Cgrammar Submission Cgroup – 9

Archaj Jain 2020A7PS0072
Siddharth Khandelwal 2020A7PS0098
Sarthak Shah 2020A7PS0092
Richi Rakesh Shrivastva 2020A7PS0108
Bhanu pratap Rathore 2020A7PS1675

Modified Cgrammar with first & follow

Start Symbol

① <program> → <moduleDeclarations> <otherModules> <driverModule> <otherModules>

FIRST(program) = FIRST(moduleDeclarations) ∪ (FIRST(otherModules) – ε) ∪ FIRST(driverModule)
= {DECLARE} ∪ {DEF, ~~DRIVERDEF~~, ε} ∪ {DRIVERDEF} = {DECLARE, DEF, DRIVERDEF}

FOLLOW(program) = {$}

② <moduleDeclarations> → <moduleDeclaration> <moduleDeclarations> | ε

FIRST(moduleDeclarations) = {DECLARE}, ε}
FOLLOW(moduleDeclarations) = {DEF, $, DRIVERDEF}

③ <moduleDeclaration> → DECLARE MODULE ID SEMICOL

FIRST(moduleDeclaration) = {DECLARE}
FOLLOW(moduleDeclaration) = {DECLARE, DEF, DRIVERDEF, $}

④ <otherModules> → <module> <otherModules> | ε

FIRST(otherModules) = {DEF, ε}
FOLLOW(otherModules) = {DRIVERDEF, $}

⑤ <driverModule> → DRIVERDEF DRIVER PROGRAM DRIVERENDEF <moduleDef>

FIRST(driverModule) = {DRIVERDEF}
FOLLOW(driverModule) = {DEF, $}

⑥ <module> → DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL <ret> <module

FIRST(module) = {DEF}
FOLLOW(module) = {DEF, DRIVERDEF, $}

⑦ <ret> → RETURNS SQBO <output_list> SQBC SEMICOL | ε

FIRST(ret) = {RETURNS, ε}
FOLLOW(ret) = {START}

⑧ <input_plist> → ID COLON <dataType> <N1>

FIRST(input_plist) = {ID}

FOLLOW(input_plist) = {SQBC}

⑨ <N1> → COMMA ID COLON <dataType> <N1> | ε

FIRST → {COMMA, ε}

FOLLOW → {SQBC}

⑩ <output_plist> → ID COLON <type> <N2>

FIRST → {ID}

FOLLOW → {SQBC}

(11) <N2> ⟶ COMMA ID COLON <type> <N2> | ∈

FIRST → { COMMA, ∈ }

FOLLOW → { SQBC }

(12) <dataType> ⟶ INTEGER | REAL | BOOLEAN | ARRAY SQBO <var-range> SQBC OF <type>

FIRST → {INTEGER, REAL, BOOLEAN, ARRAY}

FOLLOW → { COMMA, SQBC, SEMICOL }

(13) <var-range> ⟶ <index> RANGE_OP <index>

FIRST → { ID, NUM }

FOLLOW → { SQBC }

(14) <type> ⟶ INTEGER | REAL | BOOLEAN

First → { INTEGER, REAL, BOOLEAN }

Follow → { COMMA, SQBC, SEMICOL }

(15) <moduleDef> ⟶ START <statements> END

First → { START }

Follow → { DEF, DRIVERDEF, $ }

(16) <statements> ⟶ <statement> <statements> | ∈

First → { GET-VALUE, PRINT, ID, SQBO, USE, DECLARE, SWITCH, FOR, WHILE, ∈ }

FOLLOW → { END }

(17) <statements> ⟶ <ioStmt> | <simpleStmt> | <declareStmt> | <conditionalStmt> | <iterativeStmt>

First → { GET-VALUE, PRINT, ID, SQBO, USE, DECLARE, SWITCH, FOR, WHILE }

FOLLOW → END { FIRST(statements) − ∈ }

(18) <ioStmt> ⟶ GET_VALUE BO ID BC SEMICOL | PRINT BO <var> BC SEMICOL

First → { GET_VALUE, PRINT }

FOLLOW → FOLLOW (statement)

(19) <var> ⟶ <arithematic-Var> | <bool_var>

First → { ID, NUM, RNUM, TRUE, FALSE }

Follow → { BC }

(20) <bool_var> ⟶ TRUE | FALSE

First → { TRUE, FALSE }

Follow → { BC, AND, OR, SEMICOL }

(21) <arithematic_var> ⟶ ID <whichID> | NUM | RNUM

First → { ID, NUM, RNUM }

Follow → { NUL, DIV, PLUS, MINUS, LT, GT, LE, GE, EQ, NE, AND, OR, BC, SEMICOL }

(2) <whichID> ⟶ SQBO <index> SQBC | ∈

First → { SQBO, ∈ }

Follow → FOLLOW (arithmatic_Var)

(3) <simple_stmt> ⟶ <assignment Stmt> | <moduleReuse Stmt>

First → { ID, SQBO, USE }

FOLLOW → FOLLOW (statement)

(24) <assignment Stmt> ⟶ ID <whichStmt>

First → { ID }

FOLLOW → FOLLOW (statement)

(25) <whichStmt> ⟶ <lValue ID Stmt> | <lValue Arr Stmt>

First → { ASSIGNOP, SQBO }

Follow → FOLLOW (assignment Stmt)

First → {ASSIGNOPS}
Follow → FOLLOW (which stmt)

(27) <lvalue Arr Stmt> → SQBO <index> SQBC ASSIGNOP <expression> SEMICOL

First → {SQBO}

Follow → FOLLOW (which stmt)

(28) <index> → NUM | ID

First → {NUM, ID}

Follow → {RANGEOP, SQBC}

(29) <module Reuse Stmt> → <optional> USE MODULE ID WITH PARAMETERS <paramlist> SEMICOL

