Course on

Formal Language & Automata Theory

Department of CSE, GIET University, Gunupur, Odisha

Outline of the Lecture

- Definition of NFA and NFA with Ɛ
- Design NFA and NFA with Ɛ
- Acceptance of language by NFA
- NFA with Ɛ to NFA
- NFA to DFA
- Minimum DFA
- Equivalence of DFA

Non Deterministic Finite Automata (NFA /NDFA)

Definition: It is a model of machine that has finite set of state and transition function , where the machine in a current state takes i/p symbol and **not uniquely determine** the next state.

Formal Definition

NFA is defined as $M = (Q, \Sigma, \delta, q_0, F)$

Where Q:Finite set of states

Σ : Finite Alphabet

δ :Transition function from Qx$\Sigma \rightarrow 2^Q$

q_0 :Initial/Start State

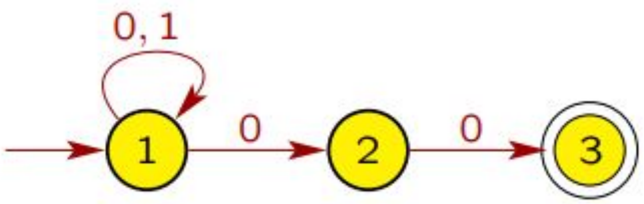
F :Set of final/accepting state

NFA Design over $\Sigma = (0, 1)$

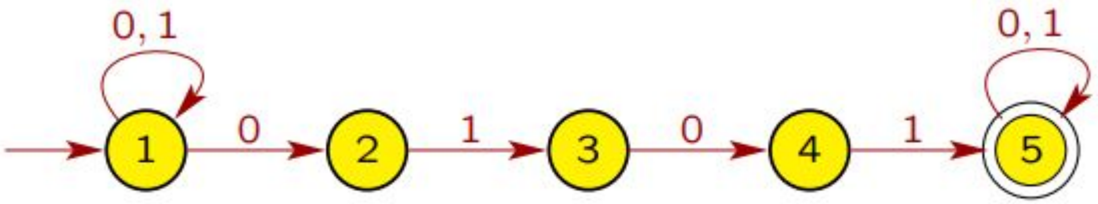
1. Ended with 00
2. Having substring 0101.

Solution:

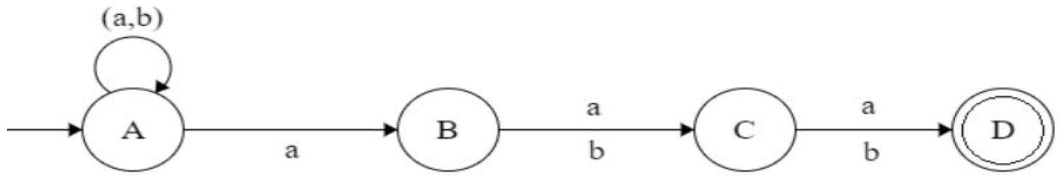
The language { w ∈ Σ* | w ends with 00 }.



Having substring 0101



3. Third symbol from right is a over $\Sigma = (a, b)$



Acceptance of string by NFA

A string w accepted by NFA 'M' if $\delta(q_0,w)$= p for some p in final state F.(ie we can reach to the final state after processing to all symbol of the string)

Note: $\delta(\{q_0,q_1\},w)=\delta(q_0,0) \cup \delta(q_1,0)$

Example: check the string **abb** is accepted by NFA or not

δ(A, abb)=δ(δ (A, a)bb)= δ({A, B}, bb) = δ(δ({A, B}, b)b)
= δ(δ(A,b) U δ(B,b),b) = δ({ A, C},b) = δ(A,b) U δ(C,b)= {A, **D**}
As D € F , the language is accepted by NFA

Non Deterministic Finite Automata with Ɛ move(NFA with Ɛ)

Definition: It is a model of machine that has finite set of state and transition function , where the machine in a current state takes i/p symbol including **Ɛ** and **not uniquely determine** the next state.

