16CS207: Formal Languages and Automata Theory

(FLAT)

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

P.Ramadoss
Assistant Professor,
Department of Information Technology,
VFSTR (Deemed to be)University,
Guntur, Andhra Pradesh.

Textbook

- Introduction to Automata Theory, Languages and Computation.
- By J.E. Hopcroft, and Ullman
- 2rd Edition
- Pearson/Prentice Hall India, 2007.

Automata Theory

Automata (singular: automation) are abstract models of machines that perform computations on an input by moving through a series of states.

At each state of the computation, a transition function determines the next configuration on the basis of a finite portion of the present state.

Automata Theory

An automaton has a mechanism to read input, which is string over a given alphabet. This input is actually written on an input tape /file, which can be read by automaton but cannot change it.

Input file is divided into cells each of which can hold one symbol. Automaton has a control unit which is said to be in one of finite number of internal states. The automation can change states in a defined way.

Finite Automata

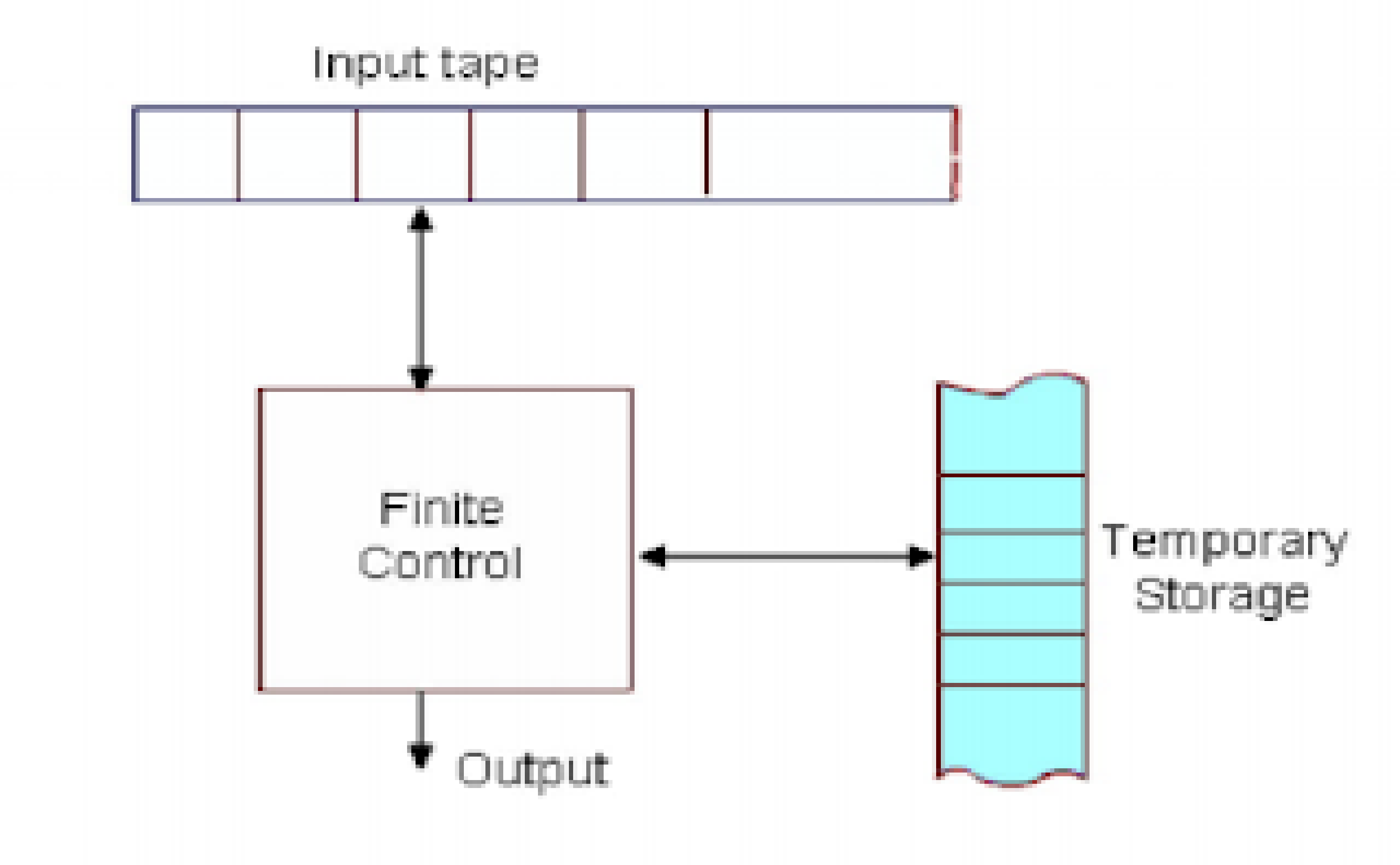


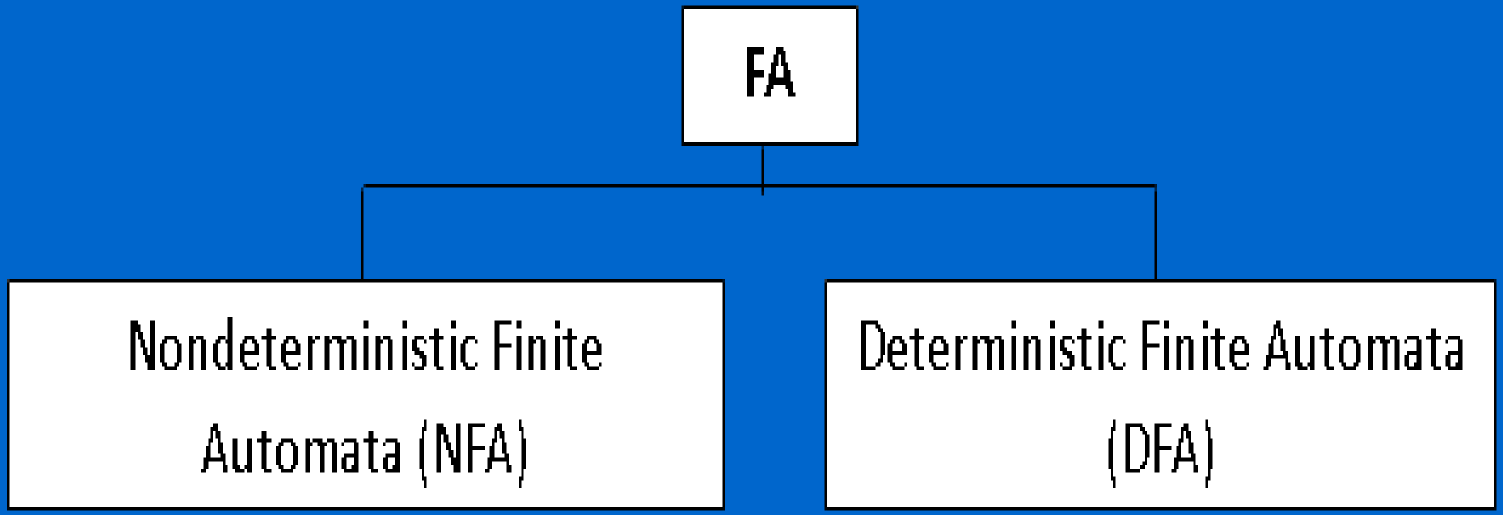
Fig: Diagrammatic representation of a generic automation

