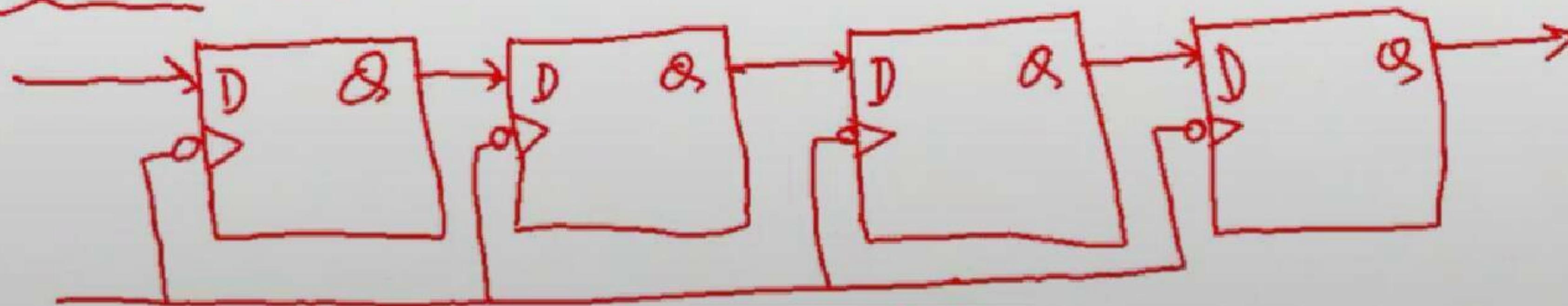


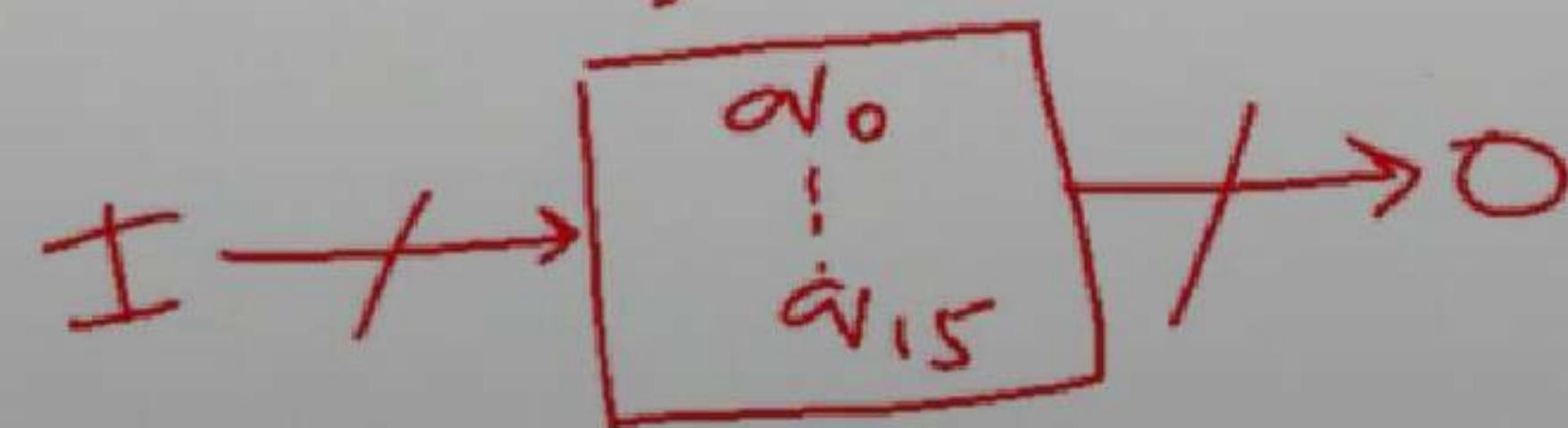
- (i) Output:  $O_1, O_2, \dots, O_n$  are the outputs which can take fixed values from the set  $O$ .
- (ii) States: At any instant the automaton can be in one of the states  $a_1, a_2, \dots, a_n$ .
- (iii) State relation: The next state is determined by the present state & present input.
- (iv) Output relation: The output is related to either state only or both the input & the state.

Example: Consider a shift register



There are  $2^4 = 16$  states  
( $0000 \rightarrow 1111$ )

$$\Sigma = \{0, 1\}, O = \{0, 1\}$$



## Definition of Finite Automata

An automata can be defined as five tuples

$\delta \rightarrow$  finite set of states

$\Sigma \rightarrow$  finite set of input symbols called alphabet

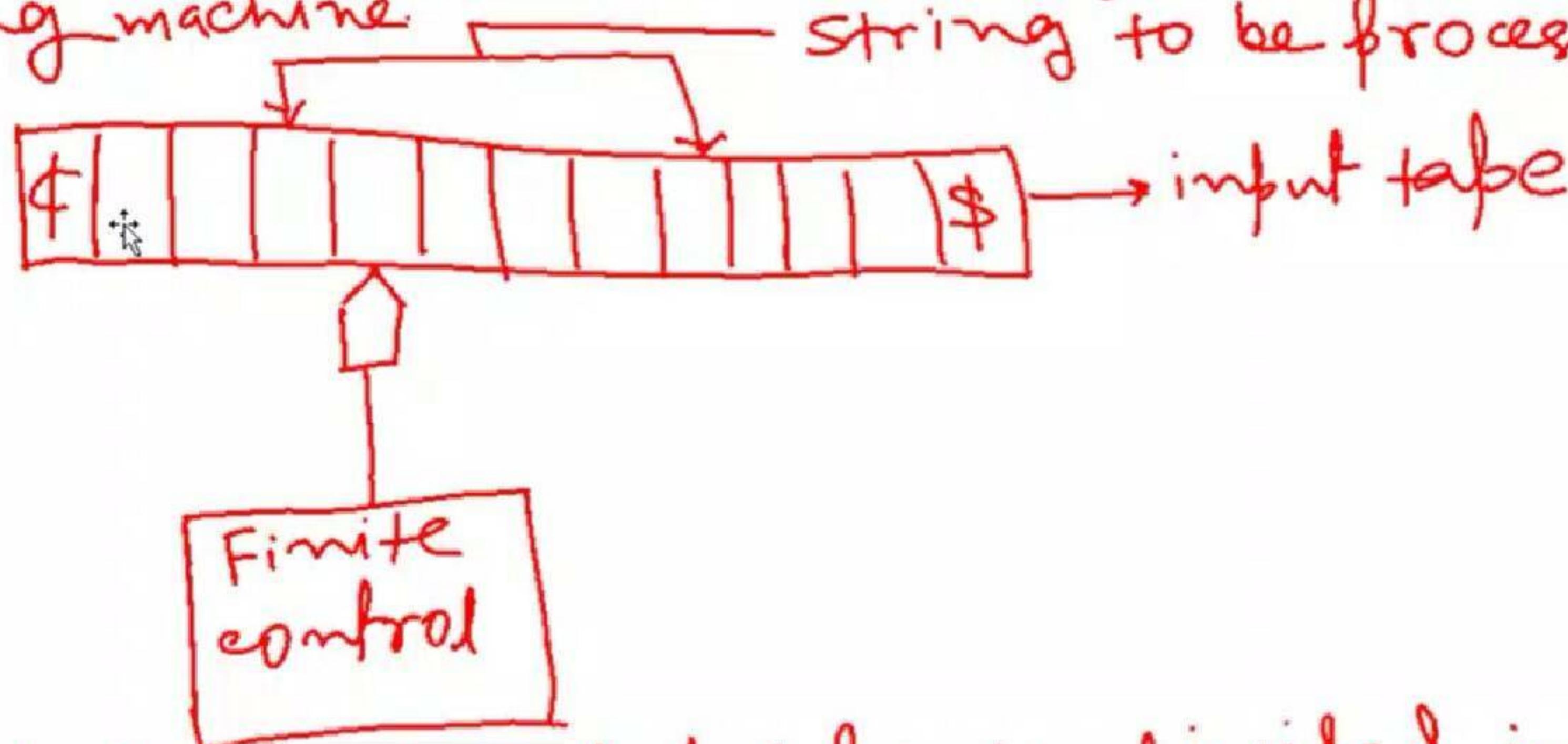
$\gamma \rightarrow$  is called transition function

$$\gamma: \delta \times \Sigma \longrightarrow \delta$$

$a_0 \rightarrow$  initial state, where  $a_0 \in \delta$

$F \subseteq \delta$  is the set of final states.

The automaton can be visualized as the following machine.



Input tape: Input tape is divided into squares, each square containing a symbol from the alphabet  $\Sigma$ .  
from the alphabet  $\Sigma$ .  
 $\$ \rightarrow$  start marker,  $\$ \rightarrow$  End marker

Reading head: reading head examines one symbol at a time.

(i.e. one square)

It can move either right or left. (We restrict the movement to the right for now).

Finite control: input to the finite control is a symbol under head and the present state of the machine.

