**正规式转NFA转DFA转MFA实验报告**

报告分为两部分，第一部分是代码原理的讲解，第二部分是源代码的展示。

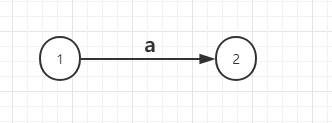
最终实现的自动机，字符集只支持大小写字母以及数字，操作只支持 与、或、闭包运算，支持括号匹配。

**第一部分 代码原理**

1. 正规式转NFA

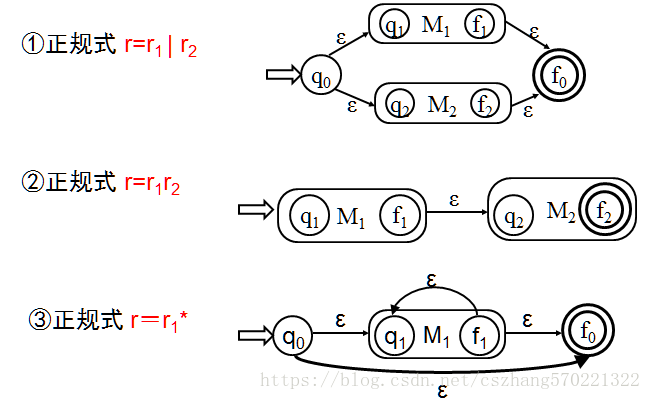
使用 **Thompson算法**由正规式构造NFA ，构造支持三个操作 分别是与运算、或运算 、闭包运算，流程如下：

1 、程序中有两个栈，一个符号栈，用于存储运算符号，一个自动机类栈，用书中的自动机类五元组定义，首先，对于每一个合法的字符构造一个自动机，如 a，构造完成后，压入自动机类栈中，构造后如下图



2、如果遇到左括号就将之后遇到的运算符进行压栈，遇到右括号的时候弹出运算符进行运算，最后把计算出的自动机类压入栈，这样可以保证括号括起来的自动机已经成为一个整体。

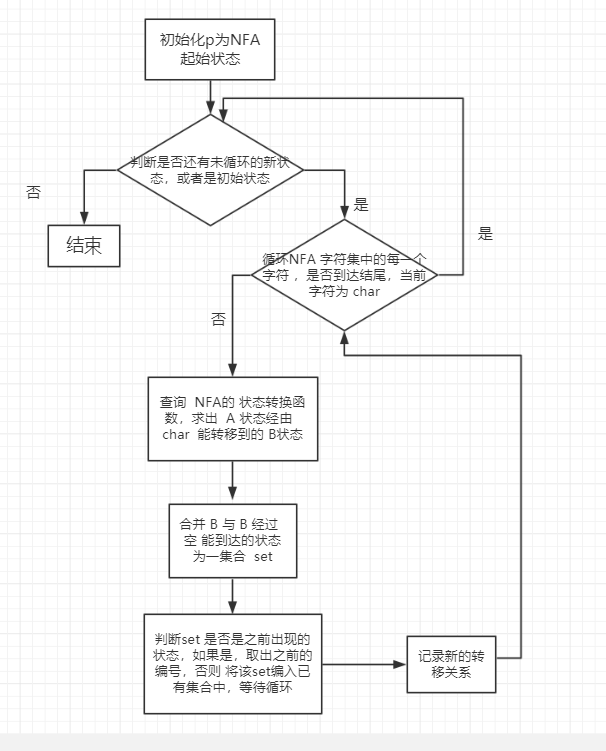
3、如果遇到 与、或、闭包运算符需要合并的就执行下图的操作



二、NFA 转 DFA

NFA 转DFA 使用了**子集构造法**，目的在于消除NFA中新建的空边。

代码流程图如下：



三、DFA 转 MFA

DFA转MFA，就是要消除自动机的无用状态和合并等价状态，本实验中，我并未完成消除自动机的无用状态，只完成了使用 **分割法** 合并等价状态。（所以假设自己测试的正规式转化的DFA 不含无用状态）

