PA1-B Report

王琛, 计 65, 2016011360

使用的 JDK 版本为 java version "1.8.0_191"

1 本阶段工作

1.1 错误恢复

使用的方法和 README 中的相同，遇到非终结符 A，若当前输入符号 $a \notin Begin(A)$，则报错并继续往下扫描。若遇到了 $Begin(A)$，恢复分析 A，若遇到了 $End(A)$，返回 null 并继续分析后面的符号。这部分需要注意的是执行 act 函数之前需要先检查 params 数组是否有为空的元素，如果有直接返回 null，否则会产生语法错误。

1.2 新特性的 LL(1) 文法

- 对象复制语句：题中的文法即是 LL(1) 文法，直接实现即可。
- 引入关键词 sealed：题中的文法即是 LL(1) 文法，直接实现即可。
- 串行条件卫士语句：文法中开头的 IF 和原来 IfStmt 产生式左侧的 IF 为左公因子，需要先提取。另外，在产生 IfBranch* IfSubStmt 时，需要注意左递归的问题，应将左递归的因子放入产生式的右边。
- 简单的自动类型推导：因为 var 关键字，没有左公因子和左递归，直接实现即可。
- 若干与一维数组有关的表达式和语句：
  - 数组常量：主要问题还是左递归，将递归因子放在右边即可
  - 数组初始化常量表达式：实现方法和 +、-基本相同，由于优先级低于 +、-，将其放在加减前面，其余部分实现完全相同。

- 数组拼接表达式：同样是一个运算符，实现方法和%% 也基本相同，由于其优先级低于%%，将其置于%% 之前。唯一区别在于右递归的实现，我的思路是：和上一步的顺序遍历相反，将 svec 倒过来遍历，将当前元素作为第一个参数，上一步得到的 expr 作为第二个参数进行构造。
- 取子数组表达式 && 数组动态下标访问表达式：这两个特性需要同时实现。原因在于两者有左公因子"[Expr"，另外为了在上一级区分两个不同的情况，在 Semvalue 中添加 expr2，expr1 用于 Expr，expr2 用于 default Expr 部分，判断两种情况是用 expr1 == null 或 expr2 == null 即可。
- python 风格的 comprehension 表达式：由于使用新增的 [| 和 |]，文法直接是 LL(1)，实现即可。
- 数组迭代语句：本身是 LL(1) 文法，直接实现。

1.3 运行截图

图 1: S1+

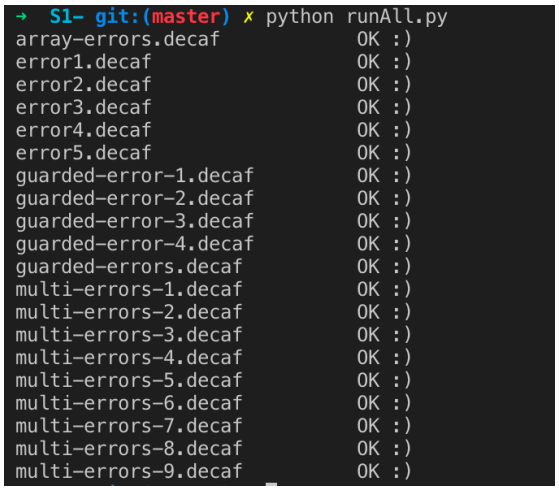

图 2: S1-

2 Decaf 语言由于允许 if 语句的 else 分支为空，因此不是严格的 LL(1) 语言，但是我们的工具依然可以处理这种冲突。请根据工具所生成的预测分析表中 if 语句相关项的预测集合先做猜测，并对照工具 wiki，理解本工具的处理方法。请在实验报告中说明此方法的原理，并举一个具有这种冲突的 Decaf 语言程序片段，说明它哪里有冲突以及如何解决。

实验工具是通过定义不同产生式的优先级来解决这个问题的。LL(1) 的文法要求为对于每个非终结符 A 的任意两个产生式，$A \to \alpha$ 和 $A \to \beta$，都有 $PS(A \to \alpha) \cap PS(A \to \beta) = \varnothing$，但是由于允许 if 语句 else 为空，这个条件不再成立。令 $C = PS(A \to \alpha) \cap PS(A \to \beta) = '('$，将 $PS(A \to \beta)$ 改成 $PS(A \to beta) - C$ 就满足了 LL(1) 文法的条件。

下面是 Decaf 语言中 if 二义性的例子：

```
class Main {
```

```
    static void main() {
        int t;
        t = 0;
        if (t > 0)
        if (t < 0) t = -1;
        else t = 1;
    }
}
```

若不加任何处理，else 不知道会匹配到哪个 if，但是利用上面的方法，匹配第一个 if 时无 else，将会使用空，到下面的 if 才会匹配到 else，语法分析树的结果如下。

```
    program
    class Main <empty>
        static func main voidtype
            formals
            stmtblock
                vardef t inttype
                assign
                    varref t
                    intconst 0
                if
                    gtr
                        varref t
                        intconst 0
                    if
                        les
                            varref t
                            intconst 0
                        assign
                            varref t
                            neg
                                intconst 1
                    else
```

```
                    assign
                        varref t
                        intconst 1
```

3 为什么把原先的数组 comprehension 表达式文法 Expr ::= [Expr for identifier in Expr <if BoolExpr>] | ... 改写为 LL(1) 比较困难?

原因在于左公因子，直接使用'[' 和']' 后产生的报错信息如下:

```
1 table generation:
 [java] Warning: conflict productions at line 973:
 [java] ElseClause -> ELSE Stmt
 [java] ElseClause -> <empty>
 [java] Warning: conflict productions at line 411:
 [java] Expr -> Expr1
 [java] Expr -> '[' Expr FOR IDENTIFIER IN Expr IF Expr ']'
 [java] Warning: unreachable production:
 [java] Expr -> '[' Expr FOR IDENTIFIER IN Expr IF Expr ']'
 [java] predictive set is empty
```

'[' 和后面数组常量的'[' 产生了冲突，并且这个左公因子是无法直接提出的。Parser.spec 中为了确定优先级是通过逐层递进的方法，由 Expr1、Expr2... 越到下面优先级越高，而数组常量位于 Expr9，层级相差较多，如果要消除左公因子会比较麻烦。

4 无论何种错误处理方法，都无法完全避免误报的问题。请举出一个语法错误的 Decaf 程序例子，用你实现的 Parser 进行语法分析会带来误报。根据你用的错误处理方法，这些误报为什么会产生?

误报程序和结果如下:

```
    class Main {
    static void main() {
        int[] x;
        x = new int[11];
        int y;
        y = x[)10];
    }
}

*** Error at (6,15): syntax error
*** Error at (6,18): syntax error
```

第 6 行多了一个')', 在匹配 [,] 中的 expr 时，首先遇到')' 匹配失败报错，由于 $) \in End(Expr)$，Expr 返回 null，即放弃匹配 Expr, 继续向后匹配，但是对于'[' 放弃 Expr 应该匹配']'，这时却出现 10 了，因此再次报错。实际上，应该只报'(' 的错，这里却产生了误报。

Acknowledgement

本次实验李映辉同学曾给予我思路的指导，尤其是数组拼接表达式和数组动态下标访问的实现。