# 形式语言与自动机实验报告

实验名称 设计上下文无关文法的变换算法

班 级 2020211302

小组成员 张清戌 2020211259 文档

赵维岗 2020211261 测试数据

鄭毓恒 2020211262 代码

丁 易 2020211275 检查及改进

### 实验环境介绍

操作系统：Windows10

编程语言：C++

编程工具：VS2022

### 二.程序设计思路

**1.消除生成式**

(1)  利用算法1, 找出N’= {A | A∈N且A=>+ ε}

(找出能推导出ε的所有非终结符A)

(2)  用以下两步组成P1

① 如果生成式A→β0 C1β1 C2…Cnβn∈P n≥0 且每个Ck (1≤k≤n) 均在N’内

而对于0≤j≤n, 没有βj在N’内 (βi也可能是终结符)

则P1应加入 A→β0Y1β1Y2β2…Ynβn

其中Yk是Ck或ε (即所有的可能组合)

但是A → ε不加入P1

② 如果S∈N’,则P1中应加入以下生成式

S1→ε|S S1是新的起始符

N1=N∪{S1}

如果S∉N’, 则N1=N, S1=S

由此得出G1=( N1,T, P1,S1)

**2.消除单生成式**

(1) 对每个A∈N, 构造非终结符集NA={B|A=>\* B}

( NA是可由A推出的单生成式中的非终结符集)。

构造方法分三步:

① N0={A}

② N’={C | 如果B→C∈P且B∈ N0}∪N0

③ 若N’≠ N0,则N0=N’,转向② (继续迭代)

否则NA = N’, 转向(2). (已得到了完整的NA)

(2) 构造P1:

如果B→α∈P且不是单生成式, 则对于B∈NA的所有A,把A→α加入到P1中

(即对每个B ∈ NA (意味着A=>+ B)的非单生成式B→α∈P, 直接将α与NA的A连接, 构成新产生式A→α加入到P1中)

(3) 得到G1=( N1, T1, P1, S)

**3.消除无用符号**

找出有用非终结符

(1) N 0 = Φ (赋为Φ) N 0为有用的非终结符集

(2) N’ = {A | A→ω且ω∈T\*} N’为非终结符集合

(3) 如果N 0≠N’ 则转(4),否则转(6)

(4) N 0=N’

(5) N’= N 0∪{ A | A→α且α∈(T∪N 0)\* }, 转(3)

(6) N 1 = N’

找出可达符号

(1) N 0 = {S}

(2) N’= {x | A∈N 0 且A→αxβ}∪ N 0

(N’为有用符号集合)

(3) 如果N 0≠N’转(4), 否则转(5)

(4) N 0=N’; 转(2) (继续迭代下去)

(5) N 1 = N’ ∩ N

T1=N’∩T

消去非生成符号及不可达符号

**三.高层数据信息定义**

**数据结构定义**

//生成式数据结构

typedef struct{

char left; //左侧非终结符

vector<string> right; //右侧生成结果

}Generate;

**类定义**

//文法类

class Grammar

{

public:

void removeEGenerate(); //消除ε产生式

void removeSingleGenerate();//消除单产生式

void removeUselessSymbol(); //消除无用符号

void input(); //输入文法

void output() const; //输出文法

private:

void algorithm1(string& set, const string& dest) const; //算法1，将可生成集合dest的符号存在set中

void algorithm2(); //算法2，消除不可达符号

int isBelong(char c, const string& set) const; //判断c是否属于集合set，是则返回1，否则返回0

int isBelong(const string& s, const string& set) const; //判断集合s是否属于集合set，是则返回1，否则返回0

void andOfSet(const string& s1, const string& s2, string& result) const; //集合s1和集合s2作与运算，结果存在result

void deleteGenerate(); //删除含有不存在符号的生成式

void judgeToE(const string& s, const string& toE, vector<int>& loc); //判断串s中哪些字符属于可致空符号集合toE，下标存在loc

void addGenerate(vector<string>& right, const string& s, const vector<int>& loc, int startLoc); //消除ε产生式时，用于递归产生新的生成式

