

编译原理实验设计说明



实验四 构造LR(0)文法的Aciton表和GOTO表

学生姓名 杨振华

学生学号 201626811225

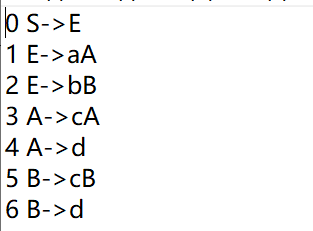
学生班级 软件工程 1603

任课教师 张端

提交日期 2018年12月9日

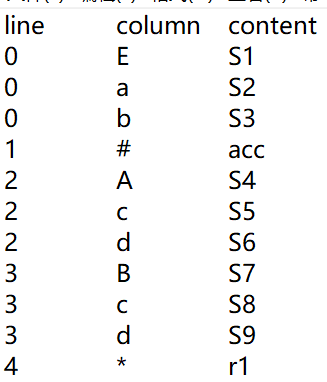
**计算机科学与技术学院**

1. 备注
2. 本实验代码采用C++14的新特性，低版本编译器无法通过编译。
3. 本程序的输入格式为



每一行由产生式序号、一个空格和产生式的内容组成。输入需要保证文法的开始符号为S，且只能出现一条S->X的产生式，X为任意非终结符。（题目中的文法满足这里的所有条件）。

