

- [Wil84] R. Wilensky. *LISPcraft*. W.W. Norton, 1984.
- [Wil88] Paul T.G. Williams. An approach to the design of a parallel functional language. Master's thesis, Department of Computing, Imperial College, London, 1988.
- [WJW+75] W. Wulf, R.K. Johnson, C.B. Weinstock, S.O. Hobbs, and C.M. Geschke. *The Design of an Optimizing Compiler*. Elsevier, New York, 1975.
- [Wol89] Michael Wolfe. *Optimising Supercompilers for Supercomputers*. Pitman/MIT Press, 1989.
- [WSWW87] Ian Watson, John Sarjeant, Paul Watson, and Viv Woods. Flagship computational models and machine architecture. *ICL Technical Journal*, 5(3):555–574, May 1987.
- [WW87] Paul Watson and Ian Watson. An efficient garbage collection scheme for parallel computer architectures. 1987. In [dBNT87a, pages 432–443].
- [WW88] J.H. Williams and E.L. Wimmers. Sacrificing simplicity for convenience: Where do you draw the line? In *Proceedings of the Fifteenth Annual Symposium on Principles of Programming Languages, San Diego, USA*. ACM SIGPLAN, 1988.
- [You85] M.F. Young. A functional language and modular architecture for scientific computing. In *Functional Programming Languages and Computer Architecture, Nancy, France*. Springer Verlag, September 1985. LNCS 201.

# Index

- $\alpha, \beta$  etc. (type variables), 6, 220
- o (compose), 24, 202
- oo (compose2), 24, 202
- $\perp$  (“bottom”), 28, 29, 54
- $\perp$ , as “not yet”, 33
- $\prec, \succeq$  (orderings), 37
- $\rightarrow$ , to form function type, 9
- $\sqsubseteq$  (“approximates”), 29
- $\sqsubseteq$  (“approximates”), 39
  - for functions, 39
  - for lists, 32
- $\underbrace{\quad}$ , 11, 197
- $\{\perp\}$ , 34
- $\square$  (“make process”), 128
- ++ (append), 9, 200
- ;, 7
- [ ], 7
  
- Abelson, Sussman and Sussman, 40, 42,
  - 44, 45
- Abramsky, 41, 45, 54, 70
- abs, 202
- abstract computer architecture, 2
- abstract interpretation, 29, 54, 70
- abstract machine, 59, 69
- abstraction
  - combinator, 53
- abstraction mechanisms, 123
  - arc, 128
  - node, 130
- abstraction rule, 27
- accumulating parameter, 56
- activation record
  - and space leaks, 73
- adder circuit, 100
- addition, bitwise, 101
- admissible predicates, 33
- Aho, 119
  
- Aho, Sethi and Ullman, 68
- Aiken, 74
- ALFALFA, 65, 74
- ALFL, 156
- algebraic data types, 6
- algorithm design
  - communication in, 159
- ALICE, 65, 73
- all, 202
- Alvey Programme, viii
- annotation, 126
  - declarative, 123
- Appel, 120
- append (++) , 9, 200
  - optimisation, 51
  - propagating into functions, 121, 169
- appendices, 3
- application areas, 163
- application box, 56, 59
  - compressed, 63
  - overwriting the, 57, 60
- APPLY, 59
- apply, 15
- apply* operation, 56
- APPLY2, 63
- ApplyLNO, 141, 201
- “approximates” ( $\sqsubseteq$ ), 29, 39
- arc, 126, 201
  - meaning of, 151
- arrays, *see* vectors, matrices
- Arvind, 23, 45, 85, 119, 155
- Arvind and Brock, 41
- Ashcroft, 119
- assignment, 42
  - and memory re-use, 40, 71
  - lack of, 40
- associativity, 79
  - extension, 81

