2016-11-16

邸思诺

141250027

语法分析实验报告

目录

[1.实验目的 3](#_Toc467061347)

[2.内容描述 3](#_Toc467061348)

[3.思路方法 3](#_Toc467061349)

[4.假设 3](#_Toc467061350)

[5.相关分析过程描述 4](#_Toc467061351)

[5.1自定义CTG 4](#_Toc467061352)

[5.2求出所有非终结符的first和follow 5](#_Toc467061353)

[5.3构建PPT 5](#_Toc467061354)

[5.4 实现ll1分析器 5](#_Toc467061355)

[6.重要数据结构 6](#_Toc467061356)

[7.核心算法 8](#_Toc467061357)

[7.1.求first 8](#_Toc467061358)

[7.2求follow 9](#_Toc467061359)

[7.3 构造表 10](#_Toc467061360)

[7.4运行截图 11](#_Toc467061361)

[8.问题与解决 13](#_Toc467061362)

[9.感受与总结 13](#_Toc467061363)

## 1.实验目的

为了更深刻的理解语法分析器的运作原理，将知识与实践相结合，锻炼动手能力。

## 2.内容描述

本实验采用了LL1 自顶向下的分析方法。较LR相比有局限性，但也可分析一些基本的文法。

与lab1的词法分析器相结合，可以将输入的简单代码，经过词法、语法的分析，输出推导过程中用到的相关文法产生式（产生式可在main里面改写），并且对不符合该文法的语句报错。

为了降低复杂度，本实验只可分析

while循环，if else语句，加法，乘法，赋值语句。

## 3.思路方法

1.与lab1相结合，首先将输入的代码转化成token序列(遇到无法识别的会报错)

2.根据输入的文法，求出所有非终结符的first，follow

3.根据first，follow，构造LL1预测分析表

4.构造一个栈底是$的符号栈，放入开始符

5.在token序列末尾添加$

6.对符号栈，token序列，依照预测分析表逐一分析：

(1)栈顶是终结符：

a.若读头与栈顶不同，报错

b.若相同，读头右移，该符号出栈

(2)栈顶是非终结符：

在PPT表中寻找横坐标是该非终结符、纵坐标是读头处符号的单元格。

a.若该单元格值为-1，报错(表示不能继续推导下去)。

b.若非负：将该数字对应的文法产生式加入结果队列，

栈顶符号出栈，

1>若该产生式的右式是，则进行下一步。

2>否则该产生式的右式从右到左一次入栈。

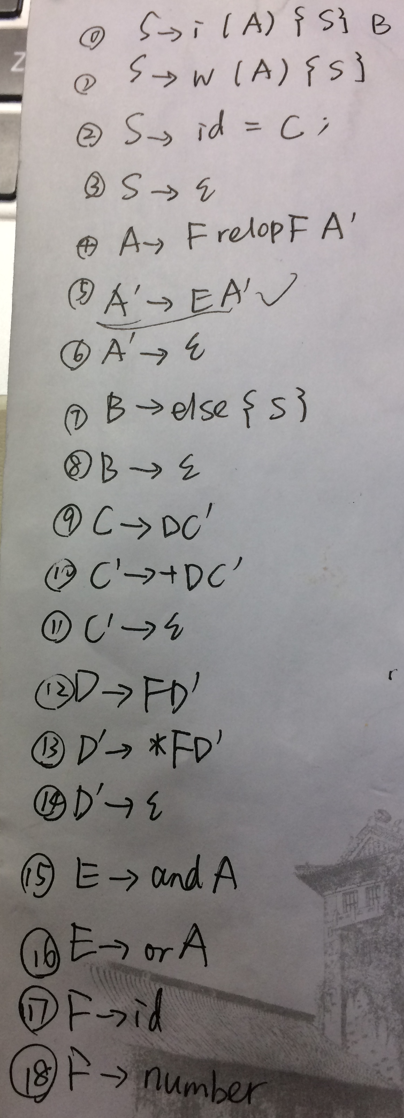
(3)循环前两步，直到栈顶，读头处 都是$.

