Report for Assignment 2

Jonathan Jong 40133041

Report Assignment 2 COMP 442

Section 1:

The following ambiguities were found using the University of Calgary tool:

The grammar is not LL(1) because:

- ARRAYSIZE has a first set conflict.
- EXPR has a first set conflict.
- FACTOR has a first set conflict.
- FUNCHEAD has a first set conflict.
- IDNEST has a first set conflict.
- LOCALVARDECL has a first set conflict.
- OPTFUNCHEAD1 is nullable with clashing first and follow sets.
- REPTFUNCTIONCALL0 is nullable with clashing first and follow sets.
- REPTVARIABLE0 is nullable with clashing first and follow sets.
- STATEMENT has a first set conflict.

The LL(1) grammar is as follows:

START -> REPTSTART0 .

APARAMS -> EXPR REPTAPARAMS1 .
APARAMS -> .

APARAMSTAIL -> comma EXPR .

ADDOP -> plus .
ADDOP -> minus .
ADDOP -> or .

ARITHEXPR -> TERM RIGHTRECARITHEXPR .

ARRAYSIZE -> lsqbr ARRAYSIZE1 .
ARRAYSIZE1 -> intlit rsqbr .
ARRAYSIZE1 -> rsqbr .

ASSIGNOP -> equal .

CLASSDECL -> class id OPTCLASSDECL2 lcurbr REPTCLASSDECL4 rcurbr semi .

CLASSDECLORFUNCDEF -> CLASSDECL .
CLASSDECLORFUNCDEF -> FUNCDEF .

EXPR -> ARITHEXPR EXPR1 .

EXPR1 -> RELOP ARITHEXPR .
EXPR1 -> .

FPARAMS -> id colon TYPE REPTFPARAMS3 REPTFPARAMS4 .
FPARAMS -> .

FPARAMSTAIL -> comma id colon TYPE REPTFPARAMSTAIL4 .

FACTOR -> FUNCTIONCALLORVARIABLE .
FACTOR -> intlit .
FACTOR -> floatlit .
FACTOR -> lpar ARITHEXPR rpar .
FACTOR -> not FACTOR .
FACTOR -> SIGN FACTOR .

FUNCBODY -> lcurbr REPTFUNCBODY1 rcurbr .

FUNCDEF -> FUNCHEAD FUNCBODY .

FUNCHEAD -> function id FUNCHEAD3 .

FUNCHEAD1 -> id lpar FPARAMS rpar arrow RETURNTYPE .
FUNCHEAD1 -> constructor lpar FPARAMS rpar .

FUNCHEAD3 -> sr FUNCHEAD1 .
FUNCHEAD3 -> lpar FPARAMS rpar arrow RETURNTYPE .

ASSIGNSTAT -> VARIABLE ASSIGNOP EXPR .

FUNCTIONCALL -> id FUNCALL3 .
FUNCALL3 -> lpar APARAMS rpar .
FUNCALL3 -> FUNCALL2 .
FUNCALL2 -> dot id FUNCALL4 .
FUNCALL4 -> INDICE FUNCALL2 .
FUNCALL4 -> lpar APARAMS rpar FUNCALL5 .
FUNCALL5 -> FUNCALL2 .
FUNCALL5 -> .

VARIABLE -> id VARIABLE3 .
VARIABLE3 -> INDICE .
VARIABLE3 -> VARIABLE2 .
VARIABLE3 -> .
VARIABLE2 -> dot id VARIABLE4 .
VARIABLE4 -> lpar APARAMS rpar VARIABLE2 .
VARIABLE4 -> INDICE VARIABLE5 .
VARIABLE5 -> VARIABLE2 .
VARIABLE5 -> .

FUNCTIONCALLORVARIABLE -> id FUNCTIONCALLORVARIABLE1 .
FUNCTIONCALLORVARIABLE1 -> INDICELOOP FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE1 -> lpar APARAMS rpar FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE2 -> dot id FUNCTIONCALLORVARIABLE3 .
FUNCTIONCALLORVARIABLE2 -> .
FUNCTIONCALLORVARIABLE3 -> INDICELOOP FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE3 -> lpar APARAMS rpar FUNCTIONCALLORVARIABLE2 .

