Finite Automata

In this topic we cover

- Introduction to FA and its Representation
- Introduction of DFA, examples and Language.

#Hemanta GC

Informal Description

- Finite automata are abstract machines with finite collections of states and transition rules that take FA from one state to another with/without some input.

- Original application of FA was sequential switching circuits, where the "state" was the settings of internal bits.
- Today, several kinds of software can be modeled by FA.

Notation Finite Automata

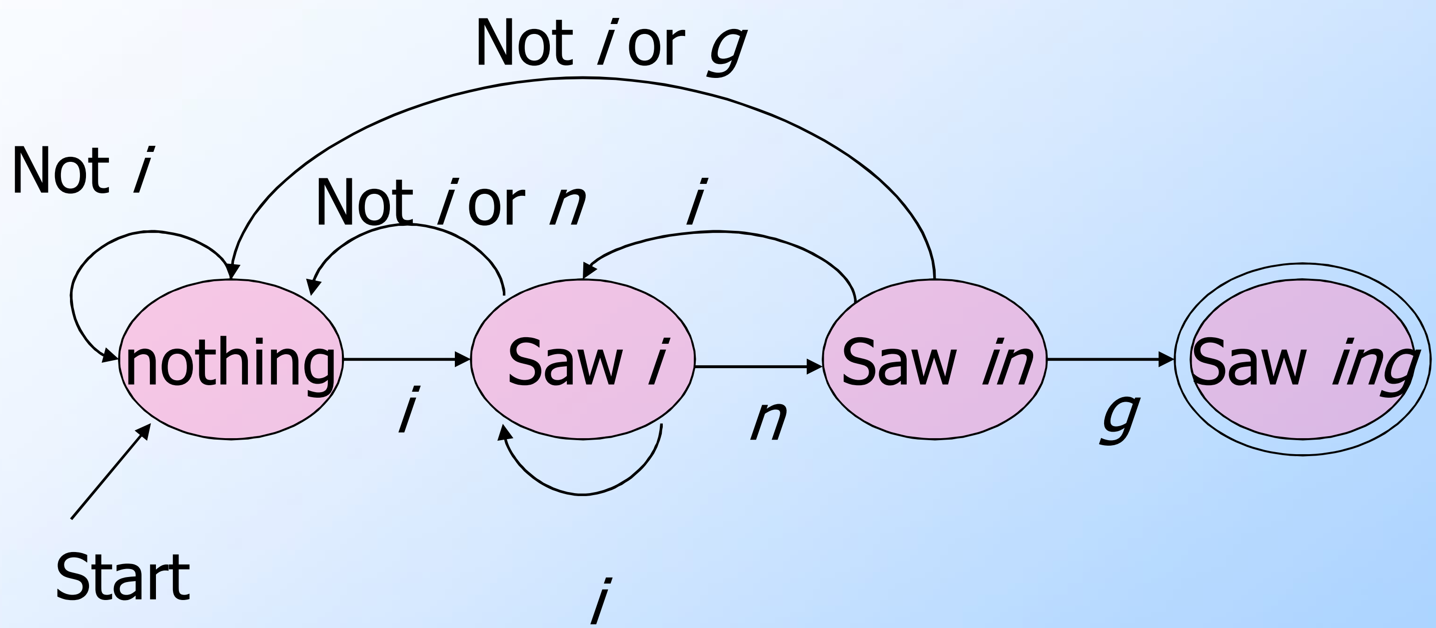
Finite Automata can be represented in following two ways

- Transition Diagram(Graph) : Graphical notation
- Transition Table: Tabular representation of transition rules

Representation of FA: Graph Representation

- A transition graph of FA is a graph such that,
  - **Nodes -**for states represented by a circle.
  - **Arcs-** represent transition function.
  - Arc from state p to state q labeled by all those input symbols that have transitions from p to q.
  - Arrow labeled “Start” for the start state.
  - **Final states** indicated by **double circles.**

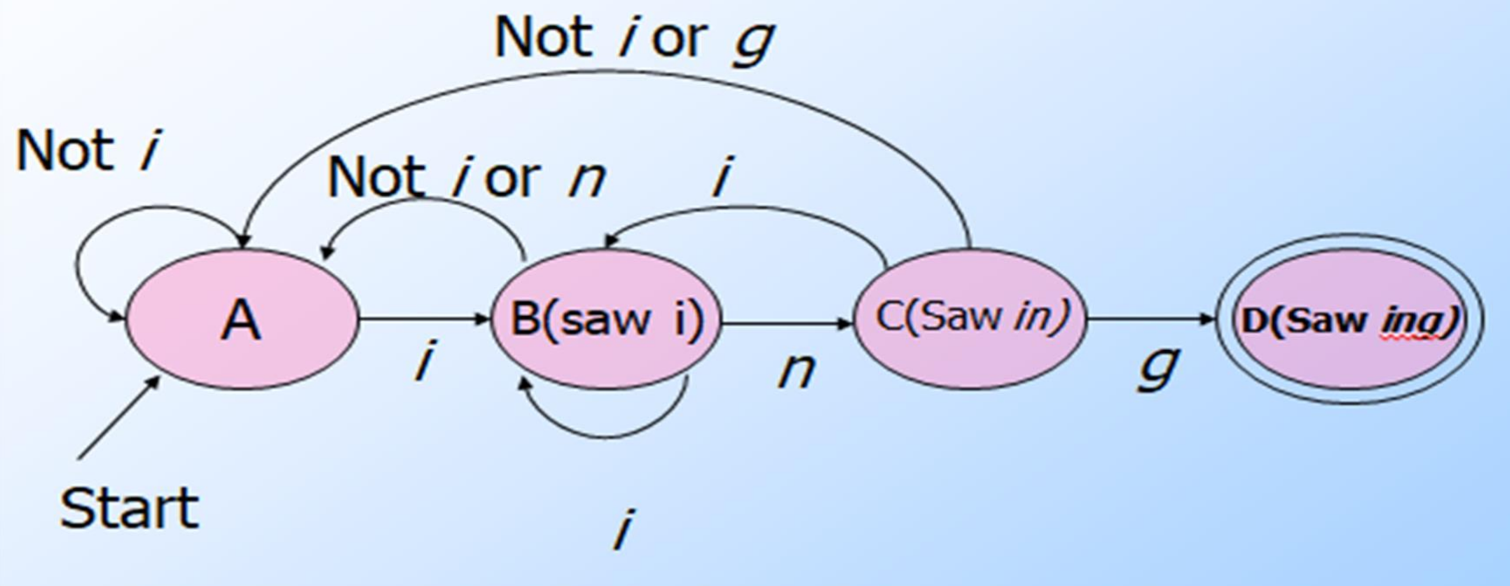
Example: Recognizing Strings Ending in "ing"



Automata to Code

- In C/C++, make a piece of code for each state. This code:
    1. Reads the next input from a state.
    2. Decides on the next state to move.
    3. Jumps to the beginning of the code for that state.

Example: Recognizing Strings Ending in "ing"

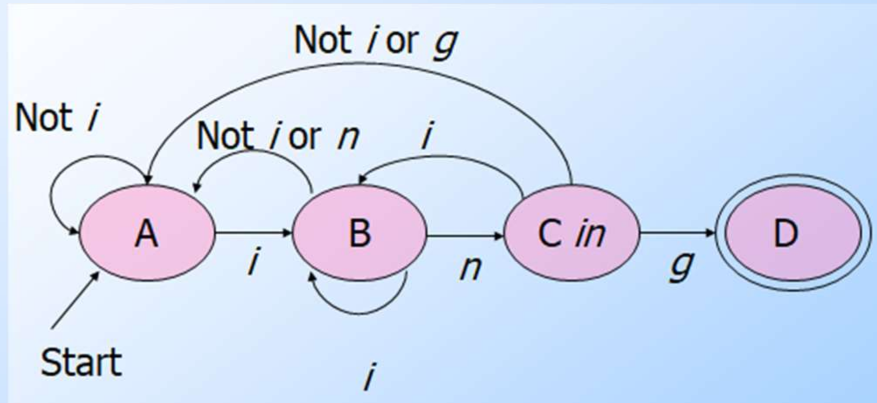


- Here A,B,C and D are states of FA
- This FA can process the string ending with substring "ing"
- The transition process for this DFA are given in transition graph.

