Unit II
Introduction to Finite Automata

Prepared By:
Ghanashyam BK

Introduction to Finite Automata

- A finite automaton is a mathematical (model) abstract machine that has a set of "states" and its "control" moves from state to state in response to external "inputs".
- The control may be either "deterministic" meaning that the automation can't be in more than one state at any one time
- or "non deterministic", meaning that it may be in several states at once.
- This distinguishes the class of automata as DFA or NFA.

Applications

- The finite state machines are used in applications in computer science and data networking.
- For example, finite-state machines are basis for programs for spell checking, indexing, grammar checking, etc
- network protocols that specify how computers communicate.

Introduction of Finite State Machine

- a machine that can, at any point in time, be in a specific state from a finite set of possible states
- It can move (transition) to another state by accepting an input.
- The simplest machine is the finite state automaton

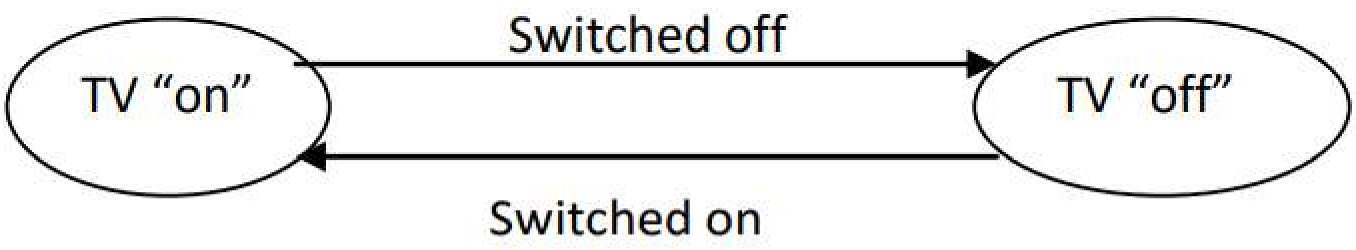


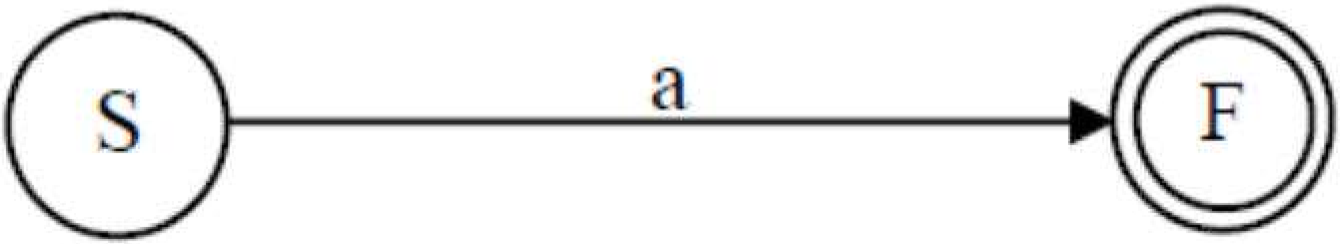
Fig: state diagram for TV on/off state

Deterministic Finite Automata (DFA)

- A deterministic finite automaton is defined by a quintuple (5-tuple) as $(Q, \Sigma, \delta, q0, F)$.
  - Where,
    - Q = Finite set of states,
    - Σ = Finite set of input symbols,
    - δ = A transition function that maps $Q \times \Sigma \to Q$
    - q_0 = A start state; $q_0 \in Q$
    - F = Set of final states; $F \subseteq Q$.
- A transistion function δ that takes as arguments a state and an input symbol and returns a state.

Deterministic Finite Automata (DFA)

- For example:



- If 'S' is a state and 'a' is an input symbol then δ(S,a) is that state F such that there are arcs labled 'a' from S to F.

How a DFA process strings?

- The first thing we need to understand about a DFA is how DFA decides whether or not to "accept" a sequence of input symbols.
- The "language" of the DFA is the set of all symbols that the DFA accepts.
- Suppose a_1, a_2, …… a_n is a sequence of input symbols.
- We start out with the DFA in its start state, q_0.
- We consult the transition function δ also for this purpose.

How a DFA process strings? (Contd…)

- Say $\delta\,(q_0, a_1) = q_1$ to find the state that the DFA enters after processing the first input symbol a_1.
- We then process the next input symbol a_2, by evaluating $\delta\,(q_1, a_2)$; suppose this state be q_2.
- We continue in this manner, finding states $q_3, q_4, \ldots, q_n$. such that $\delta\,(q_{i-1}, a_i) = q_i$ for each i.
- if q_n is a member of F, then input $a_1, a_2, \ldots, a_n$ is accepted & if not then it is rejected.

Notations for DFA

- There are two preferred notations for describing this class of automata
  - **Transition Diagram**
    - Which is a graph
  - **Transition Table**
    - which is a tabular listing of the δ function, which by implication tells us the set of states and the input alphabet.

Notations for DFA

Transition Diagram

- A transition diagram of a DFA is a graphical representation where; (or is a graph)
- For each state in Q, there is a node represented by circle,
- For each state q in Q and each input a in Σ, if δ (q, a) = p then there is an arc from node q to p labeled a in the transition diagram.
- If more than one input symbol cause the transition from state q to p then arc from q to p is labeled by a list of those symbols.
- The start state is labeled by an arrow written with "start" on the node.
- The final or accepting state is marked by double circle.

Notations for DFA

Transition Diagram Example

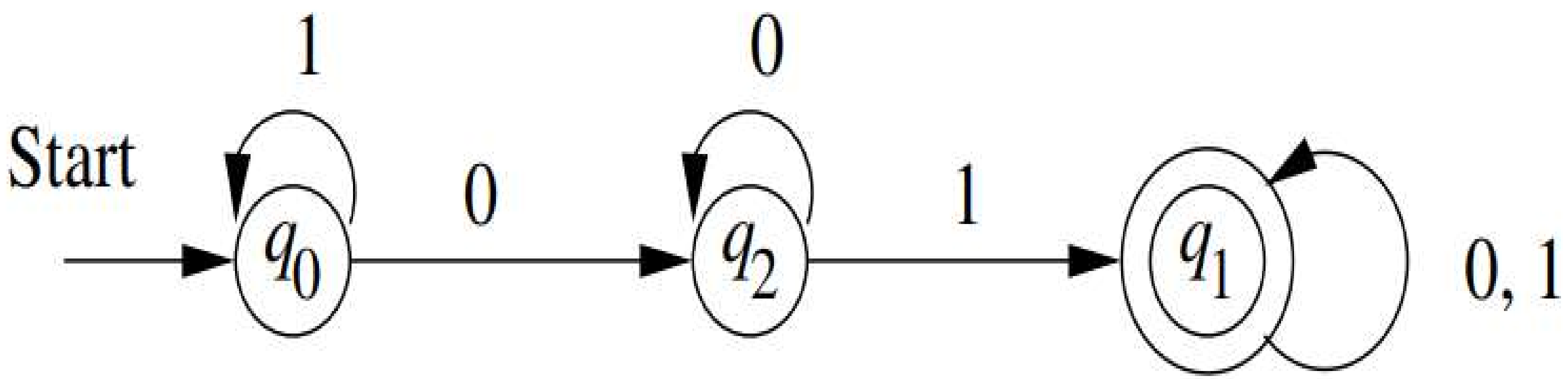


Fig: The transaction diagram for the DFA accepting all strings with a substring 01

Notations for DFA

Transition Table

- Transition table is a conventional, tabular representation of the transition function δ that takes the arguments from $Q \times \Sigma$ & returns a value which is one of the states of the automation.
- The row of the table corresponds to the states while column corresponds to the input symbol.
- The starting state in the table is represented by → followed by the state i.e. →q, for q being start state, whereas final state as *q, for q being final state.

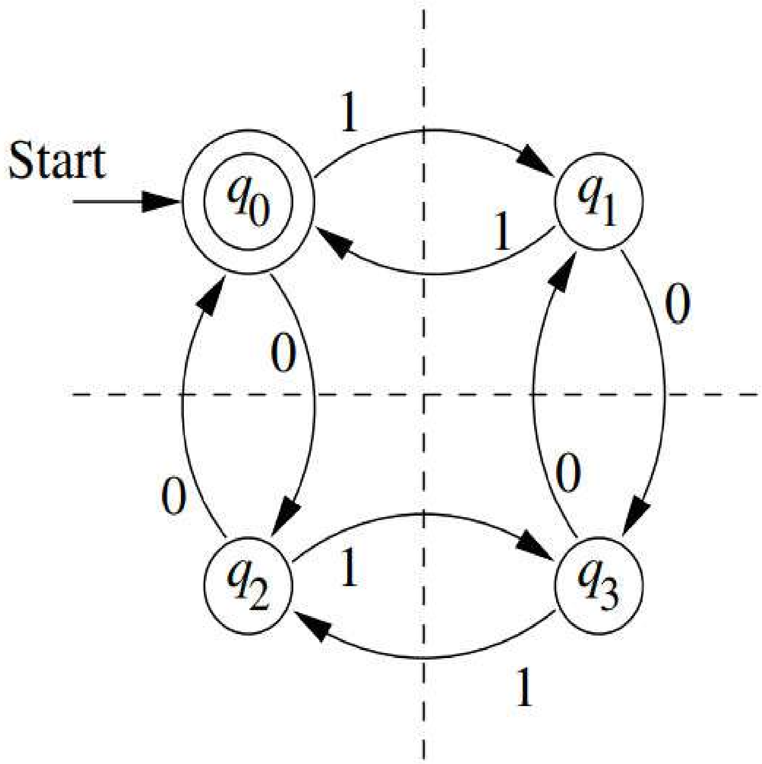
Notations for DFA

Transition Table Example

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Fig: The transaction table for the DFA accepting all strings with a substring 01

Example 2



	0	1
$* \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Fig: The state diagram & transaction table for the DFA accepting all strings with even number of 0's & even number of 1's

Extended Transition Function of DFA

- The extended transition function of DFA, denoted by $\hat{\delta}$ is a transition function that takes two arguments as input, one is the state q of Q and another is a string w∈ Σ*, and generates a state p ∈ Q.
- This state p is that the automaton reaches when starting in state q & processing the sequence of inputs w.
- i.e. $\hat{\delta}(q, w) = p$

Extended Transition Function of DFA

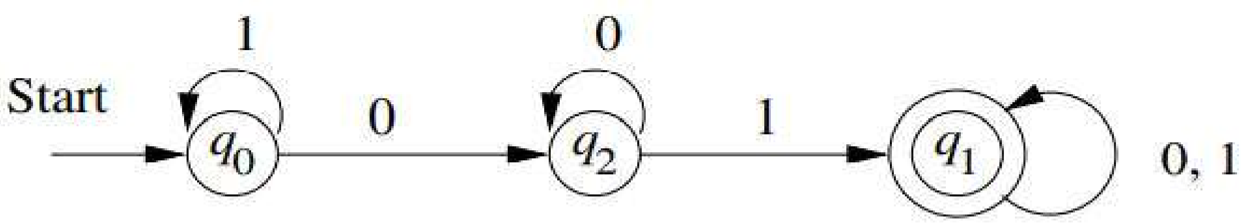
- Let us define by induction on length of input string as follows:
- **Basis step:** $\hat{\delta}(q,\epsilon)$ =q. i.e. from state q, reading no input symbol stays at the same state.
- **Induction:** Let w be a string from Σ^* such that w = xa, where x is substring of w without last symbol and a is the last symbol of w, then $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$.
- Thus, to compute $\hat{\delta}(q, w)$, we first compute $\hat{\delta}(q, x)$, the state the automaton is in after processing all but last symbol of w. let this state is p, i.e. $\hat{\delta}(q, x) = p$.
- Then, $\hat{\delta}(q, w)$ is what we get by making a transition from state p on input a, the last symbol of w. i.e. $\hat{\delta}(q, w) = \delta(p, a)$

Extended Transition Function of DFA

- Compute $\hat{\delta}$ (q_0, w) for each prefix w of 110101, starting at ϵ and going in increasing size.
  - $\hat{\delta}(q_0, \epsilon) = q_0$.
  - $\hat{\delta}(q_0, 1) = \delta\big(\hat{\delta}(q_0, \epsilon), 1\big) = \delta(q_0, 1) = q_1$.
  - $\hat{\delta}(q_0, 11) = \delta\big(\hat{\delta}(q_0, 1), 1\big) = \delta(q_1, 1) = q_0$.
  - $\hat{\delta}(q_0, 110) = \delta\big(\hat{\delta}(q_0, 11), 0\big) = \delta(q_0, 0) = q_2$.
  - $\hat{\delta}(q_0, 1101) = \delta\big(\hat{\delta}(q_0, 110), 1\big) = \delta(q_2, 1) = q_3$.
  - $\hat{\delta}(q_0, 11010) = \delta\big(\hat{\delta}(q_0, 1101), 0\big) = \delta(q_3, 0) = q_1$.
  - $\hat{\delta}(q_0, 110101) = \delta\big(\hat{\delta}(q_0, 11010), 1\big) = \delta(q_1, 1) = q_0$.

