Automata

Introduction to Automata

- **Symbols** are generally letters, digits and operators, etc. Example: 0 – 9, a – z, A – Z, *, +, –, ….

- An **Alphabet** is a finite, nonempty set of symbols and is denoted as 'Σ' (Sigma). Common alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c\}$.
3. $\Sigma = \{a, b, \ldots, z\}$, the set of all lower-case letters.

- **Powers of an alphabet**:
  - If Σ is an alphabet, we can express the set of all strings of a certain length from that alphabet by using an exponential notation.
  - We define Σ^k to be the set of strings of length k, each of whose symbols is in Σ.
  - Example: $\Sigma^0 = \{\epsilon\}$, ϵ is the only string whose length is 0.

Introduction to Automata

- If $\Sigma = \{0, 1\}$, then $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$,
- $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$ and so on.

- **String** is a finite sequence of symbols chosen from some alphabet.
  Example: w = abc – is a string from alphabet $\Sigma = \{a, b, c\}$.

- **Length of the string**: The number of symbols in the string is called the “length of the string”. It is denoted by $|w|$.
  Example: w = abc, $|w| = 3$.

- **Null string**: A string of length zero is called “null string”. It is denoted as ϵ (Epsilon). $|\epsilon| = 0$

Introduction to Automata

- **Reverse of a string**: The string obtained by writing the symbols of the string in reverse order.
  Example: $w = abc$, $w^r = cba$.

- **Palindrome**: The string and its reversal are equal then the string is called a "palindrome".
  If $w = w^r$, w is called palindrome.

- **Concatenation**: Writing one string followed by other is called "concatenation".
  Example: $w_1 = abc$, $w_2 = xyz$.
  $w_1 . w_2 = abcxyz$, '.' is called concatenating symbol.

- **Prefix**: All the leading symbols of a given string are called prefixes.
  Example: $w = abc$, prefixes are ϵ, a, ab, abc.

Introduction to Automata

- **Suffix**: All the trailing symbols of a given string are called suffixes.
  Example: w = abc, suffixes are ϵ, c, bc, abc.

- **Equivalence of strings**: Two strings are said to be equivalent if they have the same sequence of symbols.
  Example: w_1 = abc, w_2 = acb, w_3 = cba, w_4 = abc.
  $w_1 \equiv w_4$, $w_1 \neq w_2$, $w_1 \neq w_3$.

- **Proper prefix**: The prefix of a string which is not equal to the string itself is called “proper prefix”.
  Example: w = abc, prefixes are ϵ, a, ab.

- **Proper suffix**: The suffix of a string which is not equal to the string itself is called “proper suffix”.
  Example: w = abc, suffixes are ϵ, c, bc.

Introduction to Automata

- **Language**: The set of strings defined over a specific alphabet is called a "language". If that set is finite then it is a finite language, otherwise it is called infinite language.

  Example: $\Sigma = \{a, b\}$
  $L_1 = \{ x \mid |x| = 2, x \in \Sigma^* \}$ (strings of length 2)
  $L_1 = \{aa, bb, ab, ba\}$ It is a finite language.

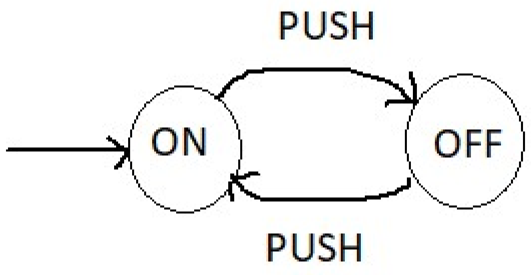
  $L_2 = \{ x \mid |x| \text{ is even}, x \in \Sigma^* \}$ (strings of length even)
  $L_2 = \{aa, ab, ba, bb, aaaa, bbbb, \ldots\}$ It is a infinite language.

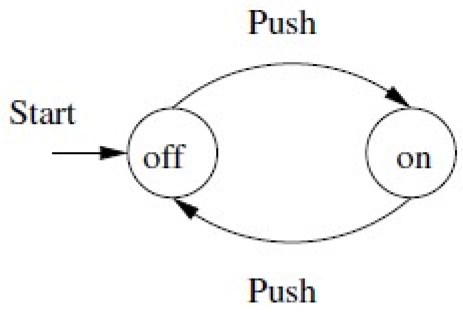
Introduction to Automata

- **Kleene's Closure (Σ^*)**: The set of all possible strings which are defined over an alphabet Σ is called Kleene's closure. It is denoted as Σ^*.
  $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

- **Positive Closure (Σ^+)**: The set of all possible strings which are defined over an alphabet Σ except ϵ (Σ^0)is called Positive closure. It is denoted as Σ^+.
  $\Sigma^+ = \Sigma^* - \{\epsilon\}$.

- **Finite state system**: Any system with finite number of states is called "finite state system".





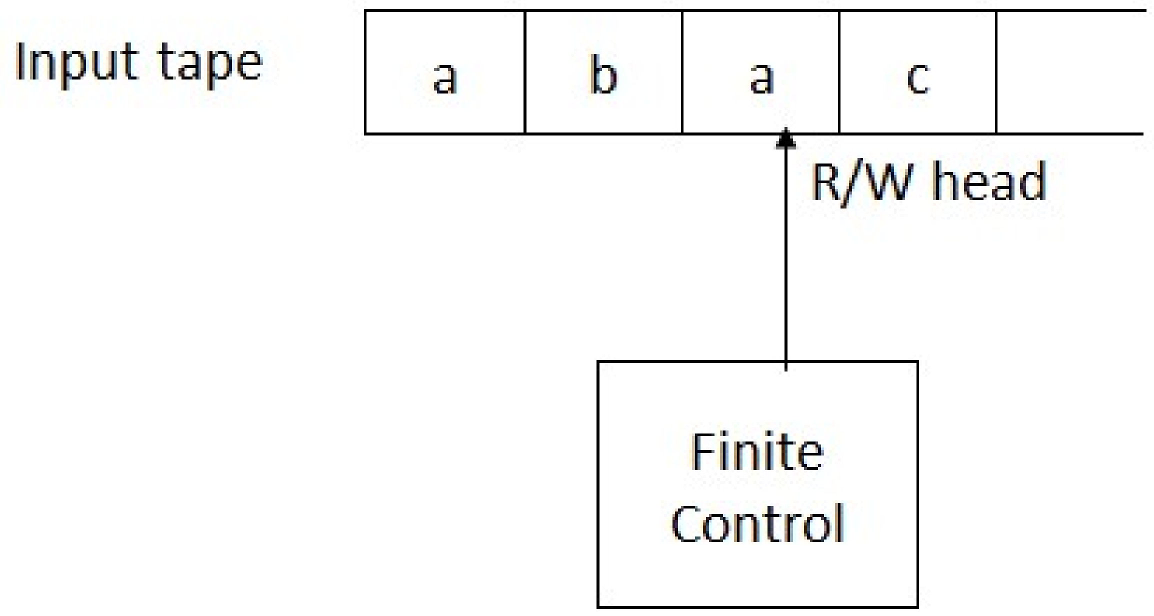
FINITE AUTOMATA

- An automaton is something which works automatically (self acting). "Automata" is a plural form of 'Automaton'.

- **Finite Automata**: The mathematical model of finite automata (FA) is given as that it consists of an input string, R/W head, finite control.

1. Input tape consists of symbols.
2. R/W head for reading.
3. Finite control for transformations and changing states.

FINITE AUTOMATA



- It is viewed as simulation of a computer.

Finite Automata

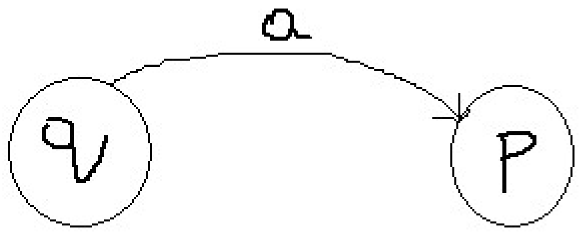
- **Formal definition of Finite Automata**: A finite automata is a 5 tuple. It is the machine
  $M = (Q, \Sigma, \delta, q_0, F)$
  where

  - Q – Non empty set of finite number of states.
  - Σ – Non empty set of finite number of symbols or Finite input alphabet.
  - δ – State transition function, defined as
    $\delta: Q \times \Sigma \rightarrow Q$ is unique.
    Q – current state, **Q – next state**.
  - q_0 – It is initial or start state, $q_0 \in Q$.
  - F – $F \subseteq Q$, It is set of Final or Accepting states.

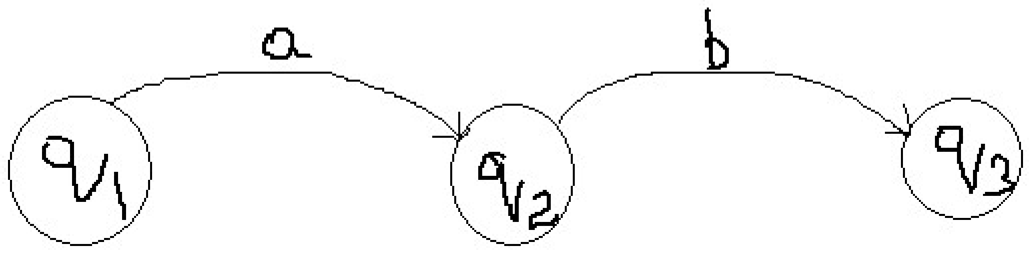
- Note: **Number of transitions in DFA = Number of symbols × Number of states**.

FINITE AUTOMATA

- The finite automata contains finite number of states, finite number of transitions which occur by applying an input symbol on a state.
- One state is designated as initial state and some states are designated as final states.
- Example:
  $\delta(q, a) = p$



- The finite automata in state 'q' on taking an input 'a' enters into a state 'p'.



FINITE AUTOMATA

- **Acceptance of a string by Finite Automata**: A string x is said to be accepted by Finite Automata. If the finite automata of state 'q_0' on processing every symbol of 'x' reaches one of the accepting states.

  $\hat{\delta}(q_0, x) \in F$, $\hat{\delta}$ is extended transition function

  x is said to be accepted by q_0.

- **Language accepted by Finite Automata**: The collection of all the strings accepted by a Finite Automata is called the language accepted by Finite Automata.
- $L(m) = \{x \mid \delta(q_0, x) \in F\}$

- Note: The language accepted by Finite Automata is called Regular Language.

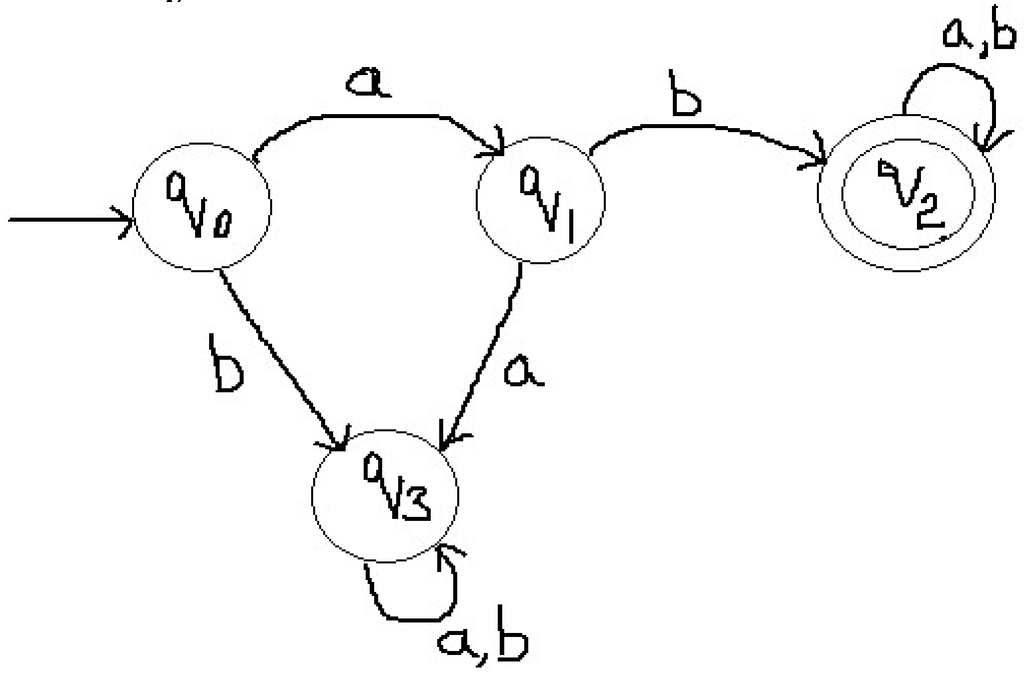
FINITE AUTOMATA

- There are 3 ways of representing transitions in **Finite Automata**:

  1. Transition diagram (TD)
  2. Transition table (TT)
  3. Transition function (TF)

Finite Automata

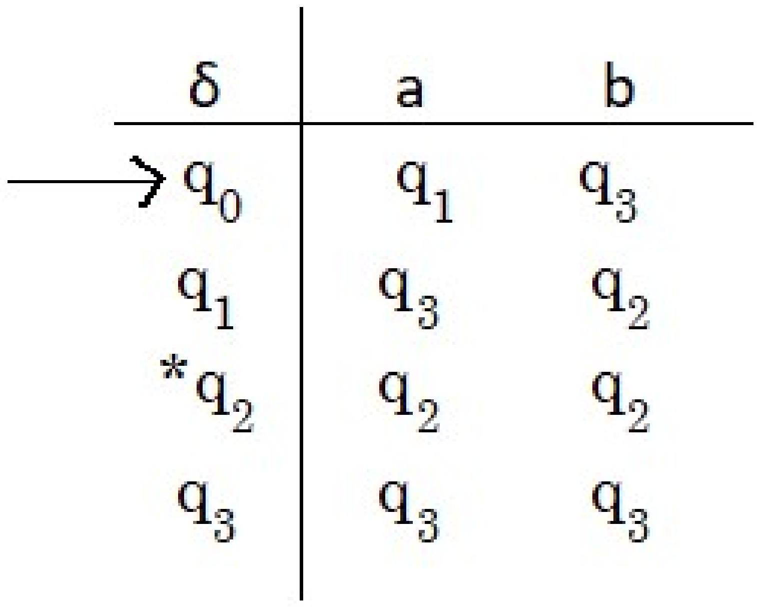
- **Transition diagram**: It is a directed graph in which the nodes in the graph represent the state. The transitions are represented by edges. The labels on edges are input symbols. The initial state is represented by pointing an arrow head to state. The final states are represented by concentric circles.



- Here, Finite Automata is given as
- **M = ({q_0, q_1, q_2, q_3,} {a, b}, δ, q_0, {q_2})**

FINITE AUTOMATA

- **Transition table**:



δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_3	q_2
$*q_2$	q_2	q_2
q_3	q_3	q_3

- Here, Finite Automata is given as
  $M = (\{q_0, q_1, q_2, q_3,\} \{a, b\}, \delta, q_0, \{q_2\})$

Finite Automata

- **Transition function**:

$\delta(q_0, a) = q_1$	$\delta(q_0, b) = q_3$
$\delta(q_1, a) = q_3$	$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_2$	$\delta(q_2, b) = q_2$
$\delta(q_3, a) = q_3$	$\delta(q_3, b) = q_3$

- Here, Finite Automata is given as
  $M = (\{q_0, q_1, q_2, q_3,\} \{a, b\}, \delta, q_0, \{q_2\})$

Finite Automata

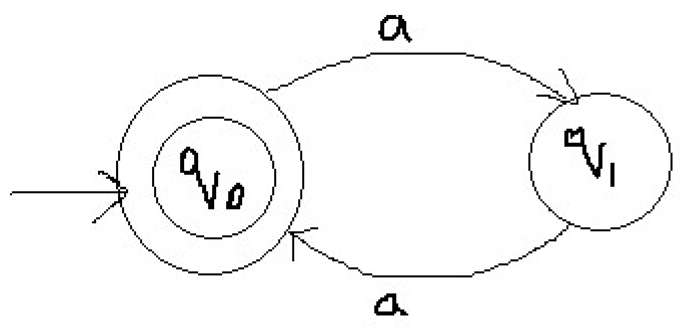
- **Finite Automata is of 3 forms**:

1. Deterministic Finite Automata (DFA)
2. Non deterministic Finite Automata (NFA)
3. ϵ – Non deterministic Finite Automata (ϵ – NFA)

Deterministic Finite Automata

- **DFA**: A finite automata in which every input symbol is applied on every state and is applied exactly once.