- output of the finite control

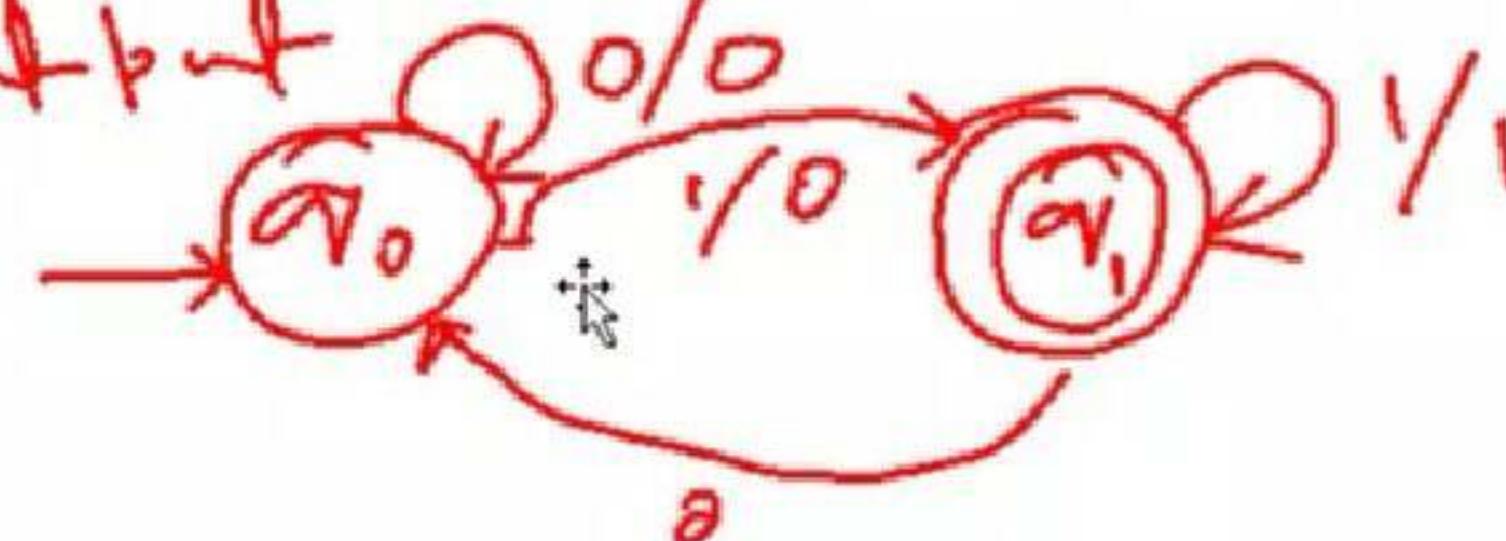
(a) A motion of the R-head to the next square or a null move.

(b) The next state is given by  $S(a, a)$ .

## Transition Systems

A transition graph or transition system is a visual representation of an automaton.

- A transition system consists of states and possible transitions shown by directed edges labelled by input/output



- States are represented as a single circle and final states are represented as double circle.

- The  $\delta$  is represented as

$$\emptyset \times \Sigma^* \times \emptyset$$

i.e.  $(\alpha_1, w, \alpha_2) \rightarrow$  means automation at state  $\alpha_1$ , after consuming the string  $w$  moves to  $\alpha_2$ .

Acceptance of a string

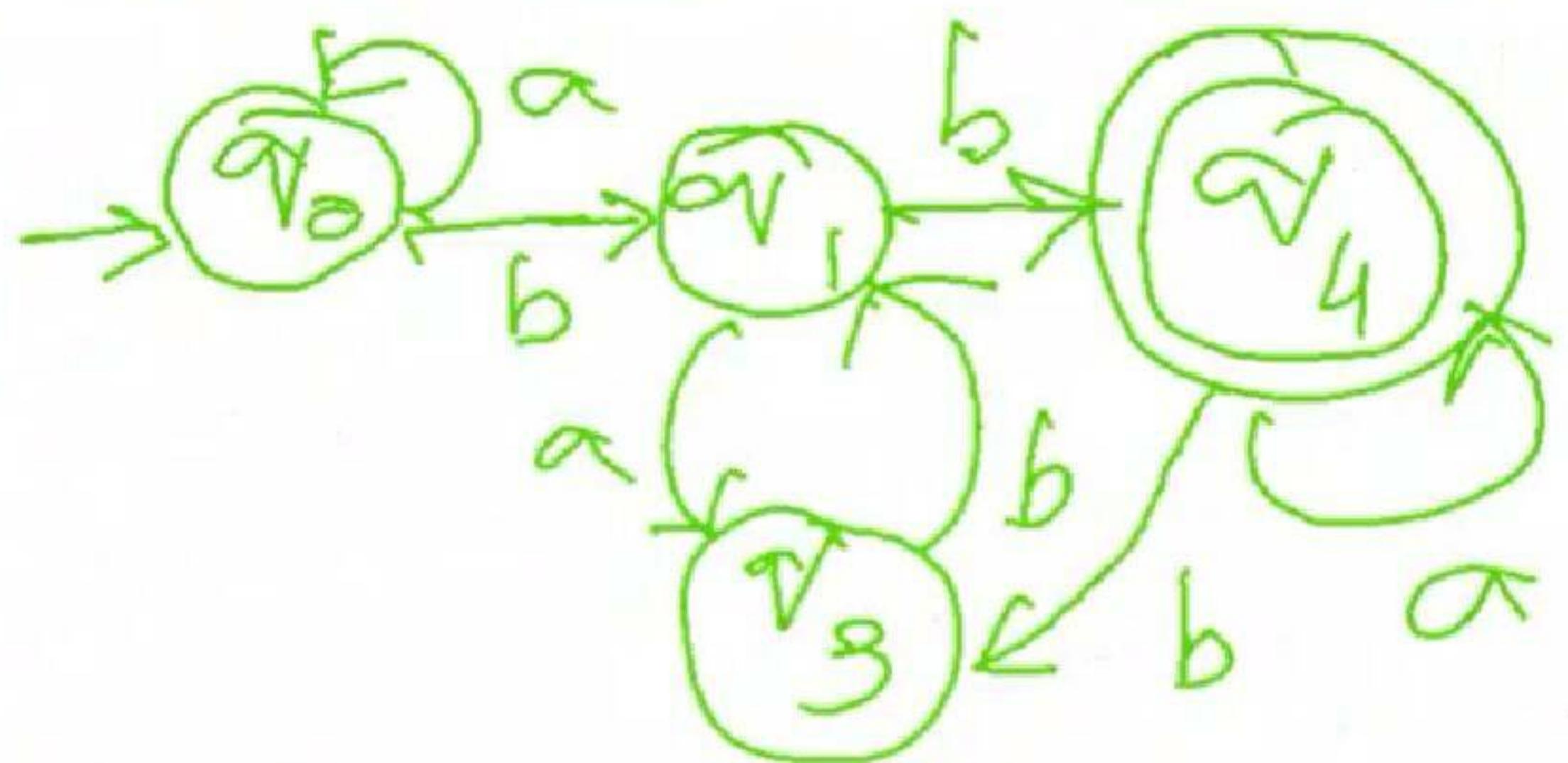
A transition system accepts a string  $w$  in  $\Sigma^*$  if

- (a) there exists a path from initial state to some final state labelled with symbols from  $w$ .
- (b) The path value obtained by concatenation of all edge labels of the path is equal to  $w$ .

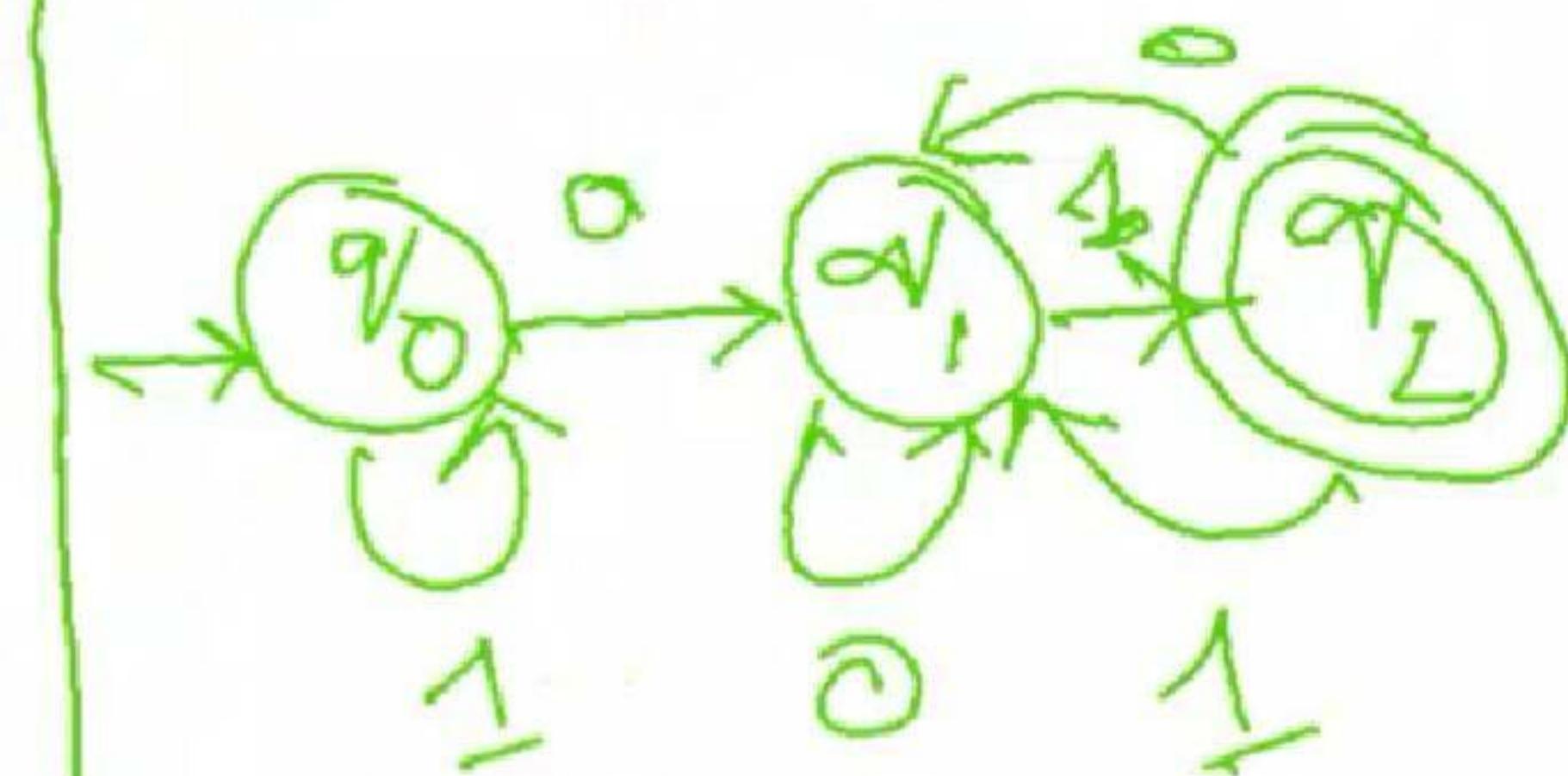
Example:



Determine acceptability 0/0  
of 101011, 111010



Define the language



test for the following

001111	1
011011	0
010000	1
111000	1
111100	0

## Properties of transition function

1.  $\delta(\alpha, \lambda) = \alpha$ , in a finite automaton

2. for all strings  $w$  and an input symbol  $a$

$$\delta(\alpha, aw) = \delta(\delta(\alpha, a), w)$$

$$\delta(\alpha, wa) = \delta(\delta(\alpha, w), a)$$

first one gives a state after consuming the first symbol of  $aw$ .

