CHAPTER 5

FINITE AUTOMATA

Deterministic Finite Automata (DFA)

- In computer science, we study different types of computer languages, such as Basic, Pascal, and C++.

- We will discuss a type of a language that can be recognized by special types of machines.

Deterministic Finite Automata (DFA)

- A deterministic finite automaton (pl. automata) is a mathematical model of a machine that accepts languages of some alphabet.

Deterministic Finite Automata (DFA)

- Deterministic Finite Automaton is a quintuple $M = \{ S, I, q_0, f_s, F\}$
  where,
  S is a finite nonempty set of states
  I is the input alphabet (a finite nonempty set of symbols)
  q_0 is the initial state
  f_s is the state transition function
  F is the set of final states, subset of S.

example

- Let $M = \{ \{q_0, q_1, q_2\}, \{0,1\}, q_0, f_s, \{q_2\}\}$ where f_s is defined as follows:

  $f_s(q_0,0) = q_1$, $f_s(q_1,1) = q_2$

  $f_s(q_0,1) = q_0$, $f_s(q_2,0) = q_0$

  $f_s(q_1,0) = q_2$, $f_s(q_2,1) = q_1$

- Note that for M:

  $S=\{q_0,q_1,q_2\}$, $I= \{0,1\}$, $F=\{q_2\}$

  q_0 is the initial state

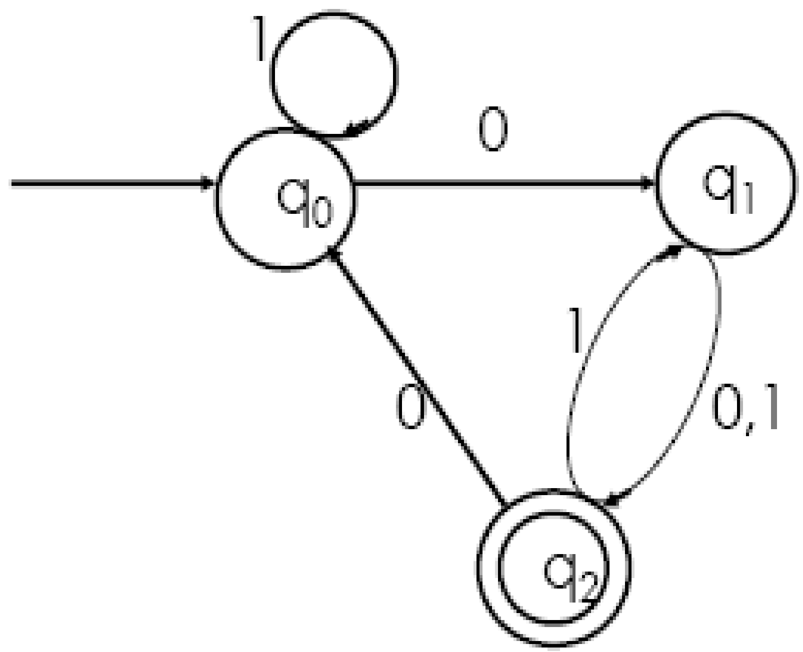
example

- The state transition function of a DFA is often described by means of a table, called a **transition table**.

f_s	0	1
q_0	q_1	q_0
q_1	q_2	q_2
q_2	q_0	q_1

example

- The transition diagram of this DFA is,



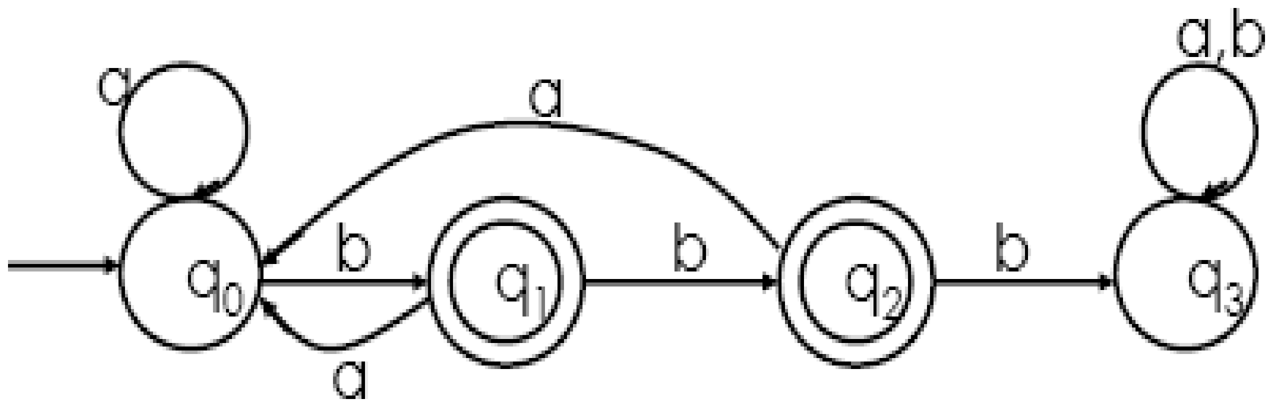
example

Let $M=(\{q_0,q_1,q_2,q_3\}, \{a,b\}, q_0, f_s , \{q_1,q_2\})$
where f_s is given by the table

f_s	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

example

- The transition diagram of this DFA is,



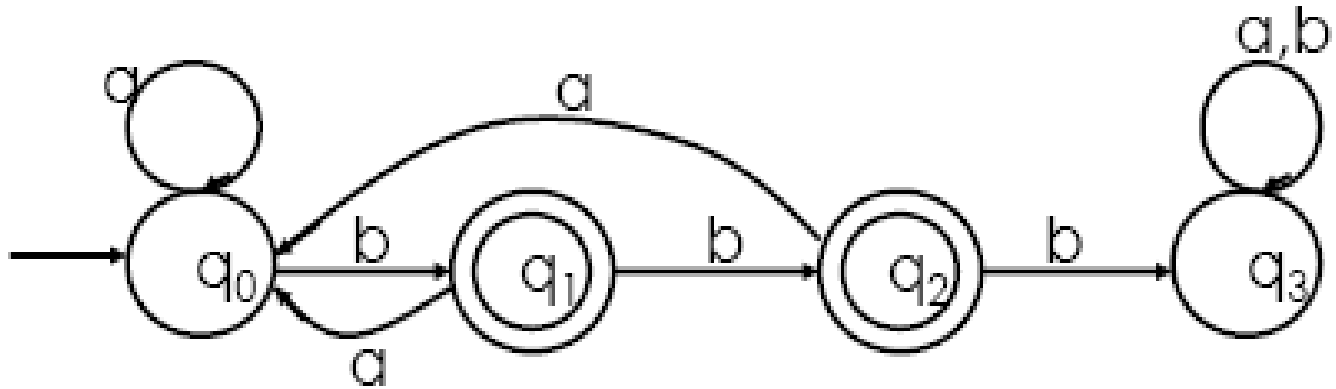
Deterministic Finite Automata (DFA)

