

# Index

## Symbols

- \*
  - closure, in SableCC 56
  - reflexive transitive closure of a relation 97–99
- +
  - restricted closure, in SableCC 56
- ?
  - optional entry, in SableCC 56
- |
  - alternate rule in SableCC 56

## A

- absolute address mode
  - Mini architecture 230
- action symbol 136
- action table
  - LR parsing 178
- actions
  - finite state machine 46–48
- ADD atom 232
- ADD, Mini Instruction 231
- addressing modes 215
  - Mini architecture 230
- algebraic local optimization 257
- algebraic transformations 248
- alloc function
  - parser for Decaf 166
- ambiguous
  - if-then-else statement
    - parsing bottom up 175
- ambiguous grammar 77
  - programming languages 89–91
- ambiguous grammar, eliminating
  - arithmetic expressions 89
  - if-then-else statements 89–91
- architecture 209
- arithmetic expression
  - attributed translation grammar 149–154
  - LL(1) grammar 128–135

- parsing bottom up 178
  - recursive descent translator 150–154
  - registers needed 225–228
  - top down parsing 126–132
- arithmetic expressions
  - eliminating ambiguity 89
- array references 199–202
- assignment operator 155
  - translation to atoms 157
- atom 70
  - JMP 155
  - LBL 155
  - TST 155
- atom file
  - input to code generator 232–233
- atom file format
  - Decaf 232
- Atom.java
  - for Decaf compiler 295
- AtomFile.java
  - for Decaf compiler 297
- atoms 9–11
  - for Decaf control structures 155
  - generating code from 230–235
  - MiniC 311
- attribute
  - inherited 143–145
  - synthesized 143–145
- attribute computation rule 143–148
- attributed grammar 143–145
  - array references 201–202
  - recursive descent parser 145–146
- attributed translation grammar
  - arithmetic expression 149–154
- automata theory 32

## B

- , bottom of stack marker 80
- back end 22, 209, 209–210, 238
- Backus-Naur Form 76
- basic block 241–245
- BDW relation 117
- begins directly with 117
- begins with 117
- binary search tree, for lexical tables 50

- bisect.decaf 277
- BNF 76
- boolean expression
  - in Decaf implementation 206
- boolean expressions
  - in Java 155
  - translation to atoms 156
- boot()
  - Mini simulator 305, 310
- bootstrapping 20
- bottom up parsing 171–198
  - summary 208
- bottom-up parsing algorithm 94
- build\_labels
  - MiniC code generator 299
- build\_labels()
  - in code generator 233
- BW relation 117
  
- C**
  
- character constant 40
- Chomsky, Noam 73
- class
  - of token 41
- closure. *See* Kleene \*, for regular expressions
  - relations 97–99
- CLR, Mini instruction 231
- CMP. Mini Instruction 231
- code generation 13–14, 209–214
  - common subexpressions 225
  - Decaf
    - input file of atoms 232–233
    - from atoms 230–235
    - summary 236
- code generator
  - Decaf compiler 299–304
- code\_gen() 233
- comment 40
- comments
  - implementing in SableCC 65
  - tokens in Decaf 205
- common subexpressions, code generation 225
- compare field, Mini instruction format 231
- comparisons
  - results as booleans 155

- compilation
  - concise notation for 19
- compile script
  - Decaf 278
- compile time 4
- compileAndGo script
  - for Decaf 278
- compiler
  - big C notation 5
  - concise notation for 5
  - definition 1–2, 29
  - examples 2
- compiler-compiler 23, 103
- compiler-compiler (SableCC) 183, 183–198, 205
- Compiler.java
  - for Decaf using SableCC 294
- concatenation, of regular expressions 35
- conflict
  - reduce/reduce 174
  - shift/reduce 174
- constant folding 248
- context free grammar 76–78
- context free language
  - deterministic 85
- context-free grammar 74
  - parsing algorithms 94
- context-free language 75
- context-sensitive grammar 74
  - example 75
- context-sensitive language 75
- control structures
  - translating to atoms 160–165
- conventional machine language 209
- conversion of atoms to instructions 215–218
- cos.decaf 277
- cosine program
  - compiling with Decaf 278–280
- cosine program in Decaf 275–276
- CPU
  - Mini architecture 230–231
- cross compiling 22
  