IDNEST1 -> dot id IDNEST2 .
IDNEST2 -> lsqbr ARITHEXPR rsqbr IDNEST2 .
IDNEST2 -> lpar APARAMS rpar .
IDNEST2 -> .

INDICE -> lsqbr ARITHEXPR rsqbr .

LOCALVARDECL -> localvar id colon TYPE LOCALVARDECL2 .
LOCALVARDECL2 -> REPTLOCALVARDECL4 semi .
LOCALVARDECL2 -> lpar APARAMS rpar semi .

LOCALVARDECLORSTMT -> LOCALVARDECL .
LOCALVARDECLORSTMT -> STATEMENT .

MEMBERDECL -> MEMBERFUNCDECL .
MEMBERDECL -> MEMBERVARDECL .

MEMBERFUNCDECL -> function id colon lpar FPARAMS rpar arrow RETURNTYPE semi .
MEMBERFUNCDECL -> constructor colon lpar FPARAMS rpar semi .

MEMBERVARDECL -> attribute id colon TYPE REPTMEMBERVARDECL4 semi .

MULTOP -> mult .
MULTOP -> div .
MULTOP -> and .

OPTCLASSDECL2 -> isa id REPTOPTCLASSDECL22 .
OPTCLASSDECL2 -> .

RELEXPR -> ARITHEXPR RELOP ARITHEXPR .

RELOP -> eq .
RELOP -> neq .
RELOP -> lt .
RELOP -> gt .
RELOP -> leq .
RELOP -> geq .

REPTSTART0 -> CLASSDECLORFUNCDEF REPTSTART0 .
REPTSTART0 -> .

REPTAPARAMS1 -> APARAMSTAIL REPTAPARAMS1 .
REPTAPARAMS1 -> .

REPTCLASSDECL4 -> VISIBILITY MEMBERDECL REPTCLASSDECL4 .
REPTCLASSDECL4 -> .

REPTFPARAMS3 -> ARRAYSIZE REPTFPARAMS3 .
REPTFPARAMS3 -> .

REPTFPARAMS4 -> FPARAMSTAIL REPTFPARAMS4 .
REPTFPARAMS4 -> .

REPTFPARAMSTAIL4 -> ARRAYSIZE REPTFPARAMSTAIL4 .
REPTFPARAMSTAIL4 -> .

REPTFUNCBODY1 -> LOCALVARDECLORSTMT REPTFUNCBODY1 .
REPTFUNCBODY1 -> .

REPTLOCALVARDECL4 -> ARRAYSIZE REPTLOCALVARDECL4 .
REPTLOCALVARDECL4 -> .

REPTMEMBERVARDECL4 -> ARRAYSIZE REPTMEMBERVARDECL4 .
REPTMEMBERVARDECL4 -> .

REPTOPTCLASSDECL22 -> comma id REPTOPTCLASSDECL22 .
REPTOPTCLASSDECL22 -> .

REPTSTATBLOCK1 -> STATEMENT REPTSTATBLOCK1 .
REPTSTATBLOCK1 -> .

RETURNTYPE -> TYPE .
RETURNTYPE -> void .

RIGHTRECARITHEXPR -> .
RIGHTRECARITHEXPR -> ADDOP TERM RIGHTRECARITHEXPR .

RIGHTRECTERM -> .
RIGHTRECTERM -> MULTOP FACTOR RIGHTRECTERM .

SIGN -> plus .
SIGN -> minus .

STATBLOCK -> lcurbr REPTSTATBLOCK1 rcurbr .
STATBLOCK -> STATEMENT .
STATBLOCK -> .

