

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 \rightarrow '(' ExprListOrEmpty ')'的产生式改为Call \rightarrow '(' ExprListOrEmpty ')', 加入Term8 \rightarrow 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，不知道怎样才能实现在碰到End(A)中符号之后，不消耗输入符号，目前应该还是存在bug。

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 \rightarrow Else Blocked先于MaybeElse \rightarrow 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 $E \rightarrow O + F$, $F \rightarrow +OF \mid e$ 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。

答：