

Subject Index

- #: sequence length, 485
- Δ : variable change, 488
- $\otimes$: partial string concatenation operator, 212
- $\frown$: sequence concatenation, 485
- χ Regress, 472
- χ SUDS, 596
- χ Suds, 472
- $\langle$: sequence construction, 485
- $\mathbb{N}$: set of natural numbers, 481
- $\mathbb{N}_1$: set of strictly positive integers, 481
- $\otimes$: onto set product, 154
- $\rangle$: sequence construction, 485
- $\backslash$: set subtraction, 464
- integer*: set of natural numbers, 481
- real*: set of reals, 481
- $\equiv$: defined as in $\mathbb{Z}$, 482
- #define, 530
- static**, 673
- this**, 672
- 1-equivalence partition, 200
- 1-switch cover, 240, 250
 - example, 240
- 2-equivalence partition, 200
- 3-way interaction fault, 391
 - incorrect arithmetic function, 392
- 5ESS, 52
- 80x86, 61

- ANSI/IEEE Std 729-1983, 93

- ABB, 91
- ABP: Alternating Bit Protocol, 190
- ABS operator, 666, 675, 677, 679
- abstract data type, 486
- abstract snapshot, 340
- abstract syntax tree, 148, 168
 - example, 148
- acceptance testing, 77
- access control, 456
- access modifier in Java, 625, 626
- ACSR: Algebra of Communicating Shared Resources, 376

- action
 - observable effects of, 275
- actions, 260
- activation request, 456
- activity diagram, 56
- Ada, 683
- adament tester, 562
- adenosine, 504
- adequacy
 - with respect to decision coverage, 541
- Adequacy assessment, 74
- adequacy assessment, 86
 - scalability, 607
- adequacy criteria
 - based on control flow, 583
 - based on data flow, 583
 - black-box, 519
 - concurrent programs, 591
 - control flow based, 530
 - more powerful, 587
 - non-code based, 587
 - quantitative, 695
 - weaker, 545
 - white-box, 519
- adequacy criterion, 522
- adequacy measurement, 519
 - example, 519, 520
- AETG, 28, 422, 423
 - US Patent 5,542,043, 423
- AETG: Automatic Efficient Testcase Generator, 422
- agile process, 91
- agile testing, 82, 91
- agriculture, 422
- AGTC: Applied Genomics Technology Center, 403
- airborne software, 592
- aircraft, 183
- aircraft engine controller, 251
- aircraft landing, 78
- alarm, 150
- Alcoa, 512

-
- algebraic languages, 56
 - ALGOL 68, 593
 - Algol compiler, 89
 - all-uses, 577, 677, 697
 - coverage, 577, 694
 - coverage example, 578
 - Alloy, 509, 511
 - alphabet, 72
 - binary, 72
 - input, 185, 186
 - input of an FSM, 191
 - output, 185
 - output of an FSM, 191
 - alphabet incompatibility, 370
 - amplitude, 595
 - AND state, 252, 298
 - AND-node, 143, 145, 148
 - BRO constraint set, 158
 - ANSI C standard, 625, 647
 - ANSI/IEEE, 89
 - Apple
 - G4, 389
 - G5, 389
 - applet, 399
 - application
 - behavior, 59
 - binary, 605
 - carefully designed, 116
 - commercial, 98
 - deterministic, 412
 - networked, 128
 - non-deterministic behavior, 412
 - payroll processing, 108
 - requirements, 107
 - array
 - balanced, 410
 - arrays, 696
 - artifact under test, 79
 - ASCII, 33
 - aspect oriented programming, 593
 - assembly language, 40, 530
 - ASSET, 594
 - assignment, 567
 - asynchronous processing, 323
 - AT&T, 52
 - ATAC, 466, 471, 472, 592, 594–596
 - use in large systems, 595
 - ATAC-M, 687
 - Athelon, 388
 - ATM machine, 262, 492
 - atomic number, 482
 - atomic propositions, 60
 - atomic weight, 486
 - audio/video protocol analysis, 377
 - author, 58
 - AutomatedQA, 472
 - automaton, 185
 - automobile, 183
 - automobile engine controller, 432
 - auxiliary transition cover set, 364
 - B, 509–511
 - Büchi automaton, 335
 - backward trace, 144
 - balanced array, 410
 - balanced design, 397
 - banking software, 432
 - basic block, 54, 61, 62, 67, 441, 459, 460, 463, 530, 566
 - example, 61
 - exponentiation program, 62
 - global and local use, 566
 - number, 63
 - super, 596
 - unreachable, 531
 - variables defined in, 566
 - batch processing, 80
 - batch processing applications, 79
 - BDTA: Bounded Deterministic Timed Automaton, 374
 - behavior
 - observable, 27
 - behavioral science, 422
 - Bellcore, 594
 - Best choice combination, 422
 - beta testing, 76
 - bi-class, 591
 - big-bang process, 46
 - billing transactions, 78
 - binary matching, 472
 - binary numbers, 72
 - binary strings, 397
 - biology, 422
 - BIST: Built-in Self Test, 50
 - black-box category, 79
 - black-box testing, 74, 82, 83
 - BLAST, 504
 - block, 54, 61, *see* basic block
 - block coverage, 530, 531, 589
 - example, 531
 - boiler
 - gas-fired, 95
 - heating system, 114
 - steam, 95
 - temperature, 113
 - boiler command

- cancel, 113
- set temperature, 113
- shut down, 113
- boiler control, 113
 - control option, 113
 - equivalence classing, 115
 - error detection effectiveness, 175
 - infinite values, 115
 - shut command, 119
 - test selection example, 117
 - weaknesses in test, 175
- Boolean and Relational Expression, 152
- Boolean expression, 133, 139, 173, 254
 - conversion between CNF and DNF, 148
 - examples, 147
 - non-singular, 148
 - singular, 147
- Boolean operator, 146, 147, 533
 - binary, 147
 - incorrect, 148
 - unary, 147
- Boolean operator fault, 169, 178
 - example, 149
 - multiple, 178
- Boolean variable, 147, 147, 148, 172, 532, 553
- Boot Camp, 387
- bop: Boolean operator, 147
- BOR adequate test set, 152, 167
 - example, 152
- BOR constraint, 153
- BOR constraint set
 - example, 155
 - generation, 154–155
- BOR technique, 172
- BOR testing criterion, 153
- BOR: Boolean Operator, 152
- boundary
 - identification, 121, 125
 - tests near, 124
 - tests off, 125
 - tests on, 125
- boundary interior cover, 238
- boundary point, 122
- boundary value analysis, 90, 96, 121, 171, 176, 286, 383
 - example, 121, 123
 - for test generation from statecharts, 314
 - procedure, 121
 - test suite, 123
- boundary value strategy, 516
- boundary-interior test paths, 590
- boundary-interior testing, 89, 603
- BR constraint, 151
- BR: Boolean and Relational, 150
- braces({, }), 95
- brake pedal, 259
- branch coverage, 589
- branch decision coverage, 534, 539
- branch points, 589
- branches
 - uncovered, 56
- Brazil, 633
- BRE adequate test set, 152
 - size of, 168
- BRE constraint set, 154
 - derivation, 179
 - example, 160
 - for relational expression, 159
 - generation, 159–160
- BRE strategy, 172
- BRO adequate test set, 152
 - size of, 168
- BRO constraint set
 - example, 158, 162
 - example for non-singular expressions, 164
 - generation, 157–158
 - non-singular expressions, 161–164
- BRO technique, 172
- BRO testing criterion, 159
- BRO: Boolean and Relational Operator, 152
- browser combinations, 386
- BTDA: Bounded Time Domain Automata, 336
- buffer
 - dynamic allocation, 78
 - size, 78
- buffer size
 - in timed automaton, 328
- bug, 27, 89
- Bugzilla, 53
- bva: boundary value analysis, 120
- C, 468, 472, 513, 530, 532, 564, 683, 696
 - coverage measurement tools, 592
- C++, 47, 90, 109, 470, 530
 - coverage measurement tools, 592
- C++Test, 472
- C-connector, 255, 280
- C-interface, 683
- c-use, 569
 - adequacy does not imply p-use adequacy, 586
 - coverage, 571, 576, 584
 - coverage example, 576
 - feasible, 600
 - infeasible, 580
 - minimal set, 571
 - minimization, 572

- number of, 575
- c-use: computational use, 565
- CADI, 510
- call graph, 472, 477
- call-by-reference, 566
 - as definition and use, 565
- call-by-value
 - as a use, 565
- cardiac pacemaker, 80
- Carnegie Mellon University, 680, 690
- cash register, 176
- catalytic convertor, 322
- category, 127
- category partition, 74, 132
 - example, 128
 - method, 126–132, 172
- category partition method
 - for object oriented programs, 172
- CATS, 422
- cauliflower, 104
- cause, 133
- cause-effect graph, 132, 177, 180
 - construction, 138
 - creation, 135–139
 - heuristics, 143
 - heuristics example, 144
 - trace back, 143
- cause-effect graphing, 96, 132, 166, 171
 - example, 136
 - first step, 137
 - incremental strategy, 137
 - notation, 133–135
 - procedure, 133
 - to partition input domain, 497
- Cause-effect graphs, 74
- cause-effect relations, 166
- causes, 132
- Celsius scale, 82
- CFG, 58, *see* control flow graph, 69
 - annotated with data flow information, 58
- chain, *see* def-use chain
- change-impact analysis, 472
- characterization set, 198, 270
 - example, 203
 - for OR statechart, 294
 - for statechart, 269
 - from flattened statechart, 317
 - in presence of guards, 284
 - in timed Wp-method, 364, 367
 - union of, 294
 - use of, 306
- Chianti, 472
- Chidamber, Shyam R., 49
- child function, 439
- child state, 291, 296
- choices, 126
- CICS: Customer Information Control System, 509
- CK metrics, 90
- classification tree method, 172
- class
 - coupling, 48
 - number of children, 48
 - parent, 669
 - response set, 48
 - weighted methods, 48
- classification tree, 510
- Cleanscape, 473, 592
- client-server testing, 80
- clock
 - constraints, 330
 - fractional part, 335
 - integral part, 335
 - interaction, 330
 - real valued, 333
 - region coverage, 348, 381
 - regions, 375
 - reset fault, 351
 - reset operation, addition of, 360
 - reset signal, 351
 - sampling example, 348
 - sampling granularity, 349
 - smallest increment, 349
 - time assignment, 333
- clock granularity, 375
- clock guard, 327
- clock region, 337
 - collection of equivalent time assignments, 339
 - coverage, 348
 - example, 337
 - infinite, 349
 - open, 350
- clock regions, 335
- clock reset fault
 - detection example in a TIOA, 373
 - example, 342, 344
- clocks
 - in timed automata, 324
- CMS: COBOL Mutation System, 676
- CNF: Conjunctive Normal Form, 148
- COBOL, 172, 676
- code
 - access to, 588
 - element hierarchy, 589
 - instrumentation, 588
- code coverage

