SLR(1)类文法判定及其分析器构造

以下参考书籍均来自：编译原理(第三版) 清华大学出版社 [王生原](https://book.douban.com/search/%E7%8E%8B%E7%94%9F%E5%8E%9F) / [董渊](https://book.douban.com/search/%E8%91%A3%E6%B8%8A) / [张素琴](https://book.douban.com/search/%E5%BC%A0%E7%B4%A0%E7%90%B4) / [吕映芝](https://book.douban.com/search/%E5%90%95%E6%98%A0%E8%8A%9D) / [蒋维杜](https://book.douban.com/search/%E8%92%8B%E7%BB%B4%E6%9D%9C)

1. 定义

大多数程序设计语言的文法不能满足LR（0）文法的要求，即不存在移入-归约冲突，归约-归约冲突。而SLR（1），其基本思想是基于容许LR（0）规范族中有冲突的项目集用向前查看一个符号的办法进行处理，以解决冲突。

移入-归约冲突，归约-归约冲突解决方法举例：

编译原理(第三版) **p139** 表6.7下的例子

SLR(1)分析程序由三部分组成：

总控程序：控制整体流程

分析表或分析函数：分析表可分为动作表（ACTION）和状态转换表（GOTO）表两个部分，均可用二维数组表示。

分析栈：包括文法符号栈和相应的状态栈。

1. 实现过程

**实现步骤**

1. 首先求出输入文法非终结符的FOLLOW集
2. 构造输入文法的LR(0)项目集规范族。
3. 使用GO函数规则识别活前缀的DFA
4. SLR（1）分析表的构造
5. SLR（1）分析器的工作过程

**原理解释**

1. 首先求出输入文法非终结符的FOLLOW集

（上次试验已经讲过，此处略过）

1. 构造输入文法的LR(0)项目集规范族、使用GO函数规则识别活前缀的DFA、SLR（1）分析表的构造。

我使用书中 **p134**页构造识别文法活前缀DFA的三种方法的第三种方法：

即把拓广文法的第一个项目{S’ -> .S}作为初态集的核， 通过求核的闭包和转换函数，求出LR（0）**项目集规范族**，由转换函数建立状态之间的连接关系，得到**识别活前缀的DFA**，由SLR（1）的构造规则，得到**分析表**。

其中使用closure闭包函数，求解LR（0）项目集规范族。具体规则也很简单，如果 . 后面的符号是非终结符，就继续将以该非终结符开头的产生式放到该项目集中。重复上述步骤。

使用转换函数GO(I,X) = CLOSURE(J)求出新状态 J的项目集并建立状态之间的转换关系。

LR（1）的分析表的构造与LR（0）分析表的构造类似，但是采用SLR（1）的处理思想，即对所有的非终结符都求出其follow集，这样仅当规约项目在面临的输入的符号包含在该规约项目左部非终结符的FOLLOW集合中时，才采用该产生式规约的动作。