STATEMENT -> FUNCTIONCALLORASIGNSTAT semi .
STATEMENT -> if lpar RELEXPR rpar then STATBLOCK else STATBLOCK semi .
STATEMENT -> while lpar RELEXPR rpar STATBLOCK semi .
STATEMENT -> read lpar VARIABLE rpar semi .
STATEMENT -> write lpar EXPR rpar semi .
STATEMENT -> return lpar EXPR rpar semi .

FUNCTIONCALLORASIGNSTAT -> id ISFUNCTIONCALLORVARIABLE .

ISFUNCTIONCALLORVARIABLE -> lpar APARAMS rpar AFTERFUNCTIONCALL .
ISFUNCTIONCALLORVARIABLE -> INDICELOOP AFTERVARIABLE .

AFTERFUNCTIONCALL -> dot id MIDDLESTATE .
AFTERVARIABLE -> dot id MIDDLESTATE .

MIDDLESTATE -> INDICELOOP AFTERVARIABLE .
MIDDLESTATE -> lpar APARAMS rpar AFTERFUNCTIONCALL .

AFTERVARIABLE -> ENDASSIGN .
AFTERFUNCTIONCALL -> .

INDICELOOP -> INDICE INDICELOOP .
INDICELOOP -> .
ENDASSIGN -> ASSIGNOP EXPR .

TERM -> FACTOR RIGHTRECTERM .

TYPE -> integer .
TYPE -> float .
TYPE -> id .

VISIBILITY -> public .
VISIBILITY -> private .
VISIBILITY -> .

Section 2:
The following table was generated from the University of Calgary tool, showing the first and follow sets for each non-terminal.

nonterminal	first set	follow set	nullable	endable
ADDOP	plus minus or	intlit floatlit lpar not id plus minus	no	yes
ARRAYSIZE1	intlit rsqbr	lsqbr semi rpar comma	no	no
ASSIGNOP	equal	intlit floatlit lpar not id plus minus	no	no
CLASSDECL	class	class function	no	no
EXPR1	eq neq lt gt leq geq	semi comma rpar	yes	no
FUNCDEF	function	class function	no	no
FUNCBODY	lcurbr	class function	no	no
FUNCHEAD	function	lcurbr	no	no
APARAMS	intlit floatlit lpar not id plus minus	rpar	yes	no
LOCALVARDECL	localvar	localvar if while read write return id rcurbr	no	no
MEMBERFUNCDECL	function constructor	public private function constructor attribute rcurbr	no	no
FPARAMS	id	rpar	yes	no
MEMBERVARDECL	attribute	public private function constructor attribute rcurbr	no	no
OPTCLASSDECL2	isa	lcurbr	yes	no
OPTFUNCHEAD1	id	id	yes	no
ARITHEXPR	intlit floatlit lpar not id plus minus	semi rsqbr eq neq lt gt leq geq comma 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
MEMBERDECL	function constructor attribute	public private function constructor attribute rcurbr	no	no
REPTCLASSDECL4	public private function constructor attribute	rcurbr	yes	no

REPTFPARAMS3	lsqbr	rpar comma	yes	no
FPARAMSTAIL	comma	comma rpar	no	no
REPTFPARAMS4	comma	rpar	yes	no
REPTFPARAMSTAIL 4	lsqbr	comma rpar	yes	no
LOCALVARDECLORSTMT	localvar if while read write return id	localvar if while read write return id rcurbr	no	no
REPTFUNCBODY1	localvar if while read write return id	rcurbr	yes	no
REPTFUNCTIONCALL0	id	id	yes	no
REPTIDNEST1	lsqbr	dot	yes	no
REPTLOCALVARDECL4	lsqbr	semi	yes	no
ARRAYSIZE	lsqbr	lsqbr semi rpar comma	no	no
REPTMEMBERVARDECL4	lsqbr	semi	yes	no
REPTOPTCLASSDECL22	comma	lcurbr	yes	no
CLASSDECLORFUNCDEF	class function	class function	no	no
IDNEST	id	id	no	no
INDICE	lsqbr	semi mult div and lsqbr dot rsqbr eq neq lt gt leq geq equal plus minus or comma rpar	no	no
RETURNTYPE	void integer float id	semi lcurbr	no	no
RIGHTRECARITHEXPR	plus minus or	semi rsqbr eq neq lt gt leq geq comma rpar	yes	no
MULTOP	mult div and	intlit floatlit lpar not id plus minus	no	no
SIGN	plus minus	intlit floatlit lpar not id plus minus	no	no
START	class function	ε	yes	no
REPTSTART0	class function	ε	yes	no
REPTSTATBLOCK1	if while read write return id	rcurbr	yes	no
STATEMENT	if while read write return id	else semi localvar if while read write return id rcurbr	no	no
ASSIGNSTAT	id	semi	no	no