Finite automata

A finite automata consists of a finite memory called input tape, a finite non empty set of states, an alphabet, a read-only head, a transition function which defines the change of configuration, an initial state and a finite-non empty set of final states.

Input tape is divided into cells and each cell contains one symbol from of which can hold one symbol.

Types of Finite Automata:



Note: Both NFA and DFA are capable of recognizing what regular expression can denote.

Deterministic Finite Automata(DFA)

Definition

A deterministic finite automaton is defined by a quintuple (5-tuple): A = (Q, ∑, δ, qo, F). Where,

Q = Finite set of states,

∑= Finite set of input symbols,

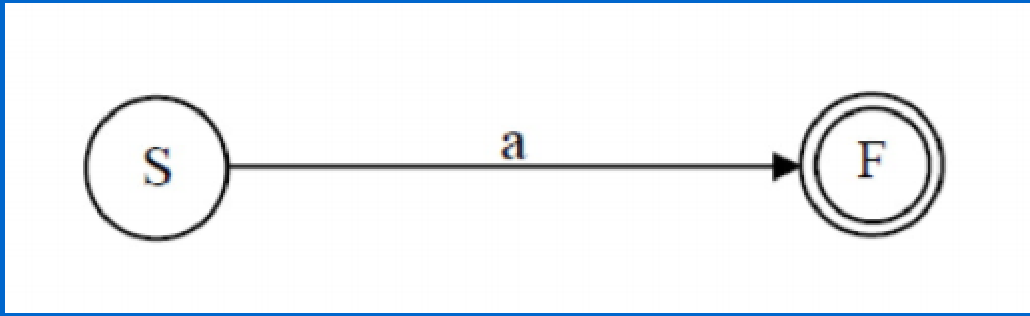
δ= A transition function that maps Q X ∑ → Q

qo: A start state; qo∈Q

F= Set of final states; F ⊆ Q.

A transition function δ that takes as arguments a state, an input symbol and returns a state.

In our diagram, is represented by arcs between states and the labels on the arcs.



If s is a state and a is an input symbol then δ(p,a) is that state q such that there are arcs labeled ‘a’ from p to q.

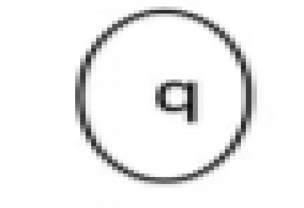
General Notations of DFA:

There are two preferred notations for describing automata.