- dispersion in, 595
- measurement, 588
- misuse of, 591
- monitoring of, 591
- code coverage criteria, 83
- code coverage measurement, 586
 - impractical, 588
 - practical aspects, 588
- code fragment
 - borrowed from Kernighan and Ritchie, 636
- code inspection, 51, 57
- code instrumentation, 596
 - binary code, 596
- code measurement tool, 588
- code testing, 80
- code-inspection, 58
- coding phase, 76
- coding step, 181
- codon, 485, 513
- color laserjet, 174
- combinatorial design, 385
- combinatorial designs
 - balanced, 411
 - fault detection ability, 422
 - handbook, 423
 - in compiler testing, 422
 - procedure, 389
 - strength, 413
 - survey, 423
- combinatorial explosion, 133
- combinatorial object, 389
- command file, 114
- comment lines, 130
- communicating TIOA, 376
- communication protocol, 189
 - as finite state machine, 189
 - control portion, 190
 - data portion, 190
- compatibility testing, 77
- Competant Programmer Hypothesis, 627, 691
- Competent Programmer Assumption, 628
- compiler, 47
 - automatic testing, 172
 - testing, 172
- compiler integrated mutation, 687
- compiler testing, 80
- complete factorial design, 426
- completeness
 - of FSM, 219
- completeness assumption
 - not satisfied, 220
- complexity
 - see cyclomatic complexity, 47
- complexity metric
 - in static testing, 59
- component testing, 80
- compound condition, 536, 539, 545, 549
- compound predicate, 152, 578
- compound type, 109
- concurrent process, 311
- concurrent program
 - test generation, 591
- condition, 459, 532
 - as decision, 533
 - atomic, 533
 - combinations, 539
 - compound, 533
 - coupled, 533
 - elementary, 533
 - simple, 533
 - within assignment, 534
- condition coverage, 537, 538, 562
- condition/decision coverage, 539
 - example, 538, 539
- conditions, 146
- confidence, 52, 84
 - false sense of, 86
- configuration
 - order, 136
 - sample, 137
- configuration space, 384
- configuration testing, 77
- conformance
 - example, 220
 - strong, 220
 - weak, 220
- conformance testing, 189, 243
- conjunctive normal form, 148
- connected assumption, 219
- connectionless transmission, 190
- connector, 280
 - C-, 255
 - elimination, 280
 - elimination of C, 280
 - fork, 257
 - join, 256
 - junction, 256
 - S-, 256
- constant mutation, 625
- constrained mutation, 683
- constraint, 151
 - example, 150
 - false, 151
 - partition, 151
 - specification, 129
 - true, 151

-
- constraint propagation, 168
 - container program, 696
 - context coverage, 589, 601
 - contract software, 76
 - control computer, 150
 - control dependence, 71, 88
 - example, 71
 - control dependency, 453
 - control flow coverage, 471
 - control flow Graph, 439
 - control flow graph, 62, 87, 438, 566, 597
 - child, 438, 439
 - construction, 60
 - edge, 63
 - main, 438
 - node, 63
 - of function, 438
 - pictorial representation, 63
 - control node
 - direct, 453
 - indirect, 453
 - control software, 113, 150
 - control-flow behavior, 91
 - CORBA, 75
 - core behavior, 220
 - example, 220
 - of FSM, 219
 - correctness, 34–35
 - correctness proof, 89
 - cost criterion, 463
 - coupling effect, 591, 624, 627, 628, 682, 691
 - fact or fiction, 682
 - coupon, 130
 - coverage, 519
 - as percentage, 532
 - block, 592
 - control flow, 56
 - edge, 592
 - extent, 519
 - hardware designs, 595
 - kernel mode, 592
 - lack of, 587
 - LCSAJ, 545
 - requirements, 587
 - residual, 463
 - user mode, 592
 - coverage analysis
 - coarse, 595
 - coverage based minimization, 460
 - coverage criteria
 - constraining of, 589
 - coverage data
 - enormous amount, 588
 - coverage domain, 519, 520, 523
 - feasibility demonstration, 525
 - identification of infeasible elements, 527
 - infeasible, 525
 - infeasible elements, 526
 - coverage measurement
 - cause for slowdown, 588
 - coarse, 595
 - for object oriented programs, 589
 - importance, 592
 - incremental, 588–589
 - insufficient disk space, 588
 - large systems, 595
 - overhead, 595
 - reducing the number of modules, 589
 - regression testing, 596
 - selective, 588
 - survey, 590
 - system testing, 596
 - coverage measurement tools, 592
 - covering array, 389, 412
 - different from orthogonal array, 412
 - example, 412
 - minimal multi-level, 423
 - mixed level, 389
 - variable strength, 423
 - CPH: Competent Programmer Hypothesis, 628
 - CPU, 136
 - crash, 78, 79
 - credit card number, 508
 - cruise control system, 251
 - CSP, 676
 - CSP specification, 690
 - CSPE, 510
 - CT: Compound Transition, 264
 - CTG: Complete Test Graph, 376
 - CTL: Computation Tree Logic, 690
 - customer acceptability, 77
 - cyclomatic complexity, 47, 48, 90, 94, 95
 - exercise, 94
 - Daimler-Chrysler, 91
 - danger signal, 608
 - Darlington Nuclear Generating Station, 607
 - DART, 28
 - data, 69
 - incorrect flow, 593
 - data context
 - coverage, 582, 602
 - covered, 581
 - table, 573
 - uncovered, 582
 - data dependence, 70

- example, 70
- Data Dependence Graph, 69
- data entry
 - errors, 119
- data entry mode, 44
- data environment, 572
- data flow
 - to augment tests derived from FSM, 249
 - for arrays, 600
 - in C programs, 593
- data flow analysis, 593
 - in a function or procedure, 593
 - interprocedural, 593
- data flow coverage, 471
- data flow error, 58
- data flow graph, 566, 572, 574, 577, 580, 582, 601
 - derived from control flow graph, 566, 567
 - example, 567
 - paths, 568
- Data flow testing, 74
- data flow testing, 630
 - classes, 593
 - effectiveness, 593
 - Java server pages, 593
 - numerical programs, 593
 - WEB applications, 593
- data flow testing tools, 594
- data-based coverage, 594
- database systems, 588
- DBMonster, 28
- DBUnit, 473
- DC: Decision Coverage, 545
- DDG, *see* Data Dependence Graph
- DDG: Dynamic Dependence Graph, 449
- deadline
 - missed, 321
- debugging, 34, 36, 54, 76, 521
- DECchip 21164, 595
- decision, 532
 - covered, 534
 - implied in **switch**, 534
 - outcomes, 533
 - undefined outcome, 533
- decision coverage, 534, 536
 - domain, 536
 - example, 535
 - fault detection effectiveness, 592
- decision table, 139, 177
 - applied to mutation, 688
 - as a matrix, 139
 - empty, 140
 - entry, 139
 - from cause effect graph, 139
 - generation example, 140
 - generation procedure, 139–140
- deep history connector, 255, 298
- def set, 567
- def-clear path, 568, 577, 600
 - existence, 570
- def-use
 - alternating, 570
 - error detection example, 583
 - reduction, 571
 - sequence, 570
- def-use chain, 470, 570
 - example, 570, 571
- def-use graph, *see* data flow graph, 571, 580
- def-use pair, 569, 570, 593
 - example, 570
- defect
 - classification, 53
 - density, 48, 49
 - discovery, 53
 - management, 53
 - open, 53
 - prediction, 53, 91
 - prevention, 46, 53
 - recording and reporting, 53
 - reported after release, 45
 - resolution, 53
 - saturation, 49
 - severity, 48
 - statistical techniques for prediction, 53
 - types, 53
- defect discovery
 - rate, 596
- defects, 27
 - cumulative count, 49
- definition
 - alternating, 570, 571
 - global, 567
 - output variable, 572
- definition clear path
 - see* def-clear path, 571
- definition clear sub-path, 578
- definition killed, 58
- definition-use, 463
- definition-use pair, 459
- definitions, 564
- DeJaVOO, 472
- DeJaVu, 472
- delay
 - between successive events, 347
 - catenations, 365
 - selection, 347

-
- delay connected state
 - distinguishing sequence, 363
 - delay sequence set, 366
 - delay transitions, 355
 - delays
 - use of clocks, 329
 - Dell Dimension Series, 389
 - DeMillo, Richard A., 629
 - dependence graph, 60, 95
 - static, 470
 - dependency modeling, *see* cause-effect graphing
 - deposit, 433
 - DES: Distinguishing Execution Sequence, 638
 - design
 - balanced, 397
 - high level, 99
 - unbalanced, 412
 - design phase, 181
 - design testing, 80
 - design verification test, 181
 - developer, 28
 - development cycle, 52
 - devious program, 508
 - DFT: Design For Testability, 91
 - dial-up connection, 383
 - Dieseltest, 28
 - diff, 470
 - DIGDEC machine, 184, 186, 188, 189
 - digital camera, continuous mode, 378
 - digital cross connect, 39
 - directed graph, 185
 - disjunctive normal form, 148
 - distinguished lady, 620
 - distinguished professor, 620
 - distinguishing sequence, 204, 231
 - brute force procedure, 246
 - minimal, 202
 - distinguishing sequence method, 241
 - distinguishing signature, 231
 - example, 232
 - divide and conquer strategy, 172
 - DMT: Dynamic Mutation Testing, 684
 - DNA codon, 513
 - DNA sequence, 494
 - DNA sequencing, 403
 - DNF partitioning, 497
 - DNF: Disjunctive Normal Form, 148
 - dog food dispenser, 176
 - dog sitter, 177
 - dogs: Raja and Shaan, 485
 - Dom, *see* dominator
 - domain coverage, 624
 - domain propagation, 498, 514
 - Domain testing, 74
 - domain testing, 172
 - dominator, 67, 94
 - algorithm, 88
 - equations, 94
 - example, 68
 - immediate, 68
 - relation, 68
 - strict, 67
 - tree, 68
 - dominator trees, 596
 - dominators, 471, 596
 - DPPS: Digital Plant Protection System, 685
 - dragon book, 88
 - DS: Distinguishing Sequence, 241
 - dynamic dependence graph
 - construction, 449
 - dynamic path condition, 470
 - dynamic slice, 448, 451, 456, 457, 459, 469
 - computation, 449
 - precise, 469
 - precision, 449
 - statement addition, 455
 - statement deletion, 455
 - dynamic slicing, 448
 - for feature location, 477
 - dynamic slicing algorithm, 469
 - E constraint, 134
 - e-Tester, 472
 - Echelon, 28, 471
 - edge, 566
 - outgoing, 570
 - edge coverage, 471
 - EDSAC computer, 88
 - effect, 133
 - identification, 137
 - effort, 47
 - EFSM: Extended Finite State Machine, 244
 - Eggplant, 28
 - elementary data context, 572, 581
 - covered, 581, 582
 - cumulative coverage, 582
 - example, 572
 - feasible, 602
 - variable definitions, 573
 - elementary data context coverage, 581
 - elementary ordered data context
 - coverage, 582
 - elevator
 - control, 50
 - hoist motor, 50
 - response, 50

