

# Index

- 2-pass compiler 26
- 2-scan compiler 26
  
- a posteriori type 455
- a priori type 455
- ABC 46
- abstract class **473**, 701, 703
- abstract data type 471–472
- abstract syntax tree **9**, 22, 52, 55, 194
- acceptable partitioning 222, **223**–224
- acceptable-set method **138**–139
- accumulating arguments 585, 595
- acquiring a lock 660, 669–670, 674, 695
- action routine 616
- ACTION table **160**, 170, 191
- activation record 32, 412, 462, **482**
- active node **285**–286
- active routine 32, **485**, 497, 711
- active-node pointer **285**–287, 297
- actual parameter in Linda 664
- Ada 447, 696
- address descriptor 319
- address space 29, 376, 657, **659**, 662, 668, 675
- administration part 483–484, **514**, 522
- aggregate node allocation 588
- Algol 68 457, 463–464, 488, 496, 622
- Algol 68 format 187
- alignment requirements 379, **399**, 435, 465, 467, 533
- alternative **37**, 113–114, 702
- ambiguous grammar **38**, 111, 172, 192, 457
- amortized cost **397**, 560
- analysis–synthesis paradigm 6
  
- ancestor routine **485**, 490
- AND/OR form **701**, 708
- annotated abstract syntax tree **9**–10, 54, 279
- annotation **9**, 96, 194
- anonymous type declaration 449
- anti-dependence 687
- ANTLR* 132
- application spine **561**, 573, 718
- applicative-order reduction **566**, 594
- applied occurrence 199, 380, 440, **441**
- arithmetic sequence construct 541
- arithmetic simplification **367**, 370, 586
- arity 546, 566, 587, **610**, 639
- array descriptor 469
- array type 450, **467**
- array without array 534
- assembly code **374**, 379–380, 583, 667
- `asserta()` 630
- `assertz()` 630–631
- assignment 248, 277, 321, 416, 458, 464, 532
- assignment under a pointer 334, 688
- associative addressing 664
- AST 9
- asynchronous message passing **662**–663, 692
- atomic action 664
- Attach clauses instruction 603
- attribute **9**, 96, 135, 234, 547
- attribute evaluation rule **196**–197, 211, 223, 227, 274
- attribute evaluator 196, **202**, 204, 231, 273
- attribute grammar 7, **195**, 547
- automatic parallelization 684, 695
- automatic programming 33

- auxiliary register 309
- available ladder sequence 328
  
- back-end 3, 6, 11, 32
- backpatch list 263, 380
- backpatching 263, 380
- backtracking 77, 117, 596, 607, 654
- Backus Normal Form 37
- Backus–Naur Form 37
- backward data-flow 262, 277
- bad pointer 462
- bag 399
- base address 467–468
- basic block 254, 320, 322, 373
- basic item 72–73, 80
- basic operand 351
- basic type 450, 460, 543, 582
- BC( $k,m$ ) parsing 152
- binary relation 598
- binding a logical variable 596, 654
- bison* 178
- bitset 470
- block 397, 399
- blocked thread 668–670, 695–696
- BNF 37, 59, 158, 202
- body 599
- Boolean control expression 502
- Boolean type 461
- bootstrapping 2, 709
- bottom-up parser 114, 150, 178, 191, 230, 234–235, 712
- bottom-up pattern matching 341, 342–343, 346
- bottom-up rewriting system 341
- bottom-up tree rewriting 320, 357, 365, 372
- bounded stack 436
- boxing analysis 582
- broad compiler 26–27, 96, 244, 320
- bucket 100, 102
- BURS 337, 341, 349, 357, 494
- busy waiting 669, 695
  
- C 25, 117, 375, 442, 549, 558, 597
- C preprocessor 107
- caching behavior 688
- call by reference 514
- call by result 514
- call by value 514
- call by value-result 514
- callee 29, 260, 513, 534, 576, 583
- callee-saves scheme 515, 534
- caller 260, 518, 576, 583
- caller-saves scheme 515, 534
- calling graph 41, 43
- calling sequence 513, 515, 517, 634
  
