MATHEMATICAL FOUNDATION FOR COMPUTER SCIENCE

Prepared by: Er. Ankit Kharel

Nepal college of information technology

FINITE STATE AUTOMATA

- *Sequential Circuits and Finite state Machine*
- *Finite State Automata*
- *Non-deterministic Finite State Automata*
- *Language and Grammars*
- *Language and Automata*
- *Regular Expression*

COMBINATIONAL CIRCUITS:

- Combination circuit is a circuit where the output depends only on the present value of input. Combinatorial circuits can be constructed using solid-state devices, called gates.

- A combinatorial circuit has no memory; previous inputs and the state of the system do not affect the output of a combinatorial circuit.

Figure 11.1.4 A combinatorial circuit.

The logic table for this combinatorial circuit follows.

x_1	x_2	x_3	y
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

SEQUENTIAL CIRCUITS:

- Sequential circuit is a circuit where the output depends on the present value of input as well as the sequence of past input.

- A Sequential circuit is a combination of combinational circuit and a storage element.

- The sequential circuits use current input variables and previous input variables which are stored and provides the data to the circuit on the next clock cycle.

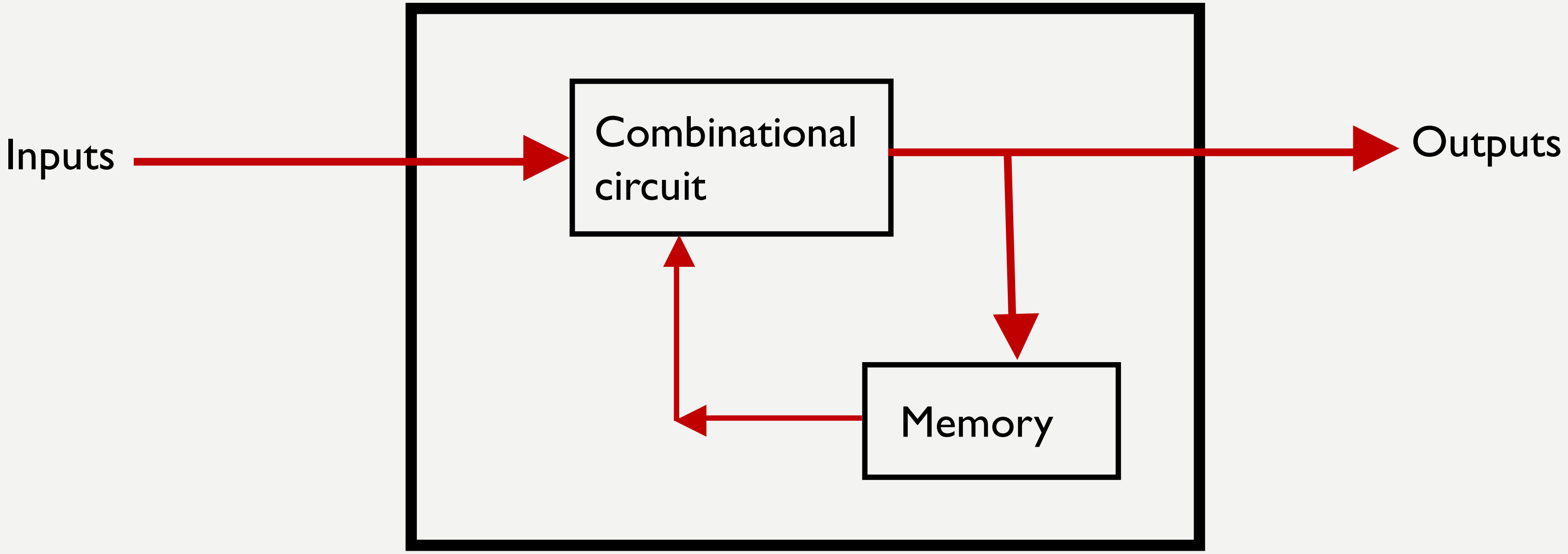


Fig. Sequential circuit

SEQUENTIAL CIRCUITS:

- We will assume that the state of the system changes only at time $t = 0, 1, \ldots$. A simple way to introduce sequencing in circuits is to introduce a **unit time delay**.

