
Index

- abstract interpretations, 14
- access expressions, 137
- access graph, 141
 - empty, 141
 - operations, 143–146
 - cleanUp*, 143
 - CN*, 143
 - extension, 145
 - factorization, 145
 - field*, 143
 - lastNode*, 143
 - makeGraph*, 143
 - path removal, 145
 - root*, 143
 - safety of, 146
 - union, 144
 - summarization, 142
- access path, 3, 137
 - aliasing of, 3
 - base of, 137
 - directly generated, 139
 - frontier of, 137
 - killed, 139
 - liveness of, 3
 - simple, 137
 - summarization, 3, 12
 - target of, 137
 - transfer of, 139
- algorithm
 - DFST construction, 60
 - elimination, 184
 - iterative
 - round-robin, 4, 19, 79, 90, 163–164
 - work list, 165–172, 312, 319, 320, 322
 - SSA construction
 - ϕ placement, 196
 - dominance frontier, 194
 - renaming, 199
 - SSA destruction, 214
 - register allocation, 224
- alias analysis, 129–135
 - data flow equations, 134
 - of parameters, 268
 - data flow equations, 269–271
- alias relations, 129
- aliasing
 - data flow equations, 134
 - direct aliases, 131
 - indirect aliases, 131
 - link aliases, 130
 - node aliases, 130
 - of access paths, 3, 9
 - of parameters, 267
 - of pointers, 2, 129–135
- all paths analysis, 33
- anti-symmetric, 64
- anticipable expressions analysis, 37–39
 - data flow equations, 38
- any path analysis, 26
- applications of data flow analysis, 16
- assignment, 176, 177, 181
 - maximum fixed point (MFP), 77
 - maximum safe, 75
 - meet over paths (MOP), 75
- associativity, 66
- available expressions analysis, 33–35
 - common subexpressions elimination, 34
 - data flow equations, 33
- backward data flow problem, 26
- backward edges, 61

- backward flow, 24, 160
- basic block, 23
- bidirectional data flow equations, 39, 42, 184
- bidirectional data flow frameworks, 73, 159, 175, 184
- bit vector, 23
- bit vector frameworks, 23–57, 73, 86, 88, 89, 162, 171
 - anticipable expressions analysis, 37–39
 - available expressions analysis, 33–35
 - combined may-must availability analysis, 53–56
 - dead variables analysis, 29
 - lazy code motion, 49–53
 - live variables analysis, 26–29
 - partial redundancy elimination, 39–49
 - partially available expressions analysis, 36–37
 - reaching definitions analysis, 29–33
 - specification in GCC, 336–340
- bit vector operations, 23, 56
- boundary information (*BI*), 26
- bounded lattice, 70
- call multigraph, 18, 234, 235
- call node, 235
- call segment, 295
- call site, 235
- call string, 295, 302
 - approximate, 328
 - data flow equations, 298, 303
 - restricted, 296
 - data flow equations, 298
 - unrestricted, 302
 - construction, 302, 303
 - data flow equations, 303
 - equivalence, 312, 317, 318
 - for bit vector frameworks, 328
 - regeneration of, 319
 - representation of, 319
 - termination length, 328
 - termination, issues in, 305–310
 - value-based termination, 317–324, 328
- calling context, 295
 - restricted, 296–301
 - unrestricted, *see* call string
- canonical SSA, 207
- chain, 67
- chordal graph, 219
- chromatic number, 216
- code movement, 33
- coloring, 222
- combined may-must availability analysis, 53
 - data flow equations, 55
- common subexpression elimination, 34
- commutativity, 66
- complete lattice, 67
- complexity of data flow analysis
 - bit vector frameworks, 99, 171–175
 - depth of CFG, 61, 94, 97, 99
 - fast frameworks, 175–179
 - information flow paths, 171–184
 - non-separable frameworks, 179–183
 - rapid frameworks, 85–99
 - width, 175
- component function, 102, 153
- composite entity functions (*cef*), 155
- computation of MFP assignment, 79
- conditional constant propagation, 116
 - data flow equations, 117
- conservative approximations, 11, 65, 142, 147, 151, 238, 246, 250, 251, 254
- constant propagation, 108–119, 157
 - conditional, 116
 - copy, 118
 - data flow equations, 112
 - linear, 119
- constraint resolution systems, 13
- context independence, 236
- context sensitive analysis, 296
- context sensitivity, 12, 15, 17, 157, 243
- control flow graph, 1

