

Index

Symbols

- \$
 - in lex pattern 55
- \$\$, in yacc grammar 178
- \$1, \$2, \$3,... in yacc grammar 178
- {... %}
 - yacc declarations section 178
- %left 179
 - miniC compiler 279
- %left directive in yacc 199
- %right 179
 - miniC compiler 279
- %right directive in yacc 199
- %start
 - miniC compiler 276
- %token 198
 - miniC compiler 279
- %type 199
 - miniC compiler 279
- %union 198
 - miniC compiler 279
- *
 - reflexive transitive closure of a relation 95–97
- +
- in lex pattern 55
- /
 - in lex pattern 55
- <S>
 - in lex pattern 56
- ?
 - in lex pattern 55
- []
 - in lex pattern 55
- [a-z]
 - in lex pattern 55
- \
 - in lex pattern 55
- ^
 - in lex pattern 55
- {m,n}
 - in lex pattern 56
- {name}
 - in lex pattern 56
- |
 - in lex pattern 55
- A**
- absolute address mode
 - Mini architecture 226
- action
 - yacc grammar 177
- action symbol 133
- action table
 - LR parsing 171
- actions
 - finite state machine 46
 - for lex 55–57
- ADD atom 228
- ADD, Mini Instruction 227
- address
 - target machine for MiniC 197
- addressing modes 210
 - Mini architecture 226
- algebraic local optimization 253
- algebraic transformations 244
- alloc function
 - arithmetic expression translation to atoms 146
 - MiniC compiler 276
 - parser for MiniC 159
- ambiguous
 - if-then-else statement
 - parsing bottom up 168
- ambiguous grammar 75
 - programming languages 87–89
 - resolution with yacc 178, 179
- ambiguous grammar, eliminating
 - arithmetic expressions 87
 - if-then-else statements 87–89
- architecture 204
- arithmetic expression
 - attributed translation grammar 145–148
 - LL(1) grammar 125–132
 - MiniC
 - yacc rules 200
 - parsing bottom up 171
 - precedence in yacc 179
 - recursive descent translator 146–148
 - registers needed 221–224
 - top down parsing 123–129
 - translation to atoms with yacc 177–184
- arithmetic expressions
 - eliminating ambiguity 87
- array references 191–194
- assignment operator 149
- atom 68
- atom file
 - input to code generator 228–229
- atom file format
 - MiniC 228
- atom() function to generate atoms 179
- atom() function 200
- atoms 9–11
 - MiniC 296
- attribute
 - inherited 140–142
 - synthesized 140–142
- attribute computation rule 140–144

attributed grammar 140–142
 array references 193–194
 recursive descent parser 142–144
 attributed translation grammar
 arithmetic expression 145–148
 automata theory 32

B

, bottom of stack marker 78
 back end 22, 204, 204–205, 234
 Backus-Naur Form 74
 basic block 237–241
 BDW relation 114
 begins directly with 114
 begins with 114
 binary search tree, for lexical tables 50
 bisect.c 273
 bison 176
 BNF 74
 boot()
 Mini simulator 290, 295
 bootstrapping 20–22
 bottom up parsing 164–190
 summary 203
 bottom-up parsing algorithm 92
 build_labels
 MiniC code generator 284
 build_labels() 229
 BW relation 114

C

character constant 40
 Chomsky, Noam 71
 class
 of token 41
 closure. *See* Kleene *, for regular expressions
 relations 95–97
 CLR, Mini instruction 227
 CMP. Mini Instruction 227
 code generation 13–14, 204–209
 common subexpressions 221
 MiniC 226–231
 input file of atoms 228–229
 summary 232
 code generator
 invoked from parser 197
 MiniC compiler 284–289
 code_gen() 197, 229
 MiniC code generator 284
 comment 40
 comments
 miniC 63
 common subexpressions, code generation 221
 compare field, Mini instruction format 227
 comparison operators
 miniC 63

comparisons 149
 compilation
 concise notation for 19
 compile time 4
 compiler
 big C notation 5
 concise notation for 5
 definition 1–2, 29
 examples 2
 compiler-compiler 23, 101, 176
 compiler-compiler (yacc) 176–190
 concatenation, of regular expressions 35
 conflict
 reduce/reduce 167
 shift/reduce 167
 constant folding 244
 constants 0.0 and 1.0
 MiniC 200
 context free grammar 74–76
 context free language
 deterministic 83
 context-free grammar 72
 parsing algorithms 92
 context-free language 73
 context-sensitive grammar 72
 example 73
 context-sensitive language 73
 control structure
 yacc grammar for MiniC 199–200
 control structures
 translating to atoms 153–158
 conventional machine language 204
 conversion of atoms to instructions 210–213
 cos.c 273
 cosine program
 compiling with MiniC 275
 cosine program in MiniC 272
 CPU
 Mini architecture 226–227
 cross compiling 22

