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：实验 4：LALR(1)分析生成器生成

指导老师：黄煜廉

开课时间：2023　～　2024　年度第　　学期

专　　业：计算机科学与技术

班　　级：3

学生姓名：蒋昕玮

华南师范大学教务处

华南师范大学实验报告

学生姓名 蒋昕玮 学　号 20222131051

专　业 计算机科学与技术 年级、班级 2022 级 3

课程名称 编译原理 实验项目 实验 4：LALR(1)分析生成器生成

实验类型 □验证 □设计 □综合 实验时间 2024 年 3 月 10 日

实验指导老师 黄煜廉 实验评分

一、实验内容

本实验的内容是实现一个简单的 LALR(1)语法分析器。该分析器包括语法定义、语法分析算法的实现、以及一个简单的图形用户界面，用于展示语法分析过程的可视化结果。

二、实验目的

- 掌握 LALR(1)分析方法及其实现过程。
- 熟悉语法分析的基本原理和步骤。
- 掌握 C++编程语言的高级用法，特别是面向对象编程的应用。
- 学会使用 Qt 框架开发图形用户界面程序。
- 提高软件工程规范书写文档的能力。

三 实验文档

1. 系统的总体结构

本实验项目主要包括以下几个模块：

- 语法定义模块（grammar.h）
- LALR(1)分析算法实现模块（LALR.cpp 和 LALR.h）
- 主程序模块（main.cpp）
- 图形用户界面模块（mainwindow.cpp 和 mainwindow.h）

文件：grammar.h

类：grammar

- void createExG()
    - 创建扩展的文法规则集合（扩展的产生式集合）。
- void createFirst()

- 计算每个非终结符的 FIRST 集合。

- void createFollow()
  - 计算每个非终结符的 FOLLOW 集合。
- 构造函数 grammar(const std::string filepath)
  - 从提供的文件路径读取文法规则，并初始化 FIRST 和 FOLLOW 集合。

文件：LALR.h

命名空间：std

- 模板特化 hash<LR1Item>
  - 为 LR1 项目提供自定义的哈希函数。
- 模板特化 hash<std::pair<std::string, std::vector<std::string>>>
  - 为项目核心提供自定义的哈希函数。
- 模板特化 hash<std::set<std::pair<std::string, std::vector<std::string>>>>
  - 为项目核心集合提供自定义的哈希函数。
- 模板特化 hash<std::unordered_set<LR1Item>>
  - 为 LR1 项目集合提供自定义的哈希函数。

结构体：LR1Item

- 成员函数 std::string rhs2str() const
  - 将项目右侧的符号序列转换为字符串。
- 重载运算符 operator==
  - 比较两个 LR1 项目是否相等。

结构体：LR1NODE

- 成员变量 std::unordered_set<LR1Item> items
  - 存储该节点包含的所有 LR1 项目。
- 成员变量 std::unordered_map<std::string, LR1NODE*> state
  - 存储从当前节点到其他节点的转移。
- 成员变量 int stateNum
  - 存储该节点的状态编号。
- 成员变量 std::string core
  - 存储该节点的核心项目集的字符串表示。

类：LALR

- std::string countcore(std::unordered_set<LR1Item> itemset)
    - 计算并返回一个项目集的核心项目集的字符串表示。
- void generateParsingTable()
    - 生成解析表。
- int statecount
    - 存储状态的数量。
- std::unordered_set<LR1Item> closure(const std::unordered_set<LR1Item>& items)
    - 计算给定 LR1 项目集的闭包。
- std::unordered_set<LR1Item> gotoState(const std::unordered_set<LR1Item>& items, const std::string& symbol)
    - 计算在给定符号下的转移得到新的项目集。
- void createLR1Automaton()
    - 创建 LR1 自动机。
- void createLALRAutomaton()
    - 创建 LALR(1)自动机。
- 构造函数 LALR(const std::string filepath)
    - 从提供的文件路径读取文法，并构建 LALR 分析器。
- 友元函数 printAutomaton(const LALR& lalr, LR1NODE* start, QTableWidget* tableWidget)
    - 打印 LR1 或 LALR 自动机的状态。
- 友元函数 printParsingTable(const LALR& lalr, QTableWidget* tableWidget)
    - 打印解析表。

