COMP 6421 the Report of Assigment2

Syntactic Analyzer

Liao Xiaoyun

40102049 sharon.liaoxy@gmail.com

Analysis

1. Left recursions

1) <arithExpr> ::= <arithExpr> <addOp> <term> | <term>

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

2) <term> ::= <term> <multOp> <factor> | <factor>

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

2. Ambiguity

RHS' first sets have comment elements

1) <statement>
<statement> ::= <assignStat> ';' | <functionCall> ';'
<assignStat> ::= <variable> <assignOp> <expr>
<variable> ::= {{<idnest>}} 'id' {{<indice>}}
<functionCall> ::= {{<idnest>}} 'id' '(' <aParams> ')'

<statement> ::= <FuncOrAssignStat>
<FuncOrAssignStat> ::= 'id' <FuncOrAssignStatIdnest>
<FuncOrAssignStatIdnest> ::= <IndiceRep> <FuncOrAssignStatIdnestVarTail>
<FuncOrAssignStatIdnest> ::= '(' <aParams> ')' <FuncOrAssignStatIdnestFuncTail>
<FuncOrAssignStatIdnestVarTail> ::= '.' 'id' <FuncOrAssignStatIdnest>
<FuncOrAssignStatIdnestVarTail> ::= <AssignStatTail>

2) <factor>
<factor> ::= <variable> | <functionCall>

<factor> ::= <FuncOrVar>
<FuncOrVar> ::= 'id' <FuncOrVarIdnest>
<FuncOrVarIdnest> ::= <IndiceRep> <FuncOrVarIdnestTail>
<FuncOrVarIdnest> ::= '(' <aParams> ')' <FuncOrVarIdnestTail>
<FuncOrVarIdnestTail> ::= '.' 'id' <FuncOrVarIdnest>
<FuncOrVarIdnestTail> ::= EPSILON

First set and flow set have common elements when first set contains Epsilon

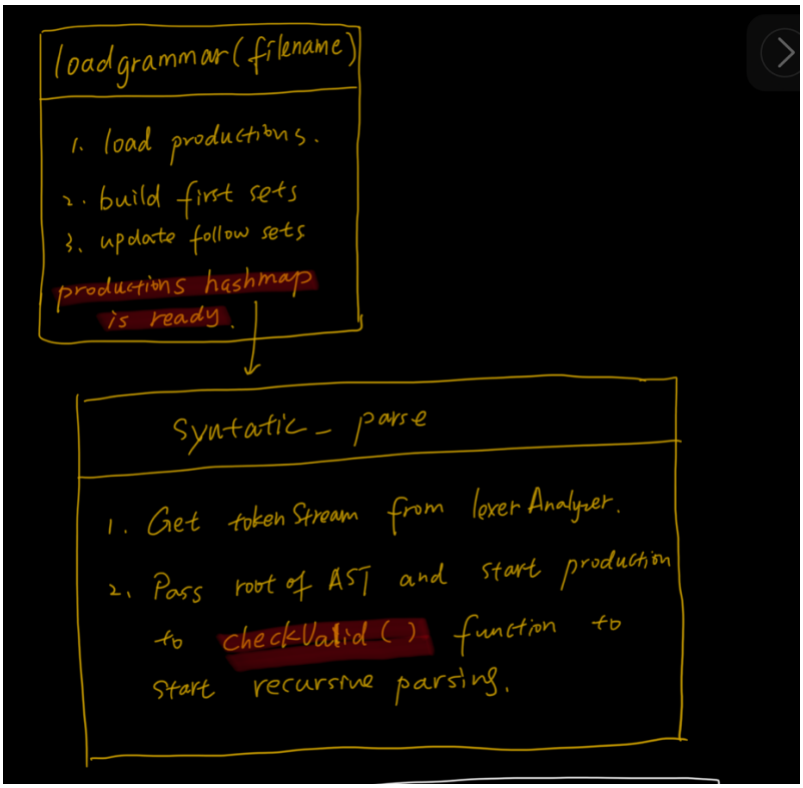
3) <funcHead>
<funcHead> ::= 'func' [['id' 'sr']] 'id' '(' <fParams> ')' ':' <type>
| 'func' [['id' 'sr']] 'id' '(' <fParams> ')' ':' 'void'

<funcHead> ::= 'func' 'id' <ClassMethod> '(' <fParams> ')' ':' <funcDeclTail>
<ClassMethod> ::= 'sr' 'id' | EPSILON