Formal Definition

NFA with **Ɛ** is defined as $M = (Q, \Sigma, \delta, q_0, F)$

Where Q:Finite set of states

Σ : Finite Alphabet

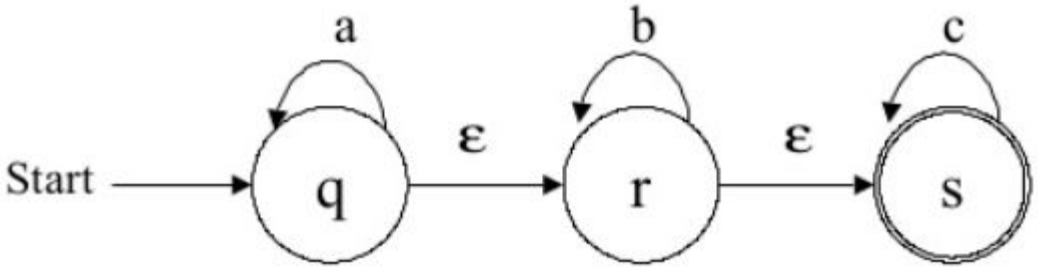
δ :Transition function from Qx (Σ U **Ɛ**) → 2^Q

q_0 :Initial/Start State

F :Set of final/accepting state

Example Σ= (a, b)

Accept the language having any number of a followed by any number of b followed by any number of c



ε-closure of a state : The ε-closure of the state q, denoted ECLOSE(q), is the set that contains q, together with all states that can be reached starting at q by following only ε-transitions.

ε-closure(q)= { q, r, s}

ε-closure(r)= {r, s}

ε-closure(s)= {s}

Construct NFA without ε from NFA with ε

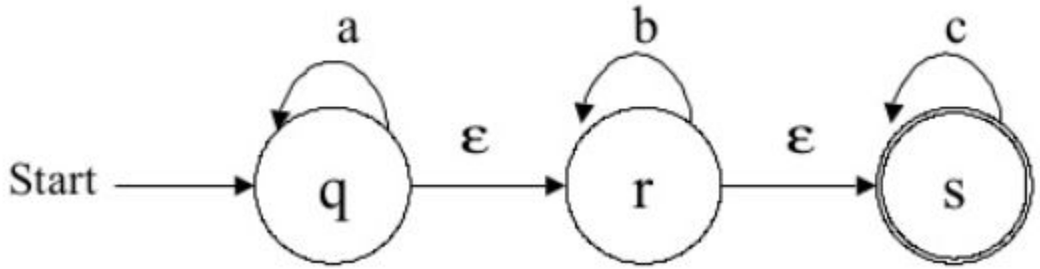
Let NFA with **ε** is M= (Q, Σ, δ, q_0, F)

We have to construct **NFA without ε , let it me M¹=**(Q, Σ, δ^1, q_0, F^1)

Step 1: find ε-closure of all state. If ε-closure of any state contains final state of NFA with ε than that will be the final state of NFA without ε.

Step 2: find transition δ^1 by the rules δ^1(q, a)= **ε-closure(δ(ε-closure(q),a))**

Example:



ε-closure(q)= { q, r, s},so q will final state in NFA without ε

ε-closure(r)= {r, s} , so r will final state in NFA without ε

ε-closure(s)= {s} , so s will final state in NFA without ε

$M^1=(Q, \Sigma, \delta^1, q_0, F^1)$

$Q=\{q,r,s\}$, $\Sigma=\{a, b, c\}$, $q_0=\{q_0\}$, $F^1=\{q, r, s\}$

δ^1:

$\delta^1(q, a)=$ ε-closure(δ(ε-closure(q),a))= { q, r, s}

$\delta^1(q, b)=$ ε-closure(δ(ε-closure(q),b))= { r, s}

$\delta^1(q, c)=$ ε-closure(δ(ε-closure(q),c))= { s}

$\delta^1(r, a)=$ ε-closure(δ(ε-closure(r),a))= {}

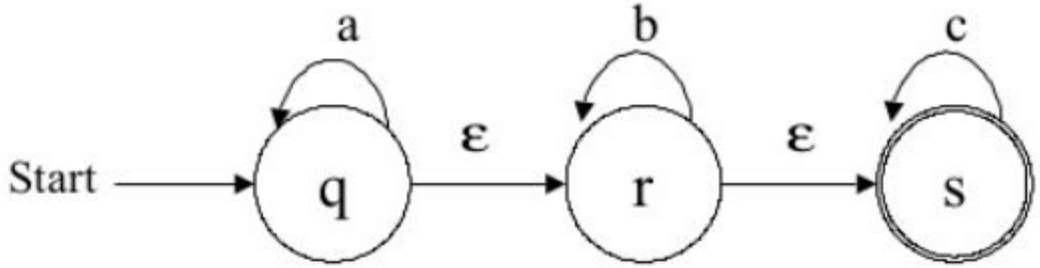
$\delta^1(r, b)=$ ε-closure(δ(ε-closure(r),b))= { r, s}

$\delta^1(r, c)=$ ε-closure(δ(ε-closure(r),c))= {s }

$\delta^1(s, a)=$ ε-closure(δ(ε-closure(s),a))= {}

$\delta^1(s, b)=$ ε-closure(δ(ε-closure(s),b))= {}

$\delta^1(s, c)=$ ε-closure(δ(ε-closure(s),c))= {s }



Example:

Construct NFA without ε from the given NFA with ε

state/Input	0	1	€
-->q0	q1		
q1(F)	q1		q2
q2		q0	

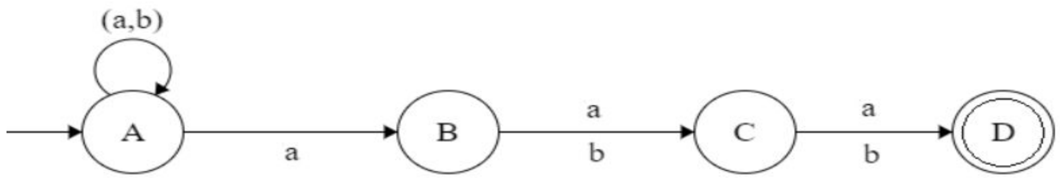
ε-closure(q0)={q0}
ε-closure(q1)={q1, q2)
ε-closure(q2)={q2}
So final state will be {q1}

δ^1(q0, 0)= ε-closure(δ(ε-closure(q0),0))= { q1,q2},
δ^1(q0, 1)= ε-closure(δ(ε-closure(q0),1))= { }
δ^1(q1, 0)= ε-closure(δ(ε-closure(q1),0))= { q1, q2},
δ^1(q1, 1)= ε-closure(δ(ε-closure(q1),1))= {q0}
δ^1(q2, 0)= ε-closure(δ(ε-closure(q2),0))= { },
δ^1(q2, 1)= ε-closure(δ(ε-closure(q2),1))= {q0 }

NFA and DFA are equivalent

Example:

Construct DFA from the given NFA without ε



State/∑	a	b
-->A	{A, B }	A
B	C	C
C	D	D
D(F)	φ	φ

Solution:

State/∑	a	b
-->[A]	[A,B]	[A]
[A,B]	[A,B,C]	[A,C]
[A,C]	[A,B,D]	[A,D]
[A,D]	[A,B]	[A]
[A,B,C]	[A,B,C,D]	[A,C,D]
[A,B,D]*	[A,B,C]	[A,C]
[A,C,D]*	[A,B,D]	[A,D]
[A,B,C,D]*	[A,B,C,D]	[A,C,D]

State/∑	a	b
-->[A]	[A, B]	[A]
[A, B]	[A,B,C]	[A,C]
[A,C]	[A, B, D]	[A,D]
[A,D]*	[A, B]	[A]
[A, B,C]	[A, B, C, D]	[A,C,D]
[A, B, D]*	[A, B,C]	[A,C]
[A,C,D]*	[A, B,D]	[A,D]
[A, B, C, D]*	[A, B,C,D]	[A, C,D]

