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	2023.9.18	

一、 实验目的

基于 MYT 算法，实现将正则表达式转换为 NFA。需要实现以下正则表达式符号：
a
ab
a|b
(ab)
a*
(ab)*

二、 实验内容

1. 输入为正则表达式，输出为一个三元组序列，每个三元组为:（起始状态，符号，结束状态）举例：

1）ab

起始状态	符号	结束状态
0	a	1
1	b	2

2）ab*

起始状态	符号	结束状态
0	a	1
1	b	1

三、 实验步骤和结果

本次试验主要需要匹配”*”,”|”,”(”,”)”四个操作符.

1.1 匹配左右括号:

思路:
匹配到左括号时,需要将左右括号中的内容开始添加到暂存栈中,同时结果中要添加状态 T

到状态 K+1 的转换,转换后 K 加一,T 变为 K
若匹配到右括号,则结果中添加状态 T 到暂存栈中的栈顶元素的状态的转换,同时添加暂存栈的栈顶元素的状态到状态 K+1 的转换,转换后 K 加一,T 变为 K,同时弹出栈顶元素

伪代码:

Algorithm 1: 匹配左右括号

if 当前字符 == 左括号 **then**
    暂存栈.*appand*(当前字符);
    最终结果.*appand*($T \xrightarrow{eps} K+1$);
    $K \leftarrow K+1$;
    $T \leftarrow K$;
end
if 当前字符 == 右括号 *and* 当前字符的下一个操作符 $\neq *$ **then**
    最终结果.*appand*($T \xrightarrow{eps}$ 暂存栈的栈顶元素);
    最终结果.*appand*(暂存栈的栈顶元素 $\xrightarrow{eps} K+1$);
    $K \leftarrow K+1$;
    $T \leftarrow K$;
    弹出暂存栈的栈顶元素;
end

1.2 匹配*和|

思路:
匹配到一个字母后,若下一个字符为*,则根据 thompson 构造法进行构造
匹配到一个字母后,若下一个字符为|,首先需要获取到符号|后面到的字符,然后根据 thompson 构造法进行构造,详细过程在伪代码中
伪代码:

Algorithm 2: 匹配 * 和 | 符号

if 当前字符是小写字母 **then**
  if 下一个字符是 * **then**
    最终结果.$append(T \xrightarrow{eps} K+1)$;
    最终结果.$append(K+1 \xrightarrow{\text{当前字符}} K+1)$;
    最终结果.$append(K+1 \xrightarrow{eps} K+2)$;
    $K \leftarrow K+2$;
    $T \leftarrow K$;
  else if 下一个字符为 | **then**
    最终结果.$append(T \xrightarrow{eps} K+1)$;
    最终结果.$append(K+1 \xrightarrow{\text{当前字符}} K+3)$;
    最终结果.$append(K+3 \xrightarrow{eps} K+5)$;
    最终结果.$append(T \xrightarrow{eps} K+2)$;
    最终结果.$append(K+2 \xrightarrow{\text{当前字符的下下个字符}} K+4)$;
    最终结果.$append(K+4 \xrightarrow{eps} K+5)$;
    $K \leftarrow K+5$;
    $T \leftarrow K$;
    当前字符向后移动两位
  end
  else
    最终结果.$append(T \xrightarrow{eps} K+1)$ $K \leftarrow K+1$;
    $T \leftarrow K$;
  end
end

对正则表达式的解析:
Input:ab
Output:

```
/Users/fanghaonan/文件/学习/编译原理/基础算法/实验3/venv/bin/python /Users/fanghaonan/文件
输入的字符串为ab
0   [a] 1
1   [b] 2
```

Input:a*b
Output:

```
/Users/fanghaonan/文件/学习/编译原理/基础算法/实验3/venv/bin/python /Users/fanghaonan
输入的字符串为a*b
0   [eps]   1
1   [a] 1
1   [eps]   2
2   [b] 3
```

Input: a（a|b）c
Output:

```
输入的字符串为a(a|b)c
0   [a] 1
1   [eps]   2
2   [eps]   3
3   [a] 5
5   [eps]   7
2   [eps]   4
4   [b] 6
6   [eps]   7
7   [c] 8
```

Input:a(a|b)*c
Output:

```
输入的字符串为a(a|b)*c
0   [a] 1
1   [eps]   2
2   [eps]   3
3   [a] 5
5   [eps]   7
2   [eps]   4
4   [b] 6
6   [eps]   7
7   [eps]   1
1   [eps]   8
8   [c] 9
```

四、 实验总结

本次试验让我了解到了如何将正则表达式转换为 NFA,并且了解了如何匹配正则表达式中的左右括号,星号和或符号

PS:源代码:

```
#! /usr/bin/env python

# coding=utf-8




S = []    # Status

ST = []    # temp status

T = 0

K = 0



input = 'a(a|b)*c'

n = len(input)




def get_current(input, i):

        return input[i]




def get_next(input, i):
```

```
        if i < len(input) - 1:
                return input[i + 1]
        else:
                return ''



def add_to_status(t0, ch, t1):
        S.append((t0, ch, t1))



def check_vocab(ch):
        if 'a' <= ch <= 'z':
                return True
        else:
                return False



def check_union(ch):
        if i < len(input) - 1:
                return input[i + 1] == '|'
        else:
                return False
```


教务处制

```
i = 0
while i < len(input):
    ch = get_current(input, i)
    sym = get_next(input, i)

    if check_vocab(ch) == True:
        if sym == '*':
            add_to_status(T, 'eps', K + 1)
            add_to_status(K + 1, ch, K + 1)
            add_to_status(K + 1, 'eps', K + 2)
            K = K + 2
            T = K
        elif check_union(ch):
            add_to_status(T, 'eps', K + 1)
            add_to_status(K + 1, ch, K + 3)
            add_to_status(K + 3, 'eps', K + 5)
            add_to_status(T, 'eps', K + 2)
            add_to_status(K + 2, get_next(input, i + 1), K + 4)
            add_to_status(K + 4, 'eps', K + 5)
            K = K + 5
```

```
                T = K
                i += 2    # skip next character
            else:
                add_to_status(T, ch, K + 1)
                K = K + 1
                T = K

    if ch == '(':
        ST.append(T)
        add_to_status(T, 'eps', K + 1)
        K = K + 1
        T = K

    if ch == ')':
        if sym != '*':
            pass
        else:
            add_to_status(T, 'eps', ST[-1])
            add_to_status(ST[-1], 'eps', K + 1)
            K = K + 1
            T = K
            ST.pop()
```

```
        i += 1


print(f"输入的字符串为{input}")
for t0, ch, t1 in S:
        print('%s\t[%s]\t%s' % (t0, ch, t1))
```


教务处制