Since, q0 is final state in example 2 so it is accepted

Extended Transition Function of DFA (Assignment- 2)



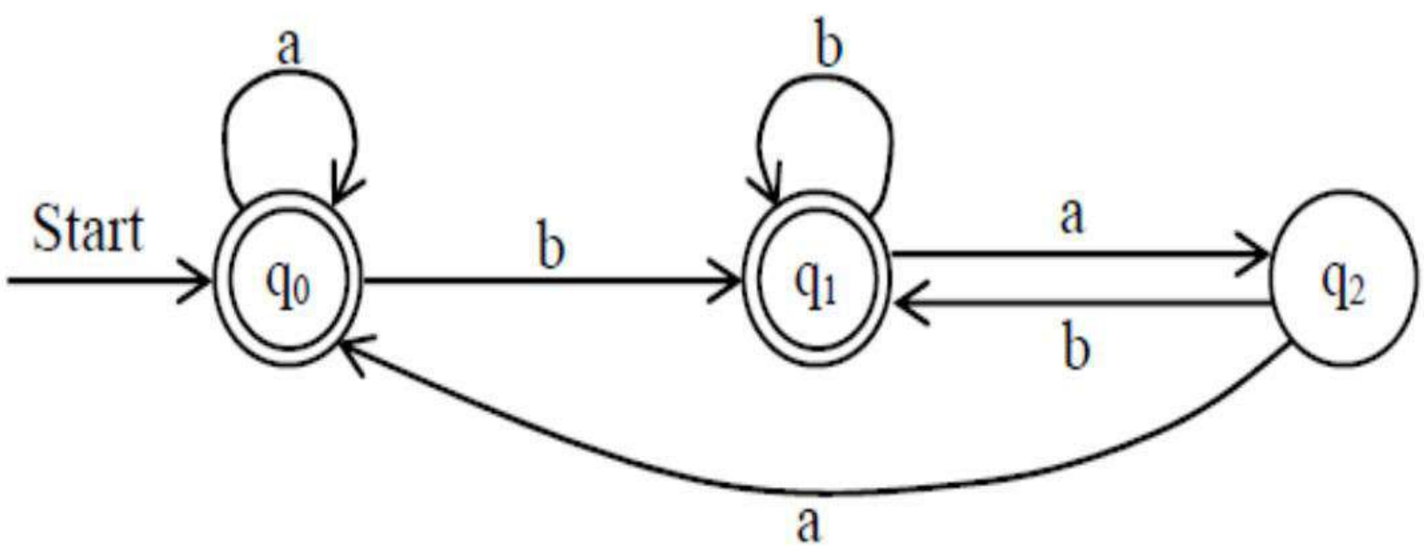
- Compute $\hat{\delta}(q0, 1001)$
- Compute $\hat{\delta}(q0, 100)$

Language of DFA

- The language of DFA $M = (Q, \Sigma, \delta, q0, F)$ denoted by $L(M)$ is a set of strings over Σ^* that are accepted by M.
- i.e; $L(M) = \{w / \hat{\delta} (q0, w) = p \in F\}$
- i.e. the language of a DFA is the set of all strings w that take DFA starting from start state to one of the accepting states.
- The language of DFA is called regular language.

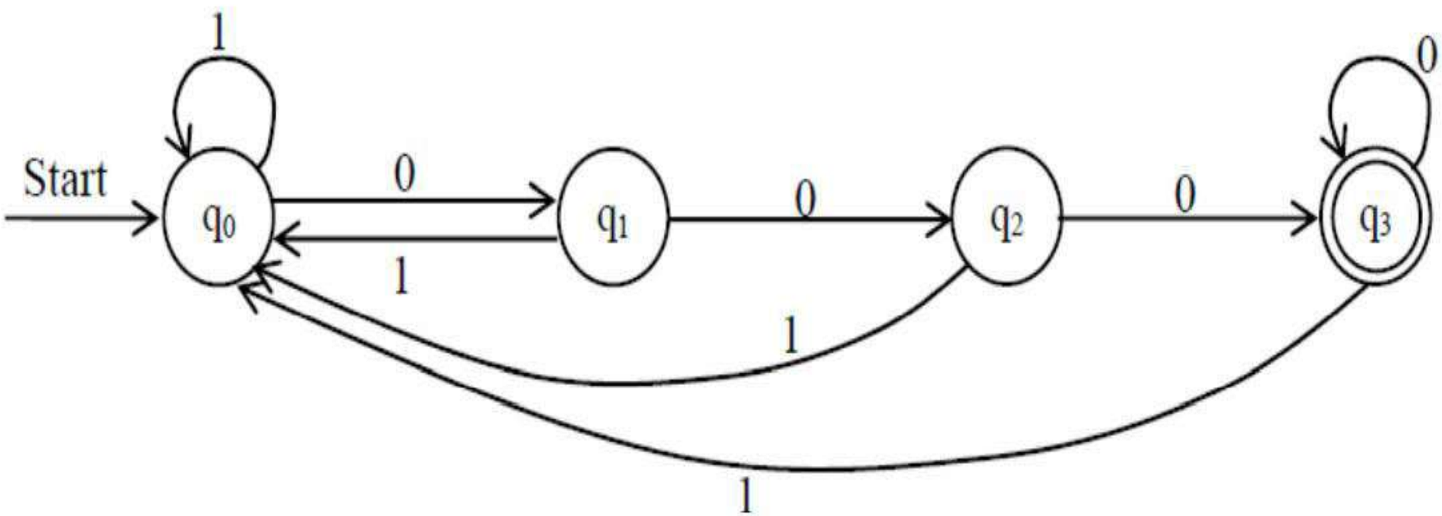
Examples

- Construct a DFA, that accepts all the strings over $\Sigma = \{a, b\}$ that do not end with ba.



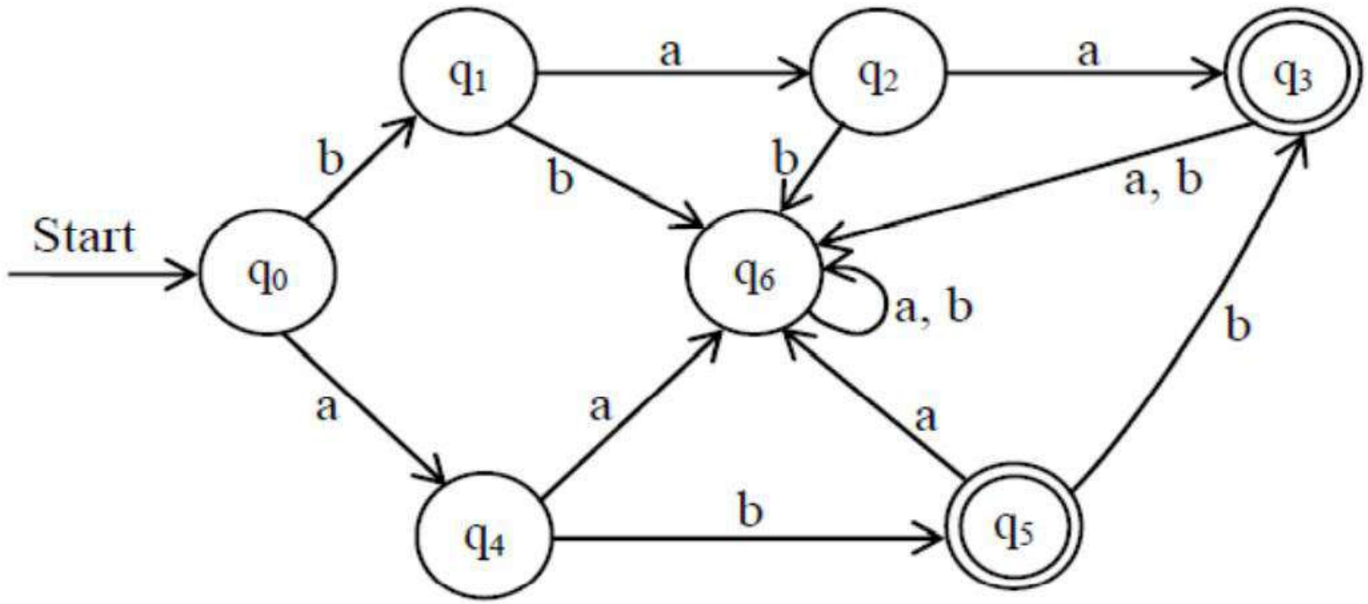
Examples

- DFA accepting all string over $\Sigma = \{0, 1\}$ ending with 3 consecutive 0‟s.



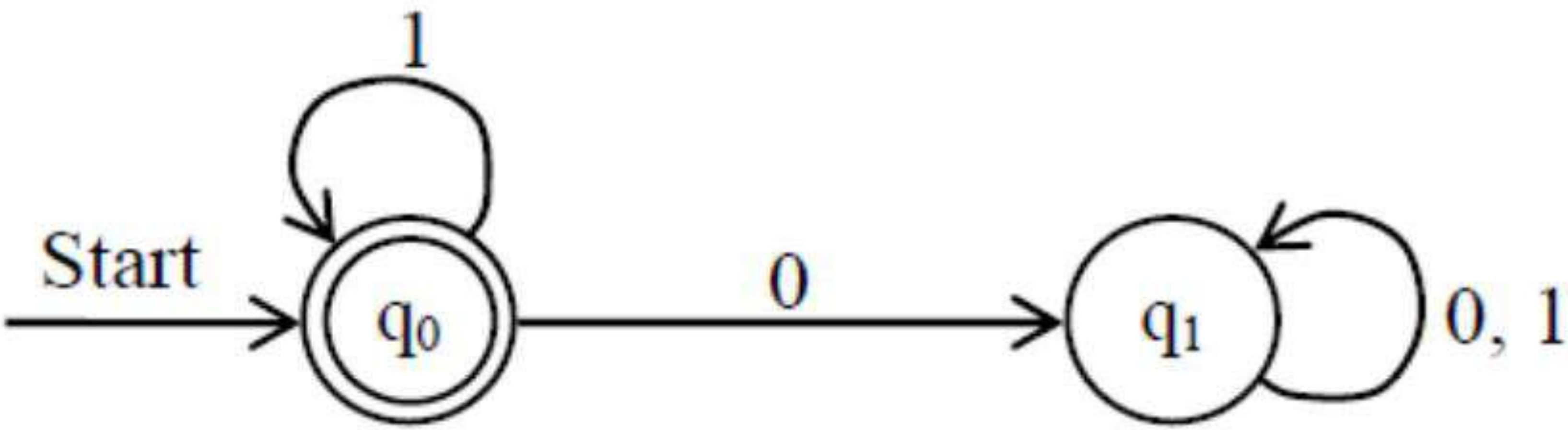
Examples

- DFA over {a, b} accepting {baa, ab, abb}



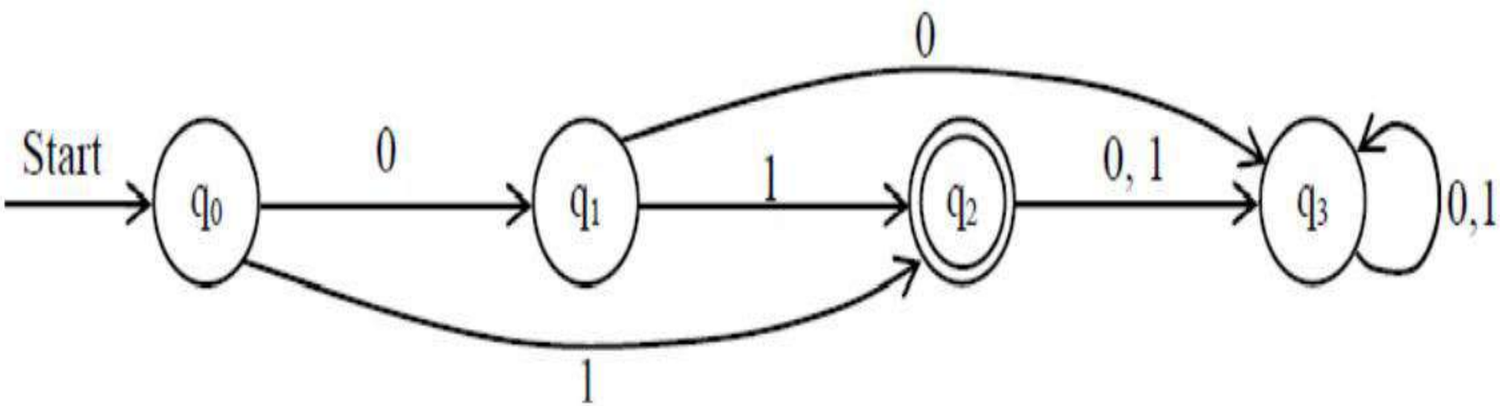
Examples

- DFA accepting zero or more consecutive 1's. i.e. L (M) = {1n | n = 0, 1, 2, ……}