3. First set and follow set

nonterminal	first set	follow set	nullable	endable
ADDOP	plus minus or	intlit floatlit stringlit lpar not qm id plus minus	no	yes
APARAMSTAIL	comma	rpar	yes	no
ARITHEXPRTAIL	plus minus or	semi eq neq lt gt leq geq comma colon rsqbr rpar	yes	no
ASSIGNOP	equal	intlit floatlit stringlit lpar not qm id plus minus	no	no
CLASSDECLBODY	public private func integer float string id	rcurbr	yes	no
EXPRTAIL	eq neq lt gt leq geq	semi comma colon rsqbr rpar	yes	no
ARITHEXPR	intlit floatlit stringlit lpar not qm id plus minus	semi eq neq lt gt leq geq comma colon rsqbr rpar	no	no
FPARAM	integer float string id	rpar comma	no	no
FPARAMSTAIL	comma	rpar	yes	no
CLASSMETHOD	sr	lpar	yes	no
FPARAMS	integer float string id	rpar	yes	no
FUNCDECLTAIL	void integer float string id	lcurbr semi	no	no
FUNCORASSIGNSTATIDNESTFUNCTAIL	dot	semi	yes	no
FUNCORASSIGNSTATIDNESTVARTAIL	dot equal	semi	no	no
FUNCORASSIGNSTATIDNEST	lpar lsqbr dot equal	semi	no	no
ASSIGNSTATTAIL	equal	semi	no	no
FUNCORVAR	id	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	no	no
FUNCORVARIDNESTTAIL	dot	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	yes	no
FUNCORVARIDNEST	lpar lsqbr dot	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	yes	no
APARAMS	intlit floatlit stringlit lpar not qm id plus minus	rpar	yes	no
FUNCSTATTAILIDNEST	dot	semi	yes	no
FUNCSTATTAIL	dot lpar lsqbr	semi	no	no
FUNCTION	func	main func	no	no
FUNCHEAD	func	lcurbr	no	no
INHERIT	inherits	lcurbr	yes	no
INTNUM	intlit	rsqbr	yes	no
MEMBERDECL	func integer float string id	public private func integer float string id rcurbr	no	no
FUNCDECL	func	public private func integer float string id rcurbr	no	no
METHODBODYVAR	var	if while read write return break continue id rcurbr	yes	no
NESTEDID	comma	lcurbr	yes	no
CLASSDECL	class	func main	yes	no
FUNCDEF	func	main	yes	no
FUNCBODY	lcurbr	main func	no	no
RELOP	eq neq lt gt leq geq	intlit floatlit stringlit lpar not qm id plus minus	no	no
SIGN	plus minus	intlit floatlit stringlit lpar not qm id plus minus	no	no
START	main class func	∅	no	no
PROG	main class func	∅	no	no
FUNCORASSIGNSTAT	id	semi	no	no
STATBLOCK	lcurbr if while read write return break continue ic	else semi	yes	no
EXPR	intlit floatlit stringlit lpar not qm id plus minus	semi comma colon rsqbr rpar	no	no
STATEMENT	if while read write return break continue id	if while read write return break continue id else semi rcurbr	no	no
STATEMENTLIST	if while read write return break continue id	rcurbr	yes	no
TERM	intlit floatlit stringlit lpar not qm id plus minus	semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	no	no
MULTOP	mult div and	intlit floatlit stringlit lpar not qm id plus minus	no	no
FACTOR	intlit floatlit stringlit lpar not qm id plus minus	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	no	no
TERMTAIL	mult div and	semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	yes	no
TYPE	integer float string id	lcurbr semi id	no	no
ARRAYSIZEREPT	lsqbr	rpar semi comma	yes	no
VARDECL	integer float string id	public private func integer float string id rcurbr	no	no
VARDECLREP	integer float string id	rcurbr	yes	no
VARIABLE	id	rpar	no	no
INDICEREP	lsqbr	mult div and semi equal dot eq neq lt gt leq geq plus minus or comma colon	yes	no
VARIABLEIDNESTTAIL	dot	rpar	yes	no
VARIABLEIDNEST	lsqbr dot	rpar	yes	no
VISIBILITY	public private	func integer float string id	yes	no

4. Design:



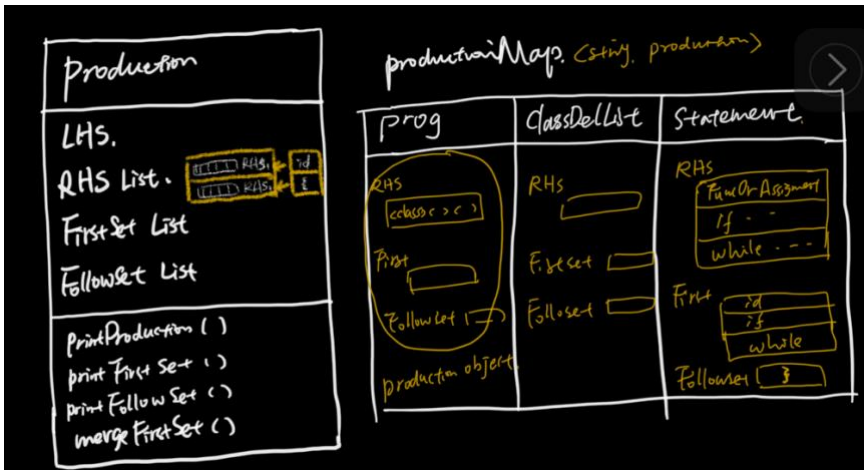
1) Loading productions from the grammar file.
   - Defining a production class.
   - Reading grammar from the grammar file line by line.
   - Each LHS has a production object, all of those productions will be stored in a hashmap productionMap <String,Production>.

```
class Production {

    public String LHS;
    public ArrayList<String[]> RHS = new ArrayList<>(); //can have multiple rhs
    public ArrayList<Set<String>> firstSet = new ArrayList<>();
    public Set<String> followSet = new HashSet<>();
    public boolean firstSetReady = false;

    public Set<String> mergeFirstSet(){
        Set<String> fullset = new HashSet<>();
        for (Set<String> fSet : firstSet) {
            fullset.addAll(fSet);
        }
        return fullset;
    }
```

2) Generating first sets for each RHS of all LHS.

I used a recursive function to generate the first sets.

