FL, 383
- FL, 195–255, 257, 259, 261, 262, 265–267, 270, 276–279, 281, 283, 285, 290–293, 295, 297, 302, 305, 307, 309–311, 314–317, 319–321, 326–328, 338, 341, 349, 356, 359, 361, 367–369, 374, 378, 384, 396, 400, 419,

- 423–426, 428, 431, 443, 446,
- 456, 459, 463–465, 467, 468,
- 470, 475, 485, 486, 503, 513,
- 515, 519–523, 525, 533, 536,
- 538, 543, 545, 546, 561, 586,
- 589, 600, 603, 669, 673–675,
- 678, 680, 682, 697, 698, 700,
- 732, 790, 832, 841
- denotational semantics, 248–255
- of Backus, 196
- f1
  - desugaring in FL, 210
- FL\*, 533–535
- FL/R, 586–590, 592–594, 596, 597,
- 608, 626, 627, 635, 638, 640,
- 655, 657–661, 663–665, 668,
- 675–680, 682, 688, 693, 694,
- 696–702, 705–708, 710, 713
- FL/RM, 638, 640–642, 644, 647, 649
- FL/X, 519–527, 529–545, 547, 548,
- 550–552, 554, 558, 559, 562,
- 567, 569, 583, 586, 589, 607,
- 608
- FL/XS, 552–561, 563
- FL/XSP, 563–567, 569, 573, 577,
- 580, 608, 610, 618, 620, 621,
- 623, 625, 627, 630
- FL/XSPD, 569, 571–577
- FL/XSPDK, 576–581
- FLAT, 292
- Flat closures, 748–754
- FLK, 197–199, 201–205, 208–213, 215,
- 216, 224, 226–230, 232, 234,
- 237, 239–255, 257, 258, 262,
- 265, 266, 268, 269, 272, 275,
- 278, 279, 281, 284, 285, 291,
- 293, 295, 317, 327, 335, 338–
- 341, 346–348, 351, 356, 382,
- 394, 413, 497
- informal semantics, 199–204
  - syntax, 197–199
- Flow analysis, 768
- FLUID, 290–292
- for, 366, 385
- forall, 563
- force, 434
- fork, 496, 501
- fork, 490
- fork, 493, 494, 501, 502
  - operational semantics, 497
- Formal parameter, 224, 784
- FORTH, 5
- FORTRAN, 362
- FORTRAN, 250, 270, 314, 356, 362,
- 436, 520, 835, 843
- FP, 196
- Frank, Michael, 93
- Free identifier, 226, 319, 566, 592
  - definition of, 228
- Fresh identifier, 237
- fresh-loc, 342
- Friedma, Dan, 104
- Frost, Robert, 365
- fst, 217
- Full abstraction, 150
- Full language, 195
- Function, 769
  - application, 777
  - bijjective, 780
  - composition ( $\circ$ ), 779
  - computable, 777
  - constant, 779
  - curried, 781, 787
  - different from procedure, 776–
  - 777
  - equal ( $=$ ), 776
  - graph, 775
  - higher-order, 780–781
  - identity, 779, 786
  - image of, 780

- inclusion, 779
- injective, 780
- lambda notation, 784–790
- one-to-one, 780
- operand, 777
- operator, 777
- partial, 777
- polymorphic, 786
- recursive, 787–788
- set theoretic, 775–790
- signature, 776
- simulating multiple arguments, 781
- simulating multiple return values, 783
- surjective, 780
- total, 274, 777
- type, 776
- Function domain, 798–802
- Function-oriented language, 196
- Functional programming language, 196, 777, 789
- Functions
  - elements of function domain, 798
  - higher order, 748
  - nested, 748
- Future, 502
- future, 502
- FV, 667
- FX, 788
- FX, x, 197, 518, 591, 598, 648, 788, 836, 843
- Garbage collection, 364, 765–768
  - reading, 768
  - replication-based, 768
- gaussian-elimination, 296
- Generating function, 159
- generic, 589, 665
- get-arg, 221, 467
- get-handler, 405
- getting-handler, 405
- Gifford, David, 652
- Girard, 581
- Global scoping, 292
- go, 597
- goto, 367, 399
- goto, 664
- Grammar
  - s-expression, 109
- Graph coloring, 768
- Graph of a function, 775
- Graph rewriting system, 105
- Greatest lower bound (glb), 167
- Halting function, 190, 777
- Halting problem, 41, 190, 514, 777
- Halting theorem, 41
- Hamming numbers, 434
- handle, 558
- handle, 402, 407–409, 411, 413, 414
- Handle an exception, 402
- Handler procedure, 403
- HASKELL, 122, 123, 197, 201, 259, 315, 319, 326, 423, 424, 428, 453, 463, 464, 480, 482, 517–519, 522, 591, 598
- Hasse diagram, 166
- Hawes, Bess, 365
- head, 434
- head, 797–798
- Heap, 364
- Heidegger, Martin, 769
- Heterogeneous lists, *see* List
- Hewlett Packard, 5
- Hiding names, 281
- Hierarchical scope, 281–293, 296
- Higher-order function, 780–781
- Higher-order functions, 748
- Hindley, 589, 598

- Hindley-Milner type reconstruction, 589
- Hoare, C. A. R., 510
- Hole in the scope of a variable, 229, 284
- Homogeneous lists, *see* List
- Homomorphism, 110
- HTML, 454
  