- A unit time delay accepts as input a bit x_t at time t and outputs x_{t-1}, the bit received as input at time $t - 1$.

- The sequential circuits use current input variables and previous input variables which are stored and provides the data to the circuit on the next clock cycle.

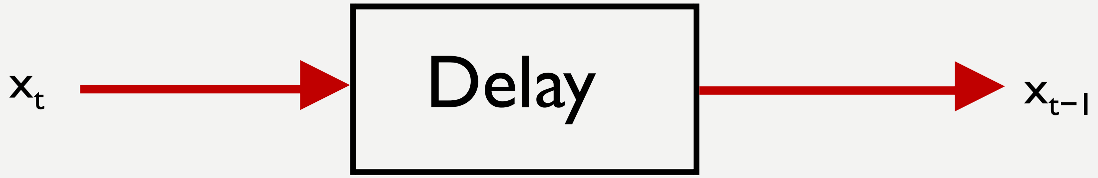


Fig. Unit time delay

- As an example of the use of the unit time delay, we discuss the **serial adder.**

SERIAL ADDER:

- A serial adder accepts as input two binary numbers

$$x = 0x_N x_{N-1} \cdots x_0 \text{ and } y = 0y_N y_{N-1} \cdots y_0 \text{ and output s the sum as:}$$

$$Z = z_{N+1} z_N \cdots z_0$$

Example: $x = 1011 \quad (x_3x_2x_1x_0)$

$+ \ y = 0101 \quad (y_3y_2y_1y_0)$

$z = 10000 \ (z_4z_3z_2z_1z_0)$

- The numbers x and y are input sequentially in pairs, x0, y0;...; x_N, y_N; . The sum is output $z_0, z_1, ..., z_{N+1}$.

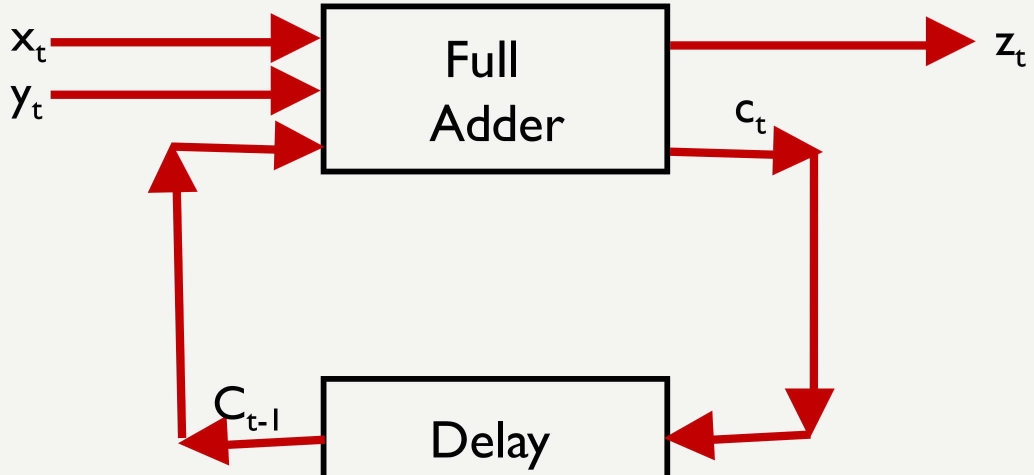


Fig. Serial adder circuit

SERIAL ADDER:

- Addition of x = 010 , y = 011.

