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：2021 级 计科 1 班

学　　生：翁行

学　　号：20212131001

华南师范大学教务处

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学　号 20212131001

专　业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

一、 实验内容

设计一个应用软件，以实现 SLR(1)分析生成器。

二、 实验要求

1. 必做

(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。

2. 选做

(1) 需要提供窗口以便用户输入需要分析的句子。

(2) 需要提供窗口以便用户查看使用 SLR(1)分析该句子的过程。【可以使用表格的形式逐行显示分析过程】

三、 实验文档

3.1 实验环境

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

- 操作系统：macOS 14.2，基于 arm64 架构
- 硬件设备：MacBook M1 Pro，16GB 运行内存
- 开发环境：
  - Xcode-Command-Line-Tools，包含 Apple clang version 15.0
  - C++17
  - QT 6.6

3.2 实验分析

3.2.1 用户界面设计

本次实验，我是用 Qt 6.6 来构建跨平台的 GUI（用户图形界面），支持在 macOS 和 Windows10 操作系统上运行。界面主要包括【导入文件】【保存文件】【文法输入】【语句输入】【文法解析】功能模块，具体界面如下：

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

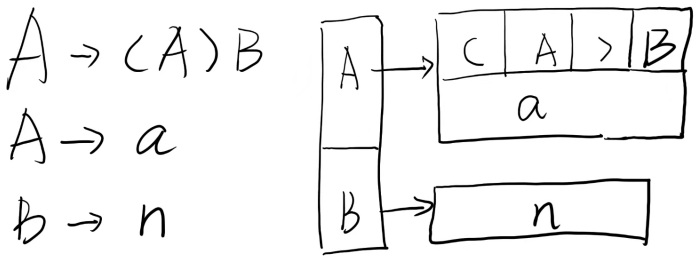
实验指导老师 黄煜廉 实验评分

3.2.2 文法输入和解析

根据实验要求，文法是由多个单行的“非终结符 -> 推导式”组成的多行文本，其中 A-Z 大写字母表示非终结符，其他字符（除了@表示 EPSILON 外）表示终结符。

首先，需要对用户输入的文法进行解析拆分，然后将所有输入转化为 map<string, vector<vector<string>>>的形式。其中，map 的 key 就是每一个非终结符号，value 是一个二维容器，第一层指示了推导式的行号，第二层指示了该行推导式的各个被拆分出来的 Token。

比如：A->(A)B \n A->a 就会解析为：



然后，由于首个非终结符号有可能有多个推导分式（或的存在），于是我额外产生一个推出当前 startToken 的推导式，例如 A' -> A（无论 A 有没有多个推导分式）

完成对输入字符串的解析之后，也就形成了上面所说的 map<string, vector<vector<string>>> 的 formula，遍历这个 formula 的 key，即可初始化得到所有的非终结符号集合，采用类似 BFS 的方法遍历 formula 所有 value 中的 Token 后，即可得到所有终结符号集合。这一部分的代码如下所示（我还加入了 | 的支持，也就是表示或不一定需要换行使用新的推导式）。

```
Grammer::Grammer(string input) {
    vector<string> lines;
    int from = 0, i = 0;
    for (i = 0; i < input.size(); ++i) {
        if (input[i] == '\n') {
            lines.push_back(input.substr(from, i - from));
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            from = i + 1;
        }
    }
    if (from != i)
        lines.push_back(input.substr(from));
    if (lines.empty()) {
        error = "未输入任何文法";
        return;
    }
    for (int i = 0; i < lines.size(); ++i) {
        string line = lines[i];
        string key;
        vector<string> raws;
        bool behind = false;
        for (int j = 0; j < line.size(); ++j) {
            if (line[j] == ' ')
                continue;
            if (line[j] == '-' && j < line.size() - 2 && line[j + 1] == '>') {
                // ->
                behind = true;
                j++;
                continue;
            }
            if (line[j] == '|') {
                if (!behind) {
                    error = "|符号不能出现在左值";
                    return;
                }
                formula[key].push_back(raws);
                raws.clear();
                continue;
            }
            if (!behind) {
                if (key.size()) {
                    error = "文法左式不支持多字符";
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
                return;
            }
            key += line[j];
            continue;
        }
        raws.push_back(string(1, line[j]));
    }
    if (!behind) {
        error = "文法输入有误";
        return;
    }
    if (!raws.empty()) {
        formula[key].push_back(raws);
    }
    if (i == 0)
        start = key;
}
// 拓广文法
formula[start + '\''].push_back(vector<string>(1, start));
start = start + '\'';
// 构建非终结符号集
for (auto it = formula.begin(); it != formula.end(); ++it) {
    notEnd.insert(it->first);
}
// 构建终结符号集合
for (auto& p : formula) {
    for (auto& raws : p.second) {
        for (auto& raw : raws) {
            if (!notEnd.count(raw) && !endSet.count(raw))
endSet.insert(raw);
        }
    }
}
}
```