- I-structure, 510
- ID
  - I-structure, 510
- ID, 502, 510, 511, 837, 843
- Idempotent, 213
- Identifier, 224, 281
  - binding occurrence, 226, 319
  - bound, 226
  - denotational semantics, 253, 358
    - CBD, 276
    - CBL, 360
    - CBN, 359
    - CBR, 360
    - CBV, 359
  - standard, 381
  - free, 226, 319
  - fresh, 237
  - occurrence of, 226
- Identity function, 779, 786
- Identity of an object, 313
- Idiom, 378
- if** as sugar for **matching**, 795
- if**, 198, 306
  - denotational semantics, 253
    - standard, 381
  - desugaring in HOOPLA, 307
  - free and bound identifiers, 228
  - operational semantics, 241, 339
  - substitution in, 236
- if*, 782
- Ill-typed, 530
  
- Image
  - of a function, 780
- impeach**, 393
- Imperative programming paradigm, 326, 329
- Implicit projection, 564
- Implicit type, *see* Type
- Import restriction, 612, 616
- Imported names, 280
- Imports, 637
- Impure, 657
- Inclusion function, 779
- Incomparable elements in a a partial order, 166
- Induction, 787
  - structural, 81, 227
- Inference of types, *see* Type, reconstruction
- Inheritance hierarchy, 302
- init**, 655
- Initial configuration of an operational semantics, 39
- inj**, 443, 446
  - denotational semantics, 445
  - operational semantics, 445
- Injection function, 793
- Injective function, 780
- inleft**, 446
- Inlining, 701, 730
- Input function of an operational semantics, 39, 43
- inright**, 446
- install-cont*, 387
- Int*, 770
- int-tree**, 572
- integer?**, 217
  - denotational semantics, 254
  - operational semantics, 243
- Interface, 280, 599, 637, 790
- Interference, 349

- Interleaving, 491, 504–506
- Internal variable capture, 233
- Intersection
  - of environments, 428
- Intersection ( $\cap$ ), 771
- `ints-from`, 433
- Invariant
  - representation, *see* Data abstraction, invariant
- Inverse limit construction, 192
- Irreducible configuration, 42
- Iso-recursive type equality, 547
- Iteration, 320, 330
- Iterative fixed point technique, 160–165
- JAVA, 196, 197, 201, 203, 314, 367, 424, 427, 436, 438, 453, 517, 518, 523, 533, 555, 763
- JCSP, 367, 415
- `join`, 496, 501
- join*, 490
- `join`, 493, 494, 501, 502
  - operational semantics, 497
- `jump`, 395, 396, 398–401, 407
  - denotational semantics, 397
- Kahn, Gilles, 105
- kernel, 802
- Kernel of a programming language, 195
- Kind, 577
- Kinds, 576–581
- Kingston Trio, 365
- `knull`, 485
- `kons`, 485
- L-value, 358, 363
- `label`, 395, 396, 398–401, 407
  - denotational semantics, 397
- `lambda`, 204, 281, 306, 311
  - desugaring in FL, 209
  - desugaring in HOOPLA, 307
- Lambda abstraction, 784
- Lambda calculus, 262
- Lambda lifting, 763
- Lambda notation, 784–790
  - recursion, 787–788
- Landin, Peter, 104
- Language
  - mostly functional, 315
  - purely functional, 315
- Latent effect, 654
- Lazy
  - product, 430
- `lazy`, 266, 434
- Lazy evaluation, 265, *see* Call-by-need, 502
- `least`, 246
- Least fixed point, 164
  - theorem, 182
- Least upper bound ( $\sqcup$ ), 595
- Least upper bound (lub), 167
- `left`, 246, 270
  - denotational semantics, 254
  - operational semantics, 243, 339
- Left hand side (LHS) of a transition, 42
- `length`, 216, 218, 220
- length*, 797–798
- `let`, 204, 255, 257, 281, 306, 311
  - desugaring in FL, 209
  - desugaring in HOOPLA, 307
- `letrec`, 204, 255, 257, 266, 281
  - desugaring in FL, 209
- Lexical environment, 289
- Lexical scope, 282
- Lifted domain, 167
- Lifting, 763
- Lightweight closure conversion, 758
- LINDA, 511

- Link time, 637
- Linking, 637
- LISP, 778
  - not lambda notation, 788–790
- LISP, 2, 22, 36, 197, 199, 205, 206, 212–214, 274, 284, 428, 453, 456, 502, 517, 839, 843
- List, 418
  - association, 418, 428
  - heterogeneous, 545
  - homogeneous, 543
- list, 204
  - desugaring in FL, 209
- list-map, 573
- list?, 218, 220
- listen, 597
- listof, 573
- lit-num, 219, 221
- lit?, 219, 221
- Literal
  - denotational semantics, 253
  - standard, 381
- load, 640, 647
- Location, 326, 332
- Lock, 505–507
- lock, 505, 603
  - operational semantics, 507
- lock?, 505
- Logic programming, 367, 510, 591
- login!, 394
- logout!, 394
- lookup, 250
- loop, 385–388
- Loophole, 456
- Lower bound, 167
- lproduct, 430, 436
  - denotational semantics, 432
  - operational semantics, 431
- lproj, 430, 436
  - denotational semantics, 432
  - operational semantics, 431
- Mann, Thomas, 1
- map-type, 573
- Match
  - clause, 465, 795
  - body, 795
  - head, 795
  - s-expression pattern, 25
- match, 464–480
  - body, 465
  - clause, 465
  - desugaring, 468–480
  - discriminant, 465
  - pattern, 465
- match, 475, 478
- match!, 332
- match-inner, 376, 377
- match-sexp, 223, 332, 375–377
- match-with-dict, 223, 332, 374–376
- matching**, 794
- matrix-invert, 296
- Meaning function, 109
- member?, 218, 220
- Memoization, 265, 430, 431, 434, 493, 502
- Memoize, 430
- merge, 218
- merge-sort, 218
- MESA, 651
- Meta-application, 736
- Meta-continuation, 735
- Meta-rules in operational semantics, 496
- Metalanguage, 13, 769–809
- method, 306, 310, 311
  - desugaring in HOOPLA, 307
- mfst, 351
- mget, 437
  - denotational semantics, 439