- asynchronous
  - instruction scheduling, 119
- Augustsson, 42, 56, 69, 70
- Backus, 44, 157, 159
- backwards analysis, 70
- Bailey, 151, 157
- balance problem, 96
- Barendregt, 44, 73, 74
- base case, 33, 166
- behaviour of a process, 7
- Bevan, 72, 74
- bidirectional data flow, 126, 146
- binary tree, 80
- BinaryTree, 80
- Bird and Wadler, 5, 43
  - on *foldl* and *foldr*, 18
- “bisection” ordering, 166
- BitwiseAdder, 101
- Bjorner, 47
- block structure, 10
  - in conventional languages, 69
- blocking, 148
- Bloss, 72
- BNF (Backus-Naur Form), 6
- Bool, 51
- bottleneck, von Neumann, 159
- “bottom” ( $\perp$ ), 28, 29, 54
- bounding volume, 112, 120
- BOX, 60, 64
- box
  - application, 56
  - proto-channel, 149
- boxed interface, 56
- boxed values, 55
- boxing analysis, 55
  - to avoid BOX, 60
- Boyer and Moore, 121
- branch elimination, 63
- breadth first, 107
- breadth-first, 106
- breadth-first tree-stream translation, 186
- Brooks, 42
- BUCKWHEAT, 74
- building blocks, 13
- BuildStreamsOfTrees, 109, 191
- BuildTree, 107
- BULLDOG, 74
- Bundle, 140
- bundling, 140
  - elements of successive iterations, 143
- Burn, 68, 70, 71, 150
- Burstall, 42, 46, 121
- Burton, 156
- bus, 157
- bus, 97
- Caliban, 127
  - and Occam, 155
  - compiler structure, 147
  - examples, 137
  - garbage collection in, 148, 149
  - implementation, 146
  - normal form, 136, 146
  - semantics, 158
  - simplification of, 131
- call-by-need, 55
- call-by-value, 55, 57
  - in parallel graph reduction, 66
  - languages, 219
  - speed of, 42
- CAM abstract machine, 70
- cancellation rule, 27
- Cardelli, 46
- CarryOf, 101
- Categorical Combinatory Logic, 70
- chain, 129, 202
- chain complete, predicates, 33
- chain of compositions, 180
- chain process network, 130
- Chambers, 119
- Chang and Lee, 121
- channels, 148
  - and processes, 148
  - carrying non-streams, 150
  - carrying pointers, 149
  - creation and deletion, 149
  - proto, 149
- Char, 6
- Chen, M.C., 120

- Church, 44
- Church-Rosser property, *see* confluence
- Church-Turing principle, 228
- Clack, 65
- Clarke, 69
- code generation, 56
- code generator, 57, 60
  - optimisations, 62
- CodeBlock, 58
- CodeGenerator, 58
- combinational logic, 97
- combinator abstraction, 53
- combinator-based abstract machines, 69
- combinators
  - S,K,I, 69
  - super, *see* supercombinators
- CombineSolutions, 76
- committed-choice non-determinism, 119, 222
- common subexpression elimination, 50
- common subexpressions, 72
- communication
  - to-computation ratio, 144, 158
  - channel, 7, 31, 33
  - in algorithm design, 159
  - optimisations, 151
  - patterns, 123
- compaction, during garbage collection, 71
- compile-time scheduling, 74
- compiler
  - first, 163
  - parallelising, 163
- compiler-generator, 160
- compilers, 49
- completeness, *see* declarative
- complexity theory
  - for VLSI, 159
- compose (o), 24, 202
- compose2 (oo), 24, 202
- computer graphics, 111
- Computing Surface*, 2
- cond, 51, 52, 72, 203
- confluence, 13
- congestion, 147
- connectivity, 95
- conquer phase, 106, 112
- CONS, 6
- const, 203
- constraint-propagation circuit simulator
  - 42
- construct, 78, 203
- Constructors, 6
  - curried, 8
- constructors
  - and weak head normal form, 68
  - irrefutable, 223
  - lexical convention for, 220
- contention, for environment, 69
- continuation-passing style, 70
- copying garbage collectors, 71
- copying policies, 67, 151
- correctness, 163
- CRAY-XMP, 119
- “crossing out” primes, 182
- CTL, ALICE, 74
- Culler, 119
- Cuny, 151, 157
- Curien, 70
- Curry, H.B., 8
- Currying, 8
- currying, 5, 56, 63
- cycle sum test, 161
- cyclic data structures, 72
  
- DACTL, 73
- Daeche, 120
- DAISY, 103
- Damas, 46, 50
- Darlington, 44, 73, 121, 178
  - on *fold/unfold*, 28
- Darlington, Henderson and Turner, 44
- data dependencies, 40
  - analysed at run-time, 119
  - chain of, 102
  - of process networks, 91
  - unexpected, 156
- data dependency
  - and starvation, 161
  - and arc, 127
  - span of, in cycle, 152