3.2.3 求非终结符号 First 集合

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

求非终结符号的 First 元素有三种情况：

1. 推导式首个字符为终结符号：First 集合为该终结符号；
2. 推导式首个字符为非终结符号，且该非终结符号 First 集合不含 EPSILON：First 集合为该非终结符号的 First 集合
3. 推导式首个字符为非终结符号，且该非终结符号 First 集合含有 EPSILON：First 集合为该非终结符号的 First 集合去掉 EPSILON，然后继续求下一个字符的 First 集合，直到 First 集合中不包含 EPSILON 即停止。如果整个推导式所有字符的 FIRST 集合都包含 EPSILON，则将 EPSILON 加入该非终结符号 First 集合中。

所以总体来说，这是一个循环更新的过程，可以写出下面的代码：

```
void Grammer::initFirst() {
    bool shouldUpdate = true;
    while (shouldUpdate) {
        shouldUpdate = false;
        for (auto& p : formula) {
            string key = p.first;                    // 非终结符
            vector<vector<string>> raws = p.second; // 产生式右侧
            for (const auto& raw : raws) {
                int cur = 0;
                for (; cur < raw.size(); ++cur) {
                    auto firstOfCur = getFirst(raw[cur]); // 当前元素的 First 集合
                    // 遍历当前 First
                    for (auto& el : firstOfCur) {
                        // 除了 EPSILON 外，新增的元素都加入 key 的 First
                        if (el != EPSILON && !first[key].count(el)) {
                            first[key].insert(el);
                            shouldUpdate = true;
                        }
                    }
                    // EPSILON 不在 cur 的 First，可以退出推导式右侧的遍历
                    if (!firstOfCur.count(EPSILON)) {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
                    break;
                }
            }
            // 右侧所有元素 First 都包含 EPSILON，则 key 的 First 也应该包含 EPSILON
            if (cur == raw.size() && !first[key].count(EPSILON)) {
                first[key].insert(EPSILON);
                shouldUpdate = true;
            }
        }
    }
}
```

3.2.4 求非终结符号 Follow 集合

求非终结符号的 Follow 集合同样有 3 种情况，使用 key 来指代要求的非终结符号：

1. Key 后为终结字符，Follow 集合为该终结字符构成的集合
2. Key 后为非终结字符 T，且 T 的 First 集合中不包含 EPSILON：Key 的 Follow 集合为 T 的 First 集合
3. Key 后为非终结字符 T，且 T 的 First 集合中包含 EPSILON：Key 的 Follow 集合为 T 的 First 集合去除 EPSILON，然后继续求下一个字符的 First 集合，直到不含 EPSILON 为止。如果当前推导式的后续字符 First 集合都包含 EPSILON，则 Key 的 Follow 集合也包含其父级的 Follow 集合。