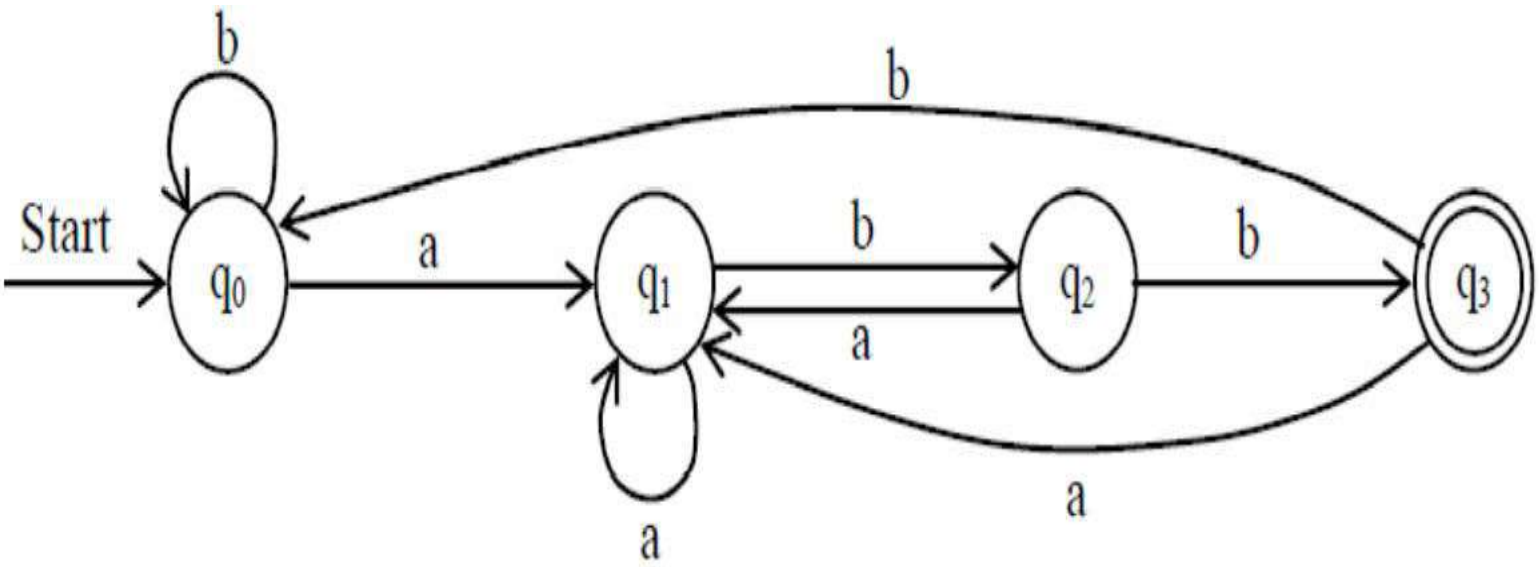
Examples

- DFA over $\{0, 1\}$ accepting $\{1, 01\}$



Examples

- DFA over {a, b} that accepts the strings ending with abb.



Assignment-3

- Give the DFA for the language of string over {0.1} in which each string end with 11
- Give the DFA accepting the string over {a,b} such that each string does not end with ab.
- Give the DFA for the language of string over {a,b} such that each string contain aba as substring
- Give the DFA for the langague of string over {0,1} such that each string start with 01
- Give the DFA for the langague of string over {0,1} such that set of all string ending in 00.
- Give the DFA for the langague of string over {0,1} such that set of strings with 011 as a substring.

Non-Deterministic Finite Automata (NFA)

- NFA can also be interpreted by a quintuple; $(Q, \Sigma, \delta, q_0, F)$
  - Where,

    Q = A finite set of states

    Σ = A finite set of input symbols, (alphabets)

    δ = A transition function that maps state symbol pair to sets of states i.e. δ is $Q \times \Sigma = 2^Q$

    A state $q_0 \in Q$, that is distinguished as a start (initial) state.

    A set of final states F distinguished as accepting (final) state. $F \subseteq Q$.
- Unlike DFA, a transition function in NFA takes the NFA from one state to several states just with a single input.

Non-Deterministic Finite Automata (NFA)

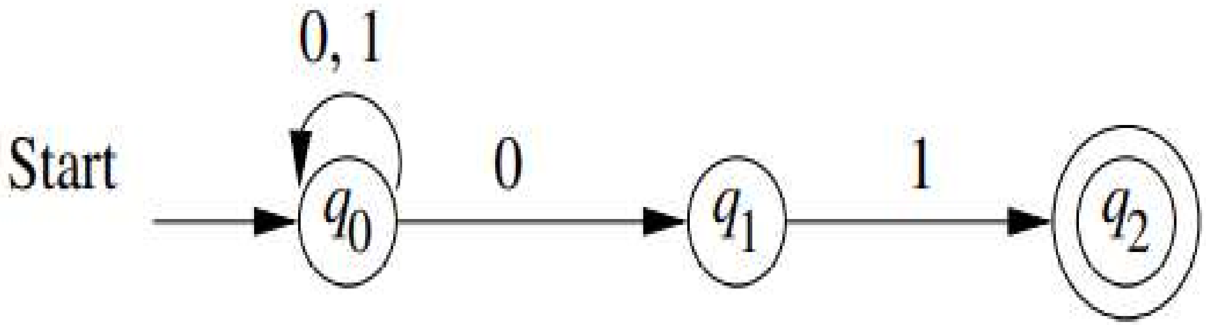
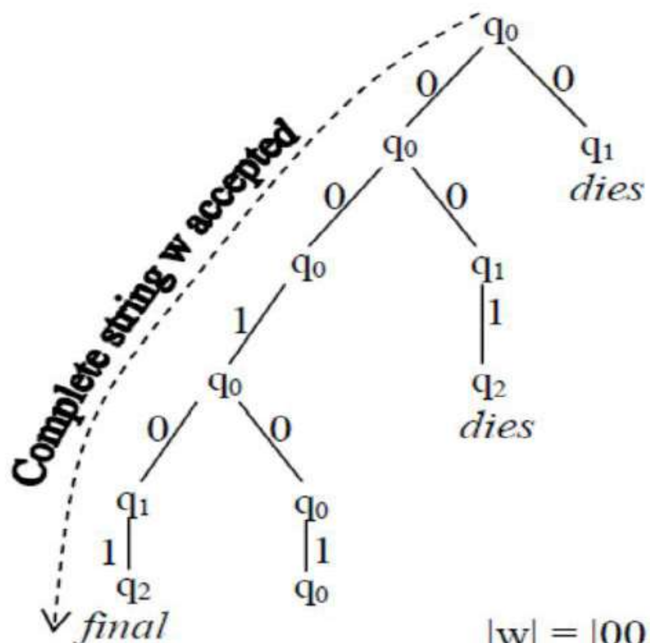


Fig: NFA accepting all strings that end in 01

Non-Deterministic Finite Automata (NFA)

- For input sequence w = 00101, the NFA can be in the states during the processing of the input are as:



$|w| = |00101| = 5$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

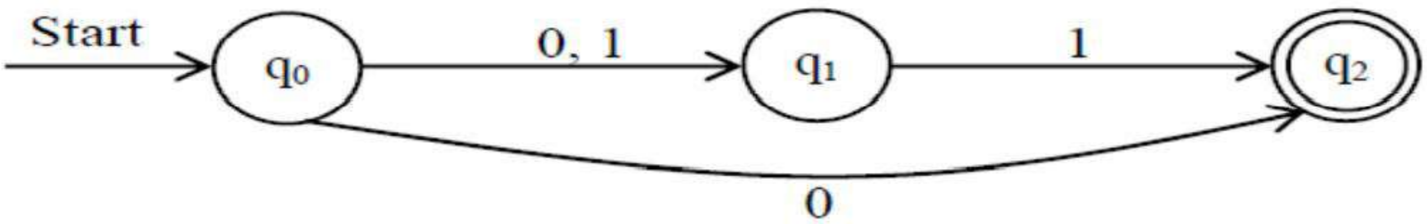
$F = \{q_2\}$

Transition table:

δ:	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\phi\}$	$\{q_2\}$
$*q_2$	$\{\phi\}$	$\{\phi\}$

Non-Deterministic Finite Automata (NFA)

- NFA over {0, 1} accepting strings {0, 01, 11}

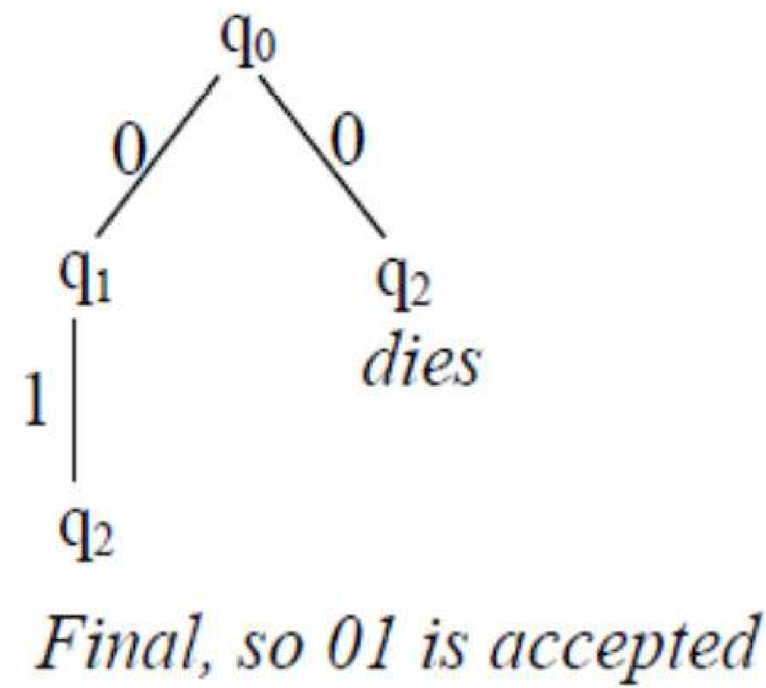


Transition table:

δ:	0	1
$\rightarrow q_0$	$\{q_0, q_2\}$	$\{q_1\}$
q_1	$\{\phi\}$	$\{q_2\}$
$*q_2$	$\{\phi\}$	$\{\phi\}$

