**基于SLR(1)分析法的语法制导翻译及中间代码生成程序设计原理与实现**

**16281208**

**熊超**

**2018.5.31**

目录

[1.实验目的 3](#_Toc11237609)

[2.设计要求 3](#_Toc11237610)

[3.实验分析 4](#_Toc11237611)

[3.1定义Vt，Vn，产生式集合 4](#_Toc11237612)

[3.2 求出first集 4](#_Toc11237613)

[3.3求follow集 5](#_Toc11237614)

[3.4 扩充文法 5](#_Toc11237615)

[3.5 闭包操作 6](#_Toc11237616)

[3.6 读操作 6](#_Toc11237617)

[3.7 构造GOTO表、ACTION表 7](#_Toc11237618)

[3.8 SLR1分析和四元式的生成 8](#_Toc11237619)

[4.实验测试 10](#_Toc11237620)

[5.实验心得与体会 11](#_Toc11237621)

## 1.实验目的

完成以下描述赋值语句SLR(1)文法语法制导生成中间代码四元式的过程。

G[A]:A→V=E

E→E+T∣E-T∣T

T→T\*F∣T/F∣F

F→(E)∣i

V→i

## 2.设计要求

**（**1）构造文法的SLR(1)分析表，设计语法制导翻译过程，给出每一产生式对应的语义动作；

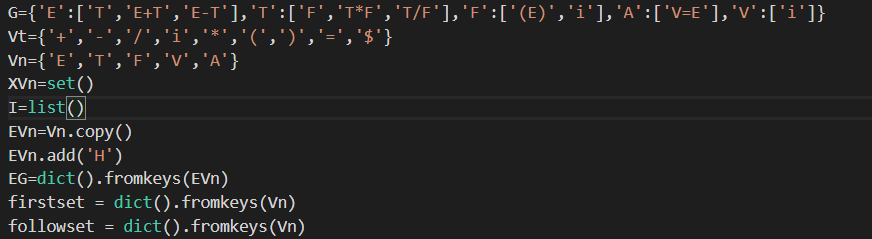
（2）设计中间代码四元式的结构；