- data flow analysis, 70
- data structures
  - evolving, 40, 42
- data type transformation, 80, 107, 165
- data type transformations, 51
- data types, 6, 220
- data-driven, 73
- dataflow, 72, 119, 155
  - compilation for shared memory, 119
  - computer architecture, 85, 119
  - graph, 85
  - in contrast to process networks, 85
  - Manchester, 118
  - programming languages, 91, 119
- deadlock, 160, 161
- declarative annotation, 123, 126
- declarative completeness, 41, 46
- Decompose, 76
- DEFINELABEL label, 63
- definition rule, 27
- definitions
  - by pattern matching, 8
  - in where clauses, 10
  - of variables, 7
  - parameterised, of functions, 7
  - pattern matching in, 12
  - recursive, *see* recursive definitions
- DeGroot, 47
- Delay, 99
- denotational semantics, 120
- destructive overwriting, 40
- determinism, 1, 41, 45, 155, 163, 222
- Deutsch, 159
- difference lists, 169
- diffusion of work, 74
- digital view of circuit design, 102, 120
- displays, 69
- distribution of processes, 123
- divide phase, 104, 112
- divide-and-conquer, 67, 76, 119
  - divide phase, 104
  - ray tracer, 111
  - using process network, 104, 184
- DivideAndConquer, 77, 104, 184, 201
- divides, 203
- “don’t know” values, 99
- “double bonus”, 125
- drop, 81, 165, 203
- dynamic process networks, 154
- eager evaluation of lists, 68
- earlier, 92, 204
- Eckert, 1, 163
- EDVAC, 1
- Ekanadham, 23, 119
- ELLA, 102, 120
- Ellis, 74
- “envelope” object, 112
- environment, 54
- environment links, 69
- environment-based abstract machines, 69
- “equal rights”, 21
- equality for functions, 27
- Equation, 58
- equations, 6
  - type, 6
- Eratosthenes’ sieve, 96, 182
- Ershov, 160
- “eureka” step, 109
- EVAL, 60
- evaluate* operator, 57
- EvaluateTree, 107, 184, 204
- evaluation transformers, 68
- EvenOnes, 83, 171, 204
- exercises, 166, 179
- explicitly-parallel programs, 163
- Expression, 58
- expression
  - reducible, 11
- extensional properties, 28
- eye, viewer’s, 111
- Facts, 179
- Fairbairn, 56, 69
- FALSE, 51
- fan, 130, 204
- Faustini, 120
- Feather, 121
- feedback, 100
- FeedbackFunction, 100
- FEEDBACKTAG, 110

Feldman, 158  
 fib, 18, 77  
 Fibonacci numbers, 18, 77, 86, 160  
 fibs, 86  
     Kleene chain of, 32  
 Field and Harrison, 43, 44, 54, 69, 70, 121  
 filter  
     low-pass, 141  
 filter, 78, 204  
 FilterMultiples, 96, 182  
 FindImpacts, 93, 124, 178, 205  
 FindRayColour, 113  
 fine-grain, 74  
 finite and total, 37  
 FiniteAndTotal, 36  
 first-order, languages, 119  
 FirstGeneration, 188  
 FirstImpact, 92, 113, 205  
 fixed point, 31  
 FL, 46  
 FLAGSHIP, 65, 67, 73, 151  
 Flo, 157  
 Floating Point Systems, 157  
 flow control, 148  
 fold rule, 27  
 fold/unfold transformation system, 28  
 followed by, 7  
 fork overhead, 65  
 Fourman, 120  
 FP, 157  
 FPM abstract machine, 70  
 fractal, 121  
 frame pointer, 59  
 from, 7, 84, 205  
 fst, 205  
 Fuh, 46, 50  
 FullAdder, 101  
 functional languages, 40  
     problems with, 41  
 functional programming, 1, 5, 163  
     and digital circuit design, 102, 120  
 functions  
     collected definitions, 200  
 functions, non-strict, 64  
 Futamura, 160  
 G-machine, 56, 69  
 Gajski, 119  
 garbage collection, 65, 71  
     compile-time, 72  
     copying, 71  
     hardware support for, 71  
     in Caliban, 148, 149  
     low cost of, 72  
 Gauss-Seidel method, 23, 45  
 GaussSeidel, 23  
 generate, 19, 205  
     in Eratosthenes' sieve, 182  
 GenerateInitialRays, 113  
 generation by generation, 106  
 GetRay, 114  
 GetSubrays, 113  
 GetSurfaceModel, 114  
 Ggen, 61  
 Glaser, vii, 56, 69, 72  
 Glaser, Hankin and Till, 43  
     on combinators, 26  
 Glauert, 73  
 Goguen, 46, 47  
 Goldberg, 65, 67, 69, 74, 75, 118  
 Gordon, 118, 121  
 grain size, 67, 144  
     fine, 74  
     in loosely-coupled multiprocessors, 67  
     run-time analysis, 158  
     with compile-time scheduling, 74  
 graph grammars, 157  
 graph mode code generator, 60  
 graph reduction  
     boxing analysis to speed up, 55  
 graph representation of an expression, 85  
 graph-rewriting, 73  
 graph-type expression, 58  
 graphical presentation, 158  
 GraphType, 58  
 Gregory, 119  
 grid, *see* mesh  
 Gries, 68  
 GRIP, 65, 74  
 guard, 52  
 guards, 8