- Milner, Robin, 510, 589, 591, 592, 594, 595, 598
- Mini-language, 5
- MIRANDA, 197, 259, 315, 518
- Mitchell, John, 550
- Mixed expression, 335
- ML, 122, 123, 197, 201, 259, 315, 367, 398, 402, 423, 424, 427, 437, 453, 463, 464, 475, 480, 482, 483, 511, 515, 517–519, 591, 598, 768
- Modelling, 492
- Modularity, 491
- Module, 296–302, 418, 636–651
  - first-class, 638
  - second-class, 638
- module, 640
- moduleof, 640
- Modules
  - exports, 637
  - imports, 637
  - linking, 637
- Monad, 326
- Monadic style, 117, 324, 345, 348
- Monadic-style, 324–326
- Monitor, 510
- Monomorphic type, *see* Type
- Monotonic
  - subtype, 553
- Monotonic function, 178
- Moon Microsystems, 394
- Morris, F. Lockwood, 414
- Mostly functional language, 315
- mpair, 351
- mprod, 437, 438
  - denotational semantics, 439
- mset!, 437
  - denotational semantics, 439
- msnd, 351
- MULTI-LISP, 511
- MULTI-SCHEME, 511
- Multi-threaded, 490, 491, 493–498
- Multiple namespaces, 294
- Multiple-value return, 369–371
- Mutable, 327
- Mutable cell, 327
- Mutable pair, 351
- Mutable variable, 356–357
- Mutation, 315
- n*-tuple, 773
- Name capture, 258
- Name control, 258, 281
- Name hiding, 281
- Named product, *see* Product, named
- Namespace, 258, 524
- Naming context, 365
- Nat*, 770
- Natural semantics, *see* big-step operational semantics (BOS)
- NAVAL, 278
- Neg*, 770
- Nested functions and objects, 748
- new-key, 603
- next-location, 342
- nil, 216, 217
- Non-determinism, 490, 494, 498
  - in operational semantics, 496
- Non-deterministic transition relation, 42, 43
- Non-hierarchical scope, 296–302
- Non-local exit, 367, 371, 395–401
- Non-strict, 201
  - pairs, 270
  - product, 428
- Non-strict function, 190
- Non-tail call, 720
- Nonce type, 619–628
- Nondeterminism, 367
- Normal continuation, 367



- Normal form, 262, 548
- Normal order reduction, 262
- not?, 217
  - denotational semantics, 254
  - operational semantics, 243
- nproc, 278
- nproduct, 428
  - denotational semantics, 429
  - operational semantics, 429
- nproj, 428
  - denotational semantics, 429
  - operational semantics, 429
- nth, 797–798
- null, 217
- Null pointer, 453
- null-object, 306, 310
- null?, 216, 217
- Nullary procedure, 215, 316
  
- O’Toole, James, 652
- Object, 302, 427
- object, 306
  - desugaring in HOOPLA, 307
- object-compose, 306, 310, 311
  - desugaring in HOOPLA, 307
- Object-oriented programming, 302–312
- Observable action, 491
- Observable properties, 319
- Observational equivalence, 82–85
  - in POSTFIX, 82–92
- OCAML, 638
- OCCAM, 367, 415
- Occurrence of an identifier, 226
- one, 447
  - denotational semantics, 451
  - operational semantics, 449
- One step transition, 42
- One-to-one correspondence, 780
- Oneof, *see* Sum, 447, 793
  
- open, 640
- Operand, 777, 802
- Operational
  - final configuration, 40
- Operational execution, 50–54
- Operational semantics, 13, 37–104
  - answer, 40
  - axiom, 46–50
  - behavior, 43–44
  - error, 44–45, 51
  - evaluation context, 64
  - final configuration, 42
  - for FL, 239–248
  - initial configuration, 39
  - input function, 39, 43
  - output function, 40, 43
  - progress rule, 46, 54–64
  - redex, 64
  - rewrite rule, 46
  - stuck state, 41, 42
  - transition path, 42
- Operational semantics
  - error, 52
- Operator, 777, 802
- or, 204
- or?, 217
  - operational semantics, 243
- Output function of an operational semantics, 40, 43
- override, 297, 428
  - denotational semantics, 300
  
- Package, 608
  - dependent, 629
  - existential, 609
- Pair, 773
  - mutable, 351
- pair, 198, 246, 270
  - denotational semantics, 253
  - non-strict, 271