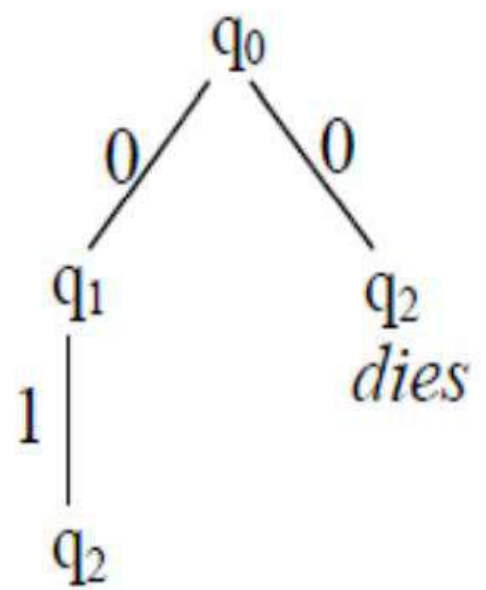
Non-Deterministic Finite Automata (NFA)

Computation tree for 01;



Non-Deterministic Finite Automata (NFA)

Computation tree for 0110



dies, so 0110 is not accepted

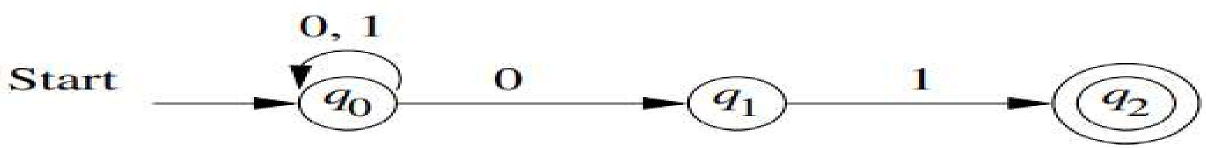
The Extended transition function of NFA

Definition by Induction:

- **Basis Step:**
  - $\hat{\delta}(q, \varepsilon) = \{q\}$ i.e. reading no input symbol remains into the same state.
- **Induction:**
  - Let w be a string from Σ^* such that w = xa, where x is a substring of without last symbol a.
  - Also let, $\hat{\delta}(q, x) = \{p_1, p_2, p_3, \ldots p_k\}$
  - and $$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$
  - Then, $\hat{\delta}(q, w) = \{r_1, r_2, r_3, \ldots r_m\}$
- Thus, to compute $\hat{\delta}(q, w)$ we first compute $\hat{\delta}(q, x)$ & then following any transition from each of these states with input a.

The Extended transition function of NFA

- Use of $\hat{\delta}$ for the processing of input 00101 by NFA



1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

Assignment-4

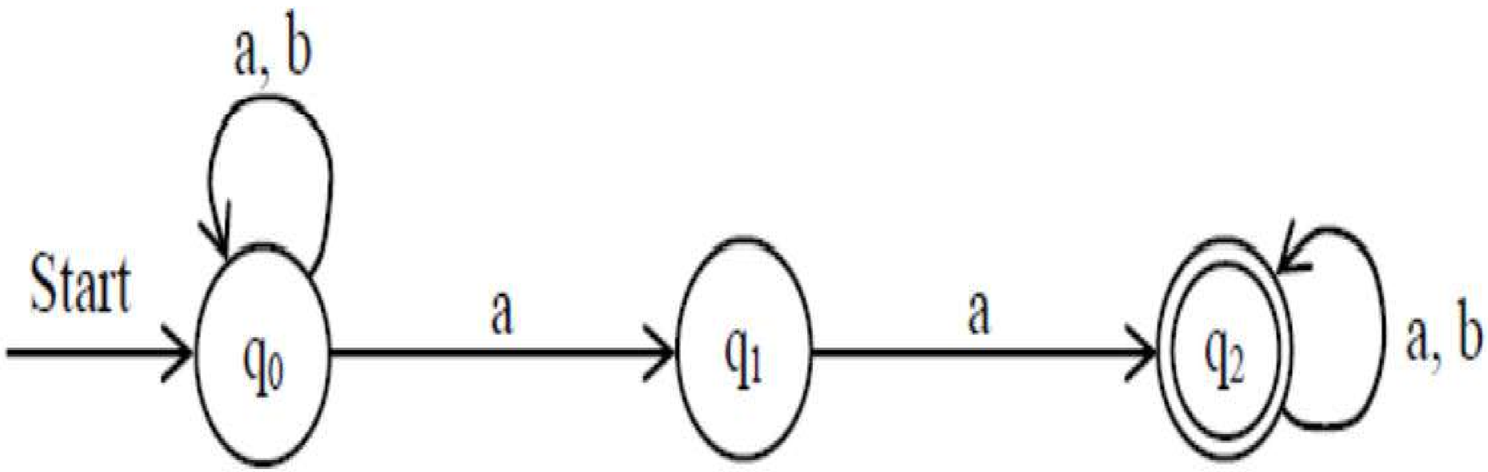
- compute for $\hat{\delta}(q_0, 01101)$
- Compute for $\hat{\delta}(q_0, 1101011)$
- Compute for $\hat{\delta}(q_0, 1010101)$

Language of NFA

- The language of NFA, $M = (Q, \Sigma, \delta, q0, F)$, denoted by L (M) is;
  - $L(M) = \{w / \hat{\delta}(q, w) \cap F \neq \varphi\}$
  - i.e. L(M) is the set of strings w in Σ^* such that $\hat{\delta}(q0, w)$ contains at least one state accepting state.

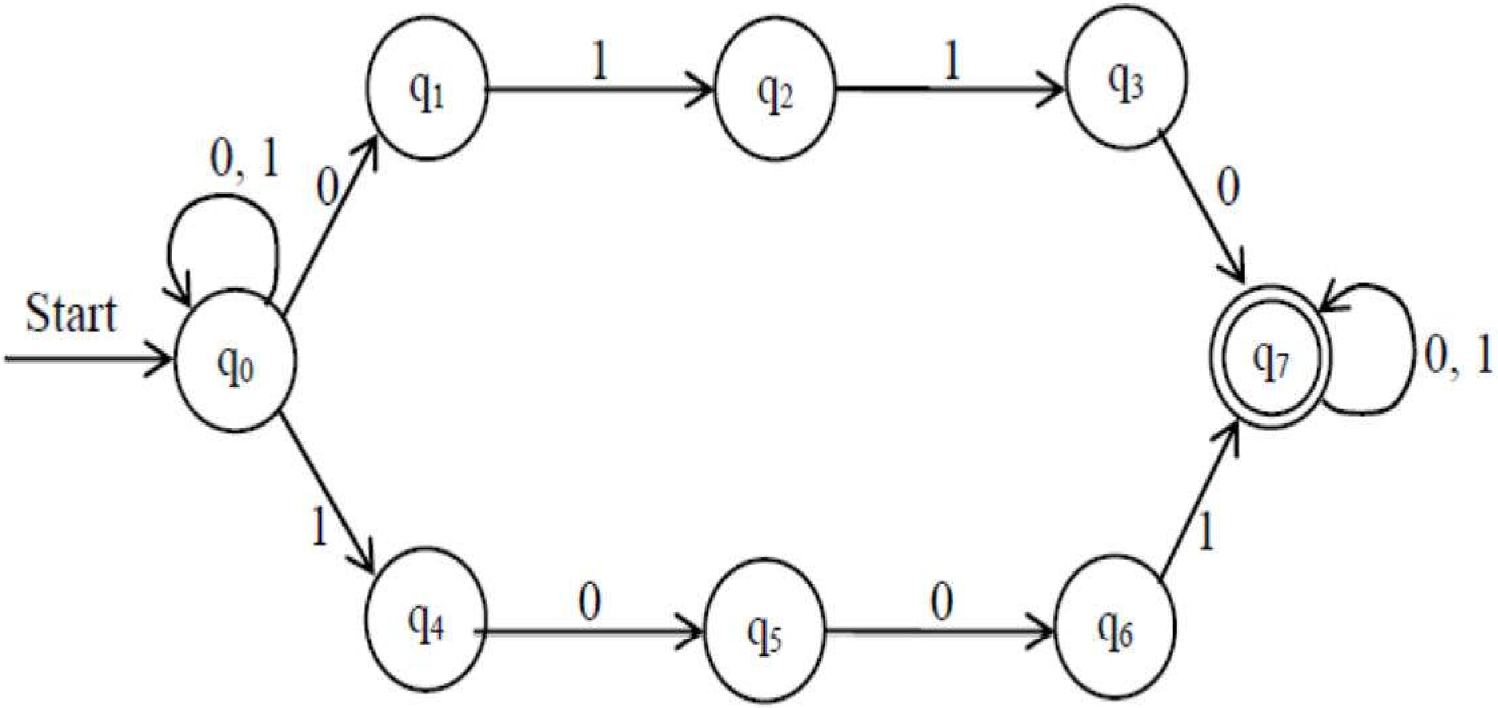
Examples

- Construct a NFA over {a, b} that accepts strings having aa as substring.



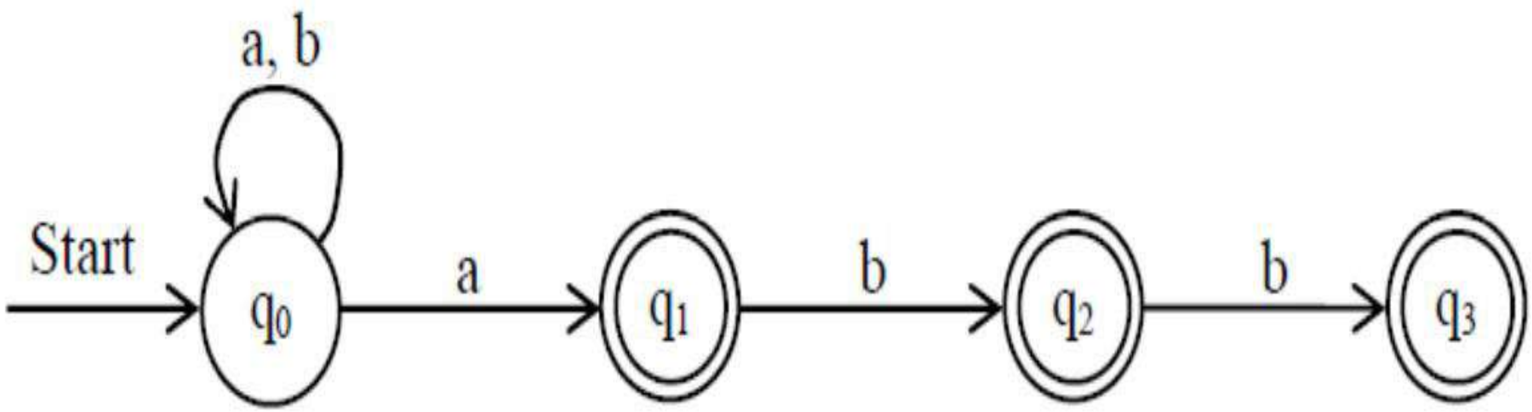
Examples

- NFA for strings over {0, 1} that contain substring 0110 or 1001



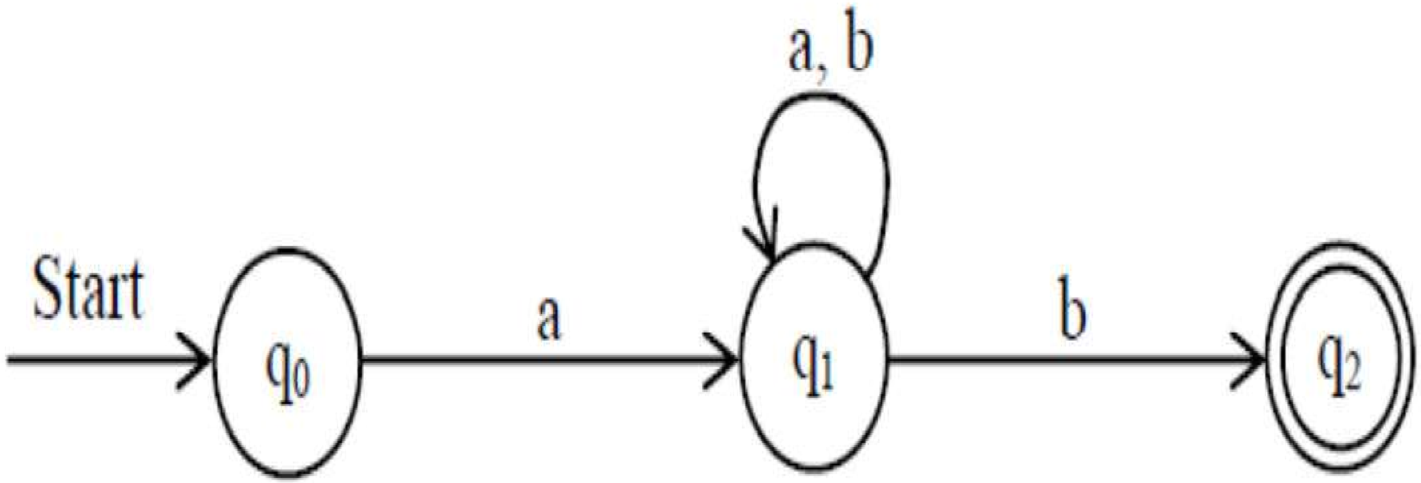
Examples

- NFA over {a, b} that have "a" as one of the last 3 characters.