Each RHS has its first set.




```java
public void buildFirstSet(){
    for(Production production : productionMap.values()) {
        System.out.println(production.printProduction());
        //check if first set of this element have bee generated,
        // if yes, just skip this element
        if(production.firstSetReady == true) {
            continue;
        }
        for(int i = 0; i < production.RHS.size(); i++){

            String firstElement = production.RHS.get(i)[0];
              System.out.println("findFirst: "+firstElement);
            if (!isNonterminal(firstElement)){
                // the first element is terminal
                // add terminal to first set
                // finish this RHS branch's first set search
                Set<String> firstset = new HashSet<>();
                firstset.add(firstElement);
                production.firstSet.add(firstset);
            } else {
                //the first element is nonterminal
                //calling findFirst() function to get this nonterminal's first set
                Set<String> firstSet =  findFirst(firstElement);
                production.firstSet.add(firstSet);
                String str = "";
                for (String substr : firstSet) {
                    str +=  ","+substr;
                }
                System.out.println("firstElement: "+ firstElement + " " + str);
                //if the first set of this nonterminal contains epsilon,
                // then continue to get next element's first set.
                if (firstSet.contains("epsilon")){
                    int rhsIndex = 1;
                    while (rhsIndex < production.RHS.get(i).length){
                        String nextElement = production.RHS.get(i)[rhsIndex];
                        System.out.println("firstSet has epsilon, get next element: "+nextElement);
                        firstSet =  findFirst(nextElement);
                        production.firstSet.get(i).addAll(firstSet);
                        if (!firstSet.contains("epsilon")) break;
                        rhsIndex += 1;
                    }
                    // check if the first set of last element contains epsilon
                    // if not, remove epsilon from the first set
                    if (rhsIndex != production.RHS.get(i).length){
                        // means this production can not be "epsilon"
                        production.firstSet.get(i).remove( o: "epsilon");
                    }
                }
            }
        }
        production.firstSetReady = true;
```

1. 若 (A∈T) ∨ (A is ε) then First(A) ⊇ {A}

2.1 First(A) ⊇ (First(Si) - {ε})

2.2 若 (ε ∈ First(S1) ... First(Si)) then First(A) ⊇ First(Si+1)

2.3 若 (ε ∈ First(S1) ... First(Sk)) then First(A) ⊇ {ε}.

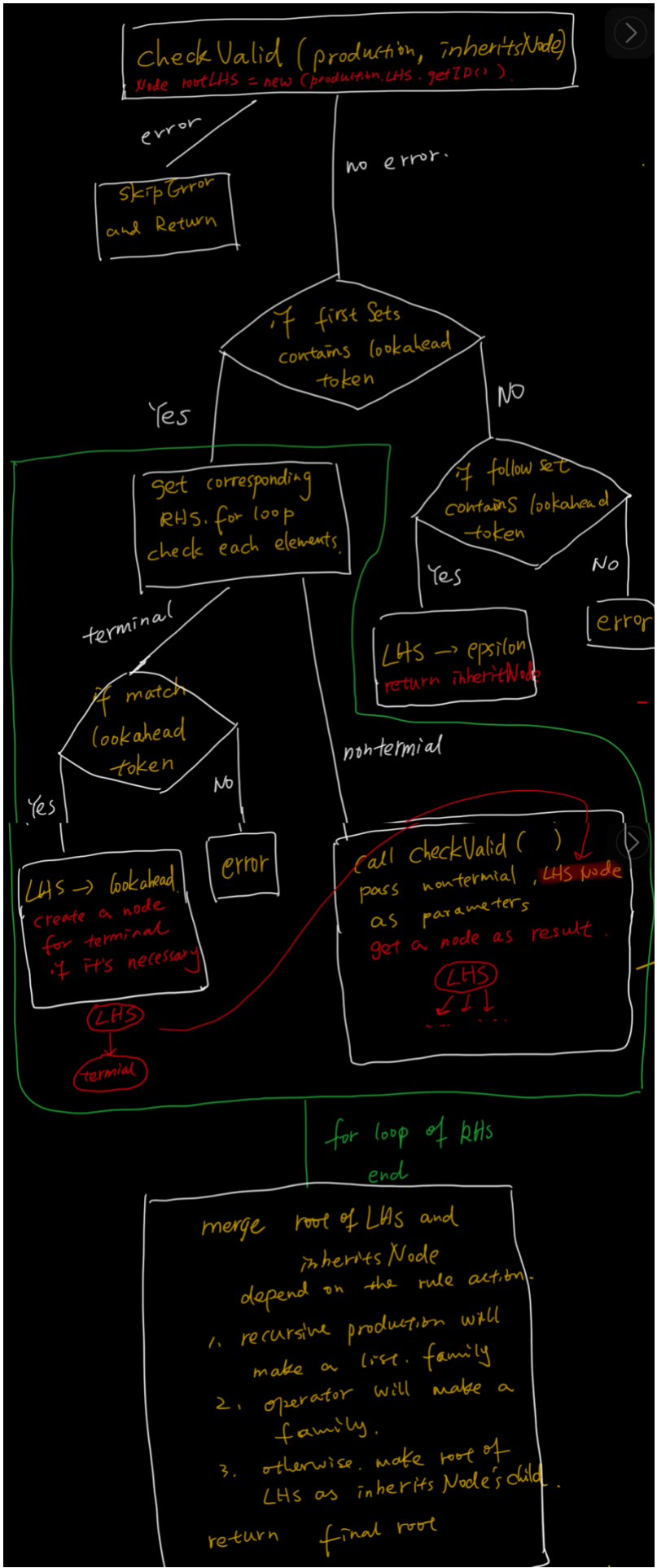
3) Loading follow set from files

   I used a tool to get follow set of the grammar, then updated the follow set of productions by loading the follow set file.

4) Parsing

   When productions, first sets and follow sets are ready, we can start to parse the src file.