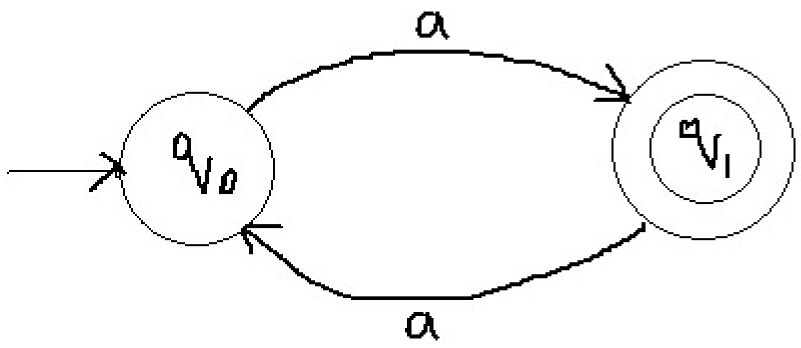
- **Example 1**: Design a finite automata for the language containing **even no. of 'a' s** where $\Sigma = \{a\}$.



- $L = \{ a^n \mid n \text{ is even} \}$

DETERMINISTIC FINITE AUTOMATA

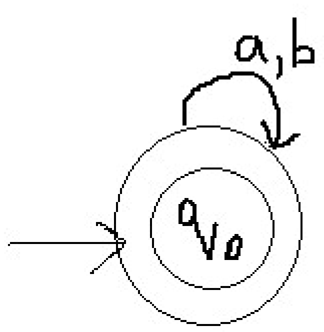
- **Example 2**: Design a finite automata for the language containing **odd no. of 'a' s** where Σ = {a}



- L = { a^n | n is odd }

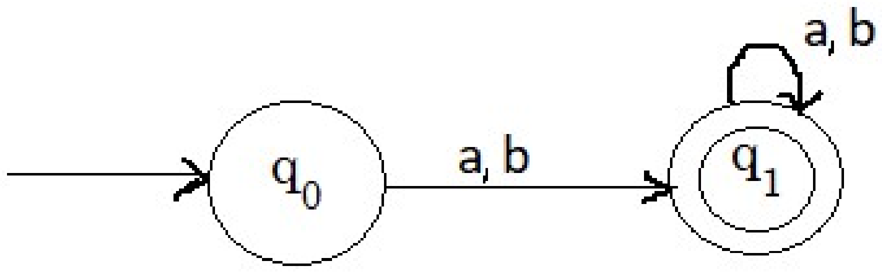
Deterministic Finite Automata

- **Example 3**: Let $\Sigma = \{a, b\}$. The finite automata for Σ^* is



Deterministic Finite Automata

- **Example 3**: Let $\Sigma = \{a, b\}$. The finite automata for Σ^+ is



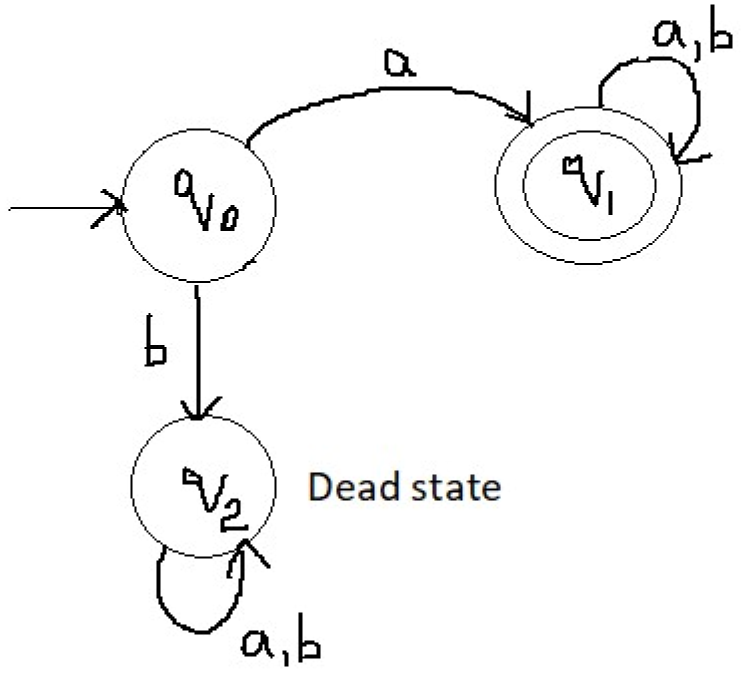
Deterministic Finite Automata

- Note: Whenever there is **constraint on the start state** we need to **take a dead state** to draw a finite automata.

- **Dead state**: A non final state is called a dead state, if there is no transition from a state to other states for any input symbol.

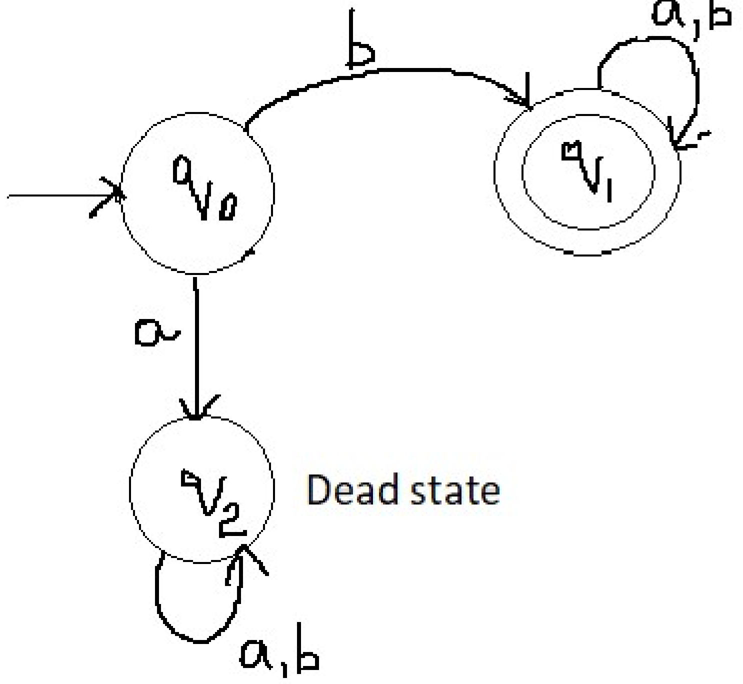
Deterministic Finite Automata

- **Example 4**: Let Σ = {a, b}. Design a finite automata for the language containing strings begin with 'a'.



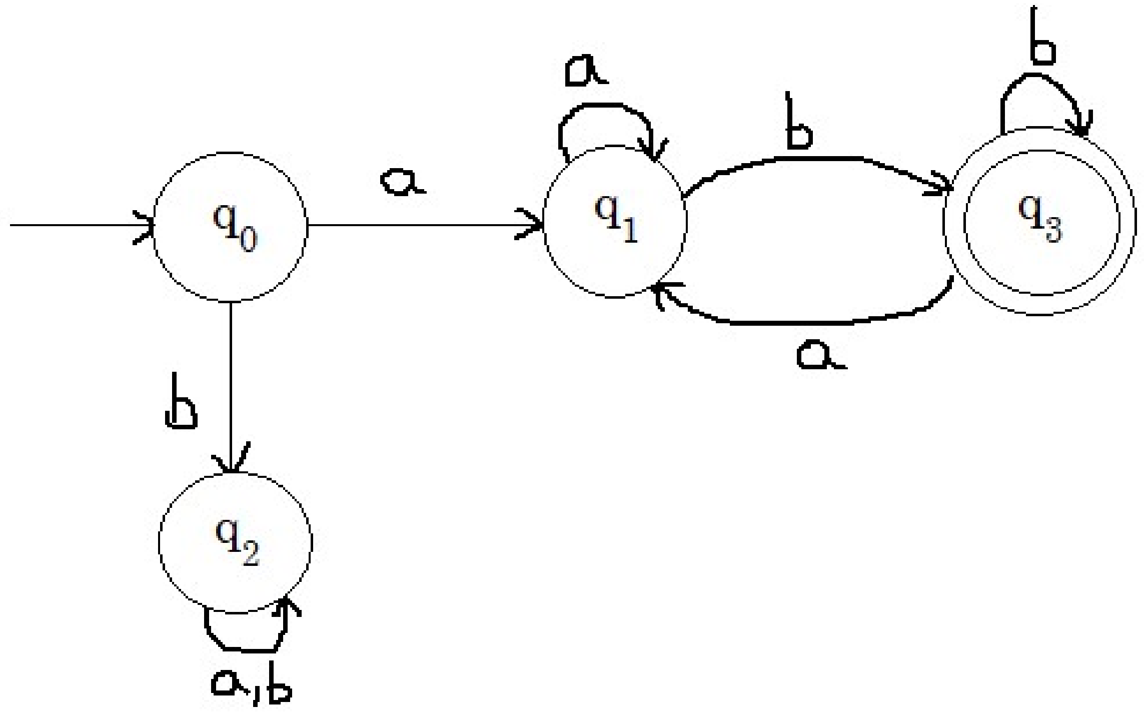
Deterministic Finite Automata

- **Example 5**: Let Σ = {a, b}. Design a finite automata for the language containing strings begin with 'b'.



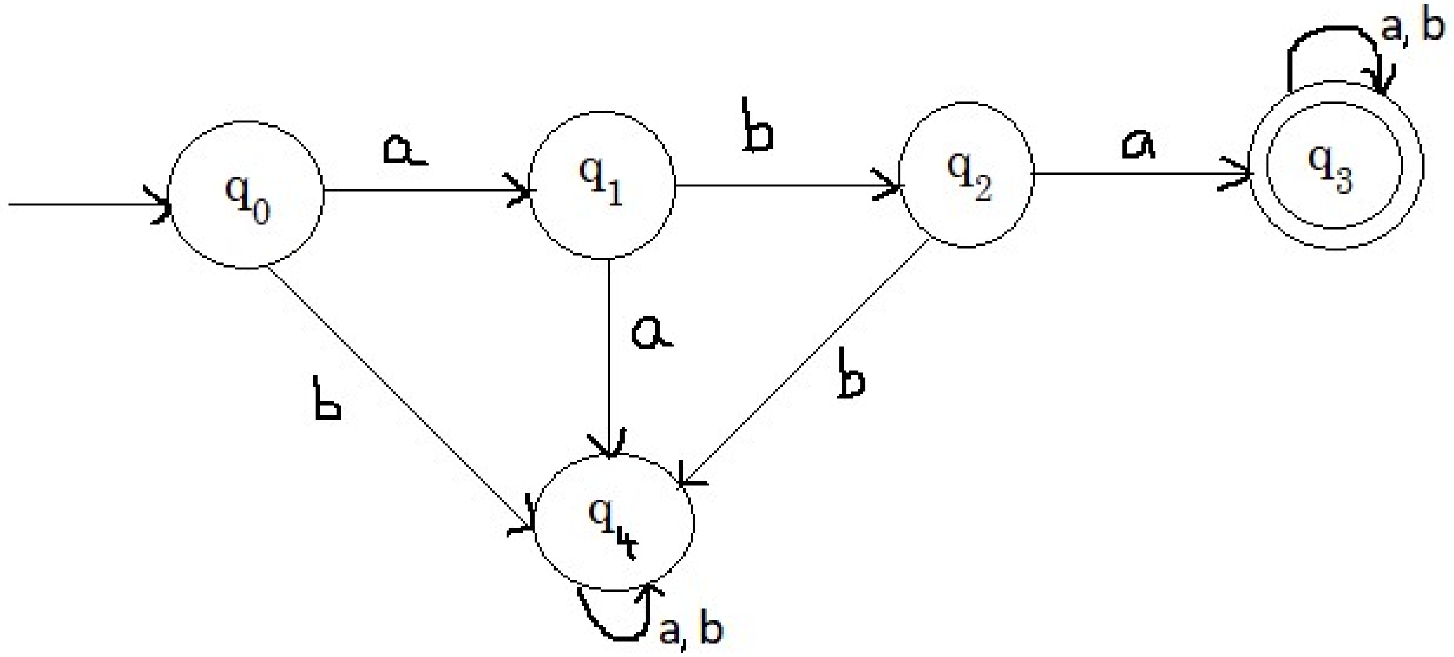
DETERMINISTIC FINITE AUTOMATA

- **Example 6**: Let Σ = {a, b}. Design a finite automata for the language containing strings begin with 'a' and end with 'b'.



DETERMINISTIC FINITE AUTOMATA

- **Example 7**: Let Σ = {a, b}. Design a finite automata for the language containing strings begin with 'aba'.



Deterministic Finite Automata

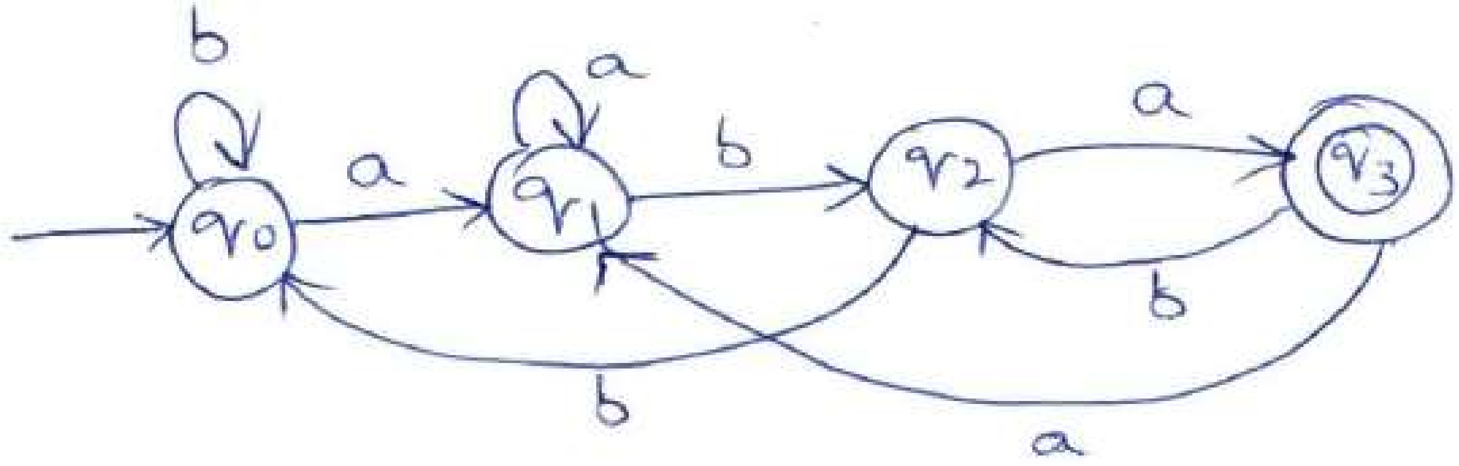
- The DFA for the given language is $M = (Q, \Sigma, \delta, q_0, F)$
  where $Q = \{q_0, q_1, q_2, q_3, q_4\}$
  $\Sigma = \{a, b\}$
  δ: shown in transition diagram
  q_0 = Initial state
  $F = \{q_3\}$
- Example: Let w = abab

$$\begin{aligned} \delta(q_0, abab) &= \delta(\delta(q_0, a), bab) \\ &= \delta(q_1, bab) \\ &= \delta(\delta(q_1, b), ab) \\ &= \delta(q_2, ab) \\ &= \delta(\delta(q_2, a), b) \\ &= \delta(q_3, b) \\ &= q_3 \end{aligned}$$

Hence the string is accepted.