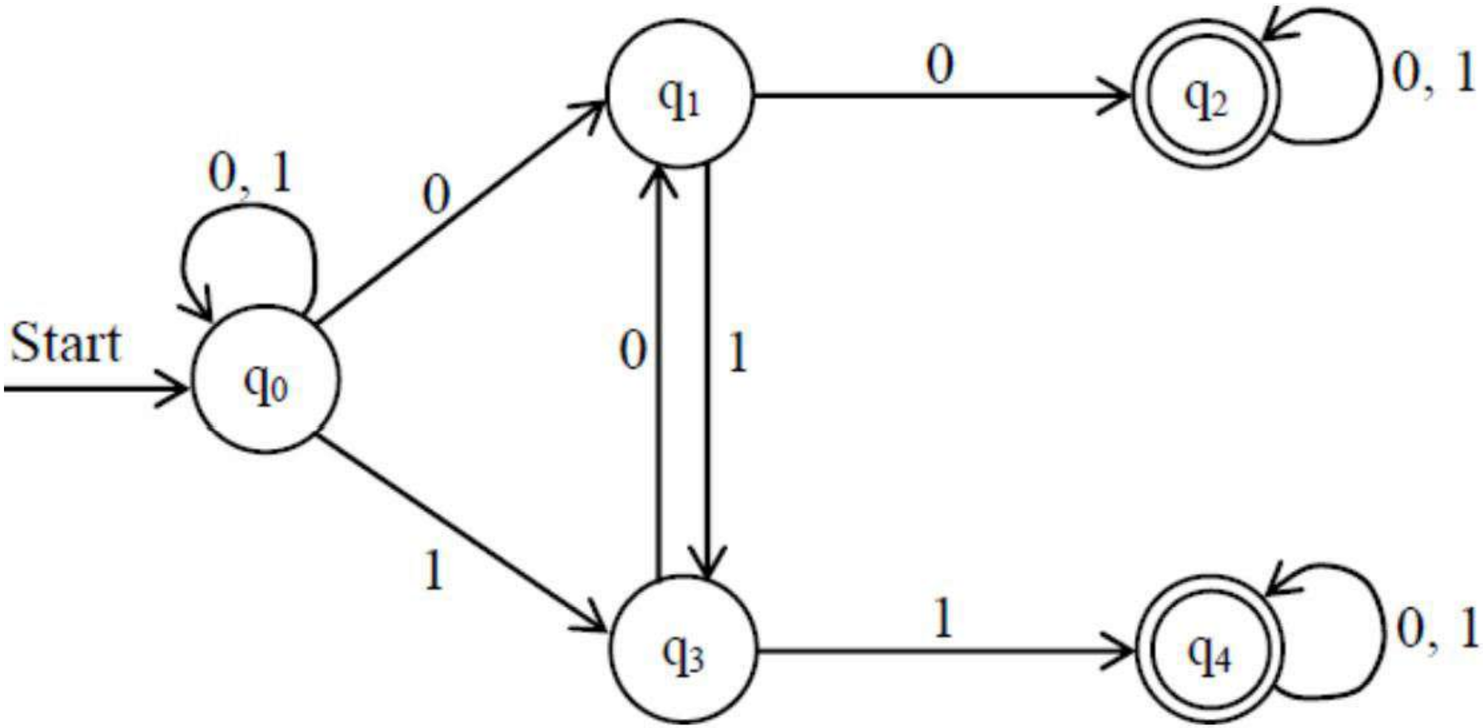
Examples

- NFA over {a, b} that accepts strings starting with a and ending with b.



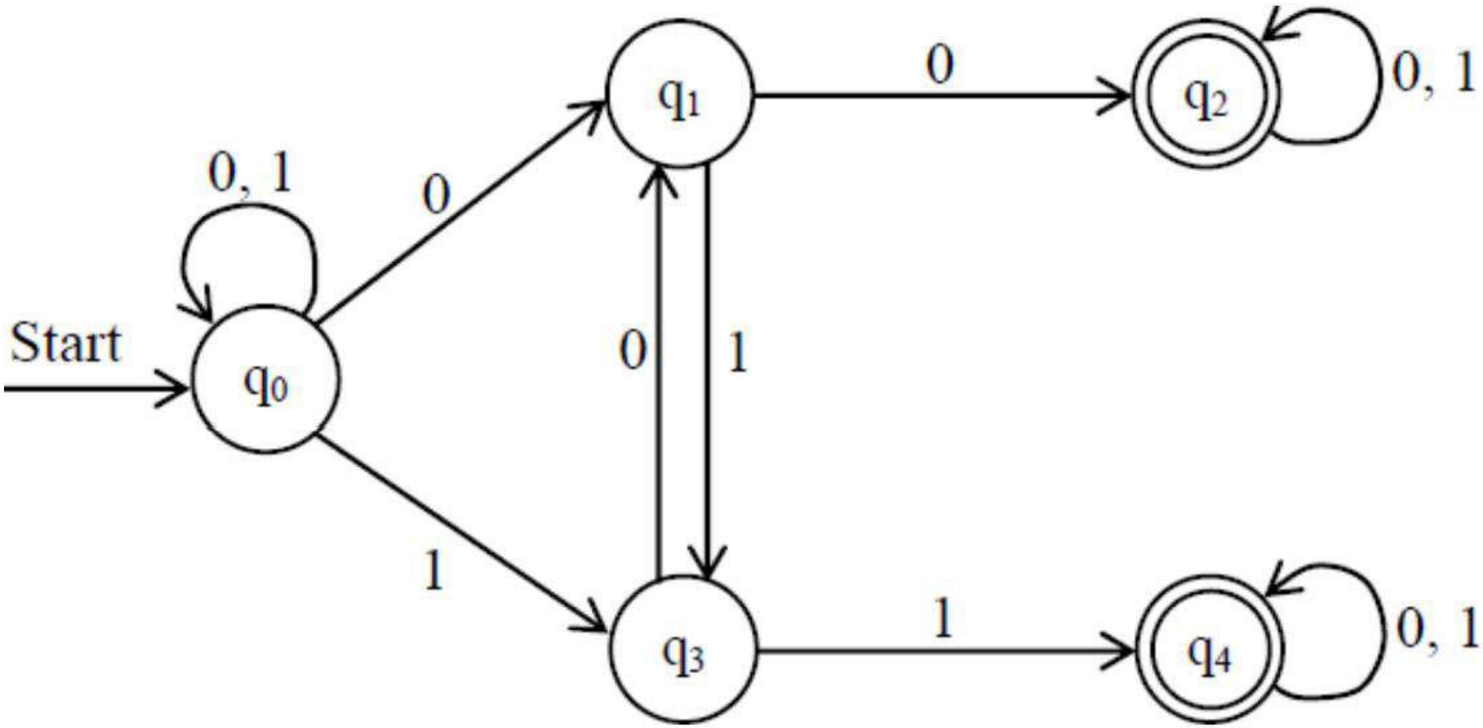
Examples

- Design a NFA for the language over {0, 1} that have at least two consecutive 0"s or1"s



Examples

- Design a NFA for the language over {0, 1} that have at least two consecutive 0‘‘s or1‘‘s



Assignment-5

- Give a NFA to accept the language of string over{a.b} in which each string contain abb as substring.
- Give a NFA which accepts binary strings which have at least one pair of „00“ or one pair of „11“

Equivalence of DFA and NFA

- every language that can be described by some NFA can also be described by some DFA
- The proof that DFA's can do whatever NFA's can do involves an important construction called **subset construction** because it involves constructing all subsets of the set of states of the NFA.
- If NFA has n-states, the DFA can have 2n states (at most), although it usually has many less.

Subset Construction Algorithm

- To convert a NFA, N = (Q_N, Σ, δ_N, q0, F_N) into an equivalent DFA D = (Q_D, Σ, δ_D, q_0, F_D), we have following steps.
- The start state of D is the set of start states of N i.e. if q_0 is start state of N then D has start state as $\{q_0\}$.
- Q_D is set of subsets of Q_N i.e. $Q_D = 2^{QN}$. So, Q_D is power set of Q_N. So if Q_N has n states then Q_D will have 2_n states. However, all of these states may not be accessible from start state of Q_D so they can be eliminated. So Q_D will have less than 2_n states.

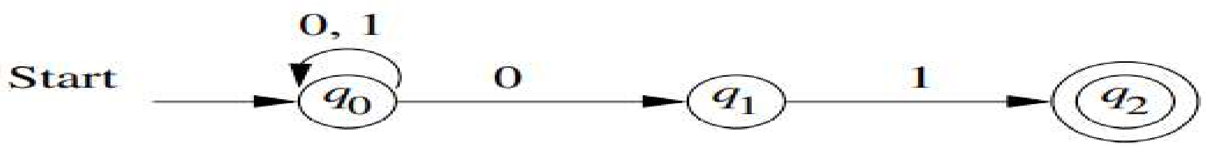
Subset Construction Algorithm (Contd...)

- F_D is set of subsets S of Q_N such that $S \cap F_N \neq \varphi$ i.e. F_D is all sets of N's states that include at least one final state of N.
- For each set $S \subseteq Q_N$ & each input $a \in \Sigma$,

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$$

- i.e. to compute δ_D (S, a) we look all the states p in S, see what states N goes to from p on input a, and take the union of all those states.