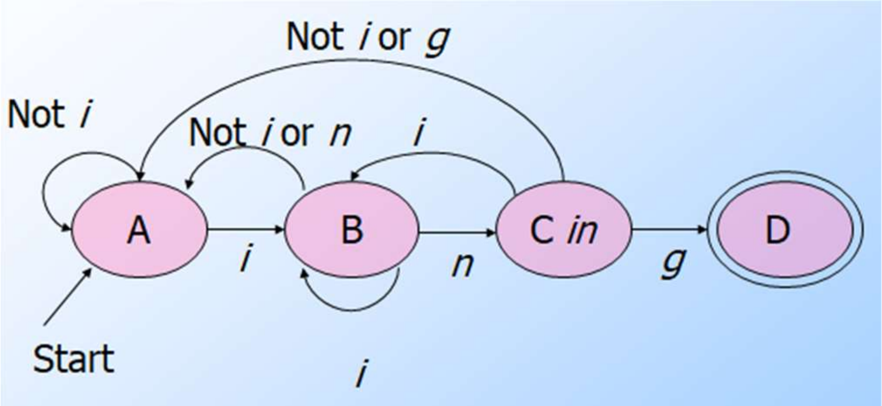
Example: Automata to Code

```
A: /*nothing seen */
  c=getNextInput();
  if(c=='i') goto B;
  else goto A;
B: /* i seen */
  c = getNextInput();
  if (c == 'n')
     goto C;
  else if (c == 'i')
     goto B;
  else goto A;
```

…Continued in next slide

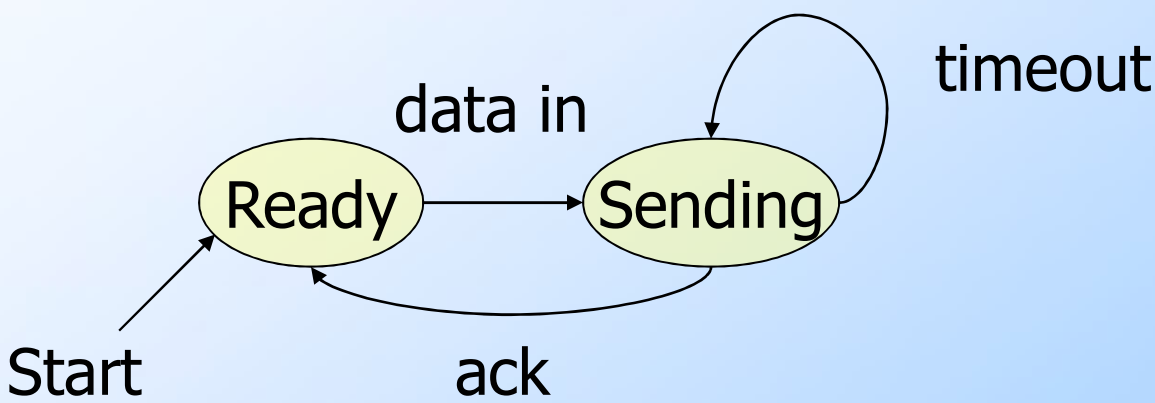


Example: Automata to Code



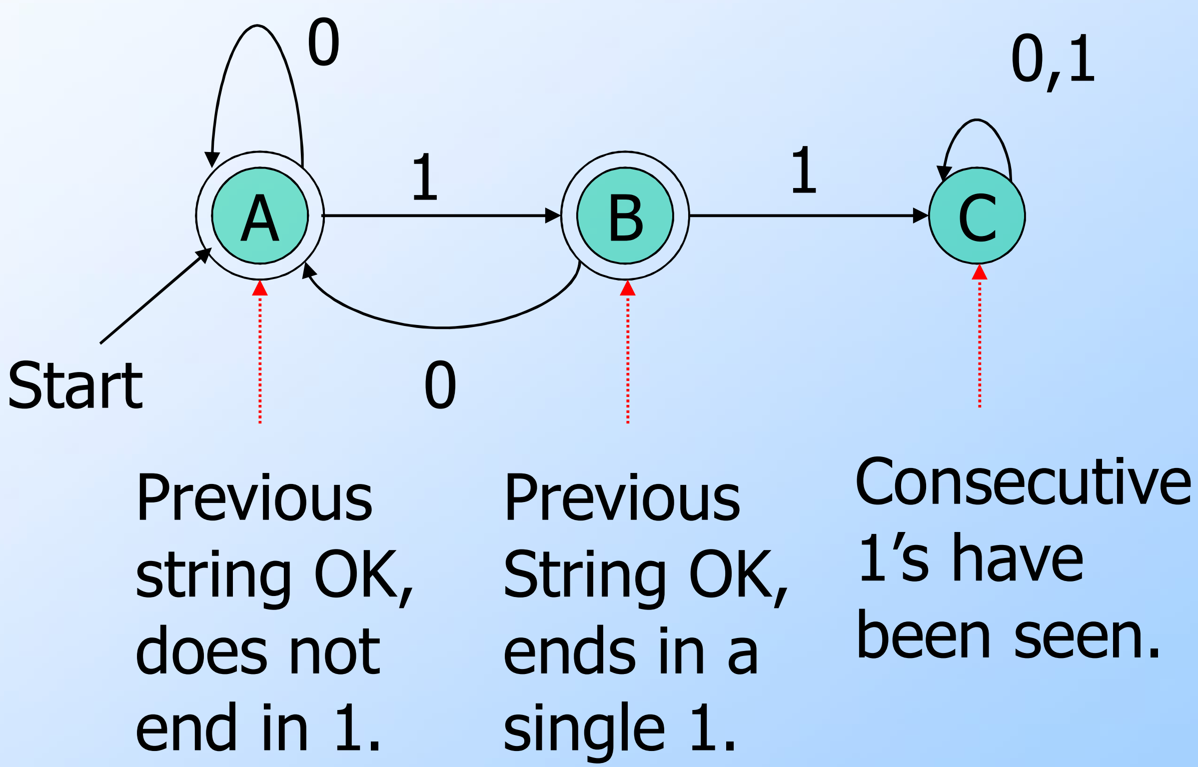

```
C: /* "in" seen */
   c = getNextInput();
   if(c == 'g')
       goto D;
   else if (c == 'i')
       goto B;
   else goto A;
D: return TRUE;
```

Example: Protocol for Sending Data FA Representation



Example: Graph of a DFA

Accepts all strings without two consecutive 1's.



FA Representation: Transition Table

- Transition Table is the tabular representation of states and transitions between the states.
- In transition table row head represent the state and column head represent the input symbol.
- Value in the table cell represents the next state to be move from the state at that row head with input at that column head
- Start state is marked with leading arrow
- Final states are marked with * symbol
- Below is the example of FA described by graph in previous slide.

Arrow for start state

Columns Head = input symbols

	0	1
→*A	A	B
*B	A	C
C	C	C

Final states starred

Rows Head = states

Deterministic Finite Automata

- A Deterministic Finite Automata(DFA) can not be more than one state at a time.
- A formal definition:
  - A DFA is defined by 5-tuples as $D=(Q, \Sigma, \delta, q_0, F)$, where

    Q = A finite set of *states.*

    Σ = An *input alphabet.*

    δ = A *transition function* that maps $Q \times \Sigma \rightarrow Q$

    q_0 = A *start state* (q_0 in Q).

    F = A set of *final states* ($F \subseteq Q$)

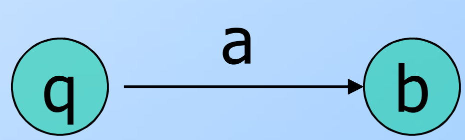
  - "Final" states are also known as "accepting" states.

The Transition Function

- Takes two arguments: a state from Q and an input symbol from its alphabet Σ and maps to a state in Q.