First → {SQBO, USE}

Follow → FOLLOW (simple stmt)

[assuming A[k] can't be passed as well as taken as function output]

(30) <paramlist> → <mdtlist> <N3>

First → {NUM, ID, RNUM, TRUE, FALSE}

Follow → {SEMICOL}

(31) <N3> → COMMA <mdtlist> <N3> | ε

First → {COMMA, ε}

Follow → {SEMICOL}

(32) <mdtlist> → NUM | RNUM | ID | TRUE | FALSE

First → {NUM, RNUM, ID, TRUE, FALSE}

Follow → {COMMA, SEMICOL}

(33) <optional> → SQBO <idlist> SQBC ASSIGNOP | ε

First → {SQBO, ε}

Follow → {USE}

(34) <idlist> → ID <N4>

First → {ID}

Follow → {SQBC}

(35) <N4> → COMMA ID <N4> | ε

First → {COMMA, ε}

Follow → {SQBC}

(36) <expression> → <arithematic or bool expr> | <unary>

First → {T, F, BO, ID, NUM, RNUM, PLUS, MINUS}

Follow → {SEMICOL}

(37) <unary> → <unary-op> <N5>

First → {PLUS, MINUS}

Follow → {SEMICOL}

(38) <N5> → BO <arithematic Expr> BC | <arithematic_var>

First → {BO, NUM, ID, RNUM}

Follow → {SEMICOL}

(39) <unary-op> → PLUS | MINUS

First → {PLUS, MINUS}

Follow → {BO, NUM, ID, RNUM}

(40) <arithematic or boolexpr> → ~~<boolexpr>~~ | <N6> <boolexpr> <N6>

First → {TRUE, FALSE, BO, ID, NUM, RNUM}

Follow → {SEMICOL, BC}

(41) <NC> ⟶ <logical op> < boolexpr> <NC> | ∈

First → {AND, OR, ∈}

Follow → {SEMICOL, BC}

(42) <boolexpr> ⟶ <bool_var> | <arithmatic Expr><N7>

First → {TRUE, FALSE, BO, ID, NUM, RNUM}

Follow → {AND, OR, SEMICOL, BC}

(43) <N7> ⟶ <rel_op> <arithmatic_expr> | ∈

FIRST → {LT, LE, GT, GE, EQ, NE, ∈}

FOLLOW → {AND, OR, BC, SEMICOL}

(44) <arithmatic_expr> ⟶ <term> <N8>

FIRST → {BO, ID, NUM, RNUM}

FOLLOW → {LT, LE, GT, GE, EQ, NE, AND, OR, BC, SEMICOL}

(45) <N8> ⟶ <op1> <term> <N8> | ∈

FIRST → {PLUS, MINUS, ∈}

FOLLOW → {LT, LE, GT, GE, EQ, NE, AND, OR, BC, SEMICOL}

(46) <term> ⟶ <factor> <N9>

FIRST → {BO, ID, NUM, RNUM}

FOLLOW → {PLUS, MINUS, LT, LE, GT, GE, EQ, NE, AND, OR, BC, SEMICOL}

(47) <N9> ⟶ <op2> <factor> <N9> | ∈

FIRST → {MUL, DIV, ∈}

FOLLOW → FOLLOW (term)

(48) <factor> ⟶ BO <boolExpr> BC | <arithmatic_var>

FIRST → {BO, ID, NUM, RNUM}

FOLLOW → {MUL, DIV, PLUS, MINUS, LT, LE, GT, GE, EQ, NE, AND, OR, BC, SEMICOL}

(49) <OP1> ⟶ PLUS | MINUS

FIRST → {PLUS, MINUS}

FOLLOW → {BO, ID, NUM, RNUM}

(50) <OP2> ⟶ MUL | DIV

FIRST → {MUL, DIV}

FOLLOW → {BO, ID, NUM, RNUM}

(51) <logical op> ⟶ AND | OR

FIRST → {AND, OR}

FOLLOW → {TRUE, FALSE, BO, ID, NUM, RNUM}

(52) <relational op> ⟶ LT | LE | GT | GE | EQ | NE

FIRST → {LT, LE, GT, GE, EQ, NE}

FOLLOW → {BO, ID, NUM, RNUM}

Assuming

a + (b < c)

a * (b <= c)

is syntactically correct

& error will be shown during symantic analysis

<Declare Stmt> → DECLARE <idList> COLON <data Type> SEMICOL

FIRST = {DECLARE}

FOLLOW = FOLLOW (Statement)

(54) <conditional Stmt> → SWITCH BO ID BC START <case Stmts> <default> END

FIRST = {SWITCH}

FOLLOW = FOLLOW (Statement)

(55) <caseStmts> → CASE <value> COLON <Stmt> BREAK SEMICOL <N10>

FIRST = {CASE}

FOLLOW = {DEFAULT, END}

(56) <N10> → CASE <value> COLON <Stmt> BREAK SEMICOL <N10> | ∈

FIRST = {CASE, ∈}

FOLLOW = {DEFAULT, END}

(57) <value> → NUM | TRUE | FALSE

FIRST = {NUM, TRUE, FALSE}

FOLLOW = {~~DEFAULT~~, ~~END~~ COLON}

(58) <default> → DEFAULT COLON <statements> BREAK SEMICOL | ∈

FIRST = {DEFAULT, ∈}

FOLLOW = {END}

(59) <iterative Stmt> → FOR BO ID IN <range> BC START <statements> END |

WHILE BO <arithematic or boolean Expn> BC START <statements> END

FIRST = {FOR, WHILE}

FOLLOW = FOLLOW (Statement)

(60) <range> → NUM RANGEOP NUM

FIRST = {NUM}

FOLLOW = {BC}