Second property gives a state after consuming the prefix of the string  $wa$ .

problem: Show that for a transition function  $\delta$  and any two strings  $x$  &  $y$

$$\delta(\alpha, xy) = \delta(\delta(\alpha, x), y)$$

Proof: Let  $|y|=1$  and  $y=\alpha \in \mathbb{Z}$

from second property

$$\text{L.H.S } \delta(\alpha, x\alpha) = \delta(\delta(\alpha, x), \alpha) = \text{R.H.S}$$

Let the rule is true for  $|y_1|=n$ , where  $y=y_1$  i.e.

$$\begin{aligned} |y| &= n+1 \\ \text{L.H.S } \delta(\alpha, x_{y_1}, \alpha) &= \delta(\alpha, x_1, \alpha) \\ &= \delta(\alpha, x_1, \alpha) \text{ where } x_1 = x_{y_1} \end{aligned}$$

from property 2,

$$\begin{aligned} \text{L.H.S. } \delta(\delta(\alpha, x_1), \alpha) \\ &= \delta(\delta(\alpha, x_{y_1}), \alpha) \text{ from induction} \end{aligned}$$

hypothesis

$$\text{L.H.S } \delta(\delta(\delta(\alpha, x), y_1), \alpha)$$

$$\text{R.H.S } \delta(\delta(\alpha, x), y) = \delta(\delta(\alpha, x), y_1, \alpha)$$

$$\text{From property 2, } = \delta(\delta(\delta(\alpha, x), y_1), \alpha)$$

Problem: PROVE that if  $\delta(\alpha, \gamma) = \delta(\gamma, \gamma)$ , then  $\delta(\alpha, \gamma z) = \delta(\alpha, \gamma z)$

Sol: 
$$\begin{aligned}\delta(\alpha, \gamma z) &= \delta(\delta(\alpha, \gamma), z) \\ &= \delta(\delta(\alpha, \gamma), z) \\ &= \delta(\alpha, \gamma z)\end{aligned}$$

Acceptability of string by a finite automaton

Definition: A string is said to be accepted by a finite automaton  $M = (Q, \Sigma, \delta, \alpha_0, F)$

if  $\delta(\alpha_0, x) = \gamma$ , for some  $\gamma \in F$ .

This is basically the acceptability of a string by the final state.

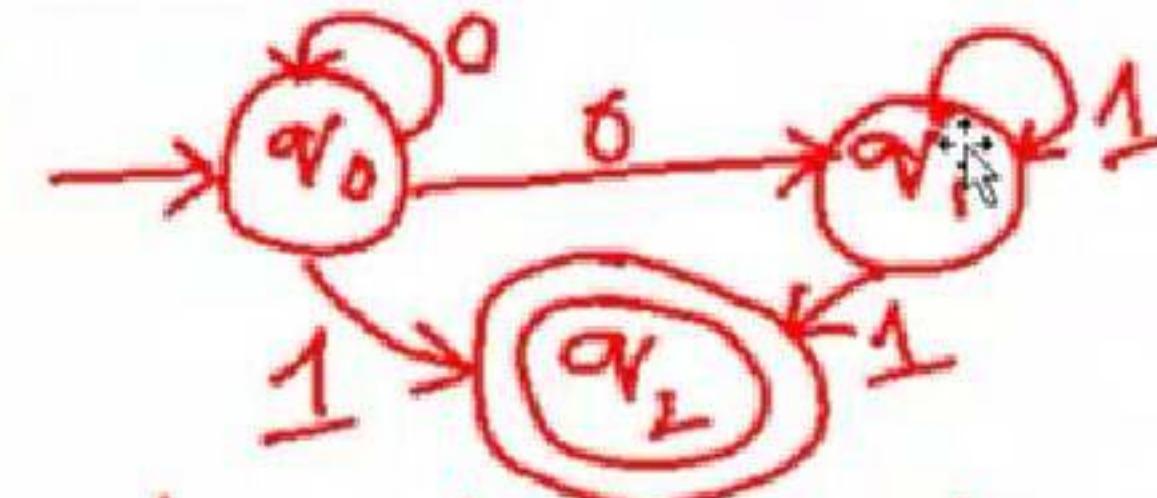
Problem: Consider a machine where  $\delta$  is defined as

state	input	
	0	1
$\alpha_0$	$\alpha_1$	$\alpha_1$
$\alpha_1$	$\alpha_2$	$\alpha_0$
$\alpha_2$	$\alpha_3$	$\alpha_0$
$\alpha_3$	$\alpha_1$	$\alpha_3$
$\alpha_4$	$\alpha_2$	$\alpha_2$

determine whether string 11001 is accepted or not  
 $F = \{\alpha_0\}$ ,  $\Sigma = \{0, 1\}$

## Non-deterministic finite state machine

Let us consider the following transition diagram (transition system)



- If the automaton is in a state  $\{q_0\}$  and the input symbol is 0, what will be the next state?
- Thus some moves of the machine cannot be determined uniquely by the input symbol and present state.

### Definition

A non-deterministic finite automaton (N DFA) is a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$ . The only difference is in  $\delta$ .

(i)  $Q$  is a finite non empty set of states.

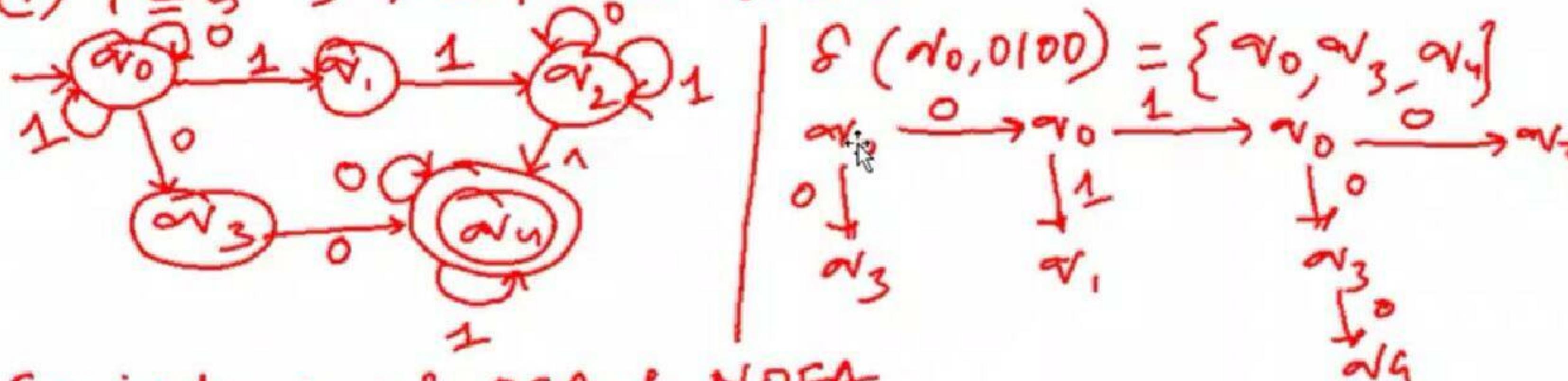
(ii)  $\Sigma$  is a finite non empty set of input symbols.

(iii)  $\delta$  is the transition function defined as

$$\delta: Q \times \Sigma \longrightarrow 2^Q$$

(iv)  $\alpha_0 \in Q$  is the initial state.

(v)  $F \subseteq Q$  is the final state.



Equivivalence of DFA & NDFA

- A DFA can simulate the behaviours of NDFA by increasing the number of states.

- In other words, a DFA  $(Q, \Sigma, \delta, \alpha_0, F)$  can be viewed as an NDFA  $(Q, \Sigma, \delta, \alpha_0, F)$  by defining  $\delta'(\alpha, a) = \{\delta(\alpha, a)\}$

NDFA is more general machine without being more powerful.