- standard, 381
  - strict, 271
- operational semantics
  - non-strict, 271
  - strict, 271
- substitution in, 236
- `pair?`, 217, 246
- Pairwise disjoint, 772
- Parallelism, 491, 492, 502
- Parameter
  - formal, 224
- Parameter passing, 258–270, 502
  - call-by-eager (CBE), 502
  - Call-by-name (CBN), 259, 361
  - Call-by-need (CBL), 361
  - Call-by-reference (CBR), 362
  - Call-by-value (CBV), 259, 359
  - in languages with mutable variables, 359–364
- Parse, 21
- Parser, 378
- Partial function, 777
- Partial order, 166, 490
  - complete, 174
  - discreet, 167
  - pointed, 176–272
- Partition, 232, 772, 774
- PASCAL, 362
- PASCAL, 519
- PASCAL, 14, 197, 201, 259, 270, 296, 314, 327, 343, 356, 362, 383, 423, 424, 427, 436, 440, 456, 517–520, 523, 562, 748, 758, 763, 842, 843
- Passable value, 270
- Pattern matching, 219, 332, 374, 464–487
- `pattern-constant?`, 223
- `pattern-variable-name`, 223
- `pattern-variable?`, 223
- `pcall`, 563
- PEBBLE, 576, 633
- Perform, 324
- Phase distinction, 632
- Phrase tag, 28
- Pierce, Benjamin, 550
- Pivot, 484
- `plambda`, 563
- Plotkin, Gordon, 104
- `point`, 308
- Pointed partial order, 176–343
- Polymorphic, 656
- Polymorphic function, 786
- Polymorphic type, *see* Type, polymorphic
- Polymorphism, *see* Type, polymorphic
- `pop`, 33, 34
- Porter, Cole, 417
- Pos*, 770
- POSTFIX, 498
- POSTFIX, x, 5–13, 32–35, 37–41, 44–47, 49–54, 56, 60, 61, 63, 64, 66, 67, 70, 71, 73, 77–88, 90–104, 107, 108, 124, 125, 127–133, 135–141, 143–146, 148–152, 192, 193, 195, 196, 199, 204, 206, 242, 246, 498, 830, 831, 841–843, 848, 850
  - deterministic behavior, 50
  - syntax, 32–35
  - termination, 77–82
- POSTFIX2, 32–35, 52, 53, 80, 135, 826, 843
- POSTHEAP, 102–104
- POSTLISP, 101–102, 292–293
- POSTLOOP, 94–95
- POSTMAC, 99–101
- POSTSAFE, 95–96
- POSTSAVE, 97–98

- POSTSCRIPT, 5, 259
- POSTTEXT, 98, 99, 101, 826, 843
- Powerdomain, 492
- Powerset, 44, 772
- Pragmatics, 2, 4–5
- prefix, 433
- primes, 433
- Primitive domain, 790, 802
- Primitive operator, 18
- Primitive set elements, 770
- Primitive syntactic domain, 23
- primop, 198, 216
  - denotational semantics, 253
  - standard, 381
  - free and bound identifiers, 228
  - operational semantics, 241, 339
  - substitution in, 236
- primop->proc, 219, 221, 467
- Principal type, 584
- proc, 198, 224, 257, 278, 282, 289, 410
  - denotational semantics, 253
  - CBN, 267
  - CBV, 267
  - dynamic scoping, 282
  - standard, 381
  - static scoping, 282
  - free and bound identifiers, 228
  - operational semantics, 241, 339
  - CBN, 260
  - CBV, 260
  - substitution in, 236
- Procedural continuation, 368–378
- Procedure, 197
  - dependent, 631
  - different from function, 776–777
- procedure?, 217
- Process algebra, 511
- prod-list, 396, 399
- Producer, 378, 390–392
- producer, 391, 392
- Producer/consumer coroutines, 378
- Product, 418
  - call-by-name, 428
  - call-by-value, 428
  - lazy, 430
  - mutable, 436–442
  - named, 426–428
  - non-strict, 428–436
  - of domains (  $\times$  ), 791–793
  - of sets (  $\times$  ), 773
  - positional, 419–426
  - strict, 428
- product, 419, 420
  - denotational semantics, 421
  - operational semantics, 420
- Product domain, 418
- product-of-list<sub>1</sub>, 374
- product-of-list<sub>2</sub>, 374
- product-of-list<sub>3</sub>, 375
- Production, 25
- Production rule, 23
- Program
  - in POSTFIX, 6
- program, 204, 306, 461
- Programming language
  - C, 315
  - ID
    - I-structure, 510
  - FL!, 498
  - FORTRAN, 362
  - FX, 788
  - LISP, 778
  - PASCAL, 362
  - POSTFIX, 498
  - POSTFIX2, 52
  - POSTTEXT, 98
  - SCHEME, 315, 357
  - actor, 415
  - ADA, 196, 314, 517, 748, 758