Construct DFA from the given NFA without ε

Construct a deterministic automaton equivalent to given NFA

State/∑	a	b
-->q0(F)	q0	q1
q1	q1	q0,q1

Find a deterministic acceptor equivalent to

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$$

State/∑	a	b
-->[q0](F)	[q0]	[q1]
[q1]	[q1]	[q0,q1]
[q0,q1] (F)	[q0,q1]	[q0,q1]

Example: Construct a deterministic automaton equivalent to given NFA

State/∑	a	b
-->q0	{q0, q1}	q0
q1	q2	q1
q2	q3	q3
q3(F)		q2

State/∑	a	b
□[q0]	[q0,q1]	[q0]
[q0,q1]	[q0,q1,q2]	[q0,q1]
[qo,q1,q2]	[q0,q1,q2,q3]	[q0,q1,q3]
[q0,q1,q3]*	[q0,q1,q2]	[q0,q1,q2]
[q0,q1,q2,q3]*	[q0,q1,q2,q3]	[q0,q1,q2,q3]

Solution

Let $Q = \{q_0, q_1, q_2, q_3\}$. Then the deterministic automaton M_1 equivalent to M is given by

$$M_1 = (2^Q, \{a, b\}, \delta, [q_0], F)$$

State/∑	a	b
□[q0]	[q0, q1]	[q0]
[q0, q1]	[q0,q1,q2]	[q0, q1]
[q0,q1,q2]	[q0,q1,q2,q3]	[q0, q1,q3]
[q0, q1,q3]*	[q0, q1,q2]	[q0, q1,q2]
[q0,q1,q2,q3]*	[q0,q1,q2,q3]	[q0,q1,q2,q3]

Theorem:

Any transition function δ, input string x and symbol y

$\delta(q, xy)= \delta(\delta(q,x),y)$

Theorem:

Any transition function δ and two input string x and y

$\delta(q, xy)= \delta(\delta(q, x),y)$

Proof:

We can proof by method of induction for |y|

Let $|y|= 1$ ie $y=a$

By known result $\delta(q, xa)= \delta(\delta(q,x),a)$

Let it is true for $|y|= n$

$\delta(q, xy)= \delta(\delta(q,x),y)$

We have to prove for $|y| = n+1$;

LHS: $\delta(q, xy)= \delta(q, xy_1a) =\delta(q,x_1 a) = \delta(\delta(q, x1),a) = \delta(\delta(q, xy_1),a) = \delta(\delta(\delta(q, x)y_1),a)$

$= \delta(\delta((q, x),y_1a)= \delta(\delta(q, x),y)$

RHS: $\delta(\delta(q,x),y) =\delta(\delta(q,x),y_1a) == \delta(\delta(\delta(q, x)y_1)a) = \delta(\delta((q, x),y_1a)= \delta(\delta(q, x),y)$

Hence it is true for all n

Theorem:

Every nondeterministic finite automata has an equivalent deterministic finite automata

Proof:

Let N= (Q, ∑, δ,q_0,F) be an NFA recognizing some language A.

We construct a DFA M recognizing A.

Consider N has no ε.

Consider M = (Q^1, ∑, δ^1,$q_0{}^1$,F^1)

1. Q^1=P(Q) every state of M is a set of state of N
2. For R € Q^1 and a € ∑ let δ^1(R, a) ={ q € Q| q € δ(r, a)for some r € R} if R is a state of M, it also set of state of N. When M reads a symbol a in state R, it shows where a takes each state of R. we can write δ(R, a) = $U_{r€R}$ δ(r,a)
3. 3. $q_0{}^1$={q_0 }

4. F^1=(R €Q^1|R contains an state of N}
   The machine M accepts if one of the possible state that N could be in this point is an accept state.

Now consider N ε move.