1. Transition Diagrams
2. Transition Tables

1. Transition Diagrams

A transition diagram for a DFA A = (Q, ∑, δ, qo, F). is a graph

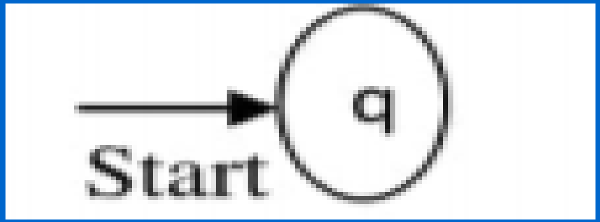
defined as follows: 

(q is State)

For each state is node represented by circle.

General Notations of DFA:

A start stat is represented by a circle with preceding arrow labeled at start.



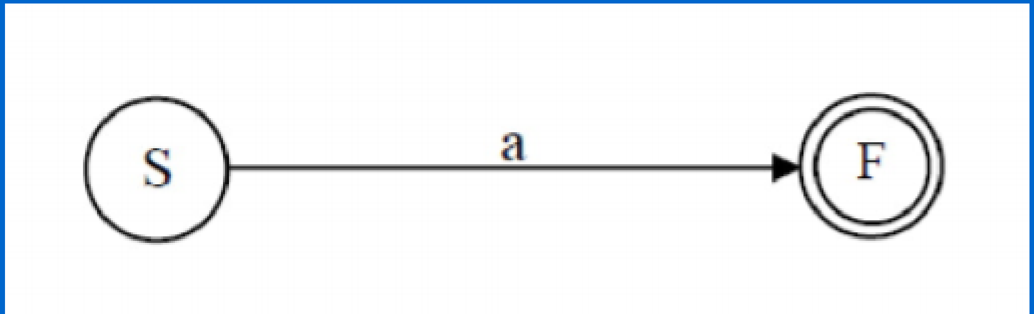
- Final state is marked by a double circle (q is Final State)

(q is Final State)

General Notations of DFA:

For each state q in Q and each input a in ∑ , if $\delta(q, a) = p$ 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 top then arc from q to p is labeled by a list of those symbols.



$\delta(q,a)=p$

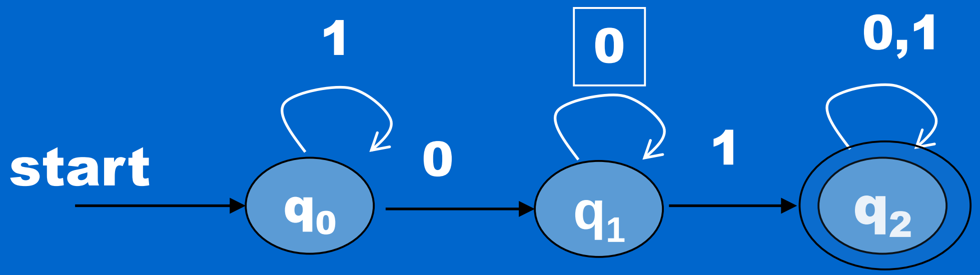
Example #1

- Build a DFA for the following language:
  - L = {w | w is a binary string that contains 01 as a substring}
- Steps for building a DFA to recognize L:
  - $\sum = \{0,1\}$
  - Decide on the states: Q
  - Designate start state and final state(s)
  - δ: Decide on the transitions:
- “Final” states == same as “accepting states”

- Other states == same as “non-accepting states”

DFA for strings containing 01

- What makes this DFA deterministic?



Accepting state

- $Q = \{q_0, q_1, q_2\}$
- $\sum = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition Table