Subset Construction Algorithm (Contd…)



	δ:	0	1
A	ϕ	ϕ	ϕ
B	$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
C	$\{q_1\}$	ϕ	$\{q_2\}$
D	$*\{q_2\}$	ϕ	ϕ
E	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
F	$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
G	$*\{q_1, q_2\}$	ϕ	$\{q_2\}$
H	$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Table: Complete Subset construction of above NFA

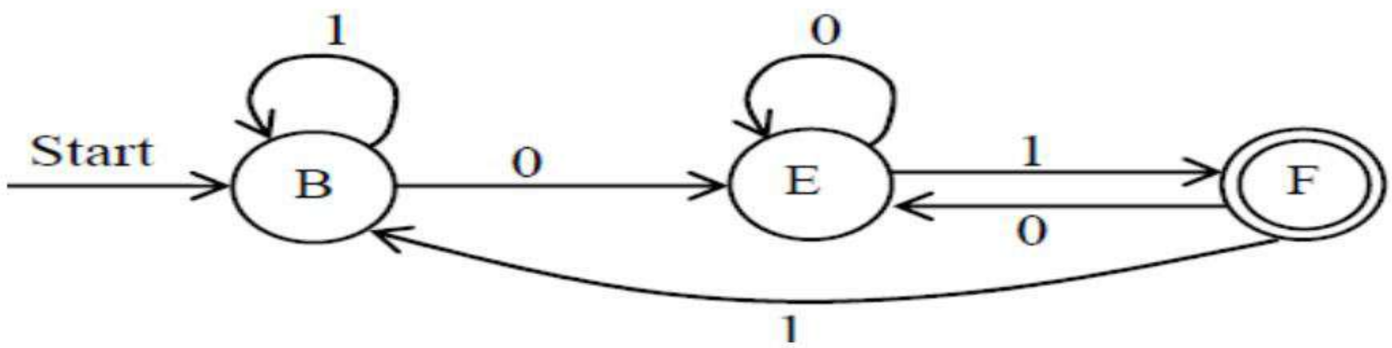
Reduction of NFA to DFA

- The same table(Previous slide) can be represented with renaming the state on table entry as

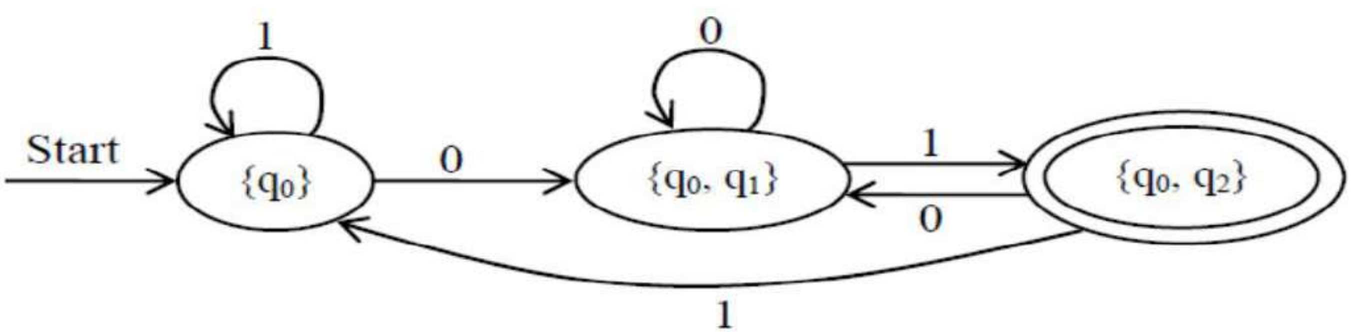
δ:	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

Reduction of NFA to DFA

- The equivalent DFA is

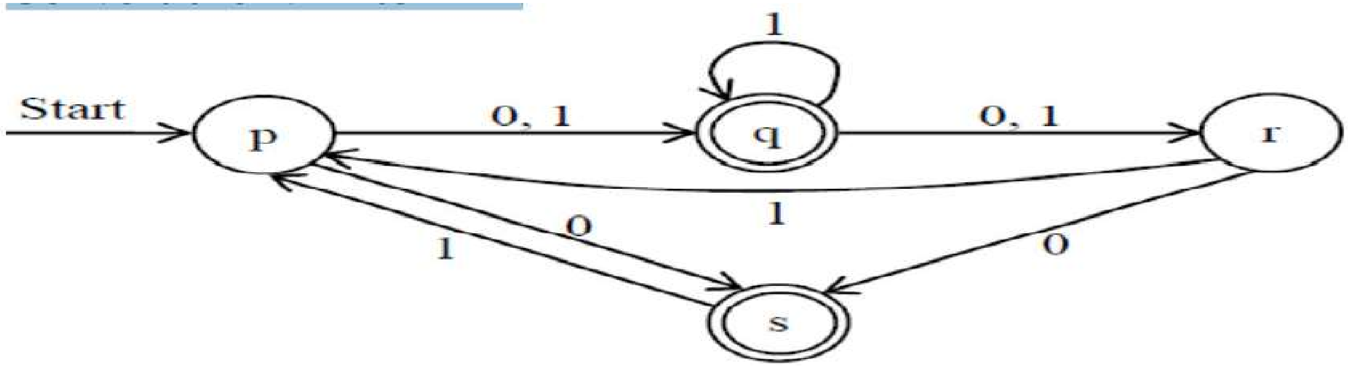


OR

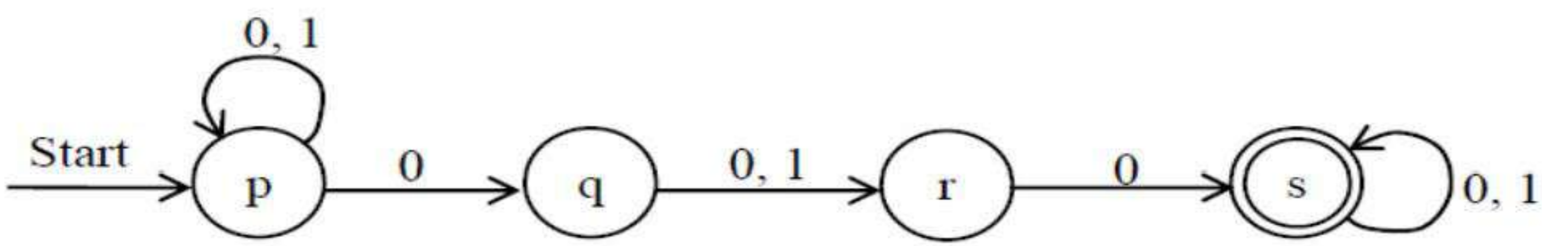


Assignment-6

- Convert the NFA to DFA

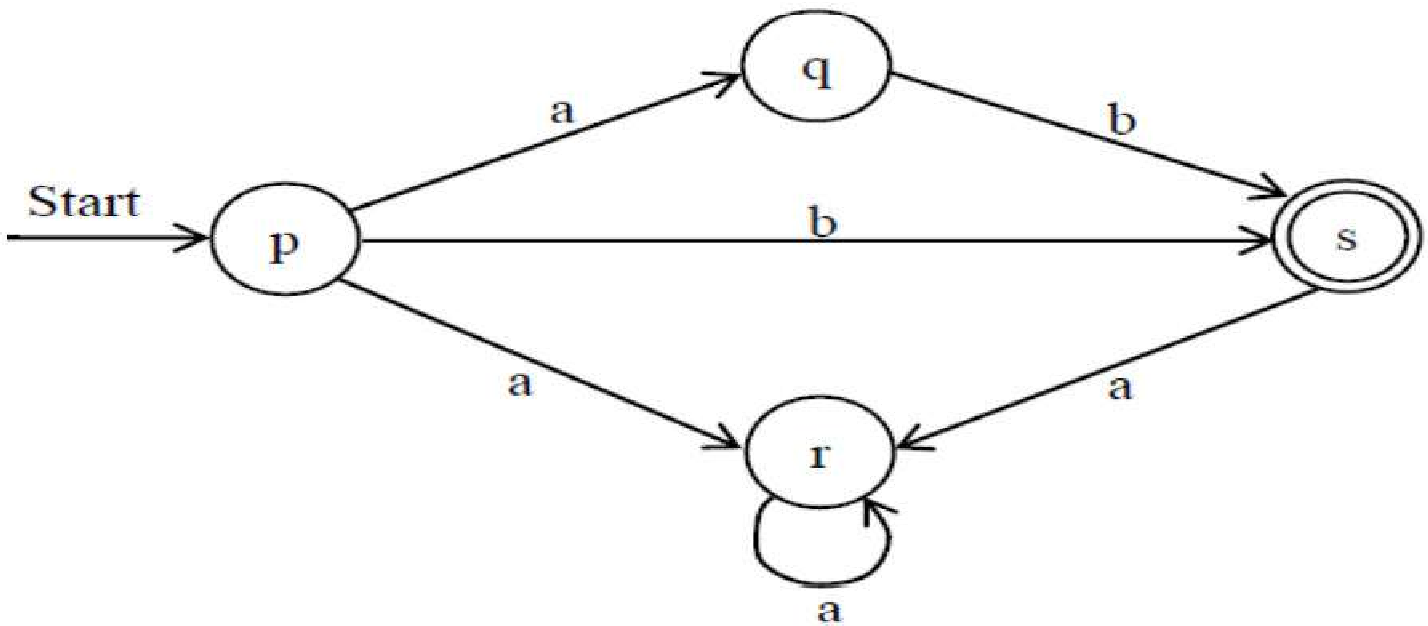


- Convert the NFA to DFA



Assignment-6

- Convert the NFA to DFA



Theorem 1

For any NFA, N = (Q_N, Σ, δ_N, q_0, F_N) accepting language L ⊆ Σ* there is a DFA D = (Q_D, Σ, δ_D, q_0',F_D) that also accepts L i.e. L (N) = L (D).

Proof:

- The fact that D accepts the same language as N is as; for any string w ∈ Σ*;

$$\hat{\delta}_N(q0, w) = \hat{\delta}_D(q0, w)$$

- Thus, we prove this fact by induction on length of w.

Theorem 1

- **Basis Step:**
  - Let |w| = 0, then w = ε,

    $\hat{\delta}_N(q0, \varepsilon) = \{q0) = q0 = \hat{\delta}_D(q0, \varepsilon)$
- **Induction step:**
  - Let |w| = n + 1 is a string such that w = xa & |x| = n, |a| = 1; a being last symbol.
  - Let the inductive hypothesis is that x satisfies.
  - Thus,
    - $\hat{\delta}_D(q0, x) = \hat{\delta}_N(q0, x)$, let these states be $\{p_1, p_2, p_3, \dots p_k\}$

Theorem 1

Now,

$\hat{\delta}_N$ (q0, w) $= \hat{\delta}_N$ (q0, xa)

$= \delta_N (\hat{\delta}_N$ (q0, x), a)

$= \delta_N$ ({p1, p2, p3, … pk}, a) [Since, from inductive step]

$= \cup\, \delta_N(p_i, a)$……………………….(1)

Also

$\hat{\delta}_D$ (q0', w) $= \hat{\delta}_D$ (q0', xa)

$= \delta_D (\hat{\delta}_D$ (q0', x), a)

$= \delta_D (\hat{\delta}_N$ (q0, x), a) [Since, by the inductive step as it is true for x]

$= \delta_D$ ({p1, p2, p3, … pk}, a) [Since, from inductive step]

Theorem 1

- Now, from subset construction, we can write,
  - $\delta_D (\{p_1, p_2, p_3, \ldots p_k\}, a) = \cup \delta_N(p_i ,a)$
- so, we have
  - $\hat{\delta}_D (q_0, w) = \cup \delta_N(p_i ,a)$…………(2)
- Now we conclude from 1 and 2 that

  $\hat{\delta}_N (q0, w) = \hat{\delta}_D (q0', w)$
- Hence, if this relation is true for $|x| = n$, then it is also true for $|w| = n + 1$.

  ∴DFA D & NFA N accepts the same language.

  i.e. L (D) = L (N) Proved.

Theorem 2

A language L is accepted by some DFA if and only if L is accepted by some NFA.