- ALGOL 60, 259
- APL, 284, 517
- assembly-level, 516
- BASIC, 270, 518
- block structured, 283, 298
- C, 196, 197, 201, 203, 259, 296,
  - 314, 315, 327, 356, 367, 420,
  - 424, 427, 436, 440, 454, 456,
  - 520, 523, 524, 556, 562, 637,
  - 748, 763
- C++, 314, 420, 436, 453
- C#, 314
- CCS, 510
- CLU, 367, 402, 423, 424, 436,
  - 438, 638, 648, 651
- COBOL, 314
- COMMON LISP, 196, 294, 315,
  - 367, 399, 402, 464
- concurrent, 490, 491
- CONCURRENT OBJECTS, 511
- CSP, 510
- DYLAN, 367, 399
- DYNALEX, 295
- EL, 18–31, 55, 56, 60, 66, 72,
  - 73, 76, 82, 90–92, 107, 110,
  - 120–122, 124, 128, 135, 139–
  - 141, 143, 145, 149–151, 196,
  - 199, 219, 221, 461
  - deterministic behavior, 73–76
- ELM, 60, 66, 67, 69, 70, 76, 92,
  - 118–122, 124, 149, 219, 221,
  - 450, 452–455, 460, 467, 468
- ELMM, 56–60, 64–66, 68–70, 73–
  - 76, 91, 111–119, 122, 123,
  - 149, 151
- ERLANG, 197
- expressive, 378
- FF, 53–54
- FL, 383
- FL, 195–255, 257, 259, 261, 262,
  - 265–267, 270, 276–279, 281,
  - 283, 285, 290–293, 295, 297,
  - 302, 305, 307, 309–311, 314–
  - 317, 319–321, 326–328, 338,
  - 341, 349, 356, 359, 361, 367–
  - 369, 374, 378, 384, 396, 400,
  - 419, 423–426, 428, 431, 443,
  - 446, 456, 459, 463–465, 467,
  - 468, 470, 475, 485, 486, 503,
  - 513, 515, 519–523, 525, 533,
  - 536, 538, 543, 545, 546, 561,
  - 586, 589, 600, 603, 669, 673–
  - 675, 678, 680, 682, 697, 698,
  - 700, 732, 790, 832, 841
  - denotational semantics, 248–
  - 255
  - of Backus, 196
- FL\*, 533–535
- FL/R, 586–590, 592–594, 596,
  - 597, 608, 626, 627, 635, 638,
  - 640, 655, 657–661, 663–665,
  - 668, 675–680, 682, 688, 693,
  - 694, 696–702, 705–708, 710,
  - 713
- FL/RM, 638, 640–642, 644, 647,
  - 649
- FL/X, 519–527, 529–545, 547,
  - 548, 550–552, 554, 558, 559,
  - 562, 567, 569, 583, 586, 589,
  - 607, 608
- FL/XS, 552–561, 563
- FL/XSP, 563–567, 569, 573, 577,
  - 580, 608, 610, 618, 620, 621,
  - 623, 625, 627, 630
- FL/XSPD, 569, 571–577
- FL/XSPDK, 576–581
- FLAT, 292
- FLK, 197–199, 201–205, 208–213,
  - 215, 216, 224, 226–230, 232,
  - 234, 237, 239–255, 257, 258,

- 262, 265, 266, 268, 269, 272,  
275, 278, 279, 281, 284, 285,  
291, 293, 295, 317, 327, 335,  
338–341, 346–348, 351, 356,  
382, 394, 413, 497
- informal semantics, 199–204
- syntax, 197–199
- FLUID, 290–292
- FORTH, 5
- FORTRAN, 250, 270, 314, 356,  
362, 436, 520, 835, 843
- FP, 196
- full, 195
- FX, x, 197, 518, 591, 598, 648,  
788, 836, 843
- HASKELL, 122, 123, 197, 201, 259,  
315, 319, 326, 423, 424, 428,  
453, 463, 464, 480, 482, 517–  
519, 522, 591, 598
- HTML, 454
- ID, 502, 510, 511, 837, 843
- idiom, 378
- JAVA, 196, 197, 201, 203, 314,  
367, 424, 427, 436, 438, 453,  
517, 518, 523, 533, 555, 763
- JCSP, 367, 415
- kernel of, 195
- LINDA, 511
- LISP, 2, 22, 36, 197, 199, 205,  
206, 212–214, 274, 284, 428,  
453, 456, 502, 517, 839, 843
- logic, 367, 510, 591
- MESA, 651
- mini, 5
- MIRANDA, 197, 259, 315, 518
- ML, 122, 123, 197, 201, 259, 315,  
367, 398, 402, 423, 424, 427,  
437, 453, 463, 464, 475, 480,  
482, 483, 511, 515, 517–519,  
591, 598, 768
- MULTI-LISP, 511
- MULTI-SCHEME, 511
- multi-threaded, 493–498
- NAVAL, 278
- object-oriented, 302–312
- OCAML, 638
- OCCAM, 367, 415
- PASCAL, 519
- PASCAL, 14, 197, 201, 259, 270,  
296, 314, 327, 343, 356, 362,  
383, 423, 424, 427, 436, 440,  
456, 517–520, 523, 562, 748,  
758, 763, 842, 843
- PEBBLE, 576, 633
- POSTFIX, x, 5–13, 32–35, 37–  
41, 44–47, 49–54, 56, 60, 61,  
63, 64, 66, 67, 70, 71, 73,  
77–88, 90–104, 107, 108, 124,  
125, 127–133, 135–141, 143–  
146, 148–152, 192, 193, 195,  
196, 199, 204, 206, 242, 246,  
498, 830, 831, 841–843, 848,  
850
- deterministic behavior, 50
- syntax, 32–35
- termination, 77–82
- POSTFIX2, 32–35, 52, 53, 80, 135,  
826, 843
- POSTHEAP, 102–104
- POSTLISP, 101–102, 292–293
- POSTLOOP, 94–95
- POSTMAC, 99–101
- POSTSAFE, 95–96
- POSTSAVE, 97–98
- POSTSCRIPT, 5, 259
- POSTTEXT, 98, 99, 101, 826, 843
- pragmatics, 2, 4–5
- PROLOG, 367, 464
- QUEST, 652
- RUSSELL, 652