D

DAG (directed acyclic graph) 237–241
 data flow analysis 242
 dead code, elimination of 242
 debugging and optimization 235
 declarations section, yacc source file 177
 DEO relation 116
 derivation 69
 derivation tree 75
 deterministic context free languages 83
 deterministic pushdown machine 79
 direct end of 116
 directed acyclic graph (DAG) 238
 DIV atom 228
 DIV. Mini Instruction 227

- dump()
 - Mini simulator 290, 294
- dump_mem()
 - MiniC code generator 284
- dumpmem()
 - Mini simulator 290, 294
- dumpregs()
 - Mini simulator 294
- E**
- elimination of dead code 242
- empty set 31
- end of 116
- endmarker
 - FB relation 117
- endmarker, for pushdown machines 78
- EO relation 116
- epsilon rule
 - parsing 106, 109
 - translation grammar 133
- epsilon rules
 - parsing quasi-simple grammar 107
- equivalent grammars 71
- equivalent programs 2
- example of a nonterminal 101
- exit, from pushdown machine 78
- expression trees
 - yacc 182–186
- expressions
 - MiniC 149–150
 - attributed translation grammar 150
- extended pushdown machine 79–80
- extended pushdown translator 80
- F**
- fact.c 273
- FB 116–117
- FDB relation 115
- finite state machine 31–33
 - example of 32–33
 - implementation for lexical analysis 44–46
 - table representation 33
 - with actions 46
- first (x) 114–115
- first of right side 115
- fixup table, for forward jumps in code generation 214
- Fol(A) 117
- follow set 106, 118
 - LL(1) grammar 117
- followed by 116–117
- followed directly by 115
- for statement
 - translation to atoms 153–158
- formal language 30
- forward jumps
 - fixup table in single pass code generator 214

- forward jumps, in code generation 214
- forward references
 - MiniC code generator 284
- front end 22, 204
- ftp 274

G

- gen () function
 - Mini code generator 229
- gen()
 - MiniC code generator 284
- gen.c 273
 - MiniC code generator 284–289
- global optimization 12–13, 233, 237–250
 - effect on debugging 13
- goto table
 - LR parsing 171
- grammar
 - classes 71–76
 - definition 69
 - examples 70–71
 - LL(2) 150
 - LR 165
 - LR(k) 167. *See also* deterministic
 - quasi-simple 106–109
 - simple 98–102
 - yacc 177

H

- handle
 - shift reduce parsing 165
- hash function 50–52
- hash table, for lexical tables 50–52
- header files for MiniC compiler 197–198
- high level language
 - advantages over assembly language 3
 - disadvantages versus assembly language 3
- high-level language 2
- HLT. Mini Instruction 227

I

- identifier 40
 - accepted by finite state machine 44
- identifiers
 - miniC 63
- if-then-else statement
 - ambiguity 168
 - translation to atoms 153–158
- implementation techniques 19–22, 29
- in lex pattern 55
- infix expression 80
 - attributed translation grammar 145
 - translation to atoms with yacc 177–184
- infix to postfix expressions 133–134
- inherited attributes 140–142

input alphabet 69
 pushdown machine 77
 input symbol 69
 instruction register
 Mini computer 290
 Intermediate form
 Java Virtual Machine 23
 intermediate form 22
 interpreter 3

J

Java Virtual Machine 23
 JMP atom 228
 jmp atom 153
 JMP. Mini Instruction 227
 jump over jump optimization 252

K

key word 40
 key words
 miniC 62
 keyword
 accepted by finite state machine 45
 Kleene *, for regular expressions 35–36
 nested 37

L

label atom 10
 label table, in code generation 214
 language 31
 context-free 73
 context-sensitive 73
 right linear 73
 simple 98
 LBL atom 228
 lbl atom 153
 left associative
 operations 199
 left precedence (associativity) in yacc 179
 left recursion 124–125
 left-most derivation 76
 lex 54–59
 actions 55–57
 example 58–59
 example of use with yacc 178–184
 execution 60
 in MiniC compiler 275–279
 library 60
 miniC
 supporting functions 63
 patterns 55–57
 section 1 of source file 54–55
 section 2 of source file 55–57
 section 3 of source file 57
 sections of source file 54

lex.yy.c
 included in a yacc program 179
 lexeme. *See* token
 lexical analysis
 summary 67
 lexical analysis 9, 30, 40
 MiniC compiler 275
 lexical item. *See* token
 lexical scanner. *See* lexical analysis
 lexical tables 50
 lexical token 40
 LL(1) grammar 113–115, 118
 arithmetic expression 125–132
 parsing
 pushdown machine 119
 recursive descent 119–121
 LL(2) grammar 150
 load/store architecture
 converting atoms to instructions 210
 load/store optimization 251
 local optimization 14–15, 233, 251–255
 locate 273
 LOD. Mini Instruction 227
 lookup() function
 Mini code generator 229
 loop invariant 13
 loop invariant code 243
 LR grammar 165
 LR parsing, with tables 171–175
 LR(k) parser, grammar 167
 lvalue 150