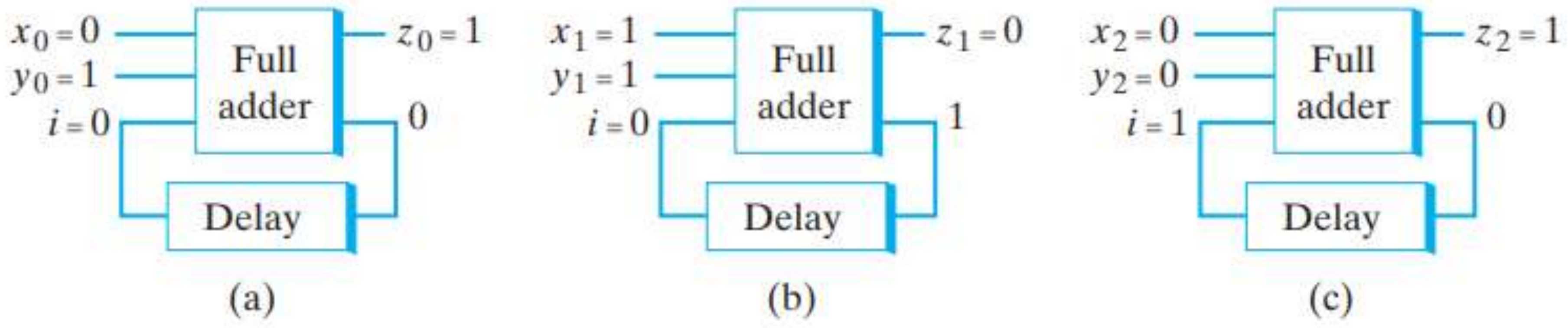


Figure 12.1.3 Computing 010 + 011 with the serial-adder circuit.

- First, we give input as x_0 = 0 and y_0 = 1 to the full adder. The adder computes z_0= (0+1)1 and sends 0 as carry to the delay.
- Second, we give input as x_1 = 1 and y_1 = 1 to the full adder. Now, the delay sends the input 0 and the adder sums z_1 = (1+1+0) 0 and carry 1 is send to the delay.
- Third, we give input as x_2 = 0 and y_2 = 0 to the full adder. Now, the delay sends the input 1 and the adder sums z_1 = (0+0+1) 1 and carry 0 is send to the delay.
- Finally , the output is : z=x+y=101

FINITE STATE MACHINE:

- A finite state machine is a model of computation based on a hypothetical machine made of one or more states. It is used to simulate sequential logic and some computer program.

- We have a fixed set of states that machine can be in.

- The machine can only be in one state at a time.

- Every state has a set of transition and every transition is associated with an input and pointing to a state

Example:

TRAFFIC LIGHT:

A simple traffic light system can be modeled with a finite state machine. Let's look at each core component and identify what it would be for traffic light.

a) **States:** A traffic light generally has three sates: RED, GREEN and YELLOW.

b) **Initial state:** GREEN(suppose)

c) **Accepting state:** In real world traffic lights run indefinitely, so there would be no accepting sate for this.

FINITE STATE MACHINE:

Example:

TRAFFIC LIGHT:

A simple traffic light system can be modeled with a finite state machine. Let's look at each core component and identify what it would be for traffic light.

a) **States:** A traffic light generally has three sates: RED, GREEN and YELLOW.

b) **Initial state:** GREEN(suppose)

c) **Accepting state:** In real world traffic lights run indefinitely, so there would be no accepting sate for this.

d) **Alphabets:** Positive integer representing seconds

e) **Transition:**

- If we are in state **GREEN**, wait for 360s and then go to state **YELLOW**
- If we are in state **YELLOW**, wait for 10s and then go to state **RED**
- If we are in state **RED**, wait for 120s and then go to state **GREEN**