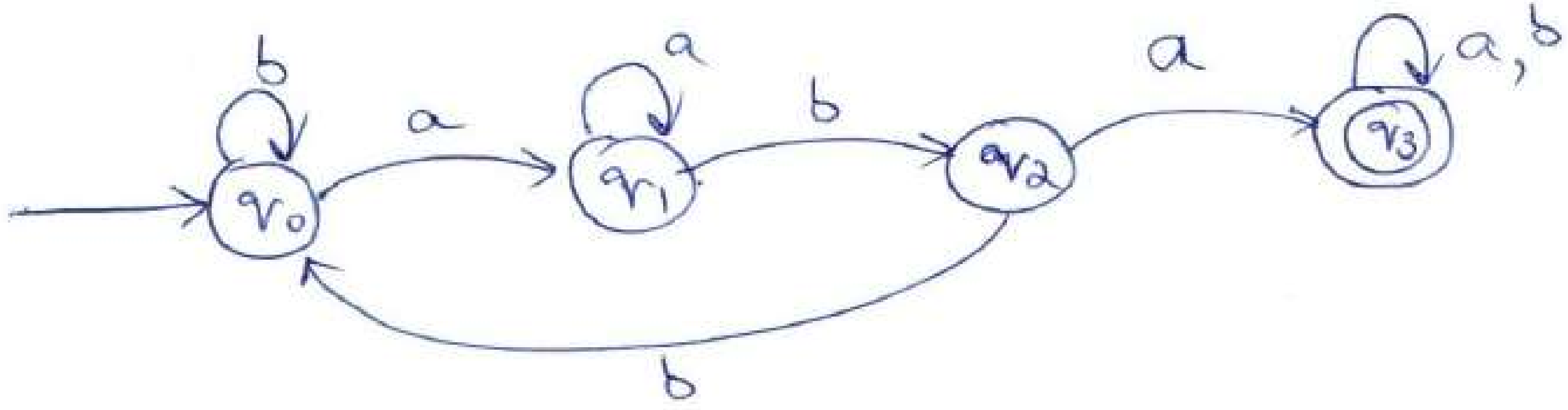
Deterministic Finite Automata

- **Example 8**: Let Σ = {a, b}. Design a finite automata for the language containing strings ending with 'aba'.



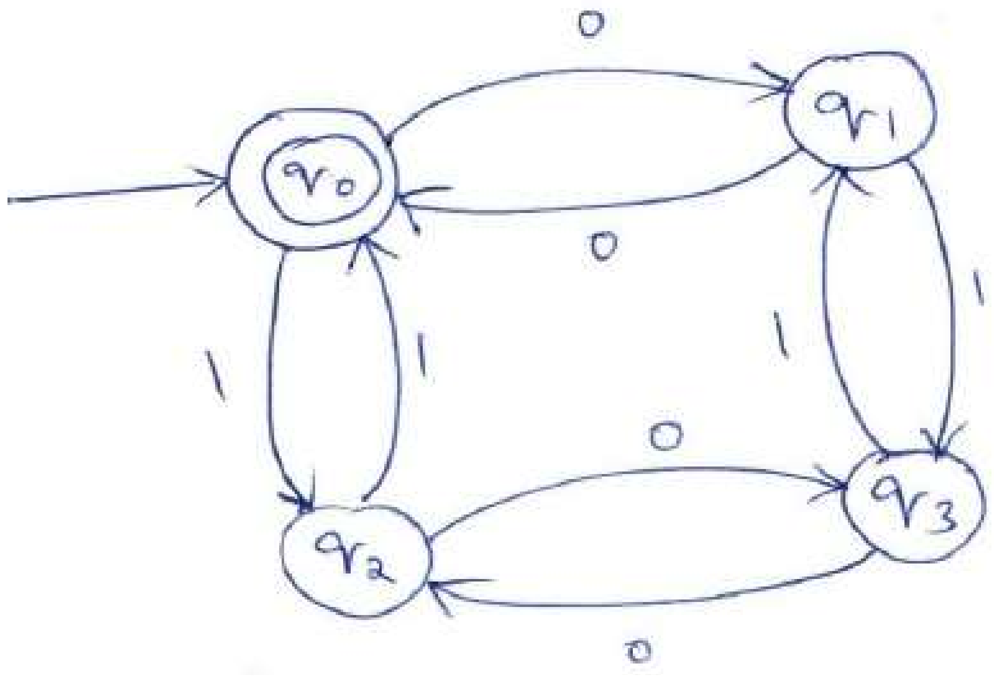
Deterministic Finite Automata

- **Example 9**: Let Σ = {a, b}. Design a finite automata for the language containing strings for 'aba' as substring.



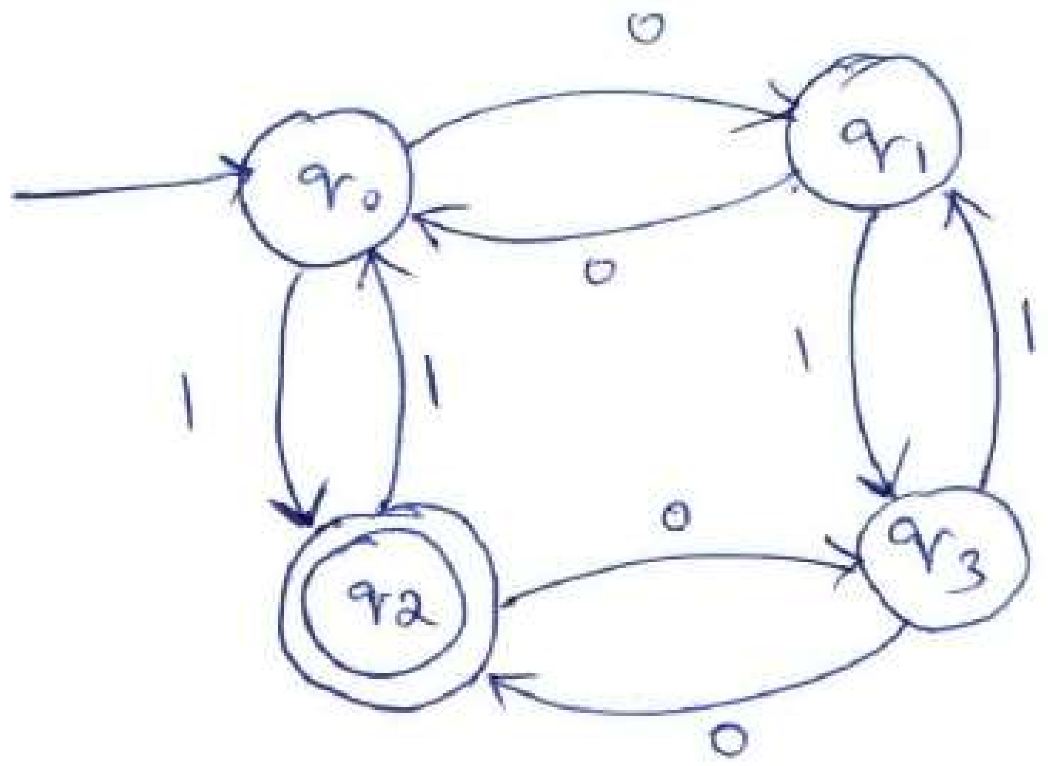
Deterministic Finite Automata

- **Example 10**: Let Σ = {0, 1}. Design a finite automata for the language containing
  a) Even no. of 0s & even no. of 1s.



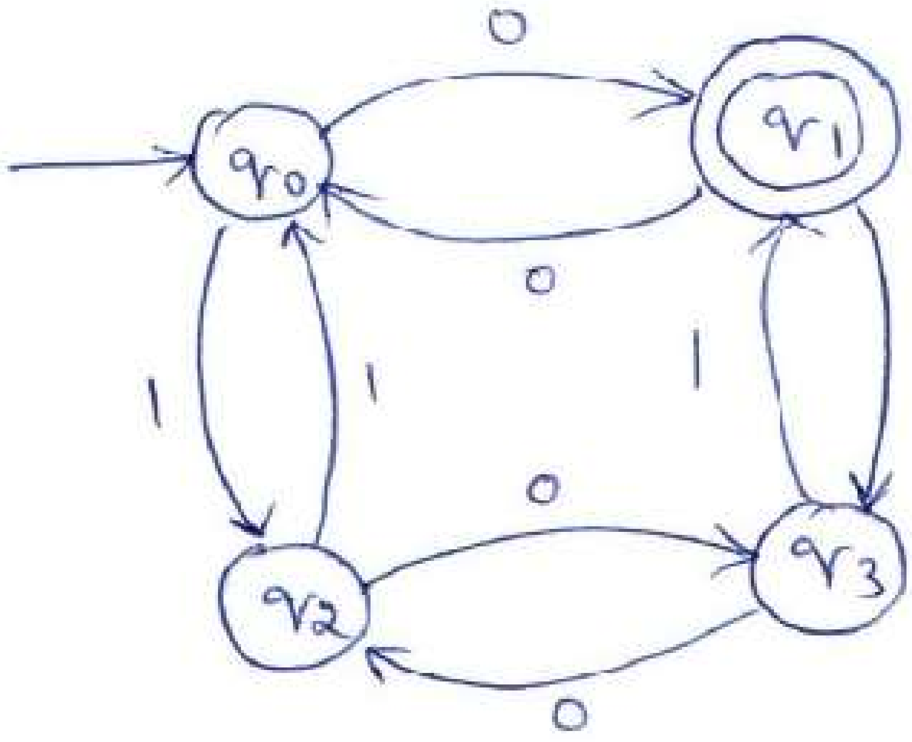
Deterministic Finite Automata

b) Even 0s & odd 1s



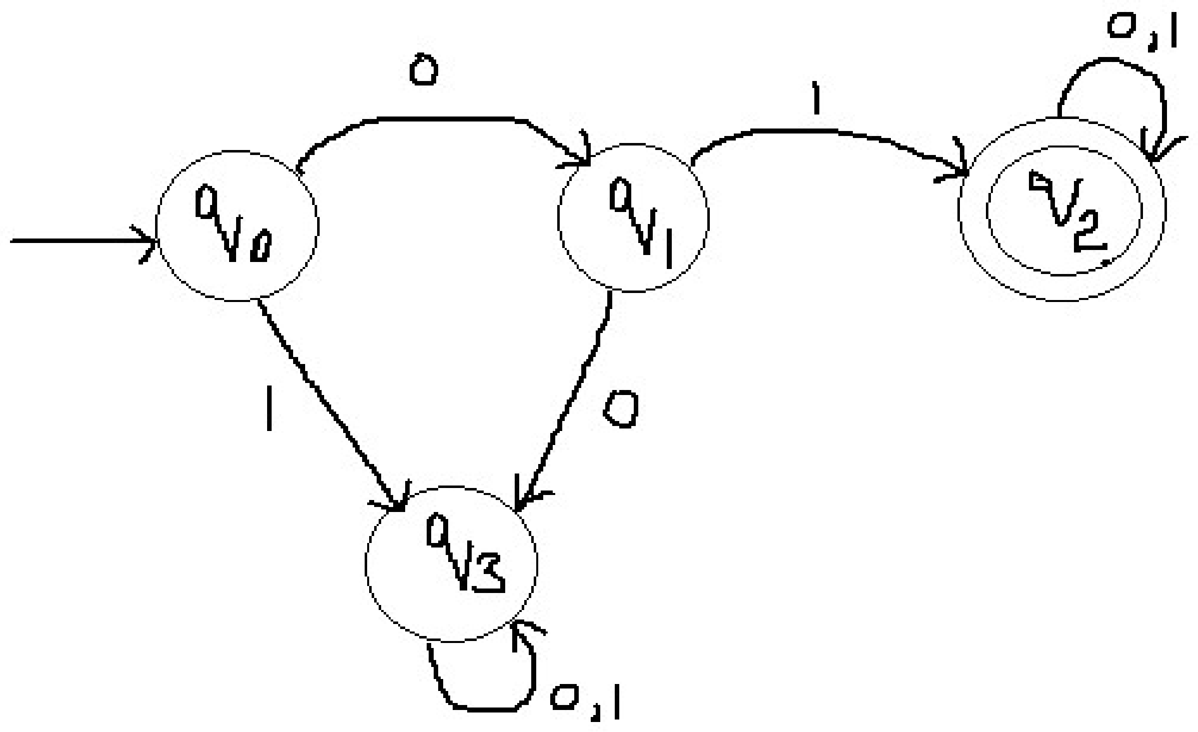
Deterministic Finite Automata

c) Odd 0s & Even 1s


$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, bend left, "0"] \arrow[d, bend right, "1"'] & q_1 \arrow[l, bend left, "0"] \arrow[d, bend left, "1"] \\
& q_2 \arrow[u, bend right, "1"'] \arrow[r, bend left, "0"] & q_3 \arrow[l, bend left, "0"] \arrow[u, bend left, "1"]
\end{tikzcd}$$

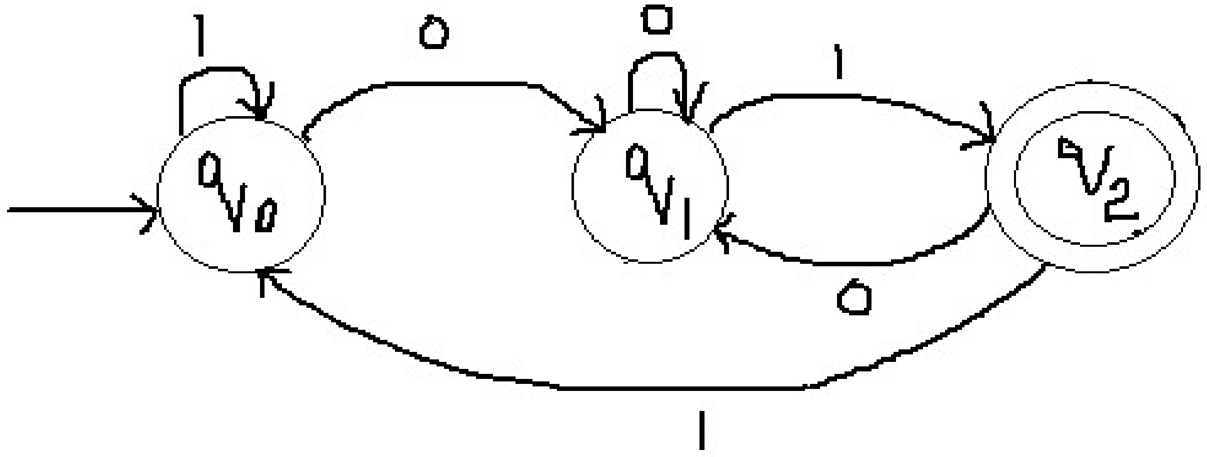
DETERMINISTIC FINITE AUTOMATA

- **Example 11**: Let Σ = {0, 1}. Design a finite automata for language containing strings starts with '01'.



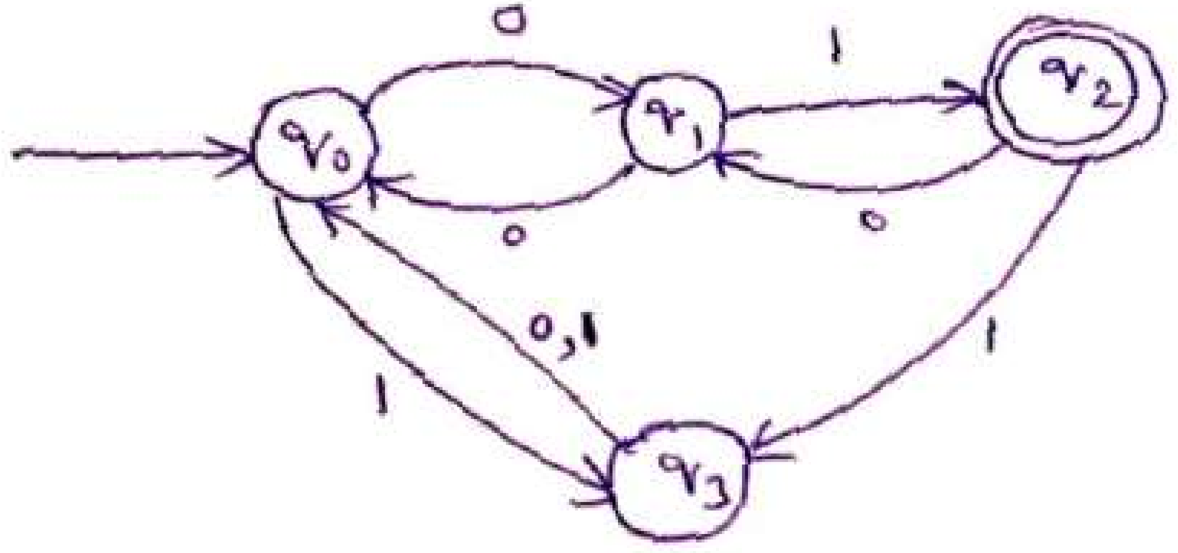
Deterministic Finite Automata

- **Example 12**: Let Σ = {0, 1}. Design a finite automata for language containing strings ends with '01'.