- control flow graph (CFG), 18, 23, 24, 235
- convergence, 12
- converging paths, 189
- copy, 32
- copy propagation, 32
- critical edges, 47, 50
- critical nodes, 48
- cross edges, 61
- CSSA, *see* canonical SSA
- cyclic call sequence, 308
- cyclic return sequence, 308

- data flow equations, 24
 - alias analysis, 134
 - alias analysis, of parameters, 269–271
 - anticipable expressions analysis, 38
 - available expressions analysis, 33
 - bidirectional, 39, 42, 160, 184
 - call string, restricted, 298
 - call string, unrestricted, 303
 - combined may-must availability analysis, 55
 - conditional constant propagation, 117
 - constant propagation, 112
 - explicit liveness, *see* explicit liveness, data flow equation
 - faint variables analysis, 103
 - generic, 160
 - interprocedural data flow analysis
 - constructing summary flow functions, 278–279
 - using summary flow functions, 283
 - lazy code motion, 49–52
 - liveness, 26
 - partial redundancy elimination, 40, 42
 - points-to analysis, 121
 - possibly uninitialized variables, 107
 - reaching definitions, 30
 - side effects of procedure calls, 261–265
 - unidirectional, 160

- data flow framework, 72
 - instance of, 73
- data flow frameworks
 - bit vector, 23–57
 - anticipable expressions analysis, 37–39
 - available expressions analysis, 33–35
 - bidirectional, 39, 42
 - combined may-must availability analysis, 53–56
 - dead variables analysis, 29
 - lazy code motion, 49–53
 - live variables analysis, 26–29
 - partial redundancy elimination, 39–49
 - partially available expressions analysis, 36–37
 - reaching definitions analysis, 29–33
 - side effects analysis of procedure calls, 259
- category by flow functions
 - bit vector, 73, 86, 162, 171
 - distributive, 73
 - fast, 89, 175
 - k-bounded, 86
 - monotone, 73
 - non-separable, 159, 179
 - rapid, 86–90
 - separable, 159
- direction of flow
 - bidirectional, 73, 159, 162, 175, 184
 - unidirectional, 73, 159, 162
- non-separable, 101–157
 - alias analysis, 129–135
 - alias analysis of parameters, 268
 - constant propagation, 108–119
 - faint variables analysis, 103–106
 - heap liveness analysis, 135–152
 - points-to analysis, 119–129
 - possibly uninitialized variables analysis, 106–108
- properties of, 86

- data flow information
 - exhaustiveness of, 11, 66
 - generation, 24, 56, 102
 - inherited, 237, 238, 254
 - invalidation, 24
 - killing, 56, 102
 - precision of, 12, 112, 115–119, 128, 131, 147, 152
 - representation of, 12, 18, 23, 25, 54, 64–70, 112, 120, 130, 141
 - safety of, 11, 65, 66, 75, 87, 117
 - synthesized, 237, 238, 254
- dead code, 33
- dead code elimination, 28
- dead variables analysis, 29
- def-use chains, 30, 185
- depth, *see* loop connectedness
- depth first spanning tree, 60
- descending chain condition, 67
- DFST, *see* depth first spanning tree
- distributive data flow frameworks, 73
- dominance, 62, 191
- dominance and reducibility, 62
- dominance frontier, 191–193
 - algorithm
 - complexity, 194
- downwards exposed, 25
- dynamic analysis, 14

- equivalence of iterated join and IDF, 201
- exhaustive analysis, 15
- explicit liveness
 - convergence of, 149
 - data flow equation, 147
 - efficiency of, 150
 - specification, 140
- explicitly live access paths, 138
- expression
 - anticipable, 38
 - available, 33
 - partially available, 36
 - partially redundant, 36
 - redundant, 34
- extensive point of function, 78

- faint variables analysis, 103–106, 157
 - data flow equations, 103
- fast data flow framework, 86, 89, 175
- fixed point, 77
- fixed point assignment, 77
- fixed point theorem, 78
- flow functions, 64, 71
 - backward, 73
 - distributive, 72, 73, 83, 90
 - entity functions
 - composite (*cef*), 155
 - primitive (*pef*), 153, 172, 176, 177, 180, 181, 184
 - fast, 86, 89
 - forward, 73
 - k-bounded, 86
 - monotonic, 71–73
 - non-separable, 101, 102, 154, 156
 - separable, 101
- flow insensitive side effects, 248, 263
- flow sensitive side effects, 251, 261
- flow sensitivity, 12, 15, 17, 157, 241, 261
- forbidden subgraph, 62
- forward edges, 61
- forward flow, 24, 160