symbols

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

states

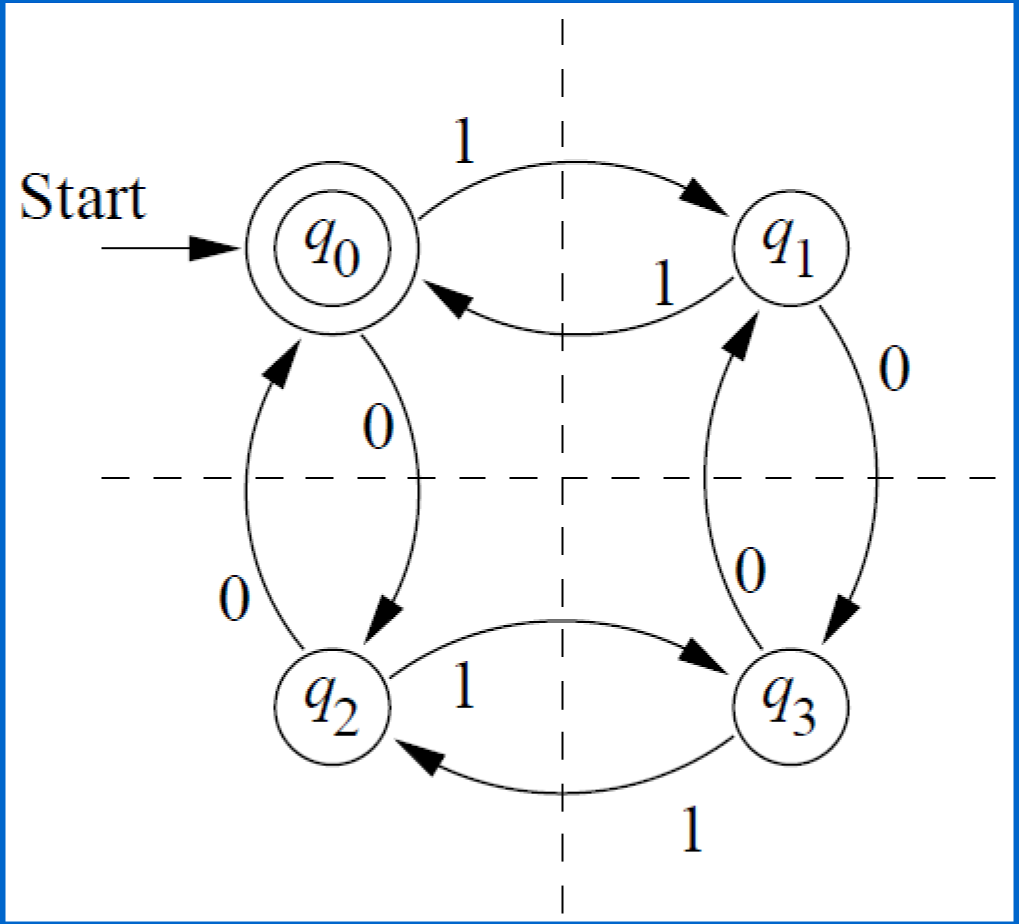
Language of a DFA

A DFA A accepts string *w* if there is a path from q_0 to an accepting (or final) state that is labeled by *w*

- ***i.e., $L(A) = \{ w \mid \hat{\delta}(q_0,w) \in F \}$***
- ***I.e., L(A) = all strings that lead to an accepting state from q_0***

Example #2

DFA accepting all and only strings with an even number of 0's and an even number of 1's.