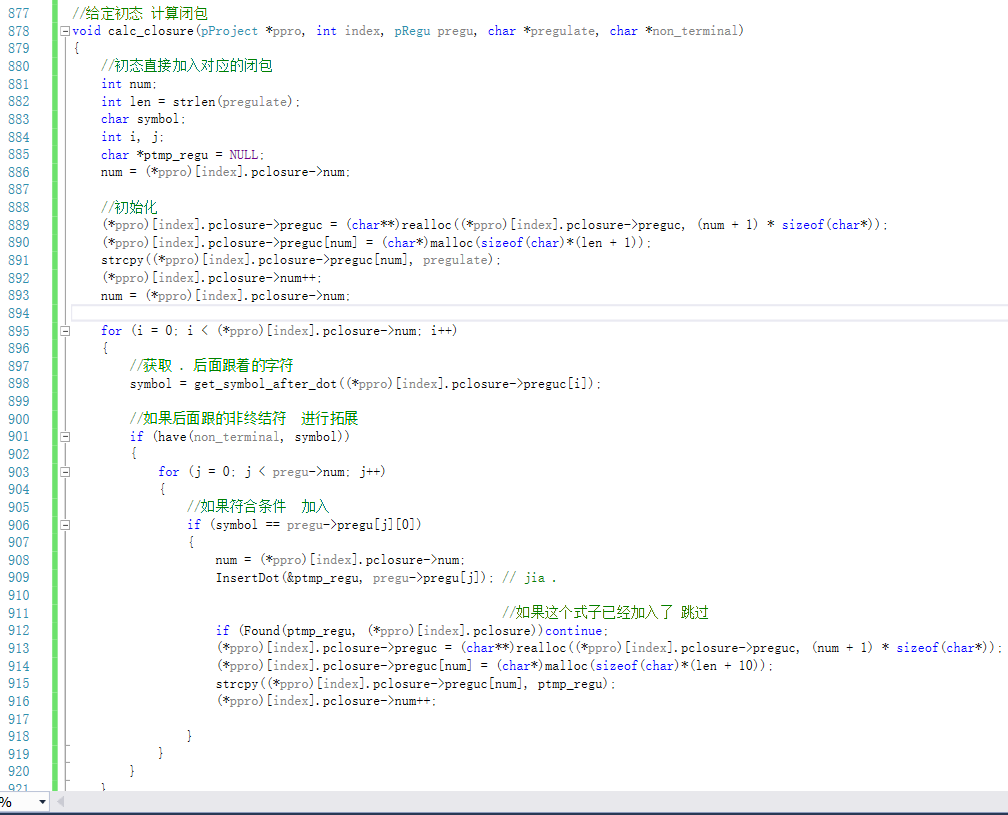
改进后的分析表的构建方法见 **p142**

1. SLR（1）的分析器规则

直接套用了LR（0）的分析器的工作过程。具体说明见书中 **p136**

主要函数截图及说明

1.Closure函数如下



2.上述**原理解释**的第二条中所指出的第三种方法的实现部分



详情见代码注释部分

calc\_analyze\_table //构造分析表

GetDFA // 用于构造项目集中某个初态产生式的所有dfa转移状态 以及对应的分析表构造

IsnewDFA //判断是否转移到已经处理完成的DFA 项目集

parsecontent //分析表工作过程

ExistDFA //判断该新的状态转移是不是已经创建了，如果是获取下标

**代码运行举例**

给出课本上的示例文法：

**E->E+T**

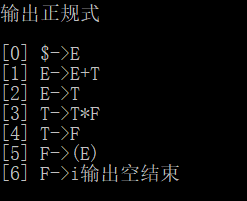
**E->T**

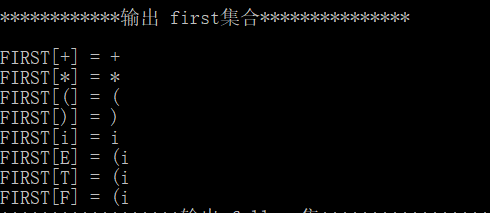
**T->T\*F**

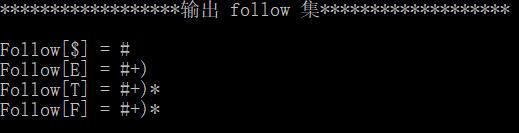
**T->F**

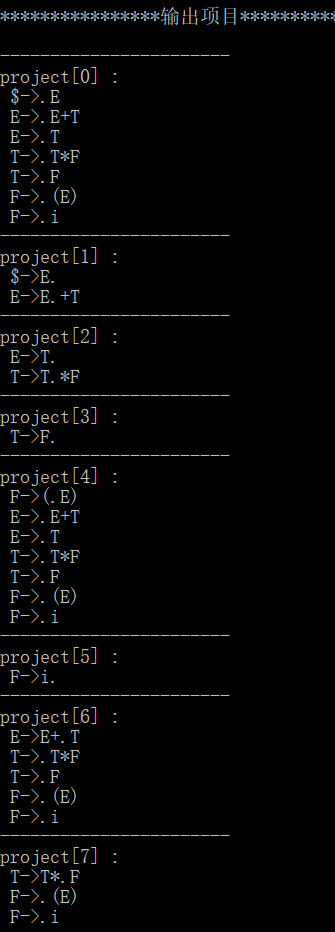
**F->(E)**

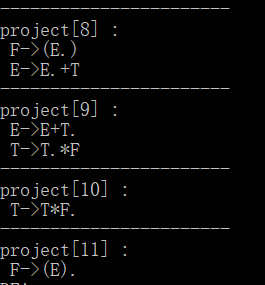
**F->i**

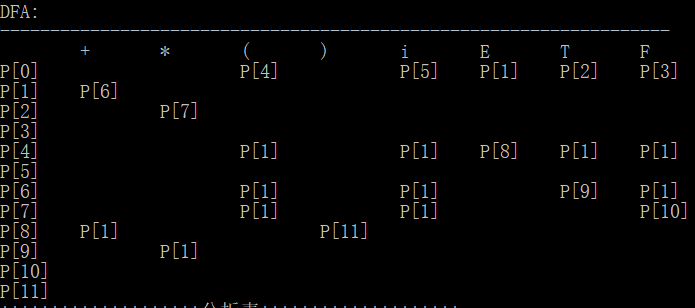


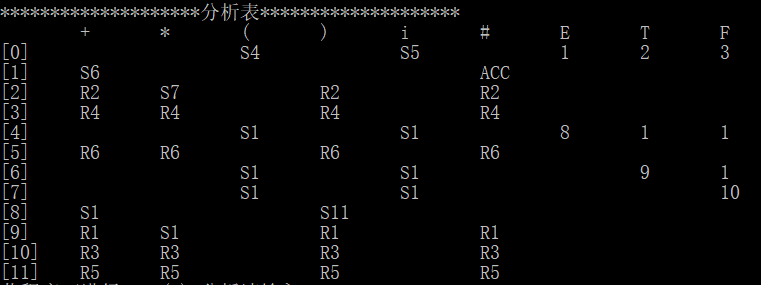












输入i\*i+i#进行分析，分析结果如下：

