Name: Atandrit Chatterjee

Reg. No.: RA2011031010042

**Experiment-3** **CONVERSION OF NFA TO DFA**

Aim:

To write a program for converting NFA to DFA.

Procedure:

1. Start

2. Get the input from the user

3. Set the only state in SDFA to “unmarked”.

4. While SDFA contains an unmarked state do:

a. Let T be that unmarked state

b. for each a in % do S = e-Closure(MoveNFA(T,a))

c. if S is not in SDFA already then, add S to SDFA (as an “unmarked”)

d. Set MoveDFA(T,a) to S.

5. For each S in SDFA if any s & S is a final state in the NFA then, mark S a final state in the DFA.

6. Print the result.

Code:

#include <vector>

#include <iostream>

using namespace std;

int main()

{

vector<vector<int>> nfa( 5 , vector<int> (3));

vector<vector<int>> dfa( 10 , vector<int> (3));

for(int i=1;i<5;i++){

for(int j=1;j<=2;j++){

int h;

if (j == 1){

cout << "nfa [" << i << ", a]: ";

}

else{

cout << "nfa [" << i << ", b]: ";

}

cin>>h;

nfa[i][j]=h;

}

}

int dstate[10];

int i=1,n,j,k,flag=0,m,q,r;

dstate[i++]=1;

n=i;

dfa[1][1]=nfa[1][1];

dfa[1][2]=nfa[1][2];

cout<<"\n"<<"dfa["<<dstate[1]<<", a]: {"<<dfa[1][1]/10<<", "<<dfa[1][1]%10<<"}";

cout<<"\n"<<"dfa["<<dstate[1]<<", b]: "<<dfa[1][2];

for(j=1;j<n;j++)

{

if(dfa[1][1]!=dstate[j])

flag++;

}

if(flag==n-1)

{

dstate[i++]=dfa[1][1];

n++;

}

flag=0;

for(j=1;j<n;j++)

{

if(dfa[1][2]!=dstate[j])

flag++;

}

if(flag==n-1)

{

dstate[i++]=dfa[1][2];

n++;

}

k=2;

while(dstate[k]!=0)

{

m=dstate[k];

if(m>10)

{

q=m/10;

r=m%10;

}

if(nfa[r][1]!=0)

dfa[k][1]=nfa[q][1]\*10+nfa[r][1];

else

dfa[k][1]=nfa[q][1];

if(nfa[r][2]!=0)

dfa[k][2]=nfa[q][2]\*10+nfa[r][2];

else

dfa[k][2]=nfa[q][2];

if (dstate[k] > 10){

if (dfa[k][1] > 10){

cout<<"\n"<<"dfa[{"<<dstate[k]/10 << ", " << dstate[k]%10 <<"}, a]: {"<<dfa[k][1]/10 << ", " << dfa[k][1]%10 << "}";

}

else{

cout<<"\n"<<"dfa[{"<<dstate[k]/10 << ", " << dstate[k]%10 <<"}, a]: "<<dfa[k][1];

}

}

else{

if (dfa[k][1] > 10){

cout<<"\n"<<"dfa["<<dstate[k] << ", a]: {"<<dfa[k][1]/10 << ", " << dfa[k][1]%10 << "}";

}

else{

cout<<"\n"<<"dfa["<<dstate[k] << ", a]: "<<dfa[k][1];

}

}

if (dstate[k] > 10){

if (dfa[k][2] > 10){

cout<<"\n"<<"dfa[{"<<dstate[k]/10 << ", " << dstate[k]%10 <<"}, b]: {"<<dfa[k][2]/10 << ", " << dfa[k][2]%10 << "}";

}

else{

cout<<"\n"<<"dfa[{"<<dstate[k]/10 << ", " << dstate[k]%10 <<"}, b]: "<<dfa[k][2];

}

}

else{

if (dfa[k][1] > 10){

cout<<"\n"<<"dfa["<<dstate[k] << ", b]: {"<<dfa[k][2]/10 << ", " << dfa[k][2]%10 << "}";

}

else{

cout<<"\n"<<"dfa["<<dstate[k] << ", b]: "<<dfa[k][2];

}

}

flag=0;

for(j=1;j<n;j++)

{

if(dfa[k][1]!=dstate[j])

flag++;

}

if(flag==n-1)

{

dstate[i++]=dfa[k][1];

n++;

}

flag=0;

for(j=1;j<n;j++)

{

if(dfa[k][2]!=dstate[j])

flag++;

}

if(flag==n-1)

{

dstate[i++]=dfa[k][2];

n++;

}

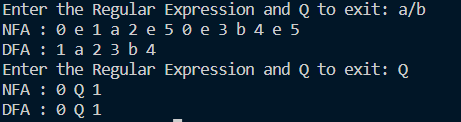
k++;

}

return 0;

}

Output:



Result:

The given NFA was converted to a DFA successfully.