<u>Proof:</u>

- **“if” part** (A language is accepted by some DFA if L is accepted by some NFA):
  - It is the subset construction and is proved in previous theorem.
- **Only if part** (a language is accepted by some NFA if L is accepted by some DFA):
  - Here we have to convert the DFA into an identical NFA.
  - To show if L is accepted by D then it is also accepted by N
    - it is sufficient to show, for any string w $\in \Sigma^*$,
      - $\hat{\delta}_D(q_0, w) = \hat{\delta}_N(q_0, w)$

Theorem 2

- We can proof this fact using induction on length of the string
- **Basis step:**
  - Let $|w| = 0$ i.e. $w = \varepsilon$
  - $\therefore \hat{\delta}_D(q_0, w) = \hat{\delta}_D(q_0, \varepsilon) = q_0$
  - $\hat{\delta}_N(q_0, w) = \hat{\delta}_N(q_0, \varepsilon) = \{q_0\}$
  - $\therefore \hat{\delta}_D(q_0, w) = \hat{\delta}_N(q_0, w)$ for $|w| = 0$ is true.
- **Induction:**
  - Let $|w| = n + 1$ & $w = xa$. Where $|x| = n$ & $|a| = 1$; a being the last symbol.
  - Let the inductive hypothesis is that it is true for x.
  - $\therefore$if $\hat{\delta}_D(q_0, x) = p$, then $\hat{\delta}_N(q_0, x) = \{p\}$
  - i.e. $\hat{\delta}_D(q_0, x) = \hat{\delta}_N(q_0, x)$

Theorem 2

Now,

$\hat{\delta}_D$ (q0, w) $= \hat{\delta}_D$ (q0, xa)

$= \delta_D (\hat{\delta}_D$ (q0, x), a)

$= \delta_D$ (p, a) [from inductive step $\hat{\delta}_D$ (q0, x,=p]

= r, say

Now,

$\hat{\delta}_N$ (q0, w) $= \hat{\delta}_N$ (q0, xa)

$= \delta_N (\hat{\delta}_N$ (q0, x), a) [from inductive steps]

$= \delta_N(\{p\}, a)$

=r [from the rule that define δ_N)

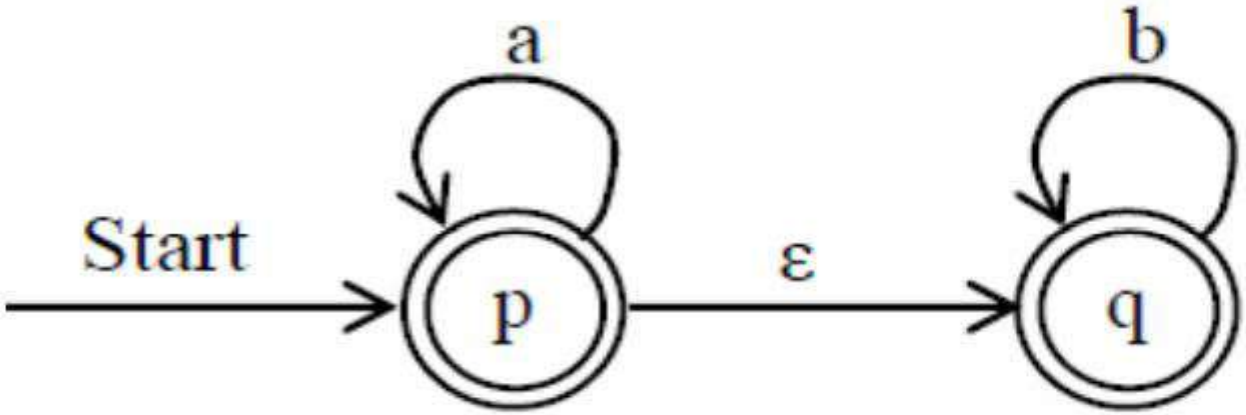
Hence proved. i.e. $\hat{\delta}_D$ (q0, w)= $\hat{\delta}_N$ (q0, w)

Finite Automaton with Epsilon Transition (ε - NFA)

- A NFA with ε-transition is defined by five tuples $(Q, \Sigma, \delta, q_0, F)$, where;
  - Q = set of finite states
  - Σ = set of finite input symbols
  - q_0 = Initial state, $q_0 \in Q$
  - F = set of final states; $F \subseteq Q$
  - δ = a transition function that maps; $Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$

Finite Automaton with Epsilon Transition (ε - NFA)

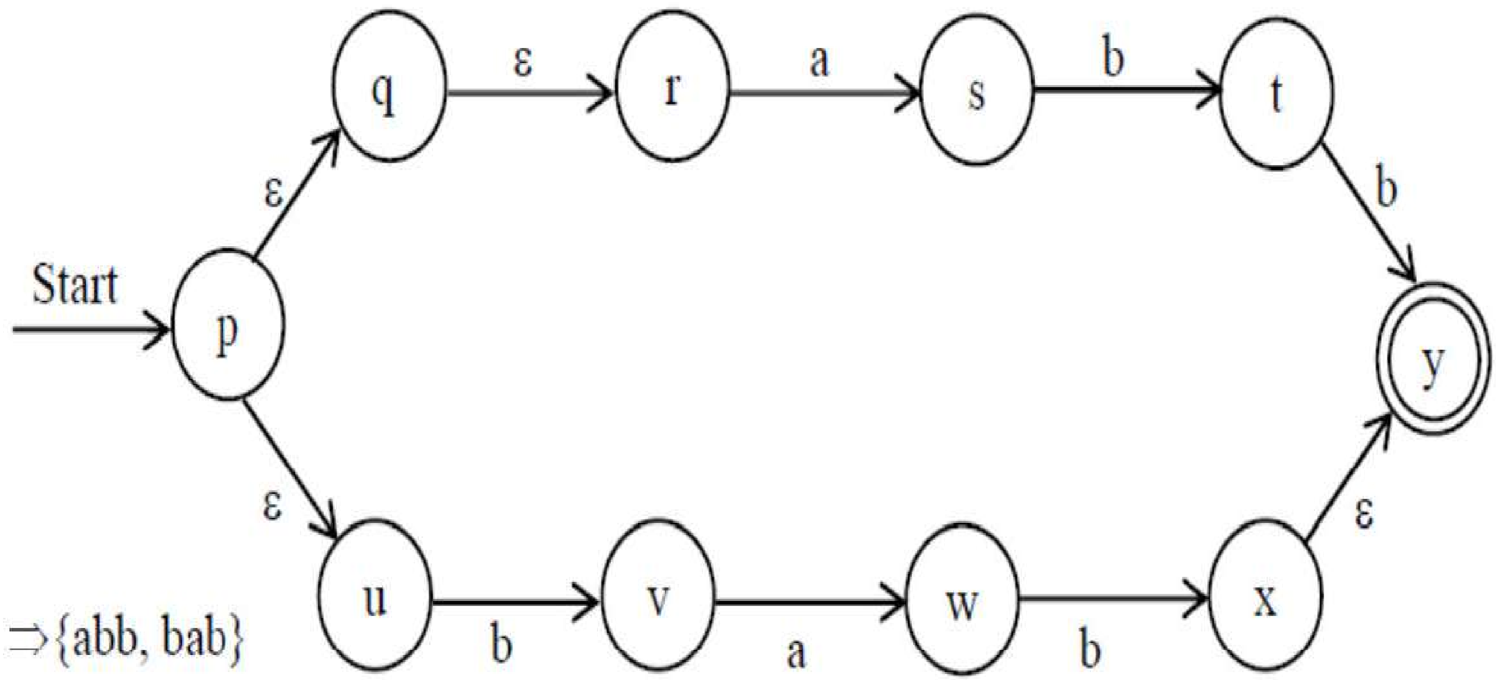
- **Examples:**



- This accepted the language
  {a,aa,ab,abb,b,bbb,……………………}

Finite Automaton with Epsilon Transition (ε - NFA)

- **Examples**



Exercise

- Design ε-NFA for the following languages
  - The set of strings consisting of zero or more a's followed by zero or more b's , followed by zero or more c's.
  - The set of strings that consists of either 01 repeated one or more times or 010 repeated one or more times.

Epsilon Closure of a State

- ε-closure of a state 'q' can be obtained by following all transitions out of q that are labeled ε.
- Formally, we can define ε-closure of the state q as;
- **Basis**: state q is in ε-closure (q).
- **Induction**: If state q is reached with ε-transition from state p then, p is in ε-closure (q). And if there is an arc from p to r labeled ε, then r is in ε-closure (q) and so on.
- If δ is the transition function of ε-NFA involved , and p is in ε-closure (q), then ε-closure (q) also contains all the states in δ(p, ε).

Epsilon Closure of a State

- Example:

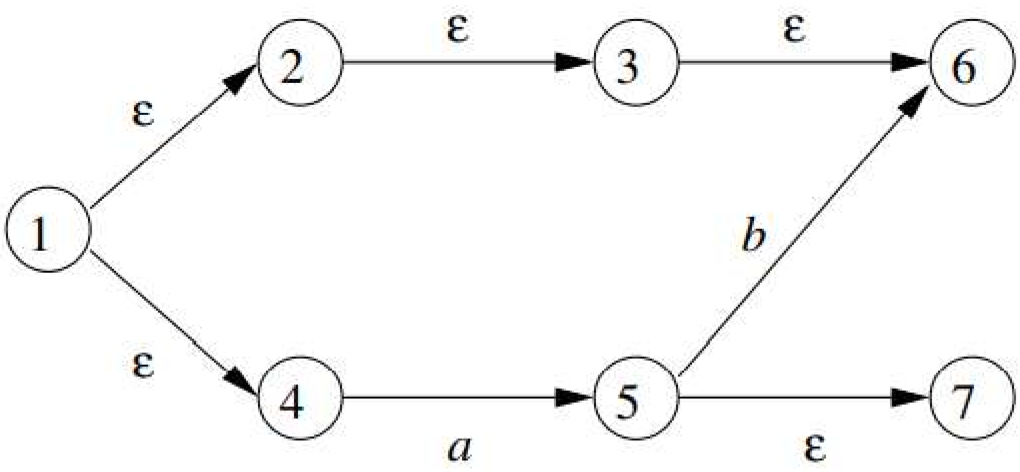


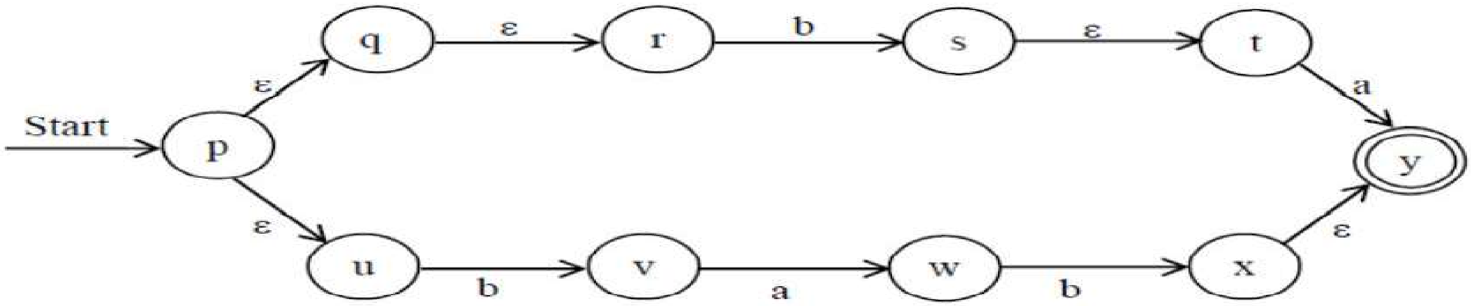
Fig: Some states and transitions

- ε-closure(1) = {1,2,3,4,6}

Extended Transition Function of ε–NFA

- **BASIS STEP**
  - $\hat{\delta}$ (q, ε) = ε-closure (q)
- **INDUCTION STEP**
  - Let w = xa be a string, where x is substring of w without last symbol a and a ∈ Σ but a ≠ ε.
  - Let $\hat{\delta}$ (q, x) = $\{p_1, p_2, \ldots p_k\}$ i.e. pi's are the states that can be reached from q following path labeled x which can end with many ε & can have many ε.
  - Also Let, $\bigcup_{i=1}^{k} \delta(p_i, a)$ be the set $\{r_1,r_2,r_3\ldots\ldots,r_m\}$
  - Then, $\hat{\delta}$ (q,w) = ε-closure($\{r_1,r_2,r_3\ldots\ldots,r_m\}$)

Extended Transition Function of ε–NFA



compute for string ba

- $\hat{\delta}$ (p, ε)= ε-closure(p)={p,q,r,u}
- Compute for b
  - i.e. δ(p,b)U δ(q,b)U δ(r,b)U δ(u,b)={s,v}
  - ε-colsure(s)U ε-closure(v)={s,t,v}
- Computer for next input 'a'
  - δ(s,a)U δ(t,a)U δ(v,a)={y,w}
  - ε-closure(y)U ε-closure(w)={y,w}
- The final result set contains the one of the final state so the string is accepted.

