http://www.cs.odu.edu/~toida/nerzic/390teched/web_course.html

Conversion of NFA to DFA

Conversion of NFA to DFA

Let $M_2 = < Q_2 , \Sigma, q_{2,0} , \delta_2 , A_2 >$ be an NFA that recognizes a language L. Then the DFA $M = < Q, \Sigma, q_0 , \delta , A >$ that satisfies the following conditions recognizes L:

Q = 2^{Q_2} , that is the set of all subsets of Q_2 ,
$q_0 = \{ q_{2,0} \}$,
δ (q, a) = $\cup_{p \in q} \delta(p, a)$ for each state q in **Q** and each symbol a in Σ and
A = { q $\in$ Q | q $\cap A_2 \neq \emptyset$ }

To obtain a DFA $M = < Q, \Sigma, q_0 , \delta , A >$ which accepts the same language as the given NFA $M_2 = < Q_2 , \Sigma, q_{2,0} , \delta_2 , A_2 >$ does, you may proceed as follows:

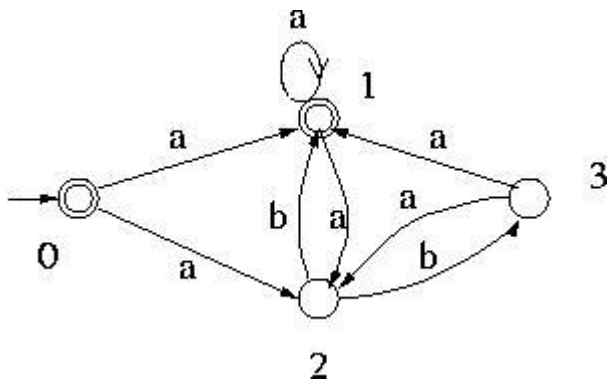
Initially **Q** = $\emptyset$.
First put $\{ q_{2,0} \}$ into **Q**. $\{ q_{2,0} \}$ is the initial state of the DFA M.
Then for each state q in **Q** do the following:
add the set $\cup_{p \in q} \delta(p, a)$, where δ here is that of NFA M_2, as a state to **Q** if it is not already in **Q** for each symbol a in Σ.
For this new state, add δ (q, a) = $\cup_{p \in q} \delta(p, a)$ to δ, where the δ on the right hand side is that of NFA M_2.

When no more new states can be added to **Q**, the process terminates. All the states of **Q** that contain accepting states of M_2 are accepting states of M.

Note: The states that are not reached from the initial state are not included in **Q** obtained by this

procedure. Thus the set of states **Q** thus obtained is not necessarily equal to 2^{Q}_{2} .

Example 1: Let us convert the following NFA to DFA.



Initially Q is empty. Then since the initial state of the DFA is {0} , {0} is added to Q.

Since δ_2(0 , a) = { 1 , 2 } , { 1 , 2 } is added to **Q** and δ ({ 0 } , a) = { 1 , 2 } .

Since δ_2(0 , b) = $\emptyset$, $\emptyset$ is added to **Q** and δ ({ 0 } , b) = $\emptyset$.

At this point Q = { {0} , { 1 , 2 }, $\emptyset$ } .

Then since { 1 , 2 } is now in **Q**, the transitions from { 1 , 2 } on symbols a and b are computed. Since δ_2(1 , a) = { 1 , 2 } , and δ_2(2 , a) = $\emptyset$, δ ({ 1 , 2 } , a) = { 1 , 2 } . Similarly δ ({ 1 , 2 } , b) = { 1 , 3 } . Thus { 1 , 3 } is added to **Q** .

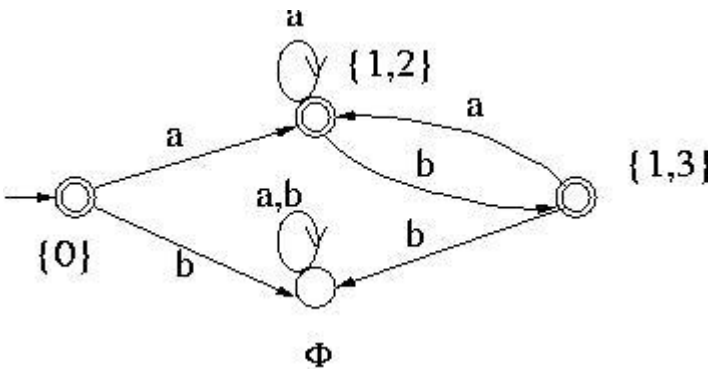
Similarly δ ({ 1 , 3 } , a) = { 1 , 2 } and δ ({ 1 , 3 } , b) = $\emptyset$. Thus no new states are added to **Q** . Since the transitions from all states of **Q** have been computed and no more states are added to **Q**, the conversion process stops here.

Note that there are no states of Q_2 in $\emptyset$. Hence there are no states that M_2 can go to from $\emptyset$. Hence δ

$\delta(\emptyset, a) = \emptyset$ $\delta(\emptyset, b) = \emptyset$.