The transition for NDFA

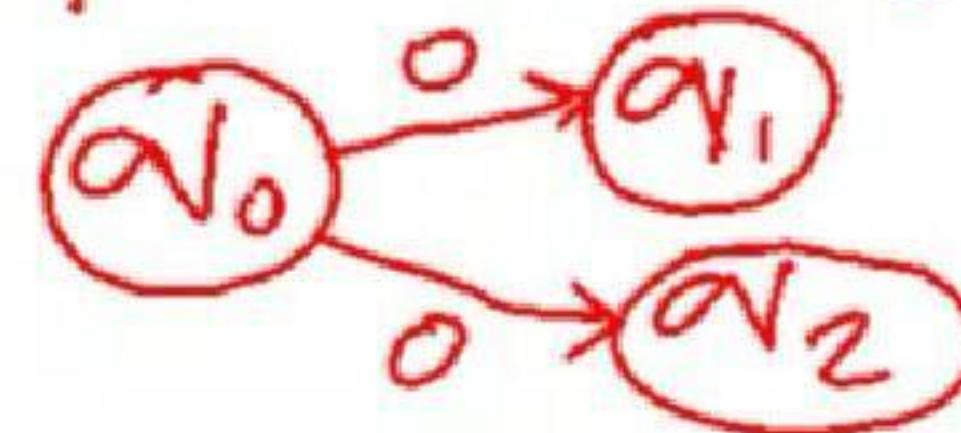
$$\delta : \Sigma \times Q \rightarrow 2^Q$$

The corresponding DFA

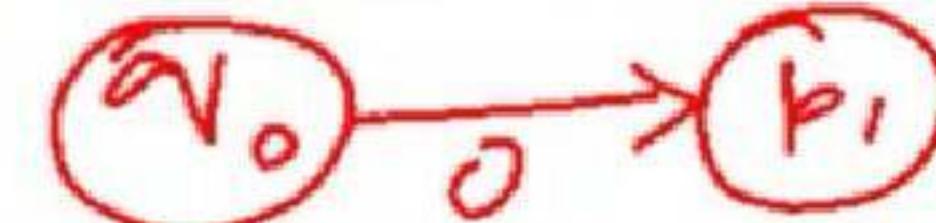
$$\delta' : \Sigma \times Q' \rightarrow Q'$$

$$Q' \subseteq 2^Q$$

gf for an NDFA



The the corresponding DFA will have



$$\text{where } p_1 = [q_1, q_2]$$

Theorem: For every NDFA, there exists a DFA which simulates the behaviour of NDFA.

Alternatively,

if  $L$  be the set of strings accepted by NDFA, then there exists a DFA which also accepts  $L$ .

Proof: Let  $M = (Q, \Sigma, S, \alpha_0, F)$  be an NDFA accepting  $L$ .  
Let us construct a DFA as follows

$$M' = (Q', \Sigma, S', \alpha'_0, F')$$

where  $Q' = 2^Q$  (Any state in  $Q'$  is denoted by  $(\alpha_1, \alpha_2, \dots, \alpha_j)$ )

(i)  $\alpha'_1 = 2^Q$  (Where  $\alpha_1, \alpha_2, \dots, \alpha_j \in Q$ )

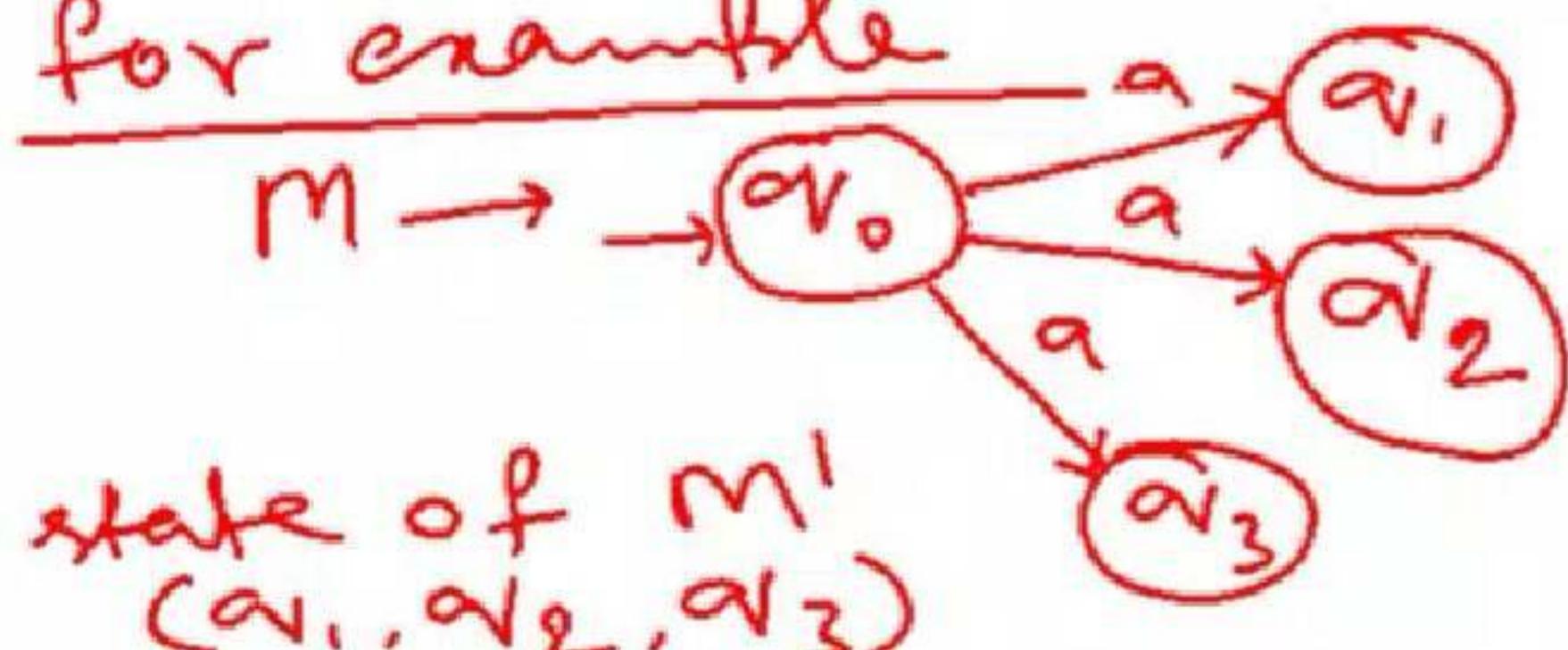
$$(ii) \alpha'_0 = [\alpha_0]$$

(iii)  $F'$  is the set of all subsets of  $Q$  containing an element of  $F$ .

- On applying any input  $a$  to  $M$  at state  $\alpha_0$ ,  $M$  can reach to state  $S(\alpha_0, a)$ .
- The state of corresponding DFA,  $M'$  on input  $a$  at

state  $\alpha'_0$  is a set of all states that can be reached by  $\delta(\alpha_0, a)$  on  $M$ .

for example



- Hence the states of  $M'$  are defined as subsets of  $Q$ . The  $M'$  should also start with the initial state of  $\alpha'_0$ , Hence  $\alpha'_0 = [\alpha_0]$
- On processing any string  $w$  by  $M$ , if  $M$  reaches a final state then  $w \in T(M)$
- Hence final state in  $M'$  should be any of the final states of  $M$ .

(iv) therefore,  $\delta'$  can be defined as

$$\delta'([\alpha_1, \alpha_2, \dots, \alpha_j], a) = \delta(\alpha_1, a) \cup \delta(\alpha_2, a) \cup \dots \cup \delta(\alpha_j, a)$$

Now let us prove that

$$\delta'(\alpha'_0, x) = [\alpha_1, \alpha_2, \dots, \alpha_i] \text{ iff } \delta(\alpha_0, x) = [\alpha_1, \alpha_2, \dots, \alpha_i]$$

- we prove it by induction on  $|x|$

when  $|x|=0$ ,  $\delta(\alpha_0, \lambda) = \alpha_0$

and by definition of  $\delta'$

$$\delta'(\alpha'_0, \lambda) = \alpha'_0 = \alpha_0$$

hence  $\delta'(\alpha'_0, x) = \alpha_i$  if  $\delta(\alpha_0, x) = \alpha_i$  for  $|x|=0$

Let the hypothesis is true for all strings  $y$  with  $|y| \leq m$

Let  $\delta(\alpha_0, y) = [p_1, p_2, \dots, p_i]$  &  $\delta(\delta(\alpha_0, y), a) = [r_1, r_2, \dots, r_k]$

where  $|y| = m+1$  and  $|y| = m$

$$\delta([p_1, p_2, \dots, p_i], a) = [r_1, r_2, \dots, r_k]$$

Now let us see

$$\delta'(\alpha'_0, \gamma a) = ?$$

We have  $\delta'(\alpha'_0, \gamma) = [p_1, p_2, \dots, p_i] = \delta(\alpha_0, \gamma)$

from hypothesis

$$\begin{aligned}\delta'(\alpha'_0, \gamma a) &= \delta'(\delta'(\alpha'_0, \gamma), a) \\&= \delta'([p_1, p_2, \dots, p_i], a) \\&= \delta'(p_1, a) \cup \delta'(p_2, a) \dots \cup \delta'(p_i, a) \\&= \delta(p_1, a) \cup \delta(p_2, a) \dots \cup \delta(p_i, a) \\&= [\tau_1, \tau_2, \dots, \tau_K]\end{aligned}$$

Hence it is proved for  $\kappa = \gamma a$ .