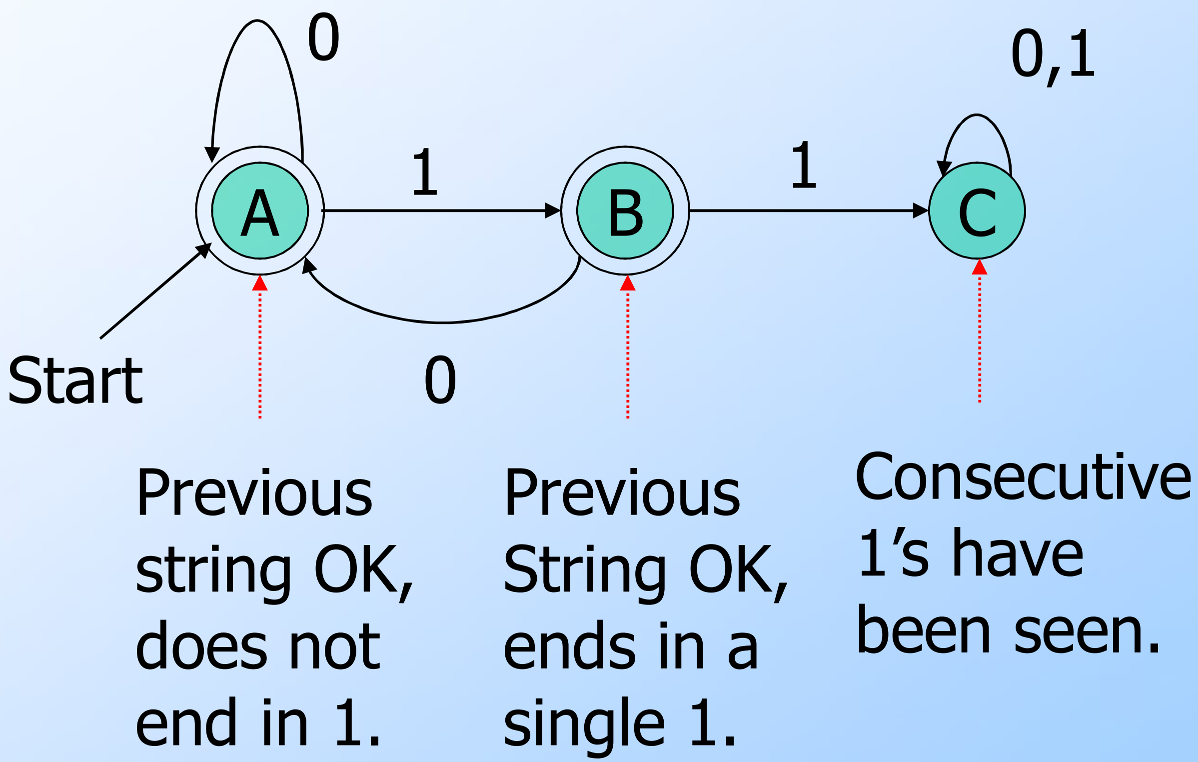
  i.e. $Q \times \Sigma \rightarrow Q$

- δ(q, a) = the state that the DFA goes to when it is in state *q* and input *a* is received.
- δ(q, a) =b is represented in graph as ,



Recall the Example:

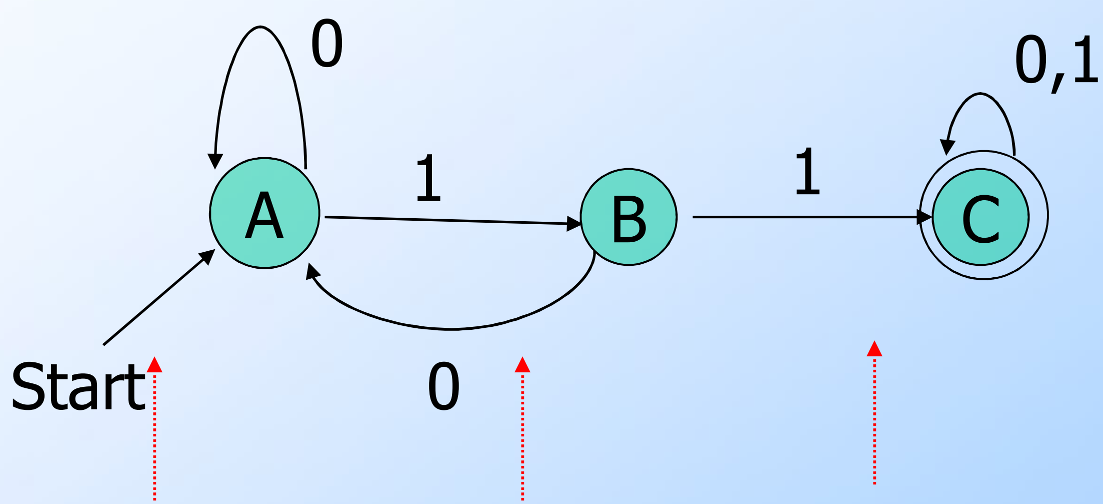
Accepts all strings without two consecutive 1's.



Graph of a DFA

Another Example:

Accepts all strings with two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

Consecutive 1's have been seen.

Graph of a DFA

Alternative Representation: Transition Table

Final states starred

Columns = input symbols

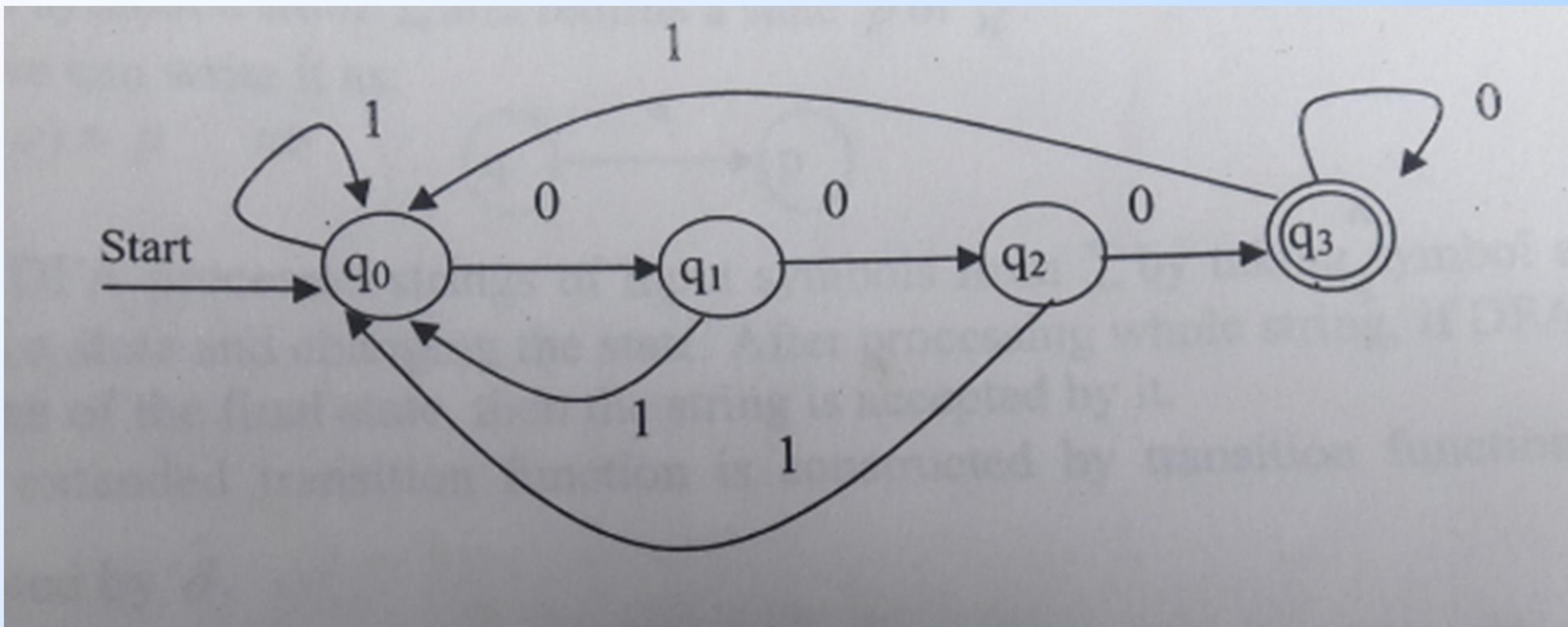
Arrow for start state

	0	1
→ A	A	B
B	A	C
* C	C	C

Rows = states

Example

DFA accepting all strings form alphabet Σ={0,1} ending with three consecutive 0s.