- Let $M = \{ S, I, q_0, f_s, F\}$ be a DFA and w is an input string,
- w is said to be accepted by M if

$$f_s^*(q_0, w) \in F$$

- f_s^* - extended transition function for M

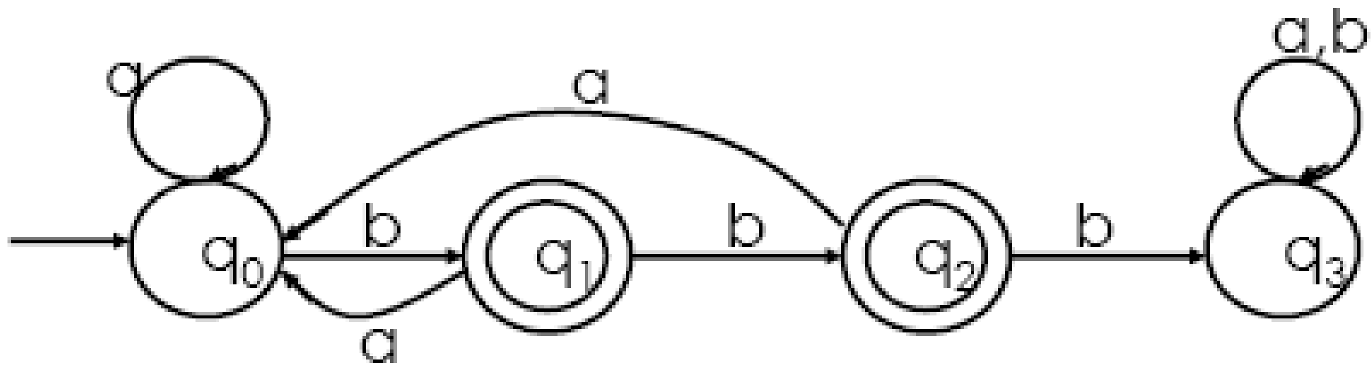
example



w= abb

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_2$ accepted by M

example

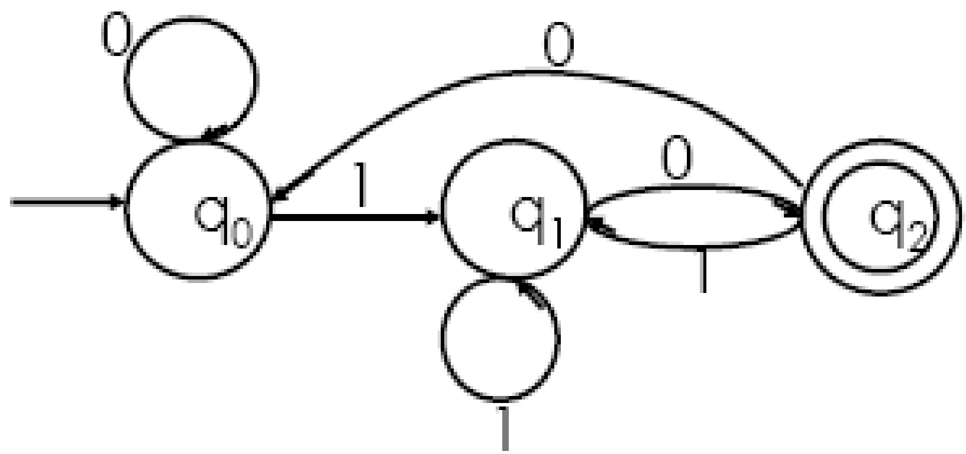


w= abba

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_0$$

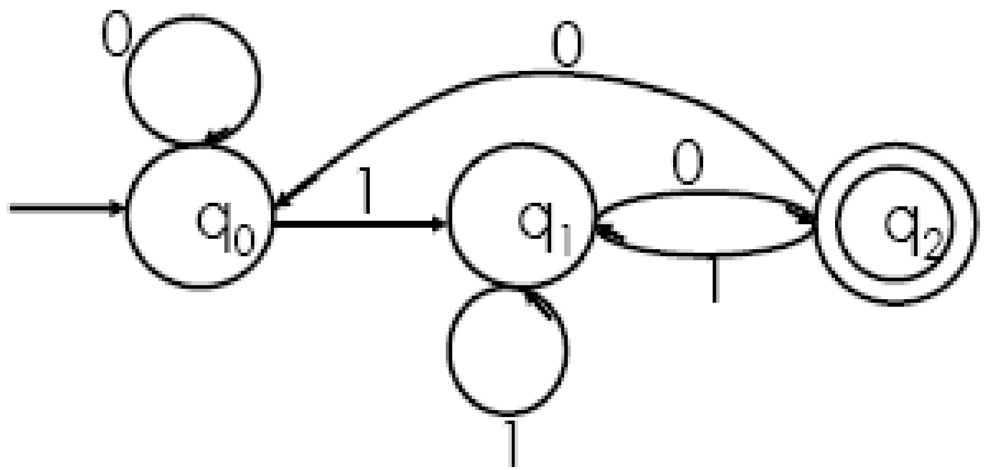
not accepted by M

example



- What are the states of M? q_0, q_1, q_2
- Write the set of input symbols. $I = \{0,1\}$
- Which is the initial state? q_0

example



- Write the set of final states. $F = \{q_2\}$
- Write the transition table for this DFA