RELEXPR	intlit floatlit lpar not id plus minus	rpar	no	no
STATBLOCK	lcurbr if while read write return id	else semi	yes	no
EXPR	intlit floatlit lpar not id plus minus	semi comma rpar	no	no
FUNCTIONCALL	id	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	no	no
TERM	intlit floatlit lpar not id plus minus	semi rsqbr eq neq lt gt leq geq plus minus or comma rpar	no	no
FACTOR	intlit floatlit lpar not id plus minus	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	no	no
RIGHTRECTERM	mult div and	semi rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes	no
TYPE	integer float id	rpar lcurbr comma lpar lsqbr semi	no	no
VARIABLE	id	semi mult div and rsqbr eq neq lt gt leq geq equal plus minus or comma rpar	no	no
REPTVARIABLE0	id	id	yes	no
REPTVARIABLE2	lsqbr	semi mult div and rsqbr eq neq lt gt leq geq equal plus minus or comma rpar	yes	no
VISIBILITY	public private	function constructor attribute	yes	no

Section 3:

The main structure involves the parse method in the Parser.java method. Here, we use the algorithm for a table driven parser that was generated by the University of Calgary for the rules that the parser will follow. The code followed the parsing algorithm that we have seen in class:

```
parse(){
    push($)
    push(S)
    a = nextToken()
    while ( stack ≠ $ ) do
        x = top()
        if ( x ∈ T )
            if  ( x == a )
                pop(x) ; a = nextToken()
            else
                skipError() ; success = false
        else
            if ( TT[x,a] ≠ 'error' )
                pop(x) ; inverseRHSPush(TT[x,a])
            else
                skipError() ; success = false
    if ( (a ≠ $) ∨ (success == false ) )
        return(false)
    else
        return(true)}
```

The main parts were split between the if and else statement following the terminal and non-terminal logic. If the top of the stack is a terminal and equal to a token, then we found a match and pop. The main else statement demonstrates if the top of the stack and token is not an error, it is a non-terminal so we pop and use the parsing table to replace using the right hand side.

Skiperrors logic was also implemented from the lecture slides:

```
skipError(){                              // A is top()
  write ("syntax error at " lookahead.location)
  if ( lookahead is $ or in FOLLOW( top() ) )
     pop()                                // pop - equivalent to A → ε
  else
     while ( lookahead ∉ FIRST( top() )
             or
             ε ∈ FIRST( top() ) and lookahead ∉ FOLLOW( top() )
           )
        lookahead = nextToken()           // scan
}
```

Section 4:
The main tool that was used was the University of Calgary tool. This tool allowed for finding the ambiguities and listing them out as well as generating the parsing table and that was the main reason it was used. Something the tool lacked was the ability to say if the correction to solve an ambiguity was good. This is where atocc came in as it checked for 2 requirements and if they were fulfilled to show if the correction was implemented properly but mainly the University of Calgary tool was used.

Once the grammar no longer had ambiguities in it, the University of Calgary tool was able to generate the first and follow sets in a table as well as the parsing table. Solving the ambiguities involved factorization and rearranging of the expressions. To get this into a format we could use in the code, an HTML to CSV tool (convertcsv.com) was used so that we could delimit the rows and columns and obtain the data for the parser in the code.