   - Recursive descent predictive parsing
   - Generating AST
   - Error recovery


checkValid (production, inheritsNode)
Node rootLHS = new (production.LHS.getID());
error
skipError and Return
no error.
if first Sets contains lookahead token
Yes
NO
get corresponding RHS. for loop check each elements.
if follow set contains lookahead token
Yes
No
error
LHS → epsilon return inheritNode
terminal
if match lookahead token
Yes
No
error
nontermial
LHS → lookahead. create a node for terminal if it's necessary
LHS
termial
call checkValid ( ) pass nontermial, LHS node as parameters get a node as result.
LHS
for loop of RHS end
merge root of LHS and inherits Node depend on the rule action.
1. recursive production will make a list. family
2. operator will make a family.
3. otherwise. make root of LHS as inherits Node's child.
return final root

Generating AST

1. Current RHS is epsilon, return inherits Node.
2. Current RHS is recursive, make a list, for example,
   <ClassDeclBodyList> ::= <ClassDeclBody> <ClassDeclBodyList>
3. Current RHS is operator recursive, make a list, for example,
   <arithExprTail>::= <addOp> <term> <arithExprTail>
4. Flat function statement or assign statement element,
   flat function or var element.
5. Otherwise, current root merge to inherits Node' children list.
6. The last step, check if the current subtree contains operators, if yes, make an operator family.

```
public Node mergeNode(Production production, int rhsRow, Node inheritsNode, Node curNode){
    Node root ;
    if(curNode.name.compareTo("null") == 0){
        // current rhs is epsilon, return inheritsNode
        root = inheritsNode;
    }else if(ifMakeList(production,rhsRow,inheritsNode, curNode)){
        //MakeList, recursive rhs
        inheritsNode.childrenList.addAll(curNode.childrenList);
        root = inheritsNode;
    }else if (ifRecursiveOperator(production,rhsRow,inheritsNode, curNode)){
        //MakeList, recursive operator addOp, mulOp
        curNode.childrenList.remove( index: 0);
        inheritsNode.childrenList.addAll(curNode.childrenList);
        root = inheritsNode;
    }
    else if(flatFuncOrStatmList.contains(inheritsNode.type) && flatFuncOrStatmList.contains(curNode.type)){
        // flat function statement or assign statement element
        inheritsNode.childrenList.addAll(curNode.childrenList);
        root = inheritsNode;
    }else if(flatFuncVarList.contains(inheritsNode.type) && flatFuncVarList.contains(curNode.type)){
        // flat function or var element
        inheritsNode.childrenList.addAll(curNode.childrenList);
        root = inheritsNode;
    }
    else {
        //merge
        inheritsNode.addChild(curNode);
        root = inheritsNode;
    }
```

```
//Operator make Family
if(makeOperatorFamilyPrdctMap.get(root.type) != null &&
        makeOperatorFamilyPrdctMap.get(root.type).compareTo(root.childrenList.get(root.childrenList.size()-1).type) == 0){
    // make an assign statement family
    Node newParent = root.childrenList.get(root.childrenList.size()-1).childrenList.get(0);
    root.childrenList.get(root.childrenList.size()-1).childrenList.remove( index: 0);
    ArrayList<Node> newChildren = new ArrayList<>();
    ArrayList<Node> var2 = root.childrenList.get(root.childrenList.size()-1).childrenList;
    root.childrenList.remove( index: root.childrenList.size()-1);
    root.type = "<var>";
    newChildren.add(root);
    newChildren.addAll(var2);
    newParent.childrenList = newChildren;
    root = newParent;
}

//shorter path
if(root.childrenList.size()==1){
    Node child = root.childrenList.get(0);
    if(isNonterminal(child.type) && canRemoveNode.contains(root.type)){
        child.type = root.type;
        root = child;
    }
}
return root;
```

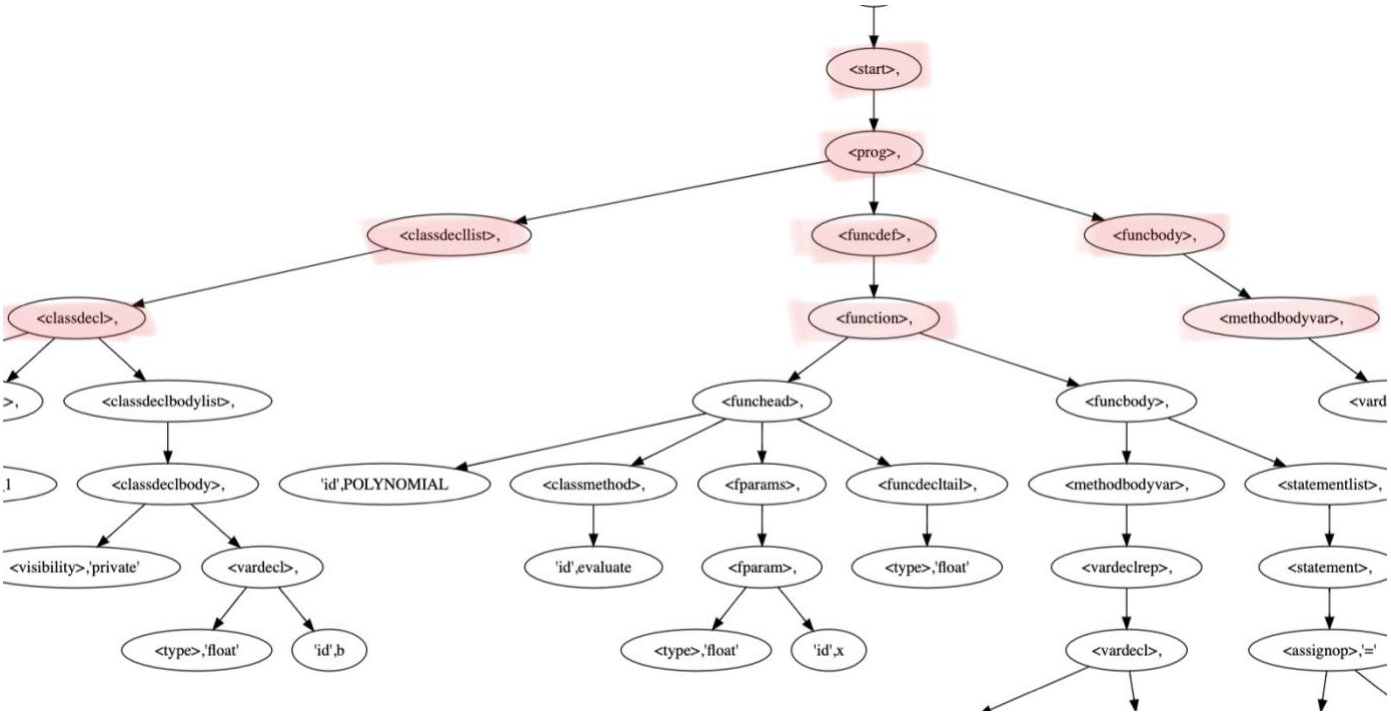
7. When parsing is done, depth traversing the ASTree and output the dot.file.

```
public void writeDot(Node astNode, int parentID) throws IOException {
    String nodeInfo = astNode.id+"[label=\""+astNode.type + ","+ astNode.name+"\"]";
    String relation = parentID + "->" + astNode.id;
    FileWriter dotWrite = new FileWriter( fileName: "src/grammar_result/"+this.dotFile, append: true);
    dotWrite.write( str: nodeInfo + "\n");
    dotWrite.write( str: relation + "\n");
    dotWrite.close();
}

void depthTraverse(Node root) throws IOException {
    if (root.childrenList.size() == 0)
        return ;
    for(Node child : root.childrenList){
        writeDot(child,root.id);
        depthTraverse(child);
    }
}
```