- overlapping, 222
- Gurd, 85, 118, 119
- Gutttag, 47
- HalfAdder, 100
- handshaking, 120
- Hankin, vii, 54, 70
- Hankin, Burn and Peyton Jones, 29
- Hanna, 120
- hardware description, 96
- hardware support
  - for combinator reduction, 69
  - for garbage collection, 71
- Harrison and Field
  - on combinators, 26
- Hartel, 119
- Haskell, 3, 219
- Hayes, 69
- hd, 13, 205
- heap, 59
  - of receiving process, 148
- Henderson, 43, 96
- Hewitt, 71
- higher-order functions, 5, 40
  - and polymorphic types, 41
  - considered unnecessary, 46
  - encapsulating process network, 130
  - environment management with, 69
  - strictness analysis of, 70
  - unfolding during compilation, 50
- higher-order logic, 120
- Hindley-Milner type system, 46, 50
- histogram, 17
- histogram problem, 42
- history, stream representation of, 120
- history-sensitive, 100
- Hoffman, 41, 70
- HOL, 120
- HOPE, 121
- horizontal parallelism, 75
  - combined with pipeline, 85
  - in divide-and-conquer, 77
- Horowitz, 119
- Hudak, 3, 46, 65, 67, 71, 72, 74, 156, 158
- Huet and Oppen, 44, 70
- Hughes, 45, 69, 70, 73
- I-structures, 45
- Ianucci, 119
- Id, 119
- Id Nouveau, 119
- IdealOr, 98
- ident, 206
- idiom, for iteration, 18
- if...then...else, 51
- image processing, 119
- Impact, 92, 113
- imperative languages, 163
- Imperial College, London, vii
- implementation
  - of Caliban, 146
- implementation techniques, 49
- induction, 28
  - computational, 33
  - partial structural, 34
  - proof by, 33
  - recursion, 37, 193
  - total structural, 36, 166
- inductive step, 33, 166
- infinite lists, 7, 28, 31, 36
  - of primes, 96
  - of samples, 98
  - proving properties of, 35
- infinite process networks, 96
- inheritance, 46
- inlining, 51
- input/output, 46
  - non-determinism with, 45
- insert, 15, 206
  - and o, 180
  - divide-and-conquer version, 79
- insertleft, 16, 206
- insertright, 16, 206
  - facts about, 179
- instantiation rule, 27
- Instruction, 59
- instruction set, 59
- integral, 22
- integrate, 84
- intensional properties, 28

- inter-process communication, 3
- inter-processor communication, 147
- interchanging rows and columns, 145
- interconnection network, 2, 66, 123
  - bus, 157
- interconnection networks
  - reconfigurable, 147
- interface, 131
- interface, boxed, 56
- interprocessor communication, 2
- intersection test, ray, 92, 111, 112, 116, 120, 124, 125, 153
- invocation, 57
- invocation chain, 60
- invocation frame, 59
- iPSC*, 2
- irrefutable, pattern, 223
- iterate, 20, 145, 206
  - alternative definitions, 100, 183
  - cyclic definition, 21
  - in Eratosthenes' sieve, 183
- Jensen and Wirth, 41
- JMP, 60
- Johnson, 102, 104, 120
- Johnsson, 56, 70
- join, 17, 111, 192, 207
- JoinLayers, 187
- Jones, C.B., 47
- Jones, N.D., 160
- Jones, S.B., 41, 45
- Kaes, 46, 50
- Kahn Principle, 88, 120
- Kahn principle
  - verification of, 120
- Kajiya, 120
- Kedem, 154, 156
- Keller, 74, 119, 120, 156
- King's College, London, vii
- Kleene chain, 31, 33, 37
- Klop, 44, 70
- Kogge, 119
- Kranz, 70, 71
- KRC, 219
- Kuck, 74
- Kung, 159
- ladder, 129, 207
- $\lambda$ -calculus, 43, 44
  - and parallel or, 44
  - versus term-rewriting, 44
- $\lambda$ -lifting, 53, 69
- Landin, 70
- LARCH, 47
- latency
  - avoiding by eager evaluation, 68
- latency, memory access, 119
- laws, 27
- layout rule, 10
- laziness, full, 53
- lazy, 5
- Lazy ML, 69, 219
- LCF, 118, 121
- Lean, 73
- Lee, 119
- leftmost-outermost reduction, 51
- Lemma 1, 173
- length, 207
- Lieberman, 71
- lifetime of values, 40
- limit, of Kleene chain, 31, 33, 37
- Lindstrom, 47
- linearisation, 170
- LispKit, 219
- List, 6, 220
- list of lists, 145
- ListOfMTreesToStream, 186, 190
- ListOfTreesToStream, 108
- lists
  - [ ] and “.” in Miranda, 220
  - finite and total, 36
  - infinite, 7
  - of characters, 7
  - recursive definition, 7
  - special notation, 7
- ListToTree, 80
- ListToTree, shuffled version, 171
- ListToTree1, 80, 165, 207
- ListToTree2, 83, 172, 208
- ListToVector, 51, 208