- candidate set 261
- case label 506, 536
- case statement 505–507, 536
- cast 457
- cell 411, 420
- character move 73
- child 112
- child routine 485, 495
- chunk 397, 399
- class 471
- class composition 700
- class extension 700, 708
- clause 596, 600
- cloning 368, 370–371
- closed hashing with chaining 100
- closure 439, 498, 589
- closure algorithm 42, 498
- closure specification 43, 51
- CM-5 657
- coalesce 400, 403, 408
- code address of a routine 483, 485, 491, 496, 499, 587
- code generated for a query 621
- code generation 19, 290, 295, 297, 302, 320, 337, 374, 501, 523, 525, 568
- code generation module 24
- code hoisting 277
- code in-lining 24, 369–370, 395, 582, 623, 638
- code segment 375, 396, 716
- code selection 293
- code-generating scan 341
- coercion 448, 455
- collision 100, 103, 444
- column-major order 468
- comb algorithm 90
- comment starter 68, 187
- common subexpression 324–325, 334–335, 391–392
- common subexpression elimination 310, 324–326, 549
- communication daemon thread 672, 674
- compaction 408, 421, 428, 437, 717
- comp.compilers newsgroup 185
- compilation by symbolic interpretation 318
- compilation unit 31, 357, 450, 523
- compiler 1
- compiler compiler 8, 178
- composition operator 58, 186
- compound result type 517
- concurrent execution 663, 675, 685
- concurrent garbage collection 409, 435
- condition code 286
- condition synchronization 660
- condition variable 660, 670

- conditional compilation 104
- conditional text inclusion **104**, 107
- conservative garbage collection 414
- conservative parallelization 687
- constant folding 24, 49, 55, **367**, 395, 549
- constant propagation **250**, 253, 369, 395, 638
- context condition **39**, 195–196, 199, 709
- context handling 9, 19, 194, 244, 440
- context handling module 23, 54, 97, 279
- context (in coercions) 455
- context switch **667**–668, 670
- context-free grammar **34**, 39, 60, 133, 194, 712
- context-handling module 195
- continuation 482–483, **488**, 618
- continuation information **483**–484, 501
- continuation of a stack 139, **140**, 175
- control flow graph **239**, 245, 253, 266, 321
- conversion 457
- core of an LR(1) state **170**–171
- coroutine 29, 33, 414, 482, 484, **485**, 500
- coroutine call 29
- ctype* package 68
- currying 489, 497, 540, **546**, 551, 560, 592–593
- cut operator 597, 601, **628**, 654
  
- dag 321
- dangerous compiler 29
- dangling else **172**–173, 175, 192
- dangling pointer 407, 463–464
- data dependency 686
- data parallelism 658, **665**–666, 685
- data segment **376**, 396, 716
- data-flow equations **253**, 262, 265, 267, 277
- data-flow paradigm 195
- data-parallel programming 658, 665, 688
- dead code elimination **368**, 395, 638
- dead variable 321
- debugging of garbage collectors 415
- declaration of a method 473
- defining occurrence 199, 380, 440, **441**
- definition of a method 473
- degree of a node 360
- dependency graph **200**, 274, 321, 326–328
- dependent inheritance 478
- dependent on **200**, 334
- dereferencing 407, **461**–462, 607, 609, 625
- dereferencing a field selection 461
- derivable from 40
- derivation 36
- deserialization 672
- desugaring 553
- deterministic **110**, 117, 134, 172, 337
- direct left-recursion 130
- directed acyclic graph 321
- directly derivable from 39
- discriminated union 467
- dispatch table **475**–476, 478–479, 481, 512
- display goal 601
- distributed system **656**, 695
- distributing an array 690
- dope vector 526
- dotted item **71**–72, 110, 153, 373
- dotted tree 342
- dvi format 8
- dynamic allocation part 515, **516**
- dynamic array **470**, 514
- dynamic attribute evaluation **204**, 210
- dynamic binding **474**, 480, 708
- dynamic conflict resolver **132**, 143
- dynamic cycle detection 211
- dynamic link **483**, 501, 514
- dynamic programming 337, 347, **350**, 357
  
