華南師範大學

本科学生实验（实践）报告

院　　系：计 算 机 学 院

实验课程：编译原理

实验项目：SLR(1)分析生成器

指导老师：黄煜廉

开课时间：2023 ～ 2024 年度第 1 学期

专　　业：计算机科学与技术

班　　级：计科 1 班

学　　生：李达良

学　　号：20203231004

华南师范大学教务处

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

一、实验题目

SLR(1)分析生成器

二、实验内容

(1)要提供一个文法输入编辑界面，让用户输入文法规则（可保存、打开存有文法规则的文件）

(2)求出文法各非终结符号的 first 集合与 follow 集合，并提供窗口以便用户可以查看这些集合结果。【可以采用表格的形式呈现】

(3)需要提供窗口以便用户可以查看文法对应的 LR(0)DFA 图。（可以用画图的方式呈现，也可用表格方式呈现该图点与边数据）

(4)要提供窗口以便用户可以查看该文法是否为 SLR(1)文法。（如果非 SLR(1)文法，可查看其原因）

(5)需要提供窗口以便用户可以查看文法对应的 SLR(1)分析表。（如果该文法为 SLR(1)文法时）【SLR(1)分析表采用表格的形式呈现】

(6)应该书写完善的软件文档

(7)应用程序应为 Windows 界面。

三、实验目的

设计一个应用软件，以实现 SLR(1)分析生成器。

四、实验文档

（1）程序界面设计

通过 QT 实现 UI 的设计，左上方给予一个文法输入框，输入框上方有两个按钮，分别是打开和保存，方便使用者的输入和保存文法。左下方是 First 集合和 Follow 集合求解，当输入了文法后，点击对应区域的求解按钮，就会显示文法中非终结符的 First 和 Follow 集合。右方是 LR(0)和 SLR(1)的求解区域，点击对应按钮，即可获得相应的答案，同时对于 LR(0)来说会有生成提示，来提示用户程序可能会对文法进行增广处理，对于 SLR(1)来说，当文法不符合 SLR(1)文法的时候，会提示不符合的原因。

华 南 师 范 大 学 实 验 报 告

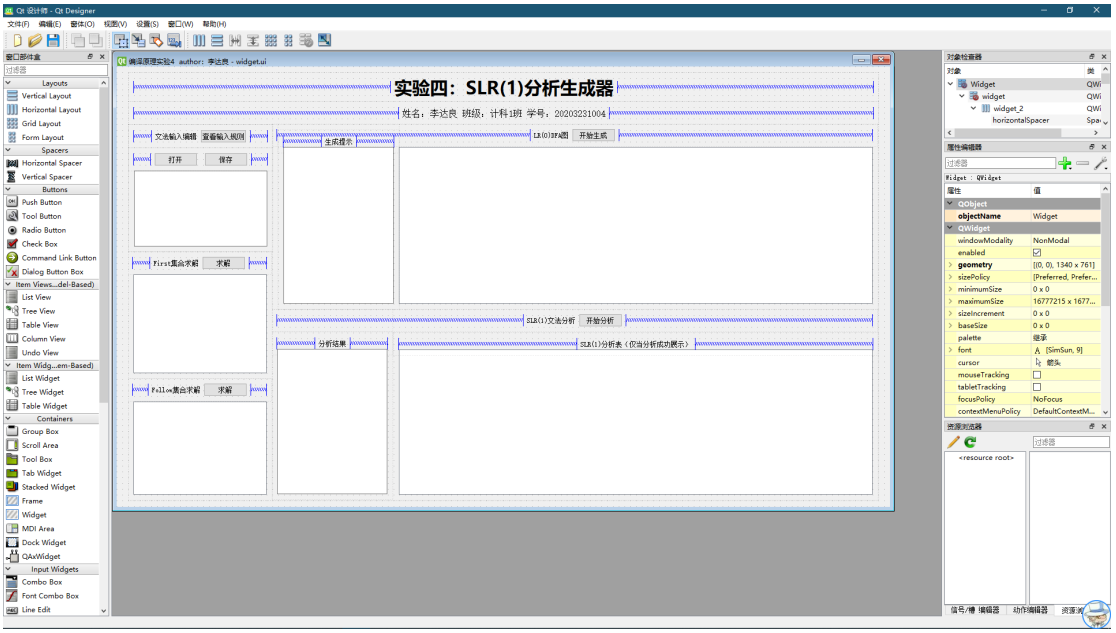
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分



