PA1-B 实验报告

2018011289 包涵

实验简述

- 任务一： 仿照框架中ll(1)文法的实现方法，处理和改写新语法。
    - 抽象类： 直接在普通类的生成式基础上，增加了抽象类的生成式。并按照pa1a中对ast节点的修改更新了相应的函数规则。
    - 抽象方法： 同抽象类。
    - 局部类型推断： 在非终结符Simple的产生式中，加入局部类型推断的产生式。
    - First-class Functions
        - 增加新expr： 引入非终结符Expr0，负责生成lambda表达式的声明。与pa1b的语法描述相同，Expr0与Expr1都可以被Expr生成，因此=>具有最低的优先级。而且因为是右结合，不需要进行reverse的特殊处理。
        - 增加新类型： 增加新类型时，主要是要对type消除左递归，因为是“左结合”的，还使用了类似merge_terms的处理方法，把匹配到的lambda参数列表从左往右处理。
        - 调用： 将`IdOrCall -> '(' ExprListOrEmpty ')'`的产生式改为`Call -> '(' ExprListOrEmpty ')'`，加入`Term8 -> Call Term8`的产生式。
    -
- 任务二： 错误恢复部分按照实验说明中的方法，当出现错误时，往后看lexer中的Token，直到遇到Begin(A)或End(A)中的符号。如果遇到Begin(A)中的符号，那么对`prod`和`rhs`赋值，继续进行分析；如果遇到End(A)中的符号，那么将lexer回退到上一个token的位置，返回`StackItem::_Fail`
- 挑战和解决方法
    - ll(1)语法复杂了很多，而且框架的实现也有几个tricky的地方，所以大致看明白用了不少时间。改写新文法时，该写在哪和如何消除递归都让人比较困惑。通过仔细对照框架实现和decaf的语法约定，以及一些尝试，基本实现了新的语法。
    - 错误恢复部分，由于对lexer的实现不了解，不知道怎么才能实现在碰到End(A)中符号之后，不消耗输入符号。后来看了下github仓库lalr1中lexer的实现，通过改lexer.col和lexer.line实现了回退，目前大概还是存在bug。有点迷惑的是，在multi-errors-7.decaf中，_Eof的loc和最后的'}'是一样的，造成输出的错误少一个，不知道是不是我实现的问题。

问题回答

1. 本阶段框架是如何解决空悬 else (dangling-else) 问题的？

   规定了，当产生冲突时，选择先出现的产生式。`MaybeElse -> Else Blocked`先于`MaybeElse ->`出现，因此if会优先结合就近的else。

2. 使用 LL(1) 文法如何描述二元运算符的优先级与结合性？请结合框架中的文法，举例说明。

    - 优先级： 通过把不同优先级的运算符放在不同“级别”的产生式中，优先级越低的运算符，所在的产生式“级别”越高。这样最高级别，即最先完成匹配的产生式对应的函数会最后返回，相应的运算符也就有了最低的级别。比如or就在Expr1级别，而and在Expr2级别，在递归分析时，and对应的规则会先返回，而or对应的函数一定在and之后返回，也就有了更低的优先级。
    - 结合性： 通过E->O+F， F->+OF|e的文法，保证了靠右的+号，在分析树更深的位置，因而对应的函数先返回，也就是在parse的时候+号的运算数会按照从右到左的顺序进入一个栈中，在完成同级别表达式的处理之后，再将运算数依次出栈并处理，就形成了正确的左结合的表达式。

3. 无论何种错误恢复方法，都无法完全避免误报的问题。 请举出一个具体的 Decaf 程序（显然它要有语法错误），用你实现的错误恢复算法进行语法分析时会带来误报。 并说明该算法为什么无法避免这种误报。

    程序如下:

```
class Main {
    static int main() {
        else (n > 10) { n = 1; }
        if (10) {
            n = 1;
        }
    }
}
```

报错为：

```
*** Error at (3,9): syntax error
*** Error at (4,9): syntax error
```

在分析StmtList时遇到else，由于else属于StmtList的end集合，跳过了StmtList，然后返回分析Block，由于else还在输入流中，无法分析Block，跳过else并在同一位置报错。返回分析FieldList，之后一直跳过符号，直到'}'，结束分析Block。返回分析ClassList，一直跳过符号直到_Eof，_Eof在ClassList的begin集合中，分析结束。主要原因是在遇到end集合中的符号时，直接跳过了在分析的非终结符，而这个情况下，非终结符的级别很高，就造成了一系列误报。