- $\epsilon$  closure 77, 79, 154, 166
- $\epsilon$  move **73**, 154, 186
- EBNF **38**, 50, 59
- elaborate a statement 284
- element of an array 467
- element selection 468
- empty list 541
- empty state **84**, 88, 354
- enumeration type **461**, 467, 475
- equation 541
- error correction **116**, 136, 144, 175
- error detection 115–116, 142, 190
- error handling in lexical analyzers 93
- error state 158
- error-recovering non-terminal 176
- escape character 59
- Escher, M.C. 301
- Ethernet 657
- evaluate an expression 284
- event queue 404
- exception handler 415, **522**, 537
- exclusive or 436
- executable code output module 24
- Extended BNF 38
- extended subset 30, 50
- extensible array 98, 287, 398, **404**, 435
- external entry point 377
- external name 377
- external reference 377
- external symbol 377
- external symbol table 377
- extra-logical predicate 630
  
- fact 596, **598**
- failure in the WAM 625

- field selection 466
- file inclusion 104
- filter 557
- final method 479
- fine-grained parallelism 685
- finite-state automaton 80
- finite-state tree automaton 347
- FIRST set 121
- first-class citizens, functions as 545
- FIRST/FIRST conflict 127
- first-fit 90
- FIRST/FOLLOW conflict 127
- flex* 187
- flow dependence 687
- FOLLOW set 123
- follow set 138
- FOLLOW-set error recovery 139
- follow-set error recovery 138
- fork statement 659
- formal attribute 195
- formal parameter in Linda 664
- for-statement 508
- forward declaration 441, 473
- forward reference 450, 594
- forwarding address 676
- FP 482, 515, 517–518
- frame 412, 482, 489, 585
- frame pointer 482, 489, 493, 559
- free bit 399, 420–421
- free list 400, 416
- from-space 408, 425
- front-end 3, 52, 539
- FSA 50, 80, 347, 373
- full symbolic interpretation 247, 251, 277
- full-LL(1) 124, 190
- fully parenthesized expression 12, 701
- function result register 517
- functional core 549, 558–559, 568, 575, 594
- functor 607, 639
  
- garbage collection 262, 398, 407, 545, 598
- garbage set 407
- gcc* 297, 333
- general closure algorithm 44
- general name space 442
- generational garbage collection 409, 429
- generator 557
- generic pointer type 462
- generic unit 109, 525
- global routine 485
- GNU C compiler 297
- goal 599
- goal list stack 601
- goto statement 251, 368, 502
  
- GOTO table 158
- grammar 34, 35
- grammar symbol 35, 195
- graph coloring 92
- graph reducer 560
  
- handle 152
- hash function 100, 682
- hash table 100, 188, 325, 443, 470, 532, 626, 682
- Haskell 540
- head 599
- header file in C 22
- heap 397, 462, 484, 495, 498, 560
- heap sort 404
- heroic compiler 659
- heuristic graph coloring 360
- hidden left-recursion 130, 149
- High Performance Fortran 693
- higher-order function 545
- HPF 693
- Huffman compression 86
  
- IC 23
- Icon 29, 33, 389, 488, 496, 615, 622
- if-statement 368, 505
- implementation language 1, 100, 399, 451
- implicit parallelism 659
- implicit receipt 662
- imported scope 448
- in* operation in Linda 664, 679
- in situ replacement 595
- inactive routine 485
- incremental garbage collection 408, 436
- independent inheritance 478
- indexing 468
- indirect left-recursion 130
- indirect routine call 261, 471, 489
- indivisible operation 660, 664–665, 669, 695, 697
- inference rule 42
- inference technique 601
- inheritance 472
- inherited attribute 196
- initial state 80, 343, 351
- in-lining 368
- input tree 339
- instantiation through dope vectors 526
- instantiation through expansion 526
- instruction ordering 293
- instruction-collecting scan 341
- instruction-selecting scan 341
- instrumenting a program 317
- interface 480
- interference graph 92, 359
- intermediate code 9, 23