（2）程序逻辑设计

对于本程序，我们主要分为四大块：

①文法处理

②First 集合求解

③Follow 集合求解

④LR(0)生成

⑤SLR(1)分析表生成

2.1 文法处理

首先我们定义以下数据结构：

```
// 全局文法变量
string grammarStr;

// 结构化后的文法 map
unordered_map<char, set<string>> grammarMap;

// 文法 unit（用于 LR0）
struct grammarUnit
{
    int gid;
    char left;
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
    string right;
    grammarUnit(char l, string r)
    {
        left = l;
        right = r;
    }
};

// 文法数组（用于 LR0）
deque<grammarUnit> grammarDeque;

// LR0 结果提示字符串
QString LR0Result;

// 开始符号
char startSymbol;

// 增广后开始符号
char trueStartSymbol;

// 文法查找下标
map<pair<char, string>, int> grammarToInt;
```

同时我们有一些公用函数，用于识别终结符和非终结符：

```
/*************  公用函数 ****************/
// 非终结符
bool isBigAlpha(char c)
{
    return c >= 'A' && c <= 'Z';
}

// 终结符
bool isSmallAlpha(char c)
{
    return !(c >= 'A' && c <= 'Z') && c != '@';
}
// 清空全局变量函数
void reset();
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 **学　号** 20203231004

专　业 计算机科学与技术 **年级、班级** 2021 级 1 班

课程名称 编译原理 **实验项目** SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 **实验评分**

首先分割输入的文法字符串为每一行。对每一行的规则进行解析，提取非终结符、产生式右侧。将文法结构化到 grammarMap 中，并为 LR0 自动机做准备。

进行增广处理，如果开始符号有多个产生式，则增广文法。对文法进行编号，并构建 LR0 项目集规范族。

增广处理：如果开始符号的产生式多于一个，说明需要增广，使用 '^' 作为增广后的字母。在 grammarDeque 中加入增广的产生式，并更新 LR0Result 字符串，提示用户程序进行了增广处理。

```
// 处理文法
void handleGrammar()
{
    vector<string> lines;
    istringstream iss(grammarStr);
    string line;

    // 防止中间有换行符
    while (getline(iss, line))
    {
        if (!line.empty())
        {
            lines.push_back(line);
        }
    }

    for (const auto& rule : lines)
    {
        istringstream ruleStream(rule);
        char nonTerminal;
        ruleStream >> nonTerminal;  // 读取非终结符

        // 验证非终结符的格式
        if (!isBigAlpha(nonTerminal))
        {
            QMessageBox::critical(nullptr, "Error", "文法开头必须是非终结符（大写字母）!");
            continue;
        }
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