example

The transition table, fs

f_s	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

example

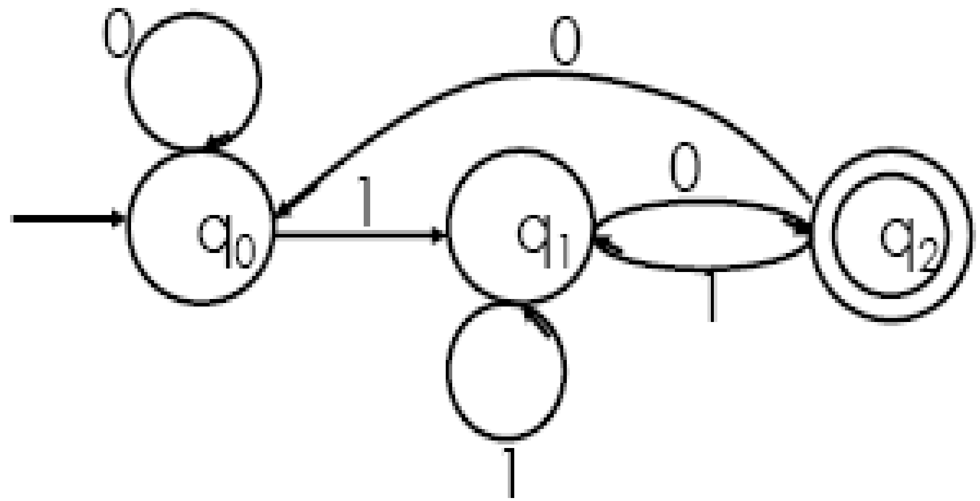
Which of the strings are accepted by M?