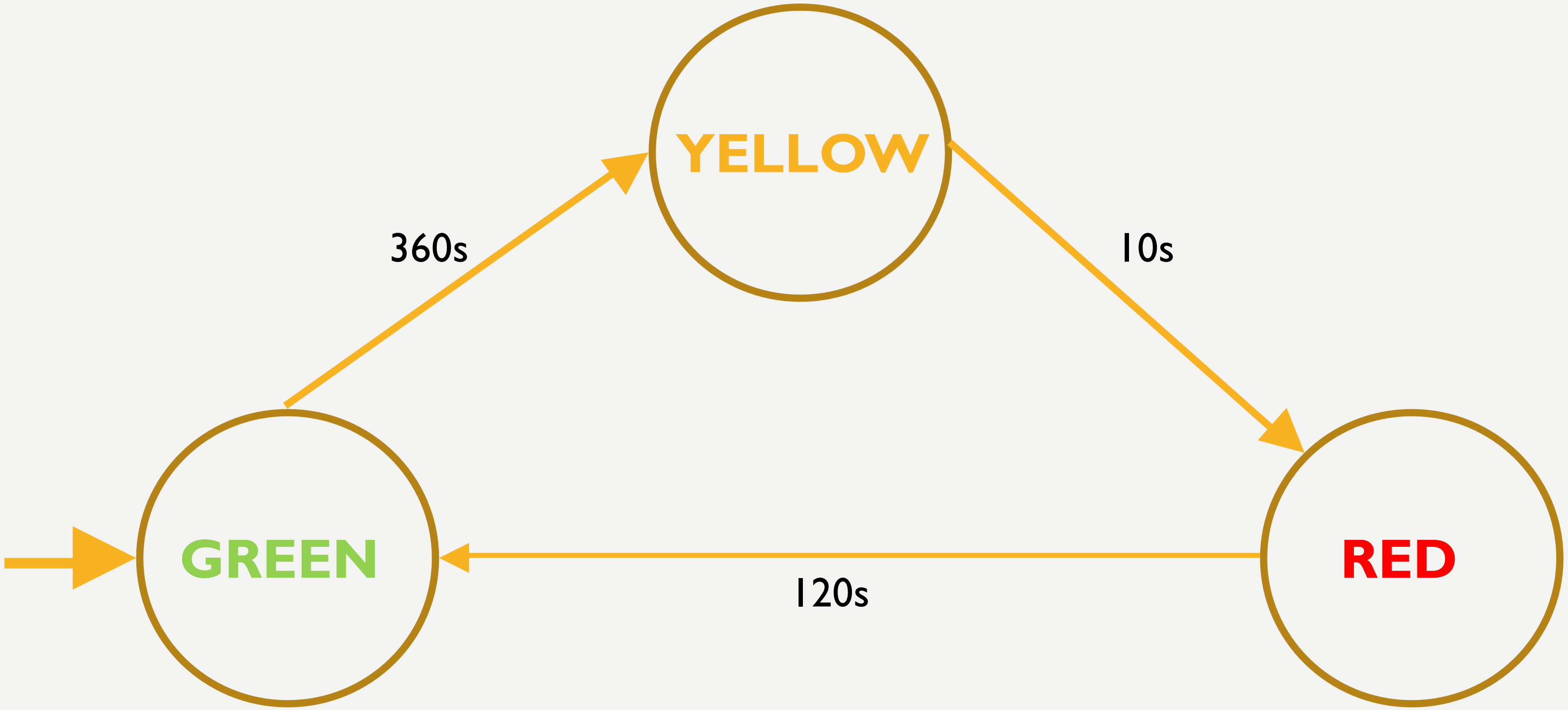


Fig: Transition diagram

FINITE STATE MACHINE:

Formal Definition of Finite State Machine(FSM)

A *finite-state machine* is an abstract model of a machine with a primitive internal memory. A finite-state machine M consists of

(a) A finite set **I** of input symbols.
(b) A finite set **O** of output symbols.
(c) A finite set **S** of states.
(d) A next-state function **f** from S × I into S.
(e) An output function **g** from S × I into O.
(f) An initial state **σ** ∈ S.

We write $M = (I, O, S, f, g, \sigma)$

FINITE STATE MACHINE:

Let, $I = \{a, b\}$, $O = \{0, 1\}$, and $S = \{\sigma_0, \sigma_1\}$. Define the pair of functions $f : S \times I \rightarrow S$ and $g : S \times I \rightarrow O$ by the rules given in Table.

	f		g	
$S \setminus \mathcal{I}$	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

Solution:

(a) State Transition Function(STF):

$f(\sigma 0, a) = \sigma 0$

$f(\sigma 0, b) = \sigma 1$

$f(\sigma 1, a) = \sigma 1$

$f(\sigma 1, b) = \sigma 1$

(b) Machine Function(MF)

$g(\sigma 0, a) = 0$

$g(\sigma 0, b) = 1$

$g(\sigma 1, a) = 1$

$g(\sigma 1, b) = 0$.

The next state Function and Output function can also be defined by the Transition Diagram.

