

Lab Report

Subject: CSE-3214 Compiler Design Lab

Experiment Name: Design and Implementation of DFA and NFA that end with “01”

Tool Used: C++ Programming Language

Submitted By:

Eyasir Ahamed

Roll No:15

Registration No: 202004017

Submitted To:

Kohinur Parvin

Lecturer,

Netrokona University, Netrokona

DFA That End with "01"

Program Code:

```
a.cpp x b.cpp x c.cpp x d.cpp x e.cpp x test.cpp
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 string transition(string input, string curState) {
5     for (auto c : input) {
6         cout<<"Present State: "<<curState<<" | ";
7         if (curState == "q0") {
8             if (c == '0')curState = "q1";
9             else curState = "q0";//1
10        } else if (curState == "q1") {
11            if (c == '0')curState = "q1";
12            else curState = "q2";//1
13        } else { //q2
14            if (c == '0')curState = "q1";
15            else curState = "q0";//1
16        }
17        cout << "Present Symbol: " << c << " | Next State: " << curState << endl;
18    }
19    cout << endl;
20    return curState;
21 }
22
23 bool endwith01(string input) {
24     string curState = "q0";
25     curState = transition(input, curState);
26     return curState == "q2";
27 }
28
29 int main() {
30     cout << "DFA that end with 01" << endl;
31     cout << "=====" << endl;
32     cout << " | --< 0 -<- || --<-- 0 --<--|" << endl;
33     cout << "->(q0) --> 0 -->-- ( q1 )-->-- 1 -->--((q2))" << endl;
34     cout << " |-----<-- 1 -----<-----|" << endl;
35     cout<<endl;
36
37     string input;
38     while (cin >> input) {
39         cout << "Input string: " << input << endl;
40         if (endwith01(input)) {
41             cout << "STATUS -> " << "ACCEPTED" << endl;
42         } else {
43             cout << "STATUS -> " << "REJECTED" << endl;
44         }
45         cout << endl;
46     }
47     return 0;
48 }
49
```

Input Example:

```
input.txt
1 01
2 00001
3 001110001
4 100
```

Output Example:

```
output.txt
1 DFA that end with 01
2 =====
3 | --< 0 -<- || --<-- 0 --<--|
4 ->(q0) --> 0 -->-- ( q1 )-->-- 1 -->--((q2))
5 |----- --<-- 1 -----<-----|
6
7 Input string: 01
8 Present State: q0 | Present Symbol: 0 | Next State: q1
9 Present State: q1 | Present Symbol: 1 | Next State: q2
10
11 STATUS -> ACCEPTED
12
13 Input string: 00001
14 Present State: q0 | Present Symbol: 0 | Next State: q1
15 Present State: q1 | Present Symbol: 0 | Next State: q1
16 Present State: q1 | Present Symbol: 0 | Next State: q1
17 Present State: q1 | Present Symbol: 0 | Next State: q1
18 Present State: q1 | Present Symbol: 1 | Next State: q2
19
20 STATUS -> ACCEPTED
21
22 Input string: 001110001
23 Present State: q0 | Present Symbol: 0 | Next State: q1
24 Present State: q1 | Present Symbol: 0 | Next State: q1
25 Present State: q1 | Present Symbol: 1 | Next State: q2
26 Present State: q2 | Present Symbol: 1 | Next State: q0
27 Present State: q0 | Present Symbol: 1 | Next State: q0
28 Present State: q0 | Present Symbol: 0 | Next State: q1
29 Present State: q1 | Present Symbol: 0 | Next State: q1
30 Present State: q1 | Present Symbol: 0 | Next State: q1
31 Present State: q1 | Present Symbol: 1 | Next State: q2
32
33 STATUS -> ACCEPTED
34
35 Input string: 100
36 Present State: q0 | Present Symbol: 1 | Next State: q0
37 Present State: q0 | Present Symbol: 0 | Next State: q1
38 Present State: q1 | Present Symbol: 0 | Next State: q1
39
40 STATUS -> REJECTED
41
```

NFA that end with "01:

Program Code

```
a.cpp  b.cpp  c.cpp  d.cpp  e.cpp
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  map<string, map<char, vector<string>>> nfa;
5
6  void buildNFA() {
7      nfa["q0"]['0'] = {"q0", "q1"};
8      nfa["q0"]['1'] = {"q0"};
9      nfa["q1"]['1'] = {"q2"};
10 }
11
12 set<string> transition(set<string> curStates, char symbol) {
13     set<string> nextStates;
14     for (auto &state : curStates) {
15         if (nfa[state].count(symbol)) {
16             for (auto &next : nfa[state][symbol]) {
17                 nextStates.insert(next);
18             }
19         }
20     }
21     return nextStates;
22 }
23
24 bool endwith01(string input) {
25     set<string> curStates = {"q0"};
26     for (auto c : input) {
27         cout << "Present Symbol: " << c << " | Next States: ";
28         curStates = transition(curStates, c);
29         for (auto s : curStates) cout << s << " ";
30         cout << endl;
31     }
32     return curStates.count("q2");
33 }
34
35 int main() {
36     buildNFA();
37
38     cout << "NFA that end with 01" << endl;
39     cout << "======" << endl;
40     cout << "->(q0) --> 0 --> (q1)--> 1 --> (q2)" << endl;
41     cout << endl;
42
43     string input;
44     while (cin >> input) {
45         cout << "Input string: " << input << endl;
46         if (endwith01(input)) {
47             cout << "STATUS -> ACCEPTED" << endl;
48         } else {
49             cout << "STATUS -> REJECTED" << endl;
50         }
51         cout << endl;
52     }
53     return 0;
54 }
```

Input and Output Example

```
input.txt
1 01
2 00001
3 001110001
4 100

output.txt
1 NFA that end with 01
2 =====
3 ->(q0) --> 0 -->-- (q1)-->-- 1 -->--((q2))
4
5 Input string: 01
6 Present Symbol: 0 | Next States: q0 q1
7 Present Symbol: 1 | Next States: q0 q2
8 STATUS -> ACCEPTED
9
10 Input string: 00001
11 Present Symbol: 0 | Next States: q0 q1
12 Present Symbol: 0 | Next States: q0 q1
13 Present Symbol: 0 | Next States: q0 q1
14 Present Symbol: 0 | Next States: q0 q1
15 Present Symbol: 1 | Next States: q0 q2
16 STATUS -> ACCEPTED
17
18 Input string: 001110001
19 Present Symbol: 0 | Next States: q0 q1
20 Present Symbol: 0 | Next States: q0 q1
21 Present Symbol: 1 | Next States: q0 q2
22 Present Symbol: 1 | Next States: q0
23 Present Symbol: 1 | Next States: q0
24 Present Symbol: 0 | Next States: q0 q1
25 Present Symbol: 0 | Next States: q0 q1
26 Present Symbol: 0 | Next States: q0 q1
27 Present Symbol: 1 | Next States: q0 q2
28 STATUS -> ACCEPTED
29
30 Input string: 100
31 Present Symbol: 1 | Next States: q0
32 Present Symbol: 0 | Next States: q0 q1
33 Present Symbol: 0 | Next States: q0 q1
34 STATUS -> REJECTED
35
```