int getSymbolNum(char s); //获取符号s的序号

void getSingleGenerate(string& set, char s); //递归找出s的单元偶对，存在set中

public:

char start; //起始符

string n, t; //非终结符集合、终结符集合

vector<Generate> p; //生成式集合

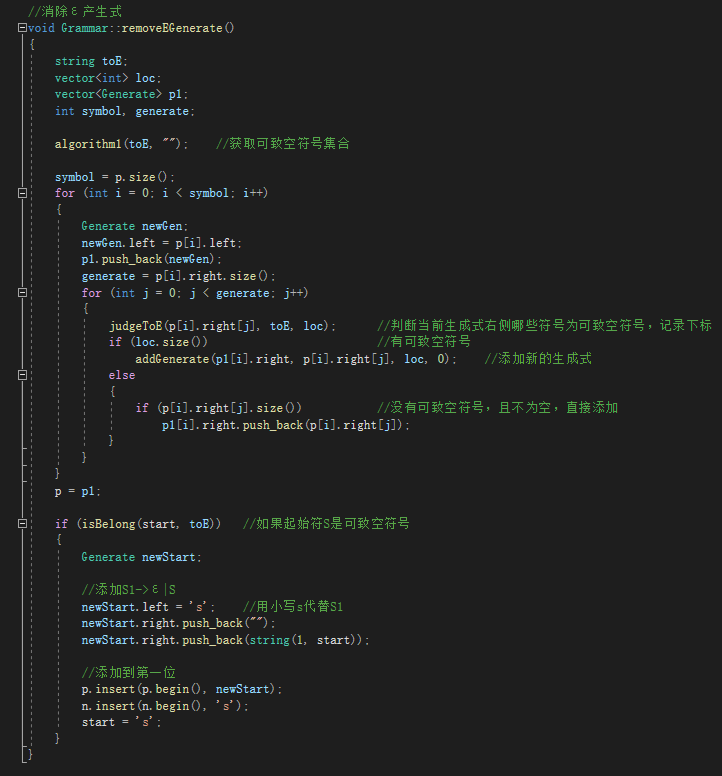
};