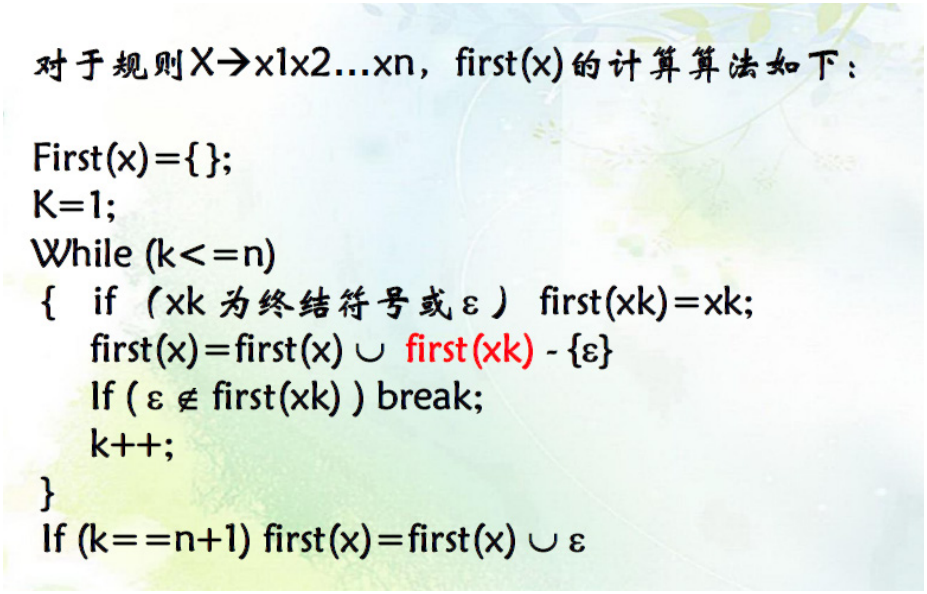
实验指导老师 黄煜廉 实验评分

```
        // 跳过箭头及空格
        ruleStream.ignore(numeric_limits<streamsize>::max(), '-');
        ruleStream.ignore(numeric_limits<streamsize>::max(), '>');

        string rightHandSide;
        ruleStream >> rightHandSide;  // 获取产生式右侧

        // 如果是第一条规则，则认为是开始符号
        if (grammarMap.empty())
        {
            startSymbol = nonTerminal;
            trueStartSymbol = startSymbol;
        }

        // 将文法结构化
        grammarMap[nonTerminal].insert(rightHandSide);

        // 为 LR0 做准备
        grammarDeque.push_back(grammarUnit(nonTerminal, rightHandSide));
    }

    // 增广处理
    if (grammarMap[startSymbol].size() > 1)
    {
        // 如果开始符号多于 2 个，说明需要增广，为了避免出现字母重复，采用^作为增广后的字母，后期输出特殊处理
        grammarDeque.push_front(grammarUnit('^', string(1, startSymbol)));
        LR0Result += QString::fromStdString("进行了增广处理\n");
        trueStartSymbol = '^';
    }

    // 开始编号
    int gid = 0;
    for (auto& g : grammarDeque)
    {
        g.gid = gid++;
        LR0Result += QString::number(g.gid) + QString::fromStdString(":")
            + QString::fromStdString(g.left == '^' ? "E\'" : string(1, g.left)) +
```

```
QString::fromStdString("->")
                + QString::fromStdString(g.right) + "\n";
            // 存入 map 中
            grammarToInt[make_pair(g.left, g.right)] = g.gid;
        }

    }
```

2.2 first 集合求解

对于 first 集合的求解，我们采用以下算法：



因此，我们设置以下数据结构：

```
// First 集合单元
struct firstUnit
{
    set<char> s;
    bool isEpsilon = false;
};

// 非终结符的 First 集合
map<char, firstUnit> firstSets;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

同时根据伪代码算法，我们可以先创建一个递归算法：遍历文法中的每个非终结符，对于每个非终结符，遍历其产生式。对于每个产生式，找到其每个符号的 First 集合，并将其加入到该非终结符的 First 集合中。如果发现某个符号的 First 集合包含空串，并且该符号后还有其他符号，则继续处理下一个符号。如果某个产生式的所有符号的 First 集合都包含空串，则将空串加入该非终结符的 First 集合中。检查原始 First 集合的大小和是否包含空串，如果发生变化，则将 flag 置为 true。

```
// 计算 First 集合
bool calculateFirstSets()
{
    bool flag = false;
    for (auto& grammar : grammarMap)
    {
        char nonTerminal = grammar.first;
        // 保存当前 First 集合的大小，用于检查是否有变化
        size_t originalSize = firstSets[nonTerminal].s.size();
        bool originalE = firstSets[nonTerminal].isEpsilon;
        for (auto& g : grammar.second)
        {
            int k = 0;
            while (k <= g.size() - 1)
            {
                set<char> first_k;
                if (g[k] == '@')
                {
                    k++;
                    continue;
                }
                else if (isSmallAlpha(g[k]))
                {
                    first_k.insert(g[k]);
                }
                else
                {
                    first_k = firstSets[g[k]].s;
                }
                firstSets[nonTerminal].s.insert(first_k.begin(),
first_k.end());
                // 如果是终结符或者没有空串在非终结符中，直接跳出
```