M

machine language 2
 makefile 273
 matrix references 191–194
 Mini
 CPU registers 297
 instruction format 291, 297
 memory 297
 operation codes 297
 Mini (computer)
 simulator for 290–297
 Mini machine
 code generation from MiniC 226–231
 lookup() function 229
 reg() function 229
 multiple pass code generator 229
 sample program 228
 Mini, simulated architecture 226–228
 mini.c 273
 Mini simulator source file 291–297
 mini.h 273
 Mini simulator header 290, 296
 mini.h, header file for MiniC compiler 197
 MiniC 26, 29

- arithmetic expression
 - yacc 200
- atoms 296
- code generator 226–231
- compiler 273–289
 - code generator 284–289
 - execution of 275
 - lexical phase 275–279
 - software files 273–275
 - syntax phase 279–284
- definition 270–272
- expressions 149–150
 - attributed translation grammar 150
- format of atom file 228
- lexical analysis 62–65
- multiply declared identifiers 200
- parser 159–162
- sample program 272
- symbol table 276
 - declaration in header file 295
- syntax error 200
- translating control structures to atoms 199–200
- undeclared identifiers 200
- yacc
 - supporting functions 200–201
- yacc parser 197–201
 - yacc grammar 199–200
- yacc stack type 198
- miniC
 - comments 63
 - comparison operators 63
 - identifiers 63
 - key words 62
 - lex
 - supporting functions 63–64
 - lexical structure 63
 - numeric constants 63
- miniC.c 273
- miniC.h 273
 - MiniC and Mini header file 295
- miniC.h, header file for MiniC compiler 197
- miniC.l 273, 275
- miniC.y 197, 273
 - MiniC compiler 279
- mk 273
- MOV atom 228
- mov atom 153
- MUL atom 228
- MUL, Mini instruction 227
- multiple pass code generator 214–220, 215
 - Mini machine 229
- multiple pass compiler 15
- multiply declared identifiers
 - MiniC 200

N

N

- endmarker 78
- natural language 30
- NEG atom 228
- newlab function
 - parser for MiniC 159
- newline 40
- nondeterministic pushdown machine 79, 83
- nonterminal symbol 69
- normal form, for derivations 76
- null string 31
- nullable nonterminal 113–114
- nullable rule 113–114
- numeric constant 40
 - accepted by finite state machine 44
- numeric constants
 - miniC 63
 - MiniC compiler 276
 - storage in MiniC 296

O

- object language 2
- object program 2
- offset computation for arrays 191–194
- operation codes
 - Mini computer 297
- operator 40
- optimization 12–13, 233–236
 - and debugging 235
 - global 233, 237–250
 - local 233, 251–255
 - jump over jump 252
 - load/store 251
 - simple algebraic 253
 - summary 256
- out_mem() 229
 - MiniC code generator 284
- output function for pushdown machine 79–80

P

- palindrome
 - grammar 70
 - with center marker 81
- parenthesis language 79
- parity bit generator 46
- parser. *See* syntax analysis
 - miniC 159–162
- parser generator, yacc 176–190
- parsing
 - arithmetic expression
 - top down 123–129
 - bottom up 164–190
 - summary 203
 - epsilon rule 106

- LL(1) grammar
 - pushdown machine 119
 - recursive descent 119–121
- quasi-simple grammar
 - pushdown machine 107–108
 - shift reduce 164–170
- parsing algorithm
 - definition 92
 - simple grammar 99–101
- parsing problem 92
- Pascal 30
- patterns
 - for lex 55–57
- pc, program counter in Mini code generator 229
- peephole optimization. *See* local optimization
- phases 9–14, 29
- pop operation 77
- postfix expression 80
- precedence
 - specified with yacc 199
- prefix expressions
 - attributed grammar 140
- production. *See* rewriting rule
- program counter
 - Mini computer 290
- programming language 2
- push operation 77
- pushdown machine
 - definition 76–78
 - examples 78–79
 - extended 79
 - with output operation 79
- pushdown translator 79–80

Q

- quasi-simple grammar 106–109
 - parsing with pushdown machine 107–108
 - parsing with recursive descent 108–109