代码如下：

```
void Grammer::initFollow() {
    bool shouldUpdate = true;
    // start 的 Follow 为 END_FLAG
    follow[start].insert(END_FLAG);
    while (shouldUpdate) {
        shouldUpdate = false;
        for (auto& p : formula) {
            string key = p.first;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
vector<vector<string>> raws = p.second;
// 遍历每一个推导式右侧
for (const auto& raw : raws) {
    // 遍历每一个非终结符号
    for (int i = 0; i < raw.size(); ++i) {
        if (!notEnd.count(raw[i]))
            continue;
        // 末尾
        if (i == raw.size() - 1) {
            for (const auto& el : getFollow(key)) {
                if (!follow[raw[i]].count(el)) {
                    follow[raw[i]].insert(el);
                    shouldUpdate = true;
                }
            }
            continue;
        }
        // 非末尾，获取后续元素的 First 集合
        set<string> firstOfBehind;
        int cur = i + 1;
        for (; cur < raw.size(); ++cur) {
            set<string> firstOfCur = getFirst(raw[cur]);
            for (const auto& el : firstOfCur) {
                if (el != EPSILON)
                    firstOfBehind.insert(el);
            }
            if (!firstOfCur.count(EPSILON)) {
                // 不含 EPSILON，First 终止
                break;
            }
        }
        if (cur == raw.size()) {
            // 每个元素的 First 都包含 Epsilon
            firstOfBehind.insert(EPSILON);
        }
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
                // 更新 Follow 集合
                for (const auto& el : firstOfBehind) {
                    if (el != EPSILON && !follow[raw[i]].count(el)) {
                        follow[raw[i]].insert(el);
                        shouldUpdate = true;
                    }
                }
                if (firstOfBehind.count(EPSILON)) {
                    // 包含 EPSILON，Follow 集合包含产生式左侧的 Follow 集合
                    for (const auto& el : getFollow(key)) {
                        // 消除曾经的父级的 EPSILON
                        follow[raw[i]].erase(EPSILON);
                        if (!follow[raw[i]].count(el)) {
                            follow[raw[i]].insert(el);
                            shouldUpdate = true;
                        }
                    }
                }
            }
        }
    }
}
```

3.2.5 生成 LR(0) DFA 图，判断是否为 SLR(1)文法

首先明确 DFA 节点的数据类型，我将其安排为 vector<vector<Node>>，其中第一层 vector 指示了 DFA 的状态编号（0 号就是位置 0），第二层指示了 DFA 内部的项目，因为一个 DFA 节点内可以存在多个推导式项目。其中 Node 的结构如下：

```
enum NodeType {
    FORWARD,
    BACKWARD
};

struct Node {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
    std::string key; // 所属非终结符号
    NodeType type; // 递进还是规约
    int rawsIndex; // 推导式编号
    int rawIndex; // 推导式内编号

    Node(std::string key, NodeType type, int rawsIndex, int rawIndex):
key(key), type(type), rawsIndex(rawsIndex), rawIndex(rawIndex) {}

    const bool operator==(const Node& node) {
        return node.key == key && node.type == type && node.rawsIndex ==
rawsIndex && node.rawIndex == rawIndex;
    }
};
```

从 startToken 生成一个 DFA 图的初始节点，然后进入循环开始反复更新 DFA 图。

进入循环的第一件事是 extend 当前的 DFA 节点，因为初始情况下的 DFA 节点只有 1 个推导式（随着生成的进度也有可能更多），其中可能包含非终结符号，则必须通过 extend 来将所有推导式加入到当前的 DFA 节点中，直到没有新的非终结符号出现。代码如下：

```
void Grammer::extend(vector<Node>& nodes) {
    for (int i = 0; i < nodes.size(); ++i) {
        Node& node = nodes[i];
        if (node.type == NodeType::BACKWARD)
            continue; // 跳过规约节点
        string cur = formula[node.key][node.rawsIndex][node.rawIndex]; // 指示
的符号
        if (!notEnd.count(cur))
            continue; // 终结字符不可扩展
        vector<vector<string>>& rawsOfCur = formula[cur]; // 非终结字符为 Key 的推
导式
        for (int j = 0; j < rawsOfCur.size(); ++j) {
            int rawOffset = 0;
            for (; rawOffset < rawsOfCur[j].size(); ++rawOffset) {
                // 寻找到非空字符
                if (rawsOfCur[j][rawOffset] != EPSILON)
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
                break;
            }
            // 新增节点，指示了 Key 对应的第 i 个推导式的第 rawOffset 个字符
            // 如果开头存在了 EPSILON，则节点类型为规约
            Node instance(cur,
                          rawOffset == 0 ? NodeType::FORWARD :
NodeType::BACKWARD, j,
                          rawOffset);
            // 无重复则扩展 state 指示的 dfa 节点
            if (!count(nodes.begin() /*+ i*/, nodes.end(), instance)) {
                nodes.push_back(instance);
            }
        }
    }
}

void Grammer::extend(int state) {
    extend(dfa[state]);
}
```

扩展节点后，就可以遍历每一个状态（state），从中再遍历 state 中的每一个分式，拿到当前的 Node，然后根据 Node 的类型（规约或移进），来记录 forwards 关系或 backwards 关系，同时在必要的时候新增 DFA 状态或状态内分式。

