CSC-257
Theory Of Computation
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Transition table of DFA

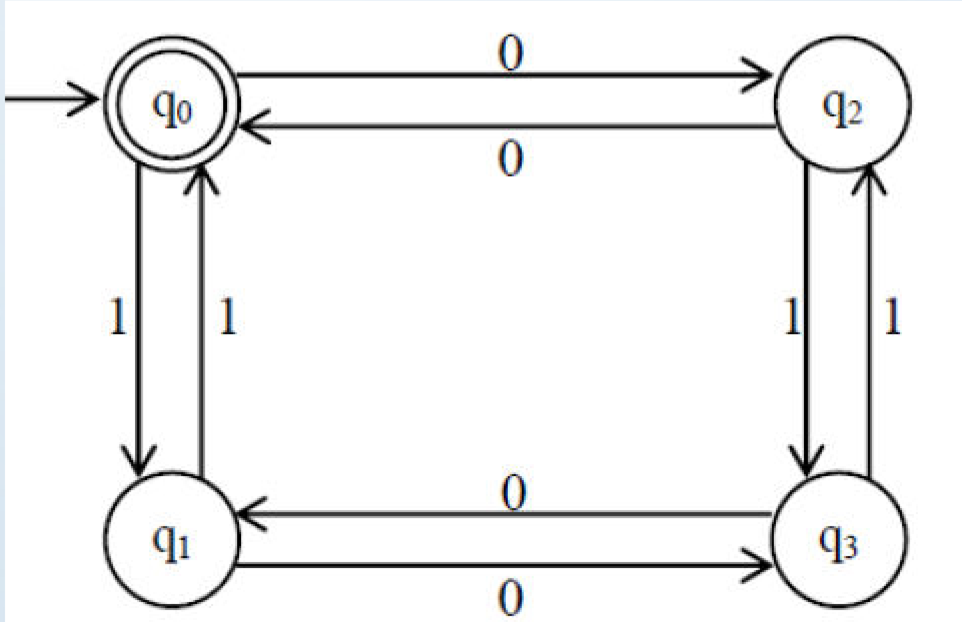
- Transition table is a conventional, tabular representation of the transition function δ that takes the arguments from Q × Σ & returns a value which is one of the states of the automation
- The row of the table corresponds to the states while column corresponds to the input symbol.
- The starting state 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.
- The entry for a row corresponding to state q and the column corresponding to input a, is the state δ (q, a)

δ	0	1
* -> q_0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

Transition table of DFA : Example

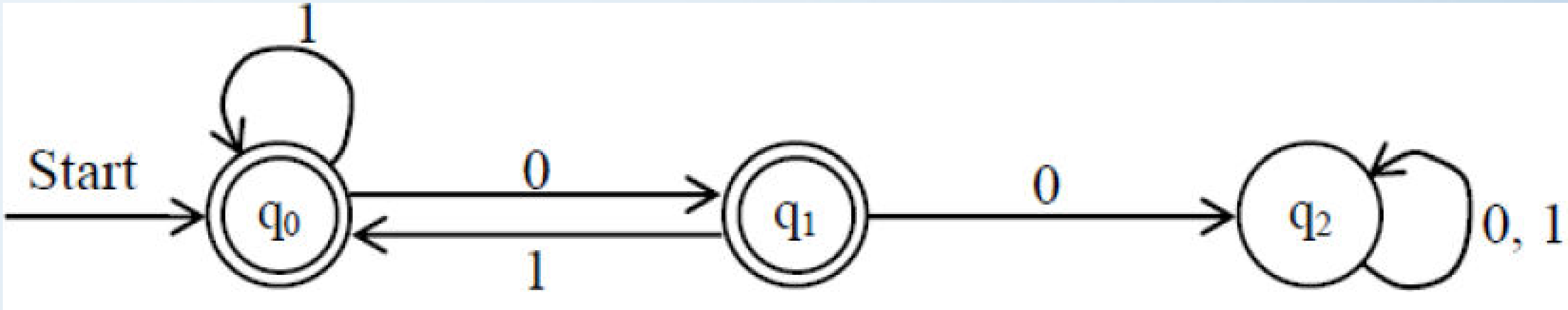
- Consider a DFA;
  - Q = {q0, q1, q2, q3}
  - Σ = {0, 1}
  - q0 = q0
  - F = {q0}
  - δ = Q × Σ -> Q
- Then the transition table transition diagrams for above DFA are as follows:
- This DFA accepts strings having both an even number of 0's & even number of 1's.

