EXPERIMENT - V
ϵ-NFA TO NFA CONVERSION

September 9, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To write a program to convert NFA with ϵ-transition to NFA without ϵ-transitions.

THEORY

Non-Deterministic Finite Automata

A non-deterministic finite automata (NFA) A over an alphabet set Σ is a structure of the form:

$$A = (Q, S, \Delta, F)$$

where,

- Q is a set of finite states.
- S ⊆ Q is the set of start states.
- Δ : Q X Σ → 2^Q is the transition function.
- F ⊆ Q is the set of final states.

NFAs can be extended with ϵ-transitions. An ϵ-transition allows the NFA to make a move without consuming any input.

ALGORITHM

Algorithm 1 Algorithm to Convert ϵ-NFA to NFA

```
Start
Input the ϵ-NFA (enfa).
Initialize an empty object of type NFA (nfa).
nfa.noofstates = enfa.noofstates
nfa.noofalphabets = enfa.noofalphabets
nfa.finalstates = enfa.finalstates
for each state i in enfa do
    Initialize t to ϵ-closure of ϵ-NFA enfa.
    for each input symbol j in Σ do
        Initialize an empty list of states f.
        for each state k in t do
            Add all states of enfa.transitiontable[k][j] to f.
        end for
        Remove all the duplicates from f.
        Compute the ϵ-closure c of f.
        Set nfa.transitiontable[i][j] = c.
    end for
end for
Stop
```

SOURCE CODE

```
def epsilon_closure(l,k):
        t=[]
        t.append(k)
        if l[k][s]=="-":
                return t
        for i in t:
                if l[i][s]=="-":
                        continue
                x=l[i][s].split(",")
                for a in x:
                        if a not in t:
                                t.append(int(a))
        return t

def ConvertToNFA(enfa):
        print()
        print("Generating NFA ...")
        Q=enfa[0]
        S=enfa[1]
        delta=[]
        F=enfa[3]
        tt=enfa[2]
        for i in Q:
                l=epsilon_closure(tt,i)
                trans=[]
                for j in range(len(sigma)):
                        f=[]
                        for k in l:
                                if(tt[k][j]=="-"):
                                        continue
                                elif(len(tt[k][j])==1):
                                        x=tt[k][j]
                                        if x not in f:
```

```
                                        f.append(int(x))
                        else:
                                x=tt[k][j].split(",")
                                for q in x:
                                        if q not in f:
                                                f.append(int(q))
                c=[]
                for k in f:
                        x=epsilon_closure(tt,k)
                        for w in x:
                                if w not in c:
                                        c.append(w)
                trans.append(c)
        delta.append(trans)
print()
print("Start states of NFA : ",end="")
for i in range(len(S)-1):
        print(S[i],end=",")
print(S[len(S)-1])
print()
print("STATE",end="\t")
for i in range(len(sigma)-1):
        print(sigma[i],end="\t")
print(sigma[len(sigma)-1])
for i in range(n):
        print(i,end="\t")
        for j in range(s-1):
                for z in range(len(delta[i][j])-1):
                        print(delta[i][j][z],end=",")
                print(delta[i][j][len(delta[i][j])-1],end="\t")
        for z in range(len(delta[i][s-1])-1):
                print(delta[i][s-1][z],end=",")
        print(delta[i][s-1][len(delta[i][s-1])-1])
print()
for i in S:
```

```
                l=epsilon_closure(tt,i)
                for j in l:
                        if j in F:
                                F.append(i)
                                break
        print("Final states of NFA : ",end="")
        F.sort()
        for i in range(len(F)-1):
                print(F[i],end=",")
        print(F[len(F)-1])

global s
global n
global sigma
n=int(input("Enter number of states in the NFA:"))
Q=[i for i in range(n)]
s=int(input("Enter number of input symbols:"))
print("State",end="\t")
x=('a','b','c','d','e','f','g','h','i','j')
sigma=[]
for i in range(s):
        print(x[i],end="\t")
        sigma.append(x[i])
print("epsilon")
enfa=[]
enfa.append(Q)
delta=[]
for i in range(n):
        print(i,end="\t")
        t=input().split()
        delta.append(t)
m=input("Enter the start states of the NFA:").split(",")
f=input("Enter the final states of the NFA:").split(",")
S=[]
F=[]
```

```
for i in range(len(m)):
        S.append(int(m[i]))
for i in range(len(f)):
        F.append(int(f[i]))
enfa.append(S)
enfa.append(delta)
enfa.append(F)
ConvertToNFA(enfa)
```

SAMPLE OUTPUT

```
user@adithya-d-rajagopal:~/s7/cd$ python3 p5.py
Enter number of states in the NFA:3
Enter number of input symbols:2
State   a       b       epsilon
0       0       1       2
1       1,2     2       -
2       0       1       -
Enter the start states of the NFA:0
Enter the final states of the NFA:2

Generating NFA ...

Start states of NFA : 0

STATE   a       b
0       0,2     1
1       1,2     2
2       0,2     1

Final states of NFA : 0,2
user@adithya-d-rajagopal:~/s7/cd$
```

RESULT

A program to convert NFA with ϵ-transitions to NFA without ϵ-transitions has been implemented using Python and the outputs have been verified.