- intermediate code generation module 23
- intermediate code optimization module 24
- intermediate code tree 279
- interpreter 3, **281**, 295, 560, 603, 630, 699
- interprocedural data-flow analysis **260**, 688
- invalidate replicated copies 677
- IS-dependency 212
- IS-SI graph **212**, 274
- item **71–72**, 110, 153, 373
- iterative interpreter **285**, 297, 389, 603
- iterator **485**, 500
  
- Java RMI 673
- join operator 255
- jump buffer 490
- jump table 506
- jumping out of a routine **488**, 496
  
- kind 449
  
- ladder sequence 328
- LALR(1) automaton 172
- LALR(1) parsing 152, 170, 176, 191, 234
- lambda lifting **499**, 559, 594
- language generated by a grammar 40
- last-def analysis 253
- late evaluation 224, 228, 328
- latency 671
- lattice 250, 277
- L-attributed grammar **230**, 269, 275
- lazy Boolean operator 117–118, 503
- lazy evaluation 328, **547**, 560, 564, 566, 575
- lcc* 389
- least fixed point 43
- left-associative 153, 540, 712
- left-factoring 129
- left-hand side 35
- leftmost derivation 36
- leftmost innermost redex 566
- leftmost outermost redex 566
- left-recursion removal 129
- left-recursive **38**, 120, 127, 144
- Lempel–Ziv compression 86
- let-expression 543, 572, 594
- lex* 83, 94, 187
- lexical analysis module 22
- lexical analyzer 10, 61, 68, 83, 93–94, 541
- lexical identification phase 98
- lexical pointer 483–484, **491**, 493, 514, 559
- library 25, 281, 297, 376, 658, 672
- lifetime 463
- Linda 664
- Linda kernel 678
- Linda preprocessor 678
- Linda Tuple Space 678
- linear-time 31, 86, **111**, 168, 224, 277, 406
- linked list 398, **403**
- linker 25, **376**, 381, 389
- lint* 247
- list 541
- list comprehension **542**, 557
- list procedure **616**, 654
- live analysis 262–263, 265, 319, 333, 358
- live variable 262
- LL(1) conflict **127–128**
- LL(1) grammar **123**, 127, 142
- LL(1) parser generation 123
- LL(1) parsing 121–122, 139
- LL(2) grammar **132**, 190
- LLgen* 132, 142, 179, 232, 237, 711
- LLgen directive 144
- loader **376**, 389
- local attribute 203
- local variable area 515
- location 458
- lock variable **660**, 669
- `long jmp ( )` 490
- look-ahead set 165–166, 190
- loop interchange 689
- loop restructuring transformation 688
- loop unrolling 511
- loop-carried dependence 687
- loop-exit list 248
- LR automaton 159
- LR item **153**, 191
- LR reduce item 154
- LR shift item 154
- LR(0) grammar 162
- LR(0) parsing 152–153, 191
- LR(1) parsing 152, 165, 191, 217
- lvalue **458**, 462, 470, 514
  
- machine code generation 374
- machine code generation module 24
- macro application 103
- macro call 103
- macro definition **103**, 444
- makefile 32, 296
- mark and scan 408, **420**
- mark and sweep 420
- marked bit 420
- marker rule 131
- marking 420
- marking phase 420
- marshaling 672
- Match instruction 603, 609
- match move 134
- matching a goal 599