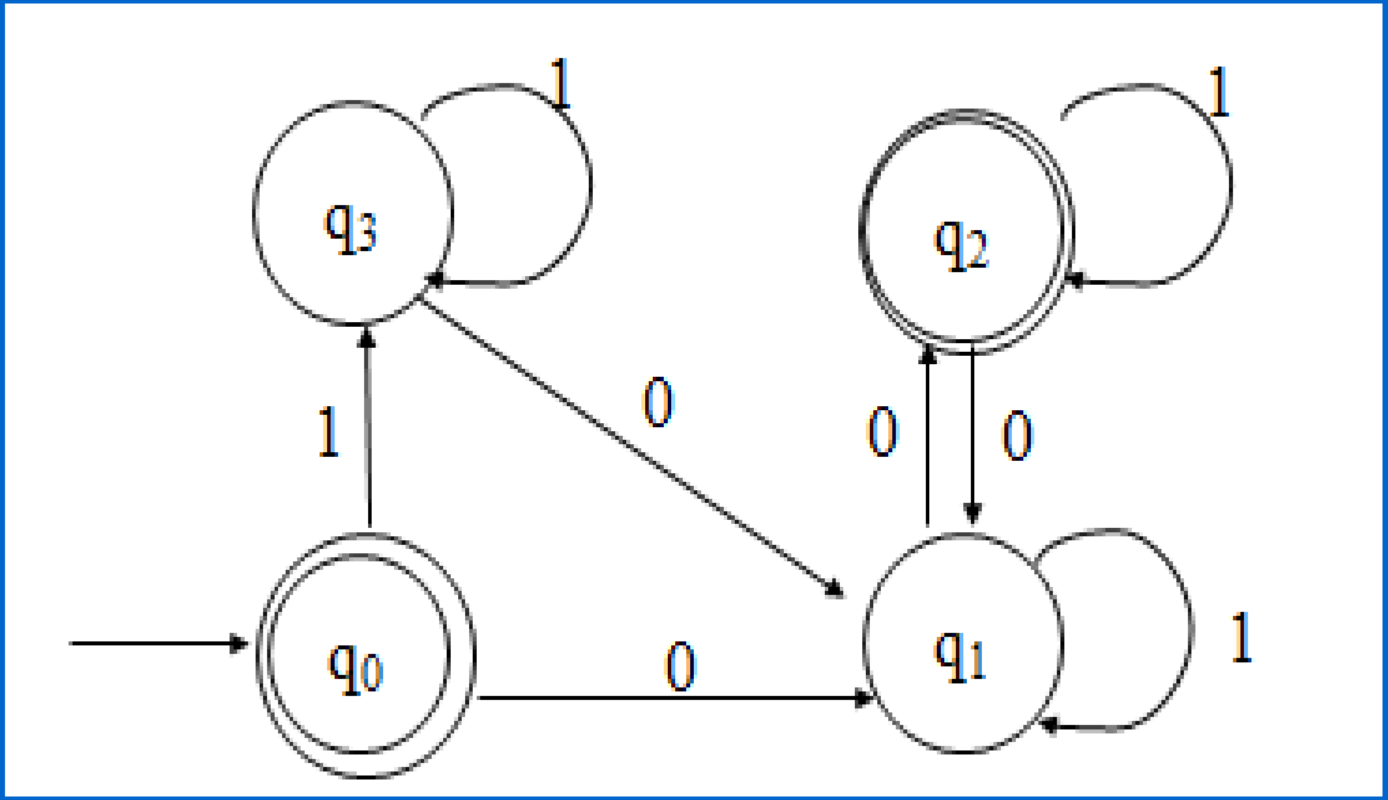
Transition Table

δ	0	1
$\star \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\sum = \{0,1\}$
- start state = q_0
- $F = \{q_0\}$

Example #3

DFA accepting all and only strings with an even number of 0's.

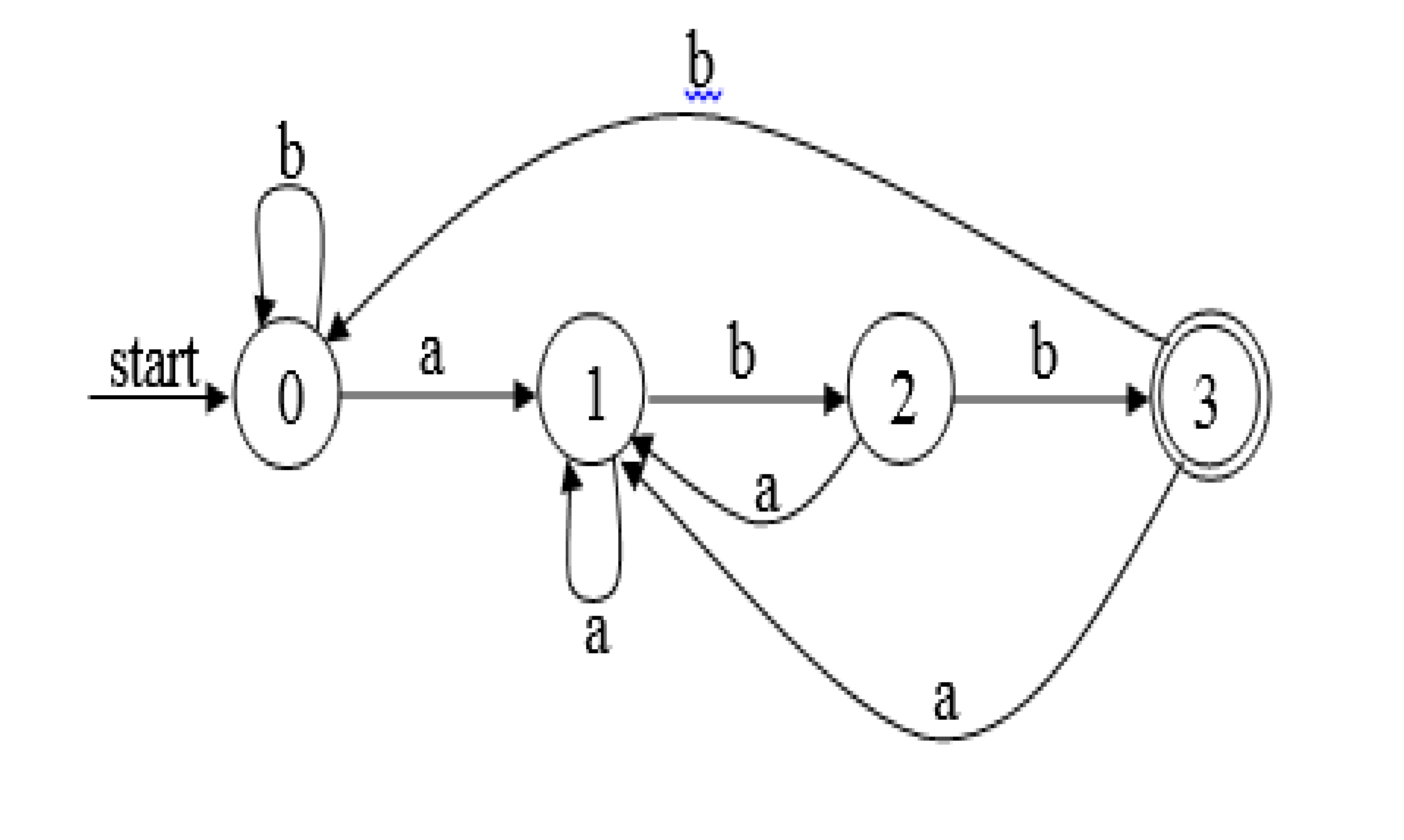


Transition Table

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

q_3

Example: The following figure shows a DFA that recognizes the language (a|b)*abb



Example: The following figure shows a DFA that recognizes the language (a|b)*abb

The Transition Table is:

