Compiler Design Lab Report

Santhisenan A
University Roll Number *TVE15CS050*
Department of Computer Science
College of Engineering Trivandrum

Aug - Nov 2018

Chapter 1

Automata Concepts

1.1 Finding ϵ - closures

1.1.1 Problem

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

1.1.2 Algorithm

The algorithm for finding ϵ Closure is as follows:

- Iterate through all the states in the NFA
- For each state, find the states reachable via epsilon transitions and add them to the epsilon closure.
- For the all the states added to epsilon closure, recursively execute the function to find the epsilon closure.

1.1.3 Code

```
void find_e_closure(nfa n, int state, unordered_set<int> &closure) {
    unordered_set<int> toStates = n.table[state][0];
    closure.insert(state);
    if (toStates.find(-1) != toStates.end())
    {
        return;
    }
    else
```

```
    {
        unordered_set<int>::iterator itr;
        for (itr = toStates.begin(); itr != toStates.end(); itr++)
        {
            if (find(closure.begin(), closure.end(), *itr) == closure.end())
            {
                closure.insert(*itr);
            }
            find_e_closure(n, *itr, closure);
        }
    }
}
```

1.2 Conversion from an NFA with ϵ transitions to an NFA without ϵ transitions

1.2.1 Problem

Write program to convert an NFA with ϵ transitions to an NFA without ϵ transitions.

1.3 Algorithm

- Initialize an empty object of type nf a with variable name t
- Initialize t.num states = a.num states, t.num alphabets = a.num alphabets and t.final states = a.final states
- Iterate through each of the state i in Q
  - Initialize l to the closure of state i of -NFA a Iterate through each of the input symbol j in
    - Initialize an empty list of states f Iterate through each state k in l and Add all states of a.transition table[k][j + 1] to f Remove all the duplicates from f
    - Compute the -closure c of f
    - Set t.transition table[i][j] = c
- Return t as the NFA without -transitions corresponding to the -NFA a

1.3.1 Code

```
        // To convert e-NFA to NFA
void enfa_to_nfa(nfa n, nfa &m)
{
    m.states = n.states;
    m.alphabets = n.alphabets;

    int s = n.states, a = n.alphabets;

    for (int i = 0; i < s; i++)
    {
```

```
    vector<unordered_set<int> > row;

    // Insert -1 to alphabet epsilon
    unordered_set<int> nullState;
    nullState.insert(-1);
    row.push_back(nullState);
    nullState.clear();

    // Find the e Closure of the current state and store in eClosure
    unordered_set<int> eClosure;
    unordered_set<int>::iterator itr;
    find_e_closure(n, i, eClosure);

    for (int j = 1; j < a; j++)
    {
        unordered_set<int> temp, tempClosure, toStates;
        unordered_set<int>::iterator it;
        for (itr = eClosure.begin(); itr != eClosure.end(); itr++)
        {
            temp = n.table[*itr][j];
            if (!temp.empty())
            {
                for (it = temp.begin(); it != temp.end(); it++)
                {
                    // cout << "Hello" << *it << endl;
                    if(*it != -1) {
                        find_e_closure(n, *it, tempClosure);
                        append_sets(toStates, tempClosure);
                    }
                }
            }
        }
        if(toStates.empty()) {
            toStates.insert(-1);
        }
        row.push_back(toStates);
        toStates.clear();
    }
    m.table.push_back(row);
    row.clear();
}
```

```
}
```