如果 Node 是规约节点，则获取 Node 所持有的 Key（非终结节点，推导式左侧）的 Follow 元素集合，遍历并判断当前 state 的 backwards 关系是否与这个 Follow 集合有交集，如果有，则不为 **SLR(1)文法**，同时无论有无交集，将 Follow 集合中的元素和当前推导式的 Offset（也就是当前推导式在 DFA 内部是第几个 Node）记录进当前 state 的 backwards 关系中。

如果 Node 是移进节点，首先获取到 Node 记录的下一个 Token（由 Node 记录的 Key、rawsIndex、rawIndex 共同在 formula 中决定），也就是移进所需要紧跟着的那个字符，记录为 raw。因为是移进节点，也就是有可能产生新的 DFA 状态，或者产生 DFA 状态内的新的推导式，于是新增一个 Node 的实例，显然这个新增的 instance 与已经存在的当前 Node 的唯一区

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

别就是 rawIndex 增加 1（rawIndex 指示的是 DFA 状态内部推导式中的 Token offset）。当然，如果 rawIndex 增加 1 后已经达到了目标推导式的最末尾（rawIndex == size()），则说明该新增的 instance 应该变为规约节点。

做完新增 Node instance 的工作后，就是要判断是否应该加入到 DFA 中或者加入已经存在的 DFA 中（新增推导式 offset）。类似于上面提到的 backwards，移进关系也有一个 forwards 来记录 state 的移进关系（从 state 通过 token 可以移进到哪一个 state）。首先在 forwards[state]中寻找是否已经存在了 token 的移进目标，如果存在，则说明目标 DFA 节点早已存在，但是 DFA 内部说不定还未存在该 instance 指向的推导式，于是在目标 DFA 中寻找是否已经存在 instance（重载的==方法见上面 Node 的结构），未存在则在目标 DFA 状态中新增这个 instance；

如果未存在，则有可能需要创建新的 DFA 节点。先创建一个可能会作为新增节点的 DFA 节点，但暂时不 push_back 进 DFA 的节点列表中，然后执行一次 extend，使之变成完整的节点，然后再在已有的 DFA 节点中找寻是否存在一模一样的 DFA 节点，如果有，则不更新，否则将这个 DFA 节点 push_back 入现有的 DFA 中。无论是否更新 DFA，都要记录进 forwards 关系中。这部分的代码见下：

```
void Grammer::initRelation() {
    // 初始节点 => start 指示的推导式的第一条的第一个符号
    vector<Node> beginState;
    beginState.push_back(Node(start, NodeType::FORWARD, 0, 0));
    dfa.push_back(beginState);
    isSLR = true; // 暂时先是
    // 遍历每一个 DFA 节点
    for (int cur = 0; cur < dfa.size(); ++cur) {
        // forwards[cur]和 backwards[cur]分别记录了移进和规约关系
        extend(cur); // 扩展当前 DFA 节点(可能右侧项目含有非终结符号)
        // 遍历 DFA 节点上的每一个项目
        for (int it = 0; it < dfa[cur].size(); ++it) {
            Node& item = dfa[cur][it]; // 取出当前项
            if (item.type == NodeType::BACKWARD) {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            // 规约项
            set<string> followOfItem = getFollow(item.key);
            for (const auto& el : followOfItem) {
                if (backwards[cur].count(el)) {
                    // 存在交集，非 SLR(1)
                    isSLR = false;
                    stringstream ss;
                    ss << "第" << cur << "个节点中规约项目的 Follow 集合有交集\n";
                    reason += ss.str();
                }
                backwards[cur][el] = it;
            }
            continue;
        }
        // 移进项
        string raw = formula[item.key][item.rawsIndex][item.rawIndex];
        // 移进新的节点
        Node instance(item.key, NodeType::FORWARD, item.rawsIndex,
                      item.rawIndex + 1);
        if (instance.rawIndex >=
            formula[instance.key][instance.rawsIndex].size()) {
            // 超过了该推导式的结尾 -> 变成规约节点
            instance.type = NodeType::BACKWARD;
        }
        if (forwards[cur].count(raw)) {
            // 已经存在该移进关系
            int target = forwards[cur][raw];
            vector<Node>& next = dfa[target];
            if (!count(next.begin(), next.end(), instance)) {
                // 如果下一 DFA 节点中未存在该 Instance 状态 -> 加入下一 DFA 节点中
                dfa[target].push_back(instance);
            }
            continue;
        }
        // 未存在该移进关系
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            vector<Node> perhapsNewState{ instance };
            extend(perhapsNewState);
            int target = findState(perhapsNewState);
            if (target == -1) {
                // 该状态不存在于任何 DFA 节点中 -> 新增一个 DFA 节点
                dfa.push_back(perhapsNewState);
                target = dfa.size() - 1;
            }
            // 加入移进关系
            forwards[cur][raw] = target;
        }
    }
}
int Grammer::findState(vector<Node>& current) {
    for (int i = 0; i < dfa.size(); ++i) {
        auto& state = dfa[i];
        bool exist = true;
        if (current.size() != state.size()) continue;
        for (auto& node : current) {
            if (!count(state.begin(), state.end(), node)) {
                exist = false;
                break;
            }
        }
        if (exist) return i;
    }
    return -1;
}
```