State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

Non-deterministic Finite Automata (NFA)

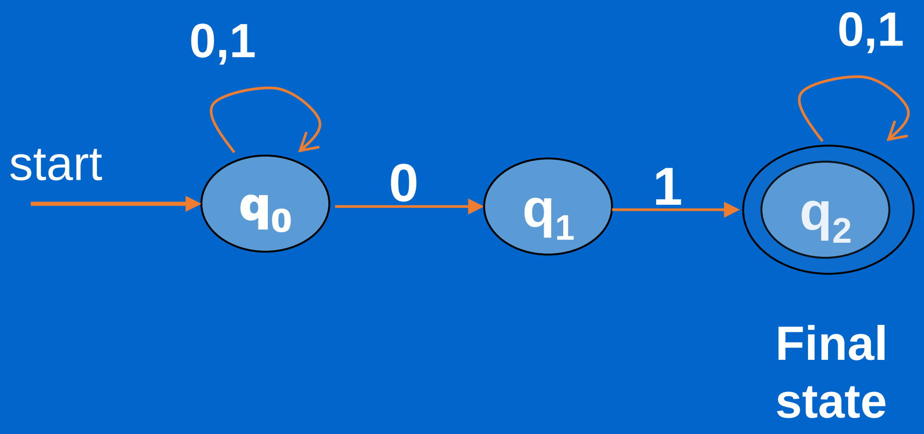
- A Non-deterministic Finite Automaton (NFA) consists of:
  - Q ==> a finite set of states
  - ∑ ==> a finite set of input symbols (alphabet)
  - q_0 ==> a start state
  - F ==> set of accepting states
  - δ ==> a transition function, which is a mapping between Q x ∑ ==> subset of Q
- An NFA is also defined by the 5-tuple:
  - {Q, ∑ , q_0,F, δ }

How to use an NFA?

- Input: a word w in $\sum^*$
- Question: Is w acceptable by the NFA?
- Steps:
  - Start at the "start state" q_0
  - For every input symbol in the sequence w do
  - Determine all possible next states from all current states, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed and if at least one of the current states is a final state then *accept w;* Otherwise, *reject w.*

NFA for strings containing 01

Why is this non-deterministic?



What will hapen if at state q_1 an input of 0 is received?

symbols

δ	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_0\}$
q_1	Φ	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$