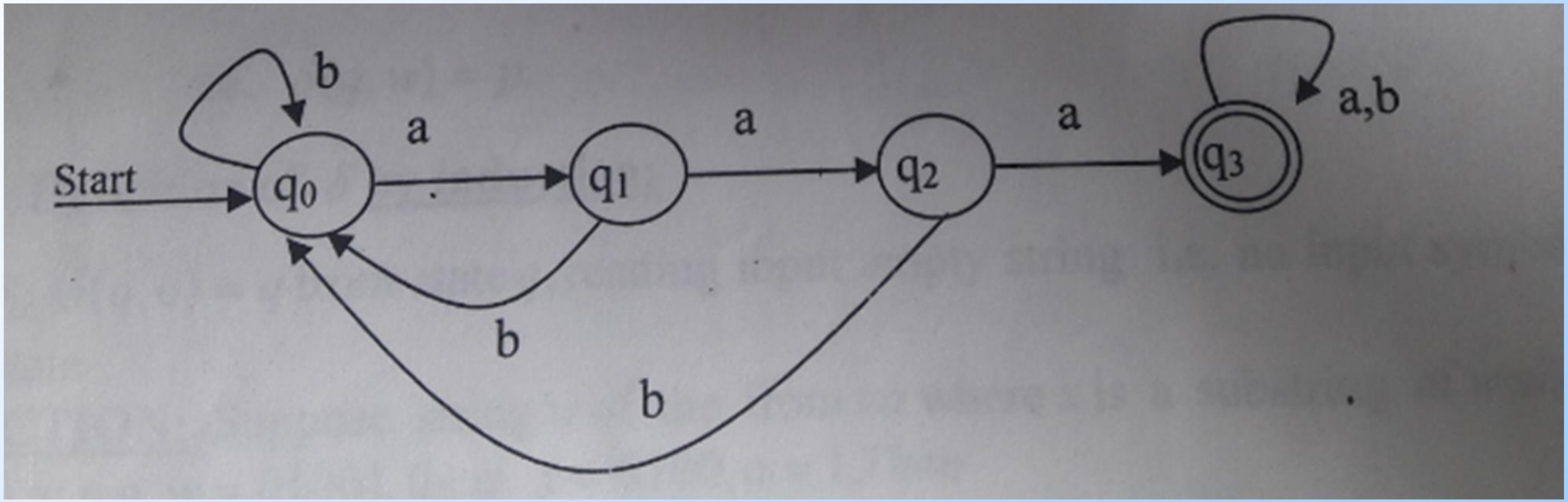


Example

DFA accepting all strings form alphabet Σ={a,b} with three consecutive a.

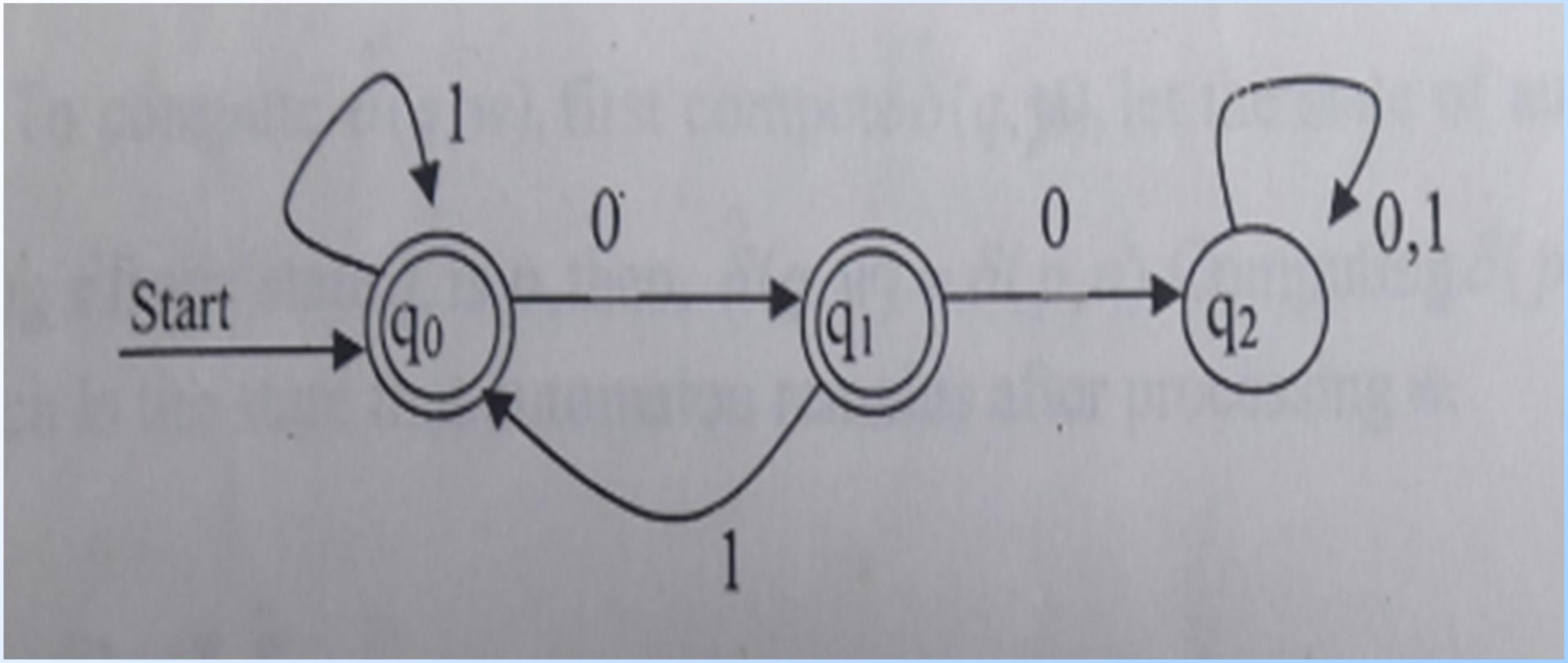
Or

DFA accepting Set of All Strings with a substring 'aaa' from alphabet Σ={a,b}



Example

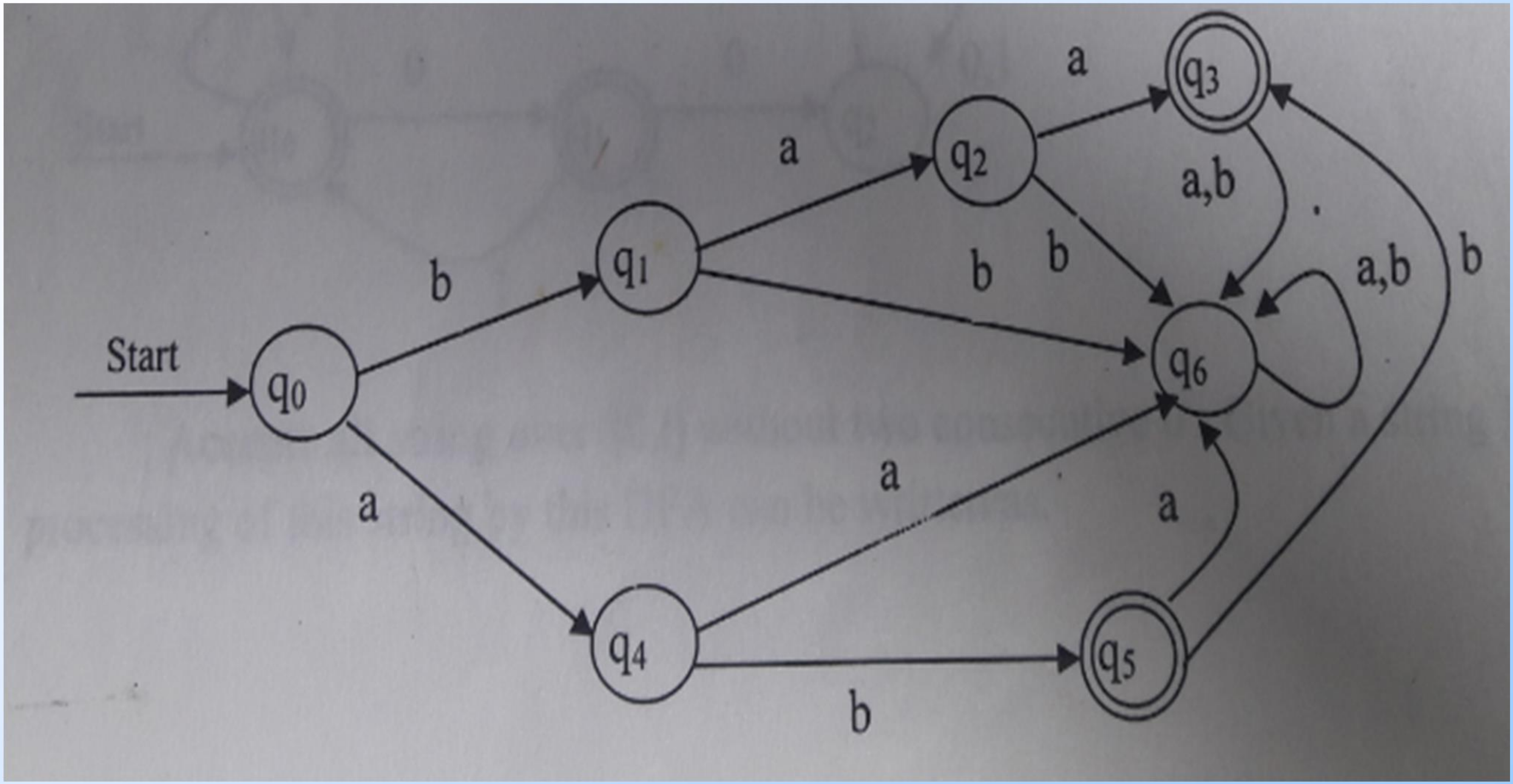
DFA accepting all strings form alphabet $\Sigma=\{0,1\}$ those do not contain two consecutive 0s.