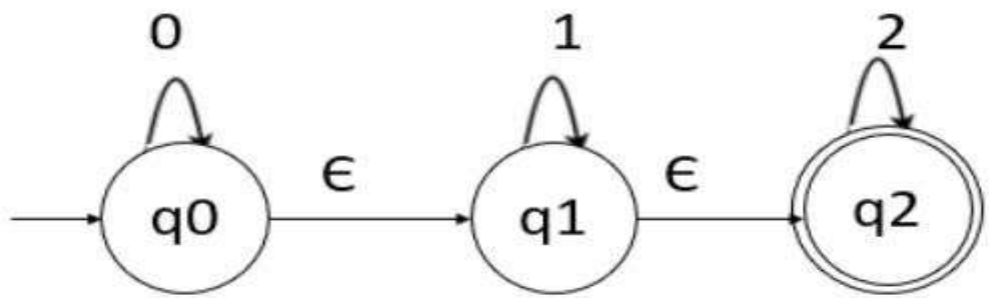
Removing Epsilon Transition using the concept of Epsilon Closure

- **Step 1** – Find out all the ε-transitions from each state from Q. That will be called as ε-closure(qi) where, qi ∈Q.
- **Step 2** – Then, $\delta 1$ transitions can be obtained. The $\delta 1$ transitions means an ε-closure on δ moves.
- **Step 3** – Step 2 is repeated for each input symbol and for each state of given NFA.
- **Step 4** – By using the resultant status, the transition table for equivalent NFA without ε can be built.
- NFA with ε to without ε is as follows –
  - $\delta 1(q,a) = \in\text{-closure } (\delta\ (\delta\hat{}(q,\epsilon),a))$

    where, $\delta\hat{}(q,\epsilon) = \in\text{-closure}(q)$

Removing Epsilon Transition using the concept of Epsilon Closure

- Convert the given NFA with epsilon to NFA without epsilon.



- We will first obtain ε-closure of each state
  - ε-closure(q0) = {q0,q1,q2}
  - ε-closure(q1) = {q1,q2}
  - ε-closure(q2) = {q2}

Removing Epsilon Transition using the concept of Epsilon Closure

- Now we will obtain $\delta 1$ transitions for each state on each input symbol
  - $\delta 1(q0, 0) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q0, \varepsilon),0))$
    - $= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q0),0))$
    - $= \varepsilon\text{-closure}(\delta(q0,q1,q2), 0))$
    - $= \varepsilon\text{-closure}(\delta(q0, 0) \cup \delta(q1, 0) \cup \delta(q2, 0))$
    - $= \varepsilon\text{-closure}(q0 \cup \Phi \cup \Phi)$
    - $= \varepsilon\text{-closure}(q0) = \{q0,q1, q2\}$
  - $\delta'(q0, 1) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q0, \varepsilon),1))$
    - $= \varepsilon\text{-closure}(\delta(q0,q1,q2), 1))$
    - $= \varepsilon\text{-closure}(\delta(q0, 1) \cup \delta(q1, 1) \cup \delta(q2, 1))$
    - $= \varepsilon\text{-closure}(\Phi \cup q1 \cup \Phi)$
    - $= \varepsilon\text{-closure}(q1)$
    - $= \{q1, q2\}$

Removing Epsilon Transition using the concept of Epsilon Closure

- $\delta 1(q0, 2) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q0, \varepsilon),2))$
  - $= \varepsilon\text{-closure}(\delta(q0,q1,q2), 2))$
  - $= \varepsilon\text{-closure}(\delta(q0, 2) \cup \delta(q1, 2) \cup \delta(q2, 2))$
  - $= \varepsilon\text{-closure}(\Phi \cup \Phi \cup q2)$
  - $= \varepsilon\text{-closure}(q2)$
  - $= \{q2\}$
- $\delta 1(q1, 0) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q1, \varepsilon),0))$
  - $= \varepsilon\text{-closure}(\delta(q1,q2), 0))$
  - $= \varepsilon\text{-closure}(\delta(q1, 0) \cup \delta(q2, 0))$
  - $= \varepsilon\text{-closure}(\Phi \cup \Phi)$
  - $= \varepsilon\text{-closure}(\Phi)$
  - $= \Phi$

Removing Epsilon Transition using the concept of Epsilon Closure

- $\delta 1(q1,1) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q1, \varepsilon),1))$
  - $= \varepsilon\text{-closure}(\delta(q1,q2), 1))$
  - $= \varepsilon\text{-closure}(\delta(q1, 1) \cup \delta(q2, 1)\ )$
  - $= \varepsilon\text{-closure}(q1 \cup \Phi)$
  - $= \varepsilon\text{-closure}(q1)$
  - $= \{q1,q2\}$
- $\delta 1(q1, 2) = \varepsilon\text{-closure}(\delta(\delta\hat{}(q1, \varepsilon),2))$
  - $= \varepsilon\text{-closure}(\delta(q1,q2), 2))$
  - $= \varepsilon\text{-closure}(\delta(q1, 2) \cup \delta(q2, 2)\ )$
  - $= \varepsilon\text{-closure}(\Phi \cup q2)$
  - $= \varepsilon\text{-closure}(q2)$
  - $= \{q2\}$

Removing Epsilon Transition using the concept of Epsilon Closure

- $\delta 1(q2, 0) = \varepsilon\text{-closure}(\delta(\delta\text{^}(q2, \varepsilon),0))$
  - $= \varepsilon\text{-closure}(\delta(q2), 0))$
  - $= \varepsilon\text{-closure}(\delta(q2, 0))$
  - $= \varepsilon\text{-closure}(\Phi)$
  - $= \Phi$
- $\delta 1(q2, 1) = \varepsilon\text{-closure}(\delta(\delta\text{^}(q2, \varepsilon),1))$
  - $= \varepsilon\text{-closure}(\delta(q2), 1)$
  - $= \varepsilon\text{-closure}(\delta(q2, 1))$
  - $= \varepsilon\text{-closure}(\Phi)$
  - $= \Phi$
- $\delta 1(q2, 2) = \varepsilon\text{-closure}(\delta(\delta\text{^}(q2, \varepsilon),))$
  - $= \varepsilon\text{-closure}(\delta(q2), 2))$
  - $= \varepsilon\text{-closure}(\delta(q2, 2))$
  - $= \varepsilon\text{-closure}(q2)$
  - $= \{q2\}$

Removing Epsilon Transition using the concept of Epsilon Closure

- summarize all the computed $\delta 1$ transitions as given below −
  - $\delta 1(q0,0)=\{q0,q1,q2\}$
  - $\delta 1(q0,1)=\{q1,q2\}$
  - $\delta 1(q0,2)=\{q2\}$

  - $\delta 1(q1,0)=\{\Phi\}$
  - $\delta 1(q1,1)=\{q1,q2\}$
  - $\delta 1(q1,2)=\{q2\}$

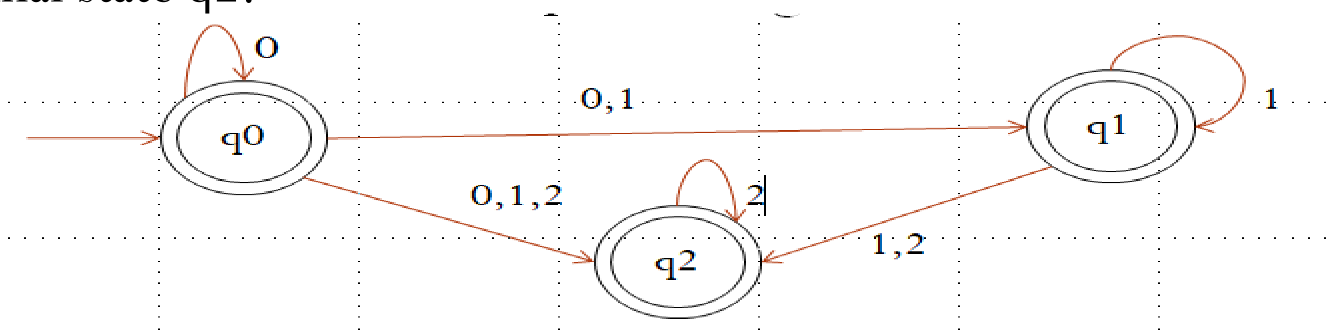
  - $\delta 1(q2,0)=\{\Phi\}$
  - $\delta 1(q2,1)=\{\Phi\}$
  - $\delta 1(q2,2)=\{q2\}$

Removing Epsilon Transition using the concept of Epsilon Closure

- The **transition table** is given below −

States/Inputs	0	1	2
q0	{q0,q1,q2}	{q1,q2}	{q2}
q1	Φ	{q1,q2}	{q2}
q2	Φ	Φ	{q2}

- The NFA without epsilon is given below − Here, q0, q1, q2 are final states because ε-closure(q0), ε-closure(q1) and ε-closure(q2) contain a final state q2.



Equivalence of NFA and ε –NFA

- every language that can be described by some NFA can also be described by some ε-NFA.

Equivalence of DFA and ε –NFA

- every language that can be described by some DFA can also be described by some ε-NFA.

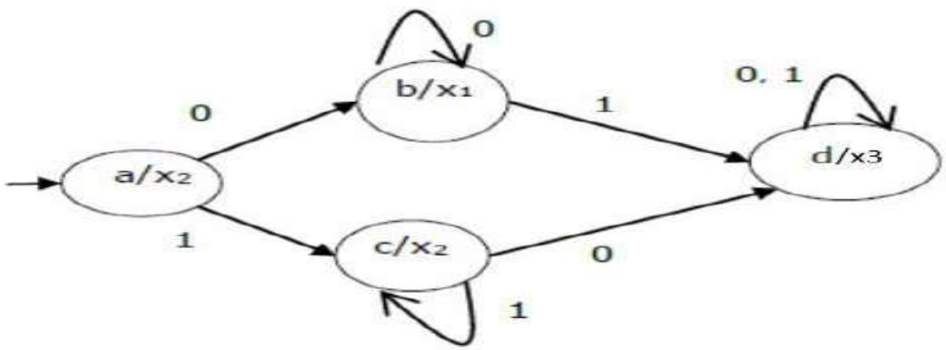
Finite State Machines with output

- Finite automata may have outputs corresponding to each transition.
- There are two types of finite state machines that generate output –
  - Moore machine
  - Mealy Machines

Moore Machine

- Moore machine is an FSM whose outputs depend on only the present state.
- A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where −
  - **Q** is a finite set of states.
  - **∑** is a finite set of symbols called the input alphabet.
  - **O** is a finite set of symbols called the output alphabet.
  - **δ** is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
  - **X** is the output transition function where $X: Q \rightarrow O$
  - $\mathbf{q_0}$ is the initial state from where any input is processed ($q_0 \in Q$).

Moore Machine

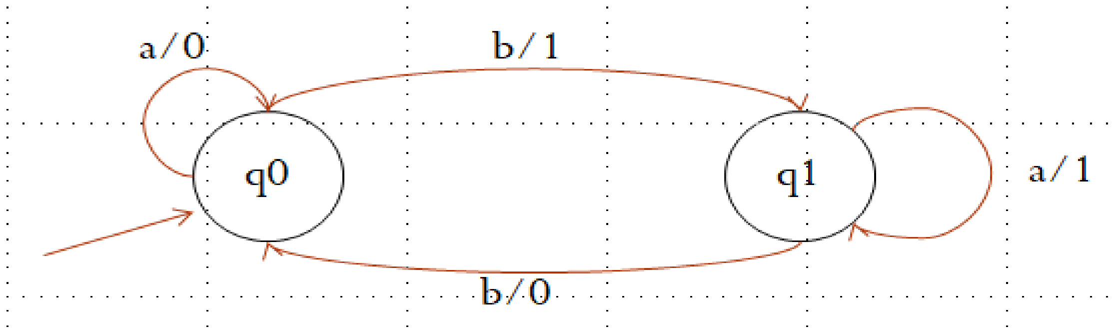


Present State	Next State		Output
	Input =0	Input = 1	
→a	b	c	x2
b	b	d	x1
c	d	c	x2
d	d	d	x3

Mealy Machine

- A Mealy Machine is an FSM whose output depends on the present state as well as the present input.
- It can be described by a 6 tuple $(Q, \Sigma, O, \delta, X, q_0)$ where −
  - **Q** is a finite set of states.
  - **Σ** is a finite set of symbols called the input alphabet.
  - **O** is a finite set of symbols called the output alphabet.
  - **δ** is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
  - **X** is the output transition function where $X: Q \times \Sigma \rightarrow O$
  - **q_0** is the initial state from where any input is processed ($q_0 \in Q$).

Mealy Machine



Present State	Next State			
	Input = a		Input = b	
	State	Output	State	Output
→ q0	q0	0	q1	1
q1	q1	1	q0	0