For any state R of M we defined E(R) to be collection of state that can reach from R going on ε arrow including the member of R themselves for R subset of Q

E(R)= { q| q can be reached from R by traveling along ε only }

We modify the transition function of M by

δ^1(R, a)= { q ε Q|q ε E(δ(r, a)) for some r ε R}

Now this DFA M equivalent to the NFA N.

Minimization of DFA:

Note: Equivalence: Two state q0 and q1 said to be equivalent if both δ(q0,x) and δ(q1,x), both are final state or both of them are non final state.

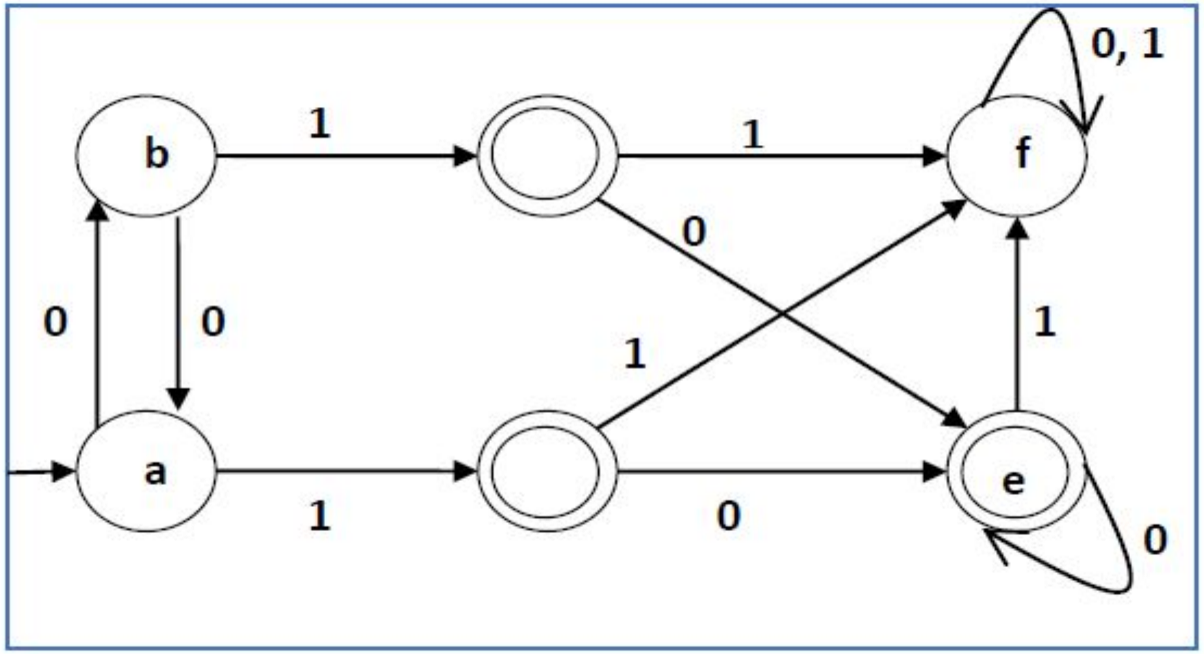
(k+1)Equivalent: Two state q0 and q1 said to be (k+1) equivalent if they are (k) equivalent and both δ(q0,x) and δ(q1,x), are also k- equivalent for every a in alphabet.

Algorithms Minimization of DFA

- Step 1: Remove all unreachable states (The states which can never be reached from initial state)
- Step 2 : Divide all vertices into two sets i.e. One set of final states and second one for non-final states.
- Step 3: For each state in both the sets , find the transition of an alphabet (only one alphabet) .
  If the transition state is the element of another set or class then separate these states from that set and make another class.
- Step 4: Repeat the Step 3 until all the classes are made.
- Step 5: Repeat Step 3 and 4 for all alphabets.
- Step 6 : After completion of all steps, draw the DFA