### 四.函数功能说明

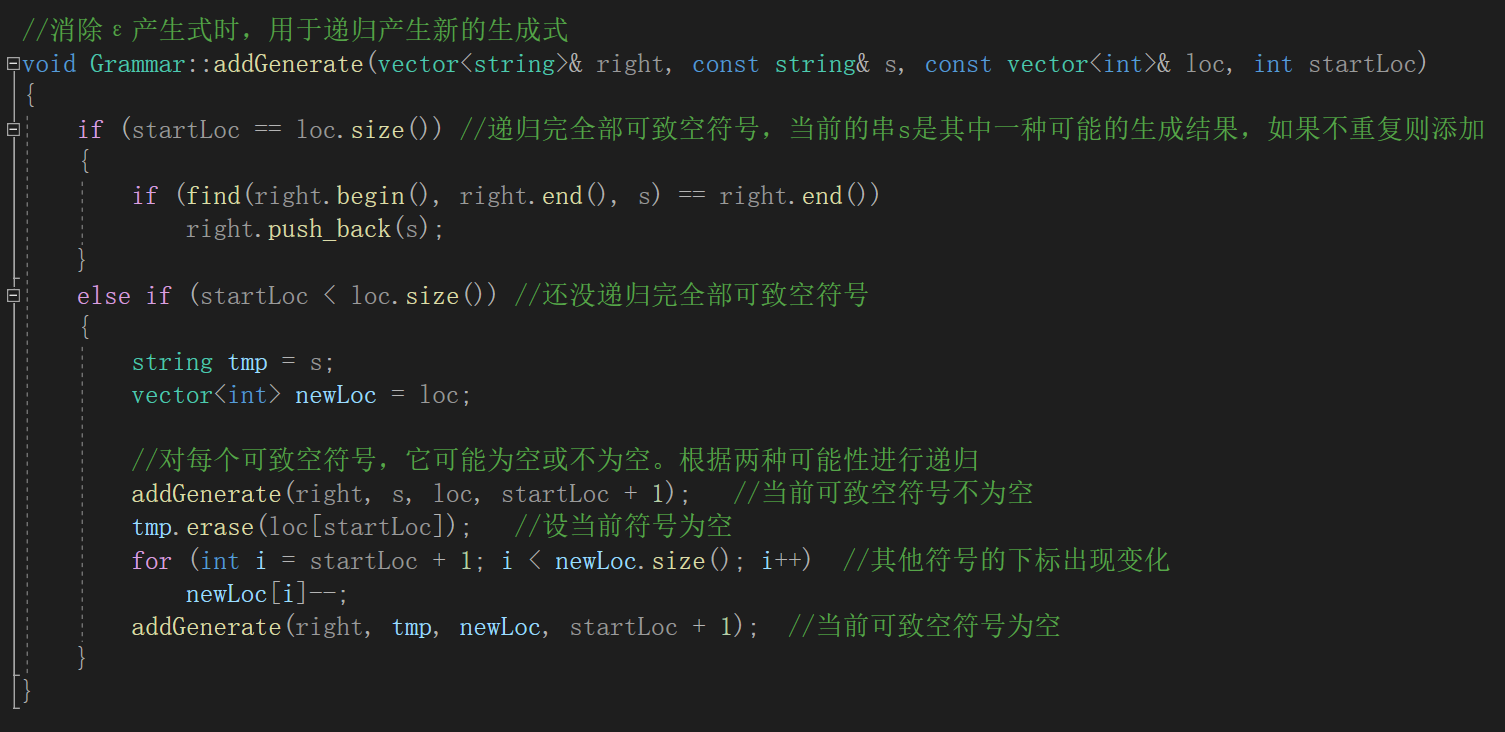
|  |  |
| --- | --- |
| 函数名 | 功能简述 |
| void Grammar::addGenerate(vector<string>& right, const string& s, const vector<int>& loc, int startLoc) | 消除ε产生式时，用于递归产生新的生成式 |
| void Grammar::algorithm1(string& set, const string& dest) const | 将可生成集合dest的符号存在set中 |
| void Grammar::algorithm2() | 消除不可达符号 |
| void Grammar::andOfSet(const string& s1, const string& s2, string& result) const | 集合s1和集合s2作与运算，结果存在result |
| void Grammar::deleteGenerate() | 删除含有不存在符号的生成式 |
| void Grammar::getSingleGenerate(string& set, char s) | 递归找出s的单元偶对，存在set中 |
| int Grammar::getSymbolNum(char s) | 获取符号s的序号 |
| void Grammar::input() | 输入文法 |
| int Grammar::isBelong(char c, const string& set) const | 判断c是否属于集合set，是则返回1，否则返回0 |
| int Grammar::isBelong(const string& s, const string& set) const | 判断集合s是否属于集合set，是则返回1，否则返回0 |
| void Grammar::judgeToE(const string& s, const string& toE, vector<int>& loc) | 判断串s中哪些字符属于可致空符号集合toE，下标存在loc |
| void Grammar::output() const | 输出文法 |
| void Grammar::removeEGenerate() | 消除ε产生式 |
| void Grammar::removeSingleGenerate() | 消除单产生式 |
| void Grammar::removeUselessSymbol() | 消除无用符号 |

