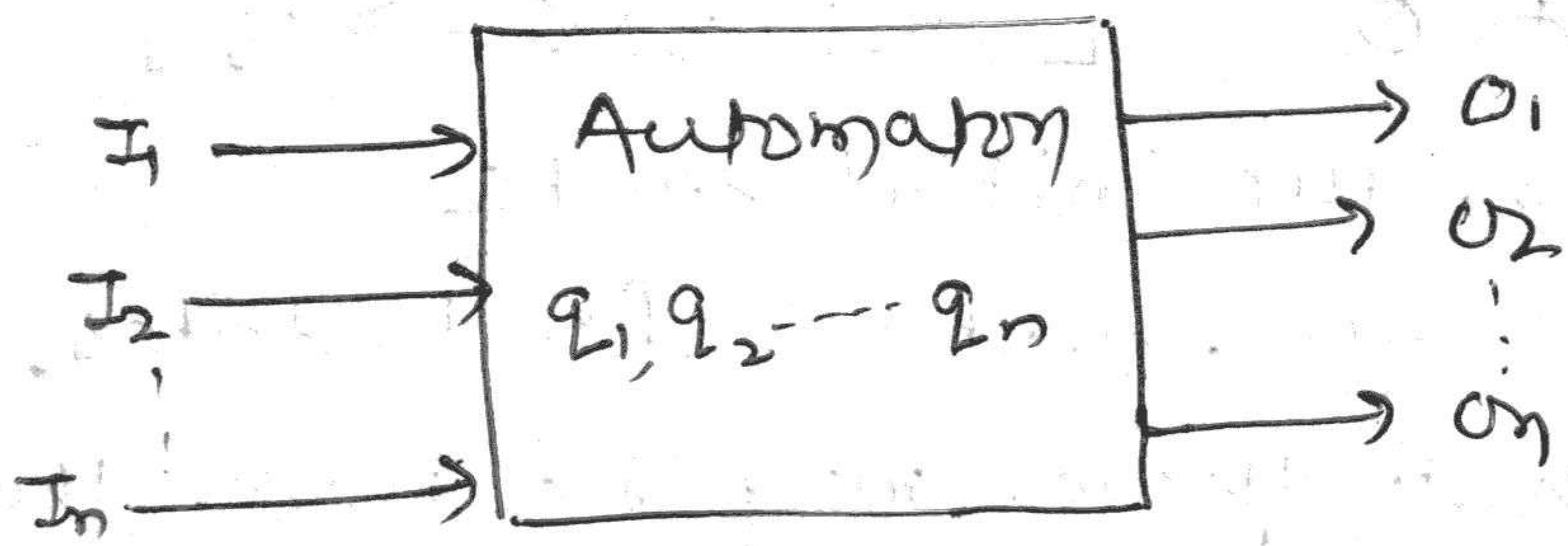
# UNIT-5
# Finite Automata

## Automaton :-

An automaton is defined as a system where energy, materials and information are transformed, transmitted and used for performing some function without direct participation of man.

In computer science the term 'automaton' means 'discrete automaton'.




→ Input :- discrete instants of time $t_1, t_2 \cdots t_n$ the input value $I_1, I_2, \cdots I_n$ each of which can take a finite no. of fixed value from the input alphabet $\Sigma$.

→ Output :- $O_1, O_2 \cdots O_n$ are output of the model.

→ States :- At any instant of time the automaton can be in one of the states $q_1, q_2 \cdots q_n$

→ State Relation :- The next state of an automaton at any instant of time is determined by the present state and the present input.

→ Output Relation :- the output relation is related to either state only or to both the input and the state.

## finite Automaton:-

A finite automaton can be represent by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

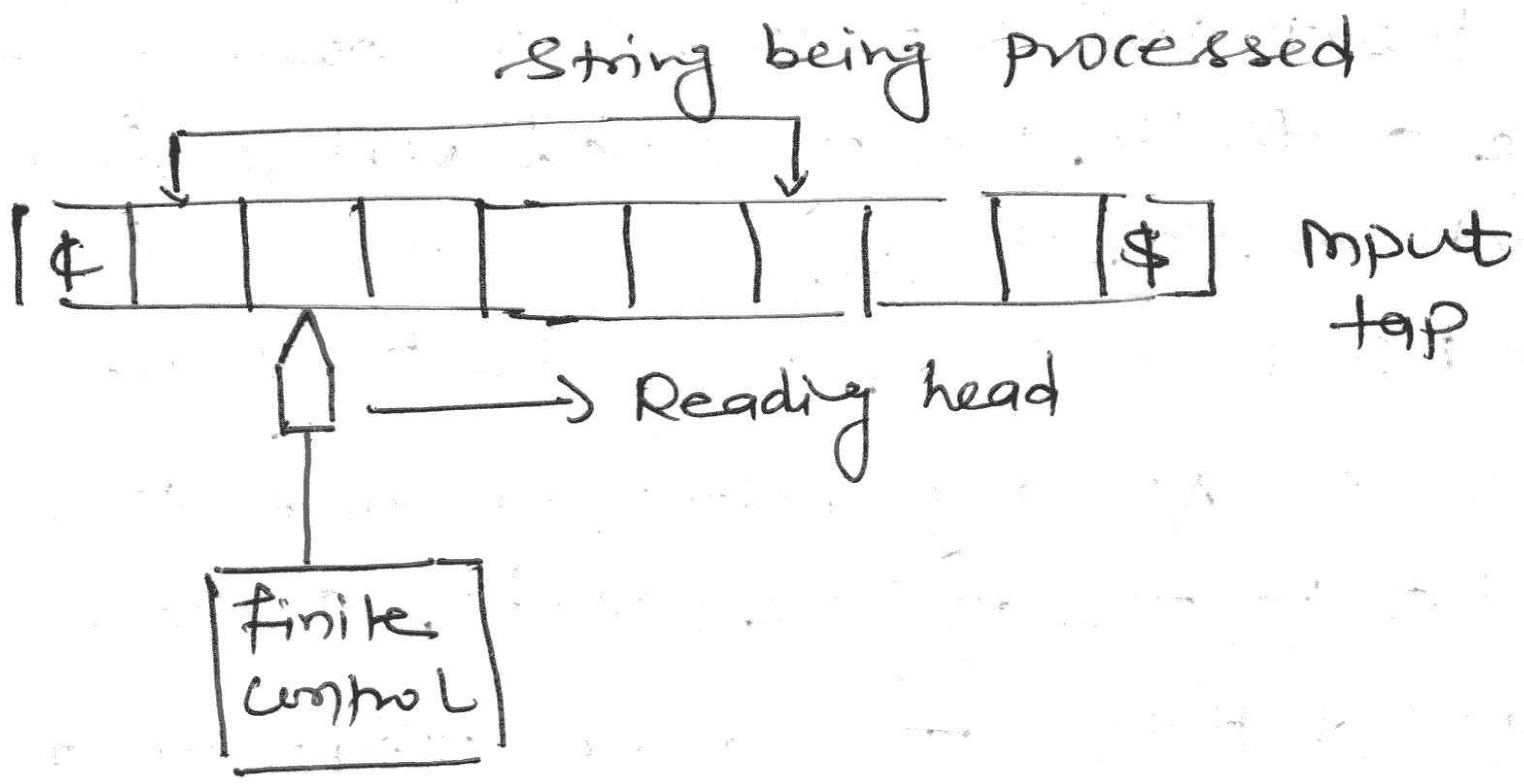
→ $Q$ is finite non-empty set of states.

