Assignment 2 Syntactic Analyzer

Yan Ren 40212201

Section 1 Transformed grammar into LL(1)

From the given COMP442.grammar.BNF.grm, first use grammartool.jar to remove left-recursion, then perform following transformation

1.

<arraySize> ::= '[' 'intNum' ']'
<arraySize> ::= '[' ']'
Replaced by:
<arraySize> ::= '[' <arraySize1>
<arraySize1> ::= 'intNum' ']'
<arraySize1> ::= ']'

2.

<expr> ::= <arithExpr>
<expr> ::= <relExpr>
Replaced by:
<expr> ::= <arithExpr> <expr1>
<expr1> ::= EPSILON
<expr1> ::= <relOp> <arithExpr>

3.

Delete
<idnest> ::= 'id' <rept-idnest1> '.'
<idnest> ::= 'id' '(' <aParams> ')' '.'
By place right hand side to where <idnest> is used

<variable> ::= <rept-variable0> 'id' <rept-variable2>
<rept-variable0> ::= 'id' <rept-idnest1> <rept-variable0>
<rept-variable0> ::= 'id' '(' <aParams> ')' <rept-variable0>
<rept-variable0> ::= EPSILON
<rept-variable2> ::= <indice> <rept-variable2>
<rept-variable2> ::= EPSILON
<indice> ::= '[' <arithExpr> ']'
Rewrite as
<variable> ::= 'id' <variableIdnest>
<variableIdnest> ::= <rept-idnest1> <variableIdnestTail0>
<variableIdnest> ::= '(' <aParams> ')' <variableIdnestTail1>
<variableIdnestTail0> ::= '.' 'id' <variableIdnest>
<variableIdnestTail0> ::= EPSILON
<variableIdnestTail1> ::= '.' 'id' <variableIdnest>

4.

<factor> ::= <variable>
<factor> ::= <functionCall>
Replaced with
<factor> ::= <funcOrVar>
<funcOrVar> ::= 'id' <funcOrVarIdnest>
<funcOrVarIdnest> ::= <rept-idnest1> <funcOrVarIdnestTail>
<funcOrVarIdnest> ::= '(' <aParams> ')' <funcOrVarIdnestTail>
<funcOrVarIdnestTail> ::= '.' 'id' <funcOrVarIdnest>
<funcOrVarIdnestTail> ::= EPSILON
5.
<statement> ::= <assignStat> ';'
<statement> ::= <functionCall> ';'
<assignStat> ::= <variable> <assignOp> <expr>
Replaced with
<statement> ::= 'id' <s1>
<s1> ::= <rept-idnest1> <s2>
<s1> ::= '(' <aParams> ')' <s3>
<s2> ::= '.' 'id' <s1>
<s2> ::= <assignOp> <expr> ';'
<s3> ::= '.' 'id' <s1>
<s3> ::= ';'

LL(1) Grammar

<START> ::= <prog>

<aParams> ::= <expr> <rept-aParams1>
<aParams> ::= EPSILON

<aParamsTail> ::= ',' <expr>

<addOp> ::= '+'
<addOp> ::= '-'
<addOp> ::= 'or'

<arithExpr> ::= <term> <rightrec-arithExpr>

<arraySize> ::= '[' <arraySize1>
<arraySize1> ::= 'intLit' ']'
<arraySize1> ::= ']'

<assignOp> ::= '='

<assignStat> ::= <variable> <assignOp> <expr>

<expr> ::= <arithExpr> <expr1>
<expr1> ::= EPSILON
<expr1> ::= <relOp> <arithExpr>

<fParams> ::= 'id' ':' <type> <rept-fParams3> <rept-fParams4>
<fParams> ::= EPSILON

<fParamsTail> ::= ',' 'id' ':' <type> <rept-fParamsTail4>

<factor> ::= <funcOrVar>
<factor> ::= 'intLit'
<factor> ::= 'floatLit'
<factor> ::= '(' <arithExpr> ')'
<factor> ::= 'not' <factor>
<factor> ::= <sign> <factor>

<funcOrVar> ::= 'id' <funcOrVarIdnest>
<funcOrVarIdnest> ::= <rept-idnest1> <funcOrVarIdnestTail>
<funcOrVarIdnest> ::= '(' <aParams> ')' <funcOrVarIdnestTail>
<funcOrVarIdnestTail> ::= '.' 'id' <funcOrVarIdnest>
<funcOrVarIdnestTail> ::= EPSILON

<funcBody> ::= '{' <rept-funcBody1> '}'

<funcDecl> ::= <funcHead> ';'

<funcDef> ::= <funcHead> <funcBody>

<funcHead> ::= 'func' 'id' '(' <fParams> ')' 'arrow' <returnType>

<implDef> ::= 'impl' 'id' '{' <rept-implDef3> '}'

<indice> ::= '[' <arithExpr> ']'

<memberDecl> ::= <funcDecl>
<memberDecl> ::= <varDecl>

<multOp> ::= '*'
<multOp> ::= '/'
<multOp> ::= 'and'

