

华南邻轮大学

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: C++ 源代码单词扫描程序(词法分析)

指导老师: 黄煜廉

开课时间: 2020~2021年度第1学期

专 业: 计算机科学与技术

班 级: 18级4班

姓 名: 陈伟卓

学 号: 20182131018

华南师范大学教务处

一、实验内容

设计一个应用软件,以实现将正则表达式-->NFA--->DFA-->DFA最小化-->词法分析程序。

- (1) 要提供一个正则表达式的输入界面,让用户输入正则表达式(可保存、打开保存着正则表达式的文件)
- (2) 需要提供窗口以便用户可以查看转换得到的NFA (用状态转换表呈现即可)
- (3) 需要提供窗口以便用户可以查看转换得到的DFA (用状态转换表呈现即可)
- (4) 需要提供窗口以便用户可以查看转换得到的最小化DFA (用状态转换表呈现即可)
- (5) 需要提供窗口以便用户可以查看转换得到的词法分析程序(该分析程序需要用C语言描述)
- (6) 应该书写完善的软件文档

二、实验目的

掌握简单正则表达式转换NFA,DFA,DFA最小化等流程。掌握词法分析过程。理解正则表达式在编译前端的作用。

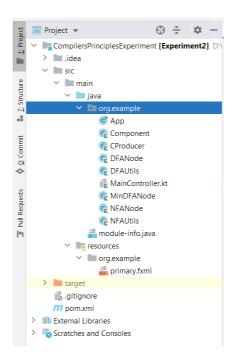
三、实验文档

实现项目分析

需要使用windows界面,核心技术设计文件读写和界面展示。本次实验选用Java/Kotlin作为编码语言,使用JavaFX作为界面展示框架。

实验设计

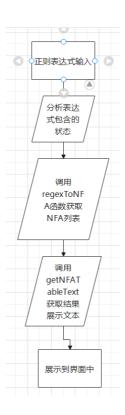
1. 项目结构



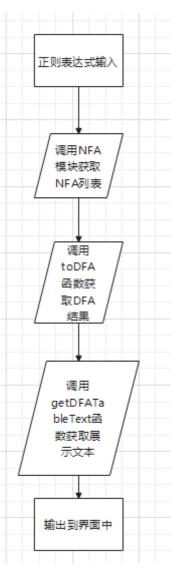
App类为应用启动入口。Component、DFANode、NFANode、MinDFANode为数据类,用于存放各类型数据。MainController负责捕捉界面事件并进行响应。DFAUtils和NFAUtils为工具类,分别处理DFA和NFA操作。primar.fxml为界面文件。CProducer为语法产生器,生成C语言格式的语法分析程序。

2. 软件功能流程图

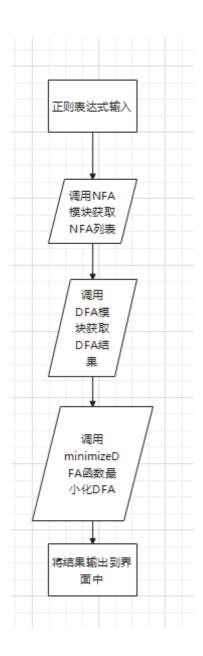
正则表达式转换为NFA功能:



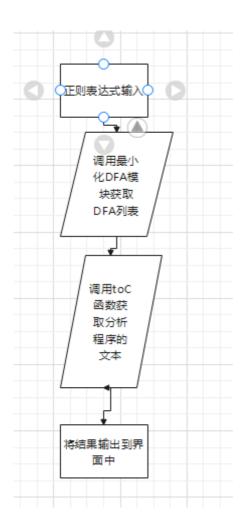
正则表达式转换为DFA功能:



DFA最小化功能:



最小化DFA转语法分析功能:



3. 数据结构选择

NFANode类:

✓ Ç¹ NFANode

nextList: ArrayList < Pair < NFANode, String >> /* = ArrayList < Pair < NFANode, String >> */

V 😉 num: Int

用于存放每个NFA状态形成的节点,成员变量nextList包含该节点指向下一节点集合,同时通过 Pair的方式记录了指向下一节点的方式(Epsilon或者其他状态),成员变量num记录了该节点的编 号,以便后续输出。

Component类:

✓ Ç[™] Component

m 🐿 constructor Component(NFANode = ..., NFANode = ..., String = ...)

headNode: NFANode

tailNode: NFANode

<class initializer>

NFA通过运算后形成一个部件,用于component对象存放。存放表达式的头结点和尾结点。使用Component类是为了方便后续表达式的计算,因为表达式的每一步计算结果都可视为一个Component,头与尾方便进行后续计算。

DFANode类:

```
    □ DFANode
    □ constructor DFANode(HashSet<Int> /* = HashSet<Int> */)
    □ containList: HashSet<Int> /* = HashSet<Int> */
    □ toStatus: HashMap<String, DFANode?> /* = HashMap<String, DFANode?> */
    □ name: String
    □ isEndStatus: Boolean
    □ isBeginStatus: Boolean
    □ equals(Any?): Boolean
    □ hashCode(): Int
```