→ $\Sigma$ is finite non-empty set of input called the input alphabet.

→ $\delta$ is the transition function which maps $Q \times \Sigma$ to $Q$ $[\ \delta : Q \times \Sigma \to Q\ ]$

→ $q_0 \in Q$ is the initial state

→ $F \subseteq Q$ is the set of final states, it is assumed here that there may be more than one final state.

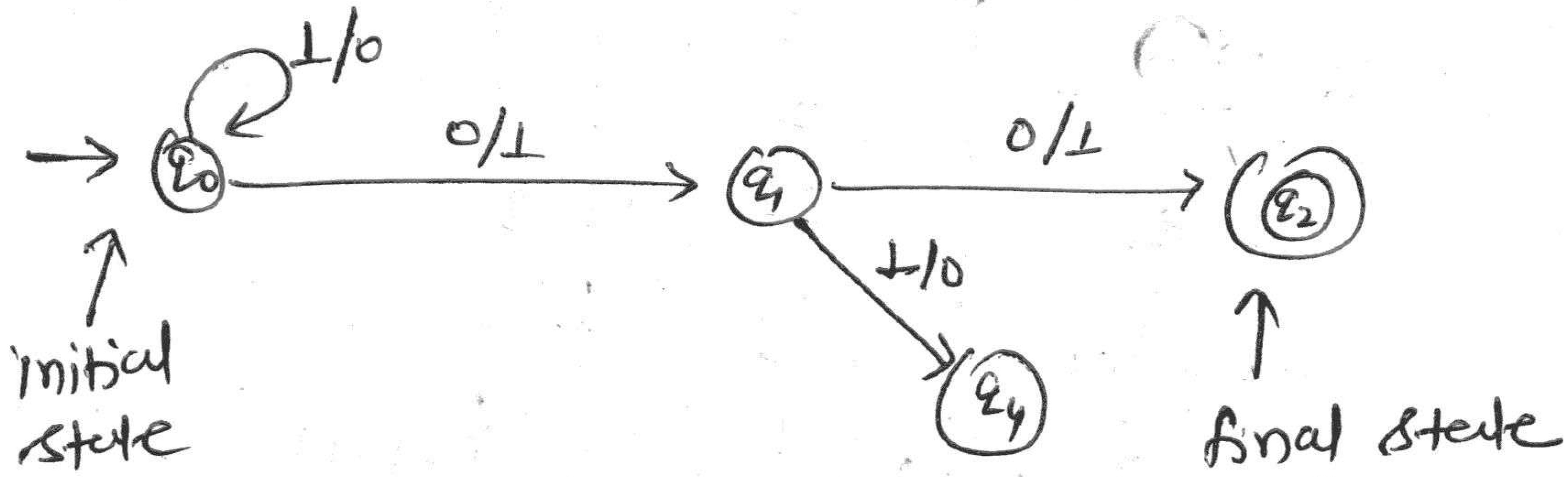



* Block diagram of a finite automaton *

## Transition System:-

A transition system or a transition graph is a finite directed labelled graph in which each vertex (or node) represent a state and the directed edges indicate the transition of a state and the edges are labbled with Input/output.

A transition system accepts a string $w$ in $\Sigma^*$ if
(I) there exist a path which originates from some initial state, goes along the arrows and terminates at some final state. and
(II) the path value obtained by concatenation of all edge-labels of the path is equal to $w$.




determine acceptability of 1100.

path value of $q_0 q_0 q_1 q_2$ is 1100. As $q_2$ is final state. so 1100 is accepted by the transition system.

Ex. 1101

Path value $q_0 q_0 q_1 q_4$ is 1101. but $q_4$ is not final state, so string 1101 is not accepted by transition system

## Acceptability of a string by a finite automaton:

Defn. -

A string $x$ is accepted by finite automata

$$M = (Q, \Sigma, \delta, q_0, F)$$

if $\delta(q_0, x) = q$ for some $q \in F$

Ex

$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0, 1\}$, $F = \{q_0\}$

input string is 110101

=) transition table

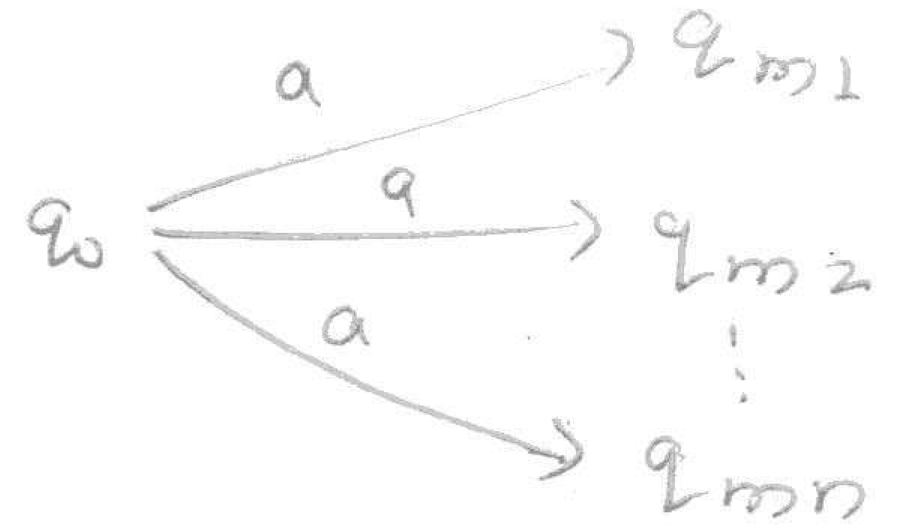
| State | Input 0 | Input 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_2$ | $q_1$ |
| $q_1$ | $q_3$ | $q_0$ |
| $q_2$ | $q_0$ | $q_3$ |
| $q_3$ | $q_1$ | $q_2$ |

$$\begin{aligned} \Rightarrow \quad \delta(q_0, 110101) &= \delta(q_1, 10101) \\ &= \delta(q_0, 0101) \\ &= \delta(q_2, 101) \\ &= \delta(q_3, 01) \\ &= \delta(q_1, 1) \\ &= q_0 \end{aligned}$$

Hence it's final state so string is accepted

## Non Deterministic finite Automata (NDFA) :-

In NDFA when an input is read the automata at each step my choose to go to any of the several possible (legal) "next states". Since this choice is not determined by anything, therefor it is called non deterministic.




Next state may be one or more than one

Defn :- A non-deterministic finite automation is a five tuple.