- SCHEME, 14, 122, 123, 186, 197, 199, 201, 205, 216, 259, 274, 279, 315, 327, 356, 357, 398, 399, 415, 424, 434, 436, 438, 454, 712, 739, 843, 848
- SELFISH, 311
- semantics, 2–4
- sequential, 490
- SILK, 675, 676, 678, 680–685, 688–696, 698, 699, 705–708, 710, 711, 713–718, 721–725, 728, 729, 732–736, 739, 741, 745, 751, 762–764
- SMALLTALK, 259, 314, 453, 517
- SML, 463, 475, 522, 623, 638, 644, 648, 651
- SNOBOL4, 284
- STACKFIX, 96–97
- standard library of, 195
- syntactic sugar for, 195
- syntax, 2–3
- typeless, 516
- universal, 41, 72, 73, 92, 93, 103
- WATER, 454
- XML, 454, 455
- Programming paradigm, 492
- Progress rule, 46, 54–64
- Progress rules
  - proof tree, 57
- proj, 419, 420
  - denotational semantics, 421
  - operational semantics, 420
- Projection, 773
- Projection function, 792
- PROLOG, 367, 464
- Proof tree, 57, 58
- Proof-carrying code, 767
- Proper subset ( $\subset$ ), 771
- Properly tail recursive, 739
- Proverbs, 257
- Pure, 350, 634, 657
- Purely functional language, 315
- Quadruple, 773
- QUEST, 652
- Queue, 507, 508
- Quintuple, 773
- quote, 204
  - desugaring in FL, 209
- Quotient (/), 775
- R-value, 358, 363
- Rabbit, 767
- Race condition, 494
- raise, 558
  - raise, 402–404, 406–409, 411, 413, 414
    - denotational semantics, 406
  - Raise an exception, 402
  - raise-quota!, 394
- Rand, 18
- Rat, 770
- Rator, 18
- rec, 198, 257, 279, 337, 338, 385
  - denotational semantics, 253, 347
    - CBN, 272
    - CBV, 273
    - standard, 385
  - free and bound identifiers, 228
  - operational semantics, 241, 272, 336, 337
- rec-handle, 408, 409
- receive!, 508
  - operational semantics, 510
- Receiver, 369
- Receiver, to simulate multiple value return, 783
- reconstruct, 592
- Reconstruction of types, *see* Type, reconstruction

- Record, 418, 426, 427, 436
  - dot notation, 427
  - variant, 456, 793, *see also* Sum
- record, 297, 311, 426
  - denotational semantics, 300
- record-delete, 427, 428
- record-insert, 427, 428
- record-size, 427
- recordrec, 297, 298, 311
  - desugaring, 298
- Recursion, 847
- Recursion, 787–788
- Recursive definition, 155
- Recursive type, *see* Type
- Recursive types
  - Equi-recursive type equality, 548
  - Iso-recursive type equality, 547
- Redex, 64, 75
- Reducible configuration, 42
- Reduct, 65
- Reduction
  - applicative order, 262
  - normal order, 262
- reelect, 393
- Referential transparency, 72, 319, 349–350, 590
- referentially transparent, 657
- Reflexive, 166, 539, 553, 774
- Reflexive domain, 192
- Reflexive transitive closure, 188
- Regions, 654
- Register allocation, 723
- Register allocation and spilling, 768
- Relation, 498, 774–775
  - anti-symmetric, 774
  - binary, 774
  - closure of, 775
  - composition, 775
  - equivalence, 539, 774
  - reflexive, 774
  - symmetric, 774
  - transitive, 774
  - transitive closure of, 775
- release!, 505
  - operational semantics, 507
- relop, 34
- rem
  - denotational semantics, 254
  - operational semantics, 243
- rename, 298
  - desugaring, 298
- Rendezvous, 508
- Replication, 768
- Representation invariant, 602
- restrict, 298
  - desugaring, 298
- resume, 411–413
- Resumption semantics of exceptions, 402
- return, 324
- Return code, 402
- Rewrite rule of an operational semantics, 46
- Rewrite rules
  - side conditions, 49
- Rewrite rules of SOS, 41
- Reynolds, John, 415
- Reynolds, John C., 581
- right, 246, 270
  - denotational semantics, 254
  - operational semantics, 243, 339
- Right hand side (RHS) of a transition, 42
- Rigor mortis, 49
- Robinson, 590
- RPN, 53
- Run time, 513, 637
- RUSSELL, 652
- Russell’s paradox, 771

- S-expression, 205
- S-expression grammar, 13, 21–31, 109
- Safe transformation, 72
- Safety
  - of program transformations, 349
- same-identifier?*, 252
- same-location?*, 342
- Scanner, 378
- scar*, 431
- scdr*, 431
- Schema, *see* Type, schema
- Scheme, 767
- SCHEME, 315, 357
- SCHEME, 14, 122, 123, 186, 197, 199, 201, 205, 216, 259, 274, 279, 315, 327, 356, 357, 398, 399, 415, 424, 434, 436, 438, 454, 712, 739, 843, 848
- Schmidt, David, 415, 550
- scons*, 431, 433
- Scope of a variable, 229, 258, 281
  - dynamic, 282
  - hierarchical, 281–293, 296
  - hole in, 229
  - lexical, 282
  - non-hierarchical, 296–302
  - static, 282
- Scott, Dana, 153
- SDT (Static Dependent Type), 634
- SECD machine, 104
- sel*, 33, 34
- select*, 297, 426
  - denotational semantics, 300
- select-sym*, 428
- Selective closure conversion, 756
- Self, 305
- self*, 306, 311, 388, 389
- SELFISH, 311
- Semantic algebra, 109
- Semantics, 2–4
  - denotational, 109
  - informal, pitfalls of, 12–14
  - of POSTFIX, 7–10
  - operational, 37–104
- Semaphore, 510
- send*, 306, 310
- send!*, 507
  - operational semantics, 510
- seq-proj*, 422
- seq-size*, 422
- Sequence, 418, 421
- sequence*, 422
- sequence*, 345, 387
- Sequence domain, 796–798
- Sequence pattern, 29
- Sequential, 489, 490
- Set, 770–772
  - builder notation, 771
  - cardinality, 772
  - difference ( $-$ ), 771
  - disjoint, 772
  - element of ( $\in$ ), 770
  - empty  $\{\}$ , 770
  - equal, 771
  - functions, 775–790
  - intersection ( $\cap$ ), 771
  - partition, 772, 774
  - powerset, 772
  - product ( $\times$ ), 773
  - proper subset ( $\subset$ ), 771
  - singleton, 770
  - subset ( $\subseteq$ ), 771
  - union ( $\cup$ ), 771
- Set theory, 769
- set!*, 357
  - denotational semantics, 358
- set-mfst* , 351
- set-msnd* , 351
- sexp=?*, 225
- sfilter*, 434