states

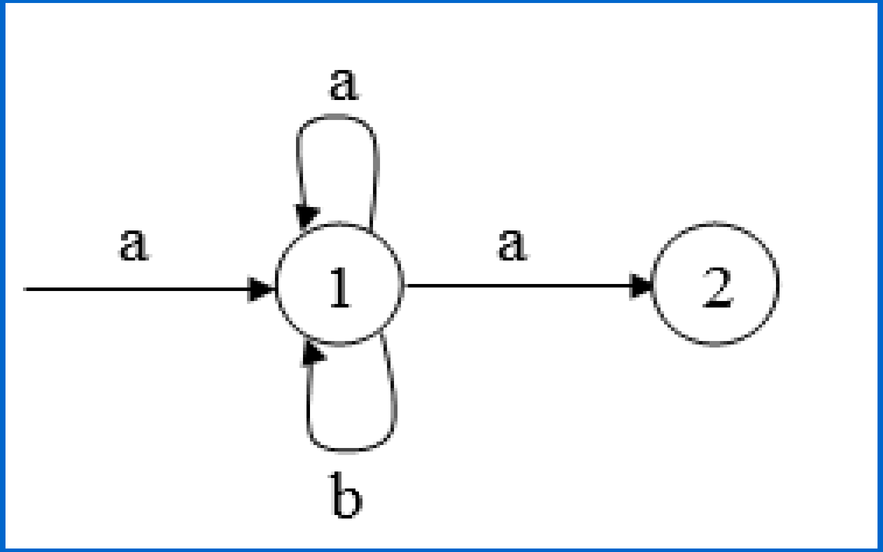
Regular expression: $(0+1)^*01(0+1)^*$

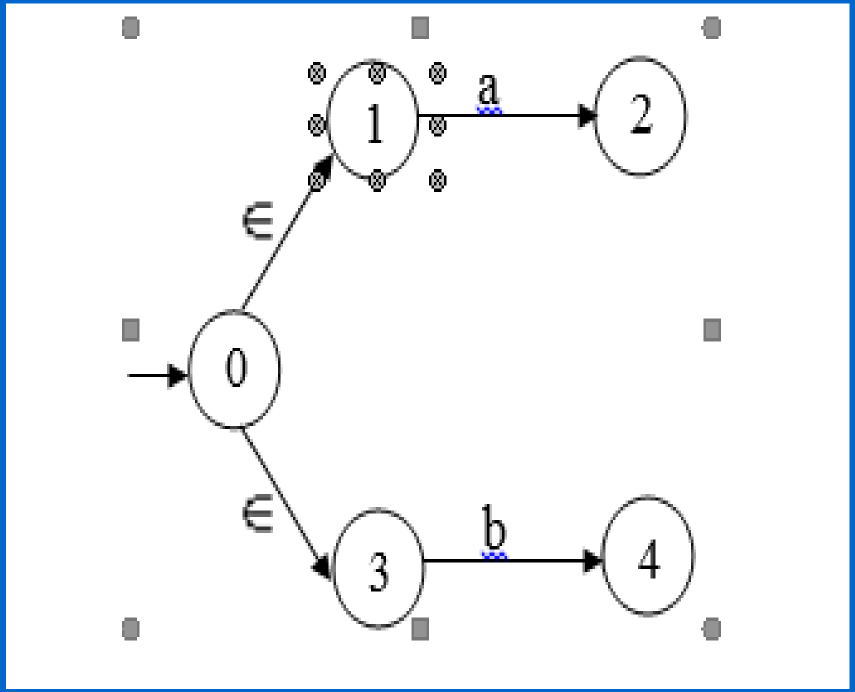
- $Q = \{q_0,q_1,q_2\}$
- $\Sigma = \{0,1\}$
- start state $= q_0$
- $F = \{q_2\}$
- Transition table

Non-Deterministic Finite Automata (NFA)

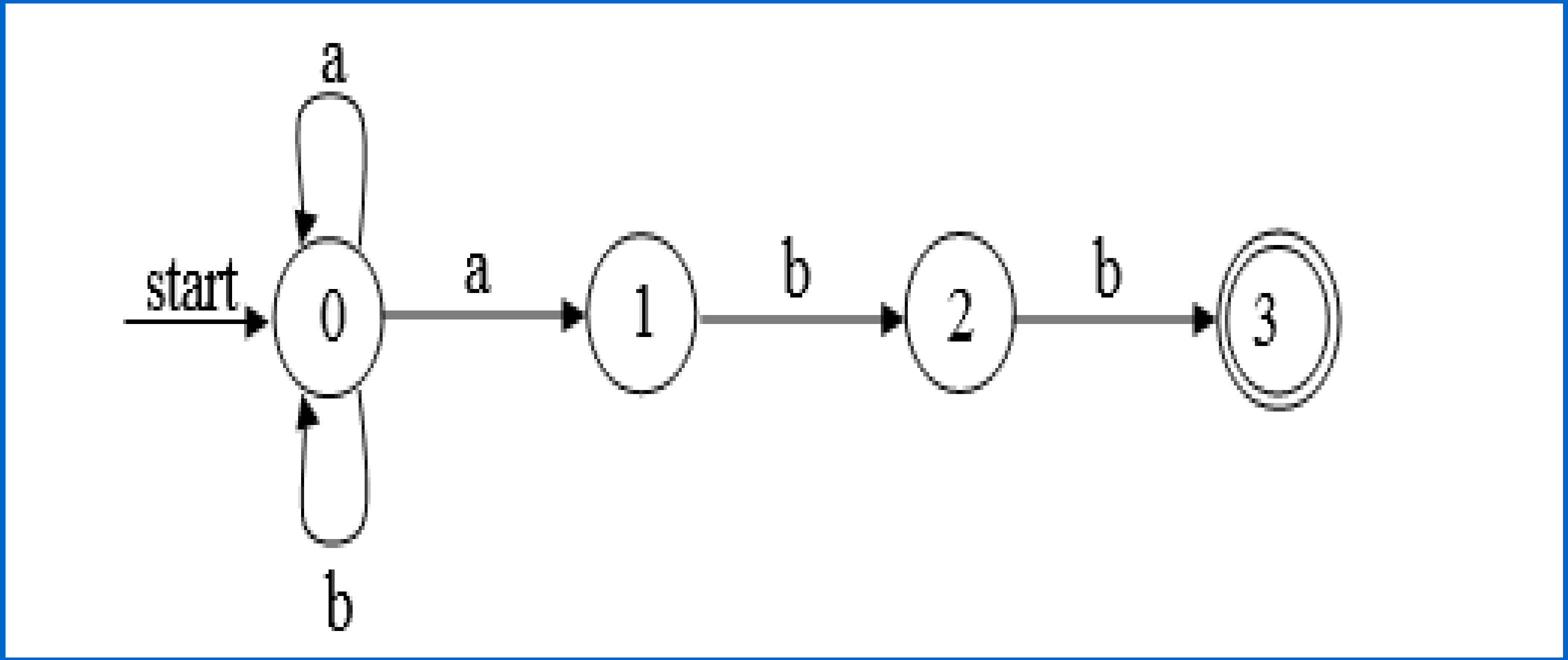
NFA: means that more than one transition out of a state may be possible on a same input symbol.

