College of Engineering Trivandrum

Compiler Design Lab

Abhishek Manoharan
S7 CSE Roll No:2

TVE17CS002

Department of Computer Science

September 9, 2020

Exp 4

1 ϵ- closure

1.1 Aim

Write program to find ϵ - closure of all states of any given NFA with ϵ transition

1.2 Theory

NFA

- NFA stands for non-deterministic finite automata. It is easy to construct an NFA than DFA for a given regular language.
- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- Every NFA is not DFA, but each NFA can be translated into DFA.
- NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ϵ transition.
  - Q: finite set of states
  - Σ: finite set of the input symbol
  - q0: initial state
  - F: final state
  - δ: Transition function
  - δ : Q x $\Sigma \rightarrow 2^{Q}$

Epsilon Closure:
Epsilon closure for a given state X is a set of states which can be reached from the states X with only (null) or ϵ moves including the state X itself. In other words,ϵ -closure for a state can be obtained by union operation of the ϵ-closure of the states which can be reached from X with a single ϵ move in recursive manner.

1.3 Algorithm

Algorithm 1: Algorithm to find Epsilon Closure

```
function EPSILONCLOSURE ( enfa , int k )
    Initialize a list t containing only state k
    Initialize an iterator to the first element of the list t
    while iterator has not crossed the last element of the list t
        Append all states in the i pair in the transition table of
        enfa which is not previously present in list t to t
        Set the iterator to the next element of the list t
        Return list t as the epsilon - closure for state k in
        epsilon - N F A enfa
end function
```

1.4 Code

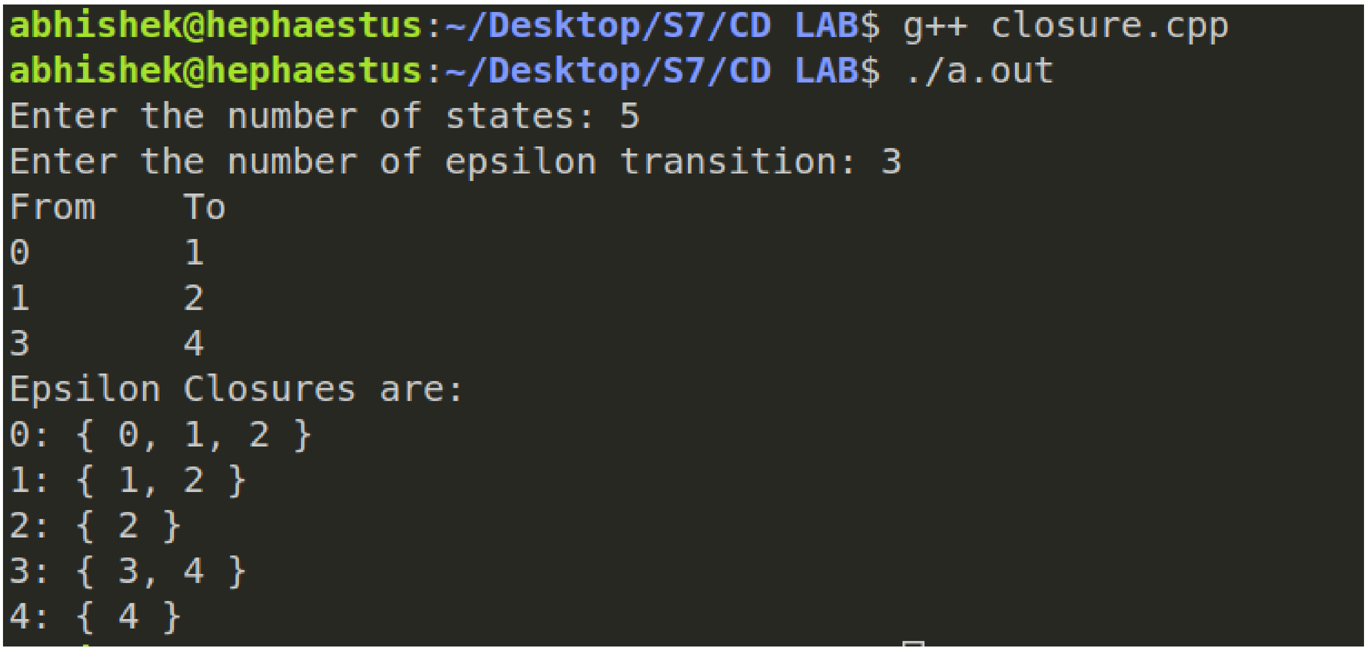
```cpp
// CPP program to find the epsilon closure of all states
#include <bits/stdc++.h>
using namespace std;

int main()
{
    cout << "Enter the number of states: ";
    int n, m;
    cin >> n;
    int start, last, temp, temp1;
    int epsilon;
    cout << "Enter the number of epsilon transition: ";
    cin >> epsilon;
    vector<pair<int, int>> eps;
    cout << "From\tTo" << endl;
    for (int i = 0; i < epsilon; ++i)
    {
        cin >> temp >> temp1;
        if (temp >= n || temp < 0 || temp1 >= n || temp1 < 0)
        {
            cout << "incorrect input: program forced to terminate" << endl;
            return 0;
        }
        eps.push_back({temp, temp1});
    }

    queue<int> check;
    cout << "Epsilon Closures are: \n";
    for (int i = 0; i < n; ++i)
    {
        vector<int> visited(n, 0);
        if (visited[i] != 1)
        {
            cout << i << ": ";
            visited[i] = 1;
            cout << "{ " << i;
            for (auto x : eps)
            {
                if (x.first == i)
                {
                    check.push(x.second);
                }
            }
            while (!check.empty())
            {
                int c = check.front();
                check.pop();
                visited[c] = 1;
                cout << ", " << c;
                for (auto x : eps)
                {
                    if (x.first == c && visited[x.second] != 1)
                    {
                        check.push(x.second);
                    }
                }
            }
            cout << " }" << endl;
        }
    }

    return 0;
}
```

1.5 Output



```
abhishek@hephaestus:~/Desktop/S7/CD LAB$ g++ closure.cpp
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the number of states: 5
Enter the number of epsilon transition: 3
From    To
0       1
1       2
3       4
Epsilon Closures are:
0: { 0, 1, 2 }
1: { 1, 2 }
2: { 2 }
3: { 3, 4 }
4: { 4 }
```

```
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the number of states: 5
Enter the number of epsilon transition: 3
From    To
0       1
1       2
3       4
Epsilon Closures are:
0: { 0, 1, 2 }
1: { 1, 2 }
2: { 2 }
3: { 3, 4 }
4: { 4 }
```

1.6 Result

Implemented the program to find epsilon closure of states of a NFA in CPP. It was compiled using g++ version 9.3.0, and executed in Ubuntu 20.04 and the above output was obtained.