- matching in regular expressions 58
- matrix 467
- maximal basic block 320
- meaning of 'to compile' 3
- meet operator 255
- member 35
- memoization **285**, 348, 350, 389, 436
- memory dump 520
- memory fragmentation **408**, 419, 427
- memory management unit 462
- message aggregation 692
- message buffer 672
- message combining 692
- message handler 662
- message passing 658, 671
- message selection 674
- method 663
- method invocation **471**, 479, 512, 673
- method overriding 472
- Miranda 593
- module 523
- module initialization 524
- monitor **660**, 669, 695
- monitor lock 670
- most recent binding 607, 613
- MPI 658
- multicomputer 657
- multiple inheritance **476**, 533
- multiprocessor 657
- multi-visit attribute evaluation **219**, 230
- mutual exclusion synchronization 660
  
- name equivalence 454
- name generation for modules 524
- name list **98**, 442
- name space **442**, 526, 532, 675
- narrow compiler **26**, 96, 232, 269, 302, 320–321, 444
- nested routine **485**, 558, 597, 618
- newline 57, 185
- non-correcting error recovery 116
- non-cyclic attribute grammar **217**, 275
- non-deterministic 162, 362
- nondeterministic-polynomial 90
- non-local assignment 494
- non-local goto 482, **488**, 496, 518
- non-local label **488**, 491, 496
- non-terminal symbol **35**, 39
- no-op instruction 379
- normal-order reduction 566
- NP-complete 89, **90**, 293, 320, 360, 373
- N*-phase compiler 27
- NPRD grammar 189
- null sequence 367
  
- nullable non-terminal **38**, 121–122, 190
  
- object **471**, 658
- object constructor **471**, 527, 702, 708
- object destructor **471**, 527
- object identifier 675
- object location 675
- object migration 676
- object replication 677
- object-oriented compiler construction 699
- object-oriented parsing 702
- occur check 608
- offset table 479
- offside rule 438, **541**
- OID 675
- one-shot garbage collection 408
- one-space copying 437
- on-the-fly garbage collection 408
- operation on an object 663
- optimization 32, 64, 102, 403, 479, 511, 646, 648
- ordered attribute evaluator 224
- ordered attribute grammar 224
- out operation in Linda 664
- outer loop 686
- outline code 46
- output dependence 687
- overloading 447
  
- package 523
- panic-mode error correction 139
- parallel loop statement 665
- parallel system 656
- parameter area 514
- parameter passing mechanism 514
- parent class **472**, 476, 481
- parent routine 485
- parse table 164
- parse tree **9**, 148
- parser 36, **110**, 702
- parsing **9**, **110**, 540, 616
- partial evaluation 297, **300**, 390, 638
- partial parameterization **488**, 499
- partition (Tuple Space) **679**, 683
- partitioning (of attributes) 222
- passing a routine as a parameter 463, **487**, 495, 617
- pattern matching **543**, 553, 593
- pattern tree 337
- PDA 134
- peephole optimization 333, **371**
- phase of a compiler 27
- PL/I preprocessor 104–105, 107–108, 710
- pointer chasing 677
- pointer consistency 410
- pointer layout 410

- pointer problems 334
- pointer (re)subtyping 474
- pointer reversal 422
- pointer scope rules **463**, 559, 717
- pointer supertyping 474
- pointer tagging 588
- pointer type 461
- pointer validity 410
- polymorphic function application 551
- polymorphic type 543
- polymorphic type checking 551
- polymorphism 474
- polynomial time 90
- pool of threads 674
- port 662
- portability 32
- portability of a compiler 32
- position-independent code 716
- post-order visit 112
- precedence 59, 174
- precedence parsing 152
- precomputation **66**, 80, 120, 157, 191, 285, 346, 469, 475
- prediction move 134
- prediction rule 155
- prediction stack 134
- predictive parser 121
- predictive recursive descent parser 17, **121**
- pre-order visit 112
- prerequisite to **200**, 210
- privatization 689
- process 658, **659**
- process abstraction 667
- production rule **35**, 39, 196, 547
- production step **35**, 40
- production tree 35
- program counter 351, 395, **482**, 493, 662, 667
- program data area **409**, 496
- program generator 7, 66, 301
- program text input module 22, 56
- program-counter relative addressing 395
- Prolog 43, 296, 597, 628, 630, 654
- property list 247
- pseudo-register **329**, 362
- pure register machine 304
- pure stack machine 303
- push-down automaton **134**, 159
- PVM 658
  
