COMPILER DESIGN

LAB 5

Aman Kalla

RA1911003010640

CODE

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

void print(vector<vector<vector<int> > > table){

cout<<" STATE/INPUT |";

char a='a';

for(int i=0;i<table[0].size()-1;i++){

cout<<" "<<a++<<" |";

}

cout<<" ^ "<<endl<<endl;

for(int i=0;i<table.size();i++){

cout<<" "<<i<<" ";

for(int j=0;j<table[i].size();j++){

cout<<" | ";

for(int k=0;k<table[i][j].size();k++){

cout<<table[i][j][k]<<" ";

}

}

cout<<endl;

}

}

void printdfa(vector<vector<int> > states, vector<vector<vector<int> > > dfa){

cout<<" STATE/INPUT ";

char a='a';

for(int i=0;i<dfa[0].size();i++){

cout<<"| "<<a++<<" ";

}

cout<<endl;

for(int i=0;i<states.size();i++){

cout<<"{ ";

for(int h=0;h<states[i].size();h++)

cout<<states[i][h]<<" ";

if(states[i].empty()){

cout<<"^ ";

}

cout<<"} ";

for(int j=0;j<dfa[i].size();j++){

cout<<" | ";

for(int k=0;k<dfa[i][j].size();k++){

cout<<dfa[i][j][k]<<" ";

}

if(dfa[i][j].empty()){

cout<<"^ ";

}

}

cout<<endl;

}

}

vector<int> closure(int s,vector<vector<vector<int> > > v){

vector<int> t;

queue<int> q;

t.push\_back(s);

int a=v[s][v[s].size()-1].size();

for(int i=0;i<a;i++){

t.push\_back(v[s][v[s].size()-1][i]);

//cout<<"t[i]"<<t[i]<<endl;

q.push(t[i]);

}

while(!q.empty()){

int f=q.front();

q.pop();

if(!v[f][v[f].size()-1].empty()){

int u=v[f][v[f].size()-1].size();

for(int i=0;i<u;i++){

int y=v[f][v[f].size()-1][i];

if(find(t.begin(),t.end(),y)==t.end()){

//cout<<"y"<<y<<endl;

t.push\_back(y);

q.push(y);

}

}

}

}

return t;

}

int main(){

int n,alpha;

cout<<" NFA to DFA "<<endl<<endl;

cout<<"Enter total number of states in NFA: ";

cin>>n;

cout<<"Enter number of elements: ";

cin>>alpha;

vector<vector<vector<int> > > table;

for(int i=0;i<n;i++){

cout<<"For state "<<i<<endl;

vector< vector< int > > v;

char a='a';

int y,yn;

for(int j=0;j<alpha;j++){

vector<int> t;

cout<<"Enter no. of output states for input "<<a++<<" : ";

cin>>yn;

cout<<"Enter output states :"<<endl;

for(int k=0;k<yn;k++){

cin>>y;

t.push\_back(y);

}

v.push\_back(t);

}

vector<int> t;

cout<<"Enter no. of output states for input ^ : ";

cin>>yn;

cout<<"Enter output states :"<<endl;

for(int k=0;k<yn;k++){

cin>>y;

t.push\_back(y);

}

v.push\_back(t);

table.push\_back(v);

}

cout<<"\*\* TRANSITION TABLE OF NFA \*\*"<<endl;

print(table);

cout<<endl<<"\*\* TRANSITION TABLE OF DFA \*\*"<<endl;

vector<vector<vector<int> > > dfa;

vector<vector<int> > states;

states.push\_back(closure(0,table));

queue<vector<int> > q;

q.push(states[0]);

while(!q.empty()){

vector<int> f=q.front();

q.pop();

vector<vector<int> > v;

for(int i=0;i<alpha;i++){

vector<int> t;

set<int> s;

for(int j=0;j<f.size();j++){

for(int k=0;k<table[f[j]][i].size();k++){

vector<int> cl= closure(table[f[j]][i][k],table);

for(int h=0;h<cl.size();h++){

if(s.find(cl[h])==s.end())

s.insert(cl[h]);

}

}

}

for(set<int >::iterator u=s.begin(); u!=s.end();u++)

t.push\_back(\*u);

v.push\_back(t);

if(find(states.begin(),states.end(),t)==states.end())

{

states.push\_back(t);

q.push(t);

}

}

dfa.push\_back(v);

}

printdfa(states,dfa);

}

Output



Result

Hence, NFA to DFA conversion is successfully done