Example

Let us consider the following DFA:



q	$\delta(q,0)$	$\delta(q,1)$
a	b	c
b	a	d
c	e	f
d	e	f
e	e	f
f	f	f

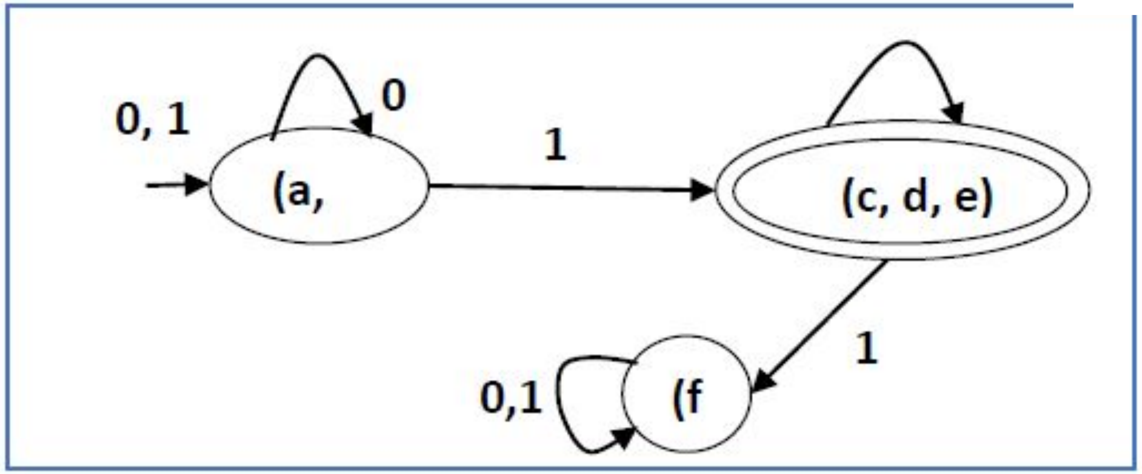
q	$\delta(q,0)$	$\delta(q,1)$
a	b	c
b	a	d
c	e	f
d	e	f
e	e	f
f	f	f

$\prod_0$={{c, d, e}, { a, b, f}}

$\prod_1$={c, d, e}, {a, b}, {f}
$\prod_2$={c, d, e}, {a, b}, {f}

There are three states in the reduced DFA. The reduced DFA is as follows:

Q	$\delta(q,0)$	$\delta(q,1)$
(a, b)	(a, b)	(c,d,e)
(c,d,e)	(c,d,e)	(f)
(f)	(f)	(f)



State Table and State Diagram of Reduced DFA

Minimization of DFA:

State/∑	0	1
-->q0	q1	q5
q1	q6	q2
q2(f)	q0	q2
q3	q2	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

$\prod_0$={ {q2}, {q0, q1, q3, q4, q5, q6, q7} }
$\prod_1$={ {q2}, {q0, q4, q6,},{ q1, q7}, { q3, q5} }
$\prod_2$={ {q2}, {q0, q4},{q6}, { q1, q7}, { q3, q5} }
$\prod_3$={ {q2}, {q0, q4},{q6}, { q1, q7}, { q3, q5} }

M=(Q^1,{0,1}, δ^1, $q0^1$, F^1}
Where
Q^1={ [q2],[q0,q4],[q6],[q1,q7],[q3,q5]}
$q0^1$={**[q0,q4]**}

State/∑	0	1
-->q0	q1	q5
q1	q6	q2
q2(f)	q0	q2
q3	q2	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

State/∑	0	1
-->[q0,q4]	[q1,q7]	[q3,q5]
[q1,q7]	[q6]	[q2]
[q2](F)	[q0,q4]	[q2]
[q3,q5]	[q2]	[q6]
[q6]	[q6]	[q0,q4]

Q^1={ [q2],[q0,q4],[q6],[q1,q7],[q3,q5]}

Minimization of DFA:

State/∑	A	b
-->q0	q0	q3
q1	q2	q5
q2	q3	q4
q3	q0	q5
q4	q0	q6
q5	q1	q4
Q6(f)	q1	q3

$\prod_0$={ {q6}, { q0,q1,q2,q3,q4,q5}}

$\prod_1$={ {q6}, { q0,q1,q2,q3,q5} ,{q4} }

$\prod_2$={ { q6}, {q0,q1,q3} ,{q2q5},{q4}}

$\prod_3$={ { q6}, {q0}, { q1},{q3} ,{q2q5},{q4}}

$\prod_4$={ { q6}, {q0},{q1},{q3} ,{q2},{q5},{q4}}

Hence the give DFA is minimized one.