- locus of computation, 161
- logic programming, 47, 222
- long instruction word architectures, 74
- look-ahead processors, 119
- loop, 18, 51, 64
- loosely-coupled multiprocessors, 2, 67, 93, 96, 114, 123
  - and parallel graph reduction, 67
- low-pass filter, 141
- LUCID, 91, 119, 161
- macro expansion, 51
- MakeList, 208
- MakeMatrix, 22, 209
- MakePipeItem, 94, 179, 209
- MakeVector, 21, 209
- Manchester Data Flow Machine, 118, 119
- Manna, Ness and Vuillemin, 34, 44
- map, 8, 209
  - divide-and-conquer version, 82
  - propagating inside ApplyLNO, 145
  - propagating into arithmetic, 88
  - propagating into compositions, 178, 180, 181
- map2, 14, 26, 85, 209
- MapElement, 18
  - and update problem, 42
- mapping and configuration, 147
- mapping, of processes, 2
- MapTree, 84
- matrices, 21, 45
- MatrixAll, 210
- MatrixBounds, 21, 210
- MatrixMap, 210
- MatrixMap2, 146, 210
- May, 155, 156
- McCarthy, 44
- McGraw, 119
- Mead and Conway, 124
- mean, 73
- memory access interference, 66
- merge, 171
- mesh, 142, 144, 211
- mesh refinement, 154
- message passing, 123
- meta-programs, 121
- microcode, 69, 157
  - support for Caliban, 151
- migration, 65
- Milner, 46, 50, 121
- Miranda, 5, 219
  - translation into, 220
- Mishra, 46, 50
- MIX, 160
- mixed computation, 160
- ML, 121
  - Lazy, 69, 219
  - standard, 46
- MNODE, 106
- modules, parameterised, 46
- Mogenson, 160
- Moldovan, 120
- Monsoon, 119
- Moon, 71
- moreover clause, 127
- Morison, 102, 120
- MTREETOKEN, 107
- MTreeToStream, 108, 211
  - derivation of, 186
- multiple applications, 63
- multiprocessor
  - sequential, 124
- multitasking, in Caliban, 150
- MultiTree, 106, 184
- MultiTreeToken, 107, 186
- mutual exclusion
  - of graph updates, 66
- mutual exclusive equations, 12
- mutually-exclusive guards, 8
- Mycroft, 70, 72
- Nat, 36
- natural numbers, 36
- neighbour-coupled multiprocessor, 2, 68, 151, 159
- network-forming operators, 129
- Newton Raphson method
  - removing sub, 174
- Newton-Raphson method, 87
  - distributed, 137

- example, 19
- Nikhil, 85, 119, 155
- NIL, 6
- NOIMPACT, 92
- non-determinism, 41, 42, 45, 163
- non-deterministic merge, 42
- non-local memory reference, 67
- non-strict functions, 64
- normal form, 12, 13, 28, 29, 51, 57
  - Caliban, 136, 146
  - weak head, 68
- normal-order reduction, 51, 52
  - and strictness analysis, 55
- normalisation strategy
  - general, 13, 41
  - inefficiency of general, 51
  - normal-order, leftmost-outermost, 51
- not, 211
- "not yet" instead of never, 33
- Num, 6
- numbers
  - natural, 36
- OBJ, 46
- object database, 112
- object-oriented programming, 42
- object-oriented specification, 47
- Occam, 155
- odd- and even-indexed sublists, 83
- OddOnes, 83, 171, 211
- offside rule, 10
- OnBoundary, 141, 211
- operating systems, 45
- optimisation, 62, 76
  - by transformation, 168
  - of communications, 151
- Or, 99
- or, non-strict, 220
- or, non-strict, parallel, 44
- or, parallel, 52
- ordering, 37
  - "bisection", 166
  - by generations, 187
  - prefix, 32
- Orwell, 219
- Osmon, vii
- otherwise, 8
- OUTPUTTAG, 110
- overhead, 67
  - fork, 65
  - join, 66
  - of non-local memory reference, 67
- overheads, 66
- overheads, of data flow, 119
- overlapping equations, 8
- overlapping patterns, 7, 12
- overloading, 46
- overwriting parameters in place, 64
- overwriting, the application box, 57, 65
  - with a boxed value, 60
  - with a value, 60
- pair, 212
- Papadopoulos, 119
- para-functional programming, 156
- PARALFL, 156, 158
- parallel
  - or, 44, 52
- parallel graph reduction, 65, 73, 76, 118, 155
  - on loosely-coupled multiprocessors, 67
  - verification of, 74
- parallel graph reduction machines, 65
- parallel programming
  - difficulty of, 1, 120, 125, 159, 163
  - distribution part of, 123
- parallel reduction, 55
- parallelising compilers, 163
- parallelism
  - divide-and-conquer, 76
  - horizontal, 75
  - pipeline, 76, 84, 119
  - vertical, 75
- parameter count, 56
- parameterised definitions, 7
- PARLOG, 119, 222
- partial application, 8
- partial data structures, 31, 36
- partial evaluation, 50, 71, 160
  - applied to ray tracing, 160