<opt-structDecl2> ::= 'inherits' 'id' <rept-opt-structDecl22>
<opt-structDecl2> ::= EPSILON

<prog> ::= <rept-prog0>

<relExpr> ::= <arithExpr> <relOp> <arithExpr>

<relOp> ::= 'eq'
<relOp> ::= 'neq'
<relOp> ::= 'lt'
<relOp> ::= 'gt'
<relOp> ::= 'leq'
<relOp> ::= 'geq'

<rept-aParams1> ::= <aParamsTail> <rept-aParams1>
<rept-aParams1> ::= EPSILON

<rept-fParams3> ::= <arraySize> <rept-fParams3>
<rept-fParams3> ::= EPSILON

<rept-fParams4> ::= <fParamsTail> <rept-fParams4>
<rept-fParams4> ::= EPSILON

<rept-fParamsTail4> ::= <arraySize> <rept-fParamsTail4>
<rept-fParamsTail4> ::= EPSILON

<rept-funcBody1> ::= <varDeclOrStat> <rept-funcBody1>
<rept-funcBody1> ::= EPSILON

<rept-idnest1> ::= <indice> <rept-idnest1>
<rept-idnest1> ::= EPSILON

<rept-implDef3> ::= <funcDef> <rept-implDef3>
<rept-implDef3> ::= EPSILON

<rept-opt-structDecl22> ::= ',' 'id' <rept-opt-structDecl22>
<rept-opt-structDecl22> ::= EPSILON

<rept-prog0> ::= <structOrImplOrfunc> <rept-prog0>
<rept-prog0> ::= EPSILON

<rept-statBlock1> ::= <statement> <rept-statBlock1>
<rept-statBlock1> ::= EPSILON

<rept-structDecl4> ::= <visibility> <memberDecl> <rept-structDecl4>
<rept-structDecl4> ::= EPSILON

<rept-varDecl4> ::= <arraySize> <rept-varDecl4>
<rept-varDecl4> ::= EPSILON

<returnType> ::= <type>
<returnType> ::= 'void'

<rightrec-arithExpr> ::= EPSILON
<rightrec-arithExpr> ::= <addOp> <term> <rightrec-arithExpr>

<rightrec-term> ::= EPSILON
<rightrec-term> ::= <multOp> <factor> <rightrec-term>

<sign> ::= '+'
<sign> ::= '-'

<statBlock> ::= '{' <rept-statBlock1> '}'
<statBlock> ::= <statement>
<statBlock> ::= EPSILON

<statement> ::= 'id' <s1>
<s1> ::= <rept-idnest1> <s2>
<s1> ::= '(' <aParams> ')' <s3>
<s2> ::= '.' 'id' <s1>
<s2> ::= <assignOp> <expr> ';'
<s3> ::= '.' 'id' <s1>
<s3> ::= ';'
<statement> ::= 'if' '(' <relExpr> ')' 'then' <statBlock> 'else' <statBlock> ';'
<statement> ::= 'while' '(' <relExpr> ')' <statBlock> ';'
<statement> ::= 'read' '(' <variable> ')' ';'
<statement> ::= 'write' '(' <expr> ')' ';'
<statement> ::= 'return' '(' <expr> ')' ';'

<structDecl> ::= 'struct' 'id' <opt-structDecl2> '{' <rept-structDecl4> '}' ';'

<structOrImplOrFunc> ::= <structDecl>
<structOrImplOrFunc> ::= <implDef>
<structOrImplOrFunc> ::= <funcDef>

<term> ::= <factor> <rightrec-term>

<type> ::= 'integer'
<type> ::= 'float'
<type> ::= 'id'

<variable> ::= 'id' <variableIdnest>
<variableIdnest> ::= <rept-idnest1> <variableIdnestTail>
<variableIdnest> ::= '(' <aParams> ')' '.' 'id' <variableIdnest>
<variableIdnestTail> ::= '.' 'id' <variableIdnest>
<variableIdnestTail> ::= EPSILON

<varDecl> ::= 'let' 'id' ':' <type> <rept-varDecl4> ';'

<varDeclOrStat> ::= <varDecl>
<varDeclOrStat> ::= <statement>

<visibility> ::= 'public'
<visibility> ::= 'private'

Section 2

nonterminal	first set	follow set	nullable	endable
ADDOP	plus minus or	intlit floatlit lpar not id plus minus	no	yes
ARRAYSIZE1	intlit rsqbr	semi lsqbr rpar comma	no	no
ASSIGNSTAT	id	∅	no	no
EXPR1	eq neq lt gt leq geq	comma semi rpar	yes	no
FUNCBODY	lcurbr	struct impl func rcurbr	no	no
FUNCHEAD	func	semi lcurbr	no	no
FPARAMS	id	rpar	yes	no
FUNCORVAR	id	mult div and rsqbr eq neq lt gt leq geq plus minus or comma semi rpar	no	no
FUNCORVARIDNESTTAIL	dot	mult div and rsqbr eq neq lt gt leq geq plus minus or comma semi rpar	yes	no
FUNCORVARI	lpar dot lsqbr	mult div and rsqbr eq neq lt gt leq geq plus	yes	no