```
                if (isSmallAlpha(g[k]) || !firstSets[g[k]].isEpsilon)
                {
                    break;
                }
                k++;
            }
            if (k == g.size())
            {
                firstSets[nonTerminal].isEpsilon = true;
            }
        }
        // 看原始大小和是否变化 epsilon，如果变化说明还得重新再来一次
        if (originalSize != firstSets[nonTerminal].s.size() || originalE != firstSets[nonTerminal].isEpsilon)
        {
            flag = true;
        }
    }
    return flag;
}
```

同时我们给一个入口，getFirstSets() 函数：

使用循环调用 calculateFirstSets() 直到不再有变化。在每次迭代中，检查是否有非终结符的 First 集合发生了变化，如果有变化则继续迭代。通过这种方式，确保计算到的 First 集合是不再变化的。

```
void getFirstSets()
{
    // 不停迭代，直到 First 集合不再变化
    bool flag = false;
    do
    {
        flag = calculateFirstSets();
    } while (flag);
}
```

2.3 Follow 集合求解

针对 Follow 集合求解，我们采用以下算法：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

我们可以定义以下数据结构：

```
// Follow 集合单元
struct followUnit
{
    set<char> s;
};

// 非终结符的 Follow 集合
map<char, followUnit> followSets;
```

然后我们编写一个递归函数，遍历文法中的每个非终结符，对于每个产生式，遍历产生式中的每个符号。如果符号是终结符或空串，则跳过。如果符号是非终结符，根据产生式的位置和后续符号计算 Follow 集合。通过两个情况（A -> αB 和 A -> αBβ）来处理 Follow 集合的计算。更新 Follow 集合，并检查是否发生变化。

```
// 添加 Follow 集合
void addToFollow(char nonTerminal, const set<char>& elements)
{
    followSets[nonTerminal].s.insert(elements.begin(), elements.end());
}

// 计算 Follow 集合
bool calculateFollowSets()
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
{
    bool flag = false;
    for (auto& grammar : grammarMap)
    {
        char nonTerminal = grammar.first;

        for (auto& g : grammar.second)
        {
            for (int i = 0; i < g.size(); ++i)
            {
                if (isSmallAlpha(g[i]) || g[i] == '@')
                {
                    continue;  // 跳过终结符
                }

                set<char> follow_k;
                size_t originalSize = followSets[g[i]].s.size();

                if (i == g.size() - 1)
                {
                    // Case A: A -> αB, add Follow(A) to Follow(B)
                    follow_k.insert(followSets[nonTerminal].s.begin(),
followSets[nonTerminal].s.end());
                }
                else
                {
                    // Case B: A -> αBβ
                    int j = i + 1;
                    while (j < g.size())
                    {
                        if (isSmallAlpha(g[j]))
                        {   // 终结符直接加入并跳出
                            follow_k.insert(g[j]);
                            break;
                        }
                        else
                        {   // 非终结符加入 first 集合
                            set<char> first_beta = firstSets[g[j]].s;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
                                follow_k.insert(first_beta.begin(),
first_beta.end());

                                // 如果没有空串在 first 集合中，停止。
                                if (!firstSets[g[j]].isEpsilon)
                                {
                                    break;
                                }

                                ++j;
                            }

                        }

                        // If β is ε or β is all nullable, add Follow(A) to
Follow(B)
                        if (j == g.size())
                        {
                            follow_k.insert(followSets[nonTerminal].s.begin(),
followSets[nonTerminal].s.end());
                        }
                    }

                    addToFollow(g[i], follow_k);
                    // 检查是否变化
                    if (originalSize != followSets[g[i]].s.size())
                    {
                        flag = true;
                    }
                }
            }


        }

        return flag;
    }
```

最后编写一个入口，初始化开始符号的 Follow 集合为 { '$' }。使用循环调用 calculateFollowSets() 直到不再有变化。在每次迭代中，检查是否有非终结符的 Follow 集合发生了变化，如果有变化则继续迭代。通过这种方式，确保计算到的 Follow 集合是不再变化的。

```
void getFollowSets()
{
    // 开始符号加入$
    addToFollow(startSymbol, { '$' });