- self application, MIX, 160
- Partition, 144
- partitioning, 66, 119
  - arrays, 144, 157
  - automatic, 154
  - in Flo, 157
  - local neighbourhood operation, 144
  - pipeline, 140
- partitioning, of processes, 2
- PartitionList, 154
- pattern matching, 12, 220
  - and irrefutable constructors, 223
  - compiling, 70
  - overlapping, 222
  - removal, 52
  - sequential, 222
- pattern-matching, 8
- pencil and paper, 118
- pending list, 66
- Pepper, 121
- performance, 42
  - comparative, 49
- permuting indices, 143
- Peyton Jones, 42, 49, 54–56, 65, 69, 70, 73, 155
- photolithography, 124
- physical universe, limitations of, 158
- Pingali, 119
- Pipeltem, 93, 179
- pipeline
  - variable length, 153
- pipeline, 91, 130, 212
- pipeline parallelism, 84
  - in ray tracer, 111
  - without streams, 150
- PipelineStage, 94, 116, 125, 179
- pipelining, 76, 119
  - successive iterations, 88
- placement, of processes, 2
- plumbing problem, 46
- ply, 15, 26, 212
- polymorphic type checking, 40
- polymorphic type system, 46, 50
- Ponder, 69
- portability, 1
- predicate
  - non-admissible, 36
- predicates
  - admissible, 33
  - chain complete, 33
- prefix operator, using + as a, 8
- prefix ordering, 32
- prime numbers, 96, 182
- primes, 96
  - derivation of, 183
  - infinite process network, 136
- prioritised equations, 8
- problem decomposition tree, 106
- process, 7
- process network
  - chain, 130
  - cyclic, 104
  - in PARALFL, 156
  - reconfiguration, 154
  - static, 103
- process networks, 85, 123
  - declarative descriptions, 126
  - dynamic, 154
  - dynamic, example, 156
  - infinite, 96
  - semi-static, 153
- process placement
  - run time, 96
  - static, 85
- process pool, 65
- process separation, 147
- processes, 65
  - and channels, 148
- programming environment, 121
  - transformation-based, 178
- programming environments, 158
- projectors, 101
- propagation delay, 99, 124
- proto-channels, 149
- pruning, 45
- PUSHDUMMY, 62
- PUSHGRAPH, 59
- PUSHPARAM, 59
- PUSHSTATUS label, 63
- PUSHVALUE, 59

- quantum computation, 159
- quantum theory, 228
- Quarendon, 96
- Quarendon, P., 96
- QuickSort, 84
- QuickSort, 79
- Quicksort, 78, 159
- Quinton, 120
  
- RABBIT, 70
- race conditions, in garbage accounting, 72
- racing, 13, 41, 45
  - not normally implemented, 51
- Ramamoorthy, 119
- ray casting, 154
- ray intersection test, 92, 111, 112, 116, 120, 124, 125, 153
  - pipelined, 178
- ray tracer
  - semi-static process network, 153
- ray tracing, 120
  - and partial evaluation, 160
  - smart algorithms for, 120
- ray-tracing, 111
- rays
  - contributory, 112
- RayTracer, 113, 116, 153
- Rayward-Smith, 119
- recalculation, 67
- receiver, 148
- reconfigurable networks, 147
- reconfiguration, 154
- records, with inheritance, 46
- recurrences, 18
  - sub can be removed, 20, 85, 174
  - iterate captures simple, 20
  - and I-structures, 45
  - idiom, 85
  - in bitwise addition, 101
  - in Gauss-Seidel method, 23, 45
  - in Newton-Raphson, 87
  - mixing list and vector, 23
  - syntactic sugar, 119
  - vector and matrix, 22, 23
- recursion induction, 193
- recursive definitions
  - as feedback in hardware, 100
  - chain of iterations, 31
  - in **where** clauses, 11
  - of lists, 7
  - of non-function values, 21
  - of types, 6
  - semantics, 28, 29
- redex, 11, 51, 57
  - locating the next, 56, 57
- REDIFLOW, 74
- reducible expression, 11
- reducible function application, 13
- reduction, 11
  - data-driven, 73
  - normal-order, 51
- reference counting, 71, 72
  - race conditions, 72
- referential transparency, 40, 156
- refraction and reflection, 111
- refutable, pattern, 223
- Register, 99
- register, 99
- register allocation, 59
- replicate, 17, 212
- research papers, 4
- resource management, 42
- RET, 60
- RETJMP, 60
- returning
  - a boxed object, 60
  - an unboxed object, 60
- reverse, 17, 212
  - total structural induction for, 36
- reverting to sequential code, 82
- ring-shaped process network, 124
- RootsOf, 187
- RUBY, 120
- run time process placement, 96
- Russell, 74
  
