PA 1-B 实验报告

计62 徐晟 2016011253

1. 实验要求

在本实验PA1-B中，实验要求：

1. 查看LL(1)分析算法的实现
2. 添加错误恢复功能
3. 增加新的特性对应的LL(1)文法

本次实验要求使用新的spec文件代替parser.y对文法进行定义。

1. 实验完成
   1. Spec文件观察：

spec文件用于定义文法，主要分成三部分：保留字的定义，运算符的定义和运算式的定义。

其中我们需要在PA 1-A的基础上添加的部分是新定义的数组初始化和数组链接。

* 1. Spec文件修改：

我们不妨定义数组初始化的token为INITIALIZE，数组链接的token为APPEND。

首先我们再定义两个oper分别对应这两个token，然后再定义两个新的文法Expr 44和Expr 444，同时因为要求是LL(1)文法，提取它们的左公因式就好了。

* 1. Lexer.java修改：

Lexer.java只需要加上两个新token，即INITIALIZE和APPEND就可以了。

* 1. Parser.parse()修改：

为了支持错误查询和修改，我修改了parser.java文件中的parse()函数，如果出现错误则调用lex()函数报错并且直接返回空表示跳过错误。

* 1. Tree.java修改：

Tree.java修改比较简单，只需要先添加两个token，即INITIALIZE和APPEND，最后在binary类的printTo()函数中添加相应的语句即可。

1. 实验问题回答
   1. If-else分支为空：

如果有一句话为if Expr1 then Expr2 else Expr3，则这是一个标准的if-else表达式。

但是如果有一句话为if Expr1 then if Expr2 then Expr3 else Expr4，则由于少了一个else，按理说这是一个错误的if-else表达式，因为我们不知道else匹配哪一个if。

但是由于我们假定这是一个LL(1)文法并且使用自顶向下的分析方法，所以这个else会先匹配里面的if，即if Expr1 then if Expr2 then Expr3 else Expr4 🡪 if Expr1 then (if Expr2 then Expr3 else Expr4) else NULL.

考虑wiki中给出的文法：

S -> if C then S E

E -> else S | <empty>

我们发现E(else or empty)的确是后匹配的，所以if Expr1 then if Expr2 then Expr3 else Expr4 🡪 if Expr1 then (if Expr2 then Expr3 else Expr4) else NULL.

* 1. 困难原因

因为”[”和”]”被两种不同的语法复用了，如果还需要实现的话，需要再进行一次左公因子提取.

* 1. parser报错语法

代码形如：

class Main {

static void main() {

var arr = 1,2,3,4];

}

}

我们发现，这时候代码会直接报错，而不会进行异常处理.

原因是代码已经把arr当做是1（一个variable）进行处理了，所以之后所有碰到的”1,2,3,4]“这部分一个都识别不出来.