苏 州 大 学 实 验 报 告

院系	计算机学院	年级专业	21 计科	姓名	方浩楠	学号	2127405048
课程名称	编译原理课程实践					成绩	
指导教师	王中卿	同组实验者	无		实验日期	2023.9.19	

实 验 名 称 从 NFA 到 DFA

一. 实验目的

1. 基于子集构造法，从 NFA 构建 DFA
2. 给定一个字符串通过遍历的 DFA 判断是否可到达结束状态

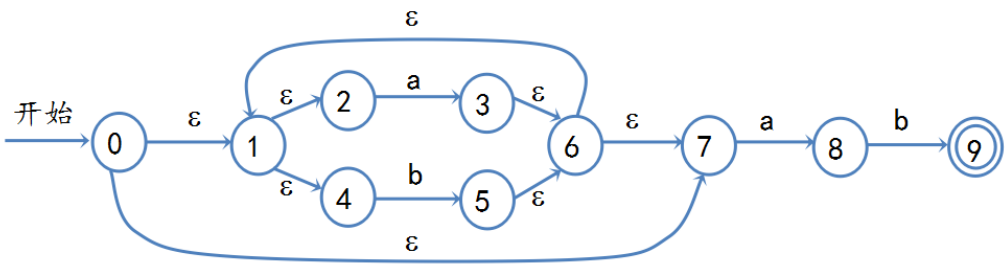
二. 实验内容

输入为基于 NFA 的状态转换表，如下图所示（和上次课得到的 NFA 状态转换表相同）

起始状态	符号	结束状态
0	a	1
1	b	2

2）输出为基于 DFA 的状态转换表

核心是要实现 eclosure（S）函数，即从状态集合 S 出发经过 ε 可以达到的所有状态集合。
完成(a|b)*ab 的 NFA 到 DFA 的转换，NFA 如下图所示：



三. 实验步骤和结果

子集构造法三个核心函数

$\epsilon - closure(s)$: 从 NFA 的状态 s 出发，只用 ϵ 转换能到达的 NFA 状态集合

$\epsilon - closure(T)$: 从 NFA 的状态集合 T 中每个状态出发，只用 ϵ 转换就能到达的状态的集合

$move(T, a)$: 从 NFA 的状态集合 T 中每个状态出发，通过 a 能到达的所有状态集合

$move(T, a)$: 从 NFA 的状态集合 T 中每个状态出发，通过 a 能到达的所有状态集合
$input : NFA$
$output : DFA$
D的转换表 : $Dtran$, 状态集 : $Dstates$
如果D的某个状态 V 至少包含一个 N 的接收状态 那么 V 是 D 的一个接收状态

伪代码如下:

Algorithm 1: converter NFA to DFA

```
1  将 ε − closure(s0) 添加到 Dstates 中, 并且是未标记;
2  while Dstates 中有未标记的状态 T do
3      标记 T;
4      for 每个输入符号 symbol do
5          U = ε − closure(move(T, symbol));
6          if U 不在 Dstates 中 then
7              将 U 作为未标记状态添加到 Dstates;
8          end
9          Dtran[T, symbol] = U ;
10     end
11 end
```

但是在实际转换时, 需要考虑更多的问题, 比如当前状态 U 是否为 DFA 的终止状态, 如果为终止状态需要对该终止状态进行特殊标记 (因为在进行表达式的匹配时需要判断最终是否是在终止状态上因此需要标记终止状态) 修改后的伪代码如下:

Algorithm 2: convert NFA to DFA

```
1  将 ε − closure(s0) 添加到 Dstates 中, 并且是未标记;
2  while Dstates 中有未标记的状态 T do
3      标记 T;
4      if 对于每个输入符号 symbol then
           // 判断状态 T 是否为终止状态
5          ε − closure(s0) 都在 Dstates 中;
6          将 Dtran[T, symbol] 标记为终止状态;
7      end
8      for 每个输入符号 symbol do
9          U = ε − closure(move(T, symbol));
10         if U 不在 Dstates 中 then
11             将 U 作为未标记状态添加到 Dstates;
12         end
13         Dtran[T, symbol] = U ;
14     end
15 end
```

给定正则表达式和字符串, 判断该字符串是否能被改正则表达式匹配的完整流程:

Algorithm 3: regular expression matching by DFA

Data: string,regular expression

Result: Matching result

```
将正则表达式转换为 NFA; // 实验 3 的内容
将 NFA 转换为 DFA; // Algorithm2 的内容
current_state ← A; // 初始状态为 A
for string 中的每一个字母 char do
    for Dtran 中的每一个转换 tran do
        if current_state = tran[0] and char = tran[1] then
            current_state = tran[2];
            // 在 current_state 状态时，通过符号 char，到达新的状态，因此更新当前状态
            break;
        end
    end
end
return current_state is DFA 的终止状态
```

实验结果:

Input: "abab","(a|b)*ab"

Output:True

```
Please enter the string to be matched: (Default = abab)

Please enter the regular expression: (Default = (a|b)*ab)

string is   "abab"
regular_expression is   "(a|b)*ab"
Do the string and re match?     True
```

Input: "aab", "(a|b)*ab"

Output: True

```
Please enter the string to be matched: (Default = abab)
aab
Please enter the regular expression: (Default = (a|b)*ab)

string is   "aab"
regular_expression is   "(a|b)*ab"
Do the string and re match?     True
```

Input: "aabbab", "(a|b)*ab"
Output: True

```
Please enter the string to be matched: (Default = abab)
aabbab
Please enter the regular expression: (Default = (a|b)*ab)

string is   "aabbab"
regular_expression is   "(a|b)*ab"
Do the string and re match?     True
```

代码在压缩包中,只需运行 converter.py 即可
可执行文件 converter 为 UNIX 可执行文件,需要在 Linux 或 macOS 环境中运行

四. 实验总结

本次试验让我了解了 NFA 到 DFA 的转换,以及正则表达式到 DFA 的转换,同时让我了解到了如何构建正则表达式从而实现对字符串的匹配