0111010, 00111, 111010,

0100, 1110

example

0111010

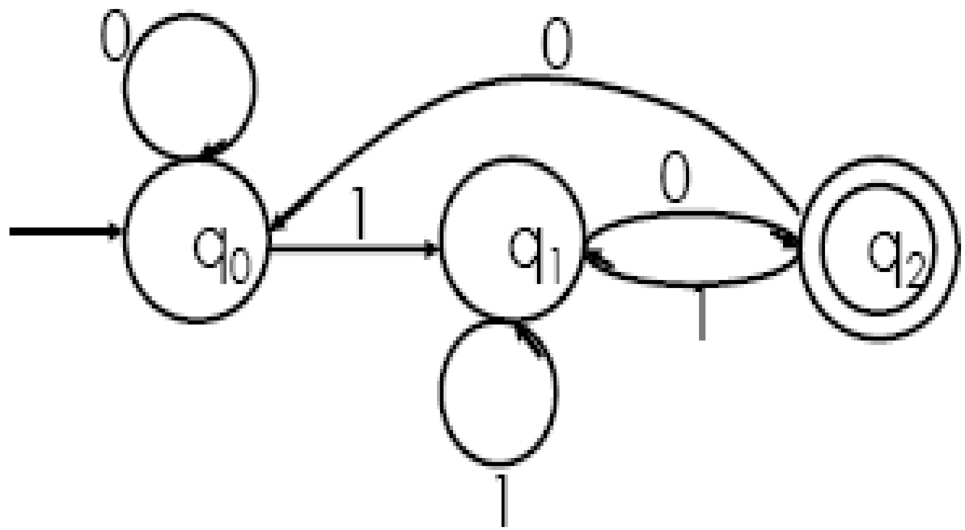


$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \xrightarrow{0} q_2$$

accepted by M

example

00111

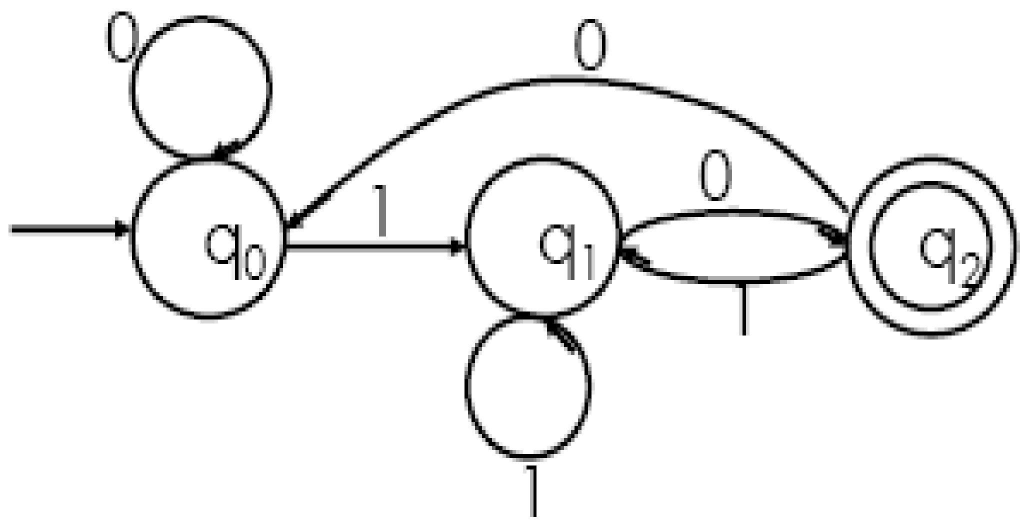


$$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{1} q_1$$

not accepted by M

example

111010

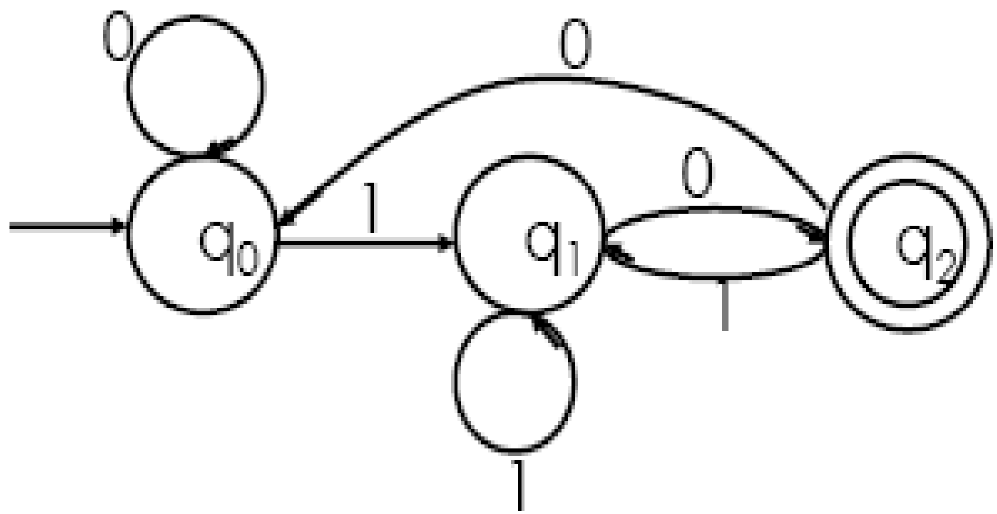


$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \xrightarrow{0} q_2$$

accepted by M

example

0100

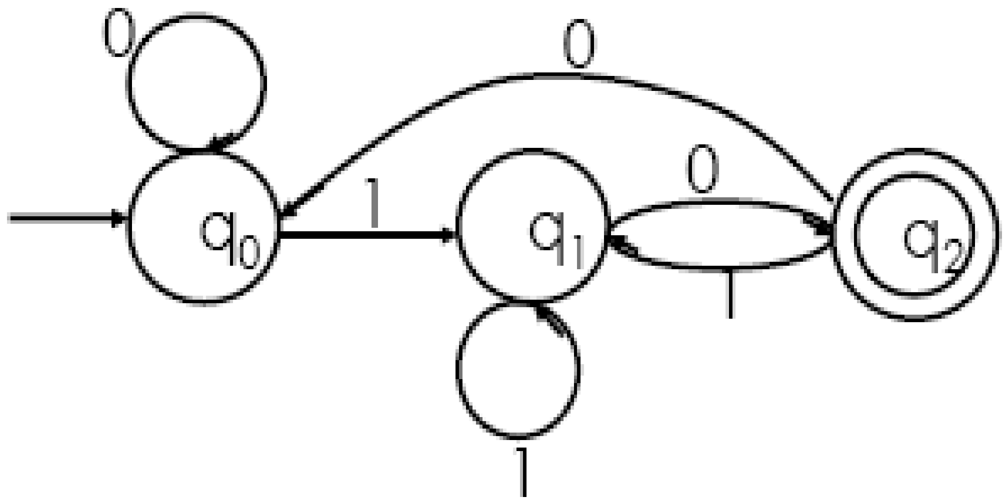


$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_0$

not accepted by M

example

1110



$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2$

accepted by M

example

Construct a state transition diagram of a DFA that accepts on {a,b} that contain an even number of a's and an odd number of b's.

Example of accepted strings:

aab, baa, baaabba

example

4 states,

q_0	even num. of a's & even num. of b's.
q_1	even num. of a's & odd num. of b's.
q_2	odd num. of a's & odd num. of b's.
q_3	odd num. of a's & even num. of b's.

$S = \{q_0, q_1, q_2, q_3\}$

example

set of states, $S = \{q_0, q_1, q_2, q_3\}$

set of input symbols, $I = \{a, b\}$

initial state, q_0

final state, q_1

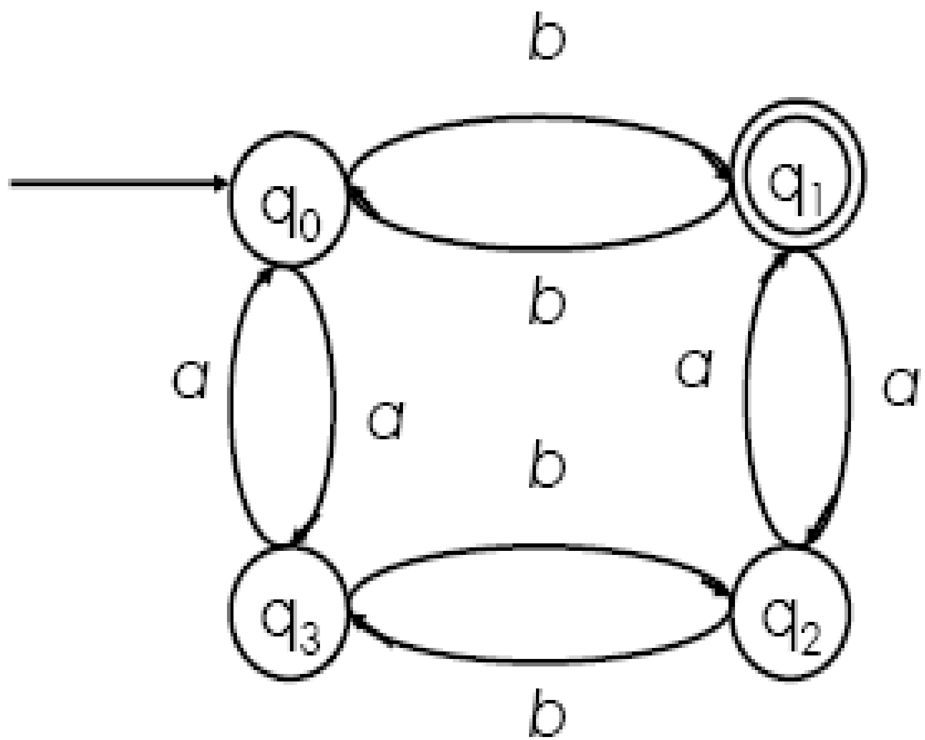
example

State transition function

f_s	a	b
q_0	q_3	q_1
q_1	q_2	q_0
q_2	q_1	q_3
q_3	q_0	q_2

example

State transition diagram



exercise

Let $M=(S, I, q_0, f_s, F)$ be the DFA such that $S=\{q_0,q_1,q_2\}$, $I=\{a,b\}$, $F=\{q_2\}$, q_0 =initial state, and f_s is given by,

f_s	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_0

exercise

Draw the state diagram of M.