δ	0	1
* -> q_0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2



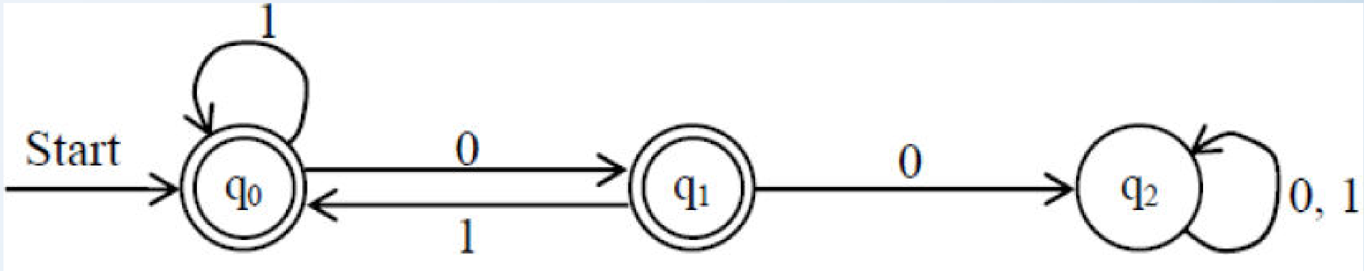
Extended Transition Function of DFA(δ*): -

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



Extended Transition Function of DFA(δ^*): -

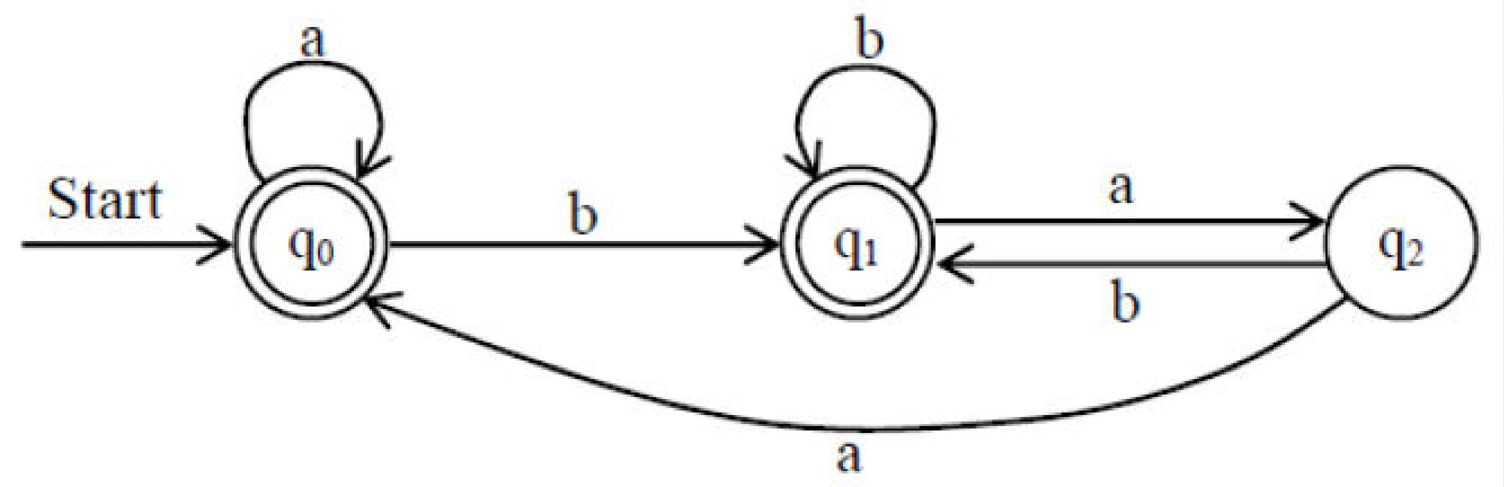
- 1) Compute $\delta^*(q_0, 1001)$
  - $= \delta(\delta^*(q0, 100), 1)$
  - $= \delta(\delta(\delta^*(q0, 10), 0), 1)$
  - $= \delta(\delta(\delta(\delta^*(q0, 1), 0), 0), 1)$
  - $= \delta(\delta(\delta(\delta(q0, 1), 0), 0), 1)$
  - $= \delta(\delta(\delta(q0, 0), 0), 1)$
  - $= \delta(\delta(q1, 0), 1)$
  - $= \delta(q2, 1)$
  - = q2, so string is rejected(or not accepted).