- Shakespeare, William, 107, 155, 673
- Shared data, 504
- Sheldon, Mark, 652
- Shelley, Percy Bysshe, 313
- Side condition, 49
- Side effect, *see* Effect
- Sieve of Eratosthenes, 433
- signal**, 402
- Signal an exception, 402
- Signal processing style, 508
- Signature of a function, 776
- SILK, 675, 676, 678, 680–685, 688–696, 698, 699, 705–708, 710, 711, 713–718, 721–725, 728, 729, 732–736, 739, 741, 745, 751, 762–764
- simp**, 467, 468
- simp-arithop**, 467, 468
- Simultaneous substitution, 238
- Single-assignment cell, 510
- Single-threaded, 322–324, 332, 338, 341, 378
- Singleton set, 770
- skip**, 33, 34
- Skolem constant, 620
- Small-step operational semantics (SOS), 41–66
- SMALLTALK, 259, 314, 453, 517
- smap**, 434
- Smash sum, 178
- SML, 463, 475, 522, 623, 638, 644, 648, 651
- snd**, 217
- SNOBOL4, 284
- snoc**, 480, 481
- snoc~**, 480, 481
- Solution to a recursive definition, 156
- Sort, 581
- SOS (small-step (or structured) operational semantics), 41–66
- Soundness
  - semantic of a type system, 592
  - syntactic of a type reconstruction algorithm, 594
- Soundness of a transformation, 677
- Soundness of denotational semantics, 145
- Source of a function, 775
- special** variable, 294
- Spinoza, Benedict, 653
- split**, 218
- square**, 366, 368
- Stack equivalence, 86
- STACKFIX, 96–97
- Standard identifiers, 207
- Standard library, 195
- Standard semantics, 379
- State, 313
- State components of a configuration, 39
- State context, 365
- State variables, 320
- Static checkability, 72
- Static dependent type, 634
- Static property, 513
- Static scoping, 281–293
- Static semantics, 513
- Static type, 517, *see* Type
- Steel Jr., Guy L., 327
- Steiner, Jacqueline, 365
- Store, 326, 332
  - in denotational semantics, 341–343
- Store effect, 655
- Stoy diagram, 229–232, 284, 319
- Stoy, Joseph, 415
- Strachey, Christopher, 153, 414
- Stream, 378, 431
  - examples of, 435
- Strict, 201

- pairs, 270
  - product, 428
- Strict function, 190
- Strictness analysis, 265
- String, 418
- String*, 770
- Strong sum, 633
- struct** in C, 454
- Structural induction, 81, 227
- Structure, 418, 427, 436
- Structure restriction, 62
- Structured operational semantics, 62
- Structured operational semantics (SOS),
  - 41–66
- Stuck state, 42
- Stuck state of operational semantics,
  - 41
- Subset, 771
  - proper ( $\subset$ ), 771
- Substitution, 224–239, 590
  - definition of, 234
  - simultaneous, 238
- Subtype, *see* Type
- Success continuation, 458, 480
- Sugar, 802
- Sum, 442–449
  - discriminant, 443
  - named, 442, 447–449
  - positional, 442–447
  - strong, 633
  - tagged, 442
  - weak, 633
- sum**, 391
- Sum domain, 442, 793–796
- Sum of products, 449–464
- sum-list**, 403, 407
- sumcase**, 443, 446
  - denotational semantics, 445
  - operational semantics, 445
- Supertype, *see* Type, supertype
- Surjective function, 780
- swap**, 33, 34
- switch**, 413, 414
- sym=?**, 217
- symbol**, 198
- symbol?**, 217
- Symbolic token, 22
- Symbolic-expression, *see* S-expression
- Symmetric, 539, 774
- Synchronization, 503, 507
- Synchronize, 490
- Syntactic algebra, 109
- Syntactic domain, 23
  - compound, 23
  - primitive, 23
- Syntactic sugar, 195, 802
- Syntactic value, 350, 590, 635, 646,
  - 648, 649
- Syntax, 2–3, 17–36
  - abstract, 18–20
  - concrete, 20–21
  - of **POSTFIX**, 6
  - S-expression grammar, 21–31
- Table, 418
- tabulate**, 425
- Tag, 442
- Tagbody, 294
- tagcase**, 447, 448
  - denotational semantics, 451
  - else**, 447
  - operational semantics, 449
- Tagged sum, *see* Sum
- Tagged union, *see* Sum, 793
- tail**, 434
- tail*, 797–798
- Tail call, 720
- Tail call optimization, 739
- Tail recursion, proper, 739
- talk** , 597