- scheduling, 50
- eLoadExpert, 28
- embedded
 - computer, 183
 - system, 183
- embedded environment, 467
- embedded microprocessor, 177
- embedded software, 432
- embedded system
 - coverage measurement, 589
 - memory constraints, 589
 - real time, 252
- emergency controller, 321
- emergency shutdown, 608
- emergency shutdown controller, 330
- emergency shutdown system, 353, 355
- empirical studies, 676
- Empirix, 472
- employee record, 79, 100
- End node, 568
- engine control, 183
- engine temperature, 328
- Enter button, 44
- entry point, 61, 530
- environment, 183
- environment object, 128
- environmental disaster, 609
- equality symbol ($=$), 146
- equivalence
 - class, 101
 - infeasible, 113
 - partitioning, 101
 - product, 118
- equivalence class, 103–106, 108, 113, 121, 174
 - arrays, 109
 - boundary, 120
 - combining, 112, 125
 - compound data, 109
 - compound data type
 - example, 109
 - cover, 119
 - discard infeasible, 115
 - don't care values, 117
 - enumerations, 109
 - explosion, 118
 - infeasible, 115
 - large number of, 118
 - of clock valuation, 348
 - output, 124
 - partitioning, 112
 - range, 107
 - strings, 108
 - testable, 116
 - unmanageable size, 113
- equivalence class partitioning, 286
 - se equivalence partitioning, 102
- equivalence partitioning
 - example, 111
- Equivalence partitioning, 74
- equivalence partitioning, 90, 96, 103, 171, 172, 348, 383
 - effectiveness, 103
 - example, 113
 - GUI, 119
 - guidelines, 107
 - multidimensional, 110
 - procedure, 112–113
 - to partition input domain, 497
 - unidimensional, 110, 384
- equivalence relation
 - between time assignments, 335
 - reflexive, 174
 - symmetric, 174
 - transitive, 174
- equivalent mutant, 629
 - co-evolutionary approach, 686
 - count, 686
 - detection using sound, 686
 - hard to identify, 686
 - identification, 686
 - relation to infeasible path problem, 686
- error, 27
 - action, 188
 - data flow, 594
 - masking, 170
 - removal, 34
- error detection
 - extra state, 210
 - process, 210
 - transfer error, 211
- error free program, 521
- error message, 127, 175
 - as an effect, 133
- error recovery, 321
- error revealing test, 166
- error seeding, 695
- error studies, 591
- errors, 25, 89
 - code-based classification, 684
 - escaped, 595
 - of T_{EX} , 684
 - semantic, 25
 - syntax, 25
- ESG: Event Sequence Graph, 244
- essential branch, 590
- Estelle, 190, 509

-
- Euler's conjecture, 423
 - proven incorrect, 423
 - event
 - arrival time, 347
 - asynchronous, 258
 - completion, 261
 - en(S), 258
 - ex(S), 258
 - expression, 259
 - external, 251, 266, 440
 - internal, 251, 258, 274
 - prune internal, 277
 - scheduling conflicts, 309
 - special, 259
 - timer, 259
 - tm, 259
 - event buffering, 328
 - event disposition, 309
 - event driven system, 89
 - event pair
 - input-output, 245
 - event scheduling, 309
 - event sequence graph, 244
 - events, 258–259, 325
 - external, 272
 - executable entity, 463
 - executable slice, 469
 - execution
 - history, 54
 - exponentiation program, 54
 - space, 54
 - slice, 54
 - trace, 54
 - execution history
 - partial, 54
 - execution sequence
 - effect of, 638
 - example, 636
 - execution slice, 439, 447, 457
 - execution trace, 438, 440, 449, 456
 - collection, 438
 - example, 439
 - exhaustive testing, 385
 - exit, 433
 - exit point, 61, 530
 - expected behavior, 31
 - expected response, 208
 - exploratory testing, 82
 - Explorer, 389, 404
 - exponentiation program, 61
 - flow graph, 65
 - extra Boolean variable, 179
 - extra state error, 247
 - f, 151
 - factor, 384
 - binary, 410
 - combination, 385
 - don't care, 392
 - identification, 388
 - levels, 385
 - factor combination, 389
 - infeasible, 387
 - procedure for generation, 404–407
 - selecting a subset, 396
 - factor level
 - unused, 386
 - used, 386
 - factorial, 527
 - factors
 - for a browser, 386
 - for sort, 386
 - in a GUI, 386
 - Fahrenheit scale, 82
 - failure, 27
 - network, 128
 - failure count, 84
 - failure indicating metric, 470
 - false, 147
 - FAST format, 504
 - fault, 27
 - arithmetic expression, 148, 152
 - Boolean operator, 148, 150, 152, 153
 - Boolean variable, 150
 - clock reset, 371
 - clock reset in timed automata, 342
 - communication, 321
 - coverage, 197, 198
 - dynamic binding, 670
 - extra Boolean variable, 150
 - extra state, 194, 371
 - incorrect Boolean operator, 559
 - incorrect loop termination, 625
 - incorrect reset, 371
 - incorrect transfer, mapped to NTFSM, 371
 - interaction, 385, 391, 422
 - mapping to NTFSM, 371
 - missing assignment, 694
 - missing Boolean variable, 150, 179
 - missing condition, 520, 559, 696
 - missing condition, detection using mutation, 630
 - missing initialization, 694
 - missing initialization, detection using static analysis, 694
 - missing reset, 371
 - missing state, 195, 247, 371
-

- missing statement, 625
- multiple, 149
- multiple state table, 244
- off-by- ϵ , 149
- off-by- ϵ^* , 149
- off-by- ϵ^+ , 149
- omission of history connector, 316
- operation, 194, 371
- pairwise interaction, 391
- polymorphism, 670
- power surge, 685
- precedence, 646
- propagation, 166
- real, 684
- relational operator, 148, 152
- sequencing, 195
- simple, 391
- single, 149
- software simulation of hardware, 685
- t-factor, 391
- t-way interaction, 391
- timing, 321, 322
- timing constraint in timed automata, 345
- timing constraint restriction, 381
- transfer, 194
- transfer in timed automata, 342
- vector, 392
- fault classes, 148
- fault detection, 590
 - comparison example, 238
 - incorrect transition in a statechart, 305
 - missing transition in a statechart, 305
 - tests derived from statechart, 305
 - using mutation, 630
 - using timed WP-method, assumptions, 369
- fault detection effectiveness, 101, 585, 591
 - analytical study, 594
 - branch and data flow, 594
 - data flow, 593
 - data flow and mutation, 594
 - of combinatorial designs, 422
 - of minimized tests, 471
 - of W-method, 237
 - of Wp-method, 237
- Fault diagnosis of computer programs, 680
- fault injection
 - hardware, 685
- fault model, 51, 171, 173
 - for finite state machines, 194
 - for timed automata, 342
 - inheritance, 682
 - polymorphism, 682
 - subtyping, 682
- fault propagation
 - example, 166
- fault revealing test, 166
- fault vector
 - example, 392
- fault-injection
 - run-time, 605
- faults
 - mixed kind, 559
- faulty predicate, 153
 - indistinguishable, 153
- FCT: Full Compound Transition, 264
- features advertised, 77
- FIND program, 607, 682, 695
- finite state machine, 44, 55, 183, 185
 - 1 distinguishable, 240
 - abstract, 312
 - complete, 198
 - completely specified, 188, 205
 - connected, 205
 - core, 220
 - core behavior, 219
 - deterministic, 205
 - equivalence, 188
 - exciting, 207
 - initial state, 205
 - k-equivalence, 189
 - minimal, 189, 198, 205
 - strongly connected, 188
 - tabular representation, 187
 - V-equivalence, 188
- finite state model, 59
- Firefox, 404
- firewall, 469
- firewall testing, 469
- firmware
 - for tape controller, 273
- flight controller, 183
- flow
 - of control, 530, 563
 - of data, 563
- flow graph, 60, 67
 - reverse, 88
- flow of control, 60
- FogBugz, 53
- Foreign language testing, 77
- fork, 257
- fork connector, 302
- formal grammar, 172
- formal method, 55
 - nuclear generating station, 607
- formal model, 55
- formal proof, 591

-
- formal specification, 509
 - document, 98
 - in Z, 480
 - Fortran, 624, 683
 - Fortran code, 607
 - frames per second, 378
 - FSM, *see* finite state machine
 - derived from statechart, 267
 - excitation, 186
 - FSM testing, 74
 - full predicate coverage, 592
 - function call, 62, 441, 567
 - function coverage, 472, 589
 - function trace, 477
 - functional requirements, 518
 - functional testing, 90, 586–588
 - Java programs, 593
 - Functional Tester, 472
 - functional unit, 127–129

 - Georgia Tech, 470
 - giveaway items, 136
 - glass-box testing, 586
 - global definition, 566
 - global time, 353
 - glutamine, 504
 - GNU-C compiler, 687
 - Goldsmiths College, 665
 - goodbye message, 527
 - goodness of a test set, 586
 - GOTCHA, 595
 - GOTO statement, 94
 - graph, 60
 - graphical notation, 55
 - graphical user interface, 104, 244
 - greedy algorithm, 461
 - grid automaton, 354, 381, 382
 - construction example, 357
 - example, 355
 - generation procedure, 355–357
 - initial state, 355
 - grid set, 376
 - Grinder, 28
 - grocery item, 103, 104
 - guard, 67, 254, 355
 - as conditions, 286
 - clock, 327
 - non-equality, 327
 - none, 328
 - GUI, 146, 530
 - boiler control, 113
 - design, 120
 - dissection from application, 119
 - factors, 386
 - for reduction in test cases, 120
 - front end, 119
 - invalid inputs, 113
 - test generation using factors, 390
 - testing, 113
 - GUI modeling
 - using FSM, 245
 - GUI testing, 28, 82

 - Halstead metrics, 47
 - Halstead, Maurice, 47
 - Hamlet’s program, 694
 - hand calculation, 43
 - hardware configuration
 - Dell Dimension Series, 387
 - hardware performance monitoring, 595
 - hardware platform, 383
 - Harvard University, 89
 - HAZOP: Hazard and Operability, 683
 - HDL: Hardware Description Language, 595
 - heart condition, 413
 - heart pacemaker, 251
 - heart rate, 413
 - heart simulator, 432
 - heavy water, 607
 - High Energy Transient Explorer (HETE-2), 592
 - HighTest Plus, 472
 - hints paper, 680
 - HIOA: Hybrid I/O Automata, 376
 - history connector, 297
 - deep, 298, 316
 - omission fault, 316
 - shallow, 316
 - history state, 254
 - deep history, 254
 - example, 255
 - shallow history, 254
 - example, 254
 - home video, 44
 - human disaster, 609
 - human errors, 25
 - human tester, 75
 - hybrid systems, 243
 - modeling using HIOA, 376
 - hypermedia application specification, 310

 - I constraint, 134
 - IAR Systems, 310
 - IBM, 80
 - IBM AS/400, 90
 - IBM Rational PurifyPlus, 592
 - IBM Rational Rose, 310