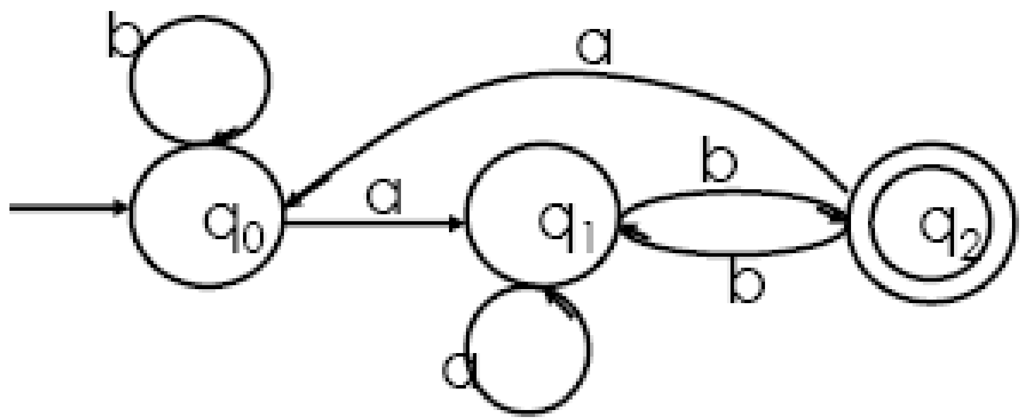
Which of the strings

abaa, bbbabb, bbbaa dan bababa

are accepted by M?

exercise

The transition diagram of M is,

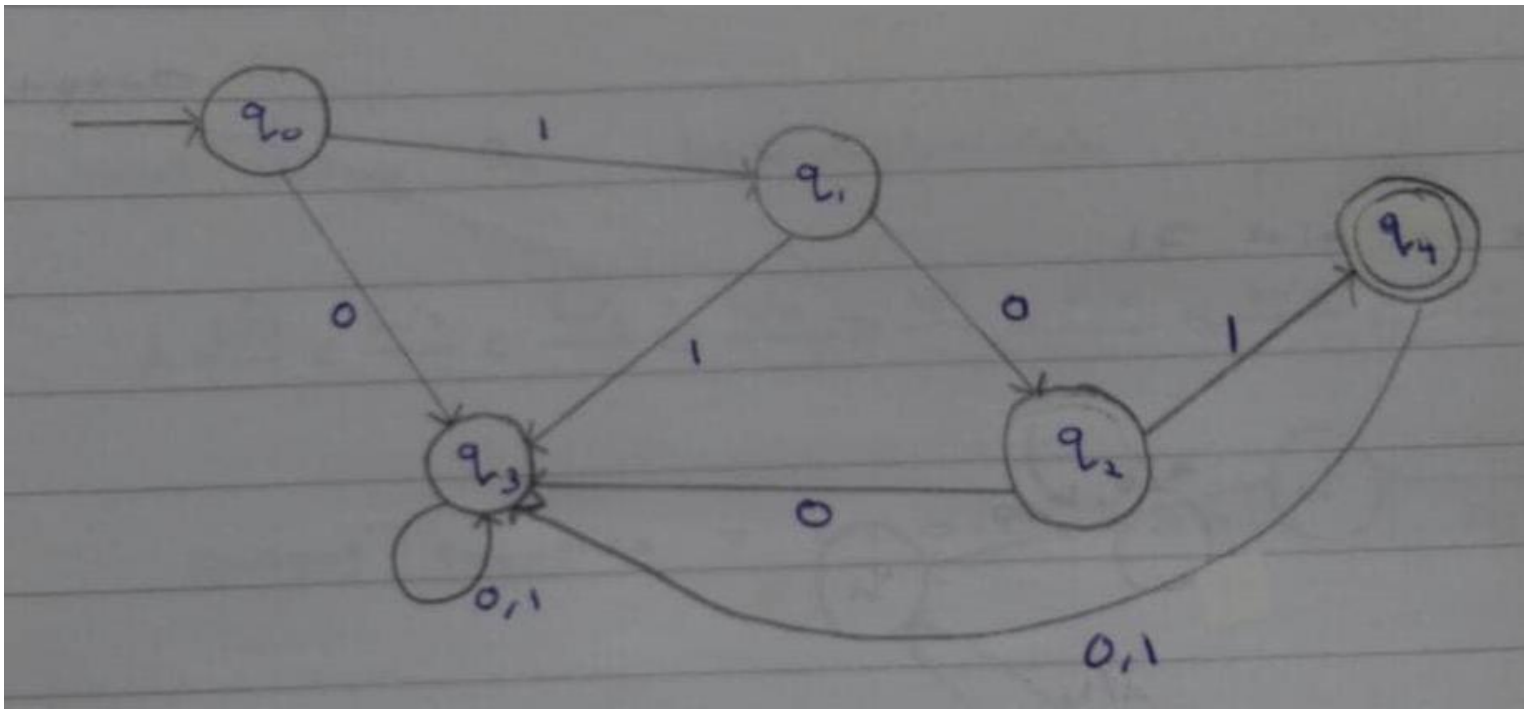


Construct the transition table of M.
Which of the strings
baba, baab, abab dan abaab
are accepted by M?

exercise

- Construct a state transition diagram of a DFA M with the input set {0,1} such that M accepts only the string 101.

ANSWER



Finite State Machines (FSM)

- Automata with input as well as output.
- Every state has an input and corresponding to the input the state also has an output.
- These types of automata are commonly called **finite state machines**.

Finite State Machines (FSM)

- A finite state machine is a sextuple,
  $M = \{ S, I, O, q_0, f_s, f_o \}$
  where,
  S is a finite nonempty set of states
  I is the input alphabet
  O is the output alphabet
  q_0 is the initial state
  f_s is the state transition function
  f_0 is the output function.

