实验文档

软件13 杨楠

# 编程语言与环境

操作系统：Windows 10

平台：VS Code

语言：C++11

编译：cmake MinGW-w64 g++

# 第一次实验

第一次实验，是根据已给定的NFA，执行输入串，如果输入串被该NFA接受，还需要返回一条接受路径。

## 实现思路

对路径进行回溯，DFS，这里使用了栈来实现。参考了“第一次实验讲解”课件第47页里的内容，也相应地实现了match()函数和backtrace()函数。



上述课件中的伪代码有些瑕疵，比如stack要push的不是input或input[1:]，而是str或str[1:]，等等。

在实验一的代码中，对于stack中的元素和path中的元素，参考了上述课件，用“剩余输入串”来记录。而到了实验三，我修改为了记录当前在字符串的位置index。

## 重难点问题

对于ε转移的死循环的处理（即，可能存在完全由ε转移构成的环）。我在程序中是使用一个set集合来存储某个状态曾经经历过的情况。由于set里元素的互异性，如果发现在该状态下，当前剩余输入串的情况已经经历过了，那么就不将其入栈。我在实验一中是记录当前状态经历过的所有“剩余输入串”，到了实验三修改为了index。

# 第二次实验

第二次实验，是根据正则表达式（所生成的语法分析树）来构造NFA。

## 实现思路

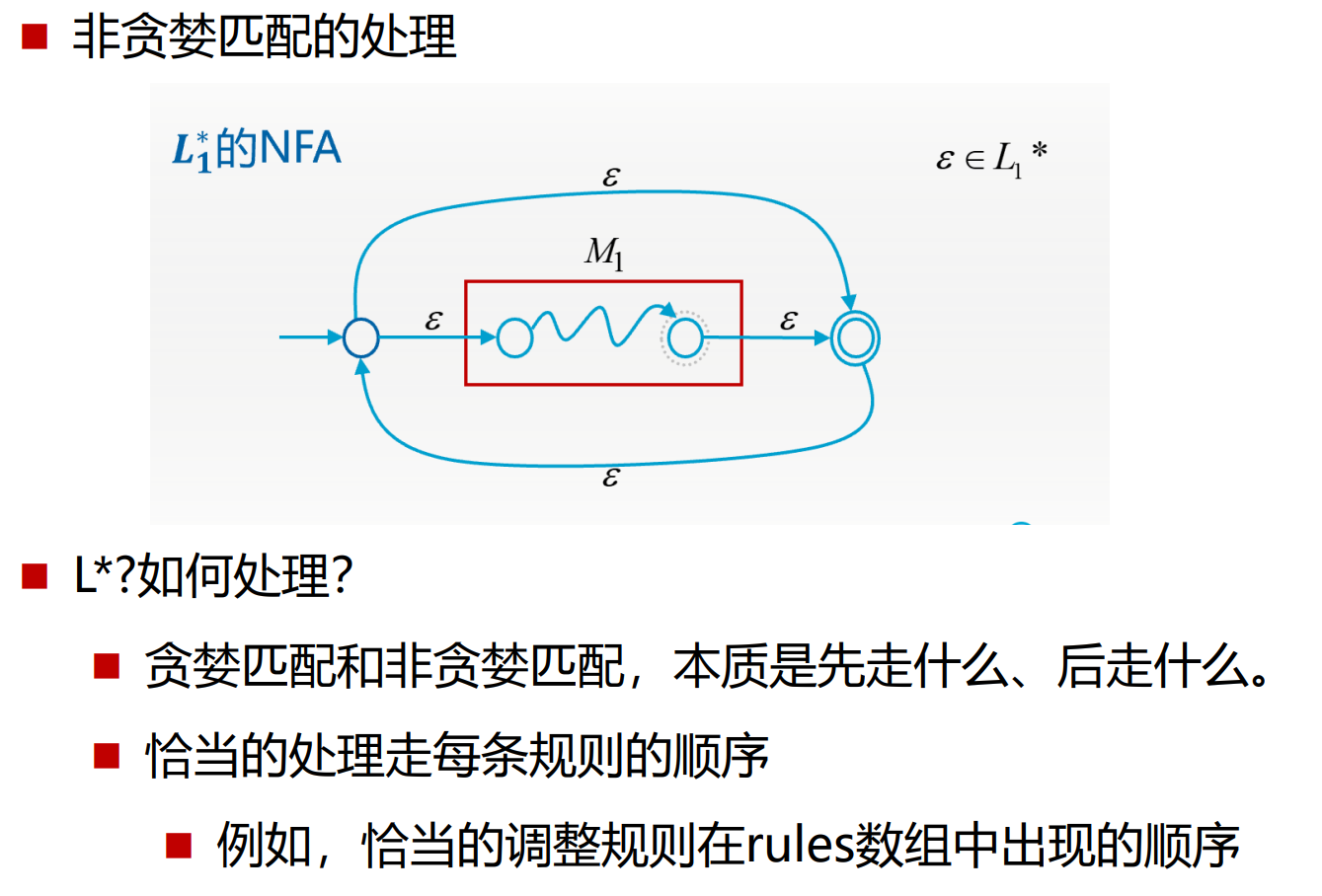
从语法分析树的根节点开始，往下访问，递归构造自动机。编号从0开始，构造的时候编号是总体往后递增的，以最后一个状态作为终态。