- **For example, 1010, 111, 1101010 accepted**
- **Similarly 00, 1100, 001, 1001,11100110101 not accepted.**

Example

DFA accepting language L={baa,ab,abb} from al phabet Σ={a,b}



Extended Transition Function

- We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.
- The transition function that takes first argument as a state and second argument as a string and finally maps to a state is called the extended transition function of DFA.
- Extended transition function is denoted by $\hat{\delta}$.
- $\hat{\delta}(q,w)=p$ means the DFA from state q takes input string w and moves to next state p after reading all symbols of string w.

Extended Transition Function ($\hat{\delta}$)

- We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.
- Induction on length of string.
  - Basis: $\hat{\delta}(q, \epsilon) = q$
  - Induction: Suppose string w=xa where x is a substring of w without last symbol 'a' , then $\hat{\delta}(q,xa) = \delta(\hat{\delta}(q,x),a)$.
  - To compute $\hat{\delta}(q,xa)$ first compute $\hat{\delta}(q,x)$ which will give a state.
  - Let $\hat{\delta}(q,x)= p$ then from p compute $\delta(p,a)$ which will give a state
  - w is a string; a is an input symbol.

Extended $\hat{\delta}$: Intuition

- Convention:
  - … w, x, y, x are strings.
  - a, b, c,… are single symbols.
- Extended δ is computed for state q and inputs $a_1a_2…a_n$ by following a path in the transition graph, starting at q and selecting the arcs with labels a_1, a_2,…,a_n in turn.

Example: Extended Delta

Let us Compute $\hat{\delta}(B,011)$ for below DFA

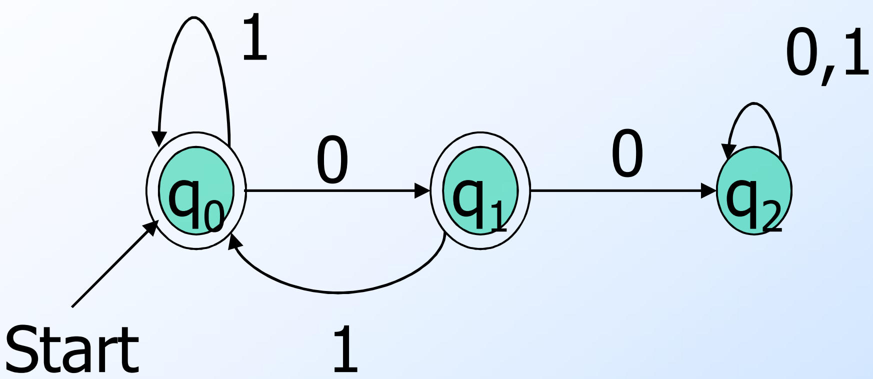
	0	1
→A	A	B
*B	A	C
C	C	C

$$\delta(B,011) = \delta(\delta(B,01),1) = \delta(\delta(\delta(B,0),1),1) =$$

$$\delta(\delta(A,1),1) = \delta(B,1) = C$$

Recall the Example:

Accepts all strings without two consecutive 0's.



Let us compute

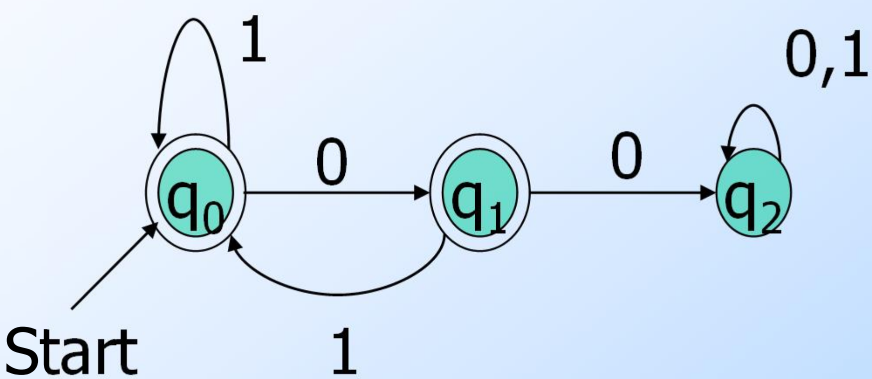
$\hat{\delta}(q_0, 10011)$ for above DFA.

$$
\begin{aligned}
\hat{\delta}(q_0,10011) &= \delta(\hat{\delta}(q_0,1001),1) \\
&= \delta(\delta(\hat{\delta}(q_0,100),1),1) \\
&= \delta(\delta(\delta(\hat{\delta}(q_0,10),0),1),1) \\
&= \delta(\delta(\delta(\delta(\hat{\delta}(q_0,1),0),0),1),1) \\
&= \delta(\delta(\delta(\delta(\delta(\hat{\delta}(q_0,\epsilon),1),0),0),1),1) \\
&= \delta(\delta(\delta(\delta(\delta(q_0,1),0),0),1),1) \\
&= \delta(\delta(\delta(\delta(q_0,0),0),1),1) \\
&= \delta(\delta(\delta(q_1,0),1),1) \\
&= \delta(\delta(q_2,1),1) \\
&= \delta(q_2,1) \\
&= q_2.
\end{aligned}
$$

String Accepted by DFA:

A string x is accepted by DFA, $D=(Q, \Sigma, \delta, q_0, F)$
if $\hat{\delta}(q_0,x)=p$ for some state $p \in F$.

For given DFA below,
Consider a string $x=100$ ➔➔



$$\hat{\delta}(q_0,x) = \hat{\delta}(q_0,100)$$
$$= \delta(\hat{\delta}(q_0,10),0) = \delta(\delta(\hat{\delta}(q_0,1),0),0)$$
$$= \delta(\delta(\delta(\hat{\delta}(q_0,\in),1),0),0)$$
$$= \delta(\delta(\delta(q_0,1),0),0) = \delta(\delta(q_0,0),0)$$
$$= \delta(q_1,0) = q_2 \notin F \text{ Hence not accepted}$$

And For string y=010 ➔➔

For string $y = 010$

$$\hat{\delta}(q_0 010) = \delta(\hat{\delta}(q_0,01),0)$$
$$= \delta(\delta(\hat{\delta}(q_0,0),1),0)$$
$$= \delta(\delta(\delta(\hat{\delta}(q_0,\in),0),1),0)$$
$$= \delta(\delta(\delta(q_0,0),1),0)$$
$$= \delta(\delta(q_0,1),0)$$
$$= \delta(q_0,0)$$
$$= q_1$$

Since $q_1 \in F$, then this string is accepted.