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

$Q$ = finite set of states

$\Sigma$ = Finite set of input

$\delta$ = Transition function

$\delta : Q \times \Sigma \to 2^Q$ ($2^Q$ is power set of $Q$)

$q_0$ = initial state

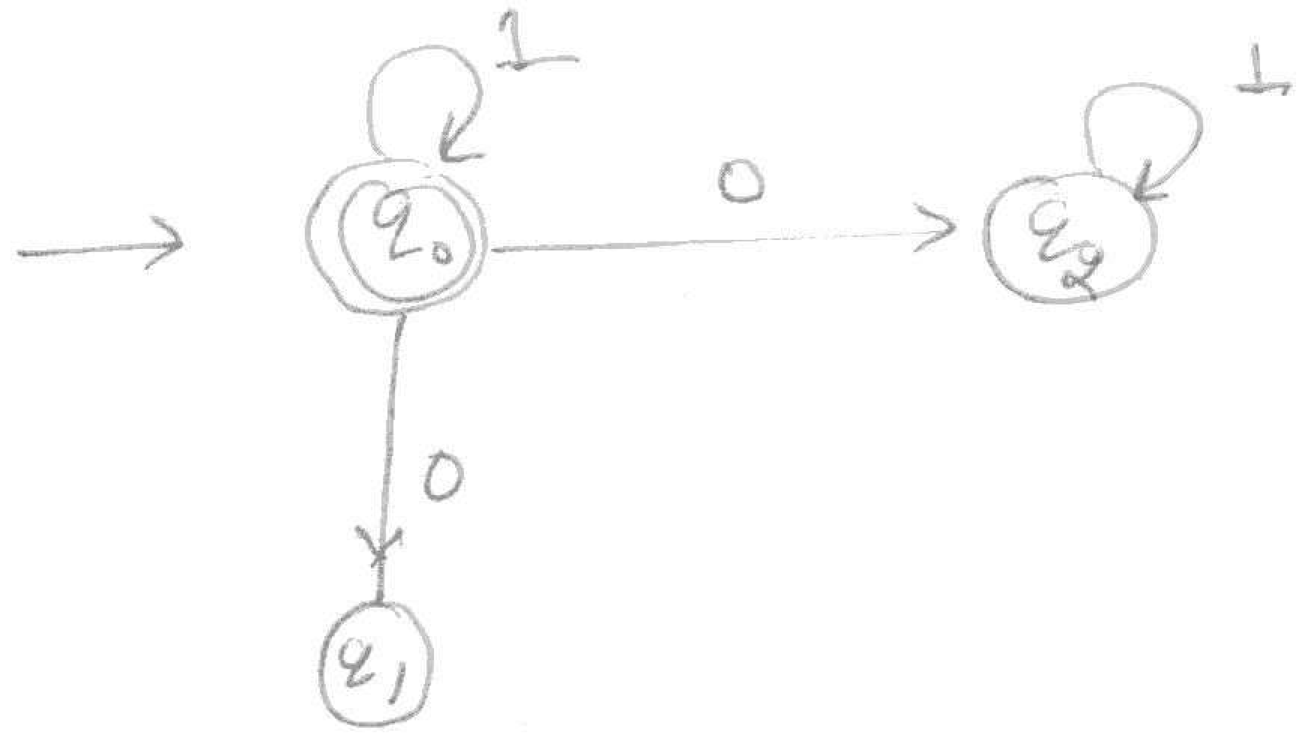
$q_0 \in Q$.

$F$ = set of final states

$F \subseteq Q$.

Ex. Test whether the strings 110011 are accepted by the transition graph.

$M = (Q, \Sigma, \delta, q_0\ F)$.




| | state | input |
|---|---|---|
| | a | b |
| $\rightarrow q_0$ | $q_1, q_2$ | $q_0$ |
| $q_1$ | $\phi$ | $\phi$ |
| $q_2$ | $\phi$ | $q_2$ |

110011 $\Rightarrow$ first we have a symbol 1.

$\therefore$ path $= q_0 \xrightarrow{1} q_0$

again 1

$\therefore$ path $q_0 \xrightarrow{1} q_0$

then 0 now there are two path for 0

$q_0 \xrightarrow{0} q_1$ and $q_0 \xrightarrow{0} q_2$

if we take path $q_0 \xrightarrow{0} q_1$ then it block so we choose $q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_2$

now again 0

But from $q_2$ there is no path for 0 input

thus, the string is not accepted. i.e. rejected.

1 Construct a DFA equivalent to

$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$

Q — $\Sigma$ — Transition function — Initial State — Final State

where $\delta$ is define by its state table

| State / $\Sigma$ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_0$ | $q_1$ |
| $q_1$ | $q_1$ | $q_0 q_1$ |

For the deterministic automaton $M_1$,

- the states are subset of $\{q_0, q_1\}$

$\Rightarrow \phi, [q_0], [q_1], [q_0 q_1]$

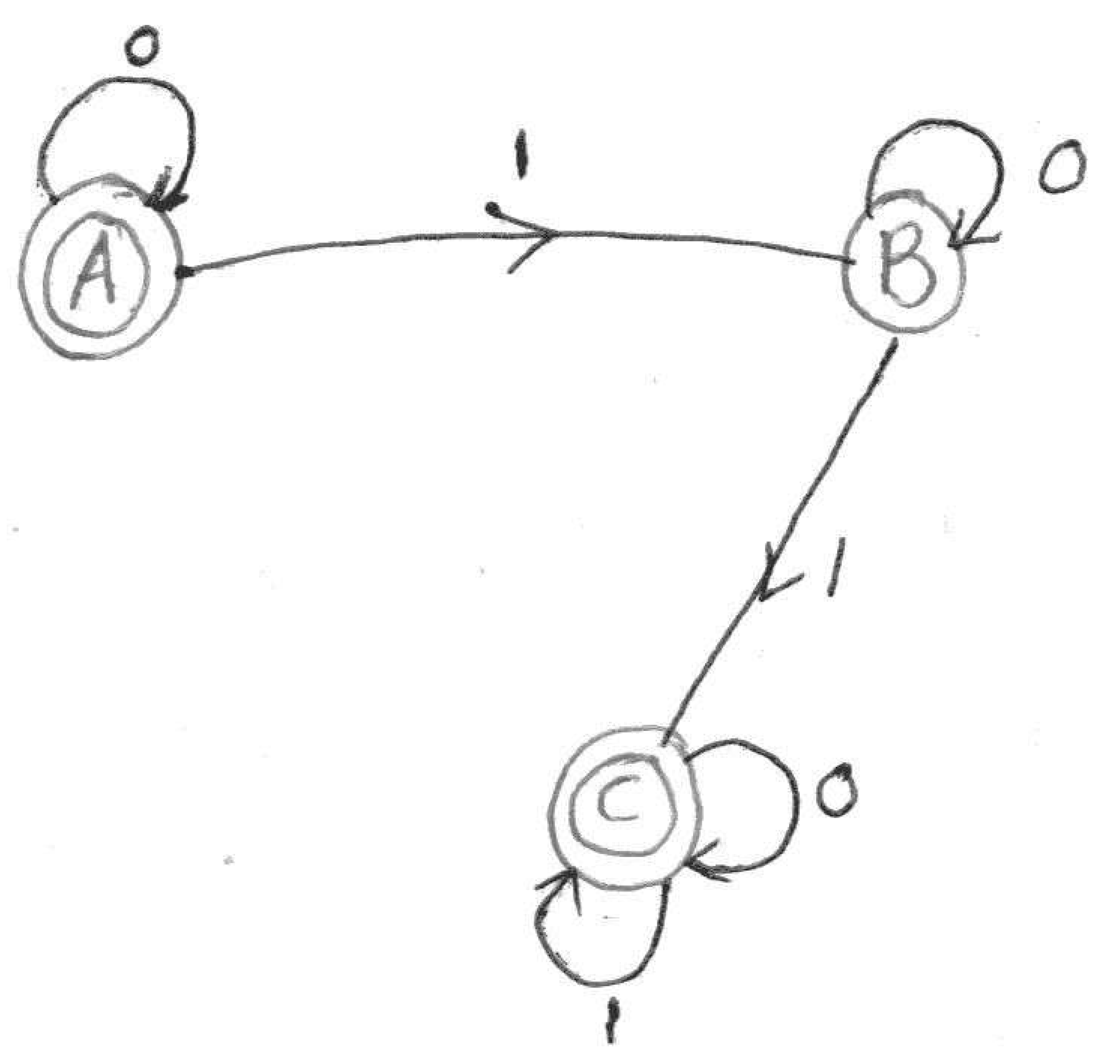
- $[q_0]$ is initial state
- $[q_0]$ and $[q_0, q_1]$ are the final states as these are the only states containing $q_0$ and

