Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Lab Report 04

CSE 2206: Sessional Based on CSE 2205

Submitted to: Sadia Zaman Mishu

Assistant Professor, Dept. of CSE

Date: February 05, 2019

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of CSE

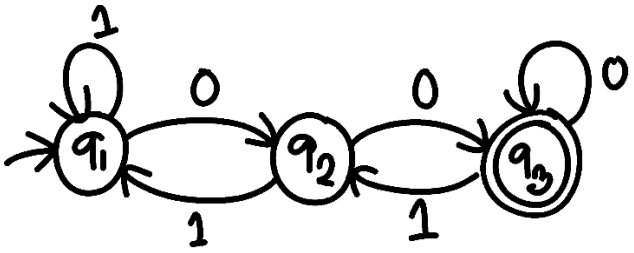
Sessional 4 – Cycle 10 – Problem A

Here is a transition table for a DFA:

	0	1
-> q1	q2	q1
q2	q3	q1
* q3	q3	q2

Give all the regular expressions $R_{ij}^{(0)}$, $R_{ij}^{(1)}$ and $R_{ij}^{(2)}$.

Theory: From the transition table of the DFA, we draw the DFA and $R_{ij}^{(0)}$ was found analysing it.



$R_{ij}^{(1)}$ and $R_{ij}^{(2)}$ was generated by this recurrence relation –

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

Code:

```
/*-----------------------------
    I N T R O D U C T I O N
-----------------------------
Author:       Fuad Al Abir
Date:         December 16, 2018
Name:         dfaToRE.cpp
Objective:    (a) This program prints all the regular expression R_ij(0), R_ij(1)
                  and R_ij(2) from a transition table of a DFA.
*/

#include <iostream>
#include <sstream>
#include <string>

using namespace std;

#define PHI "\u03C6"
#define EPSILON "\u03B5"

int states;
int symbols;
int table[100][100];

string findRE(int n, int state1, int state2, string re = "")
{
    if(n == 0)
    {
        ostringstream str1;

        if(state2 == table[state1][1])
```

```
        {
            if(state1 == state2)
                re = re + EPSILON + " + ";
            str1 << table[0][1];
            re += str1.str();
        }
        if(state2 == table[state1][2])
        {
            if(state1 == state2)
                if(re == "")
                    re = re + EPSILON + " + ";
                else
                    re += " + ";
            str1 << table[0][2];
            re += str1.str();
        }
        if(re == "")
        {
            if(state1 == state2)
                re += EPSILON;
            else
                re += PHI;
        }
        return re;
    }
    else
    {
        return findRE( n - 1, state1, state2 ) + " + " + findRE( n - 1, state1, n )
+ "(" + findRE ( n - 1, n, n ) + ")*" + findRE( n - 1, n, state2);
    }
}

void RE(int n, int states)
{
    cout << "TABLE: R" << n << endl;
    for(int i = 1; i <= states; i++)
    {
        for(int j = 1; j <= states; j++)
        {
            cout << "R_" << i << j << "(" << n << "): " << findRE( n, i, j ) <<
endl;
        }
    }
    cout << endl;
}

int main()
{

    cout << "Number of states: ";
    cin >> states;
    cout << "Number of symbols: ";
    cin >> symbols;

    for(int i = 0; i <= states; i++)
    {
        for(int j = 0; j <= symbols; j++)
        {
            cin >> table[i][j];
        }
    }

    RE(0, states);
    RE(1, states);
    RE(2, states);

    return 0;
}
```

Input/Output:

```
TABLE: R0
R_11(0): ε + 1
R_12(0): 0
R_13(0): φ
R_21(0): 1
R_22(0): ε
R_23(0): 0
R_31(0): φ
R_32(0): 1
R_33(0): ε + 0

TABLE: R1
R_11(1): ε + 1 + ε + 1(ε + 1)*ε + 1
R_12(1): 0 + ε + 1(ε + 1)*0
R_13(1): φ + ε + 1(ε + 1)*φ
R_21(1): 1 + 1(ε + 1)*ε + 1
R_22(1): ε + 1(ε + 1)*0
R_23(1): 0 + 1(ε + 1)*φ
R_31(1): φ + φ(ε + 1)*ε + 1
R_32(1): 1 + φ(ε + 1)*0
R_33(1): ε + 0 + φ(ε + 1)*φ

TABLE: R2
R_11(2): ε + 1 + ε + 1(ε + 1)*ε + 1 + 0 + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*1 + 1(ε + 1)*ε + 1
R_12(2): 0 + ε + 1(ε + 1)*0 + 0 + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*ε + 1(ε + 1)*0
R_13(2): φ + ε + 1(ε + 1)*φ + 0 + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*0 + 1(ε + 1)*φ
R_21(2): 1 + 1(ε + 1)*ε + 1 + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*1 + 1(ε + 1)*ε + 1
R_22(2): ε + 1(ε + 1)*0 + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*ε + 1(ε + 1)*0
R_23(2): 0 + 1(ε + 1)*φ + ε + 1(ε + 1)*0(ε + 1(ε + 1)*0)*0 + 1(ε + 1)*φ
R_31(2): φ + φ(ε + 1)*ε + 1 + 1 + φ(ε + 1)*0(ε + 1(ε + 1)*0)*1 + 1(ε + 1)*ε + 1
R_32(2): 1 + φ(ε + 1)*0 + 1 + φ(ε + 1)*0(ε + 1(ε + 1)*0)*ε + 1(ε + 1)*0
R_33(2): ε + 0 + φ(ε + 1)*φ + 1 + φ(ε + 1)*0(ε + 1(ε + 1)*0)*0 + 1(ε + 1)*φ
```

Discussion: All the regular expressions was generated analysing the DFA and the recurrence relation among them - but as the problem asked to simplyfy as possible - this portion of the problem was skipped to solve a simplified problem.