- Is 0101 accepted by M.? Similarly accept

Language of DFA

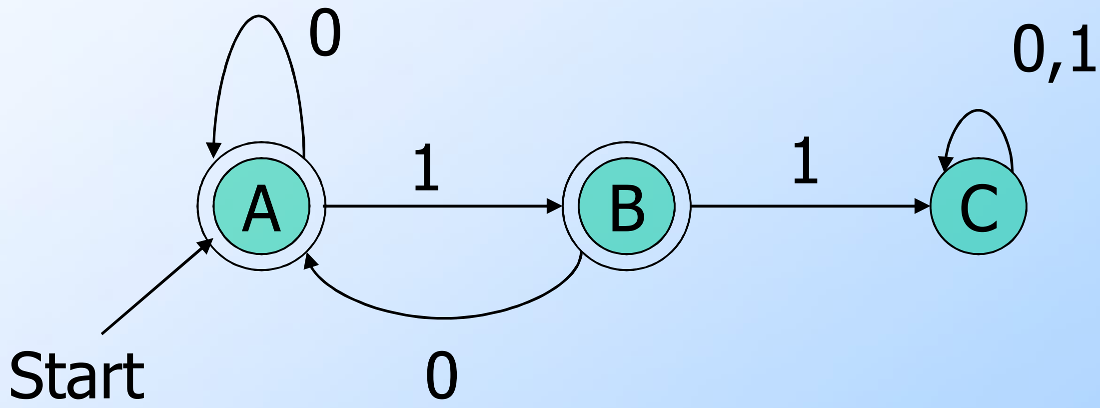
- The language accepted by a DFA
  $D=(Q,\Sigma, \delta ,q_0,F)$ denoted by L(D) is defined as
  - $L(D)= \{ w \mid \hat{\delta}(q_0,w) \in F\}$

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 final (accepting) states.

- The Language of DFA(in generally FA) is a regular language which is simplest language in the formal language and automata theory.

Example: String in a Language

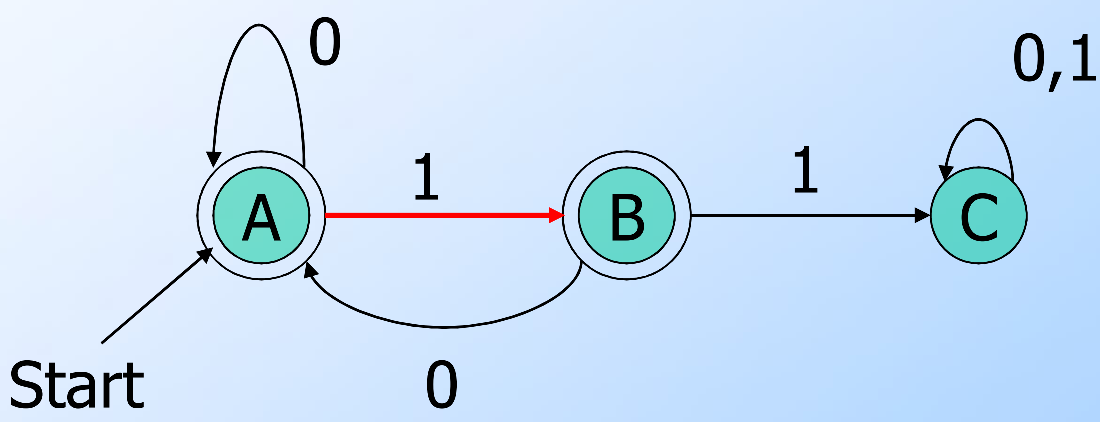
String 101 is in the language of the DFA below.
Start at A.



Example: String in a Language

String 101 is in the language of the DFA below.

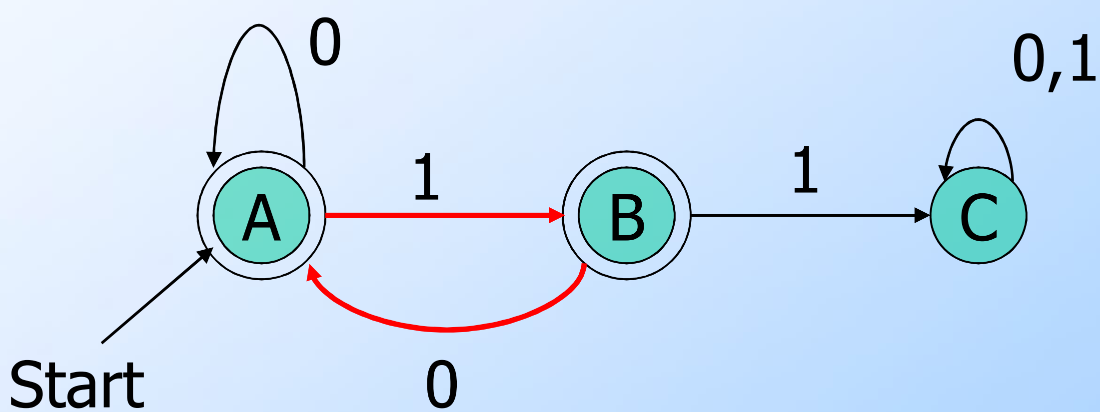
Follow arc labeled 1.



Example: String in a Language

String 101 is in the language of the DFA below.

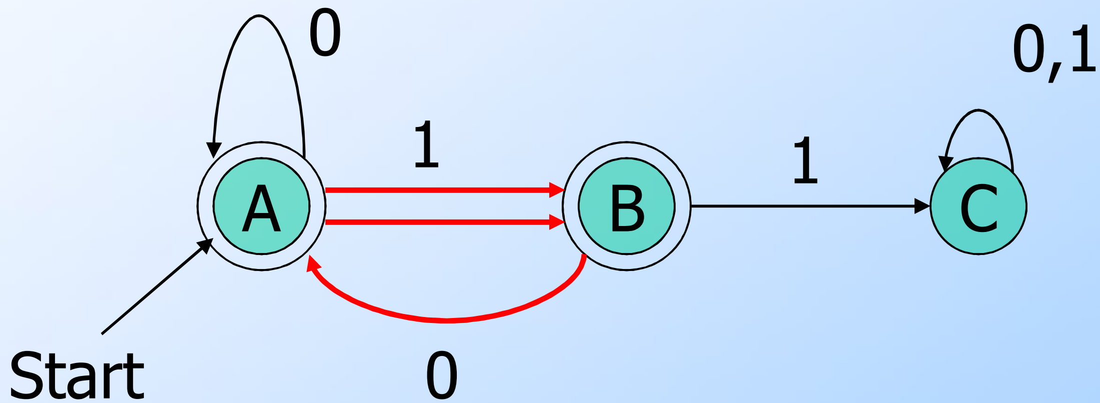
Then arc labeled 0 from current state B.



Example: String in a Language

String 101 is in the language of the DFA below.

Finally arc labeled 1 from current state A. Result is an accepting state, so 101 is in the language.



Example – Concluded

- The language of our example DFA is:

{w | w is in {0,1}* and w does not have two consecutive 1's}

Such that…

These conditions about w are true.

Read a *set former* as "The set of strings w…

Non-Deterministic Finite Automata

- A Non-deterministic Finite Automata(NFA) can have ability to be in more than one state at a time.

- **Transitions from a state on an input symbol can be to any subset of states.**

- **The property of FA to move in several state from a state with an input symbol is called the non-determinism in the transition.**

- **The non-determinism do not add power of computation to the FA but only flexibility to represent the language in terms of FA**

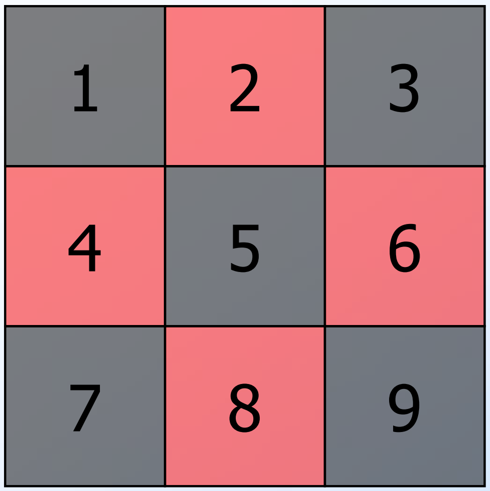
Example: Moves on a Chessboard

- States = squares, {1,2,3,4,5,6,7,8,9}

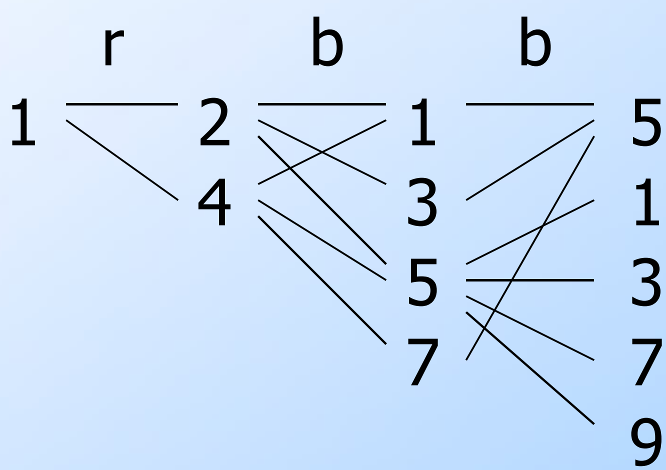
- Inputs = {r,b} such that r(move to an adjacent red square) and b (move to an adjacent black square).