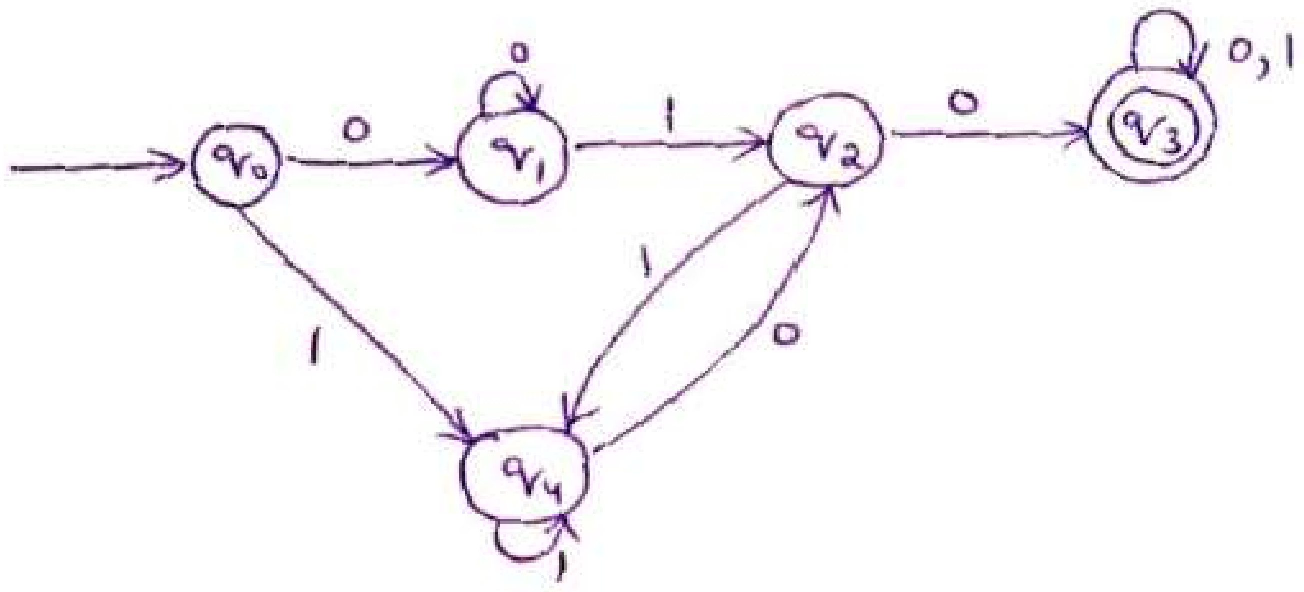
DETERMINISTIC FINITE AUTOMATA

- **Example 13**: Let Σ = {0, 1}. Design a finite automata for language containing strings ends with '01' and even in length.



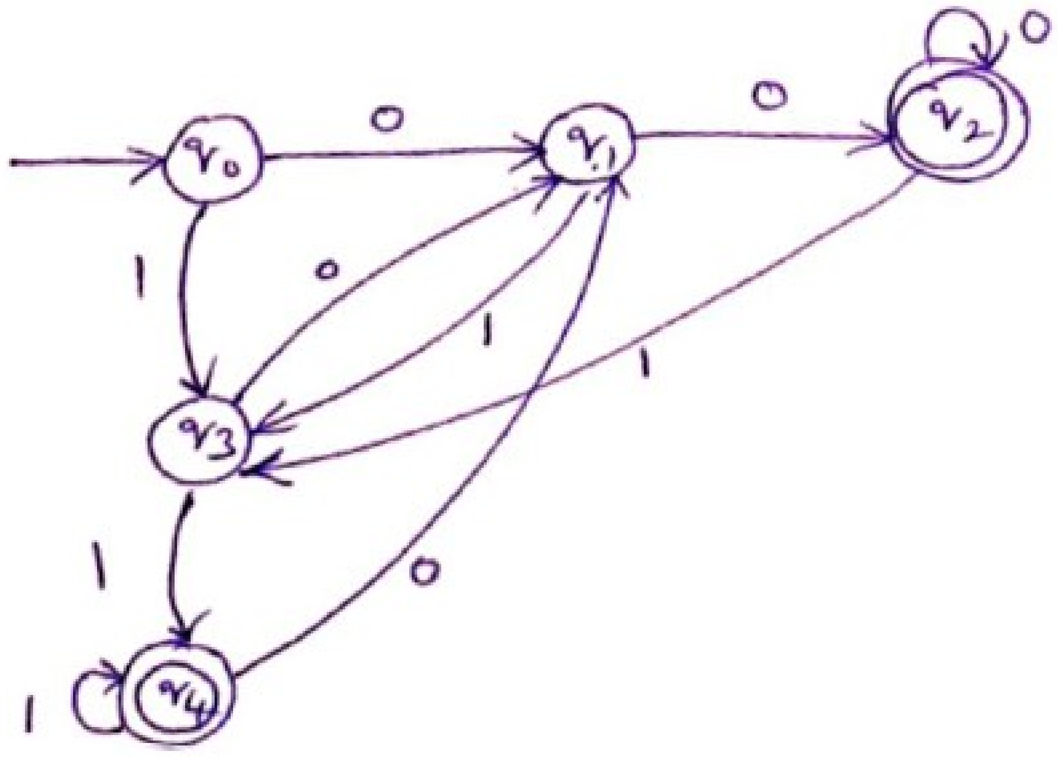
Deterministic Finite Automata

- **Example 14**: Let Σ = {0, 1}. Design a finite automata for language containing strings having substring 010 or 100.



DETERMINISTIC FINITE AUTOMATA

- **Example 14**: Let Σ = {0, 1}. Design a finite automata for language containing strings end with 00 or 11.



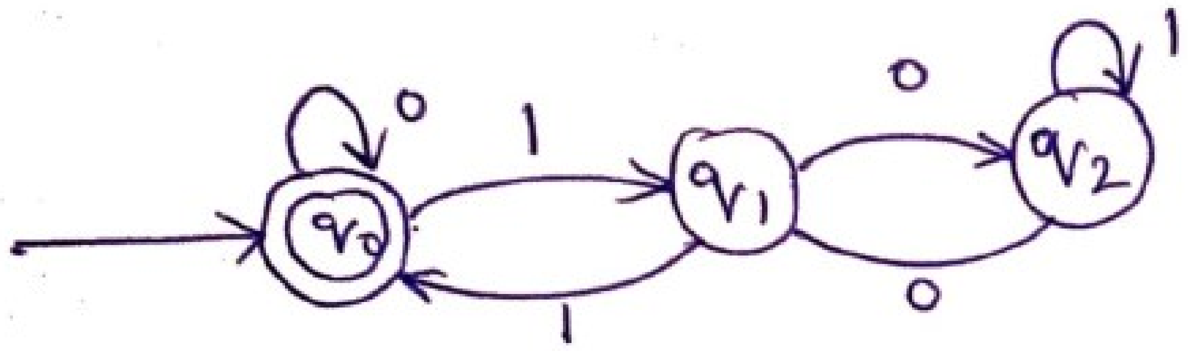
Deterministic Finite Automata

- **Example 15**: Let Σ = {0, 1}. Design a finite automata for language which contain strings that are divisible by 3 when the strings are interpreted as binary numbers.

No.	Binary value	Reminder when mod 3	State end
0	000	0	q_0
1	001	1	q_1
2	010	2	q_2
3	011	0	q_0
4	100	1	q_1
5	101	2	q_2
6	110	0	q_0
7	111	1	q_1

Deterministic Finite Automata

- **Example 15**:



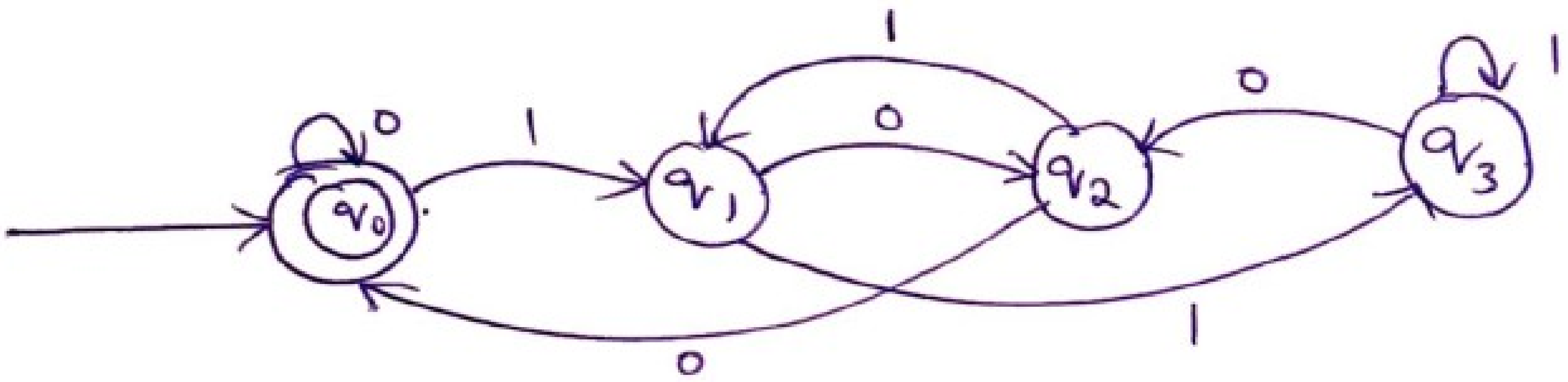
DETERMINISTIC FINITE AUTOMATA

- **Example 16**: Let $\Sigma = \{0, 1\}$. Design a finite automata for language which contain strings that are divisible by 4 when the strings are interpreted as binary numbers.

No.	Binary value	Reminder when mod 4	State end
0	0000	0	q_0
1	0001	1	q_1
2	0010	2	q_2
3	0011	3	q_3
4	0100	0	q_0
5	0101	1	q_1
6	0110	2	q_2
7	0111	3	q_3

DETERMINISTIC FINITE AUTOMATA

No.	Binary value	Reminder when mod 4	State end
8	1000	0	q_0
9	1001	1	q_1
10	1010	2	q_2



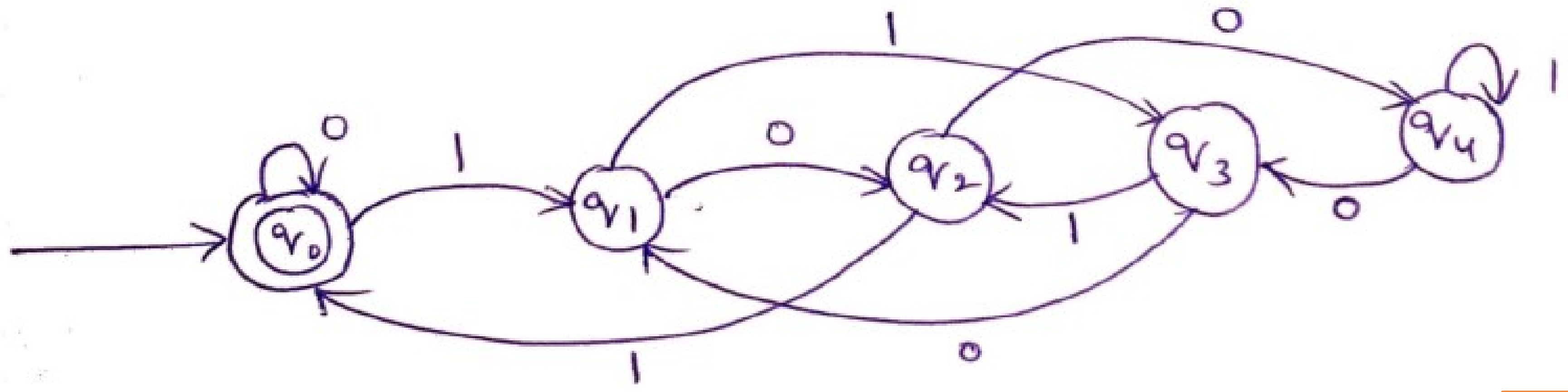
Deterministic Finite Automata

- **Example 17**: Let Σ = {0, 1}. Design a finite automata for language which contain strings that are divisible by 5 when the strings are interpreted as binary numbers.

No.	Binary value	Reminder when mod 5	State end
0	0000	0	q_0
1	0001	1	q_1
2	0010	2	q_2
3	0011	3	q_3
4	0100	4	q_4
5	0101	0	q_0
6	0110	1	q_1
7	0111	2	q_2

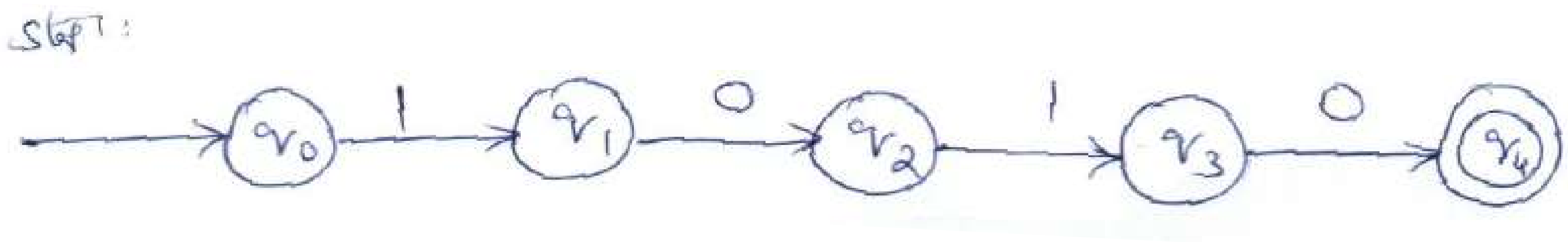
Deterministic Finite Automata

No.	Binary value	Reminder when mod 4	State end
8	1000	3	q_3
9	1001	4	q_4
10	1010	0	q_0
11	1011	1	q_1

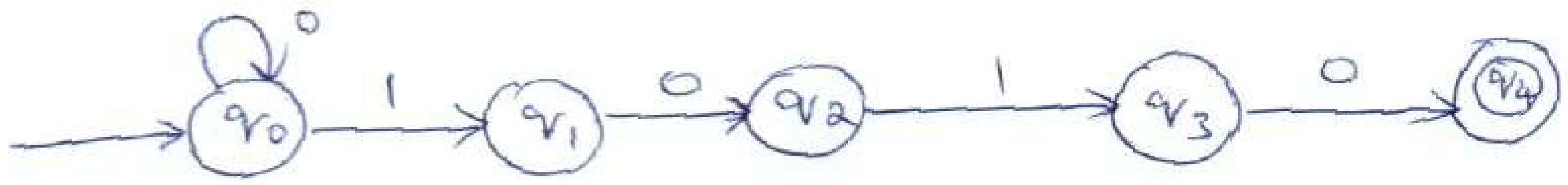


Deterministic Finite Automata

- **Example 18**: Let $\Sigma = \{0, 1\}$. Design a finite automata for language $L = \{ w \mid w \text{ contain } 1010 \}$.
- Solution:
- Step1: Find the minimum string. Here the minimum string is 1010.

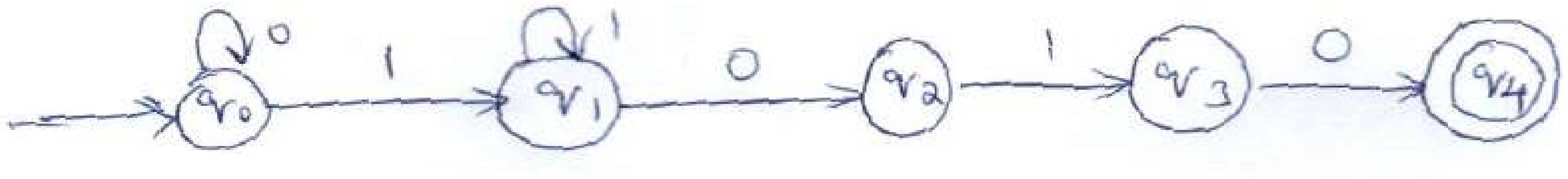


- Step2: For state q_0, apply symbol '0' then it will not lead to transition since any number of 0's at the start of string are accepted.

