## Remove left factoring

program → decl\_list

decl\_list → decl\_list decl | decl

decl → var\_decl | fun\_decl

var\_decl → type\_spec IDENT var\_decl\_

var\_decl\_ →; | [ ] ;

type\_spec → VOID | BOOL | INT | FLOAT

fun\_decl → type\_spec IDENT ( params ) compound\_stmt

params → param\_list | VOID

param\_list → param\_list , param | param

param → type\_spec IDENT param\_

param\_ → ε | []

stmt\_list → stmt\_list stmt | ε

stmt → expr\_stmt | compound\_stmt | if\_stmt | while\_stmt |

return\_stmt | break\_stmt

expr\_stmt → expr ; | ;

while\_stmt → WHILE ( expr ) stmt

compound\_stmt → { local\_decls stmt\_list }

local\_decls → local\_decls local\_decl | ε

local\_decl → type\_spec IDENT local\_decl\_

local\_decl\_ → ; | [ ] ;

if\_stmt → IF ( expr ) stmt if\_stmt\_

if\_stmt\_ → ε | ELSE stmt

return\_stmt → RETURN return\_stmt\_

return\_stmt\_ → ; | expr ;

expr → IDENT expr\_IDENT

→ expr expr\_expr

→ ! expr | - expr | + expr

→ ( expr )

→ BOOL\_LIT | INT\_LIT | FLOAT\_LIT | NEW type\_spec [ expr ]

expr\_IDENT → = expr | [ expr ] = expr | [ expr ] | ( args ) | . size | ε

expr\_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | \* expr | / expr | % expr

arg\_list → arg\_list , expr | expr

args → arg\_list | ε

# Remove left recursion

program → decl\_list

decl\_list → decl\_list decl | decl (before removing left recursion)

decl\_list → decl \_decl\_list (after removing left recursion)

\_decl\_list → decl \_dec\_list | ε

decl → var\_decl | fun\_decl

var\_decl → type\_spec IDENT var\_decl\_

var\_decl\_ →; | [ ] ;

type\_spec → VOID | BOOL | INT | FLOAT

fun\_decl → type\_spec IDENT ( params ) compound\_stmt

params → param\_list | VOID

param\_list → param\_list , param | param

param\_list → param \_param\_list

\_param\_list → , param \_param\_list | ε

param → type\_spec IDENT param\_

param\_ → ε | []

stmt\_list → stmt\_list stmt | ε

stmt\_list → \_stmt\_list

\_stmt\_list → stmt \_stmt\_list | ε

stmt → expr\_stmt | compound\_stmt | if\_stmt | while\_stmt | return\_stmt | break\_stmt

expr\_stmt → expr ; | ;

while\_stmt → WHILE ( expr ) stmt

compound\_stmt → { local\_decls stmt\_list }

local\_decls → local\_decls local\_decl | ε

local\_decls → \_local\_decls

\_local\_decls → local\_decl \_local\_decls | ε

local\_decl → type\_spec IDENT local\_decl\_

local\_decl\_ → ; | [ ] ;

if\_stmt → IF ( expr ) stmt if\_stmt\_

if\_stmt\_ → ε | ELSE stmt

return\_stmt → RETURN return\_stmt\_

return\_stmt\_ → ; | expr ;

expr → expr expr\_expr | IDENT expr\_IDENT | ! expr | - expr | + expr | ( expr ) | BOOL\_LIT | INT\_LIT | FLOAT\_LIT | NEW type\_spec [ expr ]

expr → IDENT expr\_IDENT \_expr | ! expr \_expr | - expr \_expr | + expr \_expr | ( expr ) \_expr | BOOL\_LIT \_expr | INT\_LIT \_expr | FLOAT\_LIT \_expr | NEW type\_spec [ expr ] \_expr

\_expr → expr\_expr \_expr | ε

expr\_IDENT → = expr | [ expr ] = expr | [ expr ] | ( args ) | . size | ε

expr\_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | \* expr | / expr | % expr

arg\_list → arg\_list , expr | expr

arg\_list →expr \_arg\_list

\_arg\_list →, expr \_arg\_list| ε

args → arg\_list | ε

# Final grammar

//start -> program | stmt\_list

program → decl\_list

decl\_list → decl \_decl\_list

\_decl\_list → decl \_dec\_list | ε

decl → type\_spec IDENT decl\_

decl\_ -> var\_decl\_ | ( params ) compound\_stmt

var\_decl\_ →; | [ ] ;

type\_spec → VOID | BOOL | INT | FLOAT

params → param \_param\_list | VOID

\_param\_list → , param \_param\_list | ε

param → type\_spec IDENT param\_

param\_ → ε | []

stmt\_list → stmt stmt\_list | ε

stmt → expr\_stmt | compound\_stmt | if\_stmt | while\_stmt | return\_stmt | break\_stmt

expr\_stmt → expr ; | ;

while\_stmt → WHILE ( expr ) stmt

compound\_stmt → { local\_decls stmt\_list }

var\_decl → type\_spec IDENT var\_decl\_

local\_decls → var\_decl local\_decls | ε

if\_stmt → IF ( expr ) stmt if\_stmt\_

if\_stmt\_ → ε | ELSE stmt

return\_stmt → RETURN return\_stmt\_

return\_stmt\_ → ; | expr ;

expr → IDENT expr\_IDENT \_expr | ! expr \_expr | - expr \_expr | + expr \_expr | ( expr ) \_expr | BOOL\_LIT \_expr | INT\_LIT \_expr | FLOAT\_LIT \_expr | NEW type\_spec [ expr ] \_expr

\_expr → expr\_expr \_expr | ε

expr\_IDENT → = expr | [ expr ] = expr | [ expr ] | ( args ) | . size | ε

expr\_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | \* expr | / expr | % expr

arg\_list →expr \_arg\_list

\_arg\_list →, expr \_arg\_list| ε

args → arg\_list | ε