李墨珩 2017050025

编译原理 PA1-B 实验报告

2019年10月29日

实验报告

编译原理 PA1-B

一、工作内容

1、按照指导书中的方法进行错误恢复

```
121         private SemValue parseSymbol(int symbol, Set<Integer> follow) {
122             var result = query(symbol, token); // get production by lookahead symbol
123             Set<Integer> end = followSet(symbol);
124             end.addAll(follow);
125             if (!beginSet(symbol).contains(token)) {
126                 yyerror("syntax error");
127                 while(true) {
128                     if (beginSet(symbol).contains(token)) {
129                         result = query(symbol, token);
130                         break;
131                     }
132                     if (follow.contains(token)) return null;
133                     token = nextToken();
134                 }
135             }
136 
137             var actionId = result.getKey(); // get user-defined action
138 
139             var right = result.getValue(); // right-hand side of production
140             var length = right.size();
141             var params = new SemValue[length + 1];
142 
143             for (var i = 0; i < length; i++) { // parse right-hand side symbols one by one
144                 var term = right.get(i);
145                 params[i + 1] = isNonTerminal(term)
146                         ? parseSymbol(term, end) // for non terminals: recursively parse it
147                         : matchToken(term) // for terminals: match token
148                 ;
149             }
150             for (var i = 1; i < (length+1); i++) if(params[i] == null) return null;
151             act(actionId, params); // do user-defined action
152             return params[0];
153         }
154 
```

2、特性实现

由于PA1-A阶段的工作有一部分可以复用，本阶段更改的文件主要为LLParser.java 和 Decaf.spec 两个。

由于工作内容全部是文法相关，触发动作采用的函数均为PA1-A阶段已经存在的函数，所以接下来会通过展示代码来更直观地了解工作内容。

抽象类和抽象方法：添加关键字和触发动作

```
ClassDef        :   CLASS Id ExtendsClause '{' FieldList '}'
                    {
                        // add 'false' as first parameter to pass compile
                        $$ = svClass(new ClassDef(false, $2.id, Optional.ofNullable($3.id), $5.fieldList, $1.pos));
                    }
                |   ABSTRACT CLASS Id ExtendsClause '{' FieldList '}'
                    {
                        $$ = svClass(new ClassDef(true, $3.id, Optional.ofNullable($4.id), $6.fieldList, $2.pos));
                    }
                ;
```

```
FieldList       :   STATIC Type Id '(' VarList ')' Block FieldList
                    {
                        $$ = $8;
                        // change parameter to pass compile
                        $$.fieldList.add(0, new MethodDef(Modifiers.STATIC, $3.id, $2.type, $5.varList, $7.block, $3.pos));
                    }
                |   ABSTRACT Type Id '(' VarList ')' ';' FieldList
                    {
                        $$ = $8;
                        $$.fieldList.add(0, new MethodDef(Modifiers.ABSTRACT, $3.id, $2.type, $5.varList, null, $3.pos));
                    }
```

Var 类型：添加关键字和触发动作

```
SimpleStmt      :   Var Initializer
SimpleStmt      :   NONETYPE Id '=' Expr
                    {
                        $$ = svStmt(new LocalVarDef(null, $2.id, $3.pos, Optional.ofNullable($4.expr), $2.pos));
                    }
                |   Var Initializer
                    {
                        $$ = svStmt(new LocalVarDef($1.type, $1.id, $2.pos, Optional.ofNullable($2.expr), $1.pos));
                    }
```

lambda表达式：添加关键字和触发动作，注意提取左公因式。

```
Expr            :   Expr1
                    {
                        $$ = $1;
                    }
                |   FUN '(' VarList ')' FUNExpr
                    {
                        if ($5.expr != null) {
                            $$ = svExpr(new Lambda($3.varList, $5.expr, $1.pos));
                        } else if ($5.block != null) {
                            $$ = svExpr(new Lambda($3.varList, $5.block, $1.pos));
                        }
                    }
                ;

FUNExpr         :   GOSETO Expr
                    {
                        $$ = $2;
                    }
                |   Block
                    {
                        $$ = $1;
                    }
                ;
```

函数类型：在ArrayType中直接添加括号内容，并在thunklist中添加TypeList信息用来追踪。

```
Type            :   AtomType ArrayType
                    {
                        $$ = $1;
                        for (int i = 0; i < $2.intVal; i++) {
                            if ($2.thunkList.get(i) == null){
                                $$.type = new TArray($$.type, $1.type.pos);
                            } else {
                                $$.type = new TLambda($$.type, $2.thunkList.get(i).typeList, $1.type.pos);
                            }
                        }
                    }
                ;

ArrayType       :   '[' ']' ArrayType
                    {
                        $$ = $3;
                        $$.intVal++;
                        $$.thunkList.add(0, null);
                    }
                |   '(' TypeList ')' ArrayType
                    {
                        $$ = $4;
                        $$.intVal++;
                        $$.thunkList.add(0, $2);
                    }
                |   /* empty */
                    {
                        $$ = new SemValue();
                        $$.thunkList = new ArrayList<>();
                        $$.intVal = 0; // counter
                    }
                ;
```

```
TypeList        :   Type TypeListEx
                    {
                        $$ = $2;
                        $$.typeList.add(0, $1.type);
                    }
                |   /* empty */
                    {
                        $$ = svTypes();
                    }
                ;

TypeListEx      :   ',' Type TypeListEx
                    {
                        $$ = $3;
                        $$.typeList.add(0, $2.type);
                    }
                |   /* empty */
                    {
                        $$ = svTypes();
                    }
                ;
```