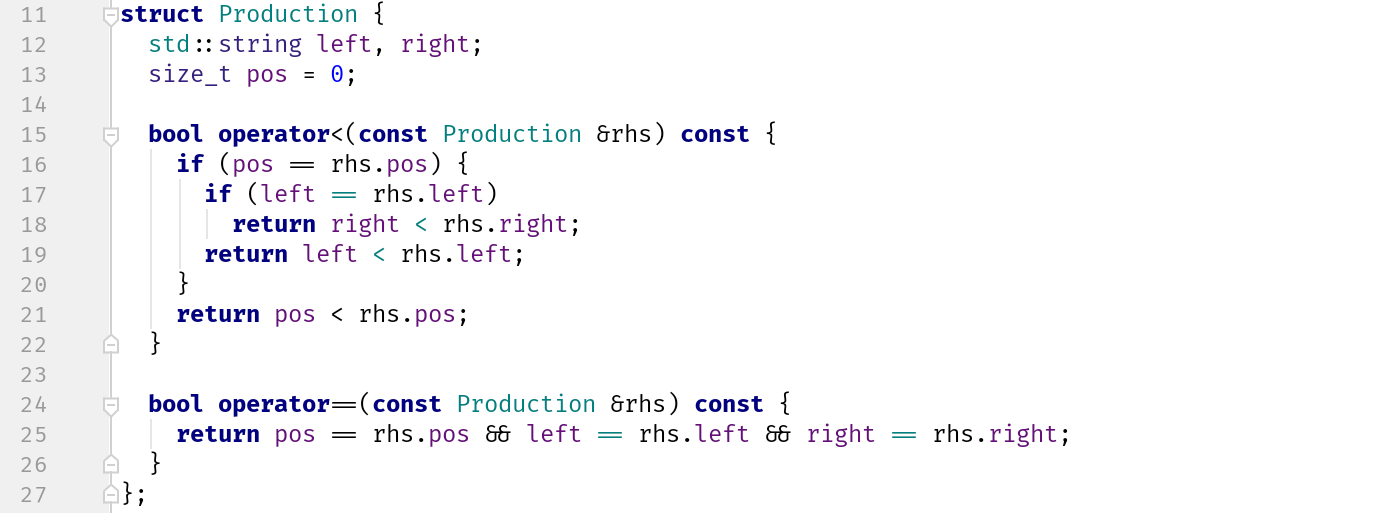
1. 本程序的输出格式为



输出由三列组成，分别表示Action、Goto表的行号、列号、非空单元格的内容（若列号为\*表示所有列号为非终结符和#的列），输出中没有包含的单元格内容均为空。

1. 由于遍历的顺序不同，输出和ppt中给出的LR(0) FSM结点序号略有不同，但等价。
2. 设计说明

首先，定义产生式类。



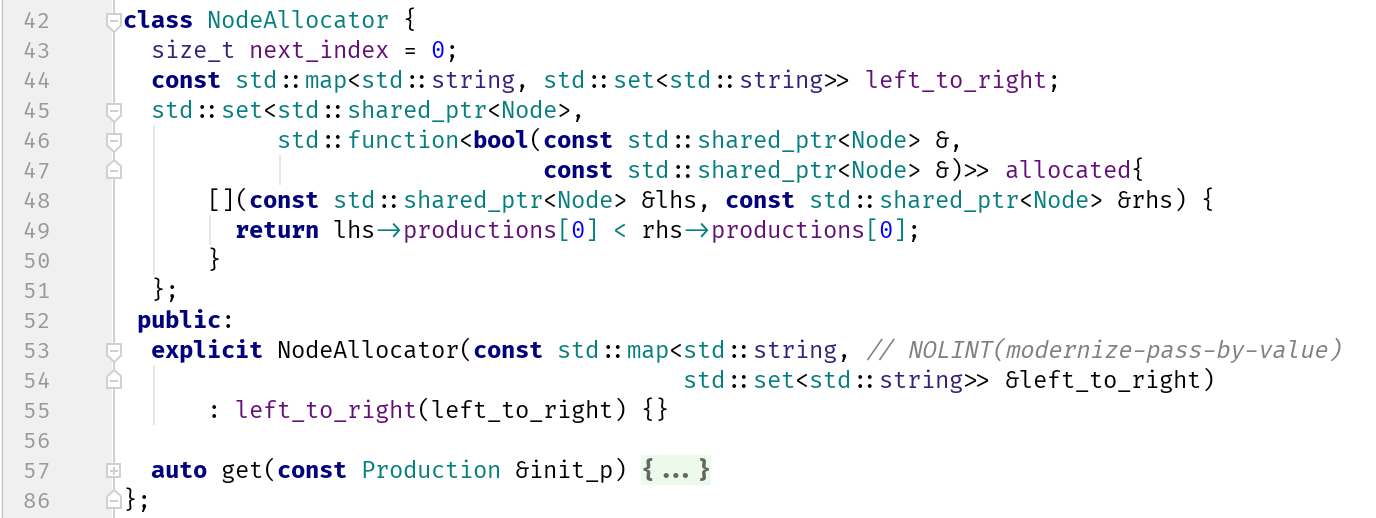
产生式类中定义了每个产生式的左部和右部，均为字符串类型，还有一个pos变量代表FSM中产生式上“点”的位置。产生式类重载了<和==，以备后续使用。