Similarly,

$$\text{if } \delta'(\alpha'_0, \kappa) = [\alpha_1, \alpha_2, \dots, \alpha_i]$$

$$\text{then } \delta'(\alpha_0, \kappa) = [\alpha_1, \alpha_2, \dots, \alpha_i]$$

$$\delta'(\alpha'_0, \gamma) = [p_1, p_2, \dots, p_i]$$

from hypothesis

$$\delta(\alpha_0, \gamma) = [b_1, b_2, \dots, b_i]$$

$$\delta'(\delta'(\alpha_0, \gamma), \alpha) = \delta'([b_1, b_2, \dots, b_i], \alpha) = [r_1, r_2, \dots, r_k] - ①$$

① is possible only if

$$\delta(b_1, \alpha) = r_1, \delta(b_2, \alpha) = r_2, \dots, \delta(b_i, \alpha) = r_k$$

Since from definition

$$\delta'([b_1, b_2, \dots, b_i], \alpha) = \delta(b_1, \alpha) \cup \delta(b_2, \alpha) \cup \dots \cup \delta(b_i, \alpha)$$

Problem: construct a deterministic automaton equivalent to  
 $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, [q_0]), S$  is given as follows

state / $\Sigma$	0	1
$\rightarrow q_0$	$q_0$	$q_1$

- Sol : For deterministic automaton  $M_1$ ,
- (i) The states are subsets of  $\{q_0, q_1\}$  i.e.  $\phi, [q_0], [q_1], [q_0, q_1]$ .
  - (ii) Final states are  $[q_0]$  and  $[q_0, q_1]$ .
  - (iii)  $[q_0]$  is the initial state.
  - (iv)  $\delta_1$  for  $M_1$  is defined as,

$s_{\text{state}} / \Sigma$	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$a_0$	$a_0$	$a_1$
$a_1$	$a_1$	$[a_0, a_1]$
$[a_0, a_1]$	$[a_0, a_1]$	$[a_0, a_1]$

$$\delta_s([a_0, a_1], 0) = \delta(a_0, 0) \cup \delta(a_1, 0) = [a_0, a_1]$$

Similarly

$$\delta_s([a_0, a_1], 1) = \delta(a_0, 1) \cup \delta(a_1, 1) = [a_1, a_0, a_1] = [a_0, a_1]$$

No new states are there, hence we stop here.

When  $M$  has  $n$  states, the corresponding deterministic automaton has  $2^n$  states. However, we need not construct transition ( $\delta_s$ ) for all  $2^n$  states, but only for those states reachable from  $[a_0]$ .

efate/ $\Sigma$

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$a_0$	$a_0$	$a_1$
$a_1$	$a_1$	$[a_0, a_1]$
$[a_0, a_1]$	$[a_0, a_1]$	$[a_0, a_1]$

$$\delta([a_0, a_1], 0) = \delta(a_0, 0) \cup \delta(a_1, 0) = [a_0, a_1]$$

Similarly

$$\delta_1([a_0, a_1], 1) = \delta(a_0, 1) \cup \delta(a_1, 1) = [a_1, a_0, a_1] = [a_0, a_1]$$

No new states are there, hence we stop here.

When  $M$  has  $n$  states, the corresponding deterministic automaton has  $2^n$  states. However, we need not construct transition ( $\delta_1$ ) for all  $2^n$  states, but only for those states reachable from  $[a_0]$ .

Problem: Find a deterministic automaton (acceptor) equivalent to  $M = (\{v_0, v_1, v_2\}, \{a, b\}, \delta, v_0, \{v_2\})$  and  $\delta$  is defined as

state / $\Sigma$	a	b
$v_0$	$v_0, v_1$	$v_2$
$v_1$	$v_0$	$v_1$
$v_2$	$\emptyset$	$v_0, v_1$

Sol: set the deterministic automaton  $M'$  equivalent to  $M$  is defined as

$$M' = (2^S, \{a, b\}, \delta', [v_0], F')$$

where 'possible  $F'$  is  $\{[v_2], [v_0, v_2], [v_1, v_2], [v_0, v_1, v_2]\}$

$$\delta'(\alpha_0, a) = \delta(\alpha_0, a) = [\alpha_0, \alpha_1]$$

$$\delta'(\alpha_0, b) = \delta(\alpha_0, b) = [\alpha_2]$$

$$\delta'(\alpha_1, a) = \delta(\alpha_1, a) = \alpha_0$$

$$\delta'(\alpha_1, b) = \delta(\alpha_1, b) = \alpha_1$$

$$\delta'(\alpha_2, a) = \delta(\alpha_2, a) = \emptyset$$

$$\delta'(\alpha_2, b) = \delta(\alpha_2, b) = [\alpha_0, \alpha_1]$$

$$\delta'([\alpha_0, \alpha_1], a) = \delta(\alpha_0, a) \cup \delta(\alpha_1, a) = [\alpha_0, \alpha_1] \cup [\alpha_0]$$

$$= [\alpha_0, \alpha_1]$$

$$\delta'([\alpha_0, \alpha_1], b) = \delta(\alpha_0, b) \cup \delta(\alpha_1, b) = [\alpha_2] \cup [\alpha_1]$$

$$= [\alpha_1, \alpha_2]$$

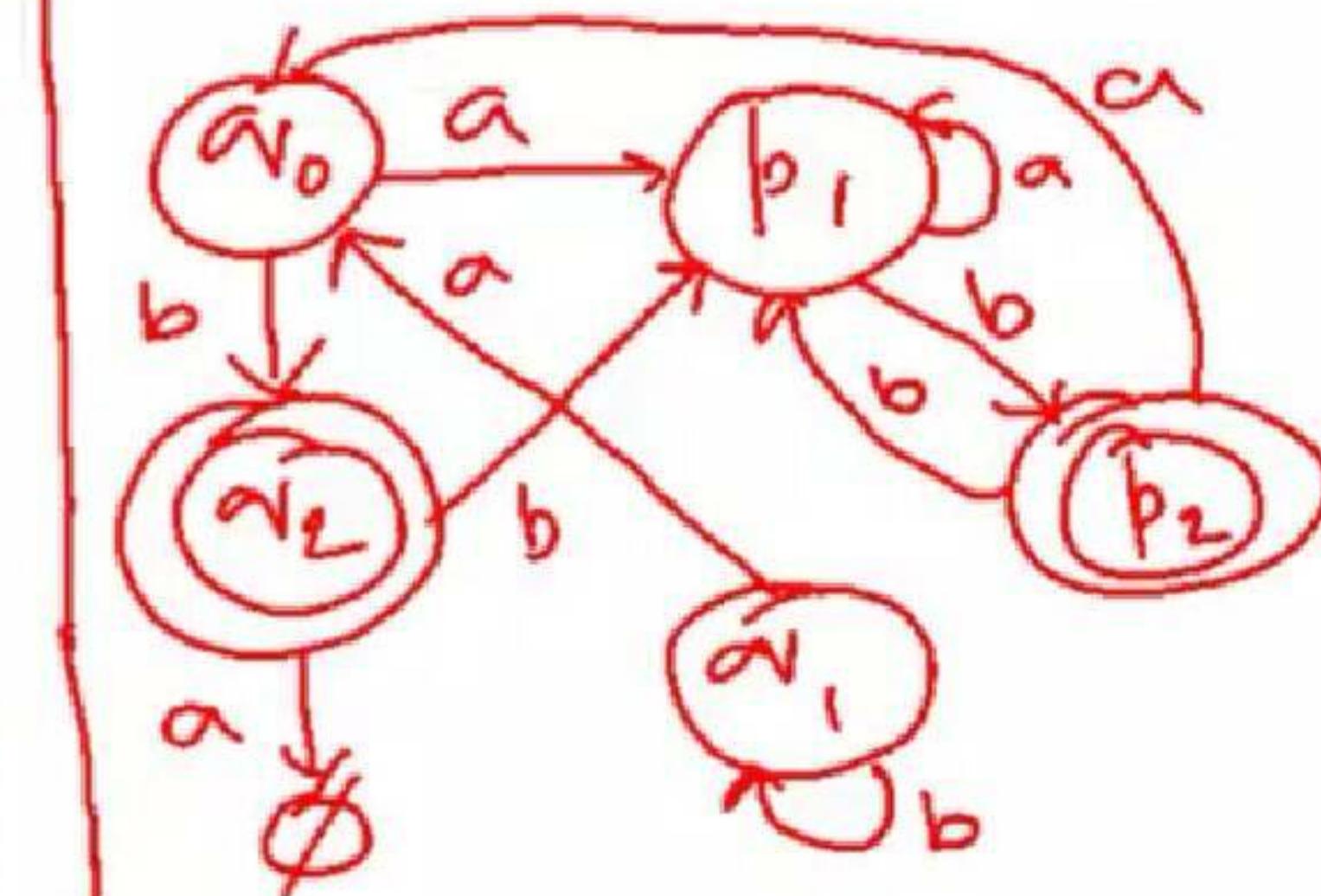
$$\delta'([\alpha_1, \alpha_2], a) = \delta(\alpha_1, a) \cup \delta(\alpha_2, a) = [\alpha_0] \cup \emptyset = [\alpha_0]$$

$$\delta'([\alpha_1, \alpha_2], b) = \delta(\alpha_1, b) \cup \delta(\alpha_2, b) = [\alpha_1] \cup [\alpha_0, \alpha_1] \\ = [\alpha_0, \alpha_1]$$

So the  $\delta'$  can be defined as states/ $\Sigma$

	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\alpha_0$	$[\alpha_0, \alpha_1]$	$\alpha_2$
$\alpha_1$	$\alpha_0$	$\alpha_1$
$\alpha_2$	$\emptyset$	$\alpha_2$
$[\alpha_0, \alpha_1]$	$[\alpha_0, \alpha_1]$	$[\alpha_0, \alpha_1]$
$[\alpha_1, \alpha_2]$	$[\alpha_0]$	$[\alpha_0, \alpha_1]$

Let  $[\alpha_0, \alpha_1] = p_1$  &  
 $[\alpha_1, \alpha_2] = p_2$



## Finite automata with output

- The acceptability/rejection these two outputs were possible for the machines we considered so far.
- Now, let us consider the machines which do not have restriction over reachability to final state and output can be chosen from some alphabet
- This output function is dependent either on current state and current input both or current state only.

i.e. if  $\lambda$  is the output function then

either  $\lambda(s(t), x(t)) \rightarrow$  (Mealy machine generalized)  
or  $\lambda(s(t)) \rightarrow$  (moore machine restricted)

where  $s(t)$  is the current state  
and  $x(t)$  is current input.