-
- IBM/Rational, 58, 472
 - identifier
 - external, 639
 - identity matrix, 43
 - IDL: Interface Definition Language, 75
 - IEEE, 49
 - if statement, 69
 - ignore action, 327
 - iLogix, 310
 - incorrect program
 - execution of, 562
 - incorrect scoping in C, 626
 - India, 431
 - inductance, 595
 - infeasibility and reachability, 561
 - infeasible
 - basic block, 567
 - combinations, 132
 - condition, 552
 - constraint, 150
 - decision, 561
 - elementary data context, 602
 - use, 580
 - example, 580
 - infeasible data, 113
 - infeasible regions
 - identification of, 379
 - Infragistics, 472
 - input
 - failure causing, 47
 - illegal, 102, 109
 - invalid, 33, 103, 120
 - isolation, 127
 - legal, 100, 102, 107, 109
 - pallette of valid, 113
 - periodic, 324
 - regions, 102
 - unintended, 78
 - valid, 33
 - input alphabet, 205, 324
 - input domain, 32, 100, 104, 107, 110, 384, 496
 - astronomical, 87
 - complexity, 100
 - derivation, 99
 - identification, 112, 114
 - of sort, 100
 - overlap, 105
 - partition, 103, 125, 174
 - partitioning, 468
 - partitioning valid, 497
 - size, 100
 - source of, 99
 - subdivision, 101
 - true, 114
 - valid, 493, 494
 - input event
 - ?, 336
 - input generator, 44, 45
 - input identified state, 362
 - input space, 384
 - invalid, 78
 - modeling, 388
 - valid, 78
 - input trace, 268
 - input variable
 - interaction, 384
 - inputs
 - sort, 33
 - inspection, 57
 - installation testing, 77
 - integer programming, 468
 - integration testing, 685
 - integration tests, 588
 - Intel iPSC/2 hypercube, 687
 - Intel processor, 78
 - inter-level transitions, 290
 - interaction fault, 391, 422
 - example, 391
 - higher level, 393
 - realistic situations, 422
 - triggered, 391
 - interaction testing, 384
 - interface mutation, 605, 683
 - interface testing, 74, 75
 - intermediate node, 143
 - intermediate result, 133
 - interrupt, 254
 - IPO
 - example, 417–421
 - horizontal growth, 414
 - vertical growth, 414
 - IPO algorithm, 423
 - IPO: In Parameter Order, 414
 - Itanium 2, 595
 - IUT
 - alphabet, 369
 - apply reset input, 191
 - bringing to start state, 191
 - clock relationship assumption, 370
 - coverage completeness, 369
 - deterministic, 369
 - extra state, 208
 - interaction with test harness, 353
 - model, 369
 - non-conformance, 190
 - number of states, 369

-
- number of states, same as in specification, 210
 - observation under timing tests, 351
 - reset, 191
 - reset operation, 369
 - testing against control specification, 191
 - testing problem, 193
 - IUT: Implementation Under Test, 182
 - J-FuT, 593
 - J2EE: Java 2 Enterprise Edition, 80
 - Java, 47, 54, 79, 90, 109, 251, 470, 472, 512, 513, 530, 564, 664, 696
 - byte-code, 592
 - condition evaluation, 560
 - coverage measurement tool, 592
 - inheritance mechanism, 664
 - java, 697
 - Java applet, 427
 - JBlanket, 592
 - JDiff, 470
 - Jester, 665
 - JJ paths, 592
 - Jmeter, 28
 - JML, 509
 - JML: Java Modeling Language, 512
 - join, 256
 - joint connector, 303
 - joke, 93
 - jump-tojump paths, 592
 - junction connector, 256
 - expansion, 313
 - JUnit, 471, 472, 592
 - k-distinguishable, 199
 - k-dr analysis, 571
 - k-dr chain, 571, 578
 - coverage example, 578
 - k-dr coverage, 601
 - k-dr interaction, 570
 - k-equivalence, 199
 - k-equivalence partition, 199
 - convergence of construction process, 201, 246
 - example, 199–201
 - uniqueness, 246
 - Kemerer, Chris F., 49
 - kernel mode, 592
 - keyboard, 119
 - kill process signal, 233
 - KLOC: Kilo, or thousand, Lines of Code, 45
 - Klockwork, 58
 - Korat, 512
 - Korea Electric Power System, 685
 - Kripke structures, 60
 - label
 - edge, 186
 - expansion, 292
 - multiple, 186
 - lamp, 183
 - language, 72
 - catenation, 72
 - finite, 72
 - procedural, 530
 - timed, 329
 - LAPSE: Lightweight Analysis for Program Security in Eclipse, 58
 - Latin alphabet, 393
 - Latin square, 424
 - Latin squares, 389, 393, 423
 - example, 393
 - isomorphic, 394
 - mutually orthogonal, 393
 - Lava, 665, 676, 682
 - lazy evaluation in C, 560
 - LCSAJ, 591
 - adequate test
 - error revealed, 564
 - example, 542–544
 - exercised, 542
 - fault detection effectiveness, 591
 - traversed, 542
 - LCSAJ: Linear Code Sequence and Jump, 542
 - LCSJ criterion, 583
 - LD relations, 89
 - leaf node, 148
 - left rotation, 656
 - leftmost digit, 127
 - Leonhard Euler, 395
 - linear code sequence and jump, 542
 - Linux, 592
 - Lipton, Richard J., 629
 - liquid container, 150
 - literal, 148
 - live definitions, 572
 - load testing, 78, 79
 - LoadRunner, 28
 - logical context negation, 644
 - logical operators: **and**, **or**, **xor**, 536
 - login, 433
 - loop, 65
 - boundary test, 590
 - infinite, 435
 - interior test, 590
 - loop paths, 67
 - LOTOS, 190, 511
-

-
- LUSTRE, 376
 - Mac, 404
 - Mac OS, 384
 - machine equivalence, 188
 - maintenance, 76
 - Malicious Programmer Hypothesis, 692
 - Marathon, 28
 - matrix
 - inversion, 43
 - multiplication, 42–44
 - MC adequacy, 546
 - MC/DC, 591
 - adequate test set, 584
 - MC/DC adequacy, 611
 - error detection, 560
 - error not revealed, 564
 - minimal tests, 558
 - MC/DC adequate test, 563
 - automatic generation, 592
 - error not detected, 559
 - size, 549
 - MC/DC coverage, 534, 545, 550, 552, 583, 592, 694, 695
 - example, 546, 552
 - fault detection effectiveness, 592
 - MC part, 550
 - MC/DC coverage criterion, 550, 697
 - MC/DC criteria, 604
 - MC/DC criterion, 86, 95
 - error detection, 599
 - MC/DC testing, 630
 - MC: Multiple condition, 545
 - McCabe, Thomas, 47
 - McIlroy, Douglas, 470
 - Mealy machine, 184
 - meaning impact strategy, 173
 - measurement, 45
 - medical device, 413, 432
 - Megellan, 592
 - memory constraint
 - how to overcome, 589
 - memory corruption, 592
 - memory leak, 51, 592
 - memory overhead, 596
 - menu
 - pulldown, 104
 - menu driven application, 41
 - Mercury, 472
 - method
 - in Java, 693
 - overloading, 671
 - method coverage, 530, 589
 - method coverage in context, 530
 - methods
 - untestable, 592
 - metric, 45
 - for project monitoring, 49
 - organizational, 45
 - product, 47
 - product quality, 45
 - project, 46
 - static, 49
 - metrics
 - complexity, 53
 - dynamic, 49
 - Halstead, 47
 - predictive ability, 90
 - software science, 47, 94
 - MI: Meaning Impact, 173
 - microcontroller, 322
 - microprocessor-memory bus, 589
 - Microsoft, 471, 472, 592
 - microwave oven, 258, 259
 - example, 192
 - input variables, 268
 - output variables, 268
 - statechart, 268
 - test generation, 283
 - MIDI: Musical Instrument Digital Interface, 190
 - milk, 104
 - Miller, Keith, 695
 - minimal covering set, 478
 - minimal FSM, 312
 - minimal machine, 246
 - minimality
 - of FSM, 219
 - Miranda, 697
 - missing initialization, 694
 - missing state error, 247
 - mistakes
 - by C programmers, 626
 - by Java programmers, 625
 - common programming, 626
 - overloading related, 671
 - MIT, 511
 - mixed level covering array
 - example, 417–421
 - generation, 414
 - mixed level covering arrays, 412
 - generation procedure, 414–417
 - mixed level orthogonal array
 - example, 409
 - model
 - finite state, 59
 - model based testing, 421
-

-
- using UPPAAL, 377
 - model checker, 59, 60
 - termination, 60
 - model checking, 59, 60, 90, 592
 - model-based testing, 59, 60, 75, 87
 - moderator, 58
 - modification traversing tests, 437, 447, 470
 - module prioritization for inspection, 59
 - molecular strings, 470
 - MOLS
 - complete set, 395
 - existence, 395
 - maximum number, 424
 - procedure for construction, 395–396
 - shortcomings in the generation of test configurations, 407
 - survey of constructions, 423
 - MOLS: Mutually Orthogonal Latin Squares, 395
 - monitor, 136
 - Moore machine, 184, 324
 - Morel, Larry, 695
 - Mosaic, 389
 - moth, 89
 - Mothra, 676, 687
 - Mozilla, 404
 - MPH: Malicious Programmer Hypothesis, 692
 - MsGAT, 676, 690
 - MuJava, 633, 665, 676, 695, 696
 - μ Java
 - see MuJava, 695
 - multi-level event coverage, 595
 - multidimensional partitioning, 110, 112, 175
 - exponential increase, 112
 - multiple condition coverage, 539, 540, 545
 - example, 540
 - multiple indirection operator, 647
 - multiple UIO sequences, 244
 - multiplexer, 51
 - multiprogramming system, 89
 - MUMCUT strategy, 173
 - numerical program, 695
 - musical instruments, 190
 - musical note, 470
 - μ SZ, 511
 - mutagenic operator, *see* mutation operator
 - mutant
 - of FSM, example, 196
 - analysis, 627
 - behavior same as parent, 610
 - classification, 616
 - compiling, 627
 - conditions for distinguishing, 621
 - considered distinguished, 621
 - container, 687
 - distinguishing conditions example, 621
 - distinguished, 195, 610, 615, 620
 - drastic, 607
 - equivalent, 618, 621
 - error hinting, 632
 - error revealing, 632
 - estimating number, 680
 - execution, 616, 627
 - execution on a vector processor, 686
 - extinguished, 620
 - first order, 501, 606, 629
 - first order, count, 197
 - fourth order, 682
 - generation, 614
 - higher order, 606, 676
 - higher order example, 606
 - instability, 680
 - killed, 620
 - live, 615, 617, 618
 - meta, 687
 - no change in behavior, 609
 - non-equivalent, 629
 - nth order, 606
 - of formal contract, 690
 - of FSM, 195, 246
 - of Java using concurrent threads, 687
 - order of execution, 691
 - parent, 609
 - reduction in compilation time, 687
 - reliability indicating, 632
 - scheduling on hypercube, 687
 - scheduling on parallel transputer, 686
 - schema, 500, 687
 - second order, 629, 682
 - semantic, 90
 - similar, 686
 - smallest number, 691
 - specification, 515
 - syntactic, 90
 - syntactic validity, 651
 - syntactically correct, 625
 - syntactically correct program, 605
 - terminated, 620
 - third order, 682
 - mutant container
 - compared with mutant schema, 687
 - mutant equivalence
 - specification, 501
 - undecidable problem, 629
 - under strong mutation, 629
 - under weak mutation, 629
 - mutant generation, 622