- Target of a function, 775
- Termination, 72
- Termination semantics of exceptions, 402
- Test, 18
- test-boolean*, 380
- test-location*, 380
- test-procedure*, 380
- the*, 524
- Thread, 367, 490–503, 507
  - handle, 493, 495
- thread?*, 496
- thread?*, 493
  - operational semantics, 497
- throw, 367, 399, 402
- Thunk, 270–272, 279, 341
- Time-slice, 491, 492
- tlet*, 524
- Token, 22
- Top ( $\top$ ), 167
- Top-level Procedure, 763
- top-level-cont*, 380
- Total function, 274, 777
- touch, 266, 434, 502
- Transaction, 352
- Transform equivalence, 85–88
- Transformation
  - safe, 72
- Transition
  - deterministic, 43
  - left hand side (LHS), 42
  - non-deterministic, 43
  - one step, 42
  - right hand side (RHS), 42
- Transition path, 42
- Transition relation
  - confluence, 75
  - deterministic, 42, 50
  - non-deterministic, 42
- Transition relation of SOS ( $\Rightarrow$ ), 42
- Transitive, 166, 539, 553, 774
- Transitive closure, 553, 775
- trap*, 403, 404, 406–408, 413
  - denotational semantics, 406
- treeof*, 572, 573
- Triple, 773
- true, 216, 217
- try, 410
- try, 355
- try-catch-finally*, 410
- Tuple, 418, 773, 791
  - equal ( $=$ ), 773
  - pair, 773
  - projection, 792
  - projection ( $\downarrow$ ), 773
  - quadruple, 773
  - quintuple, 773
  - to simulate multiple arguments, 781
  - to simulate multiple return values, 783
  - triple, 773
- tuple*, 596
- tuple-ref*, 596
- Twiddle, 459
- Type, 513–550
  - abstract, 599–652
  - algebraic schema, 665–668
  - as approximate value, 516, 527
  - as set, 516
  - checking, 525–583
  - coercion, 556
  - constructor, 522
  - dependent, 629
  - derivation, 533
  - dynamic
    - advantages of, 518
    - dynamic vs. static, 517–518
    - environment, 528–529, 591, 592
    - error, 517

- existential, 608–619
- explicit, 519, 583
- explicit vs. implicit, 518–519
- generic**, 589, 665
- inclusion, *see* Type, subtype
- inference, 518, *see* Type, reconstruction, 584
  - decidability of, 595
- judgment, 529
- monomorphic, 519–536, 561
- most general, 584
- nonce, 619–628
- of a function, 776
- of an application, 778
- polymorphic, 519, 520, 561–567, 586
  - projection, 563, 564
- principal, 584
- reconstruction, 518, 583–592
- recursive, 546–550
- rule, 529–533
- safe, 552
- schema, 589, 665–666, 668
- simple vs. expressive, 519
- static
  - advantages of, 517–518
- subtype, 551–561
  - anti-monotonic, 554
  - monotonic, 553
- supertype, 552, 556
- typed data, 536
- unification, 590–591
- well-typed, 517, 555, 584, 592
- Type checker, 517
- Type loophole, 456
- Typed assembly language, 767
- Typed data, *see* Type
- Typed intermediate languages, 767
- Typeless language, 516
- unbound, 225
- unbound?, 225
- Uncountable, 772
- Undefined, 777
- Unholy commingling, 629
- Unification, 590–591
- Union, *see* Sum
  - disjoint, 793
  - tagged, 793
- union in C, 454
- Union ( $\cup$ ), 771
- Unit*, 770
- unit, 216, 217
- unit*, 249
- unit?, 217
- Unitary, 658
- Universal programming language, 41, 72, 73, 92, 93, 103
- Universal quantification, 563, 589
- unlock, 603
- up-to, 391
- update, 345, 387
- Upper bound, 167
- useq-delete, 425
- useq-insert, 425
- useq-proj, 425
- useq-size, 425
- useq-update, 425
- usequence, 425
- val-to-comp*, 252, 344, 386, 405
- val-to-storable*, 358–360
- Valuation function, 111, 128
- Value
  - syntactic, 350, 635, 646, 648, 649
- Value constructor, 458
- Value deconstructor, 458
- Value environment, 528
- Value, syntactic, 590
- Variable, 224, 281, 802

- capture, 233, 566
- declaration, 227, 281
- dereferencing, 358
- reference, 227, 281
- scope, *see* Scope of a variable
- Variable capture
  - external, 234
  - internal, 233
- Variant record, 442, 456, 793, *see also* Sum
- Vector, 418
- `vector-map`, 573
- `vectorof`, 573
- View, 480–487
- `vproc`, 278
  
- Wadsworth, Christopher, 414
- WATER, 454
- Weak sum, 633
- Wegner, Peter, 550
- Well-typed, 467, 517, 530, 555, 584, 592, 778
  - lambda abstractions, 785
- `while`, 330, 366, 385
- Whitespace, 6, 22, 197
- `with-boolean`, 386
- `with-boolean-comp`, 252
- `with-boolean-val`, 252
- `with-denotable`, 252
- `with-fields`, 297, 298
  - desugaring, 298
- `with-lock`, 506
- `with-pair`, 324
- `with-procedure`, 386
- `with-record`, 299
- `with-value`, 252, 344, 386, 387, 405
- `with-values`, 252, 344, 386
- Witty, Carl, 93
- Wordsworth, William, 37, 513
- Wright, Steven, 313
  
- `writeln`, 383
- XML, 454, 455
- `yield`, 389, 390