2. 数据结构的选择

- **语法定义模块（grammar.h）**：用于定义上下文无关文法的规则，使用结构体存储产生式和符号表。
- **LALR(1)分析算法实现模块（LALR.cpp 和 LALR.h）**：使用 LALR(1)项集族及

其转换关系构造语法分析表，使用表驱动法进行语法分析。

- **主程序模块（main.cpp）**：负责初始化系统、加载语法规则、调用 LALR(1)分析模块进行语法分析。
- **图形用户界面模块（mainwindow.cpp 和 mainwindow.h）**：使用 Qt 框架实现，负责展示语法分析过程的可视化结果。

3. 关键算法的设计方案

1. **项集族构造**：构造 LALR(1)项集族，合并具有相同核心的 LR(1)项集。
2. **分析表构造**：根据 LALR(1)项集族构造 ACTION 和 GOTO 表。
3. **语法分析**：利用构造的分析表对输入串进行分析，输出分析过程和结果。
4. **找到集合:** 找到每个非终结符的 first 和 follow 集合

关键算法步骤（伪代码）：

```
function ConstructLALR1Table(grammar):
    C = ConstructItemSets(grammar)
    ACTION, GOTO = ConstructParseTable(C)
    return ACTION, GOTO


function ConstructItemSets(grammar):
    C = {InitialItemSet}
    while (new item sets can be added):
        for each item set I in C:
            for each grammar symbol X:
                new_set = Closure(GOTO(I, X))
                if new_set not in C:
                    add new_set to C
    return C


function ConstructParseTable(C):
    for each item set I in C:
        for each item in I:
```

```
            if item is [A -> α•aβ, b]:
                ACTION[I, a] = Shift(I')
            else if item is [A -> α•, a]:
                ACTION[I, a] = Reduce(A -> α)
        for each non-terminal A:
            GOTO[I, A] = J
    return ACTION, GOTO
```

找到 FIRST 和 FOLLOW 集合的过程

```
function ComputeFirstSets(grammar):
        for each terminal t in grammar:
            FIRST(t) = {t}
        for each non-terminal N in grammar:
            FIRST(N) = {}
        changed = true
        while (changed):
            changed = false
            for each production (A -> α) in grammar:
                for each symbol X in α:
                    old_size = size(FIRST(A))
                    FIRST(A) = FIRST(A) union (FIRST(X) - {ε})
                    if ε not in FIRST(X):
                        break
                if ε in FIRST(α):
                    FIRST(A) = FIRST(A) union {ε}
                if size(FIRST(A)) != old_size:
                    changed = true
        return FIRST


    function ComputeFollowSets(grammar, FIRST):
```

```
for each non-terminal N in grammar:
    FOLLOW(N) = {}
FOLLOW(S) = {EOF}    // S is the start symbol
changed = true
while (changed):
    changed = false
    for each production (A -> αBβ) in grammar:
        old_size = size(FOLLOW(B))
        if β is not empty:
            FOLLOW(B) = FOLLOW(B) union (FIRST(β) - {ε})
        if β is empty or ε in FIRST(β):
            FOLLOW(B) = FOLLOW(B) union FOLLOW(A)
        if size(FOLLOW(B)) != old_size:
            changed = true
return FOLLOW
```