- Start state, final state are in opposite corners, here let start state is 1 and final state is 9.

Example: Chessboard – (2)



		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5



← Accept, since final state reached

Non-Deterministic Finite Automata

- A formal definition:
  - A NFA is defined by 5-tuples as $D=(Q, \Sigma, \delta, q_0, F)$, where

    Q = A finite set of *states.*

    Σ = An *input alphabet.*

    δ = A *transition function* that maps $Q \times \Sigma \to 2^Q$

    q_0 = A *start state* (q_0 in Q).

    F = A set of *final states* ($F \subseteq Q$)

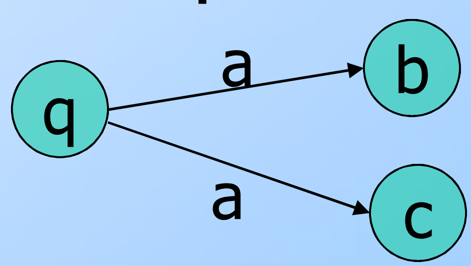
  - "Final" states are also known as "accepting" states.

The Transition Function

- Takes two arguments: a state from Q and an input symbol from its alphabet Σ and maps to a subset of states in Q.

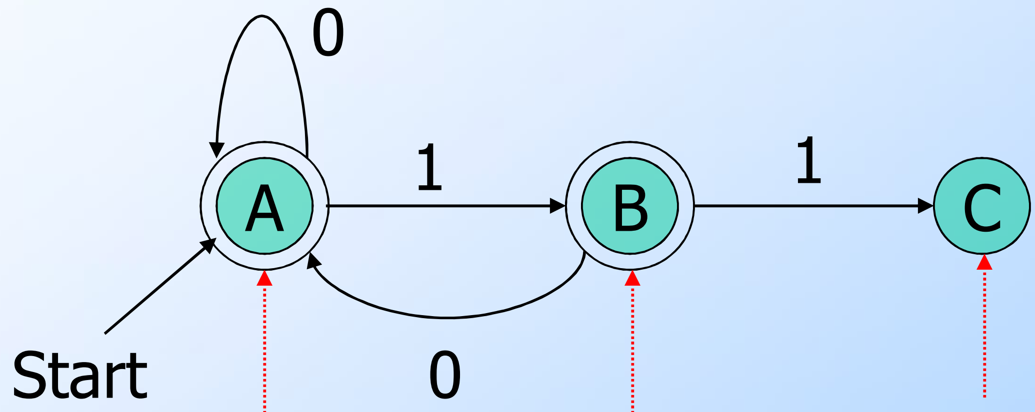
  i.e. $Q \times \Sigma \rightarrow 2^Q$

- δ(q, a) = the states that the NFA goes to when it is in state *q* and input *a* is received.
- δ(q, a) ={b,c} is represented in graph as ,cc



Recall the Example:

Accepts all strings without two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

Consecutive 1's have been seen.

Graph of a NFA

Alternative Representation: Transition Table

Final states starred

Columns = input symbols

Arrow for start state

	0	1
→ * A	{A}	{B}
* B	{A}	{C}
C	ϕ	ϕ

Rows = states

Note: Unlike DFA, the entries in the transition table of NFA are the set of states rather than a single state. If there is no transition from a state of NFA with any input symbol, then transition table entry for that transition will be written as {} or ϕ which is empty set

NFA : EXample

NFA accepting all strings over alphabet {0,1} ending with three consecutive 0s

- **Look at below examples, Figure 1 : NFA and Figure 2: DFA for the same language.**

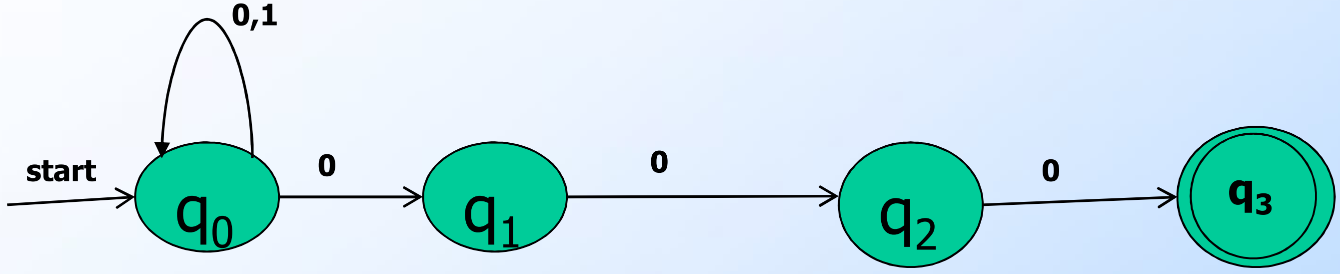


Figure 1: NFA

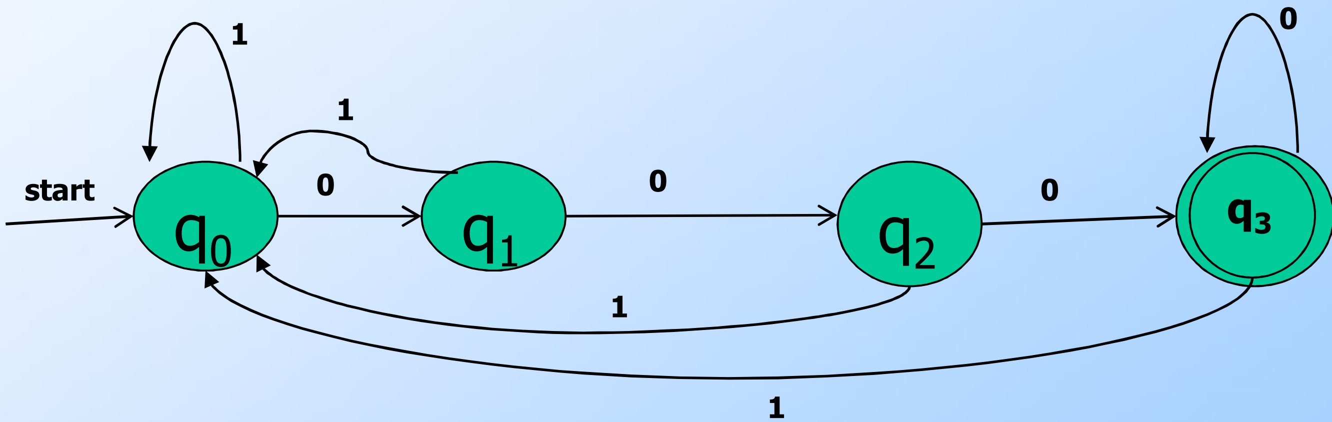


Figure 1: DFA

Here, $(Q, \Sigma, \delta, q_0, F)$
Where
$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0,$
$q_0 = q_0,$
$F = \{q_3\}$
δ is given in transition graph for each of DFA and NFA.