example

- Let $M = \{ S, I, O, q_0, f_s, f_o \}$ be the FSM
- where,

  $S = \{q_0, q_1, q_2\}$,

  $I = \{a, b\}$,

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

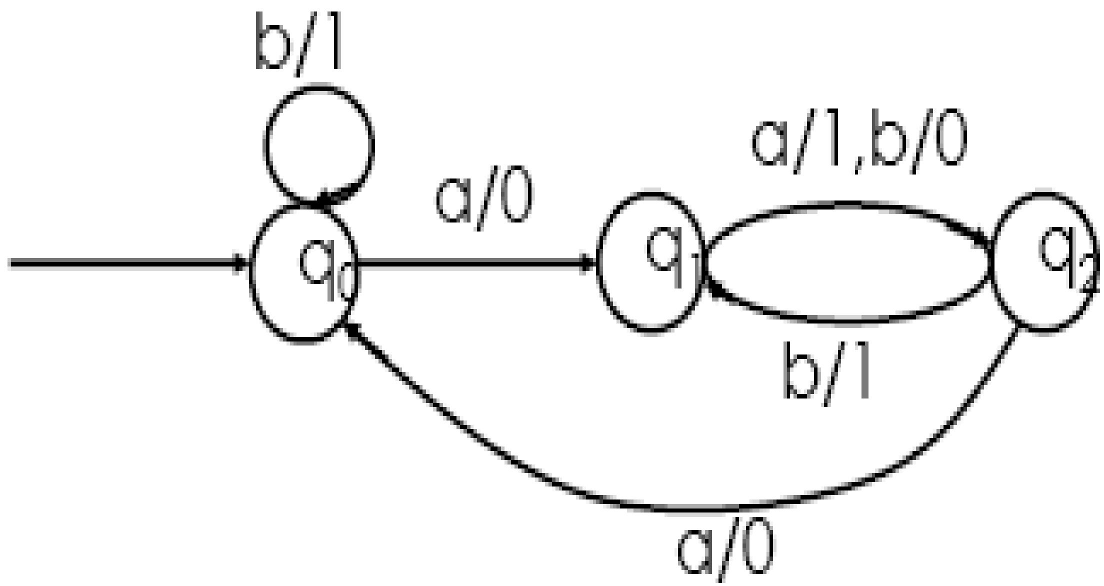
  q_0 = initial state,

example

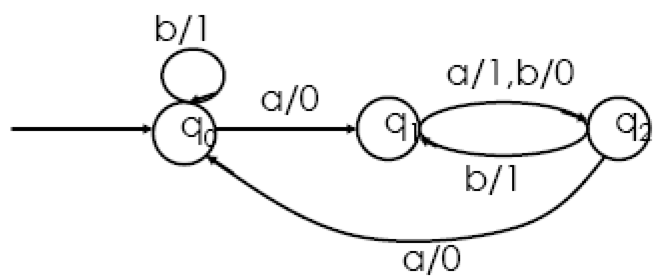
f_s and f_0

	f_s a	b	f_o a	b
q_0	q_1	q_0	0	1
q_1	q_2	q_2	1	0
q_2	q_0	q_1	0	1

example



example



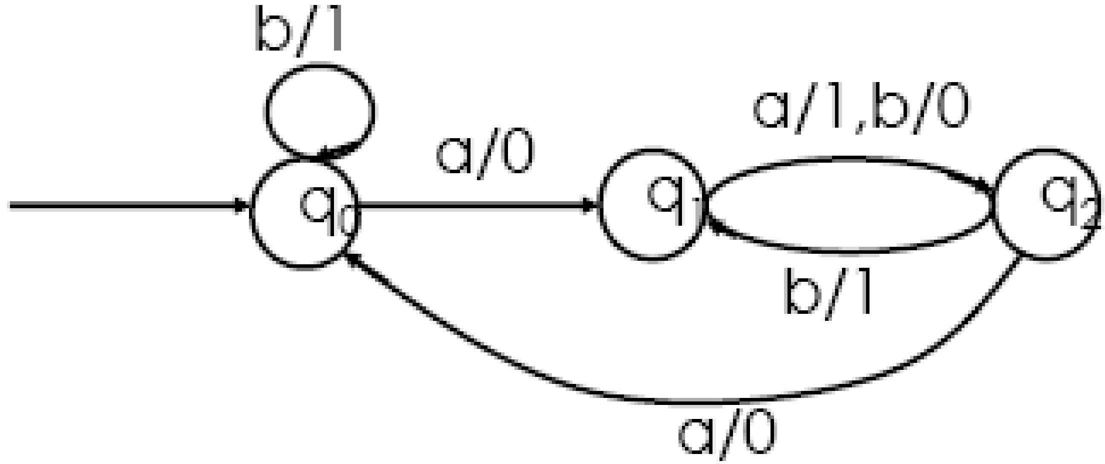
Input string: bbab

$$q_0 \xrightarrow{b} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$$

1 1 0 0

Output string: 1100 Output: 0

example



Input string: bababaa

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2$$

1 0 0 0 1 0 1

Output string: 1000101 Output: 1

example

- Let M= { S, I, O, q_0, f_s, f_o} be the FSM

- where,

  S ={q_0,q_1,q_2 ,q_3},

  I ={a,b},

  O ={0,1},

  q_0 = initial state,

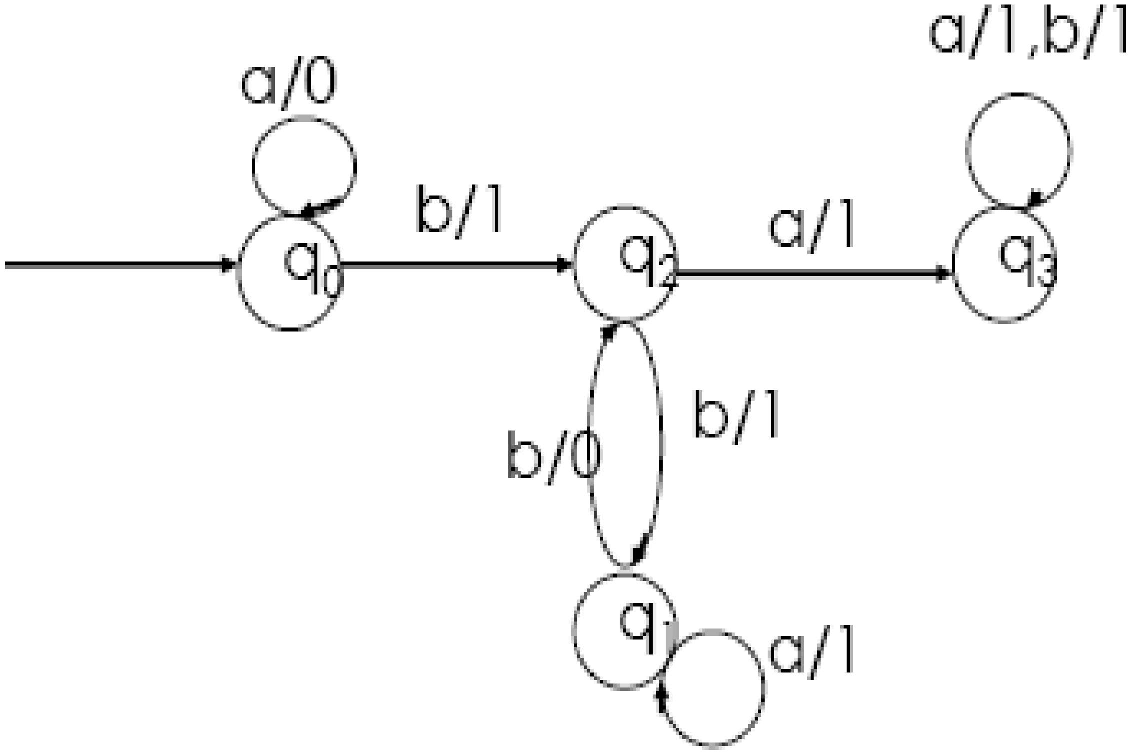
example

- f_s and f_0

	f_s a	b	f_o a	b
q_0	q_0	q_2	0	1
q_1	q_1	q_2	1	0
q_2	q_3	q_1	1	1
q_3	q_3	q_3	1	1

example

- Draw the transition diagram of M.