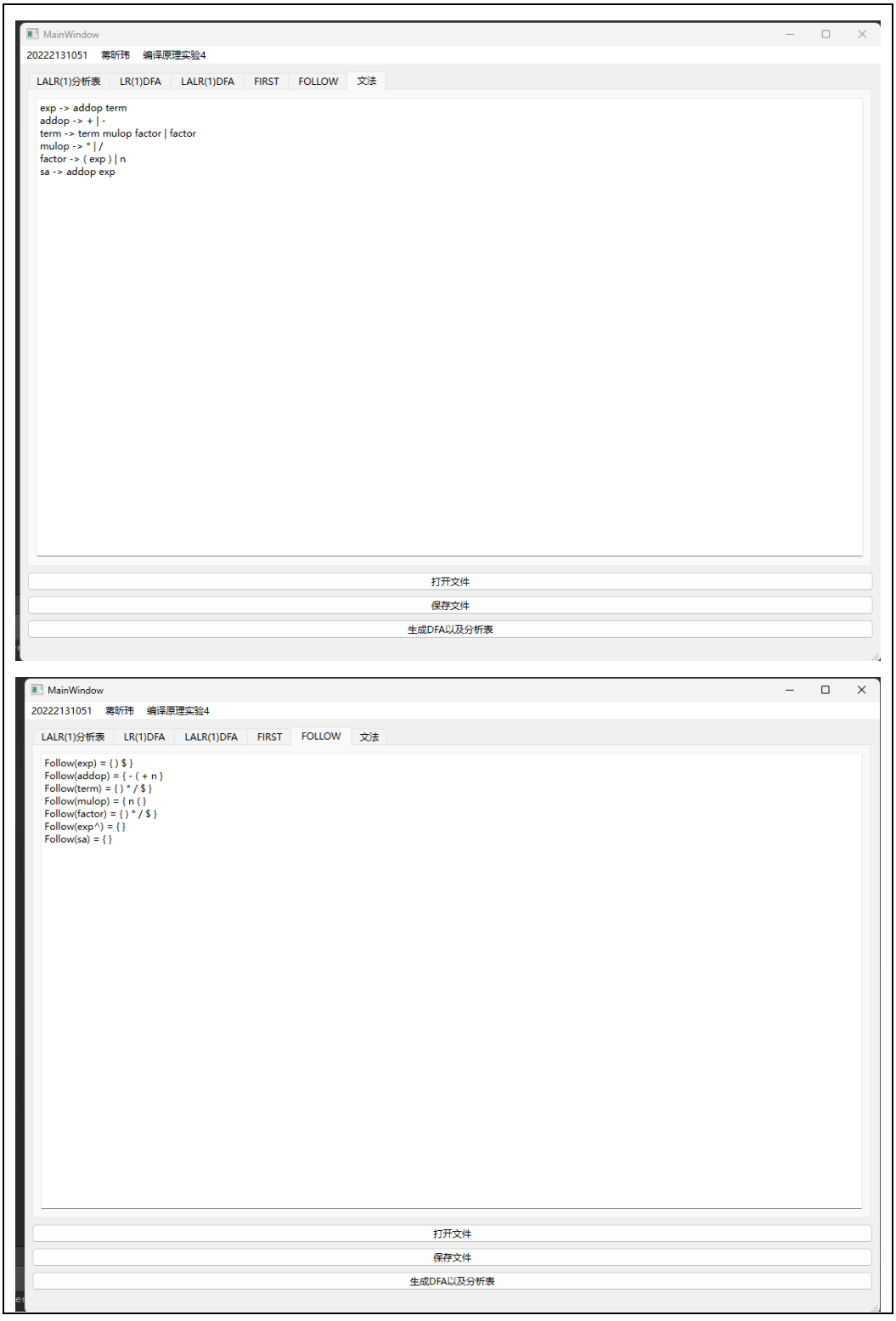
4. 实现及测试内容

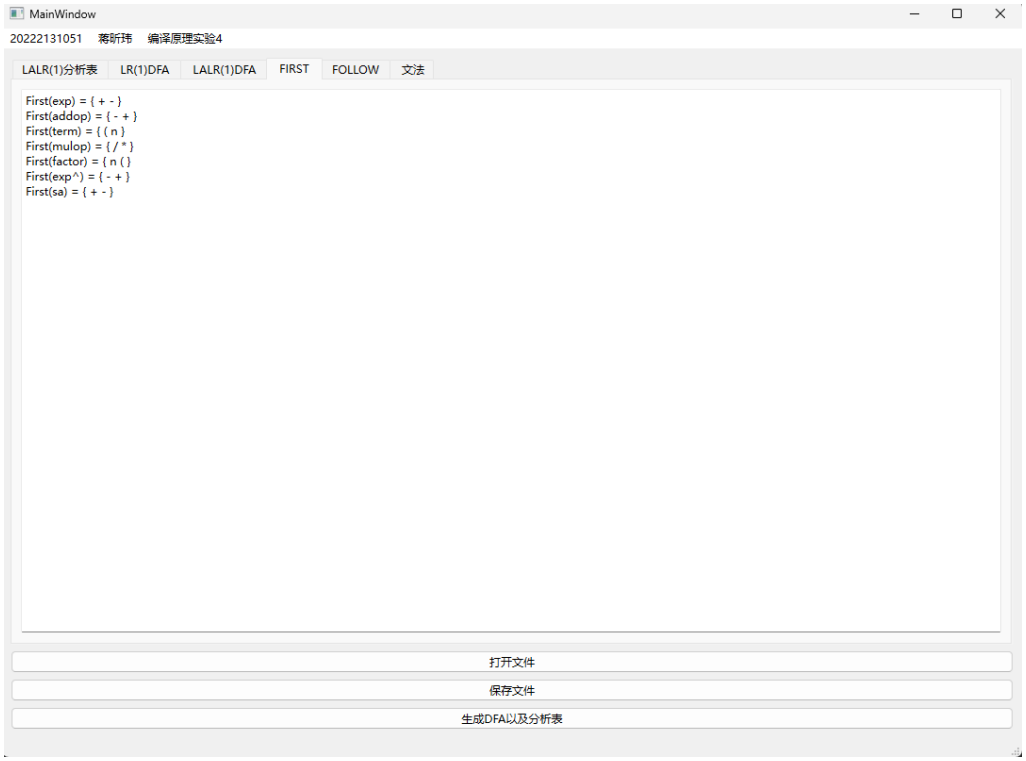
实现内容

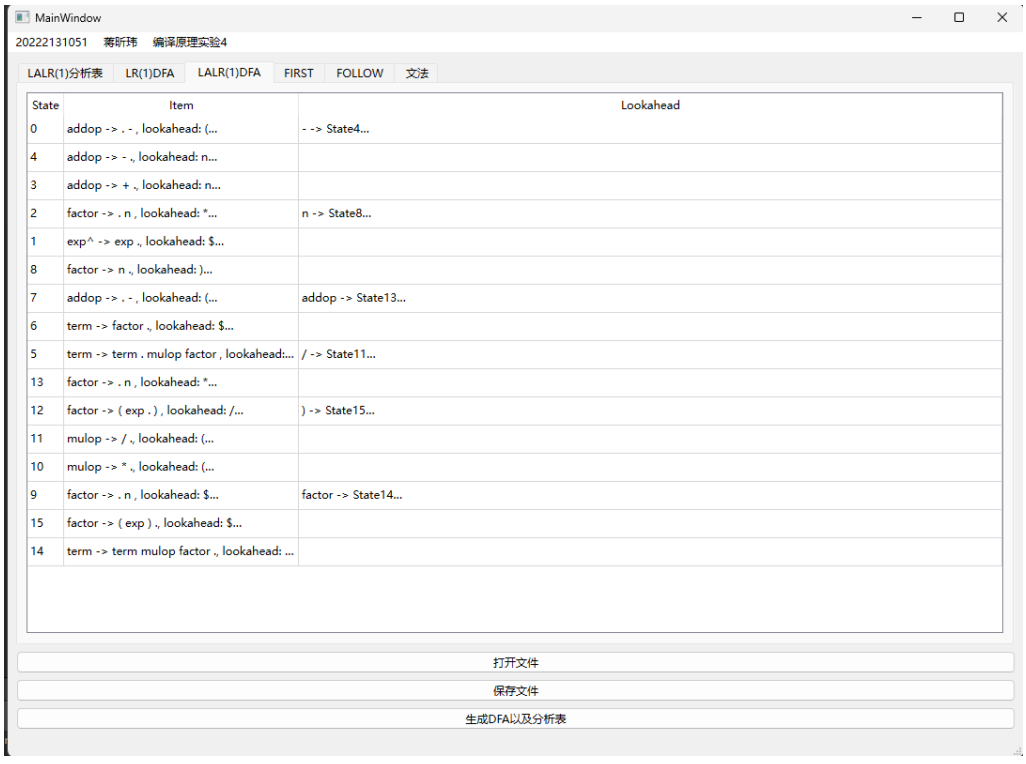
- 语法定义模块：定义文法规则，保存产生式和符号表。
- LALR(1)分析算法实现模块：实现 LALR(1)项集族构造、分析表构造和语法分析过程。
- 主程序模块：整合各模块，完成从输入到输出的整个流程。
- 图形用户界面模块：利用 Qt 框架实现图形界面，展示语法分析的步骤和结果。