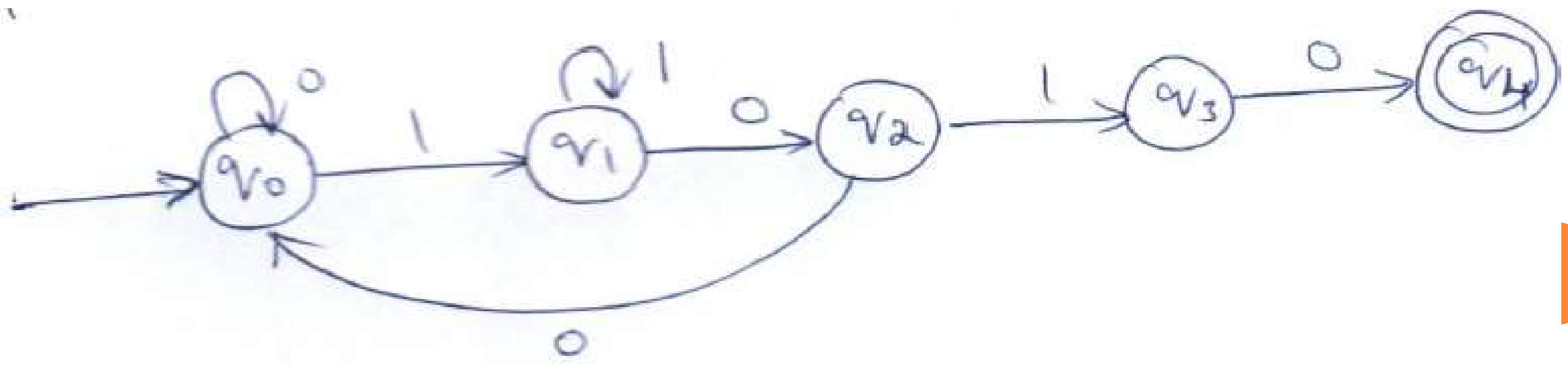


Deterministic Finite Automata

- Step3: For q1, apply symbol '1', here it will not lead to any transitions since one '1' is already obtained.

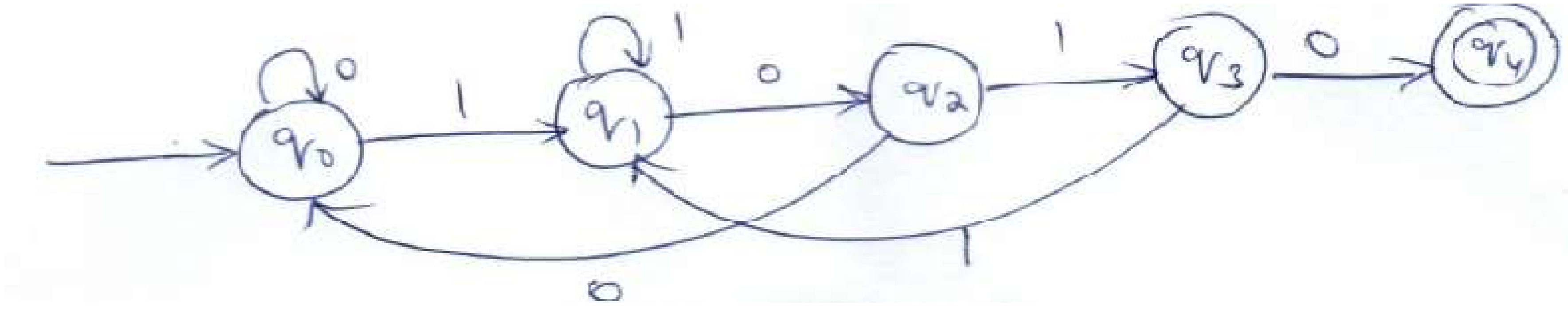


- Step4: Now apply symbol '0' on q2 then it go to the previous state q_0.
  For q_2, all transitions are completed.

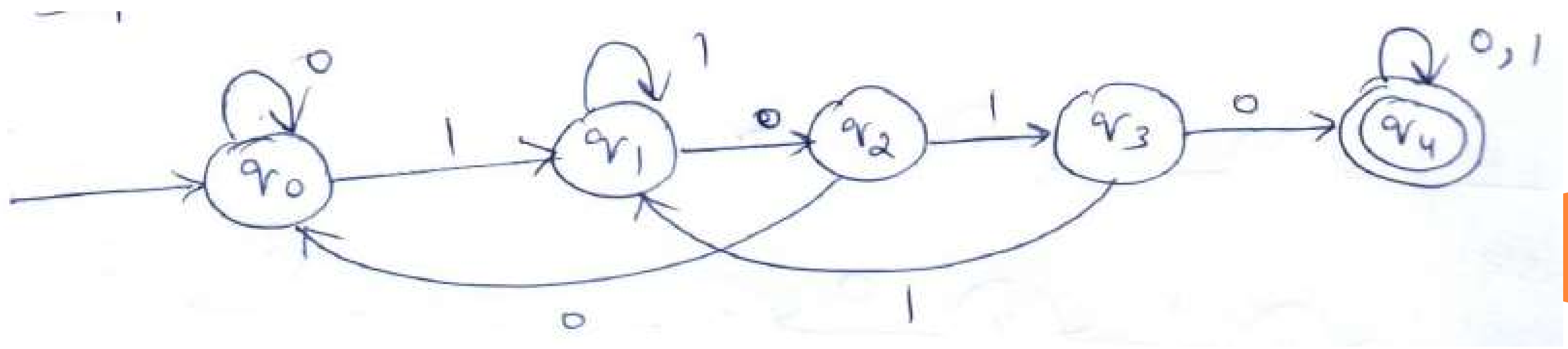


DETERMINISTIC FINITE AUTOMATA

- Step5: Apply symbol '1' on q_3, it will go to the previous state q_1 since we have already a zero at state q_1 so by applying '1' on q3, it will go to '0' on q1. For q_3, all transitions are completed.

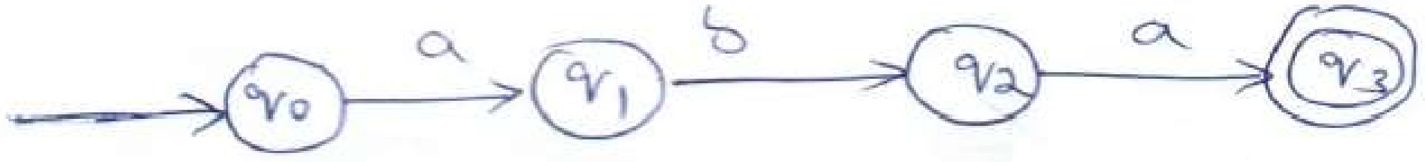


- Step6: For state q_4, since we are already in a final state, 0 and 1 can be occurred many times.



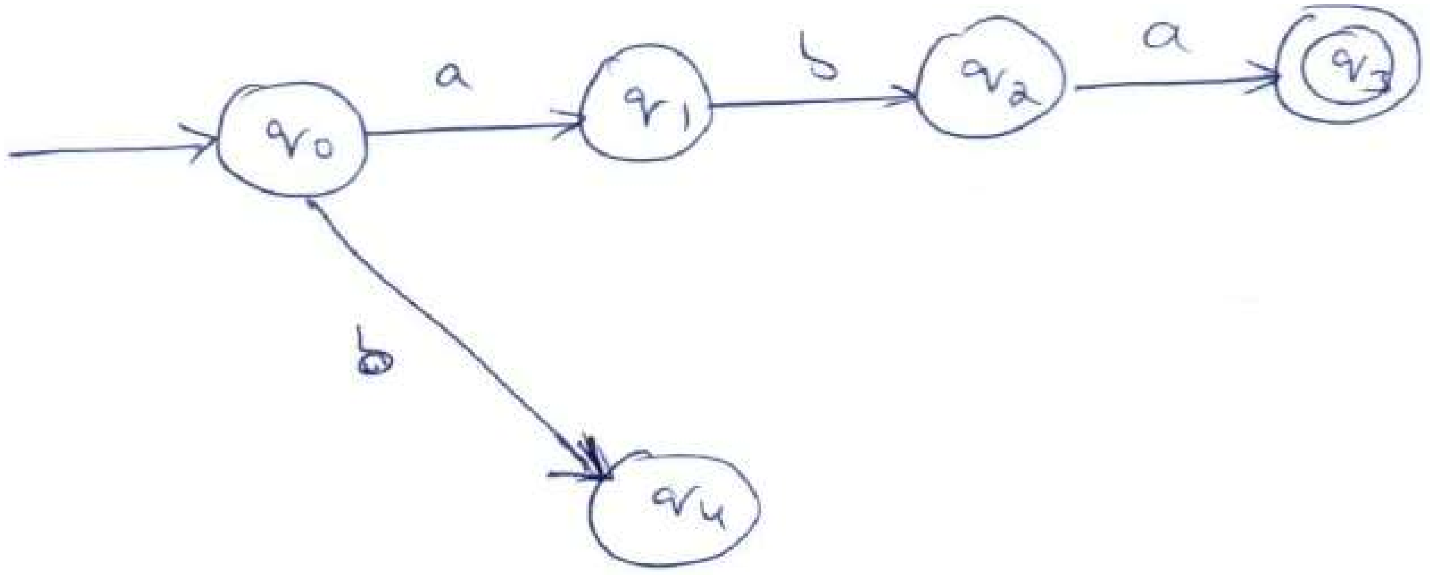
Deterministic Finite Automata

- **Example 18**: Let Σ = {a, b}. Design a finite automata for the language containing strings begin with 'aba'.
- Solution:
- Step1: Find the minimum string. Here the minimum string is aba.



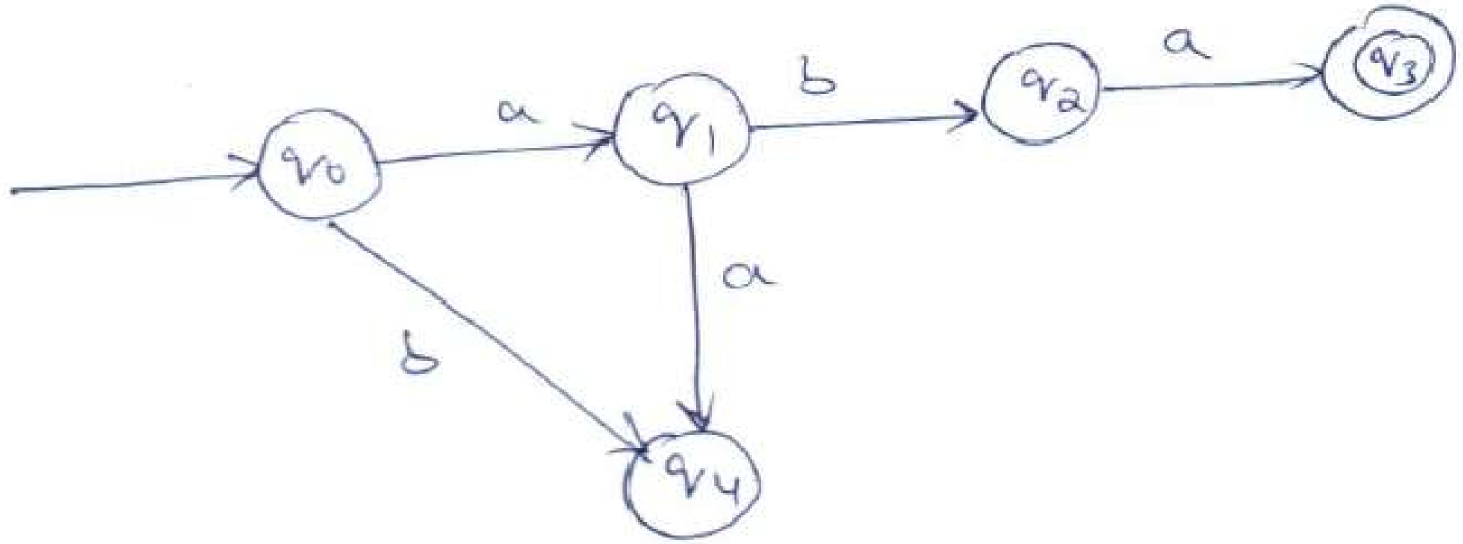
Deterministic Finite Automata

- Step2: For state q_0, the symbol 'a' is already applied. For the symbol 'b', since the string should start with 'a' by applying 'b' it goes to dead state. For q_0, all transitions are completed.



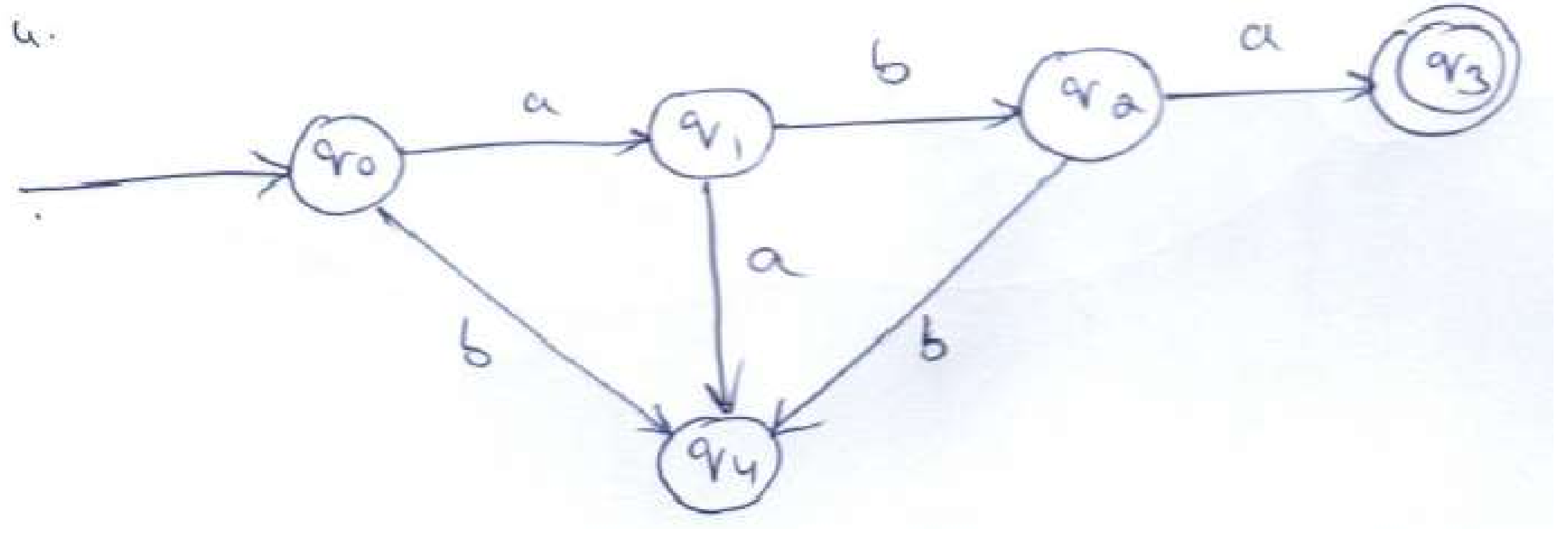
DETERMINISTIC FINITE AUTOMATA

- Step3: For state q_1, the symbol 'b' is already applied. for the symbol 'a', since the string should start with 'aba'. On applying 'a' on q_1, it will go to dead state. For q_1, all transitions are completed.