成员变量containerList为哈希集,用于存放该DFA状态包含的DFA节点编号的集合,主要是为了方便调试和结果展示。

成员变量toStatus为哈希表,

成员变量name为状态名称。isEndStatus为布尔值,记录是否为可以是终结状态。isBeginStatus为布尔值,记录是否为起始状态。

重写了equal和hash方法,用其中的containList作为代表,以便后续用该节点作为hashMap的key值。

MinDFANode类和DFANode类类似,但是多了一个代表节点,表示该最小状态指向的下一最小状态。

4. 核心功能实现

NFAUtils工具类功能:

```
NFAUtils

NFAUtils

NFAUtils

NFAUtils

NFANode> /* = ArrayList<NFANode> /* = ArrayList<NFANode> */, ArrayList<String> /* = ArrayList<NFANode> /* = ArrayList
```

先将正则表达式通过toPostFix函数转为后缀表达式并计算得到一个NFA图,然后对每个节点进行编号。

其中闭包,与、或方法皆为对Component对象的操作,模拟手工做法对后缀表达式进行操作。最后通过getNFATableText获取文本结果。getNFATableText函数中,遍历传入的NFA节点列表,将每个节点列表中的nextList根据关系分类,并进行相应的文本描述,最后返回状态转换表。

DFAUtils工具类功能:

```
✓ Q* DFAUtils
✓ Q* DFAUtils
✓ Q* Companion object of DFAUtils
✓ Q* Companion object of DFAUtils
✓ Q* AnameMap: HashMap < DFANode, String> /* = HashMap < DFANode, String> */ 
⊕ qetMinDFAText(ArrayList < MinDFANode> /* = ArrayList < MinDFANode> */): String
⊕ \( \text{inDFAText(ArrayList < MinDFANode> /* = ArrayList < MinDFANode> */ \): ParrayList < String> */, Int): ArrayList < DFANode> /* = ArrayList < DFANode> */
⊕ \( \text{array List < DFANode> /* = ArrayList < DFANode> */ \): ParrayList < DFANode> */ \)
⊕ \( \text{anameNode(ArrayList < DFANode> /* = ArrayList < DFANode> /* = ArrayList < DFANode> /* = ArrayList < DFANode> /*, Int): Unit
⊕ \( \text{anameNode(ArrayList < DFANode> /* = ArrayList < DFANode> /* | ArrayList < DFANode> /* | ArrayList <
```

先获取每个节点的epsilon闭包。将初始状态的闭包作为第一个状态,加入到队列中。

当队列不空时,弹栈,对该栈顶状态A遍历status表,找到该状态通过当前字符指向的节点的集合 B,若B为空集,则当前状态通过当前字符无通路,修改A的toStatus列表为null。若记录已有DFA 状态的statusExists表中包含B,则修改A的toStatus列表指向B。若记录已有DFA状态的 statusExists表中不包含B,则将B加入到statusExists表中,同时修改A的toStatus列表指向B,将

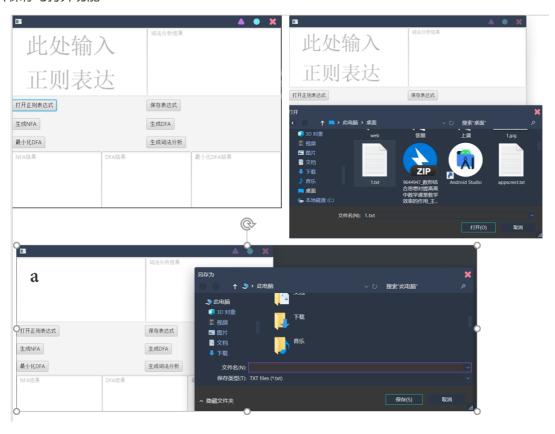
新状态加入到队列中,将A加入到结果列表中。循环上面操作直至栈空。对结果列表进行编号,并标记结束状态,以便后续最小化DFA使用。完成上述操作即可获得NFA到DFA的状态转换,得到一个DFA状态列表。遍历该列表,获取信息即可获得状态转换表。

最小化DFA: 对DFA结果列表按照是否是终结状态进行分类,压栈,加入到targetList中。当栈不空时,弹栈获取栈顶元素A,对A根据字符把指向的状态进行分组,组中的元素形成一个新的集合,如果新集合小于当前原集合且不为空,说明原集合A可以再分,分离出新集合B,将新集合B和去除新集合后的A-B压栈,同时移除targetList中的旧集合,压入新集合。重复上述操作直至栈空。返回最小DFA状态表。

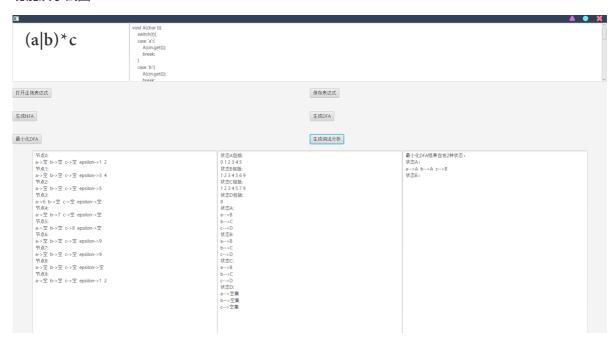
实验测试

主界面及相关功能截图:

文件保存与打开功能



功能演示截图



四、实验总结 (心得体会)

该实验对于数据结构的把握要求较高,对于一些手工做很简单的算法,程序实现起来比较复杂。所幸 Java有较好的Api直接,可以轻松实现较为复杂的遍历操作等。以后要加强编程训练,提升编程效率。

五、参考文献

Kenneth C.Louden: 《编译原理及实践》.2000.机械工业出版社