| State / $\Sigma$ | 0 | 1 |
|---|---|---|
| $\phi$ | $\phi$ | $\phi$ |
| $[q_0]$ | $[q_0]$ | $[q_1]$ |
| $[q_1]$ | $[q_1]$ | $[q_0, q_1]$ |
| $[q_0 q_1]$ | $[q_0, q_1]$ | $[q_0, q_1]$ |

For easy you can replace

$[q_0] \rightarrow A$
$[q_1] \rightarrow B$
$[q_0 q_1] \rightarrow C$

| State | 0 | 1 |
|---|---|---|
| $\rightarrow$ A | A | B |
| B | B | C |
| C | C | C |




DFA

2. Find the DFA equivalent to

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_1\})$

$\delta$ is given by

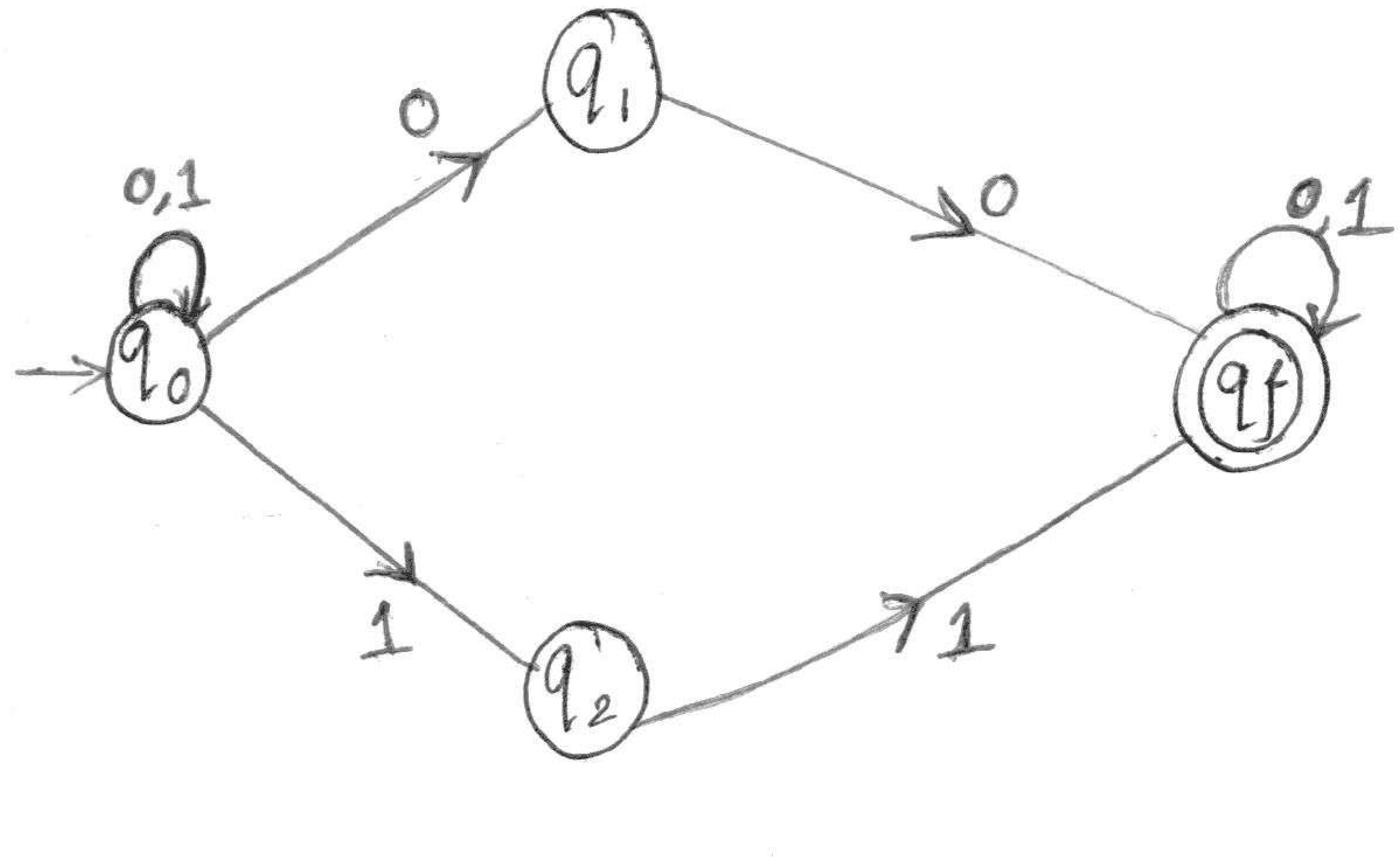
| State | a | b |
|---|---|---|
| $\rightarrow q_0$ | $q_0, q_1$ | $q_2$ |
| $q_1$ | $q_0$ | $q_1$ |
| $q_2$ | - | $q_0, q_1$ |

| State | a | b |
|---|---|---|
| → [$q_0$] | [$q_0, q_1$] | [$q_2$] |
| [$q_2$] | $\phi$ | [$q_0, q_1$] |
| [$q_0, q_1$] | [$q_0, q_1$] | [$q_1, q_2$] |
| [$q_1, q_2$] | [$q_0$] | [$q_0, q_1$] |

Converting of non-deterministic finite automata to deterministic finite automata :-

for every NDFA there exists an equivalent DFA, whenever we are convering a NDFA to DFA we are given either a state transition diagram or a state trasition table of that NDFA.