AST structure

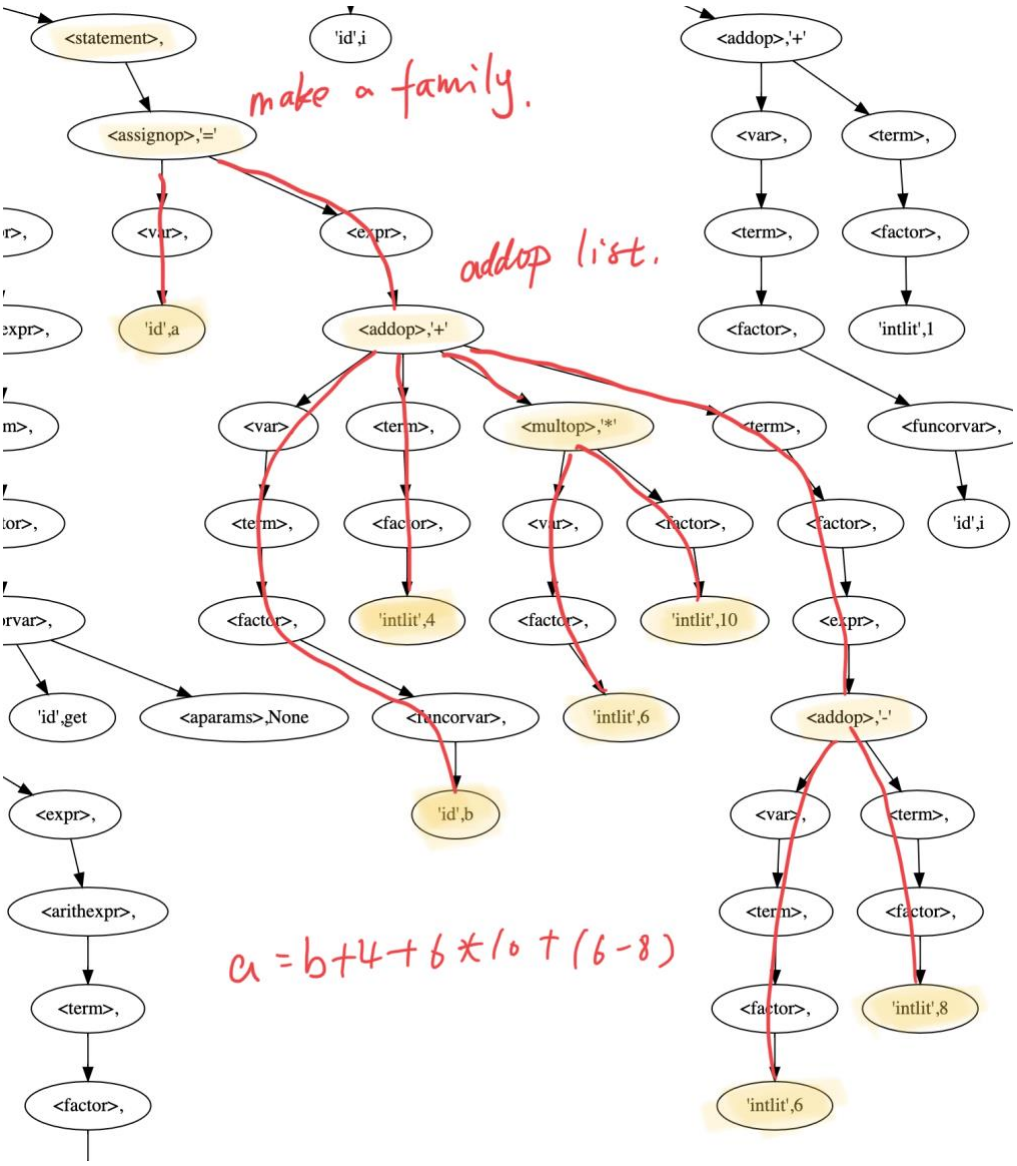
<prog> ::= <classDeclList> <FuncDef> 'main' <funcBody>