## Moore Machine

moore machine is defined using 6-tuples

$$(\mathcal{Q}, \Sigma, \Delta, \delta, \gamma, q_0)$$

$\mathcal{Q}$  → finite state

$\Sigma$  → input alphabet

$\Delta$  → output alphabet

$\delta$  → transition function

$$\Sigma \times \mathcal{Q} \rightarrow \mathcal{Q}$$

$\gamma$  → output function

$$\gamma: \mathcal{Q} \rightarrow \Delta$$

$q_0$  → initial state

Example

Present state

Next state

output

	$a=0$	$a=1$	
$\rightarrow \alpha_0$	$\alpha_3$	$\alpha_1$	0
$\alpha_1$	$\alpha_1$	$\alpha_2$	1
$\alpha_2$	$\alpha_2$	$\alpha_3$	0
$\alpha_3$	$\alpha_3$	$\alpha_0$	0

for input string 0111 the transition states given

by  
output is  $\left| \begin{array}{ccccc} \alpha_0 \rightarrow \alpha_3 \rightarrow \alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \\ 0 & 0 & 0 & 1 & 0 \end{array} \right.$

## Mealy Machine

The Mealy machine also have 6-tuples, the only difference is in  $\gamma$  defined as

$$\gamma: Q \times \Sigma \rightarrow \Delta$$

Example

Present state

Next state

Present state			Next state	
	$a=0$	$a=1$	state	o/p
	state	o/p	state	o/p
$\rightarrow q_1$	$q_3$	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 i/p 0011 we have

$$q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$$

o/p 0 1 0 0

Note: In Mealy machine we get the output only  
on the input to the machine state

- A finite automaton can be converted into a Moore machine by introducing  $\Delta = \{a, b\}$  and defining  $\gamma(a) = a$ , if  $a \in F$  and  $\gamma(a) = b$  if  $a \notin F$
- Moore machine has output string of length m+1 for i/p of length n.
- Mealy has the same o/p length as the i/p length.

for the i/p 0011 we have

$$q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$$

o/p 0 1 0 0

Note: In Mealy machine we get the output only  
on the input to the machine state

- A finite automaton can be converted into a Moore machine by introducing  $\Delta = \{a, b\}$  and defining  $\gamma(a) = a$ , if  $a \in F$  and  $\gamma(a) = b$  if  $a \notin F$
- Moore machine has output string of length m+1 for i/p of length n.
- Mealy has the same o/p length as the i/p length.

associated with each state  $a_i$  in the next state column.  
The state  $a_i$  then split into the number of states equal to the outputs associated with  $a_i$  in Mealy machine.

In the given Mealy machine  $a_1, a_3$  are associated outputs 1 and 0 respectively.

$a_2, a_4$  are associated with 0,1.

Hence we split  $a_2$  into  $a_{20}, a_{21}$  and  $a_4$  into  $a_{40}$  and  $a_{41}$ .

Hence the table is modified as shown in the next slide.



Present state

Next state

	input estate	$a = 0$ output		input estate	$a = 1$ output
$\rightarrow \alpha_1$	$\alpha_3$	0		$\alpha_{20}$	0
$\alpha_{20}$	$\alpha_1$	1		$\alpha_{40}$	0
$\alpha_{21}$	$\alpha_1$	1		$\alpha_{40}$	0
$\alpha_3$	$\alpha_{12}$	1		$\alpha_1$	1
$\alpha_{40}$	$\alpha_{41}$	1		$\alpha_3$	0
$\alpha_{41}$	$\alpha_3$	1		$\alpha_3$	0

The previous table can be rearranged as

Present state

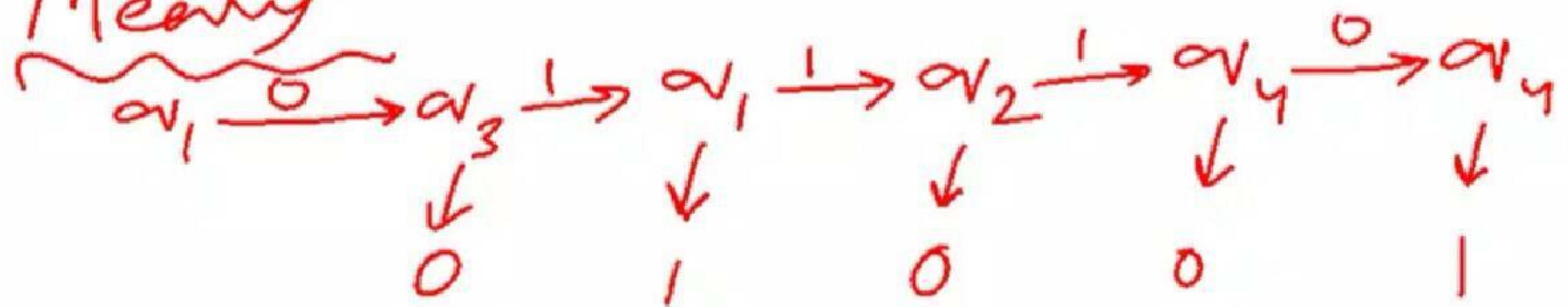
Next state

Present state	Input $a = 0$		Input $a = 1$		Output
	input	$a = 0$	$a = 1$	output	
$\rightarrow a_1$		$a_3$	$a_{20}$	1	
$a_{20}$		$a_1$	$a_{40}$	0	
$a_{21}$		$a_1$	$a_{40}$	1	
$a_3$		$a_2$	$a_1$	0	
$a_{40}$		$a_{41}$	$a_3$	0	
$a_{41}$		$a_{41}$	$a_3^*$	1	

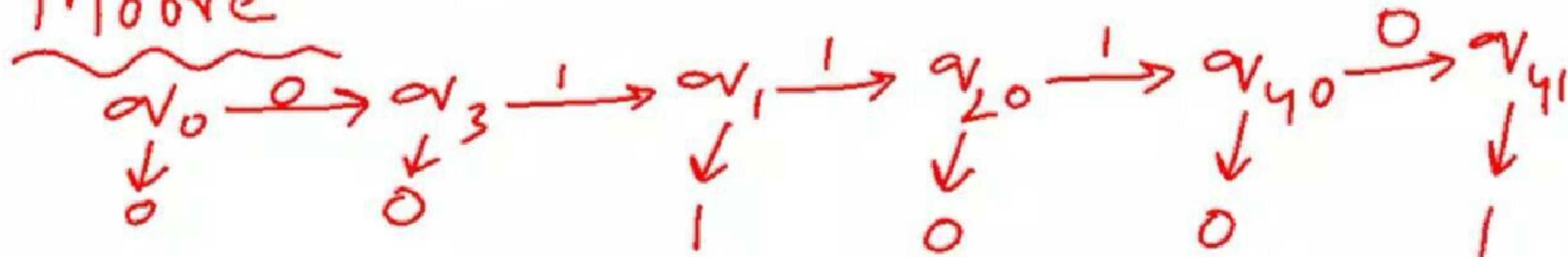
Now this Moore machine accepts a null string as it outputs 1 for  $\lambda$  with initial state  $a_1$ . In Mealy machine  $\lambda$  string is not allowed. Hence, either we can neglect the response for

Example: Show how output is generated for the input 01110 by Mealy and the corresponding Moore machines of previous example.

Mealy



Moore



Procedure to transform Moore into Mealy

A moore machine  $M'$  is equivalent to a Mealy machine  $M$ ,

$$\text{if } bZ_M(w) = Z_{M'}(w)$$

where  $Z(\cdot)$  is the output string function for  $M$  and  $M'$  for input string  $w$ , and  $b$  is the output for a string accepted by Moore machine.

Let the Moore machine

$$M' = (\Sigma, \Delta, \delta, s_0, \lambda, t_0)$$

construction

We need to define the output function  $\gamma$  of Mealy machine using output function  $\lambda$  of Moore machine as  $\gamma(s(a), a) = \lambda(s(a), a)$

Where  $\delta \rightarrow$  is the transition function of  $M'$ .

Problem: Construct an equivalent Mealy machine  
for Moore machine given below.

P. State	M. State		O/P
	$a=0$	$a=1$	
$\xrightarrow{a_0}$	$a_3$	$a_1$	0
$a_1$	$a_1$	$a_2$	1
$a_2$	$a_2$	$a_3$	0
$a_3$	$a_3$	$a_0$	0

Solution: For  $a_0$  in Moore machine and  
next state  $a_3$  output is

$$\gamma'(a_0, 0) = \gamma(\delta(a_0, 0)) = \gamma(a_3) = 0$$

As  $\alpha_3$  is the state for input 0 at  $\alpha_0$ .