### 五.核心算法

**1. 消除生成式**

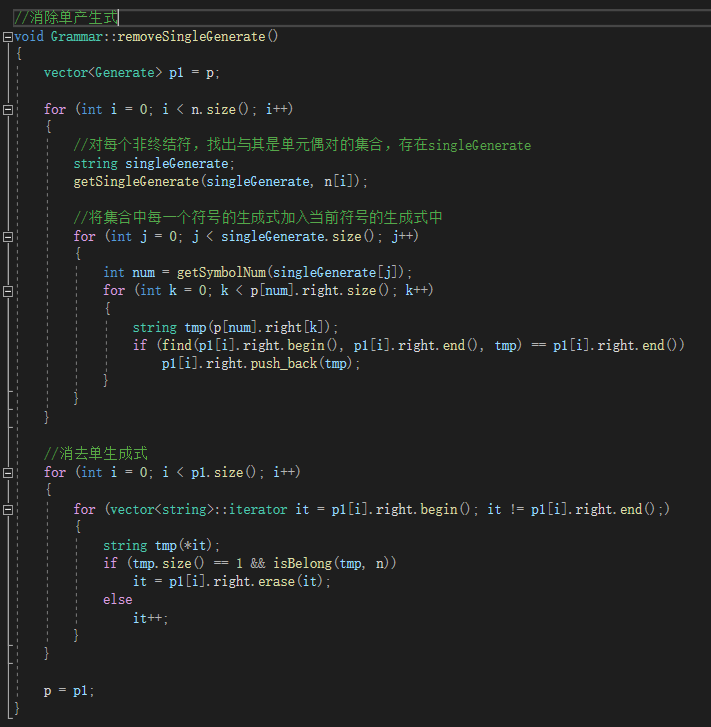


1. 调用algorithm1()函数借助算法1，获取可致空符号集
2. 逐个遍历P中的生成式，调用judgeToE()函数判断当前生成式右侧哪些符号为可致空符号，下标存储在loc向量当中。若有可致空符号，调用addGenerate()函数则添加新的生成式；若没有可致空符号，且不为空，则直接添加符号
3. 若起始符为可致空符号，则添加S1->|S到首位

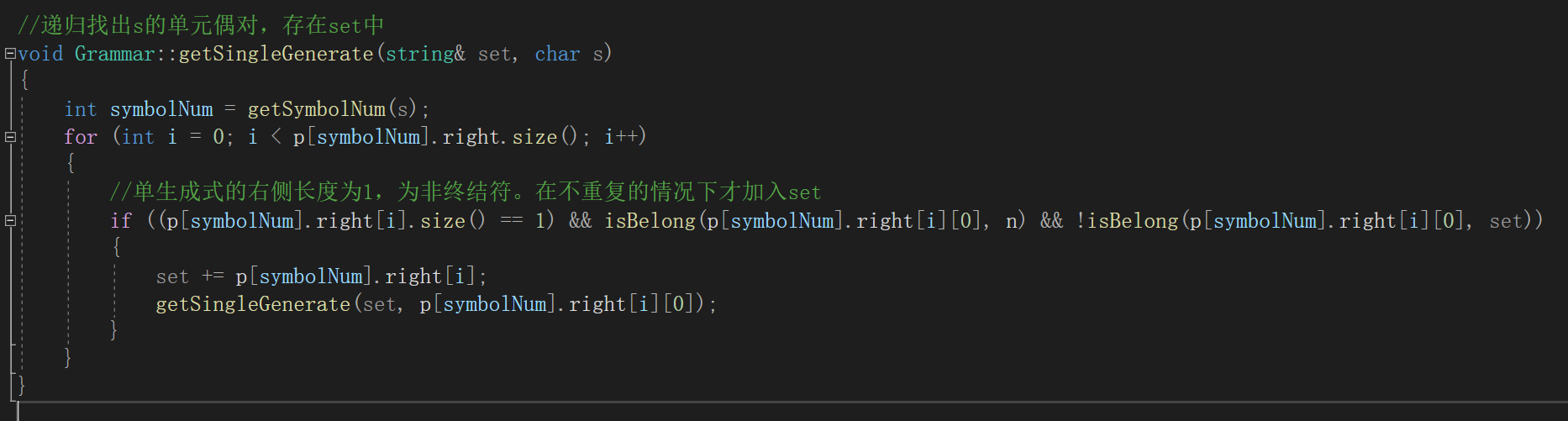


函数addGenerate()使用2路递归产生新生成式，当前可致空符号不为空继续下一次递归，当前可致空符号为空再继续第二次递归。一次递归直到startLoc等于loc向量的大小结束，也就是经过了所有的可致空符号。已递归完，s本身即为可能的结果，如果不重复则添加进right向量。未递归完继续递归。