流程和书上叙述一致，只是实现了简单的模拟。

**第二部分 源代码展示**

首先介绍实现的类

一共实现了5个类， 名称和功能分别如下：

Automata # 基本自动机类 定义为书中自动机的五元组

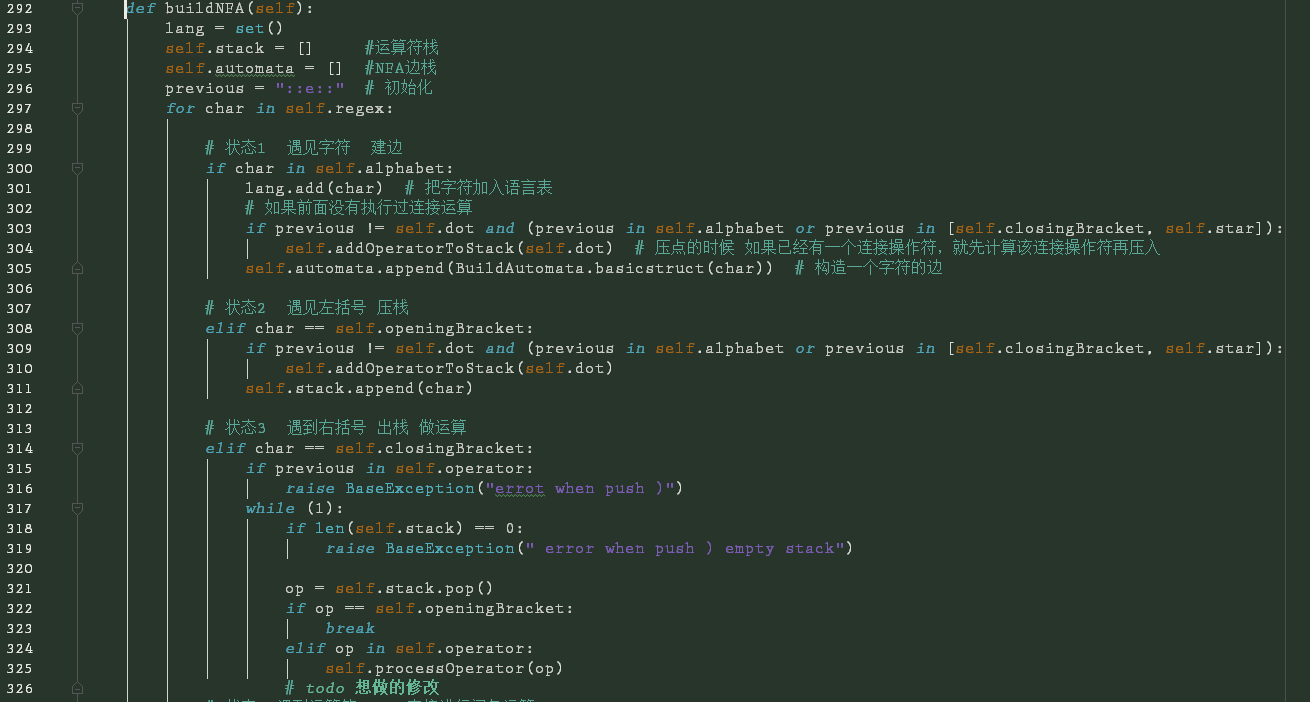
BuildAutomata #在基本自动机的基础上，实现了自动机的与、或、闭包运算