- 2) Compute (q0,101) yourself.(Ans : accepted by above DFA)

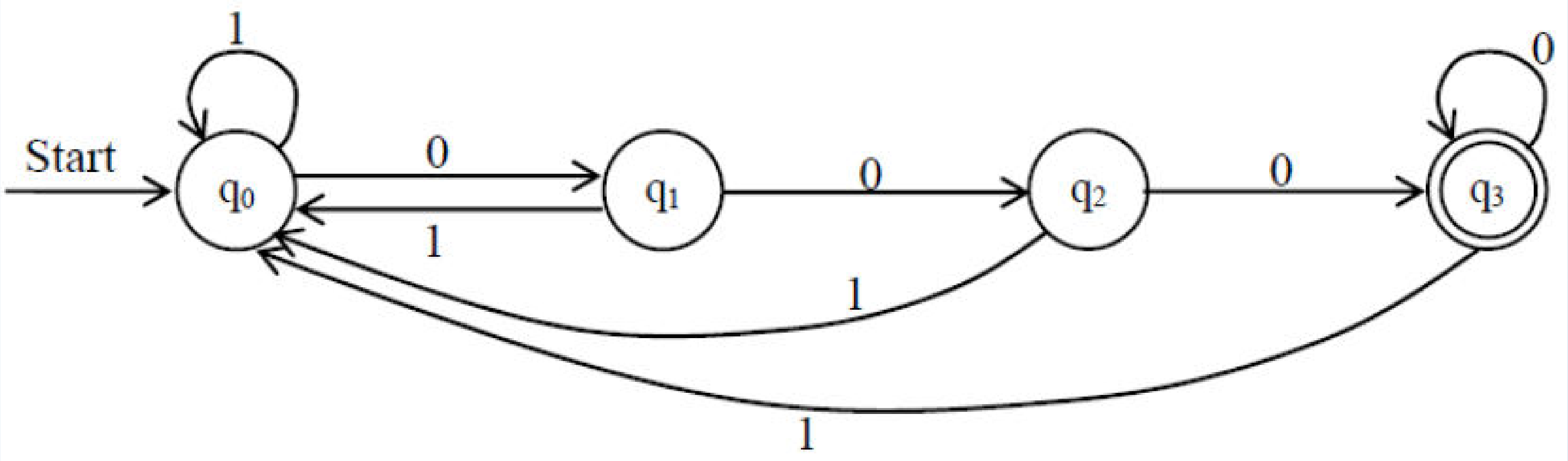
DFA Other Examples

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



DFA Other Examples

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



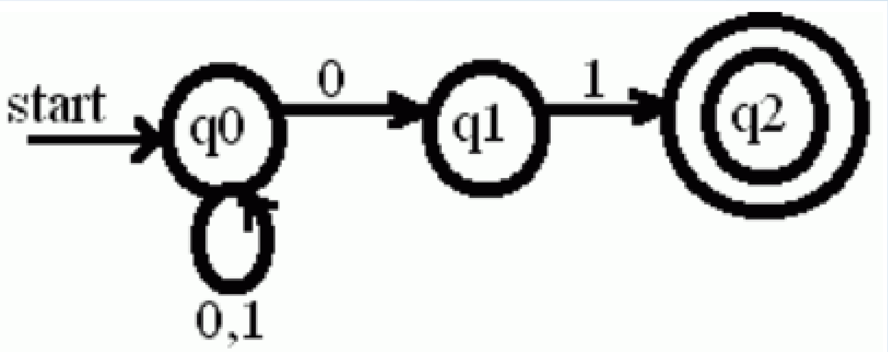
Nondeterministic finite automata(NFA)

- a nondeterministic finite automaton (NFA) or nondeterministic finite state machine is a finite state machine where from each state and a given input symbol, the automaton may jump into several possible next states.

- This distinguishes it from the deterministic finite automaton (DFA), where the next possible state is uniquely determined.

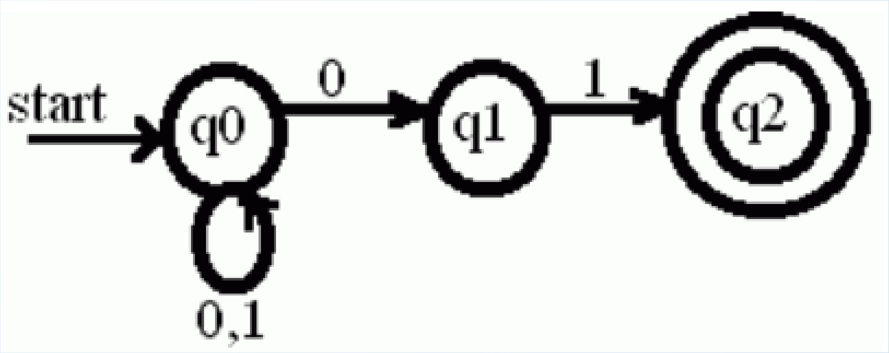
- Although the DFA and NFA have distinct definitions, a NFA can be translated to equivalent DFA using power set construction, i.e., the constructed DFA and the NFA recognize the same formal language.