**2.消除单生成式**

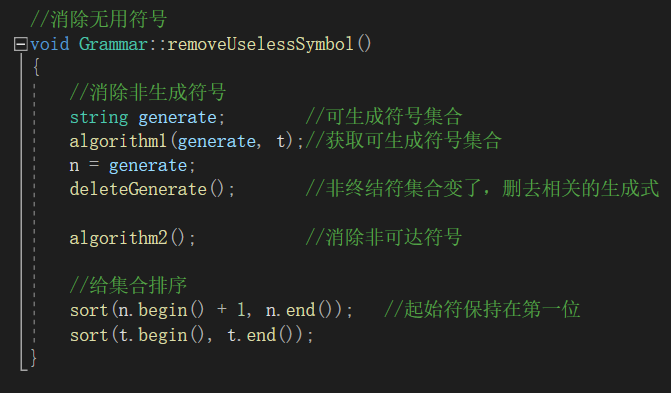


1. 对每个非终结符，找出与其是单元偶对的集合（调用getSingleGenerate()递归寻找，储存在singleGenerate中）
2. 遍历这个集合，集合中每一个符号的生成式如果不重复，加入当前符号的生成式中
3. 遍历包括新加入的生成式在内的所有生成式，消去单生成式



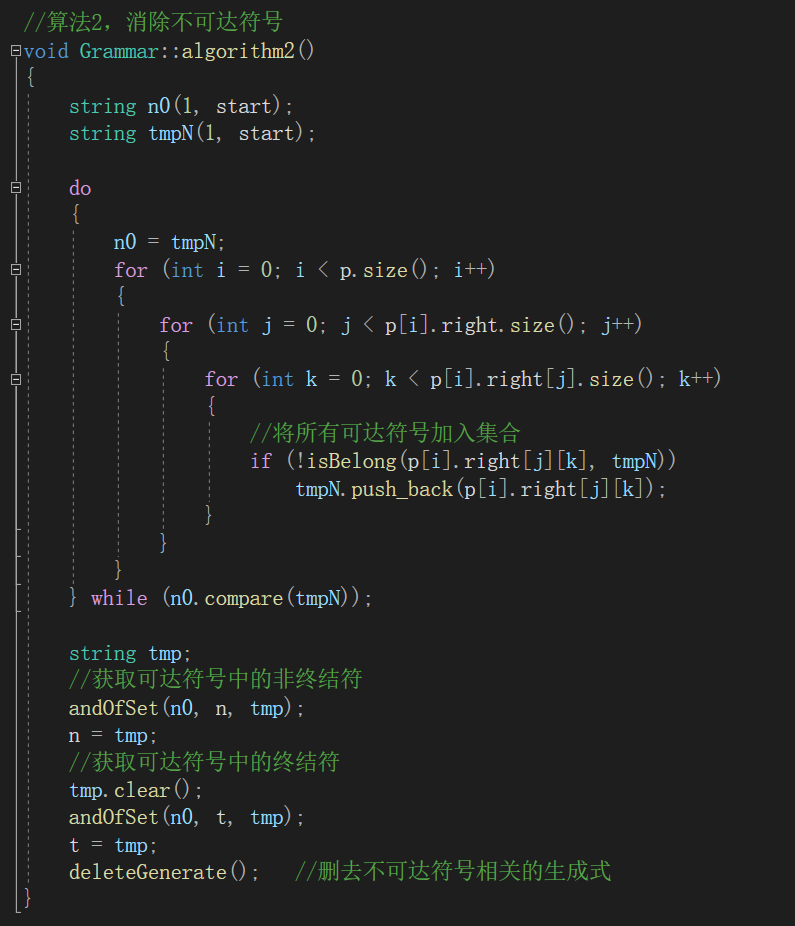
函数getSingleGenerate()有两个参数，string& set是第一次调用时的与非终结符s为单元偶对的符号集合，char s是当前递归到的符号。遍历s的生成式，如果找到单生成式且生成结果不在集合set中，加入set，用该符号进行下一次递归。当递归全部结束后，set中保存了所有和第一次调用时的s为单元偶对的非终结符的集合。