- S,K,I combinators, 69
- Sample, 97
- Sarkar, 74, 119
- SASL, 219

- sawtooth, 7
- scaffolding, 102
- scheduler, 66
- scheduling
  - compile-time, 74
  - divide-and-conquer, 119
  - in systolic arrays, 120
  - run-time vs. compile-time, 74
- Schmidt, 31, 44, 120
- scientific models, testing of, 163
- scope, 10, 40, 54, 127, 134
- script, 6, 10, 11, 26, 40, 126, 127, 156
- SECD abstract machine, 70
- select, 213
- SelectBigger, 78
- SelectMatch, 98
- SelectSmaller, 78
- self-applicable partial evaluator, 160
- semantic constraints on parameters, 46
- semantic property
  - deadlock as, 161
- semantics, 29, 54, 120
  - of Caliban, 158
- sender, 148
- sequential multiprocessor, 124
- SERC, viii
- serial combinators, 67
- Shapiro, 158
- shared memory, 118
  - traffic, 125
- shared-memory, 119
- Sheeran, 120
- shuffled tree-list translation, 171
- side-effects, 40, 50
- sieve, 96, 183
- sieve of Eratosthenes, 96, 182
- sieves, 182
- Signal, 97
- SignalCase, 97
- SIMD, 2, 144
- simplification, 50, 71
- SimplySolve, 76
- Simpson's rule, 22
- single assignment, 119
- SISAL, 119
- SizeOfNextGeneration, 188
- SKIM, 69
- slow-down, 67
- snd, 213
- sort
  - parallel, 159
  - parallel interchange, 159
  - Quick, 78, 159
- SourceCode, 58
- South, 23
- space leaks, 72
- space occupied after unfolding, 51
- space usage
  - with parallel graph reduction, 66
- spatial program distribution, 123
- specification
  - of tree-stream translation, 186
- specification languages, 46
- speed-up of a sequential multiprocessor, 124
- split, 168, 213
- SplitStream, 111, 192, 213
- sqrt, 20
  - distributed, 137
- stack, 59, 70
- stack space, 64
- Standard ML, 46
- star network, 124
- starvation, 160
- state transition function, 177
- static networks
  - implementation, 146
- static process network, 103
- Steele, 42, 70
- stepwise refinement, of proofs, 173
- storage class analysis, 50
- storage class optimisation, 63
- Stout, 119
- Stoy, 44
- Stoye, 69
- stream, 7
- stream-processing form, 145
- StreamOfMatricesToMatrixOfStreams, 143, 214
- StreamToListOfMTrees, 117, 186, 190