    // 不停迭代，直到 Follow 集合不再变化
    bool flag = false;
    do
    {
        flag = calculateFollowSets();
    } while (flag);
}
```

2.4 LR(0)生成

对于 LR(0)生成，我们首先先对第一条文法进行生成 LR0 状态，并且因为增广文法的存在，一定只会有一个入口，即一条文法。

但在生成 LR0 状态之前，我们得先定义以下数据结构和一些常用函数：

```
// 状态编号
int scnt = 0;
// 项目编号
int ccnt = 0;

// DFA 表每一项项目的结构
struct dfaCell
{
    int cellid; // 这一项的编号，便于后续判断状态相同
    int gid; // 文法编号
    int index = 0; // .在第几位，如 i=3, xxx.x，i=0, .xxxx, i=4, xxxx
};

// 用于通过编号快速找到对应结构
vector<dfaCell> dfaCellVector;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
struct nextStateUnit
{
    char c; // 通过什么字符进入这个状态
    int sid; // 下一个状态 id 是什么
};

// DFA 表状态
struct dfaState
{
    int sid; // 状态 id
    vector<int> originV;    // 未闭包前的 cell
    vector<int> cellV;  // 存储这个状态的 cellid
    bool isEnd = false; // 是否为规约状态
    vector<nextStateUnit> nextStateVector; // 下一个状态集合
    set<char> right_VNs; // 判断是否已经处理过这个非终结符
};

// 用于通过编号快速找到对应结构
vector<dfaState> dfaStateVector;

// 非终结符集合
set<char> VN;
// 终结符集合
set<char> VT;

// 判断是不是新结构
int isNewCell(int gid, int index)
{
    for (const dfaCell& cell : dfaCellVector)
    {
        // 检查 dfaCellVector 中是否存在相同的 gid 和 index 的 dfaCell
        if (cell.gid == gid && cell.index == index)
        {
            return cell.cellid; // 不是新结构
        }
    }
    return -1; // 是新结构
}
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
// 判断是不是新状态
int isNewState(const vector<int>& cellIds)
{
    for (const dfaState& state : dfaStateVector)
    {
        // 检查状态中的 originV 是否相同
        if (state.originV.size() == cellIds.size() &&
            equal(state.originV.begin(),                state.originV.end(),
cellIds.begin()))
        {
            return state.sid; // 不是新状态
        }
    }

    return -1; // 是新状态
}

// DFS 标记数组
set<int> visitedStates;
```

然后我们就可以编写一个生成第一个状态的函数：

```
// 创建 LR0 的初始状态
void createFirstState()
{
    // 由于增广，一定会只有一个入口
    dfaState zero = dfaState();
    zero.sid = scnt++; // 给他一个 id
    dfaStateVector.push_back(zero); // 放入数组中

    // 添加初始的 LR0 项，即 E' -> .S
    dfaCell startCell;
    startCell.gid = 0; // 这里假设增广文法的编号为 0
    startCell.index = 0;
    startCell.cellid = ccnt++;

    dfaCellVector.push_back(startCell);
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
    // 把初始 LR0 项放入初始状态
    dfaStateVector[0].cellV.push_back(startCell.cellid);
    dfaStateVector[0].originV.push_back(startCell.cellid);
}
```

得到第一个状态后，我们可以编写递归生成状态的函数了，主要步骤类似于 DFS 遍历，首先判断我们之前是否到达过该状态，如果到达过，说明我们不需要再对该状态递归下去，防止死循环。

然后，我们要对当前走到的状态求闭包：我们要检查点号是否在产生式末尾或者产生式是否是空串，如果是的话，说明是归约文法，不是的话，检查下一个符号是终结符还是非终结符，非终结符的话，我们需要将对应的新项目加入到一个该状态中，直到没有新项目产生，这时闭包求解完毕。

求解完毕之后，重新遍历该状态所有文法，生成新状态，即每个文法往前走一步，但不能直接存入新状态的 dfaStateVector，要检验是否和 dfaStateVector 中某个状态是一样的，如果是一样的话，本状态在该项目中往前走一步应该是回到 dfaStateVector 中某个状态上。

最后我们对下一个状态进行递归 DFS。

代码如下：

```
// 生成 LR0 状态
void generateLR0State(int stateId)
{
    // DFS,如果走过就不走了
    if (visitedStates.count(stateId) > 0) {
        return;
    }