Similarly,

$$\gamma'(\alpha_0, 1) = \gamma(\delta(\alpha_0, 1)) = \gamma(\alpha_1) = 1$$

For present state  $\alpha_1$  in Moore machine

$$\gamma'(\alpha_1, 0) = \gamma(\delta(\alpha_1, 0)) = \gamma(\alpha_1) = 1$$

$$\gamma'(\alpha_1, 1) = \gamma(\delta(\alpha_1, 1)) = \gamma(\alpha_2) = 0$$

For present state  $\alpha_2$  in Moore machine

$$\gamma'(\alpha_2, 0) = \gamma(\delta(\alpha_2, 0)) = \gamma(\alpha_2) = 0$$

$$\gamma'(\alpha_2, 1) = \gamma(\delta(\alpha_2, 1)) = \gamma(\alpha_3) = 0$$

For  $\alpha_3$  in Moore

$$\gamma'(\alpha_3, 0) = \gamma(\delta(\alpha_3, 0)) = \gamma(\alpha_3) = 0$$

$$\gamma'(\alpha_3, 1) = \gamma(\delta(\alpha_3, 1)) = \gamma(\alpha_0) = 0$$

So the transition of the corresponding Mealy machine

P. State

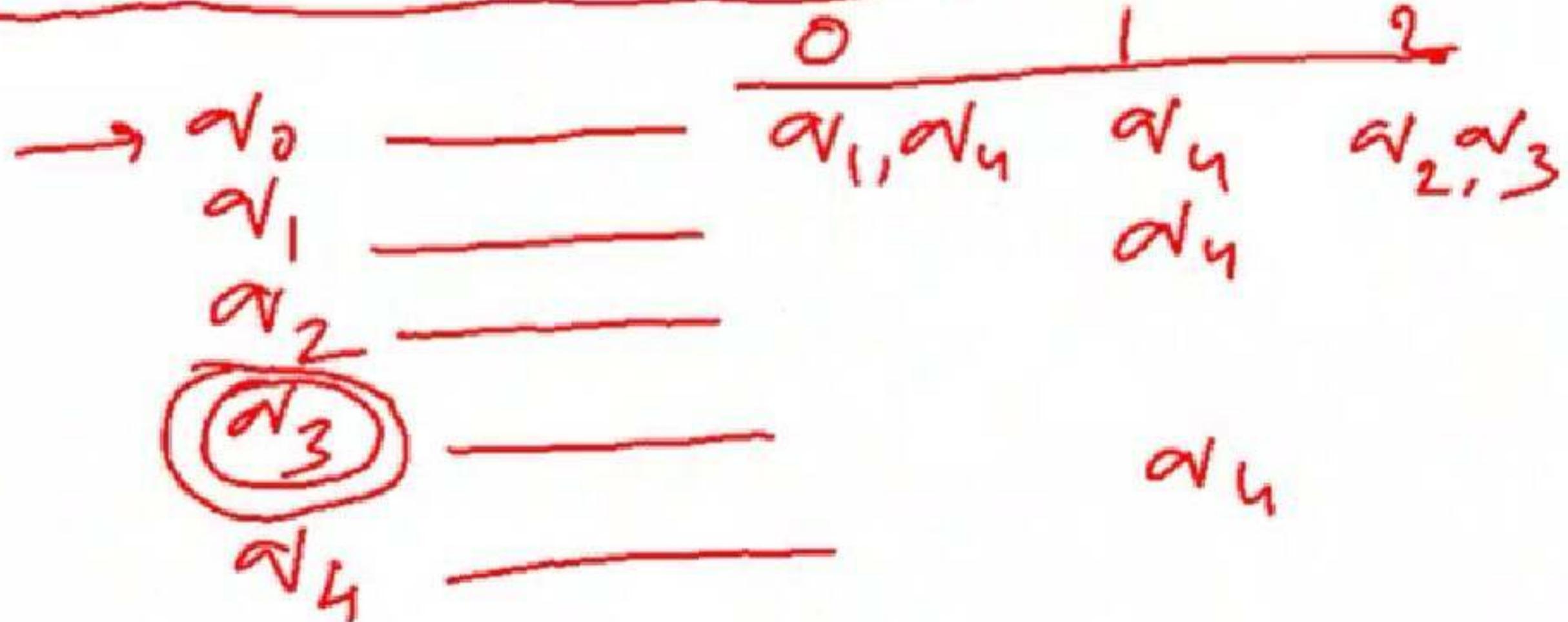
M. State

input a=0		input a=1	
state	o/p	state	o/p
$\alpha_0$	$\alpha_3$	$\alpha_1$	1
$\alpha_1$	1	$\alpha_2$	0
$\alpha_2$	0	$\alpha_3$	0
$\alpha_3$	0	$\alpha_0$	0

- Q) Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a finite automaton.  
Let  $R$  be a relation in  $Q$ , defined by  
 $q_1 R q_2$ , if  $\delta(q_1, a) = \delta(q_2, a)$ , for some  $a \in \Sigma$ .  
Is  $R$  an equivalent relation.
- Q) Construct an N DFA accepting  $\{ab, ba\}$  and  
use it to find a deterministic automaton  
accepting the same string.
- Q) The transition table for an N DFA is  
in the next slide. construct the equiv -  
ivalent DFA

state

input



## State minimization in automaton

The objective is to construct an automaton  $M'$  with minimum number of states from the given automaton  $M$ .

Definition: Two states are called equivalent i.e.  $\alpha_1$  and  $\alpha_2$  are  $\alpha_1 \equiv \alpha_2$ , if either both  $\delta(\alpha_1, x) \neq \delta(\alpha_2, x)$  are final states, or both are non final states,  
 $\forall x \in \Sigma^*$ .

Note: It is very difficult to check for all  $x \in \Sigma^*$  whether  $\delta(\alpha_1, x) \neq \delta(\alpha_2, x)$  are final or non final states.

## modified definition

Two states  $a_1$  and  $a_2$  are  $K$  equivalent ( $\epsilon \geq 0$ ) if both  $\delta(a_1, x)$  and  $\delta(a_2, x)$  are either non final or final states,  $x, x \in \Sigma^*$  such that  $|x| \leq K$ .

Base case: Every final or non final states are 0-equivalent, i.e. any two final or non final states are 0-equivalent.

## Properties of $K$ -equivalence relations

1. These  $K$ -equivalence relations are equivalence relations, i.e. reflexive, symmetric, & transitive.

2. These  $k$ -equivalence relations create partition over  $\Omega$ . These partitions can be denoted as  $\Pi$  or  $\Pi_k$  where  $\Pi_k$  denotes partitions of  $k$ -equivalence classes.
3. If  $a_1, f a_2$  are  $k$  equivalent for all  $k > 0$ , then they are equivalent.
4. If  $a_1, f a_2$  are  $(k+1)$  equivalent then they are  $k$ -equivalent.
5.  $\Pi_n = \Pi_{n+1}$ ,  $\exists n$  denotes that there can be no further partitions.  
( $\Pi_n$  denotes the set of equivalence classes under  $n$ -equivalence)

Theorem: If  $\alpha_1$  &  $\alpha_2$  are (a) K-equivalent &  
(b)  $s(\alpha_1, \alpha)$  &  $s(\alpha_2, \alpha)$  are also equivalent,  $\forall \alpha \in \Sigma$   
Then  $\alpha_1$  &  $\alpha_2$  are  $K+1$  equivalent.

Proof: We prove this by contradiction.  
Let us consider that  $\alpha_1$ , and  $\alpha_2$  are not  
 $(K+1)$  equivalent.

Let  $|aw_1| = K+1$ , where  $|w_1| = K$ .  
As  $\alpha_1$  &  $\alpha_2$  are not  $K+1$  equivalent, hence  
we can assume

$s(\alpha_1, aw_1) \rightarrow$  gives a non final state.

$s(\alpha_2, aw_1) \rightarrow$  gives a final state.

As we know  
 $\delta(a, xy) = \delta(\delta(a, x), y)$

hence  
 $\delta(\alpha_1, aw_1) = \delta(\delta(\alpha_1, a), w_1)$

&  
 $\delta(\alpha_2, aw_2) = \delta(\delta(\alpha_2, a), w_2)$

So,  $\delta(\underline{\delta(\alpha_1, a)}, w_1)$  &  $\delta(\underline{\delta(\alpha_2, a)}, w_2)$  are

$\downarrow$   $\uparrow$   
 $p_1$   $p_2$   
non final & final states respectively.

As  $|w_1| = k$ ,  $\delta(\alpha_1, a)$  &  $\delta(\alpha_2, a)$  are not  
K-equivalent, which is a contradiction.  
Hence  $\alpha_1$  &  $\alpha_2$  has to be  $k+1$  equivalent for  
 $\delta(\alpha_1, a)$  &  $\delta(\alpha_2, a)$  to be K equivalent.

## Algorithm to construct minimum state automation

Step 1: construct  $\Pi_0$ , where  $\Pi_0 = [Q_1^0, Q_2^0]$  and  $Q_1^0$  is the set of all final states &  $Q_2^0 = Q - Q_1^0$ .

Step 2: Let  $Q_i^K$  be  $K$ -equivalent class i.e.

$Q_i^K \subseteq \Pi_K$ . If  $o_1$  and  $o_2$  are in  $Q_i^K$  they are  $(K+1)$  equivalent provided  $s(o_1, a)$  &  $s(o_2, a)$  are  $k$ -equivalent i.e.

$$\{s(o_1, a), s(o_2, a)\} \subseteq \Pi_K$$

Step 3: construct  $T_m$  for  $m = 1, 2, 3, \dots$  until  $T_m = T_{m+1}$   
 The minimum automaton  $M$  is the  $m$ -equivalent partition i.e.  $T_m$

(i) construct a minimum automaton from the given automaton.

States / $\Sigma$	0	1
$a_0$	$a_1$	$a_5$
$a_1$	$a_b$	$a_2$
$a_2$	$a_0$	$a_2$
$a_3$	$a_2$	$a_6$
$a_4$	$a_7$	$a_5$
$a_5$	$a_2$	$a_6$
$a_6$	$a_6$	$a_4$
$a_7$	$a_6$	$a_2$

Solution: By applying step 1

$$\mathcal{Q}_1^0 = F = \{\alpha_2\}$$

$$\mathcal{Q}_2^0 = Q - \mathcal{Q}_1^0 = \{\alpha_0, \alpha_1, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7\}$$

$$\Pi_0 = \{\mathcal{Q}_1^0, \mathcal{Q}_2^0\} = \{\{\alpha_2\}, \{\alpha_0, \alpha_1, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7\}\}$$

$\{\alpha_2\}$  in 0th partition  $\Pi_0$  cannot be partitioned further. So  $\{\alpha_2\}$  will remain same in 1st partition  $\Pi_1$ , i.e. 1st equivalent class

$$\text{in } \Pi_1 \text{ is } \mathcal{Q}_1^1 = \{\alpha_2\}$$

Consider  $\alpha_0, \alpha_1$  in  $\mathcal{Q}_2^0$  for 0-column in transition table, we have  $\alpha_1, \alpha_6$  and for 1-Column we have  $\alpha_5, \alpha_2$ .