接下来，定义FSM的结点类。

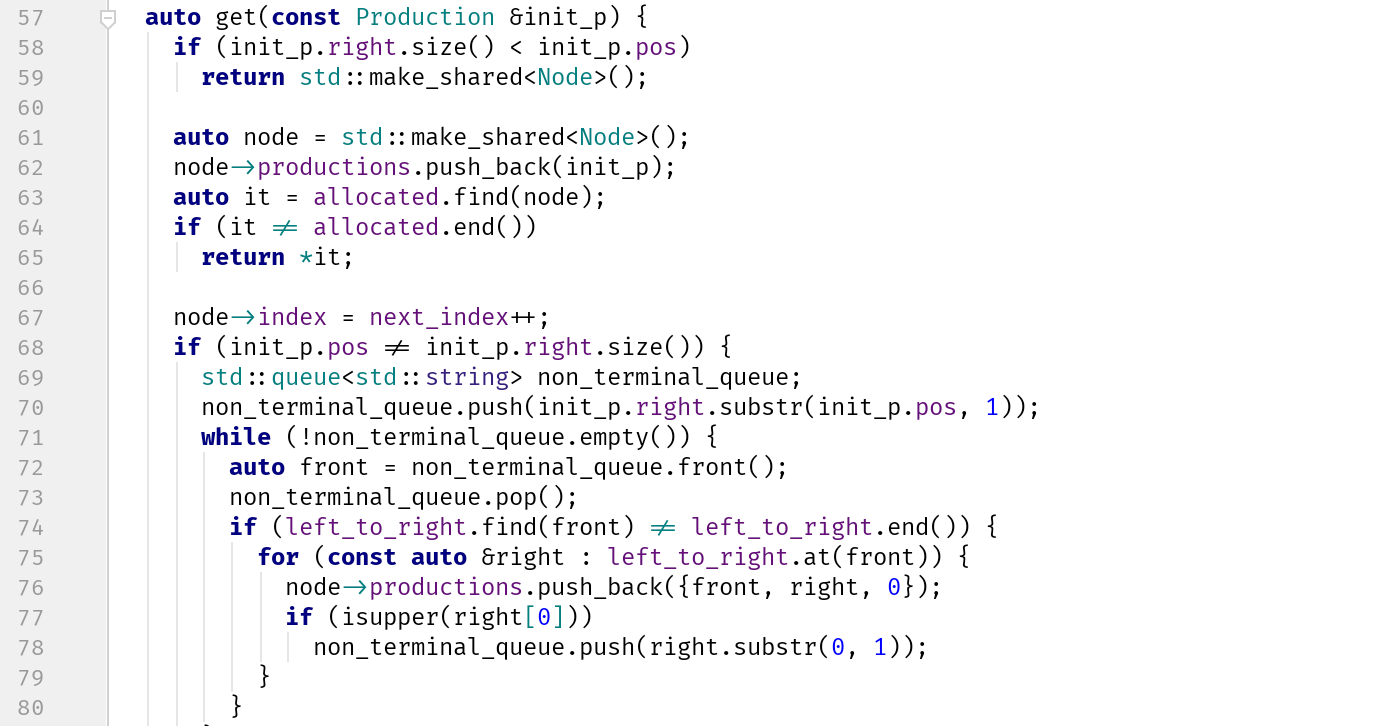


FSM的每个结点由结点序号、产生式集合和结点可转移的结点的集合组成。结点类重载了<以备后续使用。由于实现的原因，这里忽略了两个结点仅序号不同而产生的区别。

下面定义NodeAllocator类用于产生结点。



需要该类的原因是FSM中会出现两个不同结点转移到同一结点的情况。通过一个类统一产生结点，则若某个结点已经存在，只需返回它的指针即可。每次通过get函数传入某个产生式的“点”移动一格产生的产生式，它就能返回新的产生式所在的结点。下面是get函数的具体实现。





从实现中可以看到，若传入产生式的“点”之后是非终结符，那么会继续将以该非终结符为左部的产生式加入结点。

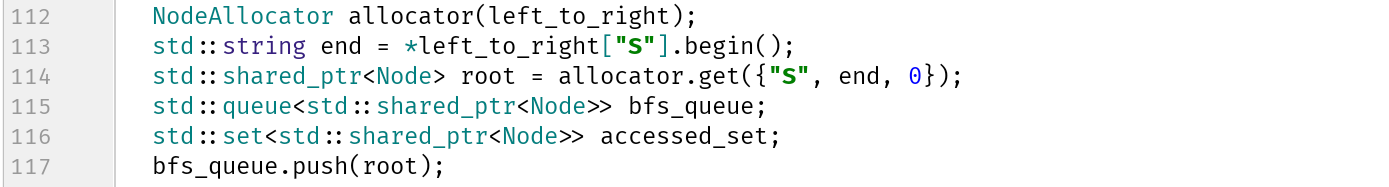
最后，就可以实现主程序。

在主程序中，首先读入产生式，构造所需要的数据结构。



production\_to\_index用于规约时根据产生式查找产生式序号，left\_to\_right以map的形式保存所有产生式。

接下来查找左部为S的产生式构造第一个结点，进行BFS的初始化。



接下来就是进行BFS，并在BFS过程中输出Action和Goto表。