- D**
  
- DAG (directed acyclic graph) 241–245
- dangling else

## Index 318

- parsing bottom-up 205
- data flow analysis 246
- dead code, elimination of 246
- debugging and optimization 239
- Decaf 26, 29
  - compiler 277–304
    - build and execute 278–280
    - code generator 299–304
    - software files 277–280
  - definition 274–276
  - download source files 278
  - expressions 155–159
    - attributed translation grammar 157
  - format of atom file 232
  - lexical analysis 65–66
  - parser
    - top-down 166–169
  - SableCC parser 205–206
  - sample program 275–276
  - source code 281
- Decaf.grammar
  - explanation 205
- decaf.grammar 281
- DEO relation 119
- derivation 71
- derivation tree 77
- deterministic context free languages 85
- deterministic pushdown machine 81
- direct end of 119
- directed acyclic graph (DAG) 242
- DIV atom 232
- DIV. Mini Instruction 231
- dump()
  - Mini simulator 305, 309
- dump\_mem()
  - MiniC code generator 299
- dumpmem()
  - Mini simulator 305, 309
- dumpregs()
  - Mini simulator 309

## E

- EBNF
  - in Decaf implementation 205
- elimination of dead code 246

- embedded action
  - in SableCC 188
- empty set 31
- end of 119
- endmarker
  - FB relation 120
- endmarker, for pushdown machines 80
- EO relation 119
- epsilon rule
  - parsing 109, 112
  - translation grammar 136
- epsilon rules
  - parsing quasi-simple grammar 110
- equivalent grammars 73
- equivalent programs 2
- example of a nonterminal 103
- exit, from pushdown machine 80
- expressions
  - Decaf 155–159
    - attributed translation grammar 157
- extended pushdown machine 81–82
- extended pushdown translator 82

## F

- fact.decaf 277
- FB 119–120
- FDB relation 118
- finite state machine 31–33
  - example of 32–33
  - implementation for lexical analysis 44–45
  - table representation 33
  - with actions 46–48
- first (x) 117–118
- first of right side 118
- fixup table, for forward jumps in code generation 219
- Fol(A) 120
- follow set 109, 121
  - LL(1) grammar 120
- followed by 119–120
- followed directly by 118
- for statement
  - translation to atoms 160–165
- formal language 30
- forward jumps
  - fixup table in single pass code generator 219

forward jumps, in code generation 219  
 forward references  
   MiniC code generator 299  
 front end 22, 209

## G

gen () function  
   Mini code generator 233  
 gen()  
   MiniC code generator 299  
 gen.c 277  
   Decaf code generator 299–304  
   source code for code generator 300  
 global optimization 12–13, 237, 241–254  
   effect on debugging 13  
 goto table  
   LR parsing 178  
 grammar  
   classes 73–78  
   Decaf (forSableCC) 277  
   definition 71  
   examples 72–73  
   for Decaf 274  
   LL(2)  
     for Decaf expressions 157  
   LR 172  
   LR(k) 174. *See also* deterministic  
   quasi-simple 109–112  
   simple 100–104

## H

handle  
   shift reduce parsing 172  
 hash function 50–52  
 hash table, for lexical tables 50–52  
 Hashtables  
   with SableCC 206  
 hashtables  
   in Decaf compiler 291  
 Helper declarations  
   in SableCC 57  
 Helpers  
   in Decaf grammar 281  
 high level language  
   advantages over assembly language 3

  disadvantages versus assembly language 3  
 high-level language 2  
 HLT. Mini Instruction 231

## I

identifier 40  
   accepted by finite state machine 44  
 if-then-else statement  
   ambiguity 175  
   translation to atoms 160–165  
 implementation techniques 19–22, 29  
 infix expression 82  
   attributed translation grammar 149  
 infix to postfix expressions 136–137  
 infix-to-atoms translation  
   using SableCC 190  
 inherited attributes 143–145  
 input alphabet 71  
   pushdown machine 79  
 input symbol 71  
 instruction register  
   Mini computer 305  
 Intermediate form  
   Java Virtual Machine 22  
 intermediate form 22  
 interpreter 3

## J

Java Virtual Machine 22  
 JMP  
   atom 155  
 JMP atom 160, 232  
 JMP. Mini Instruction 231  
 jump over jump optimization 256