R

- readme 273
- recursive descent
 - arithmetic expression
 - translation to atoms 146–148
 - attributed grammar 142–144
 - MiniC parser 159–162
 - translation grammar 134–137
- recursive descent parsing
 - LL(1) grammar 119–121
 - quasi-simple grammar 108–109
 - simple grammar 101–102
- reduce operation, parsing bottom up 164, 171
- reduce/reduce conflict 167
- reduction in strength 244
- reflexive relation 96
- reflexive transitive closure 95–97

- reg() function
 - Mini code generator 229
- register allocation 13, 204, 221–225
 - arithmetic expression evaluation 221–224
 - MiniC compiler 285
- register-displacement address mode
 - Mini architecture 226
- regular expression 35–37
- relation 95–97
- rewriting rule 69
- right linear grammar 72
- right linear language 73
- right precedence (associativity) in yacc 179
- RISC machine
 - register allocation 221
- rules section of lex program 55–57
- rules section, yacc source file 177
 - MiniC 199–200
- run time. 4

S

- scanner. *See* lexical analysis
- searchIdent()
 - in MiniC compiler 276
- searchIdent() function
 - miniC
 - lex 65
- searchNums()
 - in MiniC compiler 276
- searchNums() function
 - miniC
 - lex 65
- selection set
 - definition 98
 - LL(1) grammar 113–115, 117
 - quasi-simple grammar 106
- semantic analysis 12, 197
- semantics 133–134
- sentential form 69
- sequential search
 - for lexical table 50
- set 30
- shift operation, parsing bottom up 164, 171
- shift reduce parsing 164–170
- shift/reduce conflict 167
- simple algebraic optimization 253
- simple grammar 98–102
 - parsing with pushdown machine 99–101
 - recursive descent parsing 101–102
- simple language 98
- single pass code generator 214–220
 - fixup table 214
- single pass compiler 15
- software
 - distribution rights 230
- source language 2

- source program 2
- special character 40
- sscanf() 58
- stack, for pushdown machines 77
- start condition
 - in lex pattern 56
- starting nonterminal. 69
- starting state
 - pushdown machine 76
- STO. Mini Instruction 227
- string 31
- SUB atom 228
- SUB, Mini instruction 227
- supporting functions
 - lex
 - miniC 63
 - yacc
 - MiniC 200–201
- symbol table 50
 - MiniC 276
 - declaration 295
- syntax 2
- syntax analysis 9–11, 68, 93
 - MiniC compiler 279–284
- syntax directed translation 68, 95, 133
- syntax error
 - MiniC 200
- syntax tree 68
- syntax tree, weighted
 - register allocation 221
- syntax trees 9–11
- synthesized attributes 140

T

- tar file 275
- target machine 2
 - Mini 290–297
- terminal symbol 69
- token 40
 - class 41
 - value 41
- tokens
 - for MiniC 275
- tools 19
- top down parser
 - MiniC 159–162
- top down parsing 94
 - arithmetic expression 123–129
 - summary 163
- top-down parsing algorithm 92
- transfer of control with atoms 10
- transitive relation 95
- translation
 - control structures 153–158
 - infix to postfix 133–134
 - syntax directed 133

- translation grammar 133
 - array references 193–194
- attributed grammar
 - arithmetic expression 145–148
 - recursive descent 134–137
- traversal of syntax trees 11
- TST atom 228
- tst atom 153

U

- undeclared identifiers
 - MiniC 200
- underlying grammar, of translation grammar 133
- union, of regular expressions 35
- unreachable code 242
- unrestricted grammar 72
- user interface 1

V

- value
 - of token 41

W

- weighted syntax tree
 - register allocation 221
- while statement
 - translation to atoms 153–158
 - translation to atoms with yacc 199–200
- white space 40
- word. *See* token

Y

- y.tab.c 197
- yacc 54
 - example 177–184
 - expression trees example 182–186
 - grammar for miniC 280
 - infix to prefix expression translation 182–186
 - left and right precedence of operators 179
 - MiniC
 - arithmetic expression 200
 - supporting functions 200–201
 - MiniC compiler 279–284
 - MiniC grammar rules and actions 199–200
 - nonterminal type declaration 199
 - parser for MiniC 197–201
 - PC version with Pascal 176
 - public domain version 176
 - source file 177
 - stack type
 - MiniC 198
 - token type declarations 198
- yacc parser generator 176–190
- yyerror() function 179, 200
- yylen 57

- yylex() 54
- yylex() function 197
 - called from yyparse() 179
- yylval
 - MiniC compiler 276
 - use with yacc 179
- yyparse() 54, 179
- yyparse() function 197
- YYSTYPE 178
- yytext 57