（3）输入串应是词法分析的输出二元式序列，即某赋值语句“专题1”的输出结果，输出为赋值语句的四元式序列中间文件；

（4）设计两个测试用例（尽可能完备），并给出程序执行结果四元式序列。

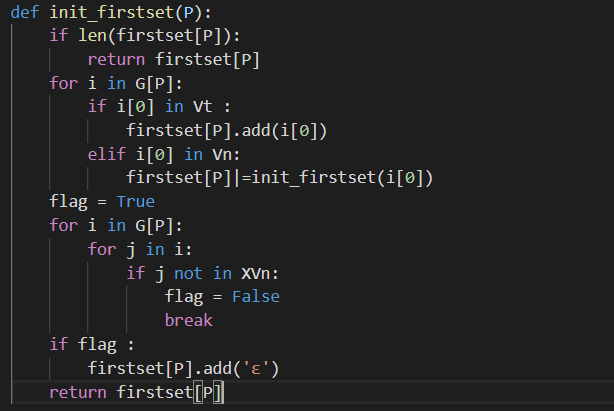
## 3.实验分析

### 3.1定义Vt，Vn，产生式集合



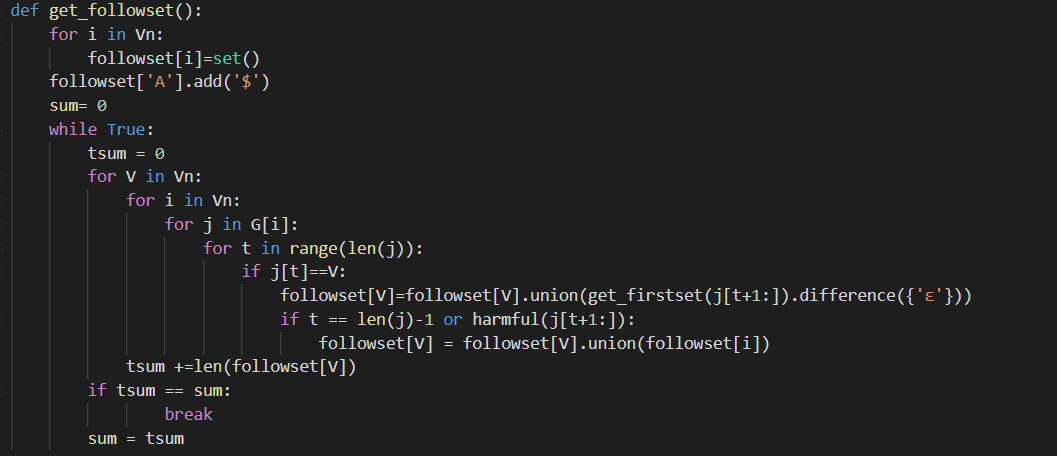
并将文法扩展为H->·E·

### 3.2 求出first集



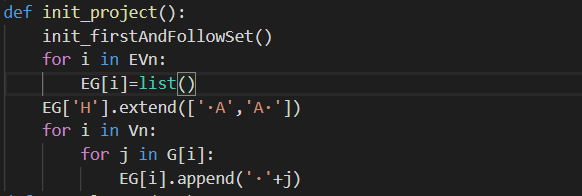
实现求解算法，使用firstset记忆化，减少递归

### 3.3求follow集



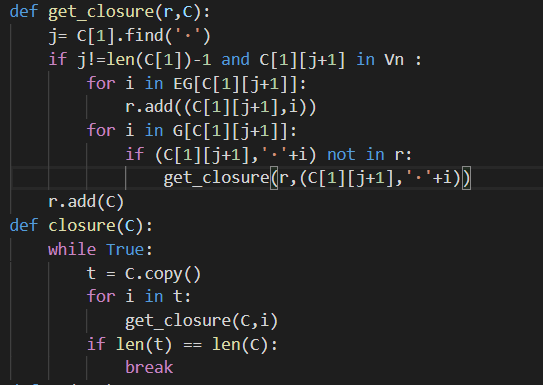
实现求解算法，当整个follow集不再增大时，停止求解

### 3.4 扩充文法



在产生式右部的每个符号之间加入·符号

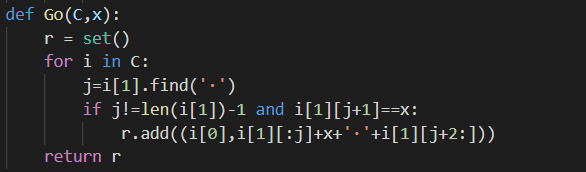
### 3.5 闭包操作



在get\_closure函数中，r为状态集合，C是产生式，格式为（右部，左部）意为将与A->P等价的产生式加入集合r

而closure函数中，C为状态集合，因为python不允许迭代过程中改变容器，则在每次扩充时，迭代另一个容器来扩充C。当C中产生式数量不变时，停止求解

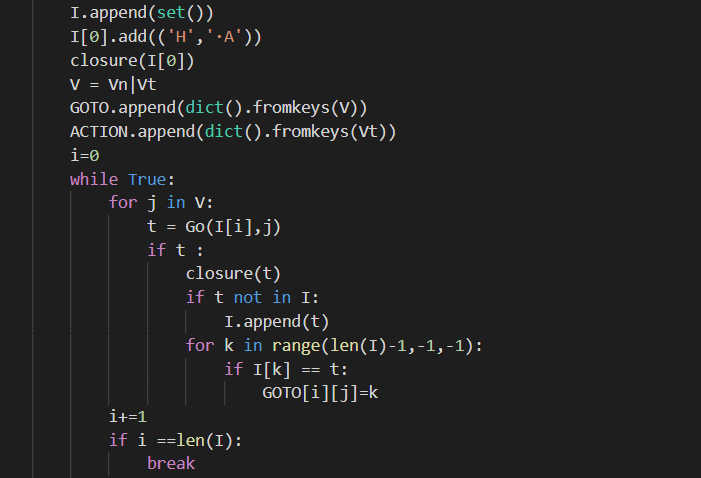
### 3.6 读操作



每次传入一个符号，找到下一个产生式，加入集合C，若不能找到返回的是空集合