## K

key word 40  
 keyword  
   accepted by finite state machine 46  
 Kleene \*, for regular expressions 35–36  
   nested 37

## L

label atom 10

## Index 320

label table, in code generation 219  
language 31  
    context-free 75  
    context-sensitive 75  
    right linear 75  
    simple 100  
LBL  
    atom 155  
LBL atom 160, 232  
Left Context  
    in lexical analysis with SableCC 58  
left recursion 127–128  
left-most derivation 78  
lexeme. *See* token  
lexical analysis  
    summary 69  
lexical analysis 9, 30, 40  
    example with SableCC 60  
    SableCC 54–63  
lexical item. *See* token  
lexical scanner. *See* lexical analysis  
lexical tables 50  
lexical token 40  
LL(1) grammar 116–118, 121  
    arithmetic expression 128–135  
    parsing  
        pushdown machine 122  
        recursive descent 122–124  
LL(2) grammar  
    for Decaf expressions 157  
load/store architecture  
    converting atoms to instructions 215  
load/store optimization 255  
local optimization 14–15, 237, 255–259  
LOD. Mini Instruction 231  
lookup() function  
    Mini code generator 233  
loop invariant 13  
loop invariant code 247  
LR grammar 172  
LR parsing, with tables 178–182  
LR(k) parser, grammar 174  
lvalue 157

## M

machine language 2  
matrix references 199–202  
Mini  
    CPU registers 312  
    instruction format 306, 312  
    memory 312  
    operation codes 311  
Mini (computer)  
    simulator for 305–312  
Mini machine  
    code generation from atoms 230–235  
    lookup() function 233  
    reg() function 233  
    multiple pass code generator 233  
    sample program 232  
Mini, simulated architecture 230–232  
mini.c 277  
    Mini simulator source file 306–312  
mini.h 277  
    header file for mini simulator 311  
    Mini simulator header 305  
MiniC  
    atoms 311  
    symbol table  
        declaration in header file 310  
miniC.h  
    header file for mini simulator 310  
MOV atom 232  
mov atom 160  
MUL atom 232  
MUL, Mini instruction 231  
multiple pass code generator 219–224, 220  
    Mini machine 233  
multiple pass compiler 15  
**N**  
N  
    endmarker 80  
natural language 30  
NEG atom 232  
newlab function  
    parser for Decaf 166  
newline 40

nondeterministic pushdown machine 81, 85  
 nonterminal symbol 71  
 normal form, for derivations 78  
 null string 31  
 nullable nonterminal 116–117  
 nullable rule 116–117  
 numeric constant 40  
   accepted by finite state machine 45  
 numeric constants  
   storage in MiniC 311

**O**

object language 2  
 object program 2  
 offset computation for arrays 199–202  
 operation codes  
   Mini computer 311  
 operator 40  
 optimization 12–13, 237–240  
   and debugging 239  
   global 237, 241–254  
   local 237, 255–259  
     jump over jump 256  
     load/store 255  
     simple algebraic 257  
   summary 260  
 out\_mem()  
   in code generator 233  
   MiniC code generator 299  
 output function for pushdown machine 81–82

**P**

palindrome  
   grammar 72  
   with center marker 83  
 parenthesis language 81  
 parity bit generator 48  
 parser. *See* syntax analysis  
   Decaf  
     top-down 166–169  
 parser generator, SableCC 183–198  
 parsing  
   arithmetic expression  
     top down 126–132

bottom up 171–198  
   summary 208  
 epsilon rule 109  
 LL(1) grammar  
   pushdown machine 122  
   recursive descent 122–124  
 quasi-simple grammar  
   pushdown machine 110–111  
 shift reduce 171–177  
 parsing algorithm  
   definition 94  
   simple grammar 101–103  
 parsing problem 94  
 Pascal 30  
 pc, program counter in Mini code generator 233  
 peephole optimization. *See* local optimization  
 phases 9–14, 29  
 pop operation 79  
 postfix expression 82  
 postfix expressions  
   example using SableCC 185  
 prefix expressions  
   attributed grammar 143  
 production. *See* rewriting rule  
 Productions  
   in Decaf grammar 282  
 program counter  
   Mini computer 305  
 programming language 2  
 push operation 79  
 pushdown machine  
   definition 78–80  
   examples 80–81  
   extended 81  
   with output operation 81  
 pushdown translator 81–82

