RA1911033010126

BVS Satwika

LAB EXP2: Regular Expression to NFA

AIM:

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

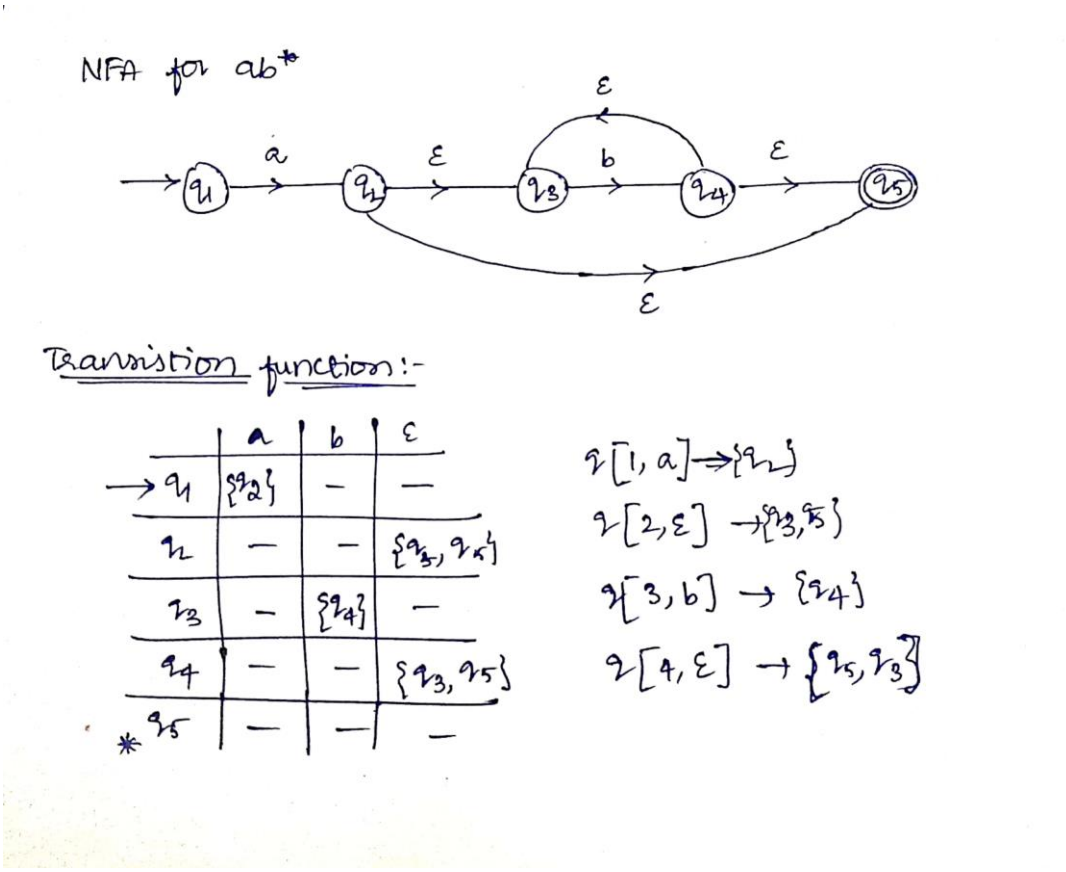
ALGORITHM:

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:



	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	–	–	–

CODE:

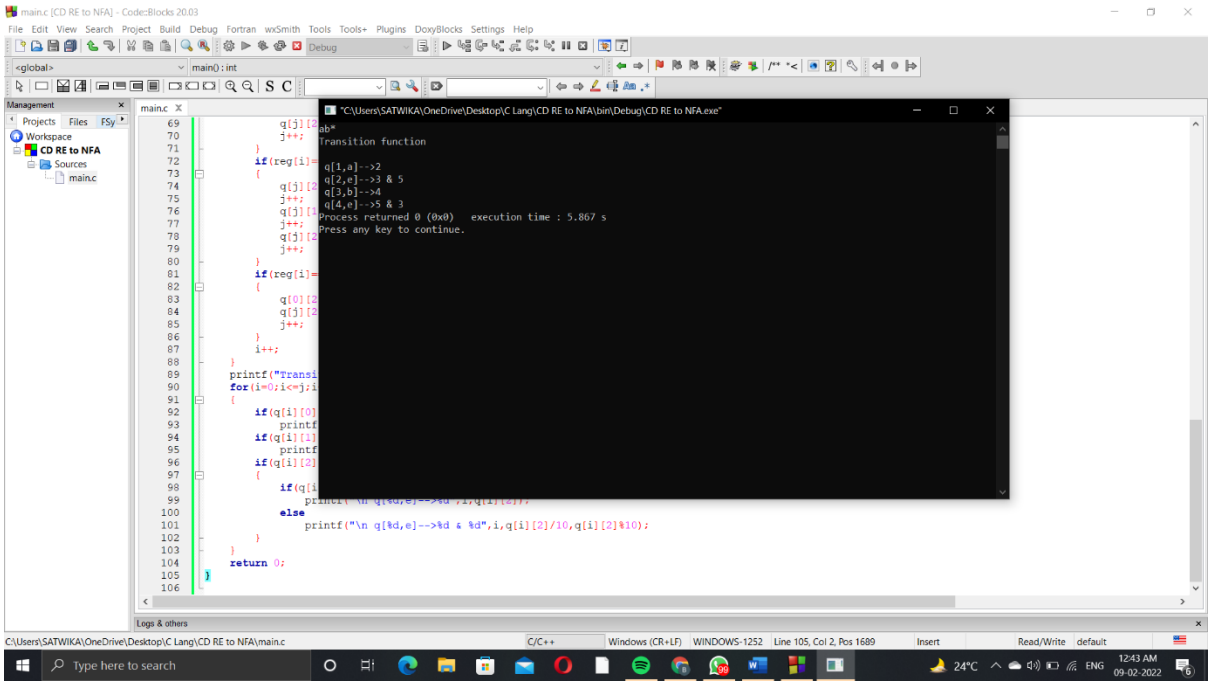
```
#include<stdio.h>
#include<string.h>
int main()
{
	char reg[20];
	int q[20][3],i,j,len,a,b;
	for(a=0;a<20;a++)
	{
		for(b=0;b<3;b++)
		{
			q[a][b]=0;
		}
	}
	scanf("%s",reg);
	len=strlen(reg);
	i=0;
j=1;
	while(i<len)
	{
		if(reg[i]=='a'&&reg[i+1]!='|'&&reg[i+1]!='*')
		{
			q[j][0]=j+1;
			j++;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



RESULT:

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