2 - Epsilon Closure

Sunday, January 28, 2024 7:12 PM

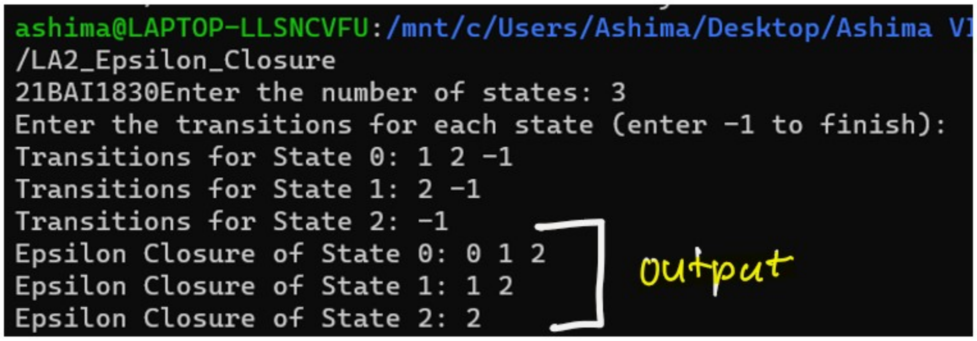
Ashima Fatima Seik Mugibur Raghman, 21BAI1830

AIM: Write a C/C++/Java program to compute epsilon closure of all the states in a given ε-NFA.

ALGORITHM:

1. initialize an Eplison NFA object with the given number of states, transitions and epsilon transitions.
2. Create a function that computs the epsilon closure for a given state using depth first search (DFS)
   a. create an empty set called 'visited'. This holds the closure set.
   b. create an empty stack called 'dfs Stack'
   c. insert the initial state into both visited and dfs Stack.
   d. while dfs Stack is not empty, pop the current state from the dfs Stack and iterate through all ~~states~~ reachable states through epsilon transition. If that next state is not in the visited set, add it to the set.
   e. return the visited (i.e. closure) set.
3. create a function to calculate epsilon closures for all states and store them in a vector.
4. create an Epsion NFA object, initialize it and call the function to get the vector of all epsilon closures.
5. Print the epsilon closures for all states.

STIMULATION:



```
ashima@LAPTOP-LLSNCVFU:/mnt/c/Users/Ashima/Desktop/Ashima VI
/LA2_Epsilon_Closure
21BAI1830Enter the number of states: 3
Enter the transitions for each state (enter -1 to finish):
Transitions for State 0: 1 2 -1
Transitions for State 1: 2 -1
Transitions for State 2: -1
Epsilon Closure of State 0: 0 1 2
Epsilon Closure of State 1: 1 2
Epsilon Closure of State 2: 2
```

INPUT:

3
1 2 -1
2 -1
-1

CODE:

```
#include <iostream>
#include <vector> //dynamic arrays
#include <set> //only hold unique elements
#include <stack> //DFS
using namespace std;
class EpsilonNFATransition {
public:
    int state;
    char input;
    int NXTstate;
}; //dont forget semicolon
class EpsilonNFA {
public:
    int numStates;
    //vector of e transitions
    vector<set<int>> eTransitions; //vector of states with their set of e
transition states
    //vector of all transitions
    vector<vector<EpsilonNFATransition>> transitions; //vector for a state, of
vectors holding NFA transitions for all inputs

    //function to compute e closure of a state
    set<int> eClosure(int state) {
        set<int> visited;
        stack<int> dfsStack;

        visited.insert(state);
        dfsStack.push(state);

        while (!dfsStack.empty()){
            int currentState = dfsStack.top();
            dfsStack.pop();

            for (int nextState : eTransitions[currentState]) {
                if (visited.find(nextState) == visited.end()) { //if nextState is
not in visited
                    visited.insert(nextState);
                    dfsStack.push(nextState);
                }
            }
        }
        return visited;
    }

    //function to compute e closure of all states
    vector<set<int>> computeEpsilonClosures() {
        vector<set<int>> epsilonClosures;
        for (int i = 0; i < numStates; ++i) {
            epsilonClosures.push_back(eClosure(i));
        }
        return epsilonClosures;
    }
};
int main() {
     EpsilonNFA eNFA;
    cout << "21BAI1830\n";
    cout << "Enter the number of states: ";
    cin >> eNFA.numStates;
    cout << "Enter the transitions for each state (enter -1 to finish):\n";
    for (int i = 0; i < eNFA.numStates; ++i) {
        set<int> transitionsSet;
        int transition;
        cout << "Transitions for State " << i << ": ";
        while (cin >> transition && transition != -1) {
            transitionsSet.insert(transition);
        }
        eNFA.eTransitions.push_back(transitionsSet);
    }
    vector<set<int>> epsilonClosures = eNFA.computeEpsilonClosures();
    for (int i = 0; i < eNFA.numStates; ++i) {
        cout << "Epsilon Closure of State " << i << ": ";
        for (int state : epsilonClosures[i]) {
            cout << state << " ";
        }
        cout << endl;
    }
    return 0;
}
```