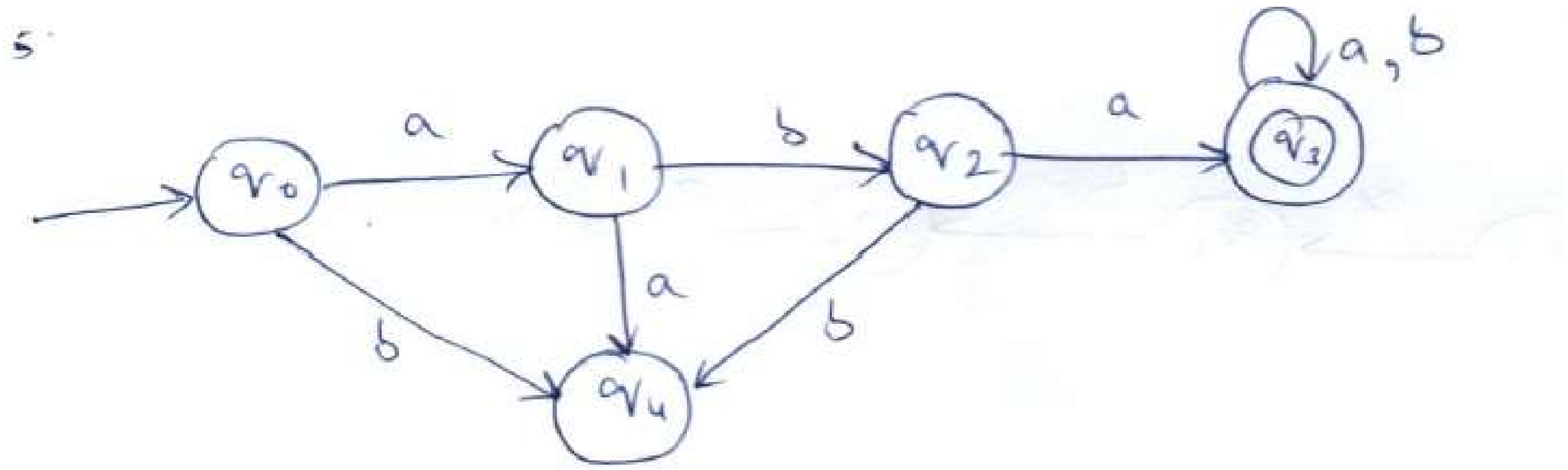
Deterministic Finite Automata

- Step4: For state q_2, the symbol 'a' is already applied. For the symbol 'b', since the string should start with 'aba' by applying 'b' it goes to dead state. For q_2, all transitions are completed.



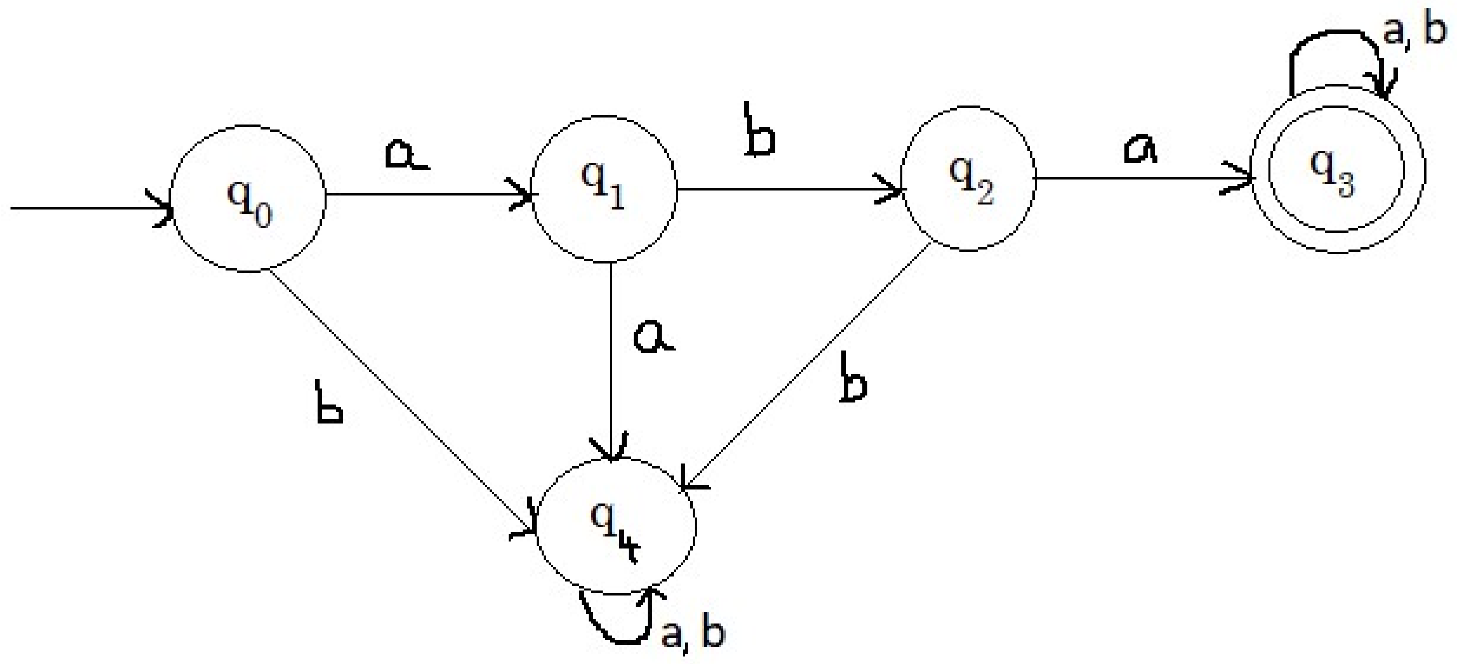
Deterministic Finite Automata

- Step5: For state q_3, since the string should start with 'aba'. Any symbol i.e., {a, b} can occur multiple times. For q_3, all transitions are completed.



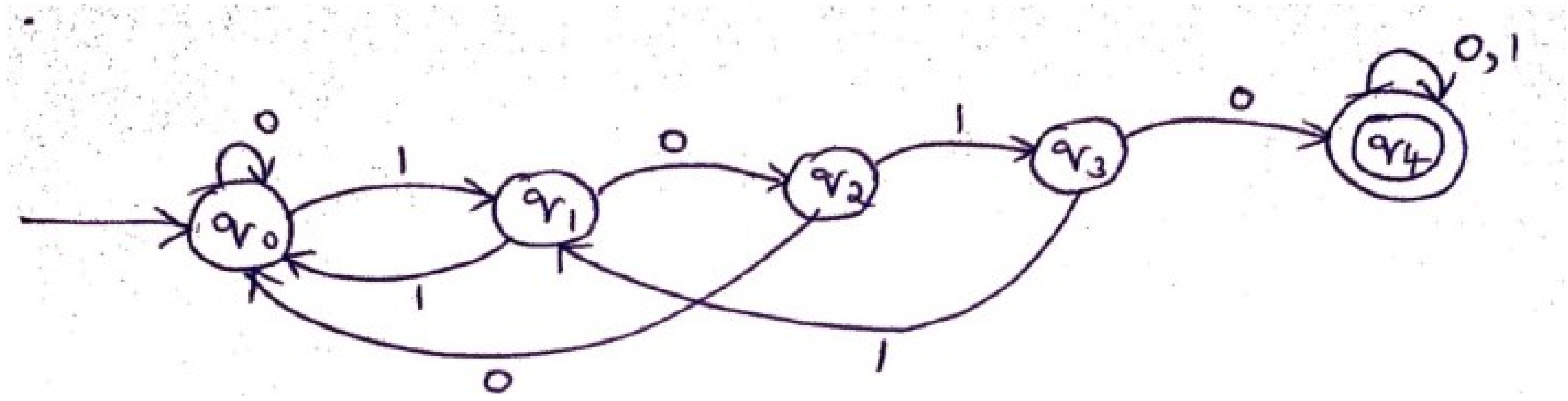
Deterministic Finite Automata

- Step6: For dead state q4, since it went to dead state {a, b} can be applied many times.
- For q_4, all transitions are completed.



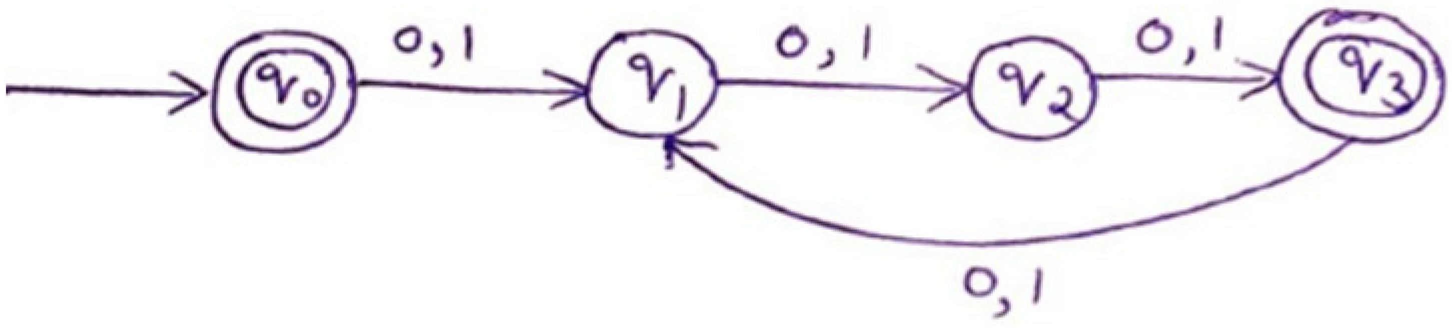
DETERMINISTIC FINITE AUTOMATA

- **Example 18**: Let Σ = {0, 1}. Design a finite automata for language L = { w | w contain 1010 }.

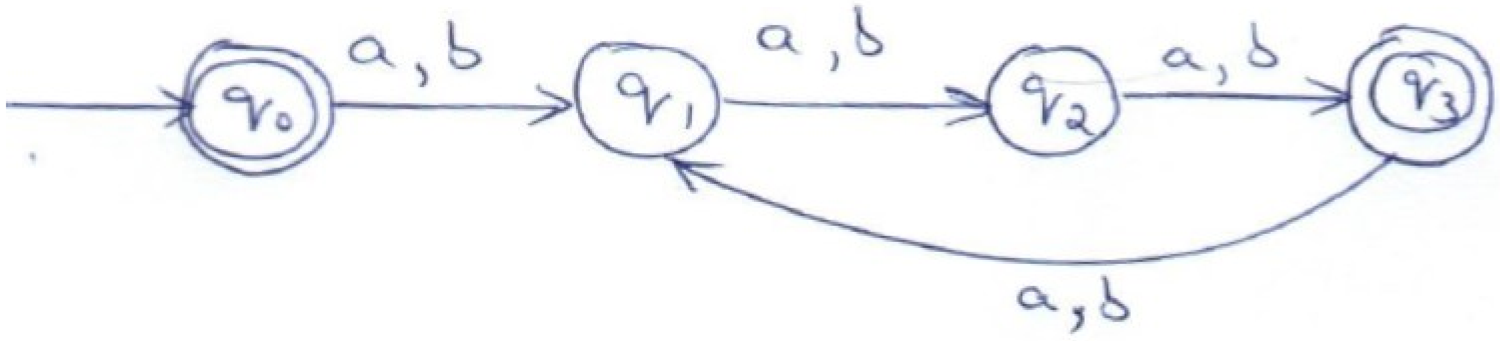


Deterministic Finite Automata

- **Example 19**: Let Σ = {0, 1}. Design a finite automata for language L = { w | |w| mod 3 = 0 }.

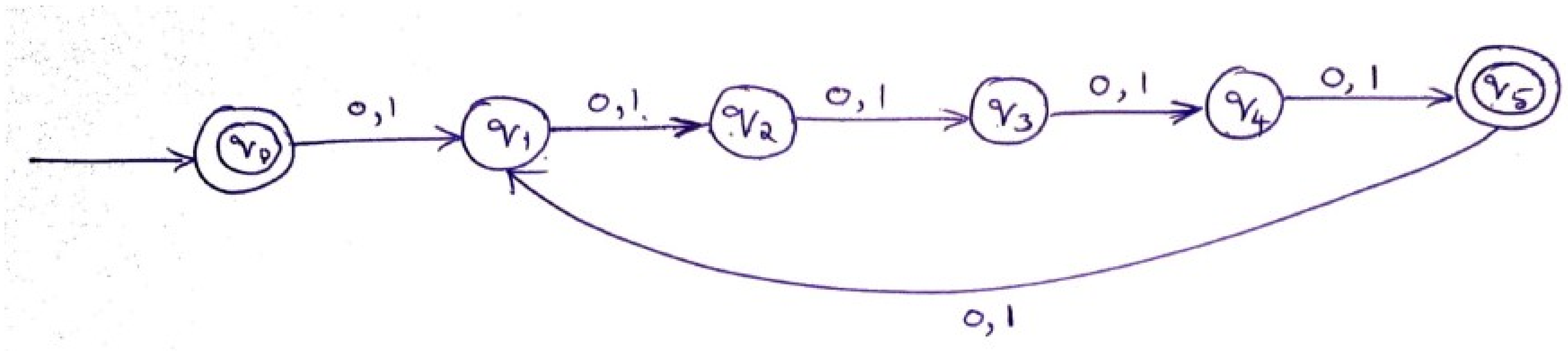


- **Example 19**: Let Σ = {a, b}. Design a finite automata for language L = { w | |w| mod 3 = 0 }.



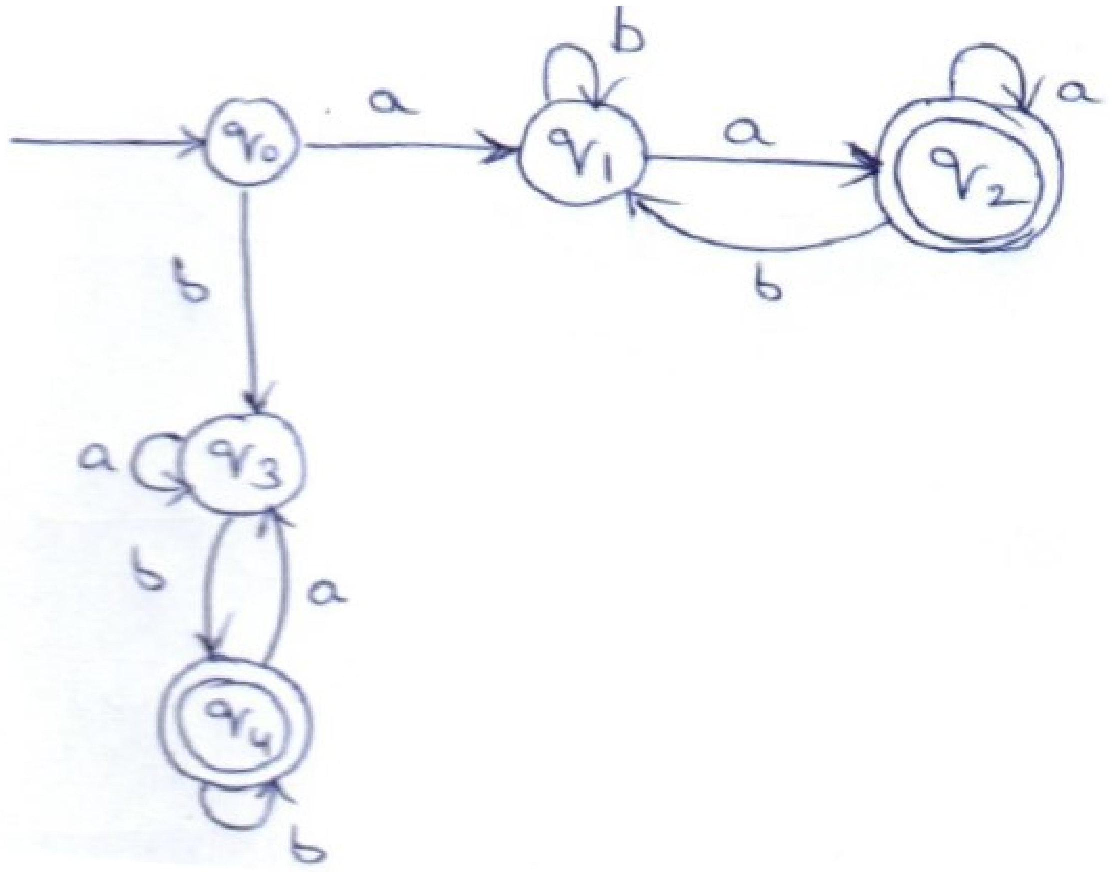
Deterministic Finite Automata

- **Example 20**: Let Σ = {0, 1}. Design a finite automata for language L = { w | |w| mod 5 = 0 }.