- StreamToMTree, 108, 117, 214
  - derivation of, 186
- strict
  - languages, 219
- strict applications, compiling, 64
- strict basic operator, 57, 75
- STRICTAPPLY, 64
- strictness, 29
- strictness analysis, 29, 54, 70, 76
  - and boxing analysis, 56
  - and optimisation, 62
  - and space usage, 43
  - improved by partial evaluation, 50
  - of higher-order programs, 70
  - on lists, 71, 150
- strictness annotations, 54
  - and parallel reduction, 65
  - interpretation, 57
  - on formal parameters, 54
- strictness assertions, 150
- strictness information, 84
- strong typing, 9
- structural induction, 166
- sub, 220
- sub, 18, 214
- sub-rays
  - tree of, 112
- SubsequentGenerations, 188
- SubtreesOf, 187
- subtypes, 37, 46
- SUCC, 36
- Sufrin, 47
- SUGAR, 219
- sugar, syntactic, 119
- sum, 17, 79, 215
- SumOf, 101
- supercombinator, 57
- supercombinator abstraction, 69
- supercombinator asbtraction, 53
- Supernode*, 2
- surface model, 112, 114
- surface-to-volume effect, 144
- Sutherland and Mead, 159
- symbolic evaluation, 103
- symbols, glossary of, 197
- synchronisation
  - control, 66, 150
  - delays, 74
  - join, 66
  - optimising out channel, 151
  - spurious, 40
  - with time-slicing, 151
- synchronous
  - digital circuits, 120
  - processor arrays, 120
- synonym, type, 6
- systolic, 120
- tacticals, 121
- tag, 180
- tagged data, unnecessary, 64
- Tagged Token Dataflow, 119
- Tagged-Token Dataflow, 85
- TaggedStreamItem, 110
- tagging, with constructor, 110
- tail recursion, 64, 70
- take, 80, 165, 215
- Takelmpact, 94, 179, 215
- telephone number, 28
- term rewriting, 73
- termination correctness, 27, 28
- Test, J.A., 74
- TestForlmpact, 92, 178, 215
- theorem prover, 121
- Theorem 5, 192
- Theorem 4, 184
- Theorem 3, 179
- Theorem 2, 172
- Theorem 1, 166
- thesis, Ph.D., vii
- Thompson, 159
- tightly-coupled machines, 49, 67, 68
  - as PEs in loosely-coupled, 151
  - process networks for, 123, 155
- time-slicing
  - in Caliban, 150
- tl, 13, 215
- tokens
  - for flow control, 148
- topology, of interconnection network, 2

- total structural induction, 166
- transformation, 2, 39, 164
  - automated support, 28, 118, 121
  - automation of, 177
  - change propagation, 82
  - data type, 80, 107, 165
- transpose, 98, 215
- transputers, 124, 158
- Tree
  - Kleene chain of, 34
- tree
  - binary, 165, 171
  - intermediate, in divide and conquer, 184
  - of meshes, quad-, 155
  - of sub-rays, 112
  - problem decomposition, 106
- TreeInsert, 81
- TreeToList, 80
- TreeToList, shuffled version, 171
- TreeToList1, 80, 165, 216
- TreeToList2, 83, 172, 216
- triangle, 38
- Trivial, 76
- TRUE, 51
- truth tables, 97
- tuples, 6
- tuples, tagged, 223
- Turner, 69
- type
  - (algebraic) data, 6
  - checking, inference, 50
  - equations, 6
  - of curried functions, 10
  - specification, 50
  - specifications, 9
  - strong polymorphic, 5
  - variables, 6
- type expression, 41
- type system
  - polymorphic, Hindley-Milner, 46, 50
  - richer, 46, 50
- type systems
  - polymorphic, 40
  - strong, 40
- type variables, 41, 220
- types
  - collected definitions, 198
- Ullman, 159
- unboxed base-type object, 60
- unboxed values, 55
- unfold rule, 27
- unfolding
  - space occupied after, 51
- universal parallel computer, 159
- universal VLSI machine, 159
- unshared applications, 62
- until, 20, 216
- update problem, 42
- UPDATEGRAPH, 60
- UPDATEVALUE, 60
- updating, *see* overwriting
- upper case, for constructors, 6, 220
- VAL, 119
- Valiant, 159
- value mode code generator, 60
- value-type expression, 58
- ValueType, 58
- van den Broek and van der Hoeven, 74
- variables
  - non-local, non-global, 69
- variants, 64, 65
- variants of functions, for boxlessness, 56
- VDM, 47
- vector pipeline processors, 74
- VectorBound, 21, 216
- VectorLadder, 130
- vectors, 21, 45
- vectors and matrices, 220
- VectorToList, 51, 217
- verification, 39
- vertical parallelism, 75, 84
- Vgen, 61
- virtual memory, 71, 72
- Vitanyi, 158
- VLSI, 96, 120, 124
  - complexity theory, 159
- VLSI algorithms, 159
- von Neumann model, 159

von Neumann programming, 40  
Vree, 119  
  
Wadge, 91, 119, 161  
Wadler, 71–73, 121, 150  
Watson, 65, 67, 72, 73, 151  
wavefront, in Gauss-Seidel computation,  
    23  
weak head normal form, 68  
Weghorst, 120  
Wegner, 46  
weights, in reference counting, 72  
well-founded ordering, 37  
    by generations, 187  
West, 23  
Westfield College, London, vii  
where clauses, 10, 53  
    effect on space use, 73  
    removal of, 50  
Whitted, 111, 112, 120  
Wilensky, 44  
Williams, viii, 46, 158  
window, for parallel evaluation, 76  
“wiring” functions, 102  
Wolfe, 74  
Wray, 56, 69  
Wu and Feng, 66  
Wulf, 68  
  
Young, 157  
  
Z (specification language), 47  
ZERO, 36