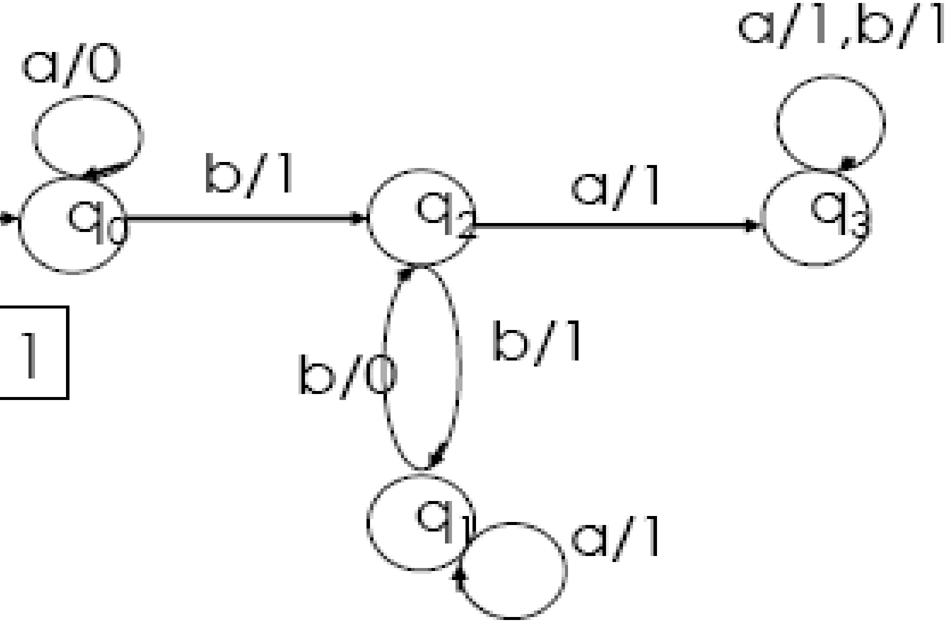
example

- What is the output string if the input string is *abbabab*?

abbabab

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{b} q_3$$

0 1 1 1 0 1 1



Output string: 0111011

example

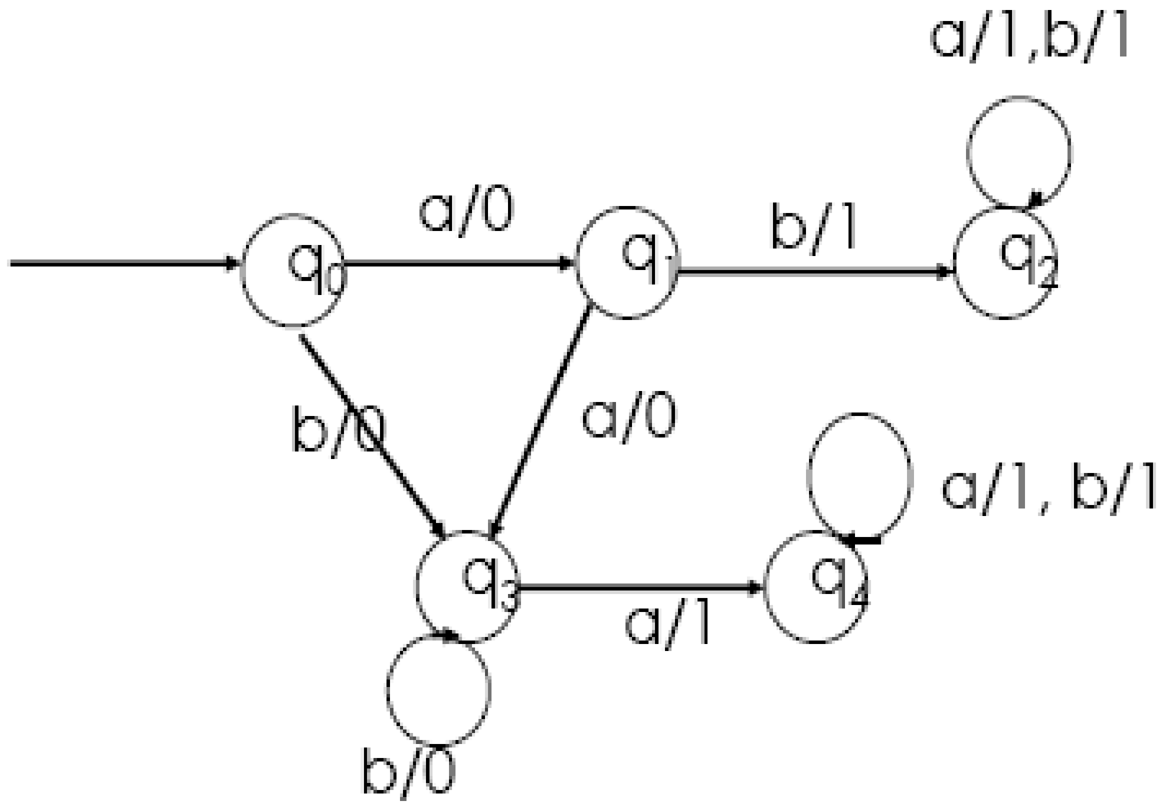
- What is the output of *abbabab*?

Output: 1

Finite State Machines (FSM)

- Let M be a FSM.
- Let x be a nonempty string in M.
- We say that x is accepted by M if and only if the output of x is 1.

example



example

- Write the transition table of M.
- What is the output string if the input string is *aaabbbb*?
- What is the output if the input string is *bbbaaaa*?

example

- Is the string *aaa* accepted by M?