- garbage collection, 2
- generation of liveness, 6
- generic data flow analyzer, 360, 368
- generic data flow equations, 160
- generic flow functions, 162
- gimple representation, 334
- gimple version of CFG, 342–346
- glb, *see* greatest lower bound, 68, 69
- global data flow analysis, 17, 23, 360–362
- global data flow information, 24
- GNU Compiler Collection (GCC), 333, 365
- graph, 59
 - access graph, 141
 - call multigraph, 18, 235
 - chordal, 219
 - coloring of chordal, 222

- control flow graph (CFG), 18, 23, 24, 235
- depth first spanning tree, 60
- depth of, 61
- memory graph, 136
- parameter binding graph, 273
- points-to graph, 129
- program summary graph, 291
- reducible, 62
- supergraph, 18, 235, 293
- graph reachability, 291
- greatest element, 64
- greatest lower bound, 66
- Hasse diagram, 65
- hoisting
 - desirability, 41
 - safety, 40, 41
- idempotence, 66
- immediate dominator, 192
- incremental analysis, 15
- induction variable detection, 189
- inference systems, 13
- information flow
 - origin of, 171, 176, 180
- information flow paths, 165, 172
 - in bit vector framework, 171
 - in fast frameworks, 175, 176
 - in non-separable framework, 179
 - width of, 174
- interference graph, 216
- interprocedural approximation, 6
- interprocedural constant propagation, 233
- interprocedural data flow analysis, 17
 - call string
 - construction, 302, 303
 - equivalence, 312, 317, 318
 - for bit vector frameworks, 328
 - regeneration of, 319
 - representation of, 319
 - termination length, 328
 - termination, issues in, 305–310
 - value-based termination, 317–324
 - context insensitive, 157
 - context sensitive, 157
 - flow insensitive, 157
 - equality-based, 157
 - subset-based, 157
 - flow sensitive, 157
 - functional approach, 238, 259–290
 - summary flow function
 - construction, 278–282
 - enumeration, 285–290
 - reduction, 275–278
 - side effects of procedure calls, 259–265
 - using restricted contexts, 296
 - using unrestricted context, 301
 - value-based, 239, 293–328
- interprocedurally valid
 - control flow path, 294
 - information flow path, 295
- intraprocedural segment, 295
- invocation graph, 329
- iterated dominance frontier, 195
 - complexity of algorithm, 196
- iterated join, 198
- join, 66, 198
- join semilattice, 70
- k-bounded data flow framework, 86
- killing liveness, 6
- lattice, 67
- lattice of flow functions, 274
- lazy code motion
 - critical edges, splitting of, 50
 - data flow equations, 49–52
 - earliest points of placement, 50, 53
 - latest points of placement, 51, 54
 - region of safe placement, 50
 - transformation, 52
- least element, 64
- least upper bound, 66
- left locations, 120
- lifetime of a name, 12
- link aliases, 130
- live range, 208

- interference, 208
- live variables analysis, 26–29
 - data flow equations, 26
- liveness, 26
 - of access paths, 3
 - of heap data, 135–152
 - of pointers, 2
 - summary, 139
- liveness analysis, 20
 - of heap, 135–152
 - of variables, *see* live variables analysis
- liveness of access paths, 4, 138
- local data flow analysis, 17, 23–25, 358–360
- local data flow information, 24
 - constant, 102
 - dependent, 102
- loop closure, 86
- loop connectedness, 61
- lost-copy problem, 208, 214
- lower bound, 66
- lub, *see* least upper bound, 68, 69

- maximum fixed point assignment, 77
- maximum safe assignment, 75
- may alias analysis, 74
- may availability analysis, *see* partially available expressions analysis
- may points-to, 121
- may-must alias analysis, 69
- may-must availability, 69
- meet, 66
- meet over paths assignment, 75
- meet semilattice, 69
- memory graph, 136
- merge operation, 64
- MFP assignment, *see* maximum fixed point assignment, 81, 82
- minimal SSA, 190
- model checking, 14
- modified Post’s correspondence problem, 84
- monotone data flow frameworks, 73

- MOP assignment, *see* meet over paths assignment, 76, 81, 82
 - undecidability of, 85
- MPCP, *see* modified Post’s correspondence problem
- must availability analysis, *see* available expressions analysis
- must points-to, 121

- node aliases, 130
- non-separable flow functions, 101
- non-separable frameworks, 29, 31, 101–157, 159, 179
 - alias analysis, 129–135
 - constant propagation, 108–119
 - faint variables analysis, 103–106
 - heap liveness analysis, 135–152
 - points-to analysis, 119–129
 - possibly uninitialized variables analysis, 106–108