-
- mutant operator, *see* mutation operator
 - mutation, 591
 - access to source code, 604
 - act of slightly changing a program, 605
 - analysis, 604
 - applied to Z specifications, 689
 - arrays, 661, 663
 - as black-box technique, 604
 - as white-box technique, 604
 - break statement, 653
 - comparison with random testing, 685
 - compiler integrated, 687
 - condition, 649
 - constant, 625
 - constrained, 679
 - continue, 654, 655
 - continue by break, 653
 - cost of, 677
 - discovery of subtle flaws, 611
 - do-while statement, 654
 - domain traps, 663
 - effectiveness compared with all-uses, 684
 - effectiveness compared with data flow, 684
 - engine, 683
 - entities not mutated, 634
 - example, 605
 - examples in COBOL, 680
 - examples in Fortran, 680
 - examples in Lisp, 680
 - fault detection studies, 684
 - goto label, 652
 - high level languages, 604
 - hints paper, 680
 - interface, 75, 83, 605, 683
 - interface for CORBA, 683
 - linearization, 634, 635
 - example, 634
 - list of tools, 676
 - move brace, 657
 - multiple trip, 655
 - of C programs, 604
 - of concurrent Java programs, 683
 - of CSP specification, 690
 - of finite state machine specification, 690
 - of Fortran programs, 604
 - of Java programs, 604
 - of sqrt to sqr, 695
 - operator, 625
 - pointers, 662
 - program constants, 642
 - reduced linearized sequence, 634, 635
 - relation to software reliability, 680
 - scalability, 607
 - score, 612
 - semantic change, 607
 - sequence operator, 656
 - serial algorithm, 687
 - specification, 500
 - statement, 651
 - statement deletion, 650
 - statement mutation, 625
 - strong, 610
 - structures, 661, 662
 - structures, example, 662
 - switch statement, 658
 - systematic procedure for scenario generation, 611
 - technique for test assessment and enhancement, 604
 - testing, 604
 - the idea, 680
 - to study effectiveness of other techniques, 684
 - trap statement execution, 648
 - treatise, 680
 - tutorial articles, 681
 - twiddle, 663
 - use in assessment of other techniques, 685
 - variable, 625, 660
 - weak, 610
 - what is not mutated, 633
 - while statement, 654
 - why?, 611
 - XML, 683
 - mutation adequacy
 - compared with equivalence partitioning and boundary value analysis, 689
 - mutation analysis
 - fast using multiple-SIMD, 687
 - mutation operator, 52
 - ABS, 623
 - accessor modifier example, 673
 - AMC, 625, 626
 - binary, 643
 - bitwise negation, 645
 - cast operator replacement, 647
 - casts not mutated, 648
 - categories, 624
 - CCCR example, 643
 - CCR, 623
 - class, 665
 - class related for Java, 665
 - comparable replacement, 643
 - complete set for C, 633
 - CRCR, 693
 - CRCR example, 642
 - dependence on language, 625

-
- design, 626
 - design guidelines, 627–628
 - domain and range, 643
 - domain of variable, 625
 - effectiveness, 628
 - for Ada, 682
 - for C, 625, 682
 - for COBOL, 682
 - for dynamic binding, 670
 - for Fortran, 682
 - for Java, 625, 682, 693
 - for polymorphism, 670
 - for Python, 682
 - generative device, 622
 - ideal set, 627, 691
 - incomparable replacement, 643
 - increment and decrement, 644
 - indirect operator precedence, 646
 - inheritence, 666
 - IOP, 693
 - Java specific, 672
 - language specific, 624
 - logical negation, 644
 - manufacturing variables, 625
 - meta, 688
 - method overloading, 671
 - method overloading example, 671
 - minimality, 628
 - OAN, 695
 - OCOR example, 647
 - poor naming choice, 692
 - precise definition, 628
 - representativeness, 627
 - ROR, 623
 - simple mistakes, 624
 - SMTc example, 656
 - SMTT example, 655
 - SMVB example, 657
 - SRSR example, 651
 - SSOM example, 656
 - SSWM example, 658
 - STRI example, 649
 - STRP, 624
 - STRP example, 648
 - subset, 678
 - traditional for Java, 665
 - unary operator, 644
 - VDTR for C, 624
 - VLSR in C, 630
 - VTWD example, 664
 - Z specification, 686
 - mutation operators
 - count of, 674
 - Java, 683
 - mutation score
 - 0, 619
 - 1, 619
 - computation of, 619
 - cost of high, 680
 - golden, 619
 - mutation testing, 51
 - algebraic specification, 688
 - class prioritization, 678
 - cost, 627
 - dynamic, 683
 - for novice, 679
 - incremental, 620, 686
 - of concurrent threads, 687
 - of CTL logic, 690
 - of decision table specification, 688
 - of Simulink models, 690
 - of specifications, 688
 - of statechart specification, 689
 - on massively parallel architectures, 687
 - predicate calculus, 688
 - reduction in cost, 686
 - speeding up, 686
 - tools, 676
 - within budget, 677
 - n-distinguishable, 239
 - n-switch, 240
 - n-switch cover, 239, 250
 - NAND gate, 51
 - Naur's program, 591
 - Ncube machine, 687
 - NDFSM, 185
 - Netscape, 384, 404
 - network connection, 383
 - network server, 192
 - networked environment, 176
 - NFA, 185
 - NFSM, 185
 - NIST: National Institute of Standards and Technology, 91
 - node, 566
 - data, 69
 - descendent, 65
 - End, 65, 67, 68
 - internal, 148
 - predecessor, 94
 - predicate, 69, 71
 - reference, 438
 - Start, 65, 67
 - successor, 65
 - Node type, 143
-

-
- non-core input, 219
 - non-deterministic FSM, 185
 - non-embedded application, 457
 - non-embedded systems, 588
 - non-full compound transition, 307
 - expansion to full compound, 281
 - non-singular expression
 - conflicts, 179
 - NOT-node, 148
 - NP hard, 471
 - NTFSM, 354, 358
 - construction from grid automaton, example, 360
 - delay connected state, 362
 - from grid automaton, 359
 - input identified state, 362
 - observable, 359
 - NTFSM trace, 362
 - NTFSM: Non-deterministic Timed Finite State Machine, 323
 - nuclear plant, 607
 - nuclear plant control, 52
 - nucleotide, 485
 - null output, 219

 - O constraint, 134
 - object coverage, 589
 - Object-Z, 89, 512
 - OBJTEST, 688
 - observation
 - external, 610
 - internal, 610
 - points of, 610
 - ODC, 91
 - ODC: Orthogonal Defect Classification, 53
 - office tools, 588
 - ON-BRIGHT, 183
 - onto set product: $\otimes$, 154
 - OO metrics, 49
 - software quality, 90
 - OO testing, 79, 80
 - open systems, *see* non-embedded systems
 - operating system, 383
 - operation
 - valid range, 506
 - operation error, 194
 - operational correctness, 77
 - operational profile, 30, 35, 47, 85
 - operator mutation, 625
 - optimizing compiler, 468
 - OR state, 252, 293
 - nested, 298
 - OR-node, 144, 145, 148

 - oracle, 42–44, 79, 89
 - automation, 93
 - construction, 44
 - construction using effects, 166
 - example, 44, 506
 - human, 42, 43
 - passive C++, 89
 - oracle creation, 494
 - ordered data context, 573, 593
 - coverage, 582
 - covered, 581
 - ordered data context coverage, 602
 - ordered elementary data context, 573, 581
 - covered, 581
 - example, 573
 - orthogonal arrays, 389, 408
 - alternate notation, 409
 - design, 422
 - example, 408
 - for a pacemaker, 413
 - index of, 409
 - mixed, 389
 - mixed level, 409
 - run of, 408
 - strength of mixed level, 409
 - web site, 425
 - orthogonal component, 298
 - Orthogonal Defect Classification, 53
 - OS bridge, 387
 - oscillation diversity, 595
 - output conditions, 132
 - output distinguishability, 318
 - output event
 - !, 336
 - output function, 185, 198
 - output table, 199
 - Oxygen, 622

 - p-use, 567, 569, 576
 - adequacy, 586
 - adequacy does not imply c-use adequacy, 586
 - coverage, 576, 584
 - coverage example, 577
 - feasible, 600
 - infeasible, 580
 - minimization, 572
 - number of, 575
 - p-use: predicate use, 565
 - pacemaker, 432
 - orthogonal array, 413
 - pairwise coverage, 397
 - pairwise design, 397
 - construction using MOLS, 401–403

- multivalued factors, 401
- procedure for generation, 398
 - example, 398
- pairwise interaction fault, 391
 - due to missing condition, 392
 - revealed, 391
- Pairwise testing, 74
- pairwise testing, 74, 384
- parallelizing compiler, 468
- parameter
 - same as factor, 414
- Parasoft, 472
- Pareto-like distribution, 595
- partial string concatenation operator: $\otimes$, 212
- partial-W method, 183, 244
- partition analysis, 171
- Partition testing, 74
- partition testing
 - analysis, 172
 - applied to Z specifications, 172
 - does not inspire confidence, 171
 - fault detection effectiveness, 172
 - proportional, 171
- path, 63–67
 - as regular expression, 591
 - data flow, 250
 - deciding infeasibility, 630
 - def-clear, 568
 - definition clear, 571
 - distinct, 67, 93
 - enumeration, 597
 - example, 64
 - execution, 54
 - exponential, 65, 66
 - infeasible, 526, 531, 580
 - example, 525
 - infinite number of, 590
 - invalid, 65
 - length, 67, 93
 - loops, 66
 - number example, 65
 - number of, 65, 523
 - shortest, 232
 - traversal, 571
 - traversal at least once, 590
 - tricky, 529
- path coverage criterion, 521, 525, 529
- path spectra, 470
- path spectrum, 470
- paths, 520
- payroll processing, 79, 100
- PC, 404
- PDG, *see* program dependence graph
- PDom, *see* post-dominator
- PDS
 - input space, 385
- PDS: Pizza Delivery Service, 385
- penetration testing, 77
- Pentium, 388
- performance, 30, 434
- performance analysis, 466
- performance testing, 78
- periodic table, 486, 513
- Pester, 676
- Petri net, 55, 98, 99, 181
- physical systems, 243
- PIE
 - analogous to DMT, 684
- PIE: Propagation, Infection, and Execution, 681
- pin code, 431
- Pizza, 385
 - toppings, 385
 - variety, 385
- pizza service
 - testing, 411
 - tests, 410
- planet name, 174
- PLAY button, 251
- PMothra, 687
- PMS: Pilot Mutation System, 680
- pointer variable, 565
- POKE-TOOL, 594
- post-dominator, 67, 69
 - algorithm, 88
 - example, 68
 - immediate, 68
 - relation, 68
 - strict, 68
 - tree, 68
- potential dependence, 451, 454, 477
- Pounder, 28
- pre-tests, 503
- precedence, 147
- predicate, 69, 146, 532
 - compound, 147
 - converted to Boolean expression, 148
 - entire, 151
 - non-singular, 169
 - simple, 147
- predicate constraint, 151
 - example, 151
- predicate logic, 56
- predicate node, 453
- predicate test generation
 - program based, 170
 - specification based, 168