    // 标记走过了
    visitedStates.insert(stateId);

    //if (dfaStateVector[stateId].isEnd)
    //{
    //    return;
    //}

    qDebug() << stateId << endl;

    // 求闭包
    for (int i = 0; i < dfaStateVector[stateId].cellV.size(); ++i)
    {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
            dfaCell&                        currentCell                        =
dfaCellVector[dfaStateVector[stateId].cellV[i]];

            qDebug()       <<       grammarDeque[currentCell.gid].left       <<
QString::fromStdString("->")                                                   <<
QString::fromStdString( grammarDeque[currentCell.gid].right) << endl;

            qDebug() << "current index:" << currentCell.index << endl;

            // 如果点号在产生式末尾或者空串，则跳过（LR0 不需要结束）
            if                        (currentCell.index                       ==
grammarDeque[currentCell.gid].right.length()                                   ||
grammarDeque[currentCell.gid].right == "@")
            {
                dfaStateVector[stateId].isEnd = true;
                continue;
            }

            char                            nextSymbol                         =
grammarDeque[currentCell.gid].right[currentCell.index];


            // 如果 nextSymbol 是非终结符，则将新项添加到状态中
            if                      (isBigAlpha(nextSymbol)                    &&
dfaStateVector[stateId].right_VNs.find(nextSymbol)                             ==
dfaStateVector[stateId].right_VNs.end())
            {
                dfaStateVector[stateId].right_VNs.insert(nextSymbol);
                for (auto& grammar : grammarMap[nextSymbol])
                {
                    // 获取通过 nextSymbol 转移的新 LR0 项
                    dfaCell nextCell = dfaCell();
                    nextCell.gid = grammarToInt[make_pair(nextSymbol,grammar)];
                    nextCell.index = 0;
                    int nextcellid = isNewCell(nextCell.gid, nextCell.index);
                    if (nextcellid == -1)
                    {
                        nextCell.cellid = ccnt++;
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
                    dfaCellVector.push_back(nextCell);

dfaStateVector[stateId].cellV.push_back(nextCell.cellid);
                }
                else dfaStateVector[stateId].cellV.push_back(nextcellid);
            }

        }
    }

    // 暂存新状态
    map<char,dfaState> tempSave;
    // 生成新状态，但还不能直接存到 dfaStateVector 中，我们要校验他是否和之前的状态一样
    for (int i = 0; i < dfaStateVector[stateId].cellV.size(); ++i)
    {
        dfaCell& currentCell = dfaCellVector[dfaStateVector[stateId].cellV[i]];

        qDebug() << grammarDeque[currentCell.gid].left << QString::fromStdString("->") << QString::fromStdString(grammarDeque[currentCell.gid].right) << endl;

        qDebug() << "current index:" << currentCell.index << endl;

        // 如果点号在产生式末尾，则跳过（LR0 不需要结束）
        if (currentCell.index == grammarDeque[currentCell.gid].right.length() || grammarDeque[currentCell.gid].right == "@")
        {
            continue;
        }

        // 下一个字符
        char nextSymbol = grammarDeque[currentCell.gid].right[currentCell.index];

        // 创建下一个状态（临时的）
```

```
            dfaState& nextState = tempSave[nextSymbol];
            dfaCell nextStateCell = dfaCell();
            nextStateCell.gid = currentCell.gid;
            nextStateCell.index = currentCell.index + 1;

            // 看看里面的项目是否有重复的，如果重复拿之前的就好，不重复生成
            int nextStateCellid = isNewCell(nextStateCell.gid,
nextStateCell.index);
            if (nextStateCellid == -1)
            {
                nextStateCell.cellid = ccnt++;
                dfaCellVector.push_back(nextStateCell);
            }
            else nextStateCell.cellid = nextStateCellid;
            nextState.cellV.push_back(nextStateCell.cellid);
            nextState.originV.push_back(nextStateCell.cellid);

            // 收集一下，方便后面画表
            if (isBigAlpha(nextSymbol))
            {
                VN.insert(nextSymbol);
            }
            else if (isSmallAlpha(nextSymbol))
            {
                VT.insert(nextSymbol);
            }
        }