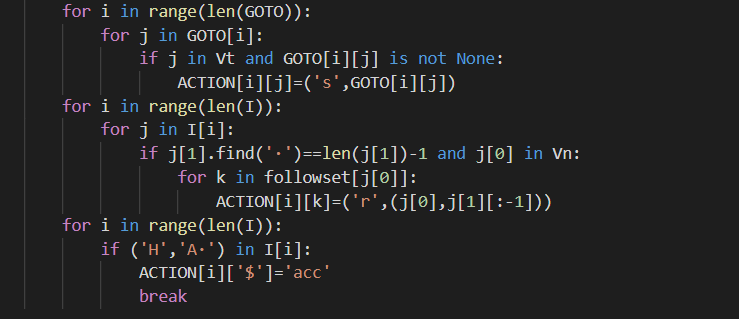
### 3.7 构造GOTO表、ACTION表

GOTO表



首先加入初始状态，并对其进行闭包操作，后面每次循环读入一个状态，进行闭包，同时将其扩充GOTO表

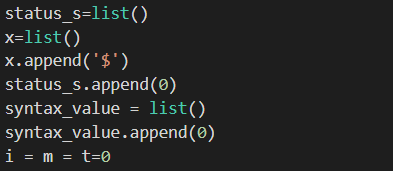
ACTION表



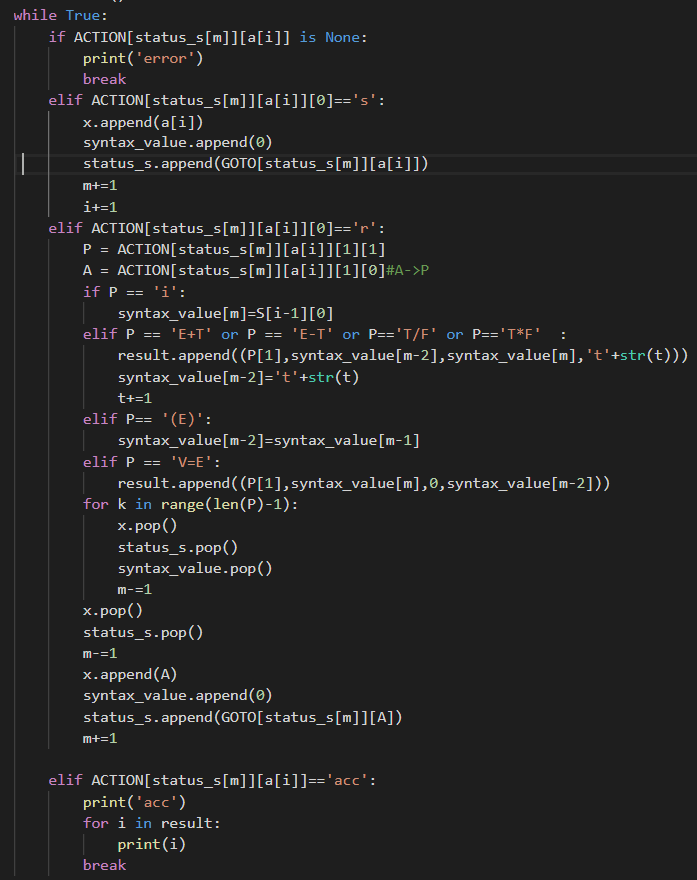
对于GOTO[i][j]若j是终结符就移入，

若是归约项目就对在其产生式的follow集符号j，ACTION[i][j]归约此处ACTION表元素为方便我设为(r,产生式)的二元组形式

### 3.8 SLR1分析和四元式的生成



Status\_s为状态栈，x为符号栈，i为输入指针，m为状态栈指针。Syntax\_value是记录当前符号栈语义属性的栈。



基本就是基本算法的实现。

不同的是在移进的时候，同时扩充语义栈，保持跟符号栈同大小。

当遇到归约时，当归约形如X->i的产生式时，说明此时符号应该是输入串上一个输入符号。

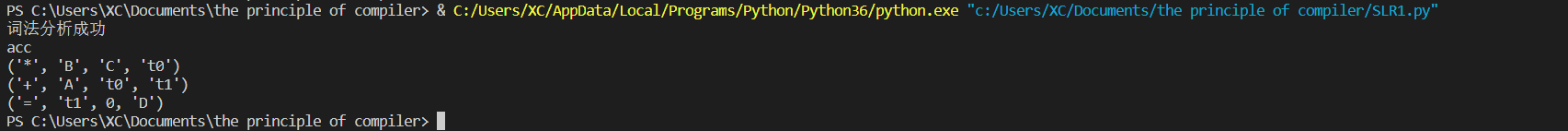
碰到算符时，说明将要产生四元式，因为语义栈和符号栈同大小，只需将语义属性填入即可，并在记录好临时变量的使用情况。

## 4.实验测试

**用例1：**

对于输入

D=A+B\*C



识别成功

**用例2：**

D=A+B/



识别失败

## 5.实验心得与体会

通过本次专题实验，我更加深入的理解了SLR1分析器的构造，锻炼了了自己的编程能力。