<classDeclList> ::= <classDecl> <classDeclList>



Making an assign statement family and make an addOp list

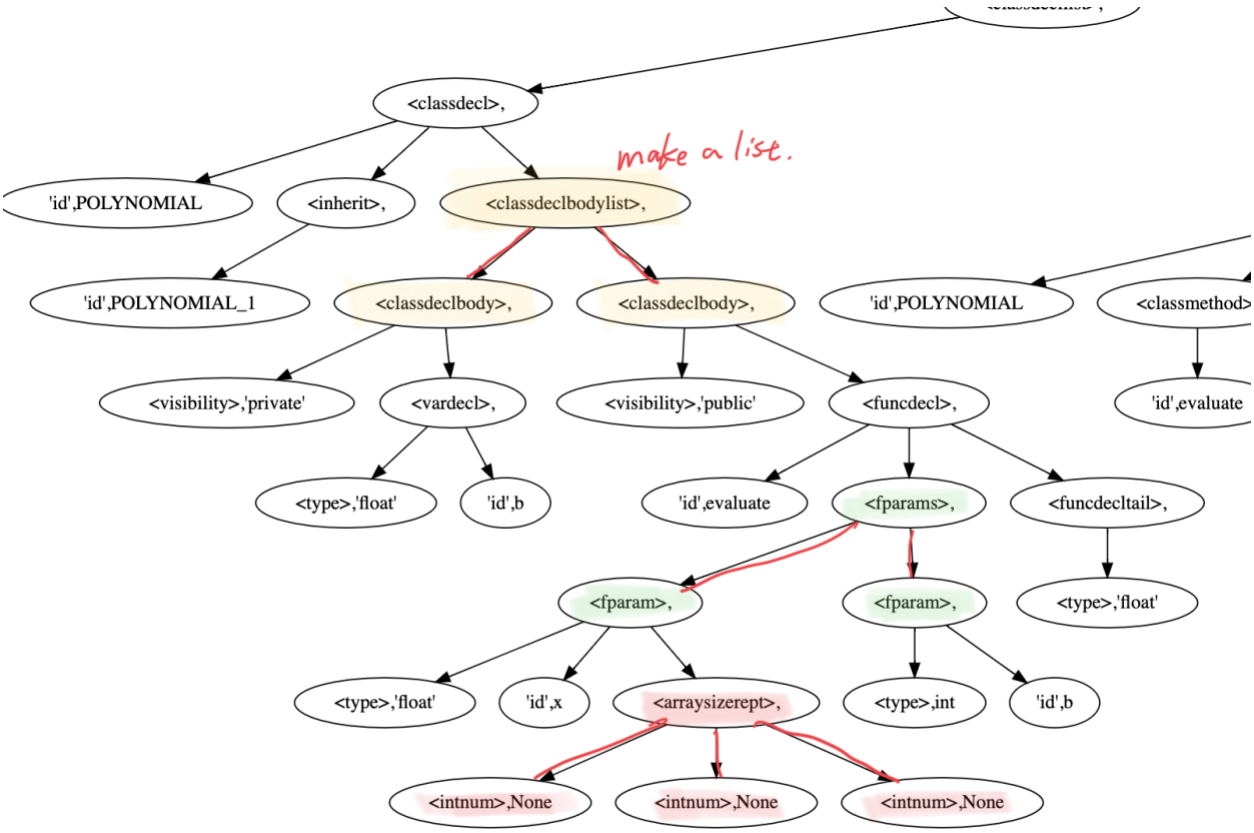
<arithExprTail>::= <addOp> <term> <arithExprTail>



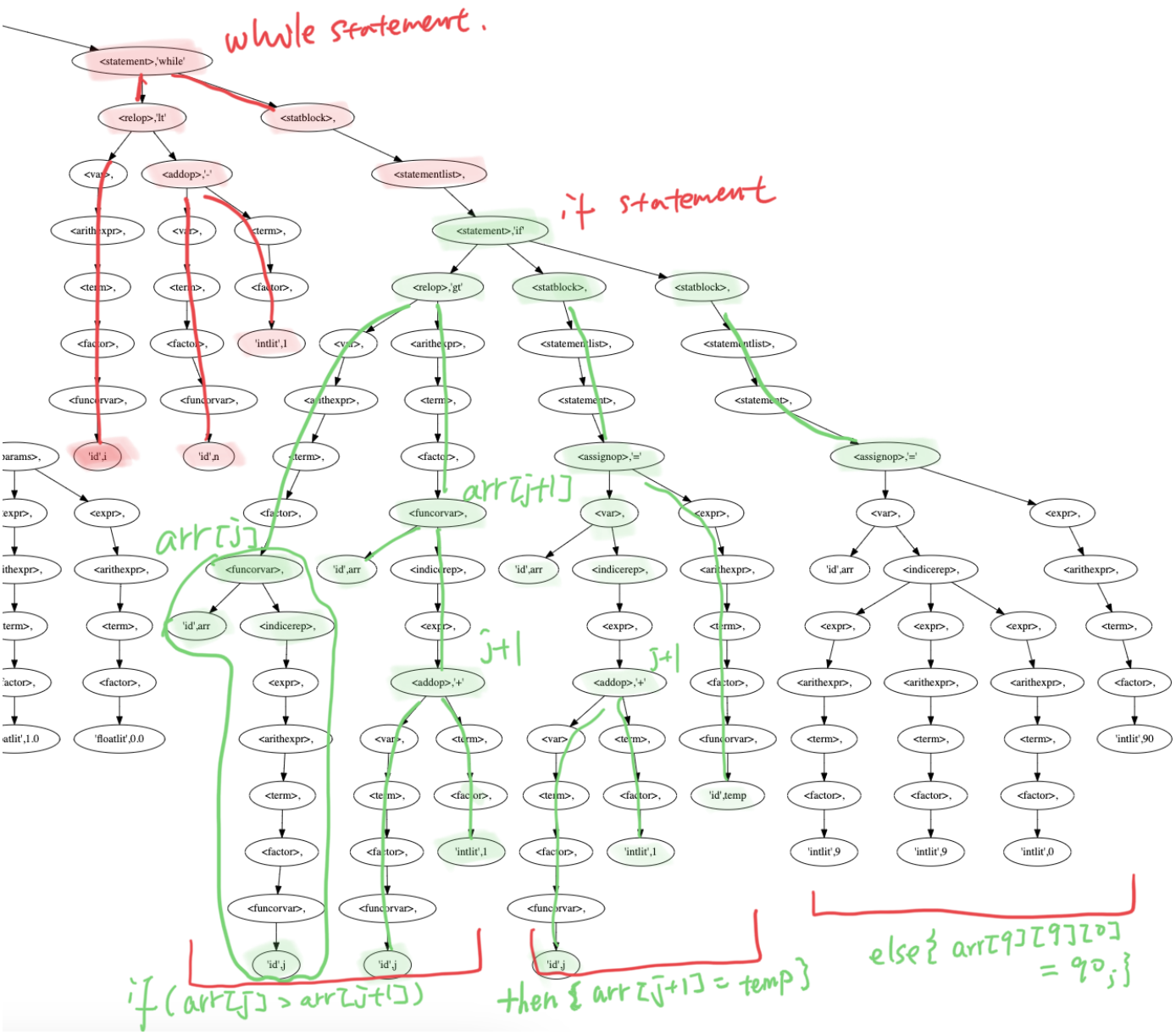
<classDeclList> ::= <classDecl> <classDeclList>