- Both types of automata recognize only regular languages



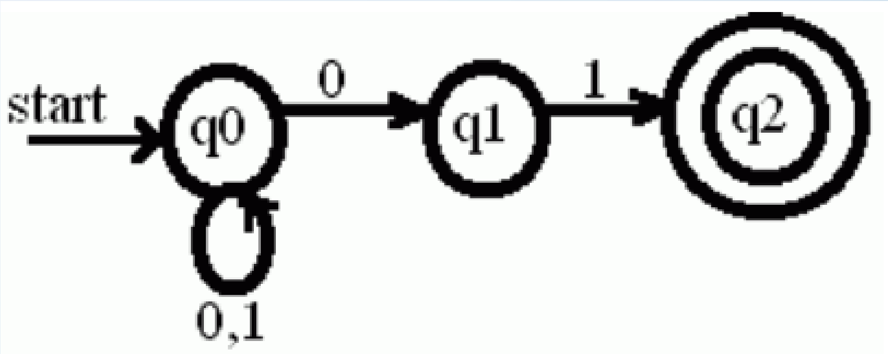
NFA : Format Definition

- An NFA is represented formally by a 5-tuple, $(Q, \Sigma, \Delta, q0, F)$, consisting of
  - a finite set of states Q
  - a finite set of input symbols Σ
  - a transition relation $\Delta : Q \times \Sigma \rightarrow P(Q)$
  - an initial (or start) state $q0 \in Q$
  - a set of states F distinguished as accepting (or final) states $F \subseteq Q$

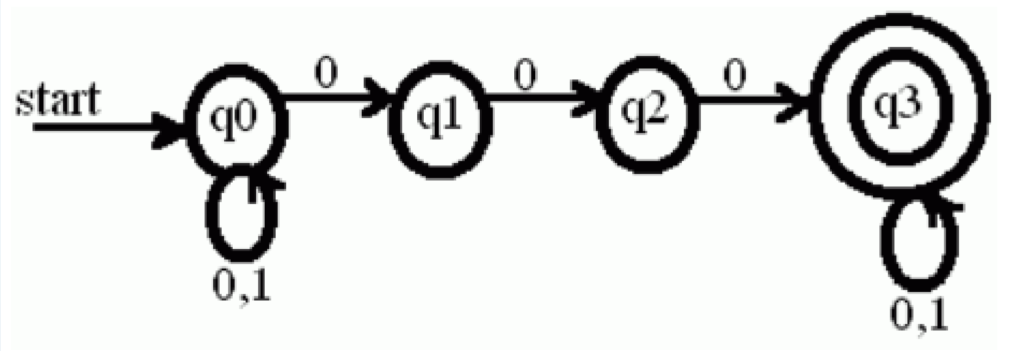


Examples

- 1.Construct an NFA to accept all strings terminating in 01

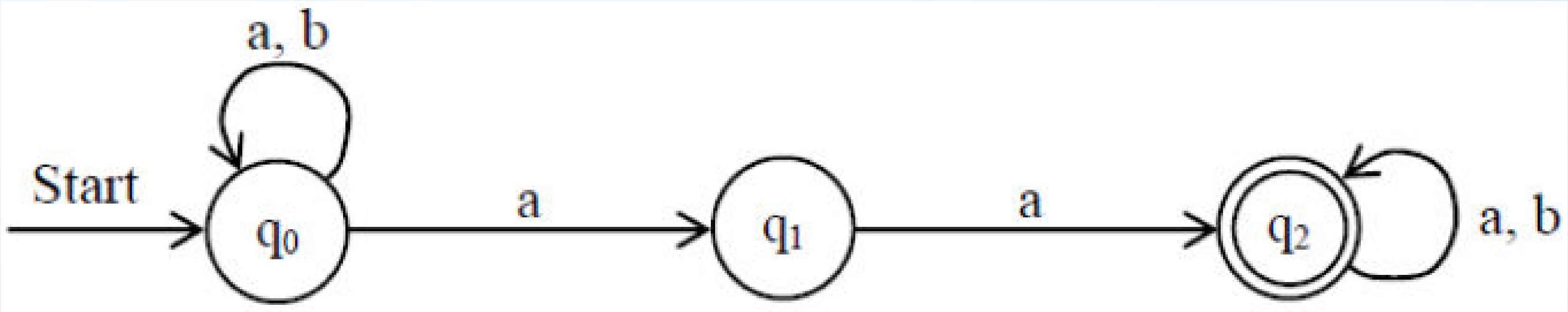


- 2.Construct an NFA to accept those strings containing three consecutive zeroes



Examples

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



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