- qualifier 557
- query 596, **599**
  
- reaching-definitions analysis 253
- read operation in Linda 664
- read/write mode 646
- receive statement 662
- reclamation 434
- record type 465
- recursive descent parser 15, **117**
- recursive interpreter 281
- redex 561
- reduce item **72**, 186, 714
- reduce move 160
- reduce-reduce conflict 162
- reducible expression 561
- reduction engine 566
- reduction (graph) 561
- reduction (in parsing) 152
- reduction stack 159
- reference counting 407, **415**
- referential transparency 406, 544
- refinement 482
- register allocation 293, 316, 332–333, 357
- register allocation by graph coloring 320, **357**
- register and variable descriptor 318
- register descriptor 319
- register interference graph 359
- register spilling technique 314
- register storage class 336
- register tracking **320**, 333
- register-memory machine 327
- regular description 56, **60**
- regular expression 34, 56, **58**, 237, 342
- regular grammar 34
- regvar descriptor 318
- relation 596, **598**
- releasing a lock 660
- relocation 380
- relocation information 377
- repeated inheritance 478
- repeated variables 593
- repetition operator **58**, 72, 124, 142
- replacement pattern 371
- replicated Tuple Space 683
- rest of program, grammar for 116
- resume statement 485
- resuming a routine 483, **485**
- retargetability 32
- return information 483
- return list 247
- returning a routine as a value 482, **487**, 495, 559
- right-associative 117, 192, 551, 593
- right-hand side 35
- right-recursion **38**, 191
- root set **409**, 496
- routine call **487**, 512–513, 616
- routine definition **487**, 616
- routine denotation 621

- routine type 471
- row displacement 87
- row-major order 468
- rule 596, **599**
- runnable thread 668
- running routine 32, 439, 482, **485**
- running thread 668
- run-time error 519
- run-time system **25**, 281, 396, 499, 545, 560, 659, 667
- rvalue **458**, 462, 470, 514
  
- safe pointer 464
- S-attributed grammar 230, **235**, 275–276
- scalar expansion 689
- scan phase 420
- scan pointer 425
- scheduler **668**, 686
- scheduler activation 667
- Schorr and Waite algorithm **422**, 436
- scope in block-structured languages 442
- scope of a pointer **463**, 491, 532, 559
- screening 98
- select statement 696
- selection statement 505
- self-descriptive chunk 411
- self-organizing list 102
- semantic checking 441
- semantic representation 2
- send statement 662
- sentence 40
- sentential form **35**, 40
- serialization 672
- set type 470
- Sethi–Ullman numbering 311
- set jmp ( ) 491
- setjmp/longjmp mechanism 490
- shadow memory 287
- shared variable 658, **659**, 668
- shift item **72**, 166, 172, 186
- shift move 159
- shift-reduce conflict 162
- shift-shift conflict 191
- short-circuiting a function application **573**, 586
- SI-dependency 212
- signal handler **521**, 537
- signal operation **660**, 670
- signal statement 521
- signature of a computer virus 83
- simple symbolic interpretation **247**, 277, 712
- simulation on the stack 245
- single inheritance 476
- Single Program Multiple Data parallelism 685
- SLR(1) parsing 152, 163, 190–191
  