- parameter alias analysis, 268
- parameter binding graph, 273
- partial order, 64
- partial redundancy elimination, 39–49
 - critical edges, 47
 - critical nodes, 48
 - data flow equations, 40, 42
 - hoisting path, 40
 - transformation, using, 44
 - limitations of, 45
- partial transfer functions, 291
- partially available expressions analysis, 36–37

- path
 - in a graph, 59
 - length, 60
- PEO, *see* perfect elimination order
- PEO ordering, 223
- perfect elimination order, 223
- periodic point, 316
- ϕ -instruction placement, 194
- ϕ -variable renaming, 196
- ϕ -congruence, 209
- ϕ -function, 186

- ϕ -instruction, 186
- ϕ -related, 209
- ϕ -variable renaming algorithm, 197–199
- ϕ -variables, 187
- pointer analysis, 157
 - strong update, 120
 - weak update, 120
- points-to analysis, 119–129
 - data flow equations, 121
 - degree of certainty, 125
 - inverse dependence of, 124
 - left locations, 120
 - may points-to, 121
 - must points-to, 121
 - right locations, 120
 - strong update, 120
 - weak update, 120
- points-to graph, 129
- poset, 64
 - for available expression, 65
 - for live variables, 64
- possibly uninitialized variables analysis, 31, 106–108
 - data flow equations, 107
- postorder, 28
- predecessor, 24
- primitive entity functions (*pef*), 153, 172, 176, 177, 180, 181, 184
- procedure cloning, 254
- procedure inlining, 254
- product lattice, 101
 - components, 68
- program entities, 23
- program representations, 16, 17
- program summary graph, 291
- pruned SSA, 190

- qualified data flow value, 295

- rapid condition, 88
- rapid data flow framework, 86–90
- reaching definitions analysis, 19, 29–33
 - data flow equations, 30
 - def-use chains, 30
 - for copy propagation, 32
 - use-def chains, 30
- reducible, 62
- reducing function compositions, 275
- reducing function confluences, 275
- reductive point of function, 78
- reflexive, 64
- register allocation, 28
- register allocation via graph coloring, 216
- register copies, 226
- register transfer graph, 226
- relation, 64
- return node, 235
- return segment, 295
- reverse postorder, 28, 61
- right locations, 120
- round-robin iterative algorithm, 4, 19, 79, 90, 163–164

- safe assignment, 75
- safety of data flow analysis, 11
- scalar variables, 23
- separable, 102
- separable frameworks, 159
- shape analysis, 157
- side effects, 256
 - flow insensitive, 263
 - flow sensitive, 261
- side effects analysis, 20, 244, 248, 259, 290
- simplicial, 222
- spanning tree, 60
- spilling algorithm, 220
- SSA
 - construction, 189
 - construction algorithm
 - ϕ placement, 196
 - dominance frontier, 194
 - renaming, 199
 - correctness of construction, 198–206
 - destruction, 207
 - destruction algorithm, 214
 - destruction through register allocation, 216
 - coalescing, 223
 - coloring algorithm, 224

- register copies, 226
 - dominance property, 206
 - form program, 186
- stabilization of descending chain, 67
- static analysis, 14
- strict dominance, 191
- strictly stronger than, 64
- strictly weaker than, 64
- strong update, 120
- stronger than, 64
- strongly live variables analysis, 157
- successor, 24
- summarization, 7, 12
 - of liveness, 6
- summary flow function, 236, 238, 290
 - construction, 278–282
 - data flow equations, 278–279
 - data flow equations, using, 283
 - enumeration, 285–290
 - reduction, 275–278
 - side effects of procedure calls, 259–265
- supergraph, 18, 235, 293
- swap problem, 208, 215
- symmetric segment, 295

- termination of call string construction, 305
- traditional register allocator, 216–217
- transfer, 101
- transfer of access path, 6
- transformed SSA, 207
- transitive, 64
- traversal
 - conforming, 174
 - non-conforming, 174
- tree edges, 61
- TSSA, *see* transformed SSA

- undecidability of MOP assignment, 83
- unidirectional data flow frameworks, 73, 159
- unrestricted context, 302
- upper bound, 66
- upwards exposed, 25

- use of the variable, 25
- use-def chains, 30, 185

- valid transformation to SSA form, 190
- variable
 - dead, 29
 - faint, 103
 - live, 26
 - possibly uninitialized, 31, 106
 - strongly live, 157
- variable definition, 206
- variable use, 206
- variable versions, 186
- very busy expressions analysis, 37

- weak update, 120
- weaker than, 64
- whole program analysis, 253, 274, 290
 - context insensitive, 253
 - context sensitive, 254
- work list iterative algorithm, 165–172, 312, 319, 320, 322