一轮增长过后，DFA 图就已经成功生成了，但是还不能说目前是 SLR(1)文法就一定是 SLR(1)文法，因为每个 state 的移进、规约关系并没有在生成 DFA 图的过程中进行判断是否产生交集，于是需要对此进行判断。

```
void Grammer::initIsSLR() {
    // DFA 图构建完成后 -> 判断移进规约是否冲突
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
    if (isSLR) {
        stringstream ss;
        for (int cur = 0; cur < dfa.size(); ++cur) {
            set<string> curForwards, curBackwards, duplicates;
            for (auto p : forwards[cur]) {
                curForwards.insert(p.first);
            }
            for (auto p : backwards[cur]) {
                curBackwards.insert(p.first);
            }
            set_intersection(curForwards.begin(), curForwards.end(),
                curBackwards.begin(), curBackwards.end(),
                inserter(duplicates, duplicates.begin()));
            if (!duplicates.empty()) {
                // 交集不空 不为 SLR
                isSLR = false;
                ss.str("");
                ss.clear();
                ss << "第" << cur
                    << "个节点的移进项 First 集合和规约项 Follow 集合有交集\n";
                reason += ss.str();
            }
        }
    }
}
```

3.2.6 解析输入的语句

此部分选做内容我也完成了，在此就介绍是因为其逻辑仍然无关 Qt，而 SLR(1)分析表和 LR(0) DFA 图其实已经在数据层面完成了，后面会介绍我是如何通过 Qt 渲染成表格呈现的。

根据课堂上老师讲授的思路，大体上是需要一个输入队列和一个输出栈，每一轮从输入队列弹出 1 个 Token，压入输出栈，同时根据 forwards 和 backwards 关系判断移进/规约到哪一个 state 中。

移进关系在这一部分其实非常简单了，因为 forwards 已经记录了 state 和遇到的 token 对应

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

的下一个 state，在下一次循环前更新 state 就可以了，下面着重介绍我对规约关系的处理方法。

首先，backwards 的记录其实并不是下一个 state，而是本次规约是利用了当前 state 下的 DFA 内部的哪个 Node 来进行的。拿到这个 Node 之后，也就获得了这个 Node 持有的 Key（非终结 Token）、指向 formula 的 rawsIndex，那么也就是说拿到了 formula 中记录的某一条推导式右部（规约 Node 记录的 rawIndex 是脏的，还记得上面说成为规约节点的条件吗？就是超过了推导式边界，也就是 rawIndex 无意义）。

获取得到 formula[rawsIndex]指代的推导式右部后，就可以根据这个右部的 size()来对现有的输出栈进行清理，也就是要把输出栈清理出右部的 size()个位置后，再压入持有该右部的非终结 Token。比如 A->(A)，就需要将(A)分别出栈，然后压入一个 A。与此同时，state 也应该回到回退右部的 size()这么多长度的时机上，对此我引入一个状态栈（类似输出栈）.