- Smalltalk 33
- sorted set 399
- source language 1
- SP 513
- SP-2 657
- sparse transition table 86
- specialization 370
- spineless tagless G-machine 592
- spinning 669
- SPMD parallelism 685
- SR 674
- stack 513
- stack, direction of growth 513
- stack of input buffers 105
- stack pointer 513
- stack representation 245
- stack segment **376**, 396
- start symbol 35, **39**, 136, 144, 199
- state 80
- state transition 80
- static array 470
- static attribute evaluation **204**, 218
- static binding 474
- static cycle checking 211
- static link 484, **491**, 514
- status indicator 282
- storage representation in Linda 680
- strength reduction 367
- strict argument 565, 574, **576**, 579, 594
- strictness analysis **576**, 594
- string 39
- strong-LL(1) parser 124
- strongly LL(1) 124
- strongly non-cyclic **217**, 275
- structural equivalence 455
- structure type 465
- subclass 472
- subroutine 485
- subset algorithm **81**, 158, 346, 352
- substitution in grammars 129
- subtree 112
- Succeed instruction 603, 605
- suffix grammar **116**, 188
- supercompilation 363
- suspend statement 485
- suspended routine 439, 482, **485**
- switch statement 505
- symbol 39
- symbol table **98**, 133, 188, 284, 442
- symbolic interpretation 245
- synchronization primitive 660
- synchronous message passing 662
- syntactic sugar **540**, 553
- syntax analysis **9**, 57, 110, 440



- syntax analysis module 22
  - syntax tree **9**, 110, 135, 230
  - synthesized attribute 196
  
- table compression 86
- table lookup **68**, 285, 347
- tail call 582
- target code optimization module 24
- target language **1**, 375, 523, 549
- target register 309
- task 33, 656
- task parallelism 665
- TCP/IP 671
- term 596
- terminal 35
- terminal production 40
- terminal symbol **35**, 39
- terminated routine 32, 462, **485**
- test-and-set instruction 669
- TeX 8
- thread 658, **662**, 667
- thread control block 667
- thread preemption 668
- threaded code 297
- threading (of an AST) **239**, 247, 276, 287
- throughput 671
- token 10, **35**
- token description **61**, 70, 77
- tokenize 10
- top-down parser **113**, 120, 142, 232
- to-space 408, **425**
- totally-ordered group communication 678
- trail 610, **634**
- transition diagram **82**, 158, 161, 163
- transition function 80
- transitive closure **42**, 498
- transitive closure algorithm 44
- translation 1
- traversing a node 112
- traversing a tree 112
- tree rewriting 290
- triple 324, **326**
- tuple matching 664
- Tuple Space 658, **664**, 678
- Tuple Space server 678
- two-scans assembly 380
- two-space copying 408, 420, **425**, 437, 545
- type 449
- type checking 449, 673
- type declaration **449**, 544, 551
- type equivalence **454**–455, 551
- type extension 472
- type table **450**, 552
- type variable **544**, 551
  
- typed pointer 462
  
- UDP 671
- unary relation 598
- unbound variable **599**, 612, 625
- undiscriminated union 467
- unification **544**, 596, 607, 609, 612, 634, 654–655
- uniform distribution in Linda 683
- uniform self-identifying data representation 281
- Unify instruction 603, 609
- union tag 467
- union type 466
- unmarshaling 672
- unrolling factor 511
- unwinding **566**, 575, 589, 718
- update replicated copies 678
- URL syntax 185
- usage count **317**, 362
- useless non-terminal 38
- user space 667
  
- validity span 463
- value 458
- value of a variable in an assembler 379
- variable descriptor 319
- variable-length graph node 587
- vector 467
- vector apply node **588**, 595
- very busy expression 277
- virtual method **473**, 479
- visiting a node 112
- visiting routine 220
- vocabulary 39
- void type 460
  
- wait operation **660**, 670
- WAM **597**, 623, 636, 638, 648, 653
- Warren Abstract Machine **597**, 653
- weight of a subtree **310**–311, 313–314, 316, 365, 390
- weighted register allocation 310, 339, 349
- while statement 276, **507**, 509
- working stack 308, 412, 461, 483, 513, **515**, 517
  
- yacc* 176, 178, 234
- yield statement 485
  
- zeroth element **469**, 533