For the accepting states of M, since states 0 and 1 are the accepting states of the NFA, all the states of **Q** that contain 0 and/or 1 are accepting states. Hence { 0 }, { 1 , 2 } and { 1 , 3 } are the accepting states of M.

The DFA thus obtained is shown below.



Example 2: Similarly the NFA

is converted to the following DFA:

Test Your Understanding of Conversion of NFA to DFA

Indicate which of the following statements are correct and which are not.
Click Yes or No , then Submit.
There are two sets of questions.

Answer the questions below on converting the following NFA to DFA.

The following notation is used in the questions:

δ : \delta

Converting NFA To DFA Compile Design in C Source Code Programming

Include the Necessary Package, Declare the variable in array. I use the checke(char a), push(char a), pop(),pushd(char *a) function to perform the NFA to DFA conversion.

Algorithm Source Code NFA to DFA Example

```
#include<iostream.h>
#include<string.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
char nfa[50][50],s[20],st[10][20],eclos[20],input[20];
int x,e,top=0,topd=0,n=0,ns,nos,in;
int checke(char a)
{
int i;
for(i=0;i<e;i++)
{
if(eclos[i]==a)
return i;
```

```
}
return -1;
}
int check(char a)
{
int i;
for(i=0;i<in;i++)
{
if(input[i]==a)
return i;
}
return -1;
}
void push(char a)
{
s[top]=a;
top++;
}
char pop()
{
top--;
return s[top];
}
void pushd(char *a)
{
strcpy(st[topd],a);
topd++;
}

char *popd()
{
topd--;
return st[topd];
}
int ctoi(char a)
{
int i=a-48;
return i;
}
char itoc(int a)
{
char i=a+48;
```

```
return i;
}
char *eclosure(char *a)
{
int i,j;
char c;
for(i=0;i<strlen(a);i++)
push(a[i]);
e=strlen(a);
strcpy(eclos,a);
while(top!=0)
{
c=pop();
for(j=0;j<ns;j++)
{
if(nfa[ctoi(c)][j]=='e')
{
if(check(itoc(j))==-1)
{
eclos[e]=itoc(j);
push(eclos[e]);
e++;
}
}
}
}
eclos[e]='\0';
return eclos;
}

void main()
{
int i,j,k,count;
char ec[20],a[20],b[20],c[20],dstates[10][10];
clrscr();
cout<<"Enter the number of states"<<endl;
cin>>ns;
for(i=0;i<ns;i++)
{
for(j=0;j<ns;j++)
{
cout<<"Move["<<i<<"]["<<j<<"]";
```

```
cin>>nfa[i][j];
if(nfa[i][j]!='-'&&nfa[i][j]!='e')
{
if((check(nfa[i][j]))==-1)
input[in++]=nfa[i][j];
}
}
}
topd=0;
nos=0;
c[0]=itoc(0);
c[1]='\0';
pushd(eclosure(c));
strcpy(dstates[nos],eclosure(c));
for(x=0;x<in;x++)
cout<<"\t"<<input[x];
cout<<"\n";
while(topd>0)
{
strcpy(a,popd());
cout<<a<<"\t";
for(i=0;i<in;i++)
{
int len=0;
for(j=0;j<strlen(a);j++)
{
int x=ctoi(a[j]);
for(k=0;k<ns;k++)
{
if(nfa[x][k]==input[i])
ec[len++]=itoc(k);
}
}
ec[len]='\0';
strcpy(b,eclosure(ec));
count=0;
for(j=0;j<=nos;j++)
{
if(strcmp(dstates[j],b)==0)
count++;
}
if(count==0)
```

```
{
if(b[0]!='\0')
{
nos++;
pushd(b);
strcpy(dstates[nos],b);
}
}
cout<<b<<"\t";
}
cout<<endl;
}
getch();
}
```

OUTPUT NFA to DFA Example

```
Enter the number of states
5
Move[0][0]-
Move[0][1]e
Move[0][2]-
Move[0][3]e
Move[0][4]-
Move[1][0]-
Move[1][1]-
Move[1][2]a
Move[1][3]-
Move[1][4]-
Move[2][0]-
Move[2][1]e
Move[2][2]-
Move[2][3]e
Move[2][4]-
Move[3][0]-
Move[3][1]-
Move[3][2]-
Move[3][3]-
Move[3][4]b
Move[4][0]-
Move[4][1]-
Move[4][2]-
```

Move[4][3]-
Move[4][4]-
a b
013 213 4
4
213 213 4

OUTPUT NFA to DFA

Enter the number of states
6

Move[0][0]-
Move[0][1]a
Move[0][2]-
Move[0][3]-
Move[0][4]-
Move[0][5]-
Move[1][0]-
Move[1][1]-
Move[1][2]b
Move[1][3]-
Move[1][4]-
Move[1][5]-
Move[2][0]-
Move[2][1]-
Move[2][2]-
Move[2][3]a
Move[2][4]e
Move[2][5]-
Move[3][0]-
Move[3][1]-
Move[3][2]c
Move[3][3]-
Move[3][4]e

a b

0 1
1 24
24 3244 5

5
3244 3244 55
55