-
- predicate testing, 146, 148
 - in practice, 168
 - prefix set, 364
 - premature termination, 435
 - prematurely terminating execution sequence, 636
 - price database, 104
 - primitive types in C, 675
 - printer, 104, 136
 - color laserjet, 105
 - manufacturer, 104
 - model, 104
 - networked, 384
 - prioritization algorithm, 466
 - probe placement, NP-complete, 596
 - probe test inadequacy, 588
 - PROBSUBSUMES relation, 689
 - procedural language, 530
 - procedure call, 62
 - process control system, 150
 - processor
 - Athelon, 388
 - Pentium, 388
 - product of variables, 147
 - product reliability, 47
 - product stability, 49
 - product type, 127
 - profiling
 - non-intrusive, 589
 - program execution, 589
 - program
 - analysis, 62, 73
 - behavior, 40, 41
 - conflict graphs, 683
 - continuously running, 192
 - correct behavior, 42
 - correctness, 52
 - devious, 508
 - error free, 521
 - execution, 39
 - execution of sequential, 542
 - exponentiation, 522
 - coverage domain, 524
 - test enhancement, 523
 - highly reliable using mutation, 684
 - initial state, 41
 - input-output mapping, 505
 - mutation, 56, 604
 - older, 530
 - review, 73
 - security relevant parts, 470
 - separate runs, 530
 - single execution, 529
 - state, 40
 - state diagram, 40
 - statement, 69
 - termination of execution, 542
 - under test, 42
 - unification, 686
 - advantages, 686
 - on SIMD, 686
 - untested parts, 527
 - variable, 54
 - verification, 52, 171
 - program based tests, 168
 - program dependence graph, 69, 71, 95, 448, 449, 468, 469
 - dynamic, 449
 - example, 71
 - program differencing, 470
 - during maintenance, 470
 - textual, 470
 - program graph, *see* control flow graph
 - program mutants, 75
 - program mutation, 52, *see* mutation
 - program optimization, 468
 - program revalidation, 428
 - program size, 47
 - program specification, 171
 - program spectra, 470
 - programmer, 29
 - programming mistakes, 673
 - prohibited pairs, 426
 - Prolog, 511
 - propagation domain
 - partition, 515
 - property, 129
 - property list, 129
 - Proteum, 633, 676, 696
 - Proteum-RS/PN, 690
 - Proteum/FSM, 245, 689
 - Proteum/IM, 676, 687
 - protocol, 98, 183
 - protocol testing, 244
 - pseudo state, 254–257
 - pseudo-random tests, 595
 - psychology in usability testing, 30
 - punched cards, 119
 - Purdue University, 633, 682
 - Purify, 58
 - Pythia, 472
 - Python, 676
 - quality
 - attribute
 - usability, 30
 - attributes, 29–30
-

-
- software, 29
 - quality attribute, 29
 - completeness, 29
 - consistency, 29
 - dynamic, 29
 - performance, 30
 - static, 29
 - quality attributes, 89
 - quasigroup, 424
 - Quicksort, 682
 - R constraint, 134
 - race
 - condition, 311
 - read-write, 311
 - write-write, 311
 - radiation leak, 25
 - RAISE, 509
 - RAM, 136
 - random request generator, 50
 - random testing, 28, 171, 508
 - comparison with mutation, 685
 - evaluation using mutation, 685
 - range
 - implicit specification, 107
 - multiple, 108
 - specification, 107
 - testing, 172
 - valid, 514
 - ranking
 - requirements, 438
 - RAP set, 649
 - RAX, 423
 - RC-4000 computer, 89
 - RCS: Revision Control System, 171
 - RDDG: Reduced Dynamic Dependence Graph, 456
 - reachability, 621, 682
 - condition, 622
 - example, 631
 - reaching definition, 453
 - reactive systems, 266
 - reactor monitoring system, 608
 - reader, 58
 - real-time constraints, 589
 - real-time software, 79
 - real-time system, 185, 321
 - as an embedded system, 322
 - hard, 321
 - soft, 321
 - real-time testing, 80
 - RealTime, 310
 - Rear Admiral Dr. Grace Murray Hopper, 89
 - record, 109
 - student, 109
 - recorder, 58
 - reference
 - global scalar, 639
 - global set example, 640
 - local scalar, 639
 - local set example, 640
 - multilevel, 639
 - scalar, 639
 - sets, 640
 - reflexive, 246
 - region coverage, 348
 - region graph, 340
 - regress, 428
 - regression test selection problem, 435
 - regression testable, 469
 - regression testing, 54, 56, 428–478
 - commercial tools, 466
 - corrective, 428
 - firewall approach, 470
 - handling declarations, 445
 - integration testing phase, 469
 - obsolete tests, 435
 - problem, 434, 435
 - progressive, 428
 - random selection, *see* random testing
 - retesting modified software, 468
 - revalidation, 430
 - safe, 436
 - system testing phase, 469
 - test all, 436
 - test minimization, 430
 - test prioritization, 437
 - test process, 430
 - test revalidation, 435
 - test selection, 430
 - using integer programming, 468
 - regular expression, 72, 591
 - regulatory organization, 608
 - relation, 103
 - relational expression, 147
 - relational operator, 147
 - testing, 172
 - relational operator fault, 179
 - RELAY model, 684
 - release decision, 47
 - relevant slice, 454
 - reliability, 30–31, 35
 - assessment, 77
 - estimated, 85
 - estimation, 35
 - high, 383
-

- testing, 77
- true, 84
- reliability estimation, 28
- reliable criterion, 591
- reliable software, 593
- reliable test, 89
- relop: relational operator, 147
- repeatability, 412
- required k-tuple, 593
- requirement
 - independence of conditions, 168
- requirement testing, 80
- requirements, 31–32, 55
 - ambiguous, 562
 - as ideas, 98
 - coverage, 587
 - familiarity, 101
 - for robustness, 507
 - functional, 78, 519
 - in English, 479
 - incomplete, 562
 - incorrect, 562
 - informal, 98
 - misunderstanding, 479
 - non-functional, 519
 - performance, 78
 - revealing ambiguities, 561
 - rigorous, 98
 - rigorously specified, 99
 - unambiguous specification, 103
 - using mathematical notation, 479
- reset input, 192, 219
- reset operation, 324
- residual coverage, 463, 464
- resistance, 595
- review plan, 57
- Richard Lipton, 680
- right associativity, 147
- RLS: Reduced Linearized Sequence, 635
- RNA codon, 513
- RNA sequence, 514
- robustness, 78
- robustness testing, 78, 79, 507
- role
 - developer, 28
 - tester, 28
- root node
 - of predicate, 166
- ROR operator, 675, 677, 679
- Rosetta, 687
- routing, 321
- RSML, 310, 509
- RTS: Regression Test Selection, 430
- run, 392
- runtime overhead, 596
- S-connector, 256
- S-module, 512
- Safari, 384
- safe regression test selection, 470
- safe techniques, 470
- safety hazard, 381
- SALT: Specification and Abstraction Language
 - for Testing, 422
- sampling granularity
 - computation, 351
 - example, 350
- satellite software, 592
- saturation effect, 84, 88, 91, 96, 596
 - impact on test process, 87
- saturation region, 85
- Sayword, F. G., 629
- SBATCG: Specification Based Adaptive Test
 - Generation, 685
- sBLAST, 504
- schema, *see* Z schema, 513
 - mutant, 500
- SDL, 190
- search mode, 44
- secure connection, 177
- selective mutation, 683
- selector expression, 129, 132
- self transition, 253
- semantic change, 607
- semantic differencing, 470
- semantic mutant, 90
- semantic relations, 119
- semicolon (;), 95
- sentence generator, 172
- separation of concerns, 593
- sequence diagram, 56, 181
- set
 - as type in Z, 483
 - binary operation, 515
 - comprehension, 482
 - extraction of elements, 482
 - in Z, 481
 - naming in Z, 481
 - of triples in Z, 482
- set cover
 - naive algorithm, 478
- set cover optimization problem, 460
- set cover problem, 460, 471
 - greedy algorithm, 461
 - naive algorithm, 461
 - tricky, 478

-
- SET interface, 697
 - set product, 115, 153
 - set type, 697
 - SETL, 697
 - short circuit evaluation, 560
 - shutdown system, 607
 - side effect, 254
 - side effects, 192
 - signature, 232
 - instead of UIO sequence, 221
 - SIMPL, 510
 - simple
 - compound, 147
 - simple change, 607
 - simple condition
 - infeasible, 537
 - simple programming mistakes, 624
 - simple relational expression, 151
 - simulated environment, 114
 - simulator, 432
 - Simulink, 690
 - slicing
 - dynamic, 469
 - static, 469
 - smart automobile, 258
 - smart card, 52, 511
 - SMV, 690
 - social security, 108
 - SOFL, 512
 - Software Engineering, 468
 - Software Engineering Terminology, 89
 - software lifecycle, 76
 - software quality
 - early estimation, 90
 - remote monitoring, 591
 - software reliability, 30
 - software systems, 243
 - Software Test Documentation, 89
 - software testing, 88
 - foundation, 75
 - sort, 43, 44, 93, 100, 530
 - factors, 386
 - test generation using factors, 390
 - source code analysis, 53
 - source document, 54
 - SourceForge, 473
 - Space program, 685
 - spanning set, 591
 - specification, 55
 - algebraic, 511
 - for smart card, 511
 - hybrid, 511
 - rewrite, 130
 - UML, 511
 - specification based test generation, 171
 - specification based tests, 168
 - specification mutant, 501
 - specification mutation, 500, 689
 - speed measurement, 258
 - SPIN model checker, 90
 - spiral model, 81
 - spiral testing, 81, 82, 90
 - SQLUNit, 473
 - square root program, 600
 - STAD, 594
 - stand-alone item, 136
 - Start node, 568
 - state
 - accepting, 185
 - ancestor, 265
 - AND, 252, 263, 298
 - basic, 264, 265
 - child, 291, 296
 - composite, 252, 254, 271
 - compound, 270
 - conditional pseudo, 255
 - current, 183
 - deferred transition, 252
 - delay connected, 362
 - destination, 253
 - diagram, 183, 185, 192
 - tabular, 187
 - diagram of DIGDEC, 184
 - distinguishable, 188
 - distinguishing in a statechart, 269
 - do activity, 252
 - entry action, 252
 - equivalence, 246
 - equivalent, 188
 - estimating maximum number of, 205
 - example, 40
 - exit action, 252
 - expected, 353
 - final, 41, 185, 253
 - history, 254, 270
 - identification set, 204, 212
 - example, 204
 - idle, 273
 - immediate ancestor, 265
 - immediate common ancestor, 266
 - infection, 621, 629, 682
 - infection condition, 621
 - infection example, 631
 - initial, 41, 185, 253, 440
 - input identified, 362
 - internal transition, 252