当一个输入 Token 既找不到移进关系又找不到规约关系时，则代表输入是错误的，无法成功被接收。上述代码实现如下：

```
ParsedResult Grammer::parse(string input) {
    string str;
    for (auto& s : input) {
        if (s != ' ' && s != '\n') str += s;
    }
    ParsedResult result;
    string output;
    queue<string> inputs;
    vector<int> stash;

    for (const char& c : input) {
        inputs.push(string(1, c));
    }
    inputs.push(END_FLAG);
    int state = 0; // 当前 DFA 状态编号
    int count = 0;
    stringstream ss;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
for (;;) {
    ss.str("");
    ss.clear();
    map<string, int>& curForwards = forwards[state]; // 当前所有可移进状态
    map<string, int>& curBackwards = backwards[state]; // 当前所有可规约状态

    string token = inputs.front(); // 当前输入的字符
    stash.push_back(state); // 当前状态入栈

    if (curForwards.count(token)) {
        // 找到了移进关系
        inputs.pop();
        ++count;
        int next = curForwards[token]; // 下一个状态
        ss << "在状态" << state << "通过" << token << "移进到状态" << next;
        state = next;
        output += token;
        result.outputs.push_back(output);
        result.routes.push_back(ss.str());
        result.inputs.push_back(str.substr(count));
        continue;
    }
    if (curBackwards.count(token)) {
        // 找到了规约关系
        int target = curBackwards[token];
        ss << "在状态" << state << "通过" << token << "规约到状态" << target;
        Node& node = dfa[state][target];
        if (count >= str.size()) {
            result.inputs.push_back("");
        }
        else {
            result.inputs.push_back(str.substr(count));
        }
        result.routes.push_back(ss.str());
        if (node.key == start) {
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            // 接收
            result.accept = true;
            result.outputs.push_back(start);
            break;
        }
        vector<string>& raws = formula[node.key][node.rawsIndex];
        int useful = 0;
        for (int i = 0; i < raws.size(); ++i) {
            // 找到不是 EPSILON 的大小
            if (raws[i] != EPSILON) useful++;
        }
        if (useful > 0) {
            output.erase(output.end() - useful, output.end());
            stash.erase(stash.end() - useful, stash.end());
        }
        int next = forwards[stash[stash.size() - 1]][node.key];
        state = next;
        output += node.key;
        result.outputs.push_back(output);
        continue;
    }
    // 找不到关系，出错
    ss << "在状态" << state << "上找不到" << token << "对应的移进/规约关系";
    result.error = ss.str();
    break;
  }
  return result;
}
```

其中 ParsedResult 的结构定义如下：

```
struct ParsedResult {
  std::vector<std::string> outputs;
  std::vector<std::string> inputs;
  std::vector<std::string> routes;
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
    bool accept = false; // 是否接受
    std::string error = ""; // 错误信息，空则无出错
};
```

3.2.7 Qt 渲染解析 Follow 集合、First 集合、文法错误和是否 SLR(1)

这里的 Core Logic 已经在上面介绍过了，主要是通过 Qt 在 GUI 中渲染出来。因为我设计 Core API 时已经刻意解藕，只剩下数据耦合，所以这里唯一要做的是调用刚才介绍的函数，然后将 std 下的 string 转化为 QString 后渲染在 UI 中渲染出来。

```
void MainWindow::renderBasicInfo() {
    if (!currentGrammer) {
        QMessageBox::information(this, "提示", "请先点击解析文法");
        return;
    }
    Grammer& grammer = *currentGrammer;
    QString error = QString::fromStdString(grammer.getError());
    if (error.isEmpty()) error = "未发现错误";
    ui->syntaxError->setPlainText(error);
    ui->syntaxType->setPlainText(grammer.slr() ? "SLR 文法" : grammer.bad() ? "
错误文法" : "LR 文法\n" + QString::fromStdString(grammer.getReason()));
    QString followSet, firstSet;
    // 渲染非终结节点的 Follow 集合和 First 集合
    std::set<std::string> notEnd = grammer.getNotEnd();
    for (std::string token : notEnd) {
        firstSet += token + ": ";
        followSet += token + ": ";
        std::set<std::string> firstOfToken = grammer.getFirst(token);
        std::set<std::string> followOfToken = grammer.getFollow(token);
        for (auto it = firstOfToken.begin(); it != firstOfToken.end();) {
            firstSet += *it;
            if (++it != firstOfToken.end()) firstSet += ", ";
        }
        for (auto it = followOfToken.begin(); it != followOfToken.end();) {
            followSet += *it;
            if (++it != followOfToken.end()) followSet += ", ";
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
        }
        firstSet += "\n";
        followSet += "\n";
    }
    ui->firstSet->setPlainText(firstSet);
    ui->followSet->setPlainText(followSet);

    // 渲染拓广文法
    QString extraGrammer = QString::fromStdString(grammer.getExtraGrammer());
    ui->extraGrammer->setPlainText(extraGrammer);
}
```

