

LAB EXP2: Regular Expression to NFA

AIM:

To write a program to convert a Regular Expression into NFA.

ALGORITHM:

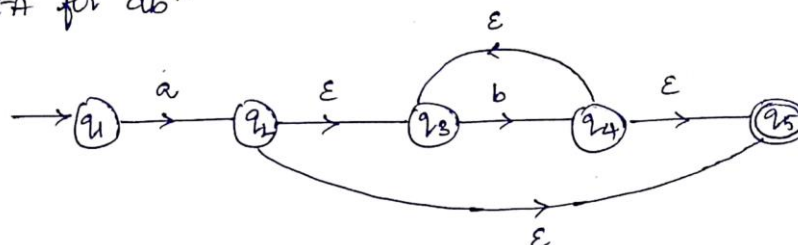
Input – A Regular Expression R

Output – Transition Function

1. First decompose R into the primitive components.
2. For each component we construct a finite automata. For eg: ab^* .
3. Having constructed components for the basic regular expressions, we proceed to combine them in ways that correspond to the way compounded regular expressions.
4. These are formed from small regular expressions.

SOLUTION:

NFA for ab^*



Transition function:-

	a	b	ε
→ q ₁	{q ₂ }	—	—
q ₂	—	—	{q ₃ , q ₅ }
q ₃	—	{q ₄ }	—
q ₄	—	—	{q ₃ , q ₅ }
* q ₅	—	—	—

$$\delta[1, a] \rightarrow \{q_2\}$$

$$\delta[2, \epsilon] \rightarrow \{q_3, q_5\}$$

$$\delta[3, b] \rightarrow \{q_4\}$$

$$\delta[4, \epsilon] \rightarrow \{q_3, q_5\}$$

CODE:

```
#include<stdio.h>

#include<string.h>

int main()
{
    char reg[20];
    int q[20][3],i,j,len,a,b;
    for(a=0;a<20;a++)
    {
        for(b=0;b<3;b++)
        {
            q[a][b]=0;
        }
    }
    scanf("%s",reg);
    len=strlen(reg);
    i=0;
j=1;
    while(i<len)
    {
        if(reg[i]=='a'&&reg[i+1]!='|'&&reg[i+1]!='*')
        {
            q[j][0]=j+1;
            j++;
        }
        if(reg[i]=='b'&&reg[i+1]!='|'&&reg[i+1]!='*')
        {
            q[j][1]=j+1;
            j++;
        }
    }
}
```

```

}
if(reg[i]=='e'&&reg[i+1]!='|'&&reg[i+1]!='*')
{
    q[j][2]=j+1;
    j++;
}
if(reg[i]=='a'&&reg[i+1]=='|'&&reg[i+2]=='b')
{
    q[j][2]=((j+1)*10)+(j+3);
    j++;
    q[j][0]=j+1;
    j++;
    q[j][2]=j+3;
    j++;
    q[j][1]=j+1;
    j++;
    q[j][2]=j+1;
    j++;
    i=i+2;
}
if(reg[i]=='b'&&reg[i+1]=='|'&&reg[i+2]=='a')
{
    q[j][2]=((j+1)*10)+(j+3);
    j++;
    q[j][1]=j+1;
    j++;
    q[j][2]=j+3;
    j++;
    q[j][0]=j+1;
    j++;

```

```

        q[j][2]=j+1;
        j++;
        i=i+2;
    }
    if(reg[i]=='a'&&reg[i+1]=='*')
    {
        q[j][2]=((j+1)*10)+(j+3);
        j++;
        q[j][0]=j+1;
        j++;
        q[j][2]=((j+1)*10)+(j-1);
        j++;
    }
    if(reg[i]=='b'&&reg[i+1]=='*')
    {
        q[j][2]=((j+1)*10)+(j+3);
        j++;
        q[j][1]=j+1;
        j++;
        q[j][2]=((j+1)*10)+(j-1);
        j++;
    }
    if(reg[i]=='')&&reg[i+1]=='*')
    {
        q[0][2]=((j+1)*10)+1;
        q[j][2]=((j+1)*10)+1;
        j++;
    }
    i++;
}

```

```

printf("Transition function \n");

for(i=0;i<=j;i++)
{
    if(q[i][0]!=0)
        printf("\n q[%d,a]-->%d",i,q[i][0]);
    if(q[i][1]!=0)
        printf("\n q[%d,b]-->%d",i,q[i][1]);
    if(q[i][2]!=0)
    {
        if(q[i][2]<10)
            printf("\n q[%d,e]-->%d",i,q[i][2]);
        else
            printf("\n q[%d,e]-->%d & %d",i,q[i][2]/10,q[i][2]%10);
    }
}

return 0;
}

```

OUTPUT:

The screenshot shows a C++ IDE with the following code in the main.c file:

```

69     q[j][2] = a;
70     j++;
71     Transition function
72     if (reg[i] == 1)
73     {
74         q[1,a]-->2
75         q[2,e]-->3 & 5
76         q[3,b]-->4
77         q[4,e]-->5 & 3
78         j++;
79         q[j][2] = a;
80         j++;
81     }
82     if (reg[i] == 2)
83     {
84         q[0][2] = a;
85         q[1][2] = a;
86         j++;
87     }
88     i++;
89     printf("Transition function \n");
90     for (i=0; i<=j; i++)
91     {
92         if (q[i][0] != 0)
93             printf("\n q[%d,a]-->%d", i, q[i][0]);
94         if (q[i][1] != 0)
95             printf("\n q[%d,b]-->%d", i, q[i][1]);
96         if (q[i][2] != 0)
97         {
98             if (q[i][2] < 10)
99                 printf("\n q[%d,e]-->%d", i, q[i][2]);
100             else
101                 printf("\n q[%d,e]-->%d & %d", i, q[i][2]/10, q[i][2]%10);
102         }
103     }
104     return 0;
105 }
106

```

The output window shows the following text:

```

Transition function
Process returned 0 (0x0)   execution time : 5.867 s
Press any key to continue.

```

RESULT:

Hence, executed the program converting a Regular Expression into NFA.