一个regex是若干个expression的并。那么从当前的节点往后接这几个自动机的并。最后这些自动机又汇总到一个状态。（这就也使得最后一个状态为唯一的终态）

每个expression是若干个expressionItem的连接，那么相应地构建自动机的连接。

每个expressionItem是由一个normalItem，以及一个可有可无的quantifier组成。根据quantifier的情况，对这部分自动机进行相应的处理。

参考了“第二次实验讲解”课件里的思路，通过添加ε转移实现相应的quantifier，通过修改这些ε转移在rules中的先后顺序，实现贪婪和非贪婪匹配。



每个normalItem是由一个single或者一个group组成。对于single，在自动机添加相应的转移。对于group，则递归处理其中的regex。

由上，设置了compileRegex()函数，compileExpression()函数，compileExpressionItem()函数，compileSingle()函数，参数分别是相应的节点（指针）。在compile()函数中，编译的入口是compileRegex(tree)。之后就是递归调用相应的函数。

关于match()函数，由于只需要返回一次匹配结果，则从头开始依次取text文本的子串，调用exec()，直到返回非空路径path。

## 重难点问题

对于是否贪婪匹配的处理。主要是修改相应规则的出现顺序。由于我DFS是用栈处理，越靠后的规则反而更靠近栈顶，导致越先访问。在实验二，我是在compileExpression()函数中处理的，在实验三，则修改为在compileExpressionItem()中处理。

# 第三次实验

第三次实验是实现更多功能。

## 实现思路

对于捕获分组，我记录了相应的左右括号所对应的状态pair，在匹配时进行处理。涉及区间限定符时，将多次重复的归为一类，匹配时尽量匹配靠后的。

对于多行文本的处理，样例txt的行尾序列是LF，即\n，我是基于此进行的处理。

对于flag，根据s的有无，设置match()函数返回值。根据m的有无，在exec()函数进行处理。

对于anchor，将其作为特殊的ε转移来处理，在exec()函数中特判。

对于rangeQuantifier，参考了实验文档的思路，比如a{2,5}处理成aaa?a?a?，并根据是否贪婪匹配进行处理。

对于matchAll和replaceAll，使用基于当前字符串位置index的处理方式。matchAll时，从前往后扫描，replaceAll时，从后往前替换文本。

修改了对于正则表达式的转义字符的处理。

优化了自动机生成相关的函数，使得状态编号是严格递增，这样就不需要记录当前最晚生成的状态编号，而只需记录当前状态总数即可。

优化了exec()函数。将stack和set改为nfa的成员变量，而非函数局部变量。将stack和set的元素中，关于“剩余输入串”的记录方式，改为记录index。

优化了backtrace()函数，将path元素记录“剩余输入串”的方式，改为记录index。

## 重难点问题

从“剩余输入串”到index的转变。这很有利于处理很多事情，比如anchor，flag，matchAll和replaceAll等等，因为这些是前后文相关的，而不单单是剩余输入串的事情。

对于区间限定符的处理。处理起来比较复杂，需要分不同的情况讨论。