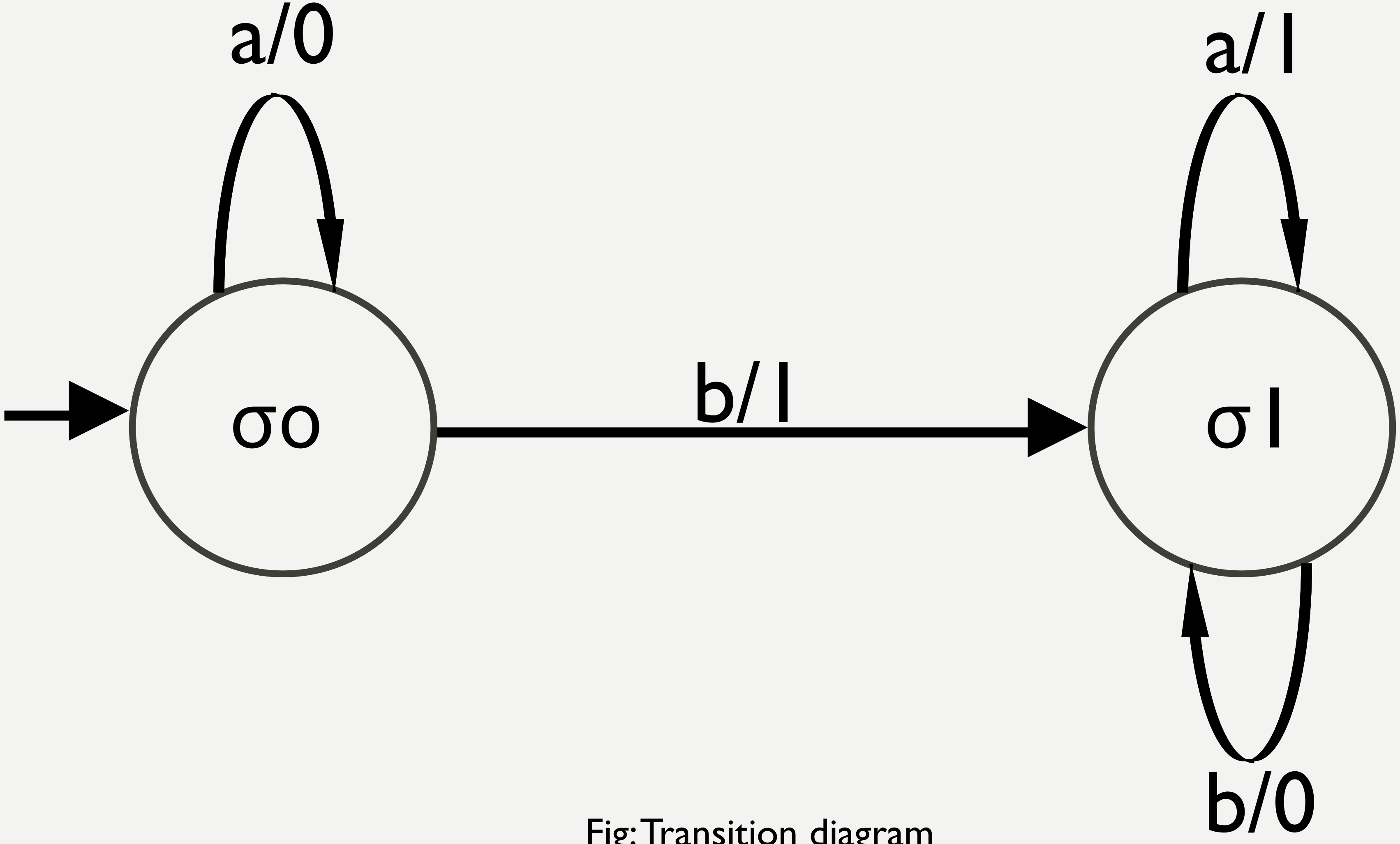


Fig: Transition diagram

Definition: Let M = (I, O, S, f, g, σ) be a finite-state machine. The transition diagram of M is a digraph G whose vertices are the members of S. An arrow designates the initial state σ. A directed edge (σ1, σ2) exists in G if there exists an input i with f(σ1, i) = σ2. In this case, if g(σ1, i) = o, the edge (σ1, σ2) is labeled i/o

FINITE STATE MACHINE:

Find the output string corresponding to the input string “aababba” for below finite-state machine.

