

EXP2: CONVERSION FROM REGULAR EXPRESSION TO NFA

AIM: To write a program for converting Regular Expression to NFA.

ALGORITHM:

1. Start
2. Get the input from the user
3. Initialize separate variables and functions for Postfix, Display and NFA
4. Create separate methods for different operators like +, *.
5. By using Switch case Initialize different cases for the input
6. For ' .' operator Initialize a separate method by using various stack functions do the same for the other operators like ' * ' and ' + '.
7. Regular expression is in the form like a.b (or) a+b
8. Display the output
9. Stop

PROGRAM:

```
transition_table = [ [0]*3 for _ in range(20) ]

re = input("Enter the regular expression: ")
re += " "

i = 0
j = 1
while(i<len(re)):
    if re[i] == 'a':
        try:
            if re[i+1] != '|' and re[i+1] != '*':
                transition_table[j][0] = j+1
                j += 1
            elif re[i+1] == '|' and re[i+2] == 'b':
                transition_table[j][2] = ((j+1)*10)+(j+3)
                j+=1
                transition_table[j][0]=j+1
                j+=1
                transition_table[j][2]=j+3
                j+=1
                transition_table[j][1]=j+1
                j+=1
                transition_table[j][2]=j+1
```

```

        j+=1
        i=i+2
    elif re[i+1]=='*':
        transition_table[j][2]=((j+1)*10)+(j+3)
        j+=1
        transition_table[j][0]=j+1
        j+=1
        transition_table[j][2]=((j+1)*10)+(j-1)
        j+=1
    except:
        transition_table[j][0] = j+1

elif re[i] == 'b':
    try:
        if re[i+1] != '|' and re[i+1] != '*':
            transition_table[j][1] = j+1
            j += 1
        elif re[i+1]=='|' and re[i+2]=='a':
            transition_table[j][2]=((j+1)*10)+(j+3)
            j+=1
            transition_table[j][1]=j+1
            j+=1
            transition_table[j][2]=j+3
            j+=1
            transition_table[j][0]=j+1
            j+=1
            transition_table[j][2]=j+1
            j+=1
            i=i+2
        elif re[i+1]=='*':
            transition_table[j][2]=((j+1)*10)+(j+3)
            j+=1
            transition_table[j][1]=j+1
            j+=1
            transition_table[j][2]=((j+1)*10)+(j-1)
            j+=1
    except:
        transition_table[j][1] = j+1

elif re[i]=='e' and re[i+1]!='|' and re[i+1]!='*':
    transition_table[j][2]=j+1
    j+=1

elif re[i]=='\'' and re[i+1]=='*':

    transition_table[0][2]=((j+1)*10)+1
    transition_table[j][2]=((j+1)*10)+1
    j+=1

i +=1

print ("Transition function:")
for i in range(j):
    if(transition_table[i][0]!=0):
        print("q[{0},a]-->{1}".format(i,transition_table[i][0]))
    if(transition_table[i][1]!=0):
        print("q[{0},b]-->{1}".format(i,transition_table[i][1]))

```

```

if(transition_table[i][2]!=0):
    if(transition_table[i][2]<10):
        print("q[{0},e-->{1}".format(i,transition_table[i][2]))
    else:
        print("q[{0},e-->{1} & {2}".format(i,int(transition_table[i][2]/10),transition_table[i][2]%10))

```

INPUT:

(a|b)*abb

OUTPUT:

The image shows two screenshots of the OnlineGDB beta IDE. The top screenshot shows the code being executed, which is a Python program that takes a regular expression as input and generates a transition table for an NFA. The bottom screenshot shows the output of the program, which is the transition table for the regular expression (a|b)*abb.

Code (main.py):

```

1 transition_table = [ [0]*3 for _ in range(20) ]
2
3 re = input("Enter the regular expression: ")
4 re += " "
5
6 i = 0
7 j = 1
8 while(i<len(re)):
9     if re[i] == 'a':
10        try:
11            if re[i+1] != '|' and re[i+1] != '*':
12                transition_table[j][0] = j+1
13                j += 1
14            elif re[i+1] == '|' and re[i+2] == 'b':
15                transition_table[j][2] = ((j+1)*10)+(j+3)
16                j+=1
17                transition_table[j][0]=j+1
18                j+=1
19                transition_table[j][2]=j+3
20                j+=1
21                transition_table[j][1]=j+1
22                j+=1
23                transition_table[j][2]=j+1
24                j+=1
25            i=i+2

```

Input:

```

Enter the regular expression : (a|b)*abb

```

Output (Transition function):

```

q[0,e]-->7 & 1
q[1,e]-->2 & 4
q[2,a]-->3
q[3,e]-->6
q[4,b]-->5
q[5,e]-->6
q[6,e]-->7 & 1
q[7,a]-->8
q[8,b]-->9
q[9,b]-->10

```

...Program finished with exit code 0
Press ENTER to exit console.

RESULT:

The program to convert regular expressions to NFA was implemented successfully.