-
- invalid, 391
 - k-distinguishable, 246
 - k-equivalent, 198
 - name, 252
 - non-basic, 265, 291
 - observability, 275, 305
 - of a lamp, 183
 - of environment, 126
 - of object, 126
 - OFF, 183
 - ON-BRIGHT, 186
 - ON-DIM, 183, 186
 - OR, 252
 - propagation, 621
 - propagation condition, 622
 - propagation example, 631
 - pseudo, 254
 - removal, 196
 - root, 253
 - scope, 265
 - self loop, 220
 - sequence, 41
 - source, 253, 266
 - start, 253
 - static reaction, 252
 - status
 - in(S), 258
 - substate
 - cross product, 263
 - super, 253
 - target, 253, 266
 - transition function, 185, 198
 - V-equivalence, 246
 - state cover, 238
 - state cover set, 211, 364, 366
 - state identification set, 364, 367
 - state space
 - partitioning, 312
 - perturbation, 629
 - state transition
 - table, 199
 - state vector, 40
 - statechart, 44, 55, 75, 183, 251–266, 686
 - behavior, 260–264
 - characterization set, 276
 - child, 306
 - complete, 275
 - completeness assumption, 320
 - configuration, 263, 266
 - example, 263
 - configurations, 311
 - control portion, 267
 - controller, 258
 - counter, 312
 - data portion, 267
 - default entrance, 299
 - differences with FSM, 266
 - do-activity, 311
 - execution, 265
 - extension, 309
 - flattened, 299
 - flattening, 264
 - flexible notation, 309
 - for microwave oven, 268
 - for tape player controller, 269
 - hierarchical composition, 267
 - hierarchical nature, 306
 - I/O transformer, 268
 - in transportation, 310
 - internal events, 268
 - internal memory, 267
 - main, 289
 - medical device modeling, 310
 - model of UML classes, 310
 - mutating tests generated, 310
 - orthogonal component, 298
 - output distinguishability, 318
 - partial, 320
 - programmed execution, 310
 - reducing clutter, 257
 - reference book, 309
 - semantically equivalent, 283
 - semantics, 310
 - signal, 258
 - specification of hypermedia applications, 310
 - substate, 289
 - t-completeness, 306
 - tracing execution, 308
 - transition labels richer than in FSM, 267
 - UML, 181
 - variation, 319
 - visual formalism, 309
 - statechart correctness
 - verification using simulation, 266
 - statechart coverage, 308
 - Statechart testing, 74
 - StateCraft, 309
 - Statemate, 252, 259, 268, 309, 310, 319
 - statement, 69
 - declarative, 530
 - executable, 530
 - unreachable, 531
 - statement coverage, 530
 - domain, 531
 - example, 531
 - statement deletion operator, 692
-

-
- states
 - distinguishing using delay, 367
 - static analysis, 46
 - example, 58
 - static analysis tool, 58
 - static reaching definition, 453
 - static reactions, 268
 - static slice, 448
 - static testing, 56
 - stopping rule, 590
 - stress testing, 51, 78
 - string, 72
 - alphanumeric, 93
 - language, 72
 - non-empty, 188
 - test input, 72
 - strong conformance
 - example, 234
 - strong mutation, 610, 692
 - Structural testing, 74
 - structural testing, 586, 588
 - part of functional testing, 588
 - structured program, 94
 - structures, 109
 - stuck-at fault model, 50
 - student, 109
 - student record, 79
 - sub-domain, 34, 384
 - sub-path, 65
 - substate, 253
 - crosss product, 263
 - subsumes relation, 585, 593, 594
 - example, 585
 - for c- and p-uses, 602
 - successor region, 338
 - example, 338
 - sufficiency condition
 - weak, 682
 - super block, 596
 - super state, 253
 - switch
 - lamp, 183
 - rotary, 183
 - switch cover, 238
 - switch cover criterion, 239
 - symbol
 - input, 186
 - output, 186
 - symmetric, 246
 - syntactic category, 634, 635
 - syntactic differencing, 472
 - syntax
 - carrier of semantics, 607
 - Syntax testing, 74
 - syntax testing, 171
 - syntax tree, 440, 441, 444, 470
 - abstract, 148
 - equivalence, 441
 - syntehsizer, 190
 - system
 - integration, 76
 - testing, 76
 - system test, 95, 96, 181
 - system tests, 588
 - t**, 151
 - t-factor fault, 391
 - t-tuple, 409
 - t-way fault vector, 392
 - t-way interaction fault, 391
 - TA: Timed Automata, 322
 - table lamp, 183, 186
 - TACTIC, 594
 - tape, 273
 - tape player
 - compartment, 312
 - error detection, 313
 - expected behavior, 273
 - tests, 313
 - timing, 313
 - tape player controller
 - statechart, 269
 - test cases, 273
 - transitions, 271
 - tape recorder, 262
 - TAPE_END, 251
 - tasks
 - conjoining of, 168
 - TBA: Timed Büchi Automaton, 336
 - TCG: Test Case Generator, 423
 - TDD: Test Driven Development, 82
 - Telcordia Technologies, 592
 - telephone network, 39
 - temporal logic, 59, 510
 - tennis coach, 26
 - terminology overlap, 79
 - Terz, 595
 - test
 - script, 80
 - adequacy, 518
 - adequacy criterion, 518
 - adequacy measurement, 519
 - completeness, 89, 590
 - coverage, 519
 - data, 594
 - data base, 692

- driver, 172
- enhancement, 588
- error revealing, 166
- fault revealing, 166
- frame, 126, 132
- generation, 54
- generic specification, 509
- harness, 172
- ideal, 591
- minimal, 598
- modification, 56
- ordering, 504
- parameters, 384
- point intermediate, 133
- point internal, 133
- redundant, 620
- representative, 102
- script, 126, 132
- sequence, 504
- specification, 132
- successful, 591
- suite, 56
- team, 98
- time, 558
- timing of application, 274
- trace back, 562
- useless, 314
- value, 384
- vector, 440
- verification, 434
- test abortion, 466
- test adequacy, 590
 - alternate procedures using mutation, 620
 - based on c-use and p-use, 574
 - based on IUT, 308
 - based on statechart, 308
 - clock region coverage, 348, 381
 - concurrent assessment, 620
 - condition coverage, 598
 - k-dr coverage, 579
 - path coverage, 521
 - region coverage, 348
 - statechart coverage, 308
 - with respect to MC coverage, 552
- test adequacy assessment
 - multi-tester version using mutation, 620
 - mutation based procedure, 613–620
- test adequacy criteria
 - program-based, 590
 - specification-based, 590
- test assessment
 - foundations, 590
 - using mutation, 612
- test automation, 28
- test automation tools, 28
- test box, 119
- test case, 36, 75
 - as a pair, 507
 - discard infeasible, 277
 - discarded, 279
 - documentation, 561
 - feasible, 276
 - from factor combination, 385
 - fuss about, 530
 - new, 527
 - redundant, 561, 692
 - satisfies constraint, 151
- test case generator, 680
- test completeness, 590
- test configuration, 383, 384
 - from combinatorial object, 389
- test construction-enhancement cycle, 522
- test coverage, 48
- test data, 37
- test design
 - as formal synthesis, 690
- test effort, 84
 - wasted, 85
- test enhancement, 521
 - example, 521
 - foundations, 590
 - incremental, 620
 - procedure, 521–522
 - process, 521
 - purpose, 527
- test environment, 85
- test execution, 28
 - error prone, 28
 - manual, 434
- test execution history, 471
- test frame, 130
 - automatic generator, 131
 - deriving a test case, 132
 - generation, 132
 - key, 131
 - not a test case, 132
 - redundant, 130, 131
 - sample, 131
- test frame generator, 126
- test generation
 - adaptive, 685
 - algorithm, 56
 - automata theoretic, 237
 - black box, 86
 - category partition method, 126
 - classification tree, 172

- code based, 56
- combinatorial explosion for flattened statechart, 299
- control flow based, 237
- deterministic implementation, 328
- exploiting hierarchy in statechart, 306
- for GUI using factors, 390
- for reactive systems using black box approach, 376
- for real-time systems using formal methods, 375
- for sort using factors, 390
- for timing constraints, 376
- from a hierarchical statechart, example, 306
- from asynchronous sequential machines, 245
- from communicating FSMs, 244
- from decision table, 146
- from EFSM, 244
- from finite state machine, 182
- from formal specifications, 480
- from non-deterministic FSM, 244
- from NTFSM, 364–365
- from NTFSM, example, 365
- from SDL specifications, 245
- from SOFL specifications, 512
- from statechart, 182
- from statechart with AND states, 300–301
- from statechart with fork and joint connectors, 303
- from statechart with history connectors, 297
- from statechart with inter-level transitions, 295
- from UML statecharts, 310
- from v-v pair, 287
- from VFSM, 245
- from Z specifications, 172
- grammar based, 172
- history state, 316
- in presence of guards, 283
- informal, 55
- MC/DC adequate, 546–549
- model-based, 55
- procedure, 182
- specification based, 83, 171
- statistical for time embedded system, 376
- theory, 89
- UIO method, 232–233
 - example, 233
- using heuristics, 144
- using Z, advantages, 509
- test harness, 39, 323
 - clock reset, 352
 - example, 39
 - example in timing tests, 353
 - for tests generated from statecharts, 272
 - for timing tests, 351
 - importance, 280
 - interaction with IUT, 353
- test minimization, 437, 463, 466, 471
 - as set cover problem, 460
 - bi-criteria, 471
 - coverage based, 477
 - delayed greedy algorithm, 471
 - example, 277, 459
 - for non-deterministic FSM, 245
 - for tests generated from TIOA, 376
 - generation of useless tests, 280
 - loss of vital information, 278
 - procedure, 461
 - using ATAC, 472
- test optimization, 234, 248
 - example, 234
- test pattern generation, 91
- test plan, 36, 89
- test pool, 39
- test prioritization, 431, 437, 463, 467, 471
 - coverage based, 471
 - economic basis, 472
 - history based, 471
 - industrial strength, 472
 - JUnit tests, 471
 - LRU method, 471
 - using ATAC, 472
 - using Echelon, 472
- test scenario, 88
- test script, 80
- test selection
 - boiler control, 118
 - equivalence class, 116–119
 - example, 117
 - slicing based, 469
- test selection tool, 437
- test sequence
 - upper bound on length, 244
- test sequences
 - relative lengths, 242
- test sequencing, 432
 - example, 432
- test sequencing constraint, 478
- test set, 36, 100
 - BOR adequate, 178
 - BRE adequate, 178
 - construction using W-method, 207
 - decision adequate, 597, 693
 - enhancement, 587
 - goodness of, 586