3. Convert the following non-deterministic transition system to deterministic system.

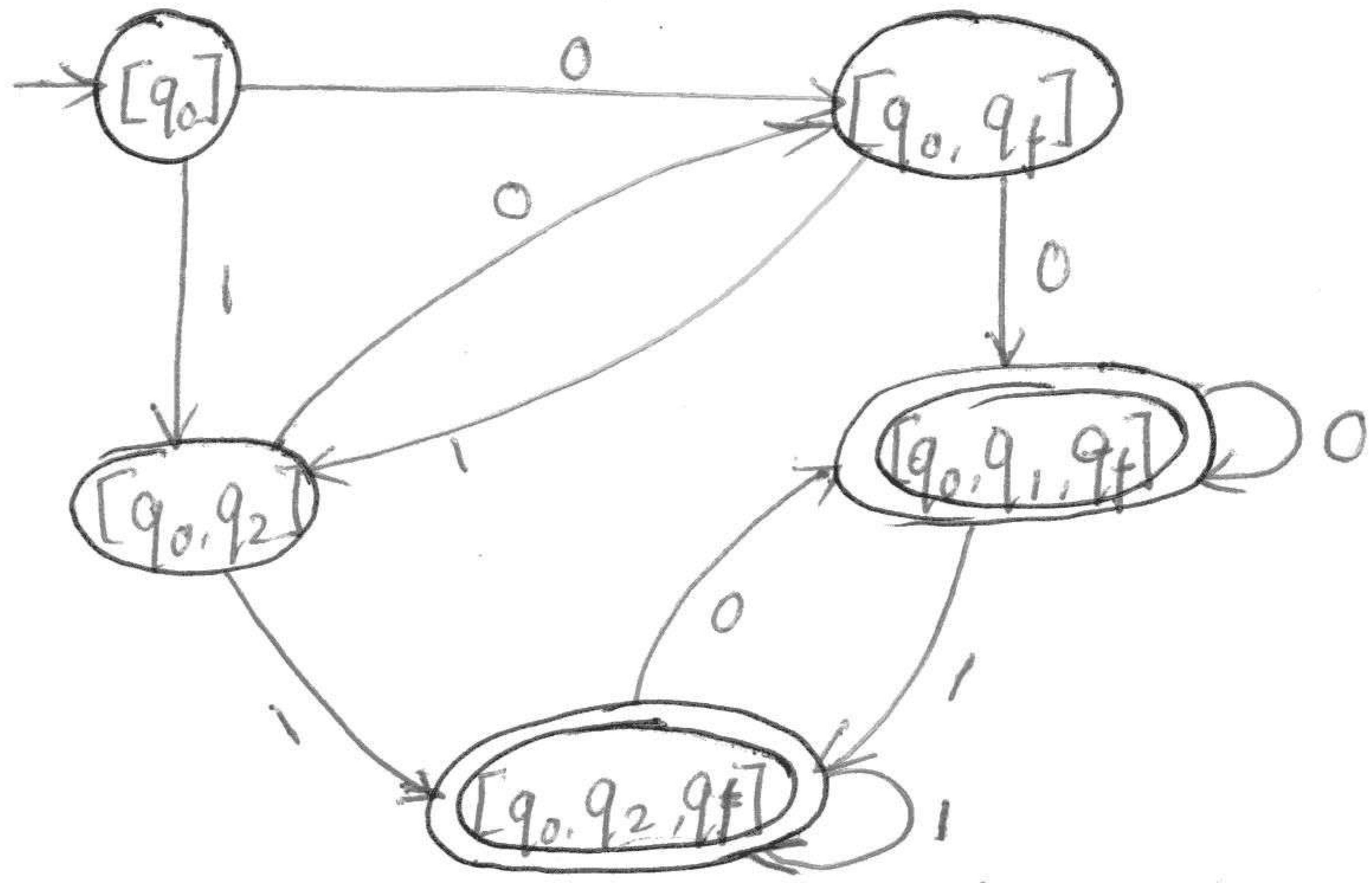


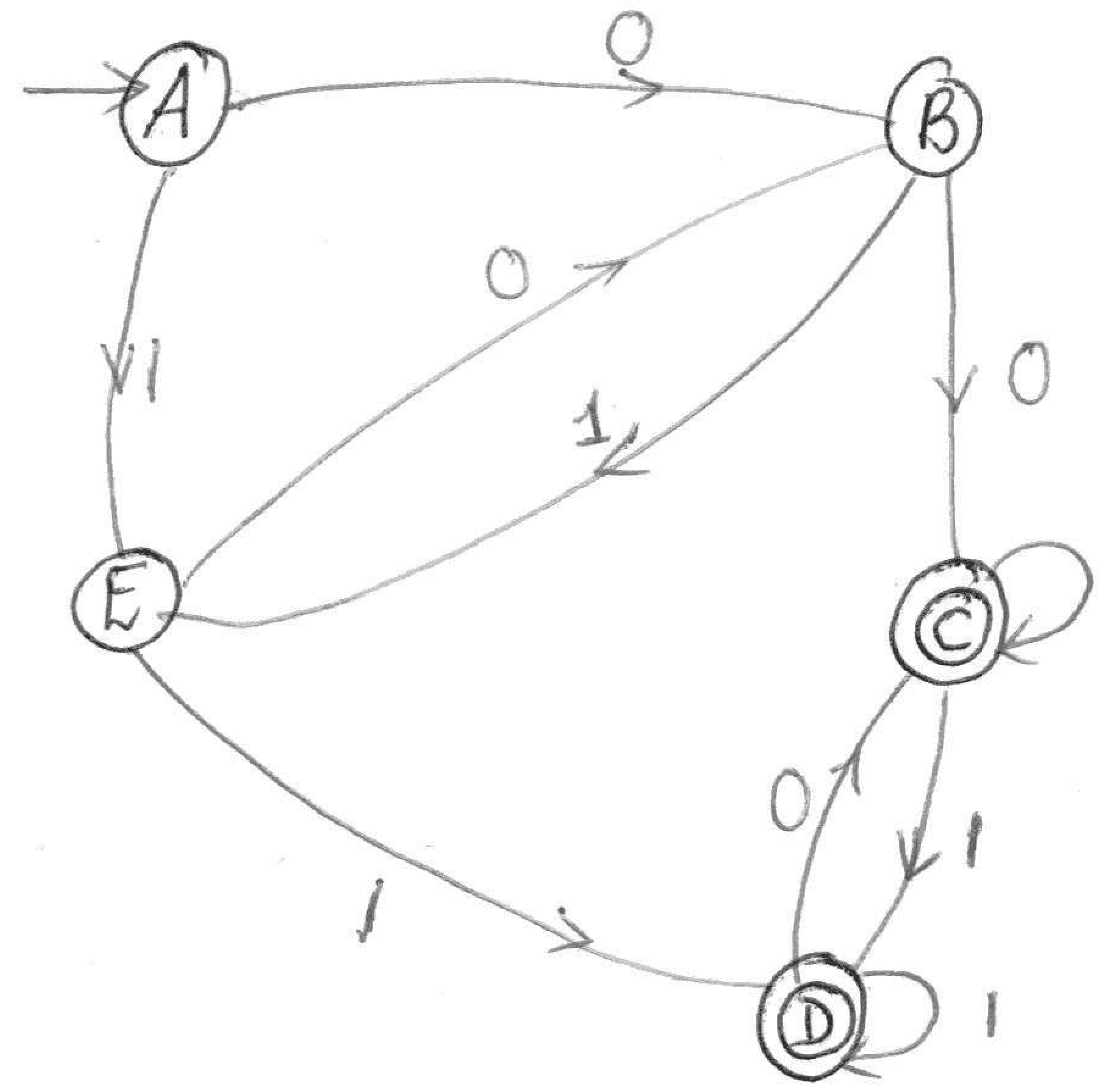

Construction of state transition table for the given non-deterministic system.