**Q**

quasi-simple grammar 109–112  
   parsing with pushdown machine 110–111  
   parsing with recursive descent 111–112

**R**

readme 278

## Index 322

- recursive descent
  - arithmetic expression
    - translation to atoms 150–154
  - attributed grammar 145–146
  - Decaf parser 166–169
  - translation grammar 137–140
- recursive descent parsing
  - LL(1) grammar 122–124
  - quasi-simple grammar 111–112
  - simple grammar 103–104
- reduce operation, parsing bottom up 171, 178
- reduce/reduce conflict 174
- reduction in strength 248
- reflexive relation 98
- reflexive transitive closure 97–99
- reg() function
  - Mini code generator 233
- register allocation 13, 209, 225–229
  - arithmetic expression evaluation 225–228
  - MiniC compiler 299
- register-displacement address mode
  - Mini architecture 230
- regular expressions 35–37
  - in SableCC 56
- relation 97–99
- rewriting rule 71
- right context
  - in lexical analysis with SableCC 60
- right context specification
  - in SableCC 58
- right linear grammar 74
- right linear language 75
- RISC machine
  - register allocation 225
- run time. 4

## S

- SableCC
  - advantages over JavaCC 54
  - altering of names 189
  - dangling else 205
  - embedded actions 188
  - example 185
  - example for lexical analysis 60–61
  - execution 61–62

- Helper declarations 57
- Input File 54
- left context specification 58
- lexical analysis 54–63
  - parser for Decaf 205–206
  - right context specification 58, 60
  - source file 184–185
  - State declarations 58
  - token declarations 55–56
- SableCC parser generator 183–198
- scanner. *See* lexical analysis
- selection set
  - definition 100
  - LL(1) grammar 116–118, 120
  - quasi-simple grammar 109
- semantic analysis 12, 205
- semantics 136–137
- sentential form 71
- sequential search
  - for lexical table 50
- set 30
- shift operation, parsing bottom up 171, 178
- shift reduce parsing 171–177
- shift/reduce conflict 174
- simple algebraic optimization 257
- simple grammar 100–104
  - parsing with pushdown machine 101–103
  - recursive descent parsing 103–104
- simple language 100
- single pass code generator 219–224
  - fixup table 219
- single pass compiler 15
- software
  - distribution rights 234
- source language 2
- source program 2
- special character 40
- stack, for pushdown machines 79
- starting nonterminal. 71
- starting state
  - pushdown machine 78
- State declarations
  - in SableCC 58
- STO. Mini Instruction 231
- string 31

SUB atom 232  
 SUB, Mini instruction 231  
 Symbol table  
   in Decaf implementation 206  
 symbol table 50  
   MiniC  
     declaration 310  
 syntax 2  
 syntax analysis 9–11, 70, 95  
 syntax directed translation 70, 97, 136  
 syntax tree 70  
 syntax tree, weighted  
   register allocation 225  
 syntax trees 9–11  
 synthesized attributes 143

## T

target machine 2  
   Mini 305–312  
 terminal symbol 71  
 token 40  
   class 41  
   value 41  
 token declarations  
   in SableCC 55  
 Tokens  
   in Decaf grammar 281  
 tools 19  
 top down parser  
   Decaf 166–169  
 top down parsing 96  
   arithmetic expression 126–132  
   summary 170  
 top-down parsing algorithm 94  
 transfer of control with atoms 10  
 transitive relation 97  
 translation  
   control structures 160–165  
   infix to postfix 136–137  
   syntax directed 136  
 Translation class  
   SableCC implementation of Decaf 205  
 translation grammar 136  
   array references 201–202  
   attributed grammar

    arithmetic expression 149–154  
     recursive descent 137–140  
 Translation.java 284–294  
 traversal of syntax trees 11  
 TST  
   atom 155  
 TST atom 160, 232

## U

underlying grammar, of translation grammar 136  
 union, of regular expressions 35  
 unreachable code 246  
 unrestricted grammar 74  
 user interface 1

## V

value  
   of token 41

## W

weighted syntax tree  
   register allocation 225  
 while statement  
   translation to atoms 160–165  
 white space 40  
 word. *See* token