## 4.假设

1.输入的代码是正常代码

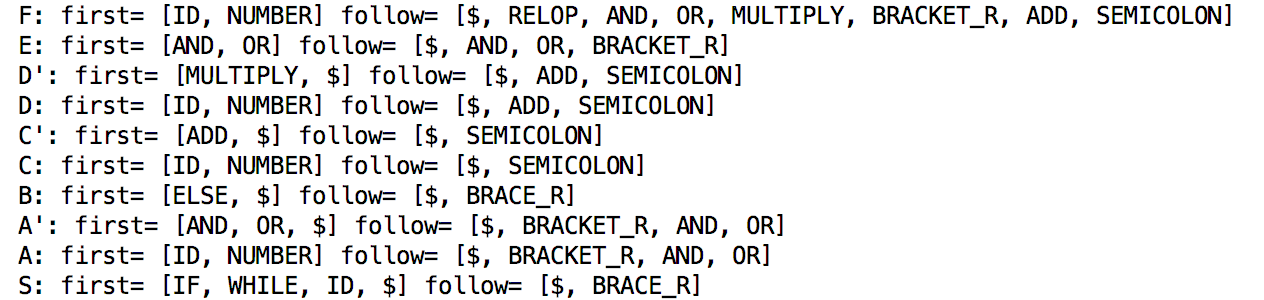
## 5.相关分析过程描述

### 5.1自定义CTG

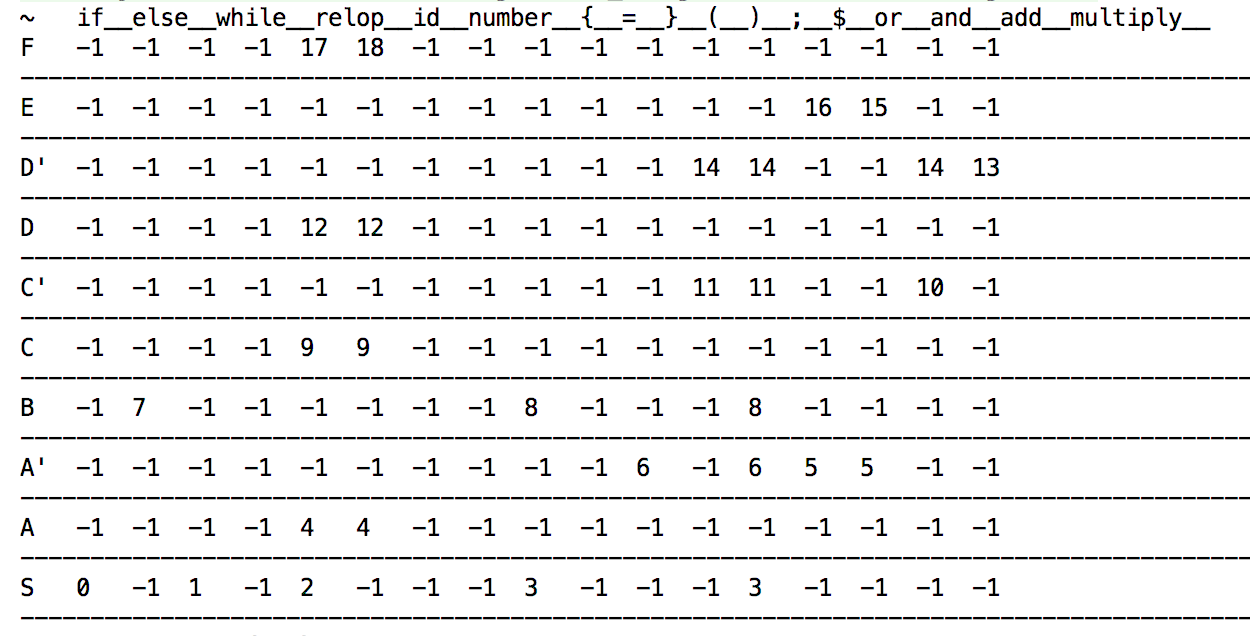


### 5.2求出所有非终结符的first和follow

并存储在全局的arraylist里面。此文法对应的first follow集合如下



### 5.3构建PPT



纵坐标是所有非终结符，横坐标的所有终结符，正数代表该数字对应的产生式。-1代表不能继续分析了，会报错。

### 5.4 实现ll1分析器

如前面思路方法的描述。以防助教大大直接看这里我就粘贴一下~

1.构造一个栈底是$的符号栈，放入开始符

2.在token序列末尾添加$

3.对符号栈，token序列，依照预测分析表逐一分析：

(1)栈顶是终结符：

a.若读头与栈顶不同，报错

b.若相同，读头右移，该符号出栈

(2)栈顶是非终结符：

在PPT表中寻找横坐标是该非终结符、纵坐标是读头处符号的单元格。

a.若该单元格值为-1，报错(表示不能继续推导下去)。

b.若非负：将该数字对应的文法产生式加入结果队列，

栈顶符号出栈，

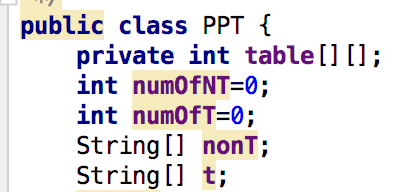
1>若该产生式的右式是，则进行下一步。

2>否则该产生式的右式从右到左一次入栈。

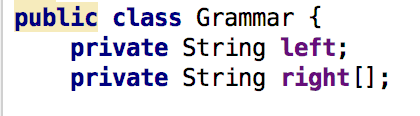
(3)循环前两步，直到栈顶，读头处 都是$.