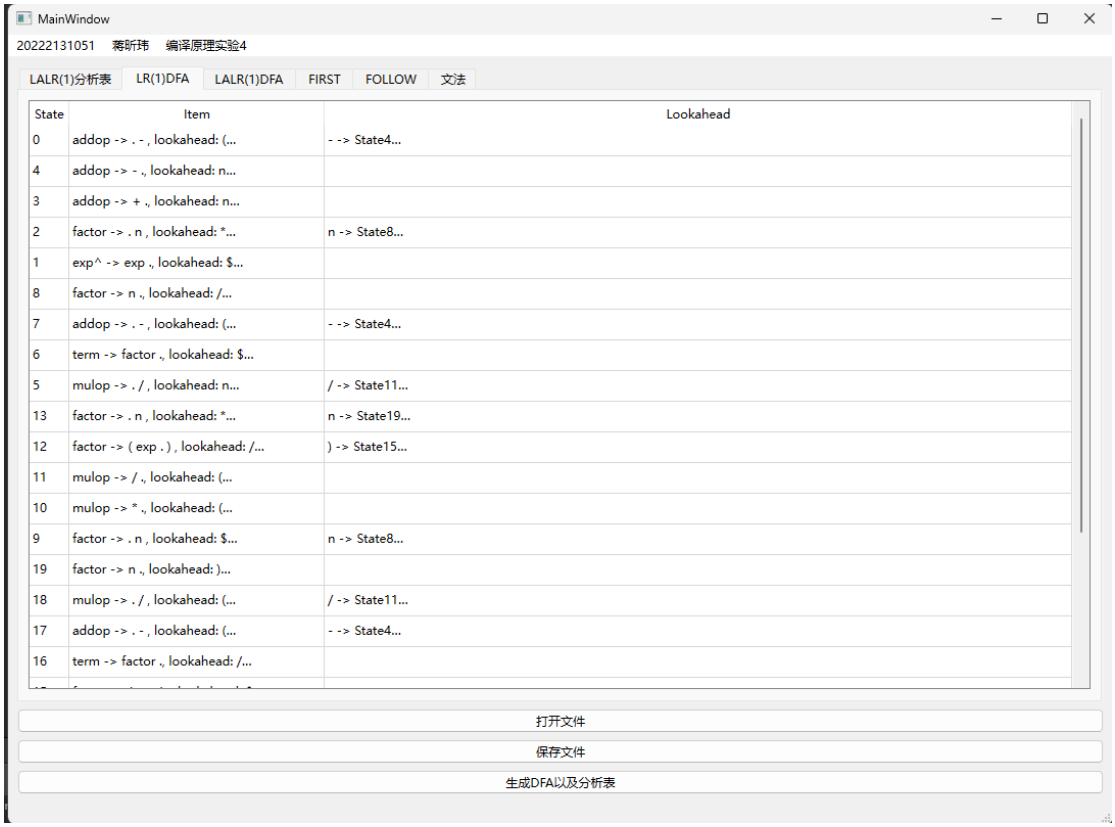
测试内容

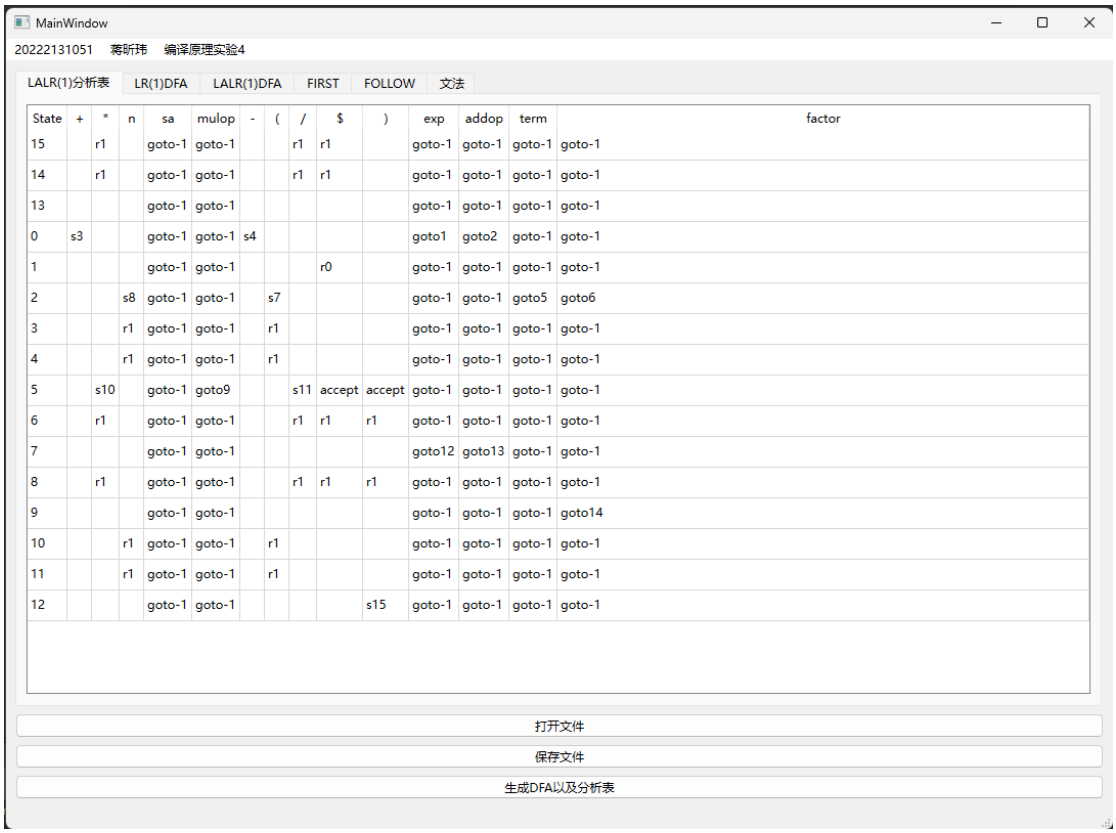
- 使用多组文法规则和输入串进行测试，验证分析器的正确性和鲁棒性。
- 测试界面交互功能，确保用户可以方便地输入文法规则和待分析的字符串，并查看分析结果。


MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
exp -> addop term
addop -> + | -
term -> term mulop factor | factor
mulop -> * | /
factor -> (exp) | n
sa -> addop exp
打开文件
保存文件
生成DFA以及分析表
MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
Follow(exp) = {) $ }
Follow(addop) = { - (+ n }
Follow(term) = {) * / $ }
Follow(mulop) = { n (}
Follow(factor) = {) * / $ }
Follow(exp^) = { }
Follow(sa) = { }
打开文件
保存文件
生成DFA以及分析表


MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
First(exp) = { + - }
First(addop) = { - + }
First(term) = { (n }
First(mulop) = { / * }
First(factor) = { n (}
First(exp^) = { - + }
First(sa) = { + - }
打开文件
保存文件
生成DFA以及分析表


MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
State
Item
Lookahead
0 addop -> . - , lookahead: (... - -> State4...
4 addop -> - ., lookahead: n...
3 addop -> + ., lookahead: n...
2 factor -> . n , lookahead: *... n -> State8...
1 exp^ -> exp ., lookahead: $...
8 factor -> n ., lookahead:)...
7 addop -> . - , lookahead: (... addop -> State13...
6 term -> factor ., lookahead: $...
5 term -> term . mulop factor , lookahead:... / -> State11...
13 factor -> . n , lookahead: *...
12 factor -> (exp .) , lookahead: /...) -> State15...
11 mulop -> / ., lookahead: (...
10 mulop -> * ., lookahead: (...
9 factor -> . n , lookahead: $... factor -> State14...
15 factor -> (exp) ., lookahead: $...
14 term -> term mulop factor ., lookahead: ...
打开文件
保存文件
生成DFA以及分析表


MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
State Item Lookahead
0 addop -> . - , lookahead: (... - -> State4...
4 addop -> - ., lookahead: n...
3 addop -> + ., lookahead: n...
2 factor -> . n , lookahead: *... n -> State8...
1 exp^ -> exp ., lookahead: $...
8 factor -> n ., lookahead: /...
7 addop -> . - , lookahead: (... - -> State4...
6 term -> factor ., lookahead: $...
5 mulop -> . / , lookahead: n... / -> State11...
13 factor -> . n , lookahead: *... n -> State19...
12 factor -> (exp .) , lookahead: /...) -> State15...
11 mulop -> / ., lookahead: (...
10 mulop -> * ., lookahead: (...
9 factor -> . n , lookahead: $... n -> State8...
19 factor -> n ., lookahead:)...
18 mulop -> . / , lookahead: (... / -> State11...
17 addop -> . - , lookahead: (... - -> State4...
16 term -> factor ., lookahead: /...
打开文件
保存文件
生成DFA以及分析表


MainWindow
20222131051 蒋昕玮 编译原理实验4
LALR(1)分析表
LR(1)DFA
LALR(1)DFA
FIRST
FOLLOW
文法
State + * n sa mulop - (/ $) exp addop term factor
15 r1 goto-1 goto-1 r1 r1 goto-1 goto-1 goto-1 goto-1
14 r1 goto-1 goto-1 r1 r1 goto-1 goto-1 goto-1 goto-1
13 goto-1 goto-1 goto-1 goto-1 goto-1 goto-1 goto-1
0 s3 goto-1 goto-1 s4 goto1 goto2 goto-1 goto-1
1 goto-1 goto-1 r0 goto-1 goto-1 goto-1 goto-1
2 s8 goto-1 goto-1 s7 goto-1 goto-1 goto5 goto6
3 r1 goto-1 goto-1 r1 goto-1 goto-1 goto-1 goto-1
4 r1 goto-1 goto-1 r1 goto-1 goto-1 goto-1 goto-1
5 s10 goto-1 goto9 s11 accept accept goto-1 goto-1 goto-1 goto-1
6 r1 goto-1 goto-1 r1 r1 r1 goto-1 goto-1 goto-1 goto-1
7 goto-1 goto-1 goto12 goto13 goto-1 goto-1
8 r1 goto-1 goto-1 r1 r1 r1 goto-1 goto-1 goto-1 goto-1
9 goto-1 goto-1 goto-1 goto-1 goto-1 goto-1 goto-1 goto14
10 r1 goto-1 goto-1 r1 goto-1 goto-1 goto-1 goto-1
11 r1 goto-1 goto-1 r1 goto-1 goto-1 goto-1 goto-1
12 goto-1 goto-1 s15 goto-1 goto-1 goto-1 goto-1
打开文件
保存文件
生成DFA以及分析表