<fParams> ::= <fParam> <FParamsTail>

<StatementList> ::= <statement> <StatementList>



While statement
If statement



Error recovery

1. Check if first set contains lookahead token, if not, then check if first set contains epsilon and follow set contains lookahead tokens, if still no, report the error.
2. Skip error tokens and find a nearest recovery point.

```java
public boolean skipErrors(Production production) throws IOException {
    boolean result = true;
    boolean checkFirstSet = production.mergeFirstSet().contains(peakToken());
    boolean checkFollowSet = checkIfInFLS(production,lookahead);
    if(checkFirstSet || checkFollowSet){
        // no error detected, parse continues in this parsing function
        result =  true;
    }else {
        String errorStr = "syntax error around :"+ getPreToken(tokenIndex).toString() + ", " +lookahead.toString();

        writeError(errorStr);
        // try to skip the error tokens, find a new start
        while (!(checkFirstSet || production.followSet.contains(peakToken())) && lookahead.tokenType.compareTo("$") != 0) {
            lookahead = nextToken();
            if (checkIfInFLS(production,lookahead)){
                result =  false;
                break;    // error detected and parsing function should be aborted
            } else
                continue; // error detected and parse continues in this parsing function
        }
        System.out.println(errorStr);
    }
    return result;
}
```

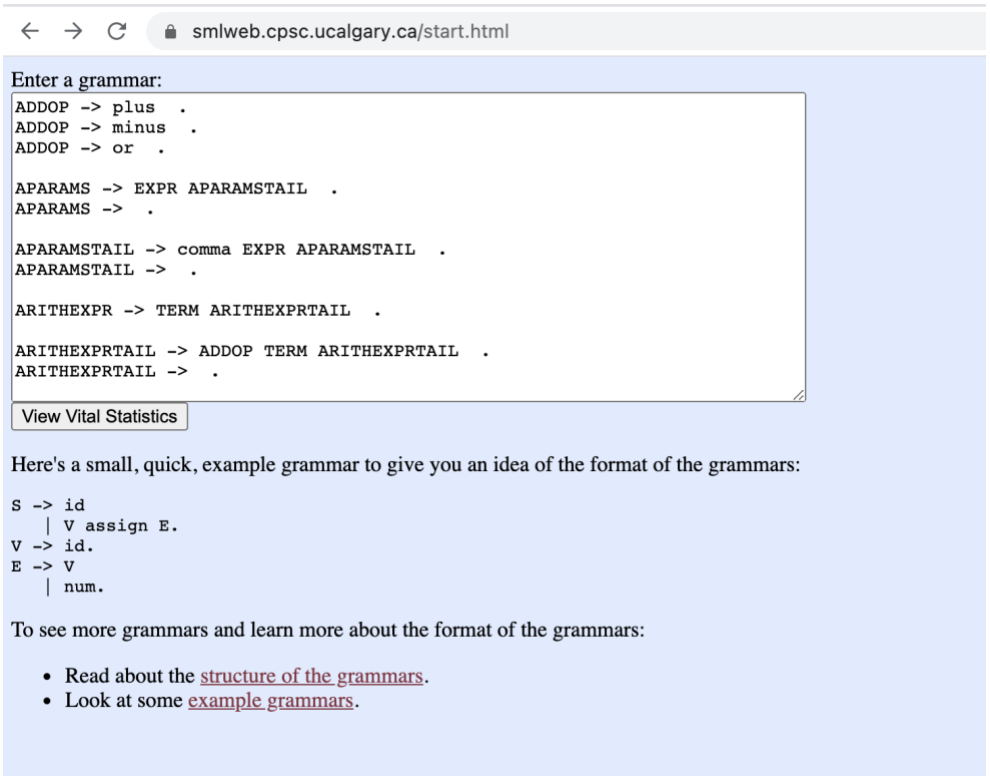
5. Use of tools

- Grammartool.jar
  I used this tool to generate the Ucalgary format grammar.