3.2.8 渲染 DFA 图、SLR(1)分析表

DFA 的生成方法已经在上面介绍，这里通过遍历生成的 DFA 和 forwards、backwards，配合 QTableWidget 生成 GUI

```
void MainWindow::renderDfaTable() {
    if (!currentGrammer) {
        QMessageBox::information(this, "提示", "请先点击解析文法");
        return;
    }
    Grammer& grammer = *currentGrammer;
    std::vector<std::vector<Node> > dfa = grammer.getDfa();
    std::set<std::string> endSet = grammer.getEnd();
    std::set<std::string> notEndSet = grammer.getNotEnd();
    std::string startToken = grammer.getStart();
    auto* table = ui->dfa;
    table->setColumnCount(1+endSet.size()+notEndSet.size());
    table->setRowCount(dfa.size());
    QStringList header;
    header << "状态" << "状态内文法";
    for (auto& token : notEndSet) {
        if (token == startToken) continue;
        header << QString::fromStdString(token);
    }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
    for (auto& token : endSet) {
        header << QString::fromStdString(token);
    }
    table->setHorizontalHeaderLabels(header);
    table->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    // 文法
    auto formula = grammer.getFormula();
    // 迭代生成 cell
    for (int state = 0; state < (int)dfa.size(); ++state) {
        QTableWidgetItem *id = new QTableWidgetItem(); // 状态编号
        QTableWidgetItem *inner = new QTableWidgetItem(); // 状态内文法

        id->setText(QString::fromStdString(std::to_string(state)));
        table->setItem(state, 0, id); // 加入编号列

        QString innerText;
        for (int offset = 0; offset < (int)dfa[state].size(); ++offset) {
            // 遍历节点内部
            Node& cur = dfa[state][offset];
            std::vector<std::string> rawOfCur =
formula[cur.key][cur.rawsIndex]; // 那一行文法
            innerText += QString::fromStdString(cur.key) + " -> ";
            for (int tokenOffset = 0; tokenOffset <= (int)rawOfCur.size();
++tokenOffset) {
                // 构造类似 A -> (.a)
                if (tokenOffset == cur.rawIndex) innerText += ".";
                innerText += QString::fromStdString(rawOfCur[tokenOffset]);
            }
            innerText += "\n";
        }
        inner->setText(innerText);
        table->setItem(state, 1, inner);

        int col = 2;
        for (auto& token : notEndSet) {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            if (token == startToken) continue;
            int target = grammer.forward(state, token);
            if (target >= 0) {
                QTableWidgetItem *item = new QTableWidgetItem();
                item->setText(QString::fromStdString(std::to_string(target)));
                table->setItem(state, col, item);
            }
            col++;
        }
        for (auto& token : endSet) {
            int target = grammer.forward(state, token);
            if (target >= 0) {
                QTableWidgetItem *item = new QTableWidgetItem();
                item->setText(QString::fromStdString(std::to_string(target)));
                table->setItem(state, col, item);
            }
            col++;
        }
    }

}
```