- hopelessly inadequate, 619
- inadequacy, 521
- inadequate with respect to mutation, 612
- MC adequate, 598
- MC/DC adequate, 559, 598, 599, 601
- minimal, 600
- minimization, 695
- multiple-condition adequate, 597
- satisfying BOR, 152
- satisfying BRE, 152
- satisfying BRO, 152
- union, 169
- test set up, 431
- test specification, 89
- test suite
 - astronomically large, 376
 - precision, 471
 - recall, 471
- test suite reduction, 471
- test template, 493, 495, 496
 - derivation, 515
 - division, 497
 - division example, 497
 - division using domain propagation, 498
 - division using domain propagation, example, 498
 - example, 495
 - fault detection effectiveness, 516
 - for robustness, 508
 - hierarchy, 498
 - instantiation, 502
 - leaf, 494
 - refinement, 497
 - terminal, 502
- test template hierarchy, 493
- test tools, 572
- testability
 - against FSM designs, 244
 - dynamic metric, 49
 - guided test pattern generation, 91
 - hardware designs, 91
 - improvement using assertions, 91
 - improvement using oracles, 91
 - Object-oriented systems, 91
 - of timing constraints, 376
 - requirement, 50
 - static metric, 49
- testable code, 29
- testable code segment, 470
- testable entity, 459
- testable unit, 127
- TestAdvantage, 472
- TestComplete, 472
- tested, 519
- tester, 28, 29, 127
 - adament, 562
 - experience, 101
 - frustration, 527
- tester's interpretation, 479
- testing, 25, 34, 36
 - in waterfall development process, 82
 - a joke, 93
 - acceptability, 76
 - ad-hoc, 75
 - agile, 82
 - algebraic specification, 688
 - and verification, 52
 - balanced design not needed, 411
 - beta, 76
 - black-box, 74, 76
 - boundary-interior, 89
 - complete, 518, 597
 - concurrent Java program, 683
 - confidence, 52
 - conformance, 189
 - dynamic, 73
 - exhaustive, 100, 101
 - fault-injection, 604
 - function, 109
 - functional, 78, 586
 - glass-box, 586
 - goal, 25
 - goal directed, 77
 - good enough, 518
 - grey box, 322, 323, 354
 - GUI, 82
 - human intensive, 28
 - in V model, 82
 - integration, 76
 - interaction, 384
 - interface, 75
 - load, 78
 - loops, 590
 - model-based, 59, 75
 - mutation, 75, 604, 609
 - mutation based, 636
 - Nortel's email system, 422
 - object, 109
 - of actions, 25
 - of product, 25
 - of thoughts, 25
 - OO, 79
 - pairwise, 75, 76, 384
 - penetration, 77
 - performance, 78
 - predicate, 146

-
- random, 75
 - random for robustness, 508
 - range, 172
 - regression, 76, 77
 - robustness, 78, 507
 - security, 77
 - sort, 26
 - spiral, 82
 - static, 56, 73
 - stress, 78
 - structural, 586
 - system, 76
 - thorough, 128, 518
 - types, 73
 - unit, 76
 - user-centric, 30
 - using W-method, 208
 - using Wp-method, phase 1, 212
 - using Wp-method, phase 2, 212
 - vulnerability, 77
 - web services, 82
 - white-box, 75, 76, 586
 - testing problem, 193
 - testing subtree, 293
 - testing techniques
 - classification, 73
 - complexity, 90
 - testing tree, 205, 207, 240, 271
 - example, 206
 - example of merging, 292
 - merged, 293
 - properties, 315
 - tests
 - minimal, 122
 - modification traversing, 437
 - obsolete, 435
 - program based, 168
 - redundant, 435
 - regression, 435
 - specification based, 168
 - thoroughly designed, 52
 - timed, 323
 - TestTube, 28, 466, 472
 - text formatter, 89
 - text processing, 43
 - textual differencing, 470, 472
 - time assignment
 - equivalence, 334
 - time complexity, 478
 - timed automata, 324, 375
 - abstract snapshot, 339
 - example, 325, 331
 - timed automaton, 335
 - example, 330
 - snapshot, 334
 - timed Büchi automata, 375
 - timed input automata, 98
 - timed input output automata, 376
 - timed language, 329, 378
 - example, 329
 - timed trace, 329, 378
 - example, 329
 - timed Wp-method, 322, 354, 363
 - variation of, 376
 - timer, 332
 - example, 332
 - timing, 59
 - requirements, 370
 - timing constraint fault
 - example, 346
 - timing constraints, 98, 101, 329
 - timing error, 322
 - timing properties, 59
 - timing requirements
 - partitioning, 375
 - TIOA: Timed Input Output Automata, 322
 - to mutate
 - act of mutation, 605
 - tomato, 104
 - tools
 - data flow testing, 594
 - tour, 233
 - toy, 183
 - trace, *see* execution trace
 - trace back to requirements, 561
 - train collision avoidance, 321
 - transaction flow testing, 80
 - transfer error, 194, 209, 247
 - not revealed, 239
 - transfer fault, 371
 - detection example, in a TIOA, 371
 - transfer sequence, 231
 - transition, 253
 - action, 260
 - completion, 254, 261
 - completion event, 261
 - compound, 264
 - delay, 340
 - enable, 265
 - fire, 265
 - firing, 261
 - I/O, 340
 - incoming, 253
 - inter-level, 295, 316
 - label, 254
 - action, 254

- guard, 254
- trigger, 254
- label in a statechart, 270
- label with guard, 274
- no associated action, 304
- out of a joint connector, 303
- out of fork connector, 302
- outgoing, 253
- scope, 265
- semantics, 254
- trigger, 261
- with lone guard, 275
- transition cover, 238
- transition cover set, 205, 247, 270, 364
 - for AND state, 317
 - in presence of guards, 284
 - non-basic states, 291
- transition diagram, 247
- transition error
 - detection of, 299
- transition function, 187
- transition tour method, 241, 249
- transition tours, 244
- transitions
 - instantaneous, 382
- transitive, 246
- transputer, 686
- tree transformations, 470
- triangle classification, 516
- triangle problem, 515
- trigger
 - label
 - empty, 254
 - optional components, 254
- true, 147
- truth value, 148
- TSL, 132, 172
 - different terminology, 132
 - specification, 131
 - syntax, 130
- TSL: Test Specification Language, 130
- TV, 262
- UIO method, 241, 270, 290
 - assumptions, 217
 - failure to detect faults, 236
 - fault detection example, 235
- UIO sequence, 217
 - algorithm for generation of, 221–223
 - checking, 219
 - does not exist, 221
 - example, 219, 224–226, 228–231
 - length, 217
 - optimal length, 244
- UIO: Unique Input/Output Sequence, 217
- UIOv method, 241, 244
- UML, 56, 98, 99, 252
 - sequence diagram, 181
 - statechart, 181
- UML specifications, 511
- unbalanced design, 412
- unconstrained pointers, 469
- undefined, 40
- unidimensional partitioning, 110, 175
- unification technique
 - formal analysis, 686
- unit testing, 76
- United States, 431
- universal input set, 508
- universe defining entities, 633
- University of London, 665
- UNIX, 470
- Unix, 40, 592
- UPPAAL, 377, 509
- usability testing, 30
- USB: Universal Serial Bus, 28
- use
 - global, 566, 567
 - input variable, 572
 - local, 566
 - potential, 594
- use case, 99
- use set, 567
- user centric testing, 30
- user mode, 592
- uses, 564
- V model, 82
- V-equivalence, 188
- V-model, 80, 81
- v-v pair, 315
- v-v pair: variable-value pair, 286
- valid criterion, 591
- validity check, 123
- variable
 - global set, 642
 - live definition, 568
 - local set, 642
- variable negation, 147
- variable redefinition, 58
- VCR, 251, 262
- VDM, 509
- vector of factor levels, 392
- vector processor, 686
- verification, 52
- Verilog, 595

-
- Vermont Creative Software, 472
 - VFSM: Variable Finite State Machine, 245
 - video cassette, 44
 - Virtual PC, 387
 - Visual Basic, 90
 - visualState, 310
 - VLSI, 51
 - Voas, Jeffery, 695
 - vulnerability testing, 77, 79

 - W procedure, 202
 - W set
 - example, 198
 - W set: state characterization set, 198
 - W-method, 183, 204, 242, 243
 - assumptions, 205
 - error detection, 209
 - fault detection effectiveness, 237
 - improvements over, 243
 - upper bound on number of tests, 247
 - walkthroughs, 57
 - WAPT, 28
 - Wayne State University, 403
 - weak conformance, 245
 - weak mutation, 610, 681, 692
 - applied to basic blocks, 682
 - combined with black-box testing, 682
 - effectiveness, 682
 - Hamlet's version, 681
 - mutation, comparison, 681
 - score, 686
 - strength of, 682
 - web based company, 136
 - web search, 470
 - web service, 80, 82, 431, 434
 - testing, 80, 82
 - web services, 588
 - WebSphere Studio, 80
 - weekly salary, 100
 - while statement, 69
 - white-box
 - test adequacy criteria, 519
 - white-box category, 79
 - white-box testing, 75, 586
 - white-box tests, 170
 - whole execution trace, 469
 - Windows forms testing, 472
 - Windows operating system, 52
 - Windows XP, 383
 - WinRunner, 28, 472
 - withdraw, 433
 - Wp method: partial W method, 211
 - Wp-method, 204, 242, 270, 290
 - example, 213–215
 - fault detection effectiveness, 237
 - procedure, 212
 - timed, 322
 - writing pen, 174

 - X-testing, 73
 - X.25, 190
 - XML message, 683
 - XOR-node, 148
 - XP, 91
 - XP: eXtreme Programming, 82
 - χ SUDS, 572, 592, 594
 - XTest, 28

 - Year 2000 problem, 470
 - yogurt, 104

 - Z, 75, 697
 - Δ , indicates a variable might change, 488
 - front* operation, 485
 - head* operation, 485
 - last* operation, 485
 - tail* operation, 485
 - arithmetic expressions, 483
 - arithmetic operators, 483
 - axiomatic description, 483
 - combining schemas, example, 487
 - commentary, 481
 - non-empty sequence, 485
 - notation, 481, 509
 - object oriented extension, 510
 - predicate construction, 484
 - predicates, 484
 - robustness testing, 507
 - schema, 481, 486, 495, 511
 - schema as macro, 488
 - schema calculus, 487
 - schema signature, 486
 - schema with type compatible signature, 487
 - sequence, 484
 - sequence as function, 485
 - set comprehension, 491
 - specification, 481, 492, 502, 505, 509–511
 - test generation, 494
 - test template, 493, 495
 - to specify statechart behavior, 511
 - why use?, 509
 - Z notation, 82, 95
 - Z specification, 686
 - Ada, 686
 - Z: pronounced as Zed, 480
 - zip code, 385, 431
 - ZUS: Z User Studio, 511