Minimization of DFA:

State/∑	a	b
-->q0	q1	q0
q1	q0	q2
q2	q3	q1
q3(f)	q3	q0
q4	q3	q5
q5	q6	q4
q6	q5	q6
q7	q6	q3

$\prod_0$={ { q3}, {q0,q1,q2,q4,q5,q6,q7}}

$\prod_1$={ {q3}, {q0,q1,q5,q6} ,{ q2,q4, q7} }

$\prod_1$={{q3}, {q0,q1,q5,q6} {q2,q4}{q7} }

∏2={{q3},{q0 ,q6} {q1,q5,}, {q2,q4}{q7}

∏2={{q3},{q0 ,q6}, {q1,q5,}, {q2,q4}{q7}

∏3={{q3},{q0 ,q6}, {q1,q5,}, {q2,q4}{q7}

M=(Q^1,{a,b}, δ^1, $q0^1$, F^1}

Where

Q^1={[q3],[q0,q6],[q1,q5],[q2,q4],[q7]}

$q0^1$={[q0,q6]}

Hence the minimal automata is ----(next page)

State/∑	a	b
-->[q0,q6]	[q1,q5]	[q0,q6]
[q1,q5]	[q0,q6]	[q2,q4]
[q0,q6]	[q1,q5]	[q0,q6]
[q2,q4]	[q3]	[q1,q5]
[q3](F)	[q3]	[q0,q6]

Q^1={[q3],[q0,q6],[q1,q5],[q2,q4],[q7]}
$q0^1$={**[q0,q6]**}

State/∑	a	b
-->q0	q1	q0
q1	q0	q2
q2	q3	q1
q3(f)	q3	q0
q4	q3	q5
q5	q6	q4
q6	q5	q6
q7	q6	q3

Minimization of DFA Algorithms:

Step 1: (construction $\prod_o$). by the definition of o-equivalence $\prod_o$={ q_1^0, q_2^0}, where q_1^0 is set of final state and q_2^0 is set of non final state.

Step 2: (construction $\prod_{k+1}$ from $\prod_k$). Let Q_i^k be any subset in $\prod_k$. If q1,q2 are in Q_i^k , they are (k+1) equivalent provided δ(q1,a) and δ(q2,a)are k-equivalent. Find out whether δ(q1,a) and δ(q2,a) are in same equivalence class in $\prod_k$ for every a €∑.if so q1 and q2 are (k+1) equivalent. Repeat this for every Q_i^k in Q_i^k to get all the element of $\prod_{k+1}$.