Extended Transition Function($\hat{\delta}$)

- In NFA, $\hat{\delta}$ is a transition function that takes a state ***q*** and a string from its alphabet ***w*** as arguments and returns the set of states that NFA will be in if it starts in q and processes the string ***w***.
- Look at the example below,

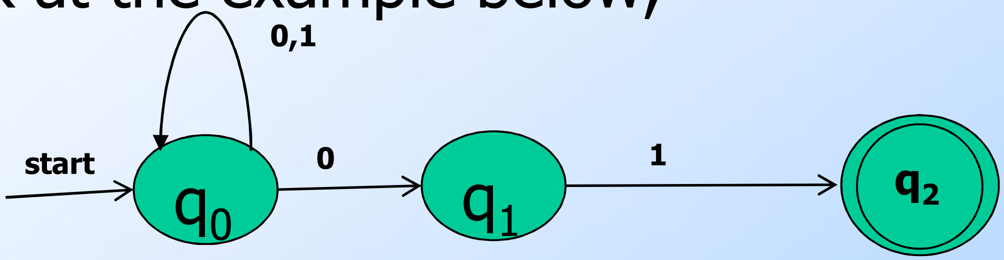


Figure: NFA

  - $\hat{\delta}(q_0,001)= \{q_0,q_2\}$
  - $\hat{\delta}(q_0,00110) = \{q_0,q_1\}$
  - $\hat{\delta}(q_0,000) = \{q_0,q_1\}$ and so on…

Extended Transition Function ($\hat{\delta}$)

- We describe the effect of a string of inputs on a NFA by extending δ for input a state and a string.
- Induction on length of string.
  - Basis: $\hat{\delta}(q, \epsilon) = q$, i.e. without reading any input NFA is at same state.
  - Induction: Suppose string w=xa where x is a substring of w without last symbol ‘a’ , then **$\hat{\delta}(q,xa) = \delta(\hat{\delta}(q,x),a)$.**
  - To compute $\hat{\delta}(q,xa)$ first compute $\hat{\delta}(q,x)$ which will give a set of subset of states in Q.
  - Let **$\hat{\delta}(q,x)= \{p_1,p_2, \dots \dots p_k\}$** then from **$\{p_1,p_2, \dots \dots p_k\}$** compute **$\delta(\{p_1,p_2, \dots \dots p_k\} ,a)$** which will give another subset of states in Q.
  - So we can write if , $\bigcup_{i=1}^{k} \delta(pi, a) = \{r1, r2, r3, \dots \dots rm\}$ say
  - Then **$\hat{\delta}(q,w) = \hat{\delta}(q,xa) = \{r1, r2, r3, \dots \dots rm\}$**
  - w is a string; a is an input symbol.

Example: Processing of string 01101 by NFA

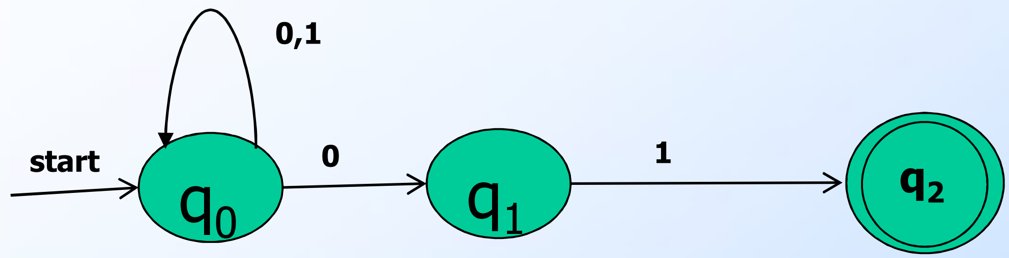
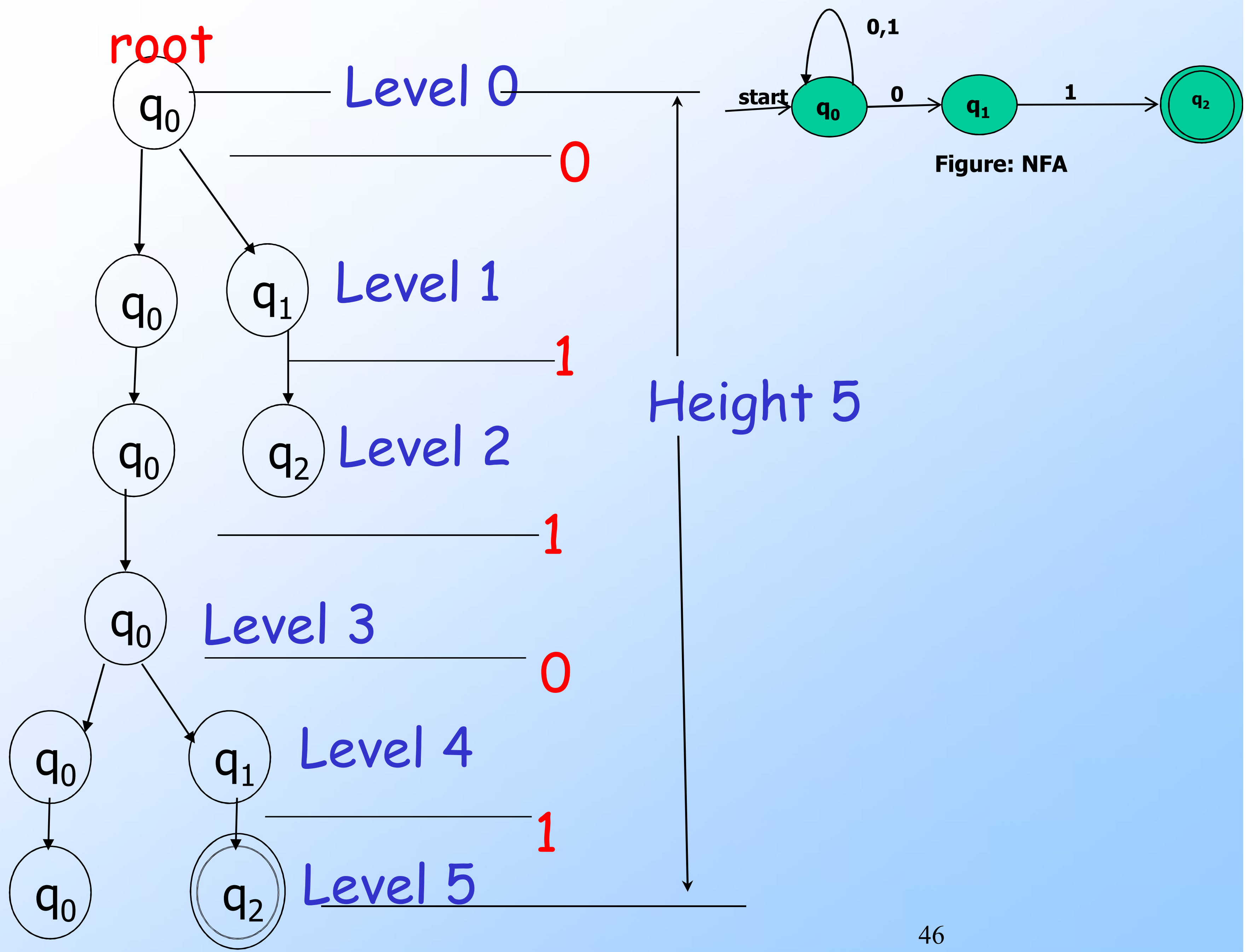


Figure: NFA

- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 01) = \delta(\{q_0, q_1\}, 1\} = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$
- $\delta(q_0, 011) = \delta(\{q_0, q_2\}, 1\} = \{q_0, \}$
- $\hat{\delta}(q_0, 0110) = \delta(\{q_0\}, 0\} == \{q_0, q_1\}$
- $\hat{\delta}(q_0, 01101)\ \delta(\{q_0, q_1\}, 1\} = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$
- Here, the states that the given NFA remains after processing string 01101 are $\{q_0, q_2\}$ which is subset of Q and cotains a final state q_2, so we say this string is accepted by the NFA

The computation tree for NFA

- For processing of a given input string to the NFA can be explained using a tree- called computation tree
- In Computation tree, root is always the start state of NFA.
- From root node of tree, the path of the NFA that follows to process the given string is shown in the arcs to next state as node.
- All possible paths are traced and at the end of processing, look at the last level of tree.
- At last level , if there is any one final state node, we conclude that the given string is accepted by NFA otherwise not.


root
Level 0
0
Level 1
1
Level 2
1
Level 3
0
Level 4
1
Level 5
Height 5
0,1
start
0
1
Figure: NFA

Language of an NFA

- A string w is accepted by an NFA if $\hat{\delta}(q_0, w)$ contains at least one final state
- Formally, $L(N)=\{w \mid w \in \Sigma^* \text{ and } \hat{\delta}(q_0, w) \cap F \neq \phi\}$
- The language of the NFA is the set of strings it accepts.
- For example , the language of NFA described in previous slide is set of all strings of {0,1} ending with 01.