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

<u>Solution:</u>

Initial State	Input	Output State	Output
σ_0	a	σ_0	0
σ_0	a	σ_0	0
σ_0	b	σ_1	1
σ_1	a	σ_1	1
σ_1	b	σ_1	0
σ_1	b	σ_1	0
σ_1	a	σ_1	1

The Output is:

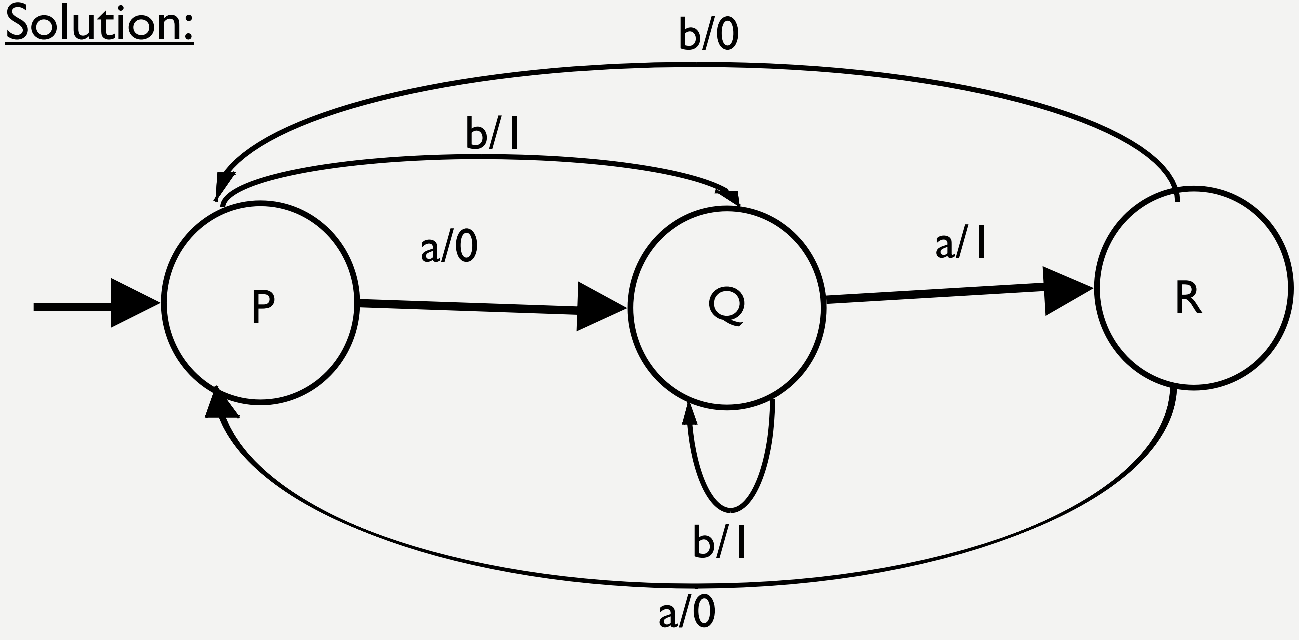
0011001

FINITE STATE MACHINE:

Draw the transition diagram of the finite state machine M, where, I = {a, b} O = {0, 1} S = {P, Q, R} σ = P and transition given by below table. Find the output string corresponding to the input string "aabbaba"

	f		g	
S/I	a	b	a	b
P	Q	Q	0	1
Q	R	Q	1	1
R	P	P	0	0

Solution:



Initial State	Input	Output State	Output
P	a	Q	0
Q	a	R	1
R	b	P	0
P	b	Q	1
Q	a	R	1
R	b	P	0
P	a	Q	0