**3.消除无用符号**



1. 调用algorithm1()获取可生成符号集合
2. 删除非终结符集合变化相关的生成式
3. 调用algorithm2()删除非可达符号
4. 消除过程改变了符号顺序，进行排序





函数algorithm1()和algorithm2()就是书里的算法1和算法2的代码实现。

### 六.程序的输入输出

**测试用例**

S—>a|bA|B|ccD|ε

A—>abB|ε

B—>aA|D

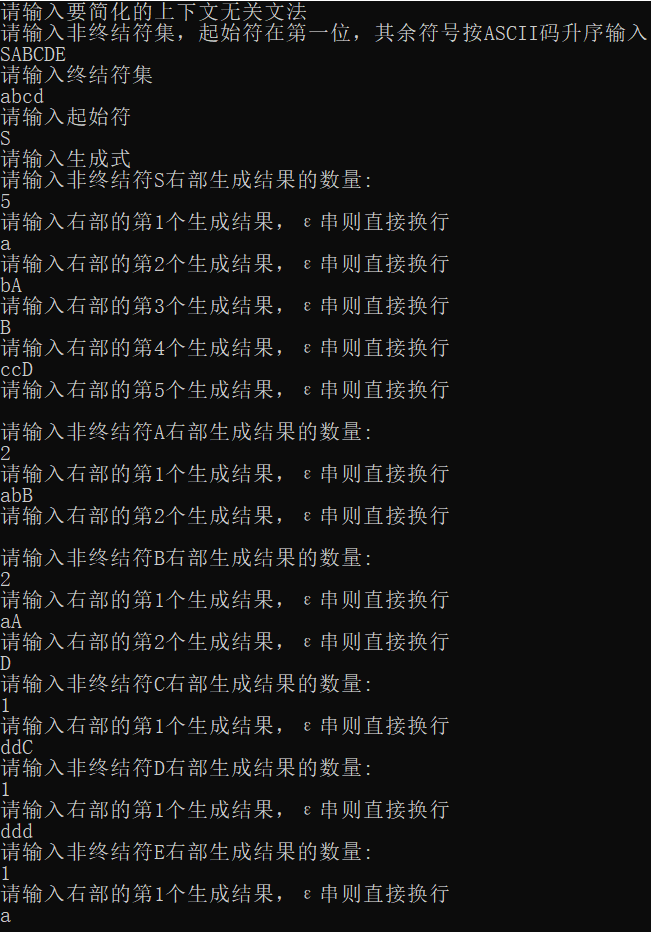
C—>ddC

D—>ddd

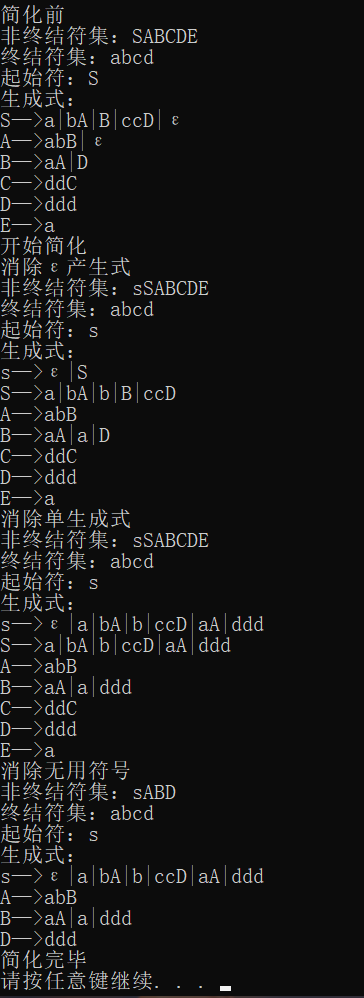
E—>a

（格式见图）

**程序的输入**



**程序的输出**



程序输出了简化过程每一步之后的结果，最后的结果输出正确。