Where  $\alpha_1 \in Q_2^0$  &  $\alpha_6 \in Q_2^0$

Similarly  $\alpha_5 \in Q_2^0$  &  $\alpha_2 \in Q_1^0$ .

Hence,  $s(\alpha_0, 1)$  &  $s(\alpha_1, 1)$  are not 0-equivalent,  
so they are not 1-equivalent.

Similarly,  $\alpha_0$  is not 1-equivalent with  $\alpha_3$ ,  
 $\alpha_5$  &  $\alpha_7$ .

Consider  $\alpha_0, \alpha_4$  for input-0, we have  $\alpha_1 \in Q_2^0$  and  
 $\alpha_7 \in Q_2^0$ .

for input-1, we have  $\alpha_5 \in Q_2^0$  for both  $\alpha_0, \alpha_4$ .  
hence  $\alpha_0, \alpha_4$  are 1-equivalent as  $s(\alpha_0, 0)$ ,  
 $s(\alpha_0, 1)$ ,  $s(\alpha_4, 0)$  &  $s(\alpha_4, 1)$  are 0-equivalent.

With similar arguments we can say

$\alpha_0, \alpha_6$  are 1-equivalent

as  $0 - \alpha_1 \in \mathcal{Q}_2^\circ$  &  $\alpha_6 \in \mathcal{Q}_2^\circ$

$1 - \alpha_4 \in \mathcal{Q}_2^\circ$  &  $\alpha_5 \in \mathcal{Q}_2^\circ$

So  $\{\alpha_0, \alpha_4, \alpha_6\} \in \Pi_1$ , say  $\mathcal{Q}'_2 = \{\alpha_0, \alpha_4, \alpha_6\}$

Now, let us consider  $\alpha_1, \alpha_3, \alpha_5$ , &  $\alpha_7$ .

Amongst these  $\alpha_1, \alpha_3$ , &  $\alpha_5$  are not 1-equivalent  
 $\alpha_1, \alpha_7$  are 1-equivalent.

So,  $\mathcal{Q}'_3 = \{\alpha_1, \alpha_7\} \in \Pi_1$

Now  $\alpha_3, \alpha_5$  are 1-equivalent, as

$0 - \{\alpha_2\} \in \mathcal{Q}_1^\circ$

$1 - \{\alpha_5\} \in \mathcal{Q}_2^\circ$

$$\begin{aligned} \text{So } \Pi_1 &= \{\mathcal{Q}_1^1, \mathcal{Q}_2^1, \mathcal{Q}_3^1, \mathcal{Q}_4^1\} \\ &= \{\{\alpha_2\}, \{\alpha_0, \alpha_4, \alpha_6\}, \{\alpha_1, \alpha_7\}, \{\alpha_3, \alpha_5\}\} \end{aligned}$$

Now try to partition  $\mathcal{Q}_2^1$ ,

$\alpha_0, \alpha_4$  gives us

$0 - \alpha_1, \alpha_7$  and

$1 \rightarrow \alpha_5$

$\alpha_1, \alpha_7 \in \mathcal{Q}_3^1$  hence they are 1-equivalent

Similarly,  $\alpha_5 \in \mathcal{Q}_4^1$

As  $s(\alpha_0, 0), s(\alpha_0, 1), s(\alpha_4, 0) \notin s(\alpha_4, 1)$  are  
1-equivalent,  $\alpha_0, \alpha_4 \in \Pi_2$

Let us take  $\alpha_0, \alpha_6$   
 $\alpha_0 - \alpha_1, \alpha_6$ , which are not 1-equivalent, hence

hence  $\{\alpha_0, \alpha_6\} \notin \Pi_2$ ,  $\{\alpha_0, \alpha_4, \alpha_6\}$  in  $\Pi_1$  breaks to  $\{\alpha_0, \alpha_4\} \notin \{\alpha_6\}$  in  $\Pi_2$ .

$\alpha_1, \alpha_7$  have

$0 - \alpha_6 \in Q_2^1$ ,  $1 - \alpha_2 \in Q_1^1$ , hence  $\{\alpha_1, \alpha_7\} \in \Pi_2$

Similarly,

$\{\alpha_3, \alpha_5\} \in \Pi_2$

$\therefore \Pi_2 = \{\{\alpha_2\}, \{\alpha_0, \alpha_4\}, \{\alpha_6\}, \{\alpha_1, \alpha_7\}, \{\alpha_3, \alpha_5\}\}$

we can compute  $\Pi_3$  further as

$\Pi_3 = \{\{\alpha_2\}, \{\alpha_0, \alpha_4\}, \{\alpha_6\}, \{\alpha_1, \alpha_7\}, \{\alpha_3, \alpha_5\}\}$ .

So the minimum state automaton is  $\Pi_2$   
 where  $\delta' \rightarrow \Pi_2$   
 The transition table will be modified  
 by replacing  $\alpha_i$  with  $[\overset{*}{\alpha_i}, \alpha_i]$

$\alpha_1$  with  $[\alpha_1, \alpha_7]$

$\alpha_3$  with  $[\alpha_3, \alpha_5]$

States/ $\Sigma$

0

1

$[\alpha_0, \alpha_4]$	$[\alpha_1, \alpha_7]$	$[\alpha_3, \alpha_5]$
$[\alpha_1, \alpha_7]$	$[\alpha_6]$	$[\alpha_2]$
$[\alpha_2]$	$[\alpha_0, \alpha_4]$	$[\alpha_2]$
$[\alpha_3, \alpha_5]$	$[\alpha_2]$	$[\alpha_6]$
$[\alpha_6]$	$[\alpha_6]$	$[\alpha_0, \alpha_4]$

## Formal Languages

let S be a variable denoting the sentence  
we can device the following rules to generate  
two types of sentences

$$S \rightarrow \langle \text{noun} \rangle \langle \text{verb} \rangle \langle \text{adverb} \rangle$$

$$S \rightarrow \langle \text{noun} \rangle \langle \text{verb} \rangle$$

$$\langle \text{noun} \rangle \rightarrow \text{Ram}$$

$$\langle \text{noun} \rangle \rightarrow \text{Sam}$$

$$\langle \text{noun} \rangle \rightarrow \text{Gita}$$

$$\langle \text{verb} \rangle \rightarrow \text{ran}$$

$$\langle \text{verb} \rangle \rightarrow \text{ate}$$

$$\langle \text{verb} \rangle \rightarrow \text{walked}$$

$$\langle \text{verb} \rangle \rightarrow \text{slowly}$$

$$\langle \text{adverb} \rangle \rightarrow \text{quickly}$$

$$\langle \text{adverb} \rangle \rightarrow \text{slowly}$$

- Each arrow represents a rule, that is the word on the right of the arrow can replace the word on the left side of the arrow.
- The set of such rules is denoted by P. Each of these rules are called production.

- As our language is restricted to vocabulary, 'Ram', 'Sam', 'Gita', 'ate', 'ran', 'walked', etc.
- The set of vocabulary is called the set of terminals.
- The vocabulary words are called the terminals in formal language.
- Nouns, adverbs, verbs, adjectives, preposition are called 'non-terminals' which can be replaced by the respective terminals.
- The grammar of formal languages consists of L-tuples.

$V_H$  → Set of non-terminals  
 $\Sigma$  → Set of terminals  
 $P$  → set of production  
 $S$  → Starting non-terminal

P is a finite set of rules of the form  
 $\alpha \rightarrow \beta$ , where  
 $\alpha, \beta \in (V \cup \Sigma)^*$

## Regular Languages

- A regular language is that language which is recognized by some finite automaton.
- A regular language follows regular grammar.
- A regular grammar is a type-3 grammar.
- Type-3 grammar has productions of the (called type-3 production)  
 $A \rightarrow a$   
or  $A \rightarrow aB$ , where  $A, B \in V_N$   
and  $a \in \Sigma$

- A grammar is called a type-3 or regular grammar, if all its productions are type-3.

Q) Let us consider the following set of strings  
 $(\lambda, 0, 01, 10, 110, 1010, \dots)$   
 design a regular grammar.

Solution :  $\Sigma = \{0, 1\}, V_N = \{S, A, B\}$

$$\begin{array}{l}
 S \rightarrow \lambda \mid 0 \mid 1A \mid 1OB \\
 A \rightarrow 1A \mid OB \\
 B \rightarrow 1A \mid \lambda \\
 C \rightarrow 1
 \end{array} \quad \left| \quad \begin{array}{l}
 S \rightarrow IA \rightarrow 1\underline{OB} \rightarrow 10 \\
 S \rightarrow IA \rightarrow 11A \rightarrow 11\underline{OB} \rightarrow 110 \\
 S \rightarrow IA \mid 1\underline{OB} \mid 10\underline{IA} \mid 101\underline{OB} \mid 1010
 \end{array} \right.$$

Q) If  $G = (\{S\}, \{a\}, \{S \rightarrow SS\}, S)$

find the language generated by G

Solution :  $L(G) = \emptyset$ , As the only production  $S \rightarrow SS$  does not have any terminal on the right hand.