Also a transition on input ε (ε-transition) is possible

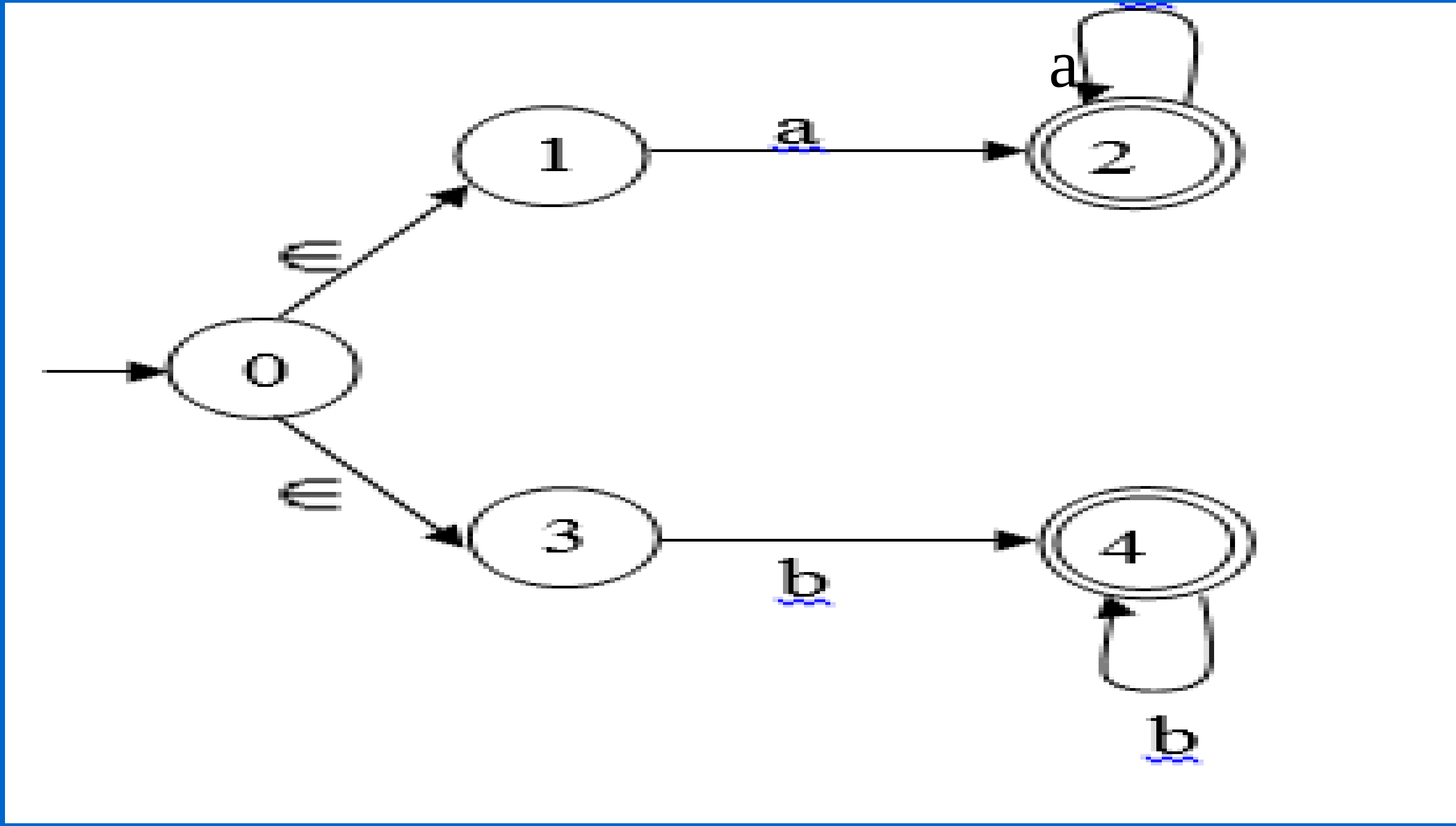




Example #1 : The NFA that recognizes the language $(a \mid b)^*abb$ is shown below:



<u>Example #2</u> The NFA that recognizes the language $aa^*|bb^*$ is shown below:



Language of an NFA

An NFA accepts *w* if *there exists at least one* path from the start state to an accepting (or final) state that is labeled by *w.*

- $L(N) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \Phi \}$

But, DFAs and NFAs are equivalent in their power to capture langauges !!

Differences: DFA vs. NFA

DFA

1. All transitions are deterministic each transition leads to exactly one state.
2. Accepts input if the last state visited is in F
3. Sometimes harder to construct because of the number of states.
4. Practical implementation is feasible

NFA

1. Some transitions could be non-deterministic A transition could lead to a subset of states
2. Accepts input if *one of* the last states is in F
3. Generally easier than a DFA to construct
4. Practical implementations limited but emerging (e.g., Micron automata processor)

Summary

- **DFA**
  - Definition
  - Transition diagrams & tables
- Regular language
- NFA
  - Definition
  - Transition diagrams & tables
- DFA vs. NFA

Exercise

1. Give DFA's accepting the following strings over the alphabet {0,1}:

a) The set of all strings beginning with 101.

b) The set of all strings containing 1101 as a substring.

c) The set of all strings with exactly three consecutive 0's.

THANK YOU