## 6.重要数据结构

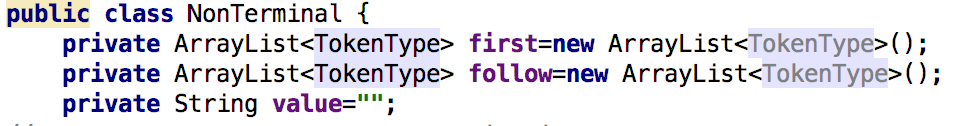
1.预测分析表：持有所有非终结符列表，所有终结符列表，和一个二维int数组的分析表



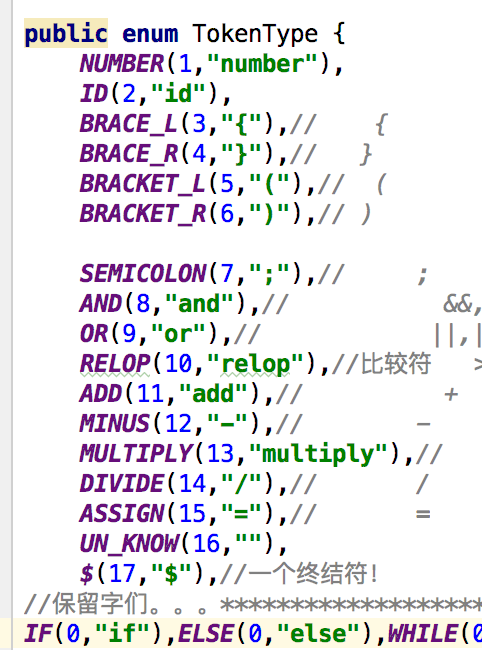
2．文法：持有左式 和右式的string序列（一个token对应一个string）



3.非终结符：持有自己的符号，和first，follow序列（由于first，follow都由终结符构成，而终结符都属于tokentype（已添加$））



4.tokentype:这里也相当于终结符



## 7.核心算法

### 7.1.求first



### 7.2求follow



### 7.3 构造表

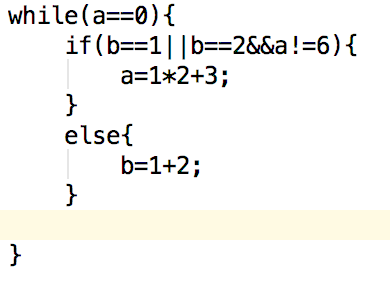


7.4 分析token序列

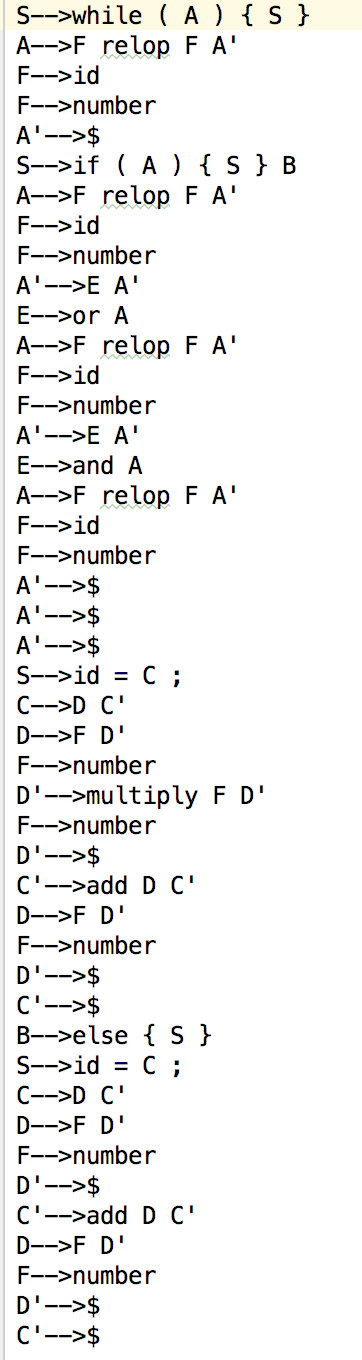


## 7.4运行截图

输入：



输出：



## 8.问题与解决

1.Q：如果多调用几次求first的函数，有的非终结符的first集就越来越少了。。

A:后来发现原来ArrayList是传递的引用！并没有拷贝。于是对参数的操作就反映到对应非终结符的first集合上了

2.Q：lab1 tokenType的设计不够详细，导致无法精确分析。比如加法和乘法是有优先级差别的，不应仅仅作为运算符混在一起

A：重新定义tokentype，增加了一些枚举常量

3.Q：follow有时候需要往回看。比如A的follow是B与C的follow的并集，但此时B和C的follow集还是空的

A：用一个while循环，不断调用求follow的函数，直到每个非终结符的follow集合的size不再变化为止。

## 9.感受与总结

每次都是，想的比做的简单。。本来也是满腔热血的准备写LR的，但是这个难度对我来说耗时太多。。要考试了，就，，以后有机会再尝试吧~

写完实验觉得对语法分析器的实现过程理解的更加深刻啦~