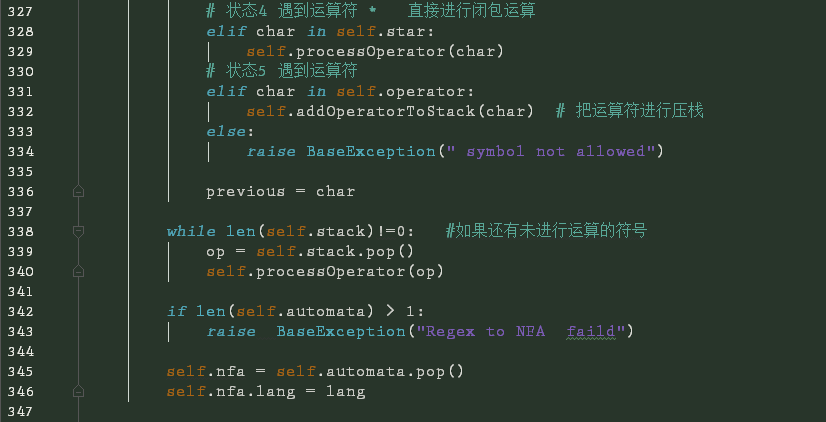
RegextoNFA #实现了 正规式转NFA

NFAtoDFA # 实现了 NFA 转 DFA

DFAtoMFA # 实现了 DFA 转 MFA

一、正规式转NFA核心代码





二、NFA 转 DFA核心代码



三、DFA 转MFA核心代码

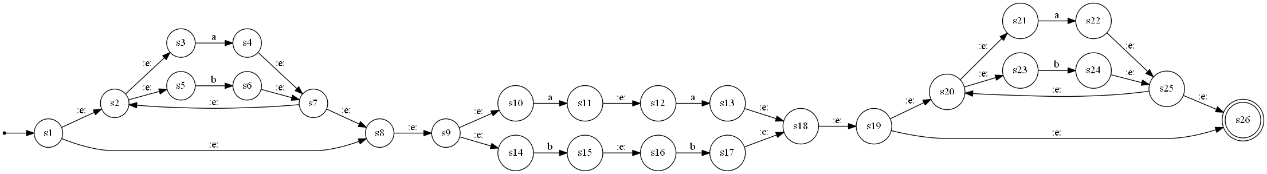


测试正规式

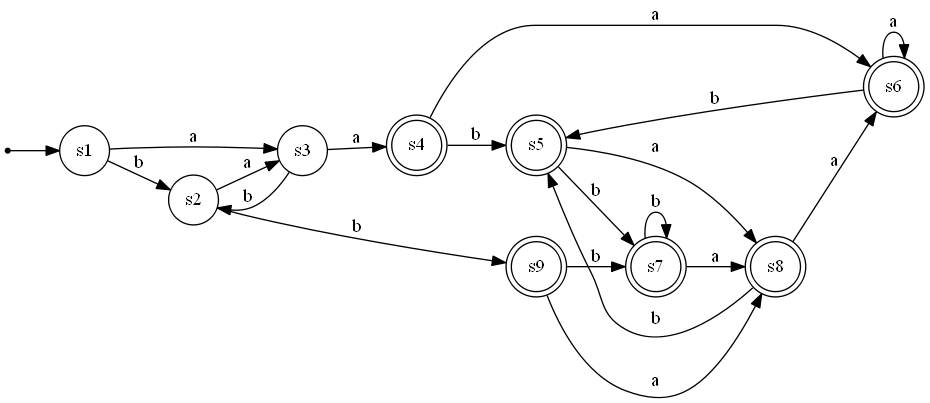
inp ="(a|b)\*.(a.a|b.b).(a|b)\*"

使用graghviz 函数库生成的图片如下：

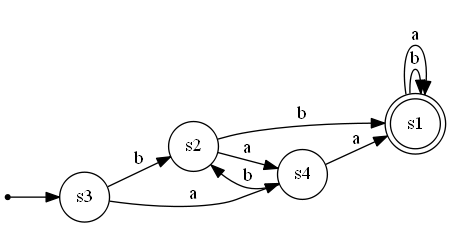
NFA



DFA



MFA



具体解释请见代码中的注释