| State | Input | |
|---|---|---|
| | 0 | 1 |
| → $q_0$ | $q_0, q_1$ | $q_0, q_2$ |
| $q_1$ | $q_f$ | $\phi$ |
| $q_2$ | $\phi$ | $q_f$ |
| $q_f$ | $q_f$ | $q_f$ |

| State | Input | |
|---|---|---|
| | 0 | 1 |
| → $[q_0]$ | $[q_0, q_1]$ | $[q_0, q_2]$ |
| $[q_0, q_1]$ | $[q_0, q_1, q_f]$ | $[q_0, q_2]$ |
| $[q_0, q_2]$ | $[q_0, q_1]$ | $[q_0, q_2, q_f]$ |
| $[q_0, q_0, q_f]$ | $[q_0, q_1, q_f]$ | $[q_0, q_2, q_f]$ |
| $[q_0, q_2, q_f]$ | $[q_0, q_1, q_f]$ | $[q_0, q_2, q_f]$ |

Now the state transition diagram is






$\because$ A = $q_0$

B = $q_0, q_1$

C = $q_0, q_1, q_f$

D = $q_0, q_2, q_f$

E = $q_0 q_2$

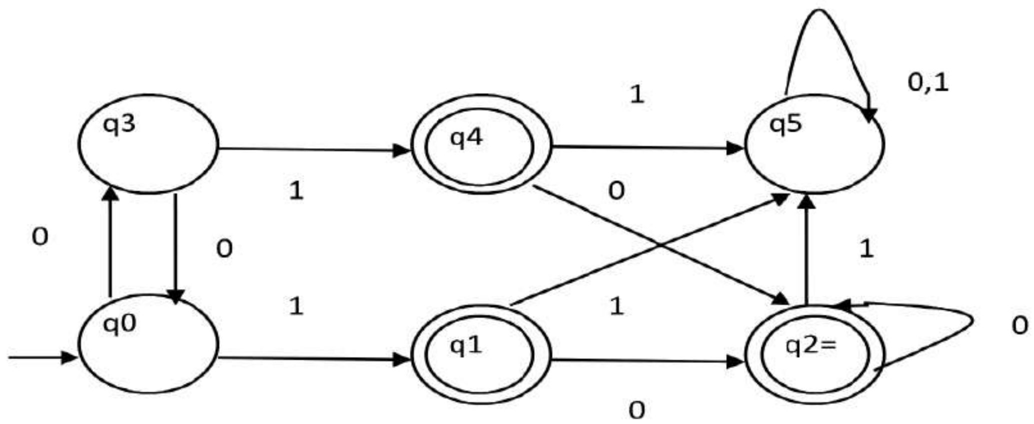
# Minimization of DFA

DFA minimization stands for converting a given DFA to its equivalent DFA with minimum number of states.

Minimization of DFA: Suppose there is a DFA D < Q, Σ, q0, δ, F > which recognizes a language L. Then the minimized DFA D < Q', Σ, q0, δ', F' > can be constructed for language L as:

Step 1: We will divide Q (set of states) into two sets. One set will contain all final states and other set will contain non-final states. This partition is called $P_0$.
Step 2: Initialize k = 1
Step 3: Find $P_k$ by partitioning the different sets of $P_{k-1}$. In each set of $P_{k-1}$, we will take all possible pair of states. If two states of a set are distinguishable, we will split the sets into different sets in $P_k$.
Step 4: Stop when $P_k = P_{k-1}$ (No change in partition)
Step 5: All states of one set are merged into one. No. of states in minimized DFA will be equal to no. of sets in $P_k$.

**How to find whether two states in partition $P_k$ are distinguishable?**
Two states ( qi, qj ) are distinguishable in partition $P_k$ if for any input symbol a, δ ( qi, a ) and δ ( qj, a ) are in different sets in partition $P_{k-1}$.
**Example**
Consider the following DFA shown in figure.




Step 1. P0 will have two sets of states. One set will contain q1, q2, q4 which are final states of DFA and another set will contain remaining states. So P0 = { { q1, q2, q4 }, { q0, q3, q5 } }.
Step 2. To calculate P1, we will check whether sets of partition P0 can be partitioned or not:

i) For set { q1, q2, q4 } :
δ ( q1, 0 ) = δ ( q2, 0 ) = q2 and δ ( q1, 1 ) = δ ( q2, 1 ) = q5, So q1 and q2 are not distinguishable.
Similarly, δ ( q1, 0 ) = δ ( q4, 0 ) = q2 and δ ( q1, 1 ) = δ ( q4, 1 ) = q5, So q1 and q4 are not distinguishable.
Since, q1 and q2 are not distinguishable and q1 and q4 are also not distinguishable, So q2 and q4 are not distinguishable. So, { q1, q2, q4 } set will not be partitioned in P1.

ii) For set { q0, q3, q5 } :
δ ( q0, 0 ) = q3 and δ ( q3, 0 ) = q0
δ ( q0, 1) = q1 and δ( q3, 1 ) = q4
Moves of q0 and q3 on input symbol 0 are q3 and q0 respectively which are in same set in partition P0. Similarly, Moves of q0 and q3 on input symbol 1 are q3 and q0 which are in same set in partition P0. So, q0 and q3 are not distinguishable.

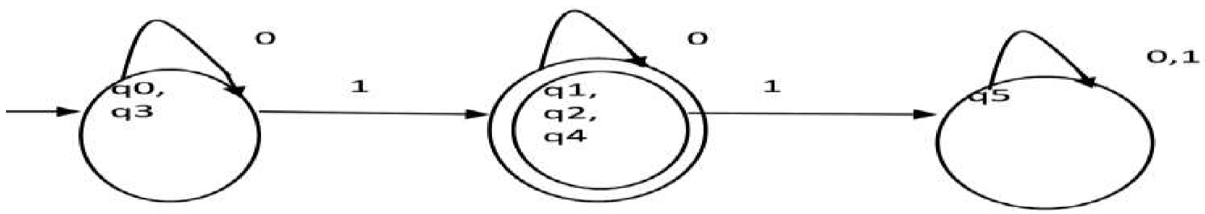
δ ( q0, 0 ) = q3 and δ ( q5, 0 ) = q5 and δ ( q0, 1 ) = q1 and δ ( q5, 1 ) = q5
Moves of q0 and q5 on input symbol 1 are q3 and q5 respectively which are in different set in partition P0. So, q0 and q5 are distinguishable. So, set { q0, q3, q5 } will be partitioned into { q0, q3 } and { q5 }. So,
P1 = { { q1, q2, q4 }, { q0, q3}, { q5 } }