        // 校验状态是否有重复的
        for (auto& t : tempSave)
        {
            dfaState nextState = dfaState();
            int newStateId = isNewState(t.second.originV);
            // 不重复就新开一个状态
            if (newStateId == -1)
            {
                nextState.sid = scnt++;
                nextState.cellV = t.second.cellV;
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
            nextState.originV = t.second.originV;
            dfaStateVector.push_back(nextState);
        }
        else nextState.sid = newStateId;
        // 存入现在这个状态的 nextStateVector
        nextStateUnit n = nextStateUnit();
        n.sid = nextState.sid;
        n.c = t.first;
        dfaStateVector[stateId].nextStateVector.push_back(n);
    }

    // 对每个下一个状态进行递归
    int nsize = dfaStateVector[stateId].nextStateVector.size();
    for (int i = 0; i < nsize; i++)
    {
        auto& nextunit = dfaStateVector[stateId].nextStateVector[i];
        generateLR0State(nextunit.sid);
    }
}

// 生成 LR0 入口
void getLR0()
{
    visitedStates.clear();

    // 首先生成第一个状态
    createFirstState();

    // 递归生成其他状态
    generateLR0State(0);
}
```

2.5 生成 SLR(1)分析表

首先我们要检验文法是否属于 SLR(1)文法，主要判断规则如下：

•当且仅当对于任何状态 s，以下的两个条件：

•1) 对于在 s 中的任何项目 A→a.Xb，当 X 是一个终结符，且 X 在 Follow (B) 中时，s 中没有完整的项目 B→c.。 [移进-归约冲突]

•2) 对于在 s 中的任何两个完整项目 A→a.和 B→b.，Follow(A)∩Follow(B)为空。[归约-归约冲突]

均满足时，文法为 SLR(1)文法。

因此根据规则，我们可以得到以下代码：

```
// 检查移进-规约冲突
bool SLR1Fun1()
{
    for (const dfaState& state : dfaStateVector)
    {
        // 规约项目的左边集合
        set<char> a;
        // 终结符
        set<char> rVT;
        // 不是规约状态不考虑
        if (!state.isEnd) continue;
        // 规约状态
        for (int cellid : state.cellV)
        {
            // 拿到这个 cell
            const dfaCell& cell = dfaCellVector[cellid];
            // 获取文法
            const grammarUnit gm = grammarDeque[cell.gid];
            // 判断是不是规约项目
            if (cell.index == gm.right.length() || gm.right == "@")
            {
                a.insert(gm.left);
            }
            // 判断是不是终结符
            else
            {
                if (isSmallAlpha(gm.right[cell.index]))
                {
                    rVT.insert(gm.right[cell.index]);
                }
            }
        }
        for (char c : a)
        {
```

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
            for (char v : rVT)
            {
                if (followSets[c].s.find(v) != followSets[c].s.end())
                {
                    return true;
                }

            }
        }
    }
    return false;
}

bool SLR1Fun2()
{
    // 检查规约-规约冲突
    for (const auto& state : dfaStateVector)
    {
        // 规约项目的左边集合
        set<char> a;
        // 不是规约状态不考虑
        if (!state.isEnd) continue;

        // 规约状态
        for (int cellid : state.cellV)
        {
            // 拿到这个 cell
            const dfaCell& cell = dfaCellVector[cellid];
            // 获取文法
            const grammarUnit gm = grammarDeque[cell.gid];
            // 判断是不是规约项目
            if (cell.index == gm.right.length() || gm.right == "@")
            {
                a.insert(gm.left);
            }
        }

        for (char c1 : a)
```

```
        {
            for (char c2 : a)
            {
                if (c1 != c2)
                {
                    // 判断 followSets[c1]和 followSets[c2]是否有交集
                    set<char> followSetC1 = followSets[c1].s;
                    set<char> followSetC2 = followSets[c2].s;
                    set<char> intersection;

                    // 利用 STL 算法求交集
                    set_intersection(
                        followSetC1.begin(), followSetC1.end(),
                        followSetC2.begin(), followSetC2.end(),
                        inserter(intersection, intersection.begin())
                    );

                    // 如果交集非空，说明存在规约-规约冲突
                    if (!intersection.empty())
                    {
                        return true;
                    }
                }
            }
        }
    }


    return false;
}
```