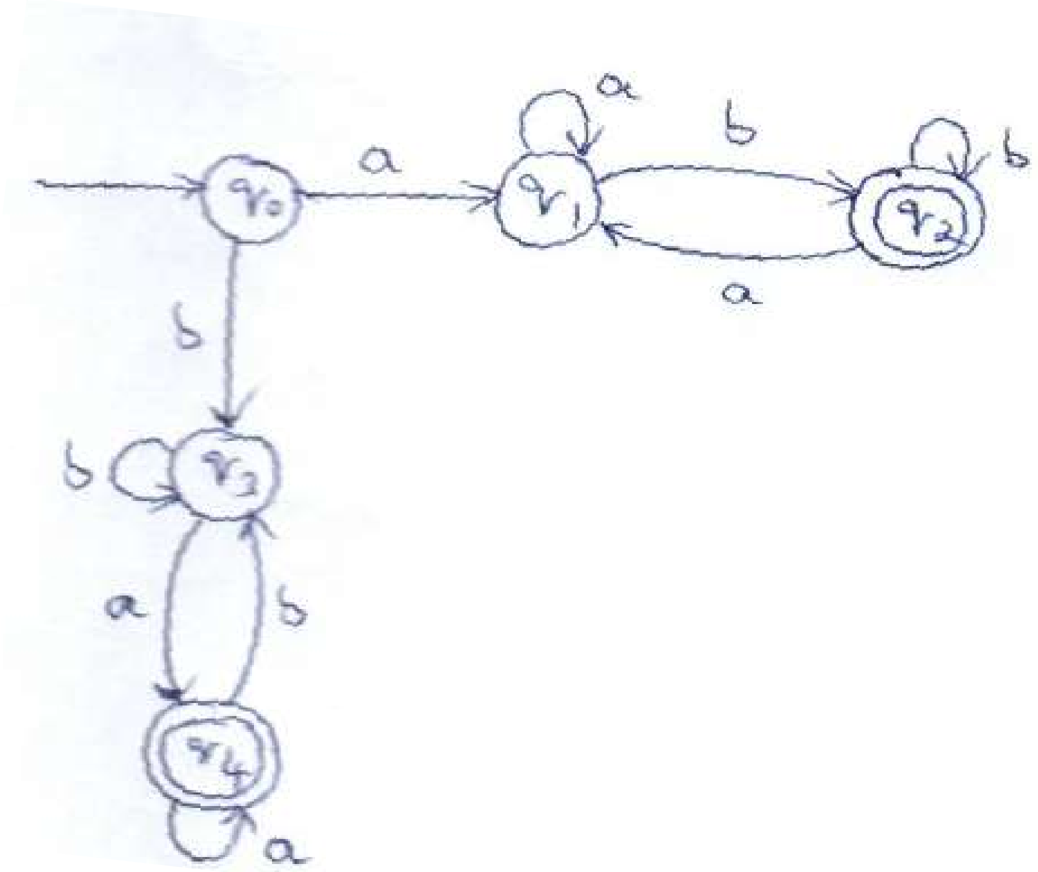
Deterministic Finite Automata

- **Example 21**: Design a DFA which accepts the language of strings those begin & end with same symbol over the alphabet $\Sigma = \{a, b\}$.
- **Solution**:



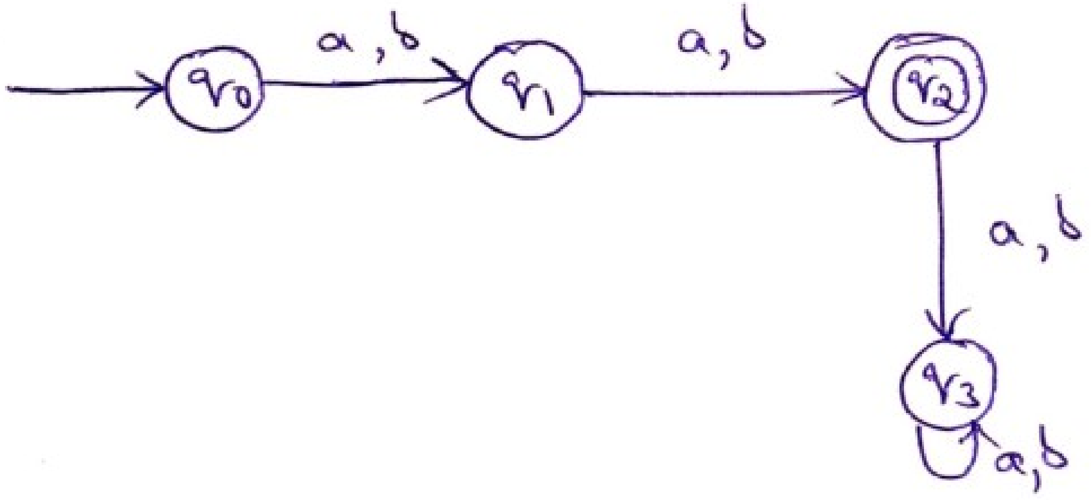
Deterministic Finite Automata

- **Example 22**: Design a DFA which accepts the language of strings those begin & end with different symbols over the alphabet Σ = {a, b}.
- **Solution**:



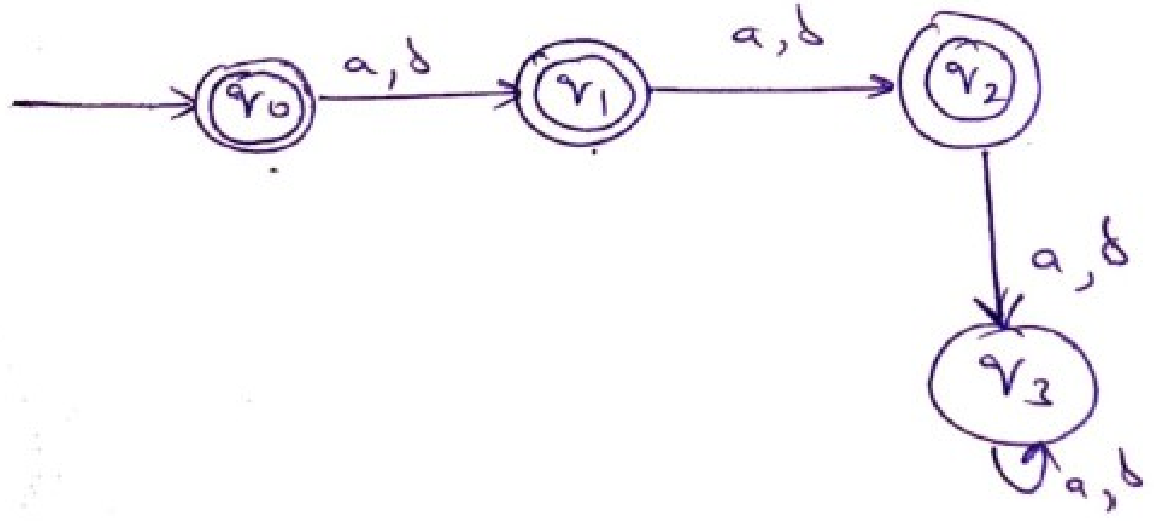
Deterministic Finite Automata

- **Example 23**: Design a DFA which accepts the language of strings of length exactly 2 ($|w|=2$) over the alphabet $\Sigma = \{a, b\}$.
- **Solution**: $L = \{aa, ab, ba, bb\}$



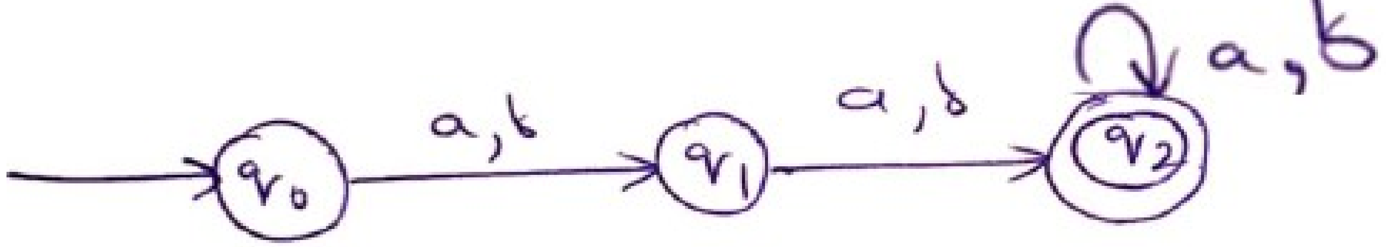
Deterministic Finite Automata

- **Example 24**: Design a DFA which accepts the language of strings of length at most 2 (|w|<=2) over the alphabet Σ = {a, b}.
- **Solution**: L ={ϵ, a, b, aa, ab, ba, bb}



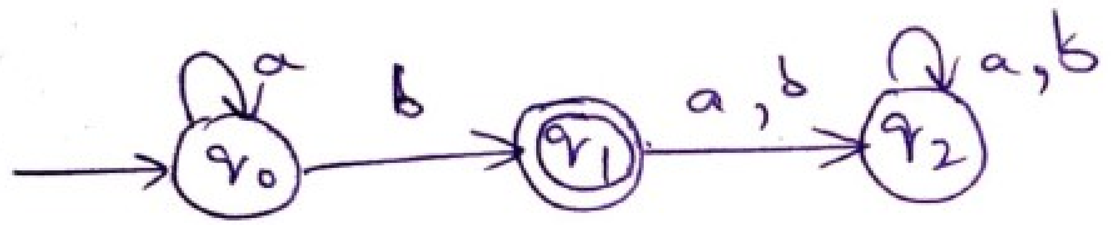
Deterministic Finite Automata

- **Example 25**: Design a DFA which accepts the language of strings of length at least 2 (|w|>=2| over the alphabet Σ = {a, b}.
- **Solution**: L ={aa, ab, ba, bb, aaa, …, bbb, ….}



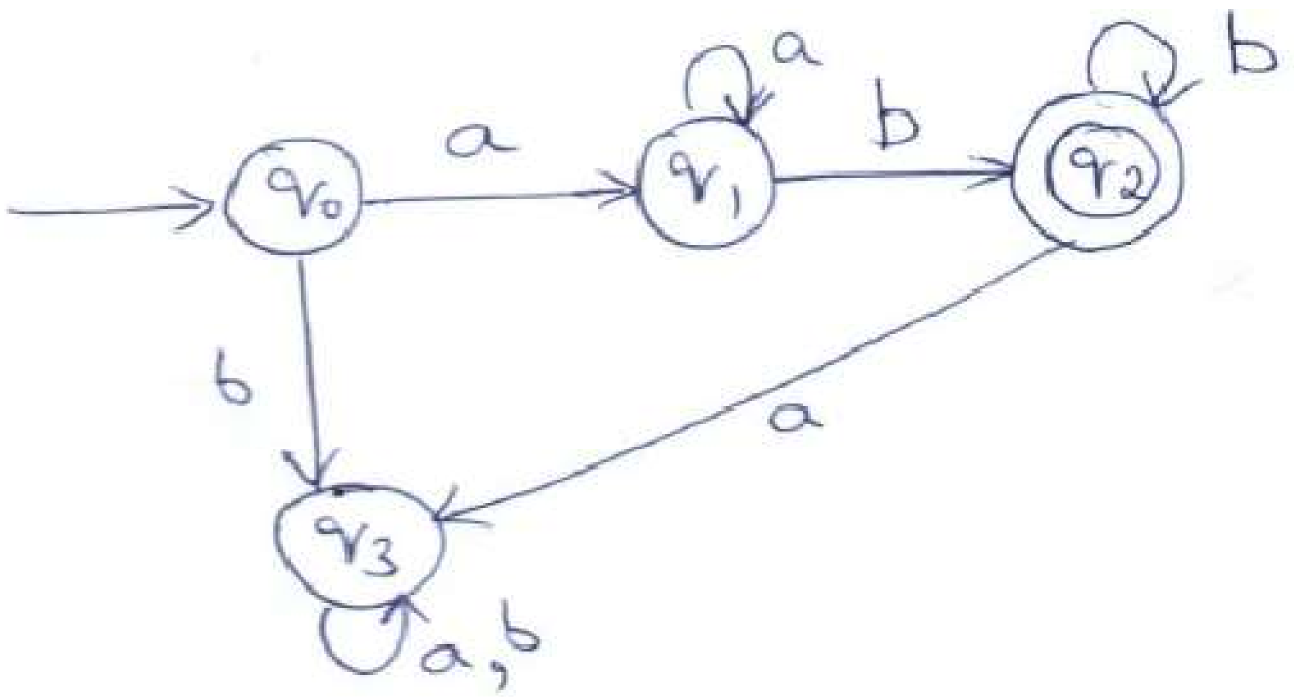
DETERMINISTIC FINITE AUTOMATA

- **Example 26**: Design a DFA which accepts the language L= { a^n b, m, n >=0 } over the alphabet Σ = {a, b}.
- **Solution**: L ={b, ab, aab, aab, aaab, ...}



DETERMINISTIC FINITE AUTOMATA

- **Example 27**: Design a DFA which accepts the language L= { a^m b^n, m, n >0 } over the alphabet Σ = {a, b}.
- **Solution**: L ={ab, aab, abb, aabb, aaabb, aaabbb, …}



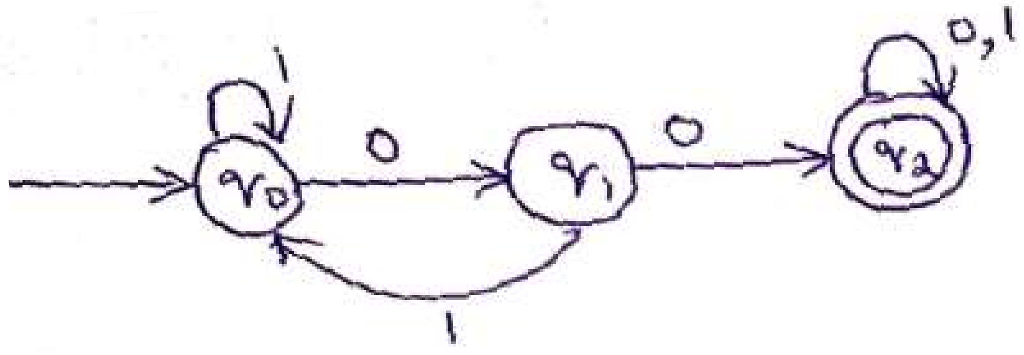
DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- If the language is L, the complement is denoted as $\bar{L}$.

1. In order to design a DFA, consider the given language as $\bar{L}$.
2. Design a DFA for L.
3. Change the final state to non final state and non final state to final state.
4. The initial state remains the same.
5. Which gives DFA for given language.

DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

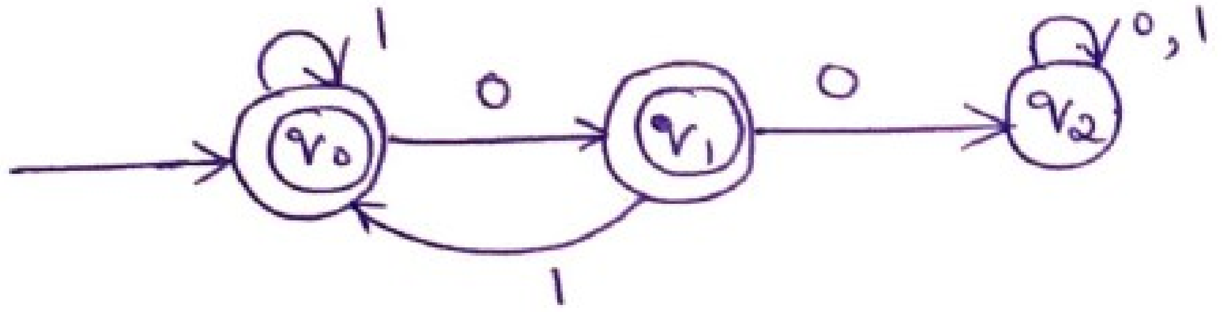
- **Example 1**: Let Σ = {0, 1}*. Design a DFA for a language containing strings which does not contain 00 as substring.
- **Solution**: L = { w | w contain 00 as substring }.
- Let us design a DFA for a language containing strings with 00 as substring.



DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- $\bar{L}$ = { w | w does not contain 00 as substring }.

The DFA for $\bar{L}$ is

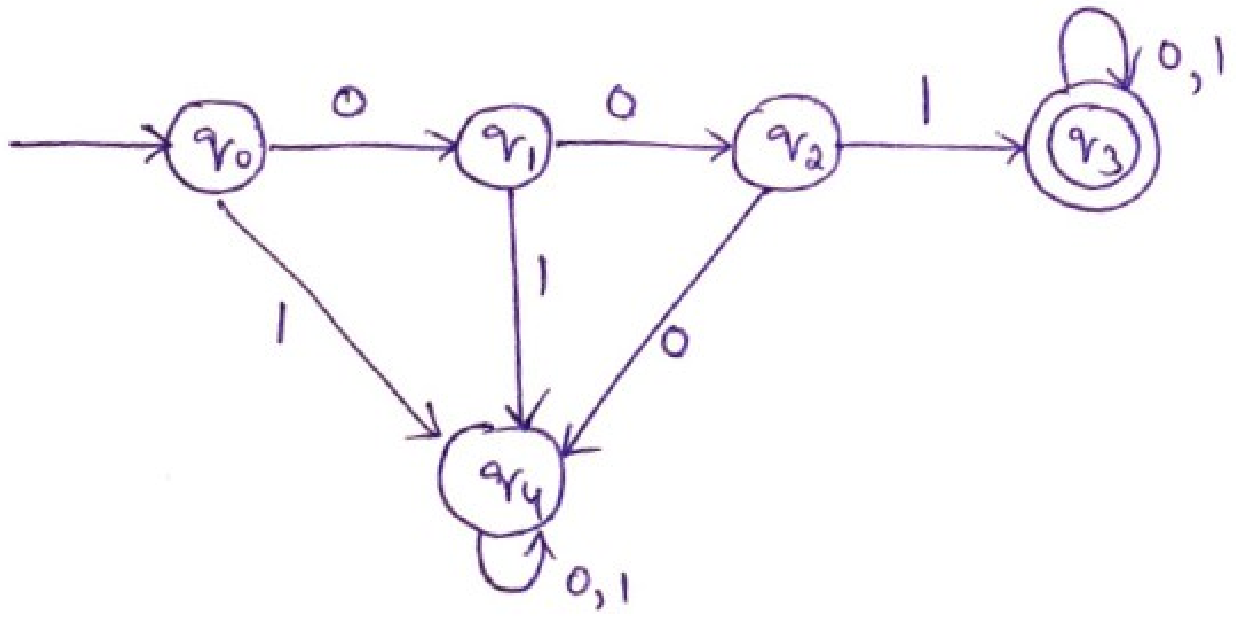


DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- **Example 2**: Let $\Sigma = \{0, 1\}^*$. Design a DFA for a language containing strings which does not starts with 001.
- **Solution**:

$L = \{ w \mid w \text{ starts with } 001 \}$

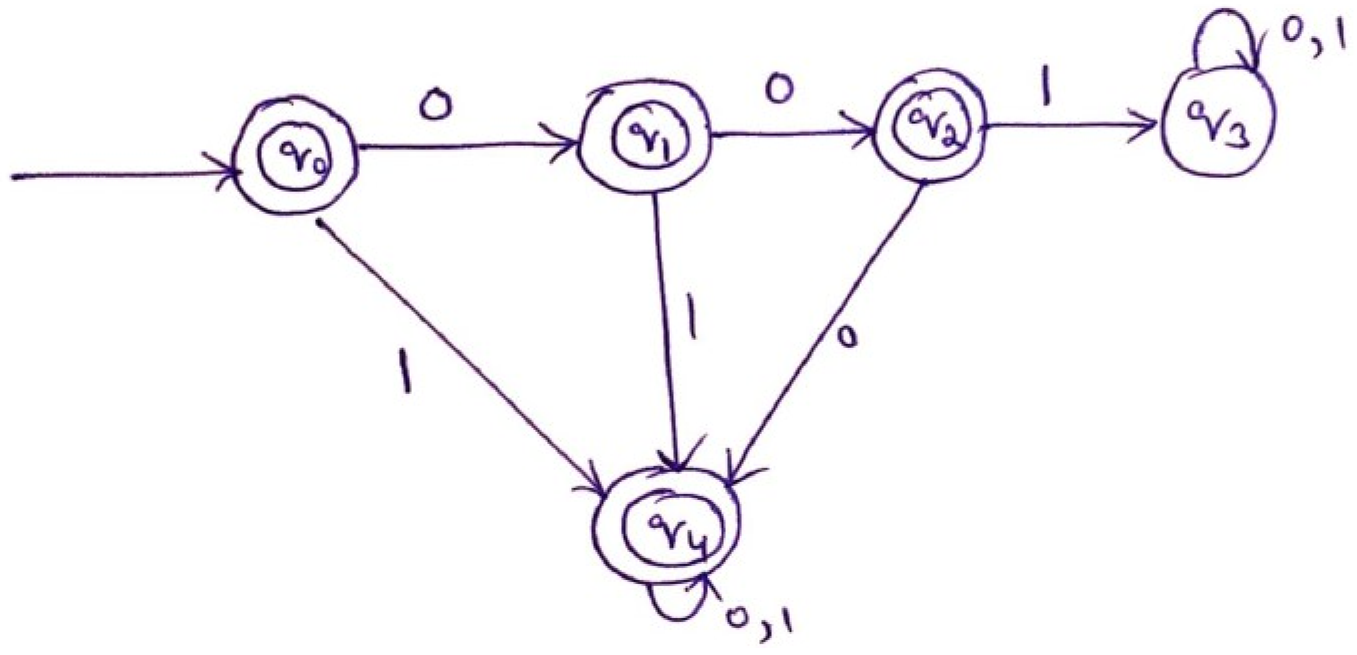
The DFA for L is



DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

$\bar{L} = \{ w \mid w$ does not starts with $001 \}$.

The DFA for $\bar{L}$ is

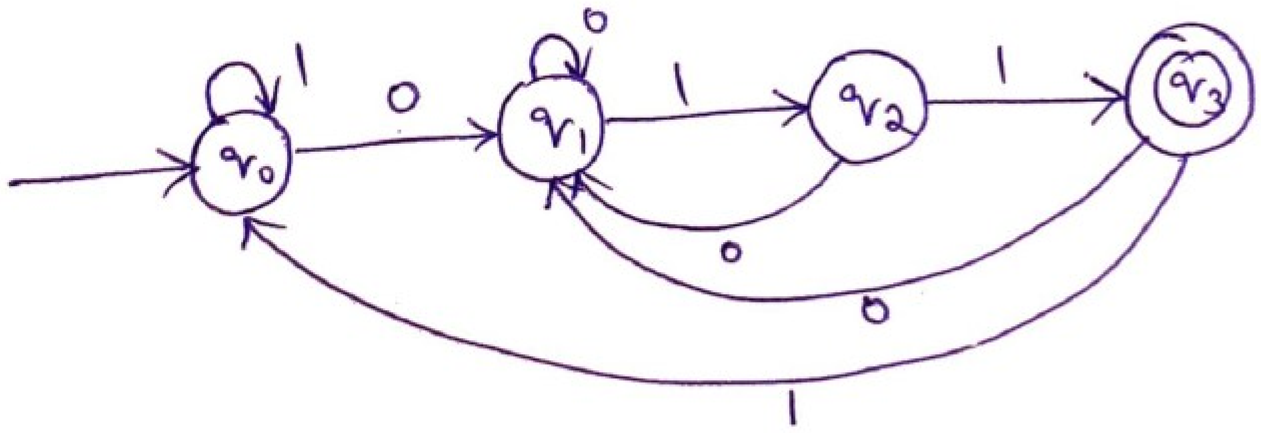


DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- **Example 3**: Let $\Sigma = \{0, 1\}^*$. Design a DFA for a language containing strings which does not end with 011.
- **Solution**:

$L = \{ \omega \mid \omega$ end with 011 $\}$

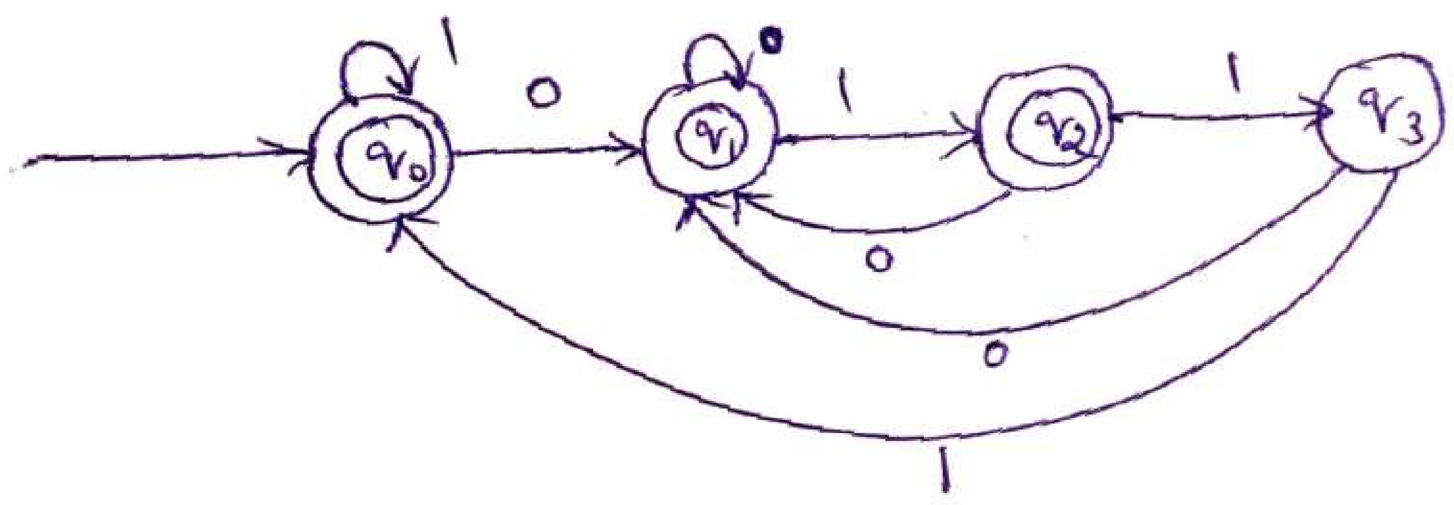
The DFA for L is



DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

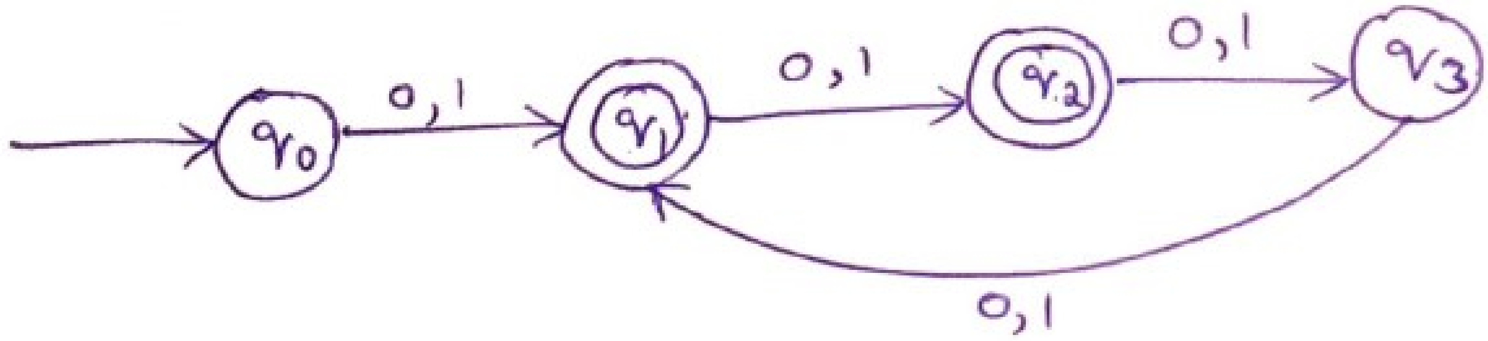
$\overline{L} = \{w \mid w \text{ does not end with } 011\}$

The DFA for $\overline{L}$ is



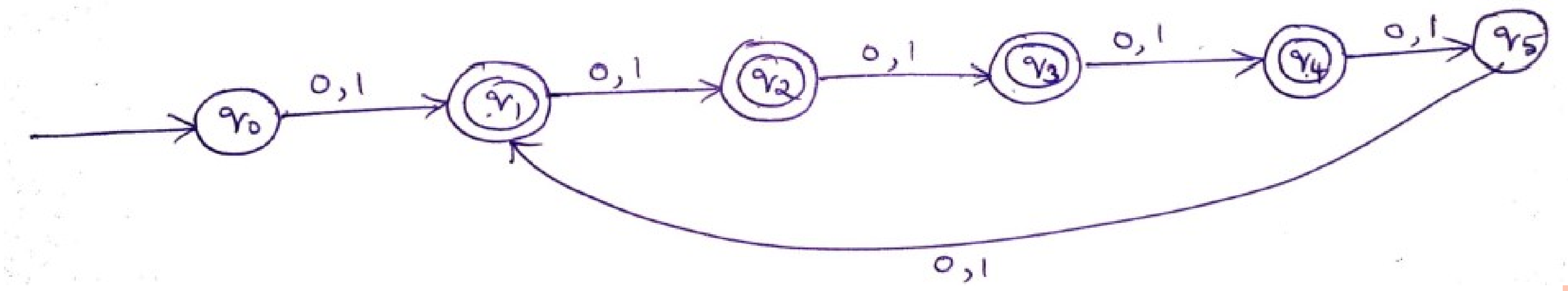
DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- **Example 4**: Let Σ = {0, 1}. Design a finite automata for language L = { w | |w| mod 3 ≠ 0 }.



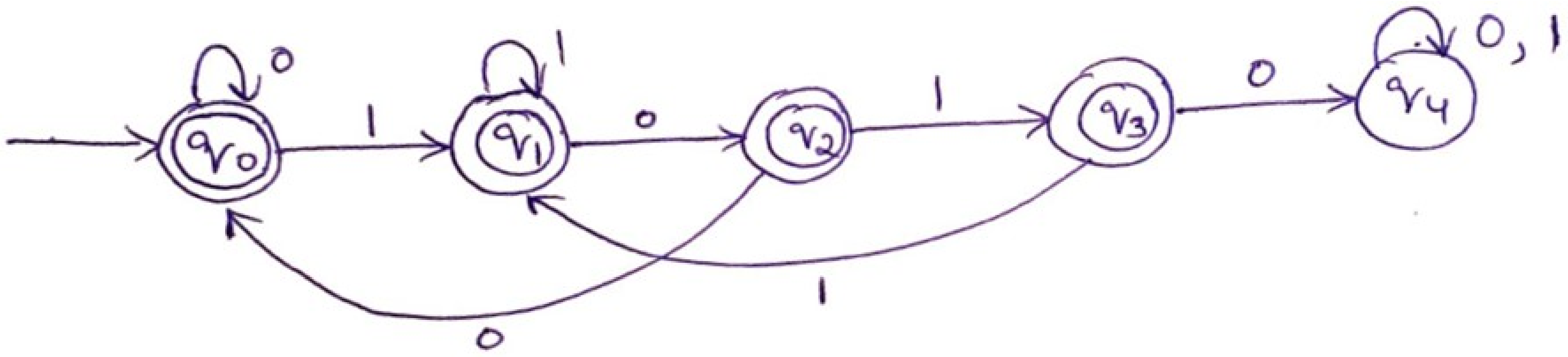
DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- **Example 5**: Let Σ = {0, 1}. Design a finite automata for language L = { w | |w| mod 5 ≠ 0 }.



DFA FOR COMPLEMENT OF A GIVEN LANGUAGE

- **Example 6**: Let Σ = {0, 1}. Design a finite automata for language L = { w | w does not contain 1010 }.



q_0
q_1
q_2
q_3
q_4
0
1
1
0
1
0
0, 1
1
0