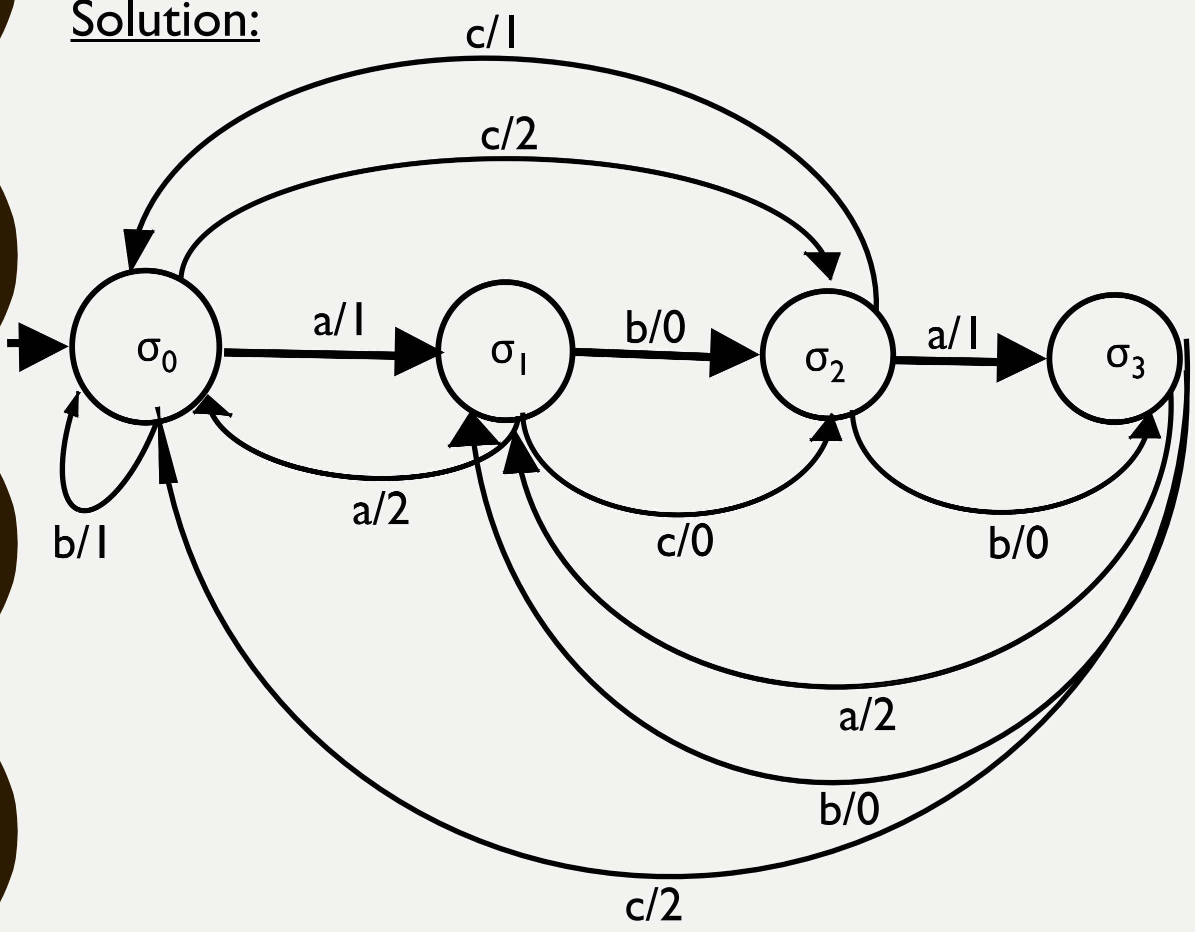
Output: 0101100

FINITE STATE MACHINE:

Draw the transition diagram of the finite state machine M, where, I = {a, b, c} O = {0, 1, 2} S = {$\sigma_0, \sigma_1, \sigma_2, \sigma_3$}, $\sigma = \sigma_0$ and transition given by below table. Find the output string corresponding to the input string "aabaab"

$\mathcal{S}$ \ $\mathcal{I}$	f: a	f: b	f: c	g: a	g: b	g: c
σ_0	σ_1	σ_0	σ_2	1	1	2
σ_1	σ_0	σ_2	σ_2	2	0	0
σ_2	σ_3	σ_3	σ_0	1	0	1
σ_3	σ_1	σ_1	σ_0	2	0	2

<u>Solution:</u>



1. $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1\}$

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_1	1	1
σ_1	σ_0	σ_1	0	1

2. $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1\}$

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_0	0	0
σ_1	σ_0	σ_0	1	1

In Exercises 6–10, find the sets $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{S}$, the initial state, and the table defining the next-state and output functions for each finite-state machine.

6.