SLR(1)的分析表生成上基本逻辑是：遍历 DFA 每一个 state，在循环内部分别遍历所有终结符号和非终结符号，分别判断是否存在移进/规约关系（类似上面介绍的语句分析方法）。根据移进/规约关系渲染表格。

```
void MainWindow::renderSlrTable() {
    Grammer& grammer = *currentGrammer;
    if (!grammer.slr()) {
        return;
    }
    // 是 SLR(1)文法
    std::set<std::string> endSet = grammer.getEnd();
    std::set<std::string> notEndSet = grammer.getNotEnd();
    std::string startToken = grammer.getStart();
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
auto dfa = grammer.getDfa();
auto formula = grammer.getFormula();
endSet.insert(END_FLAG);
auto* table = ui->slr;
table->setColumnCount(endSet.size() + notEndSet.size());
table->setRowCount(dfa.size());
QStringList header;
header << "状态";
for (auto& token : notEndSet) {
    if (token == startToken) continue;
    header << QString::fromStdString(token);
}
for (auto& token : endSet) {
    header << QString::fromStdString(token);
}
table->setHorizontalHeaderLabels(header);
table->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
// Cell
for (int state = 0; state < (int)dfa.size(); ++state) {
    QTableWidgetItem *id = new QTableWidgetItem(); // 状态编号
    id->setText(QString::number(state));
    table->setItem(state, 0, id);
    int column = 1;
    for (auto& token : notEndSet) {
        if (token == startToken) continue;
        int target = grammer.forward(state, token);
        if (target > -1) {
            QTableWidgetItem *end = new QTableWidgetItem(); // 状态编号
            end->setText("s" + QString::number(target));
            table->setItem(state, column, end);
        } else if ((target = grammer.backward(state, token)) > -1) {
            QTableWidgetItem *end = new QTableWidgetItem(); // 状态编号
            Node& node = dfa[state][target];
            if (node.key == startToken) {
                end->setText("ACCEPT");
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            } else {
                QString endText = "r(" + QString::fromStdString(node.key) +
"->";
                for (auto &token : formula[node.key][node.rawsIndex]) {
                    endText += QString::fromStdString(token);
                }
                endText += ")";
                end->setText(endText);
            }
            table->setItem(state, column, end);
        }
        column++;
    }
    for (auto& token : endSet) {
        int target = grammer.forward(state, token);
        if (target > -1) {
            QTableWidgetItem *end = new QTableWidgetItem(); // 状态编号
            end->setText("s" + QString::number(target));
            table->setItem(state, column, end);
        } else if ((target = grammer.backward(state, token)) > -1) {
            QTableWidgetItem *end = new QTableWidgetItem(); // 状态编号
            Node& node = dfa[state][target];
            if (node.key == startToken) {
                end->setText("ACCEPT");
            } else {
                QString endText = "r(" + QString::fromStdString(node.key) +
"->";
                for (auto &token : formula[node.key][node.rawsIndex]) {
                    endText += QString::fromStdString(token);
                }
                endText += ")";
                end->setText(endText);
            }
            table->setItem(state, column, end);
        }
```

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
            column++;
        }

    }
}
```

3.2.9 Qt 渲染语句分析过程

ParsedResult 的获取过程已经在上面做好，这里仅仅是简单转换为 Qt 表格的形式输出：

```
void MainWindow::on_toParseStatement_clicked()
{
    QString statement = ui->statement->toPlainText();
    if (statement.isEmpty()) {
        QMessageBox::information(this, "提示", "请输入待解析语句");
        return;
    }
    if (!currentGrammer) {
        QMessageBox::information(this, "提示", "请先解析文法后再解析语句");
        return;
    }
    qDebug() << "待解析语句: " << statement;
    Grammer& grammer = *currentGrammer;
    ParsedResult result = grammer.parse(statement.toStdString());
    auto* table = ui->parseProcess;
    table->setColumnCount(3);
    table->setRowCount(result.outputs.size() + 1);
    QStringList header;
    header << "输入"
           << "操作"
           << "输出";
    table->setHorizontalHeaderLabels(header);
    table->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    for (int i = 0; i < result.inputs.size(); ++i) {
        auto* input = new QTableWidgetItem();
```

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

```
    auto* output = new QTableWidgetItem();
    auto* action = new QTableWidgetItem();
    input->setText(QString::fromStdString(result.inputs[i]));
    output->setText(QString::fromStdString(result.outputs[i]));
    action->setText(QString::fromStdString(result.routes[i]));
    table->setItem(i, 0, input);
    table->setItem(i, 2, output);
    table->setItem(i, 1, action);
  }
  auto* status = new QTableWidgetItem();
  status->setText(result.accept ? "接收" : "出错");
  table->setItem(result.outputs.size(), 0, status);
  if (result.error.size()) {
    auto* reason = new QTableWidgetItem();
    reason->setText(QString::fromStdString(result.error));
    table->setItem(result.outputs.size(), 1, reason);
  }
}
```

3.3 实验结果

实现了以下功能：

1. 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。【表格的形式】
3. 提供窗口以便用户可以查看文法对应的 LR(0)DFA 图。（表格方式）
4. 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. 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）【表格的形式】

华 南 师 范 大 学 实 验 报 告

学生姓名 翁行 学 号 20212131001

专 业 计算机科学与技术 年级、班级 2021 级计科 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 17 日

实验指导老师 黄煜廉 实验评分

6. 提供窗口以便用户输入需要分析的句子。
7. 提供窗口以便用户查看使用 SLR(1)分析该句子的过程。【使用表格的形式逐行显示分析过程】

下面展示程序运行时截图。

四、 实验总结

1. 体会了自底向上分析方法的精妙；
2. 加深了对 Qt 开发的了解；
3. 加深了对 LR(0)、SLR(1)文法和其分析方法的理解。

五、 参考文献

1. 编译原理课程讲稿；
2. QT 开发文档。