grammartool.jar

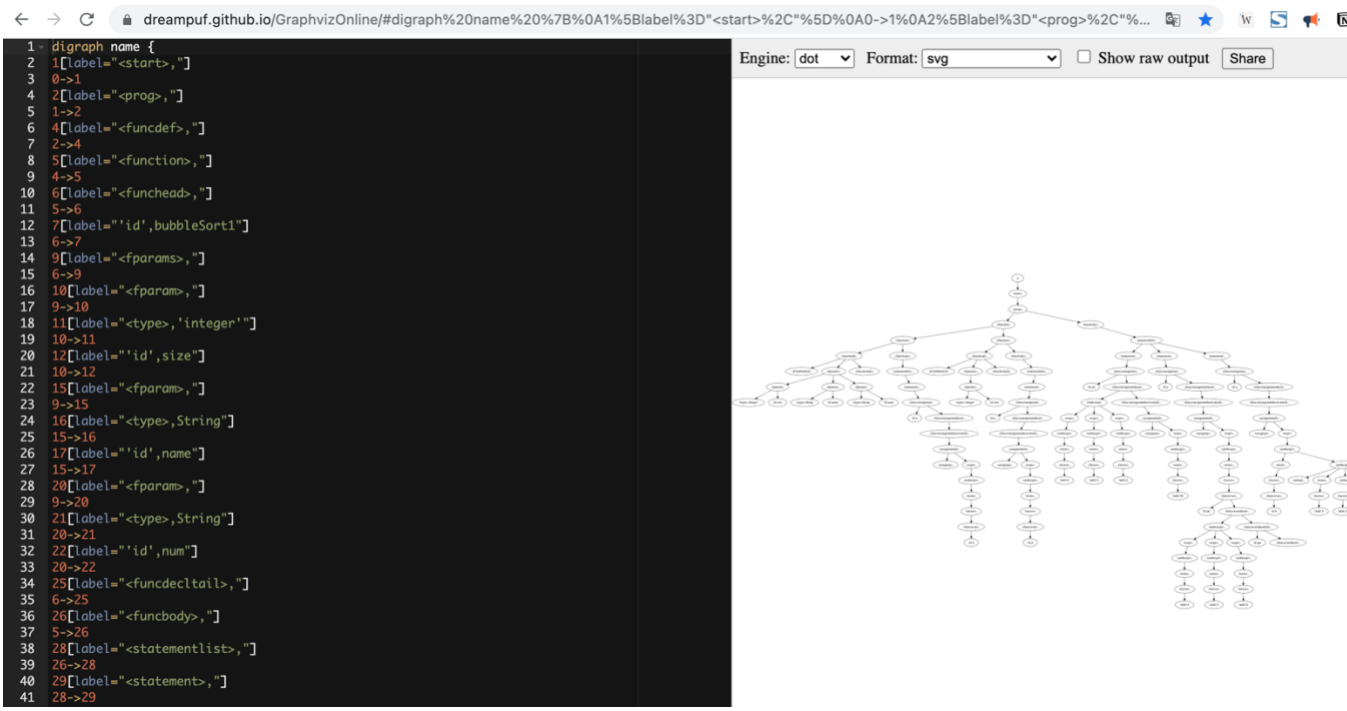
- https://smlweb.cpsc.ucalgary.ca/start.html

  Then, I used this tool to generate the first set and the follow set.



nonterminal	first set	follow set	nullable	endable
ADDOP	plus minus or	intlit floatlit stringlit lpar not qm id plus minus	no	yes
APARAMSTAIL	comma	rpar	yes	no
ARITHEXPRTAIL	plus minus or	semi eq neq lt gt leq geq comma colon rsqbr rpar	yes	no
ASSIGNOP	equal	intlit floatlit stringlit lpar not qm id plus minus	no	no
CLASSDECLBODY	public private func integer float string id	public private func integer float string id rcurbr	no	no
CLASSDECLBODYLIST	public private func integer float string id	rcurbr	yes	no
CLASSDECL	class	func main class	no	no
EXPRTAIL	eq neq lt gt leq geq	semi comma colon rsqbr rpar	yes	no
ARITHEXPR	intlit floatlit stringlit lpar not qm id plus minus	semi eq neq lt gt leq geq comma colon rsqbr rpar	no	no
FPARAM	integer float string id	rpar comma	no	no
FPARAMSTAIL	comma	rpar	yes	no
CLASSMETHOD	sr	lpar	yes	no
FPARAMS	integer float string id	rpar	yes	no
FUNCDECLTAIL	void integer float string id	lcurbr semi	no	no
FUNCORASSIGNSTATIDNESTFUNCTAIL	dot	semi	yes	no
FUNCORASSIGNSTATIDNESTVARTAIL	dot equal	semi	no	no
FUNCORASSIGNSTATIDNEST	lpar lsqbr dot equal	semi	no	no
ASSIGNSTATTAIL	equal	semi	no	no
FUNCORVAR	id	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	no	no
FUNCORVARIDNESTTAIL	dot	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	yes	no
FUNCORVARIDNEST	lpar lsqbr dot	mult div and semi eq neq lt gt leq geq plus minus or comma colon rsqbr rpar	yes	no
APARAMS	intlit floatlit stringlit lpar not qm id plus minus	rpar	yes	no
FUNCSTATTAILIDNEST	dot	semi	yes	no
FUNCSTATTAIL	dot lpar lsqbr	semi	no	no
FUNCTION	func	main func	no	no
FUNCHEAD	func	lcurbr	no	no

- https://dreampuf.github.io/GraphvizOnline/
  I used this website to generate a AST graph.


dreampuf.github.io/GraphvizOnline/#digraph%20name%20%7B%0A1%5Blabel%3D"<start>%2C"%5D%0A0->1%0A2%5Blabel%3D"<prog>%2C"%...
digraph name {
1[label="<start>,"]
0->1
2[label="<prog>,"]
1->2
4[label="<funcdef>,"]
2->4
5[label="<function>,"]
4->5
6[label="<funchead>,"]
5->6
7[label="'id',bubbleSort1"]
6->7
9[label="<fparams>,"]
6->9
10[label="<fparam>,"]
9->10
11[label="<type>,'integer'"]
10->11
12[label="'id',size"]
10->12
15[label="<fparam>,"]
9->15
16[label="<type>,String"]
15->16
17[label="'id',name"]
15->17
20[label="<fparam>,"]
9->20
21[label="<type>,String"]
20->21
22[label="'id',num"]
20->22
25[label="<funcdecltail>,"]
6->25
26[label="<funcbody>,"]
5->26
28[label="<statementlist>,"]
26->28
29[label="<statement>,"]
28->29
Engine: dot
Format: svg
Show raw output
Share