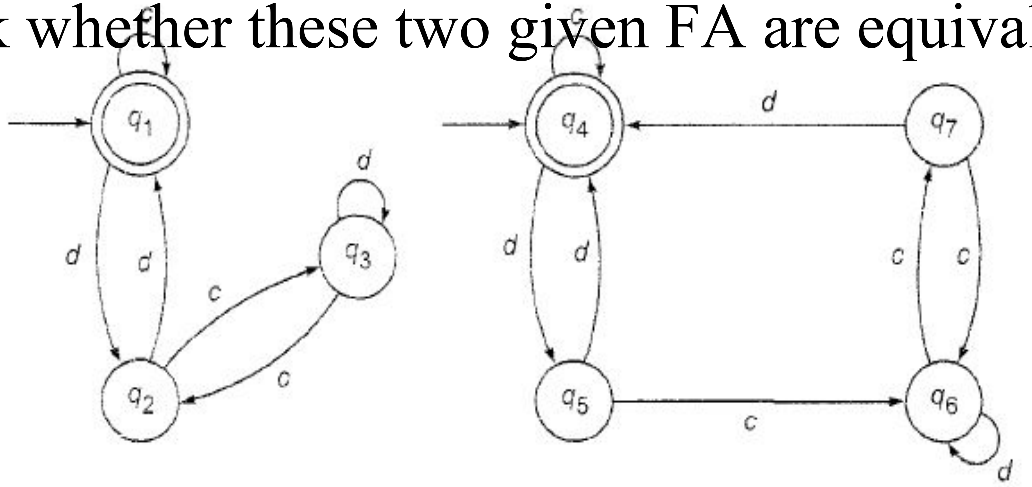
Step 3: construct $\prod_n$ for n=1, 2, 3, ... until $\prod_n$= $\prod_{n+1}$

Step 4:(construction of minimum Automata). For the required minimum state automata , the state are equivalence classes obtained in step 3.The state table is obtained by replacing a state q by the corresponding equivalence class [q].

EQUIVALENCE OF TWO FINITE AUTOMATA

Example:

Check whether these two given FA are equivalent or not



(q, q^1)	(q_c, q_c^1)	(q_d, q_d^1)
(q1,q4)		

Solution:

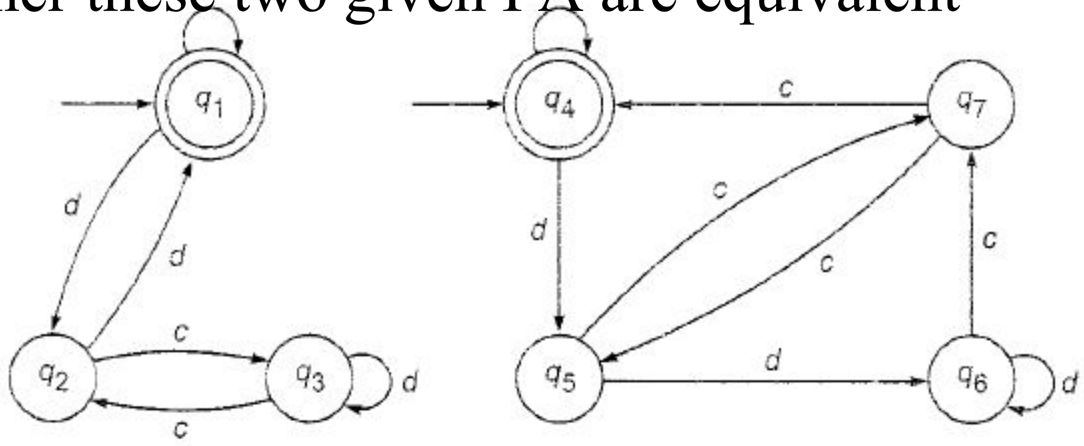
(q, q^1)	(q_c, q_c^1)	(q_d, q_d^1)
(q1,q4)	(q1,q4)	(q2,q5)
(q2,q5)	(q3,q6)	(q1,q4)
(q3,q6)	(q2,q7)	(q3,q6)
(q2,q7)	(q3,q6)	(q1,q4)

As in the pair (q, q^1), are both final state or both are non final state so two FA M and M^1 are equivalent.

EQUIVALENCE OF TWO FINITE AUTOMATA

Example:

Check whether these two given FA are equivalent or not



(a) (b)

(q, q^1)	(q_c, q_c^1)	(q_d, q_d^1)
(q1,q4)		

Solution:

(q, q^1)	(q_c, q_c^1)	(q_d, q_d^1)
(q1,q4)	(q1,q4)	(q2,q5)
(q2,q5)	(q3,q7)	(q1,q6)
(q1,q6)		

As in the pair (q1, q6), q1 is final state in M and q6 is non final state in M^1, the given two FA is not equivalent.

Thank You