最后我们构建 SLR(1)分析表，其实 SLR(1)分析表是在 LR(0)基础上构建得来的，因此，只是对 LR(0)内数据进行一些处理和特殊判断，如归约状态的处理、最终接受的处理，同时还需要求解 FOLLOW 集合，来使归约状态得到更好的判断，因此我们可以编写出下面的代码：

```
// SLR1 分析表
int getSLR1Table()
{
    // SLR1 分析错误就直接停止
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
    int r = SLR1Analyse();
    if (r != 0) return r;
    // 如果分析正确，通过 LR0 构造 SLR1 分析表（必须先调用 getLR0）
    for (const dfaState& ds : dfaStateVector)
    {
        SLRUnit slrunit = SLRUnit();
        // 如果是归约，得做特殊处理
        if (ds.isEnd)
        {
            // 规约项目的非终结符
            char gl;
            string gr;
            // 规约状态
            for (int cellid : ds.cellV)
            {
                // 拿到这个 cell
                const dfaCell& cell = dfaCellVector[cellid];
                // 获取文法
                const grammarUnit gm = grammarDeque[cell.gid];
                // 判断是不是规约项目
                if (cell.index == gm.right.length() || gm.right == "@")
                {
                    gl = gm.left;
                    gr = gm.right;
                    break;  // 前面的 SLR1 校验保证了只有一个归约项目
                }
            }
            // 得到这个非终结符 Follow 集合
            set<char> follow = followSets[gl].s;
            // follow 集合每个元素都能归约
            for (char ch : follow)
            {
                if (gl == trueStartSymbol) slrunit.m[ch] = "ACCEPT";
                else slrunit.m[ch] = "r(" + string(1, gl) + "->" + gr + ")";
            }
            // 对于下一个节点（可能会存在）
            for (const auto& next : ds.nextStateVector)
            {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

```
                char ch = next.c;
                int sid = next.sid; //  下一个状态 id

                // 获取下一个状态具体信息
                dfaState d = dfaStateVector[sid];
                if (isBigAlpha(ch))
                {
                    slrunit.m[ch] = to_string(sid);
                }
                else
                {
                    slrunit.m[ch] = "s" + to_string(sid);
                }
            }
        }
        else
        {
            for (const auto& next : ds.nextStateVector)
            {
                char ch = next.c;
                int sid = next.sid; //  下一个状态 id
                // 获取下一个状态具体信息
                dfaState d = dfaStateVector[sid];
                if (isBigAlpha(ch))
                {
                    slrunit.m[ch] = to_string(sid);
                }
                else
                {
                    slrunit.m[ch] = "s" + to_string(sid);
                }
            }
        }
        SLRVector.push_back(slrunit);
    }
    return 0;
}
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理 实验项目 SLR(1)分析生成器

实验时间 2023 年 12 月 12 日

实验指导老师 黄煜廉 实验评分

五、实验测试

详见《测试报告》。

六、小结

在本次实验中，我们成功设计并实现了一个用于处理文法的应用程序。我们成功计算并呈现了文法中各非终结符的 First 集合与 Follow 集合。这些集合的计算对于语法分析表的构建非常关键，用户可以通过窗口直观地查看结果，以验证文法的正确性。我们提供了一个窗口，展示了文法对应的 LR(0) DFA 图。这有助于用户理解文法的状态转换和 LR(0)自动机的结构。通过图形方式，用户能够更清晰地了解文法的 LR(0)自动机。并且程序还能判断文法是否为 SLR(1)文法。如果文法不是 SLR(1)文法，用户可以查看原因，这对于纠正文法错误或优化文法设计非常有帮助。在文法为 SLR(1)文法时，我们提供了一个窗口展示文法的 SLR(1)分析表。这个表格显示了文法符号、状态、动作等信息，帮助用户理解 SLR(1)分析过程。

整体而言，实验取得了成功。程序实现了实验要求的各项功能，用户可以方便地输入、管理文法，查看 First 集合、Follow 集合、LR(0) DFA 图、SLR(1)文法判断和 SLR(1)分析表的结果，从而更好地理解和分析文法。