# 18CSC304J/ Compiler Design

**Submitted By:- ANANNYA P. NEOG (RA1911003010367)**

## Exp-2: Conversion from Regular Expression to NFA

**Aim:-** To convert a Regular Function into NFA

## 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;
		}
	}
	printf("Enter Regular Expression:\n");
	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;
    }
```

```
        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:-

## (Output 1)


## (Output 2)