函数调用：调整了 Expr8 Expr8T Expr9 以及 AfterLParen中的文法，删除了ExprListOpt。

```
ExprT8          :   '[' Expr ']' ExprT8
                    {
                        var sv = new SemValue();
                        sv.expr = $2.expr;
                        sv.pos = $1.pos;

                        $$ = $4;
                        $$.thunkList.add(0, sv);
                    }
                |   '.' Id  ExprT8
                    {
                        var sv = new SemValue();
                        sv.id = $2.id;
                        sv.pos = $2.pos;


                        $$ = $3;
                        $$.thunkList.add(0, sv);
                    }
                |  '(' ExprList ')'  ExprT8
                    {
                    var sv = new SemValue();
                    sv.pos = $1.pos;

                    if ($2.exprList != null) {
                        sv.exprList = $2.exprList;

                    }
                    $$ = $4;
                    $$.thunkList.add(0, sv);
                    }

                | /* empty */
                    {
                        $$ = new SemValue();
                        $$.thunkList = new ArrayList<>();
                    }
                ;
```

```
AfterLParen     :   CLASS Id ')' Expr7
                    {
                        $$ = svExpr(new ClassCast($4.expr, $2.id, $4.pos));
                    }
                |   Expr ')' ExprT8
                    {
                        $$ = $1;
                        for (var sv : $3.thunkList) {
                            if (sv.expr != null) {
                                $$ = svExpr(new IndexSel($$.expr, sv.expr, sv.pos));
                            } else if (sv.exprList != null) {
                                $$ = svExpr(new Call($$.expr, sv.exprList, sv.pos));
                            } else {
                                $$ = svExpr(new VarSel($$.expr, sv.id, sv.pos));
                            }
                        }
                        $$.pos = $$.expr.pos;
                    }
                ;

Expr8           :   Expr9 ExprT8
                    {
                        $$ = $1;
                        for (var sv : $2.thunkList) {
                            if (sv.expr != null) {
                                $$ = svExpr(new IndexSel($$.expr, sv.expr, sv.pos));
                            } else if (sv.exprList != null) {
                                $$ = svExpr(new Call($$.expr, sv.exprList, sv.pos));
                            } else {
                                $$ = svExpr(new VarSel($$.expr, sv.id, sv.pos));
                            }
                        }
                        $$.pos = $$.expr.pos;
                    }
                ;
```

二、回答问题

1、在处理else的时候，与最近的else进行匹配。如下图，生成查找表时会出现一个关于else的warning，在处理文法时候，会选择前者，因此是与最近的else匹配。

```
1  Warning: conflict productions at line 260:
2  ElseClause -> ELSE Stmt
3  ElseClause -> <empty>
```

2、以Expr为例：

通过从左开始文法分析，使用如图中的文法规则会使得优先级 op7>op6>op5…>op1 。对于结合性，是通过非终结符在文法产生式中的位置来确定的。在本次的文法中，以lambda表达式为例，它是右结合的。

```
Expr1           :   Expr2 ExprT1
                    {
                        $$ = buildBinaryExpr($1, $2.thunkList);
                    }
                ;

ExprT1          :   Op1 Expr2 ExprT1
                    {
                        var sv = new SemValue();
                        sv.code = $1.code;
                        sv.pos = $1.pos;
                        sv.expr = $2.expr;

                        $$ = $3;
                        $$.thunkList.add(0, sv);
                    }
                |   /* empty */
                    {
                        $$ = new SemValue();
                        $$.thunkList = new ArrayList<>();
                    }
                ;

Expr2           :   Expr3 ExprT2
                    {
                        $$ = buildBinaryExpr($1, $2.thunkList);
                    }
                ;

ExprT2          :   Op2 Expr3 ExprT2
                    {
                        var sv = new SemValue();
                        sv.code = $1.code;
                        sv.pos = $1.pos;
                        sv.expr = $2.expr;

                        $$ = $3;
                        $$.thunkList.add(0, sv);
                    }
                |   /* empty */
                    {
                        $$ = new SemValue();
                        $$.thunkList = new ArrayList<>();
                    }
                ;
```

```
Expr            :   Expr1
                    {
                        $$ = $1;
                    }
                |   FUN '(' VarList ')' FUNExpr
                    {
                        if ($5.expr != null) {
                            $$ = svExpr(new Lambda($3.varList, $5.expr, $1.pos));
                        } else if ($5.block != null) {
                            $$ = svExpr(new Lambda($3.varList, $5.block, $1.pos));
                        }
                    }
                ;

FUNExpr         :   GOSETO Expr
                    {
                        $$ = $2;
                    }
                |   Block
                    {
                        $$ = $1;
                    }
                ;
```

3、

```
class Main {
    static abstract void main() { }
}
```

```
parallels@zhaohengs-ubuntu:/media/p
*** Error at (2,12): syntax error
*** Error at (2,33): syntax error
*** Error at (3,1): syntax error
```

此时理想状态应该是扔掉abstract，只报（2，12）一处错误，然而却报出了3个错误，这是因为abstract在class的follow集合里面，因此也就在static的end集合里面，进而造成了这种错误。