Rajshahi University of Engineering & Technology

CSE 2206: Sessional Based on CSE 2205

Lab Report 02

Date: November 23, 2018

Submitted to

Sadia Zaman Mishu

Instructor, CSE 2205 & CSE 2206

Assistant Professor, Dept. of CSE

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of CSE

Sessional 2 – Cycle 6 – Problem A

Write a program to construct DFA from NFA.

Theory: We've taken an NFA that accepts all strings starts with '0' as an input NFA. For this transformation, first we've generated all the distinct states by subset construction and then the DFA was constructed from the table. Figure of the NFA and DFA is stated below –

NFA = {{A, B}, {0, 1}, d, A, B}

DFA = {{a, b, c}, {0, 1}, d, a, b}

Code:

```
/*-----------------------------
    I N T R O D U C T I O N
------------------------------
Author:      Fuad Al Abir
Date:        November 26, 2018
Name:        nfa-dfa.cpp
Objective:   This program converts an NFA to DFA and determines if the
             accepting results are the same.
*/

#include<stdio.h>
#include<string.h>
#include<math.h>

int ninputs;
int dfa[100][2][100] = {0};
int state[10000] = {0};
char ch[10], str[1000];
int go[10000][2] = {0};
int arr[10000] = {0};

int main()
{
      int st, fin, in;
      int f[10];
      int i,j=3,s=0,final=0,flag=0,curr1,curr2,k,l;
      int c;
      int p,q,r,rel;

      printf("Enter the number of states: "); scanf("%d", &st);
      printf("States are numbered from 0 to %d\n", st-1);
      for(i=0;i<st;i++) state[(int)(pow(2,i))] = 1;

      printf("Enter number of final states: "); scanf("%d", &fin);
      printf("Enter final state(s): ");
      for(i=0;i<fin;i++) scanf("%d",&f[i]);
```

```
	printf("Enter the number of rules according to NFA: "); 
scanf("%d",&rel);
	printf("Define transition rule as \"Initial state | Input symbol | Final 
state\"\n");
	for(i=0; i<rel; i++)
	{
		scanf("%d%d%d",&p,&q,&r);
		if (q==0) dfa[p][0][r] = 1;
		else dfa[p][1][r] = 1;
	}

	printf("Enter Initial state: "); scanf("%d",&in);
	in = pow(2,in);
	i=0;
	printf("Solving according to DFA:\n");
	int x=0;
	for(i=0;i<st;i++)
	{
		for(j=0;j<2;j++)
		{
			int stf=0;
			for(k=0;k<st;k++)
				if(dfa[i][j][k]==1)
					stf = stf + pow(2,k);
			go[(int)(pow(2,i))][j] = stf;
			printf("%d-%d-->%d\n",(int)(pow(2,i)),j,stf);
			if(state[stf]==0)
				arr[x++] = stf;
			state[stf] = 1;
		}

	}

	for(i=0;i<x;i++)
	{
		for(j=0;j<2;j++)
		{
			int mew=0;
			for(k=0;k<st;k++)
			{
				if(arr[i] & (1<<k))
				{
					int h = pow(2,k);
					if(mew==0)
						mew = go[h][j];
					mew = mew | (go[h][j]);
				}
			}
			if(state[mew]==0)
			{
				arr[x++] = mew;
				state[mew] = 1;
			}
		}
	}

	printf("The total number of distinct states are: \n");
	printf("STATE\t0\t1\n");
	for(i=0;i<10000;i++)
	{
		if(state[i]==1)
		{
			int y=0;
			if(i==0)
```

```
                        printf("q0 ");
                else
                        for(j=0;j<st;j++)
                        {
                                int x = 1<<j;
                                if(x&i)
                                {
                                        printf("\nq%d ",j);
                                        y = y + pow(2,j);
                                }
                        }
                printf("\t%d\t%d",go[y][0],go[y][1]);
        }
    }

    while(1)
    {
    printf("\nEnter string: "); scanf("%s",str);
    l = strlen(str);
    curr1 = in;
    flag = 0;
    printf("\nString takes the following path: "); printf("%d-",curr1);
    for(i=0;i<l;i++)
    {
        curr1 = go[curr1][str[i]-'0'];
        printf("->%d",curr1);
    }
    printf("\nFinal state - %d\n",curr1);
    for(i=0;i<fin;i++)
    {
        if(curr1 & (1<<f[i]))
        {
                flag = 1;
                break;
        }
    }

    if(flag)
        printf("String Accepted.\n");
    else
        printf("String Rejected.\n");
    }
    return 0;
}
```

Input/Output:

```
Enter the number of states: 2
States are numbered from 0 to 1
Enter number of final states: 1
Enter final state(s): 1
Enter the number of rules according to NFA: 3
Define transition rule as "Initial state | Input symbol | Final state"
0 0 1
1 0 1
1 1 1
Enter Initial state: 0
Solving according to DFA:
1-0-->2
1-1-->0
2-0-->2
2-1-->2
```

```
The total number of distinct states are:
STATE   0       1
q0      0       0
q0      2       0
q1      2       2

Enter string: 001

String takes the following path: 1-->2->2->2
Final state - 2
String Accepted.

Enter string: 11

String takes the following path: 1-->0->0
Final state - 0
String Rejected.

Enter string: 1100

String takes the following path: 1-->0->0->0->0
Final state - 0
String Rejected.

Enter string: 0001

String takes the following path: 1-->0->0->0->0->0
Final state - 0
String Accepted.

Enter string: 0

String takes the following path: 1-->2
Final state - 2
String Accepted.

Enter string: 0000

String takes the following path: 1-->2->2->2->2
Final state - 2
String Accepted.

Enter string:
String takes the following path: ^C
```

Discussion: Though, input/output of the program is shown with respect to above mentioned NFA to DFA construction, it works preety fine for the other NFA's. Furthermore, the operations with empty sets and epsilon (empty string) is not taken under consideration. Despite of these limitations, the program works fine for the global circumstances of NFA to DFA construction.