DNEST		minus or comma semi rpar		
FUNCDECL	func	rcurbr public private	no	no
ARITHEXPR	intlit floatlit lpar not id plus minus	rsqbr eq neq lt gt leq geq comma semi rpar	no	no
RELOP	eq neq lt gt leq geq	intlit floatlit lpar not id plus minus	no	no
APARAMSTAIL	comma	comma rpar	no	no
REPTAPARAMS1	comma	rpar	yes	no
REPTFPARAMS3	lsqbr	rpar comma	yes	no
FPARAMSTAIL	comma	comma rpar	no	no
REPTFPARAMS4	comma	rpar	yes	no
REPTFPARAMSTAIL4	lsqbr	comma rpar	yes	no
REPTFUNCBODY1	let id if while read write return	rcurbr	yes	no
INDICE	lsqbr	mult div and lsqbr dot rsqbr eq neq lt gt leq geq equal plus minus or comma semi rpar	no	no
REPTIMPLDEF3	func	rcurbr	yes	no
REPTOPTSTRUCTDECL22	comma	lcurbr	yes	no
REPTPROG0	struct impl func	∅	yes	no
MEMBERDECL	let func	rcurbr public private	no	no
ARRAYSIZE	lsqbr	semi lsqbr rpar comma	no	no

RETURNTYPE	void integer float id	semi lcurbr	no	no
RIGHTRECARITHEXPR	plus minus or	rsqbr eq neq lt gt leq geq comma semi rpar	yes	no
MULTOP	mult div and	intlit floatlit lpar not id plus minus	no	no
S2	dot equal	else semi let id if while read write return rcurbr	no	no
ASSIGNOP	equal	intlit floatlit lpar not id plus minus	no	no
S3	dot semi	else semi let id if while read write return rcurbr	no	no
SIGN	plus minus	intlit floatlit lpar not id plus minus	no	no
START	struct impl func	∅	yes	no
PROG	struct impl func	∅	yes	no
REPTSTATBLOCK1	id if while read write return	rcurbr	yes	no
S1	lpar dot lsqbr equal	else semi let id if while read write return rcurbr	no	no
RELEXPR	intlit floatlit lpar not id plus minus	rpar	no	no
STATBLOCK	lcurbr id if while read write return	else semi	yes	no
EXPR	intlit floatlit lpar not id plus minus	comma semi rpar	no	no
OPTSTRUCTDECL2	inherits	lcurbr	yes	no
REPTSTRUCTDECL4	public private	rcurbr	yes	no
STRUCTORIMPLORFUNC	struct impl func	struct impl func	no	no
STRUCTDEC	struct	struct impl func	no	no

L				
IMPLDEF	impl	struct impl func	no	no
FUNCDEF	func	struct impl func rcurbr	no	no
TERM	intlit floatlit lpar not id plus minus	rsqbr eq neq lt gt leq geq plus minus or comma semi rpar	no	no
FACTOR	intlit floatlit lpar not id plus minus	mult div and rsqbr eq neq lt gt leq geq plus minus or comma semi rpar	no	no
RIGHTRECTERM	mult div and	rsqbr eq neq lt gt leq geq plus minus or comma semi rpar	yes	no
TYPE	integer float id	rpar lcurbr comma lsqbr semi	no	no
REPTVARDECL4	lsqbr	semi	yes	no
VARDECLORSTAT	let id if while read write return	let id if while read write return rcurbr	no	no
VARDECL	let	public private let id if while read write return rcurbr	no	no
STATEMENT	id if while read write return	else semi let id if while read write return rcurbr	no	no
VARIABLE	id	equal rpar	no	no
REPTIDNEST1	lsqbr	mult div and dot rsqbr eq neq lt gt leq geq equal plus minus or comma semi rpar	yes	no
APARAMS	intlit floatlit lpar not id plus minus	rpar	yes	no
VARIABLEIDNESTTAIL	dot	equal rpar	yes	no
VARIABLEIDNEST	lpar dot lsqbr	equal rpar	yes	no
VISIBILITY	public private	let func	no	no

Section 3 Design

Parser.java is implemented as table driven predictive parsing. parse() and skipErrors() functions are implemented by strictly following the algorithm provided in the lecture. Parser table and first/follow set is generated by the ucalgary tool. I downloaded the HTML page from ucalgary tool and parsed HTML content to generate the parsing table and first/follow set as Java data structure, which is used in the parse function.

Section 4 Use of Tool

- Ucalgary tool: validate the LL(1) grammar; generate parsing table; generate first/follow set
- org.jsoup.Jsoup: Java library for parsing HTML

Assignment Submission

/src - contains the source code
/input - contains the test files
/output - contains the output files
/parserdriver.jar - executable

Jar executable is built by Maven using Java11, which requires Java Runtime Environment 11+
If you wish to rebuild the jar, use the pom.xml. Otherwise, you can compile and run the program from source code.

How to run jar executable

Run with default .src files in /input folder

```
$ java -jar parserdriver.jar ./input/bubblesort.src
```

.outsyntaxerrors .outderivation files will be available in ./output