To calculate P2, we will check whether sets of partition P1 can be partitioned or not:

iii)For set { q1, q2, q4 } :
δ ( q1, 0 ) = δ ( q2, 0 ) = q2 and δ ( q1, 1 ) = δ ( q2, 1 ) = q5, So q1 and q2 are not distinguishable.
Similarly, δ ( q1, 0 ) = δ ( q4, 0 ) = q2 and δ ( q1, 1 ) = δ ( q4, 1 ) = q5, So q1 and q4 are not distinguishable.
Since, q1 and q2 are not distinguishable and q1 and q4 are also not distinguishable, So q2 and q4 are not distinguishable. So, { q1, q2, q4 } set will not be partitioned in P2.

iv)For set { q0, q3 } :
δ ( q0, 0 ) = q3 and δ ( q3, 0 ) = q0
δ ( q0, 1 ) = q1 and δ ( q3, 1 ) = q4
Moves of q0 and q3 on input symbol 0 are q3 and q0 respectively which are in same set in partition P1. Similarly, Moves of q0 and q3 on input symbol 1 are q3 and q0 which are in same set in partition P1. So, q0 and q3 are not distinguishable.

v) For set { q5 }:
Since we have only one state in this set, it can’t be further partitioned. So,
P2 = { { q1, q2, q4 }, { q0, q3 }, { q5 } }
Since, P1=P2. So, this is the final partition. Partition P2 means that q1, q2 and q4 states are merged into one. Similarly, q0 and q3 are merged into one. Minimized DFA corresponding to DFA of Figure 1 is shown in Figure 2 as:



# Mealy & Moore Model

**Finite Automata with output** ~ In earlier consider DFA have binary outputs, either they accept the string or they don't accept the string.

Now we remove this restriction and consider the model where the output can be chosen from the some other alphabet.

The value of the output function $z(t)$ in the most general case is a function of the present state $q(t)$ and present input $x(t)$ i.e.

$$z(t) = \lambda(q(t), x(t))$$

where $\lambda$ is called the output function. This generalized model is called **Mealy Machine.**

If the output function $z(t)$ depends only on the present state and is independent of the current input; the output function may be written as

$$z(t) = \lambda(q(t))$$

This restricted model is called the **Moore Machine.**

i.) Moore Machine ~

A Moore Machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

- $Q$ ~ is a finite set of states
- $\Sigma$ ~ is the input alphabet
- $\Delta$ ~ is the output alphabet
- $\delta$ ~ is the transition function $\Sigma \times Q$ into $Q$.
- $\lambda$ ~ is the output function mapping $Q$ into $\Delta$ &
- $q_0$ ~ $q_0$ is the initial state.

## Example

| Present State | Next State | | Output |
|---|---|---|---|
| | $q=0$ | $q=1$ | $\lambda$ |
| $\rightarrow q_0$ | $q_3$ | $q_1$ | 0 |
| $q_1$ | $q_1$ | $q_2$ | 1 |
| $q_2$ | $q_2$ | $q_3$ | 0 |
| $q_3$ | $q_3$ | $q_0$ | 0 |

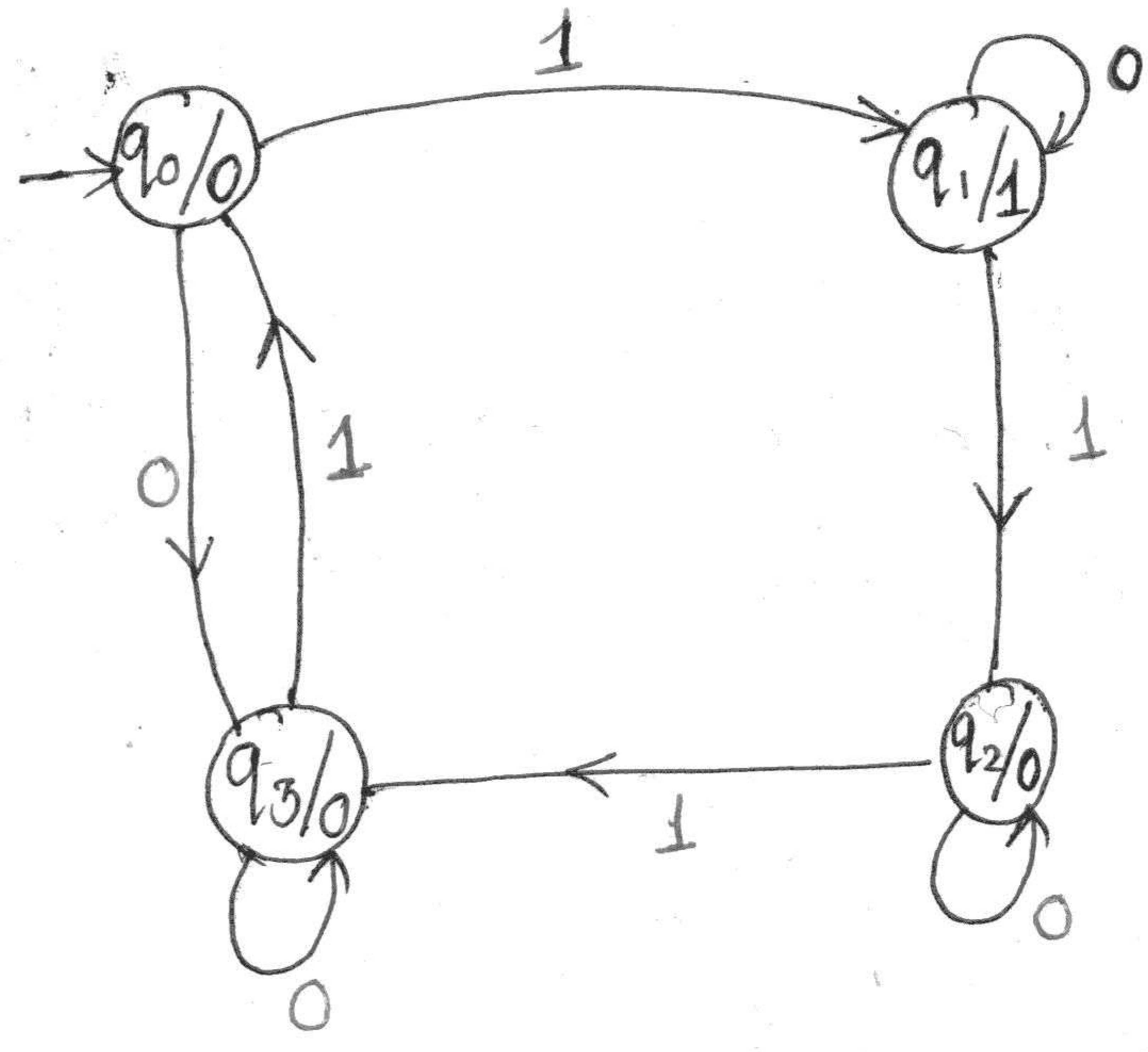
For the input string 0111, the transition of states is given by $q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$ the output string is 00001

For Moore Machine if the input string is of length n, the output string is of length n+1. The first output is $\lambda(q_0)$ for all output String.

and

In case of a Mealay Machine if the input string is of length n, the output string is also of the same length n.

Transition Diagram

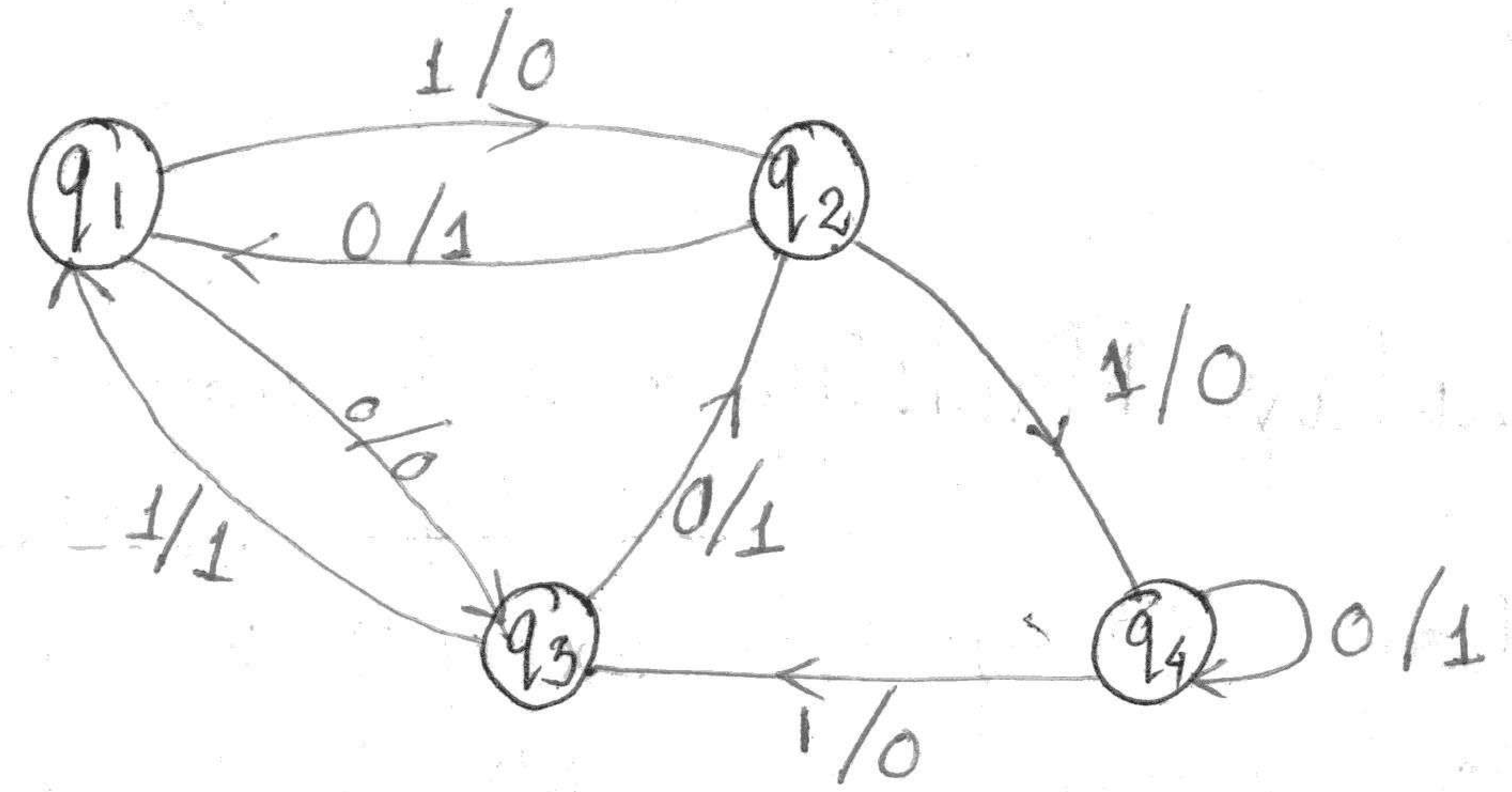



**Mealay Machine** ~ A Mealay Mechine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$. Where all symbols except $\lambda$ have the same meaning as in the Moore Machine. $\lambda$ is the output function mapping $\Sigma \times Q$ into $\Delta$

Example

| Present state | Next State | | | |
|---|---|---|---|---|
| | a=0 | | a=1 | |
| | State | output | State | Output |
| $\rightarrow q_1$ | $q_3$ | 0 | $q_2$ | 0 |
| $q_2$ | $q_1$ | 1 | $q_4$ | 0 |
| $q_3$ | $q_2$ | 1 | $q_1$ | 1 |
| $q_4$ | $q_4$ | 1 | $q_3$ | 0 |

For the input string 0011 the transition of states is given by $q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$ and the output string is 0100




Transition Diagram

Design a program for generating lexical analyzers :- X

Regular expression :- (RE)

→ first introduce a very usefull notation, called R.E., Suitable for describing tokens.

→ Later show how regular expression cab be converted automatically into finite automata, which is the formal specification of transition diagrams.

→ Then implementation of finite automata by program.

String and Language :-

A string is a finite sequence of symbols. Such as 001.

Ex set {0, 1} → consist two symbols 0 and 1. is also called alphabet.

programming language alphabets are the ASCII and EBCDIC character set.

→ ~~Sen~~ Sentence and word are synonyms for string.

→ the length of a string $x$, usually denoted by $|x|$ is the total no. of symbol in $x$.

Ex 0110 is string of length 4.

→ A special string is empty string which is denoted by $\varepsilon$ or $\lambda$. This is string of length zero.

→ If $x$ and $y$ are string the the concatenation of $x$ and $y$, written by $x \cdot y$

Ex. $x = \{001\}$ and $y = \{110\}$

the $xy$ = 001110

→ the concatenation of the empty string and any string is again the same string.

$$\varepsilon x = x \varepsilon = x.$$

→ In general $x^i$ is the string $x$ repeated $i$ times

Ex $x^2 = xx$.

$x^0$ to be $\varepsilon$ for any string $x$.

thus $\varepsilon$ plays the role of 1 (multiplicative identity)

→ the union of two RE $R_1$ and $R_2$ written as $R_1 + R_2$ is also R.E.

→ Concatenation of two RE $R_1$ and $R_2$ written in $R_1 R_2$ also a RE.

→ Language to mean any set of string formed by some specific alphabet.

→ $L_1$ = the set of all string of 0's and 1's ending with 00.

⇒ So cancadenation of {0,1} and 00 then {0,1} is represented by 0+1

So $RE = (0+1)^*00$

( $R^*$ is the iteration of R

→ $L_2$ = set of all string of 0's and 1's beginning with zero and ending with 1.

⇒ $0(0+1)^*1$

→ $L_3 = \{\wedge, 11, 1111, 111111 \text{---}\}$

$= (11)^*$ [∵ ∧ is even like 0 in Math]

→ $\{\wedge, 0, 00, 000 \text{--}\}$ Represent by $0^*$

→ $0^+ = \{0, 00, 000 \text{--}\}$ [∵ excluding ∧ from $0^*$