- Which of the strings
  ba, aabbba, bbbb, aaabbbb
  are accepted by M?

example

- The transition table of *M*.

	f_s a	b	f_o a	b
q_0	q_1	q_3	0	0
q_1	q_3	q_2	0	1
q_2	q_2	q_2	1	1
q_3	q_4	q_3	1	0
q_4	q_4	q_4	1	1

example

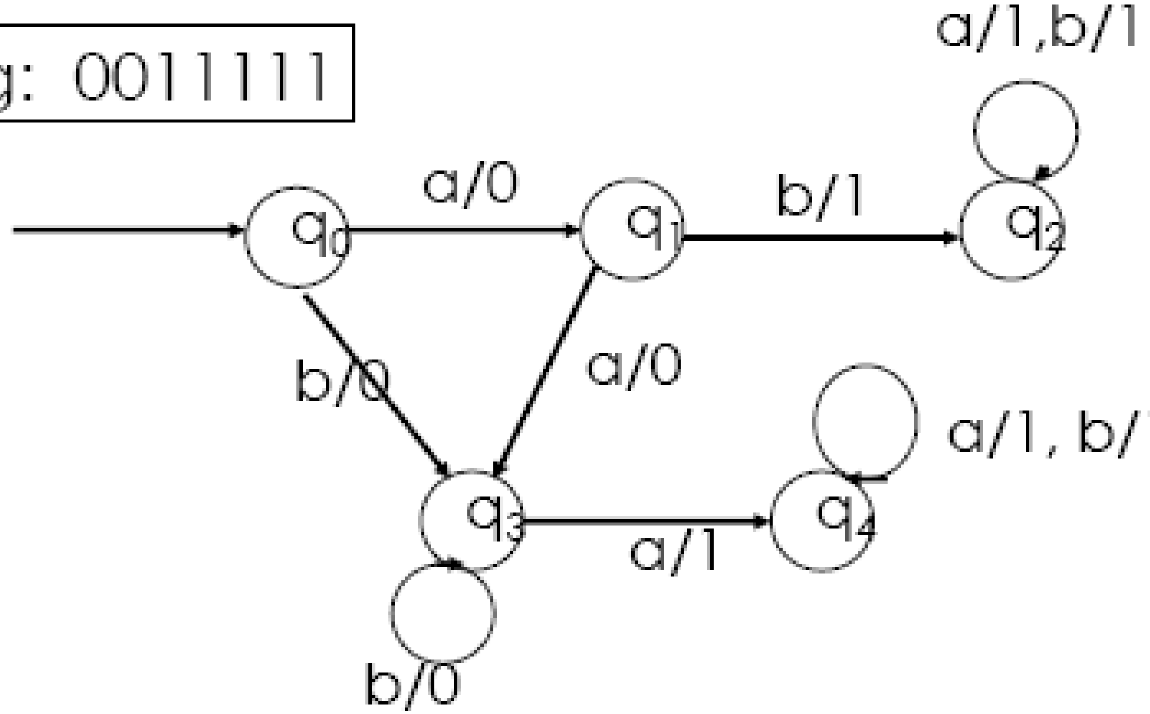
- What is the output string if the input string is aaabbbb?

aaabbbb

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_3 \xrightarrow{a} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4$

0	0	1	1	1	1	1

Output string: 0011111



example

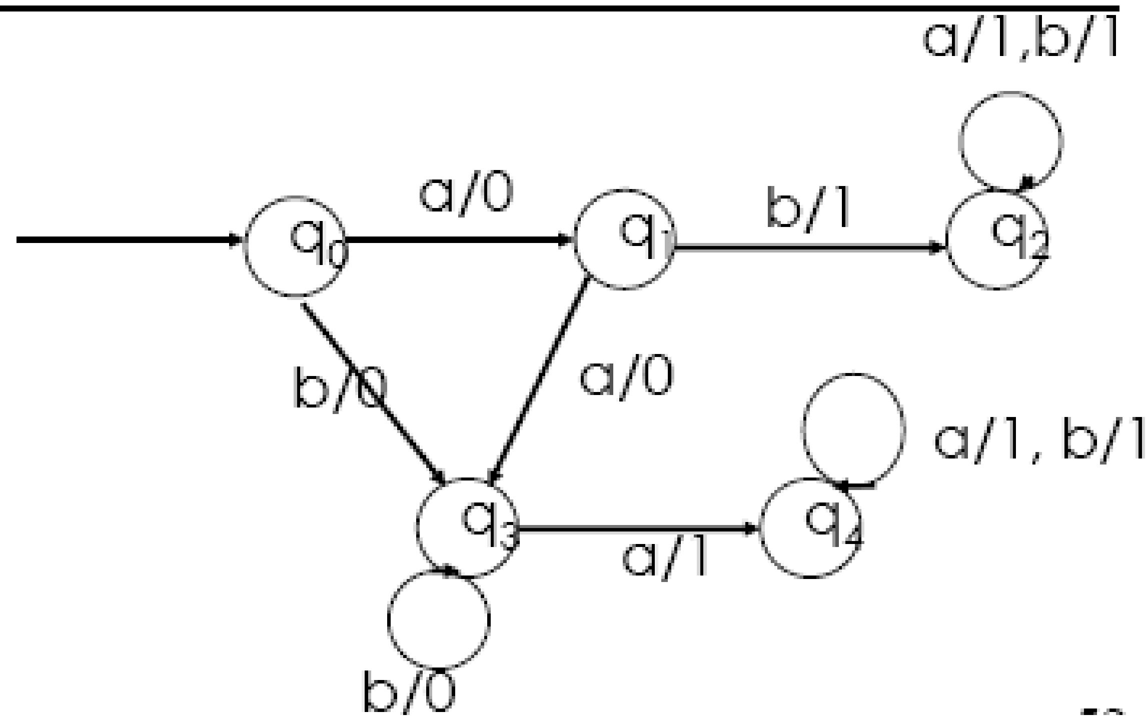
- What is the output if the input string is *bbbaaaa*?

bbbaaaa

$q_0 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{a} q_4 \xrightarrow{a} q_4 \xrightarrow{a} q_4 \xrightarrow{a} q_4$

0 0 0 1 1 1 1

Output: 1



example

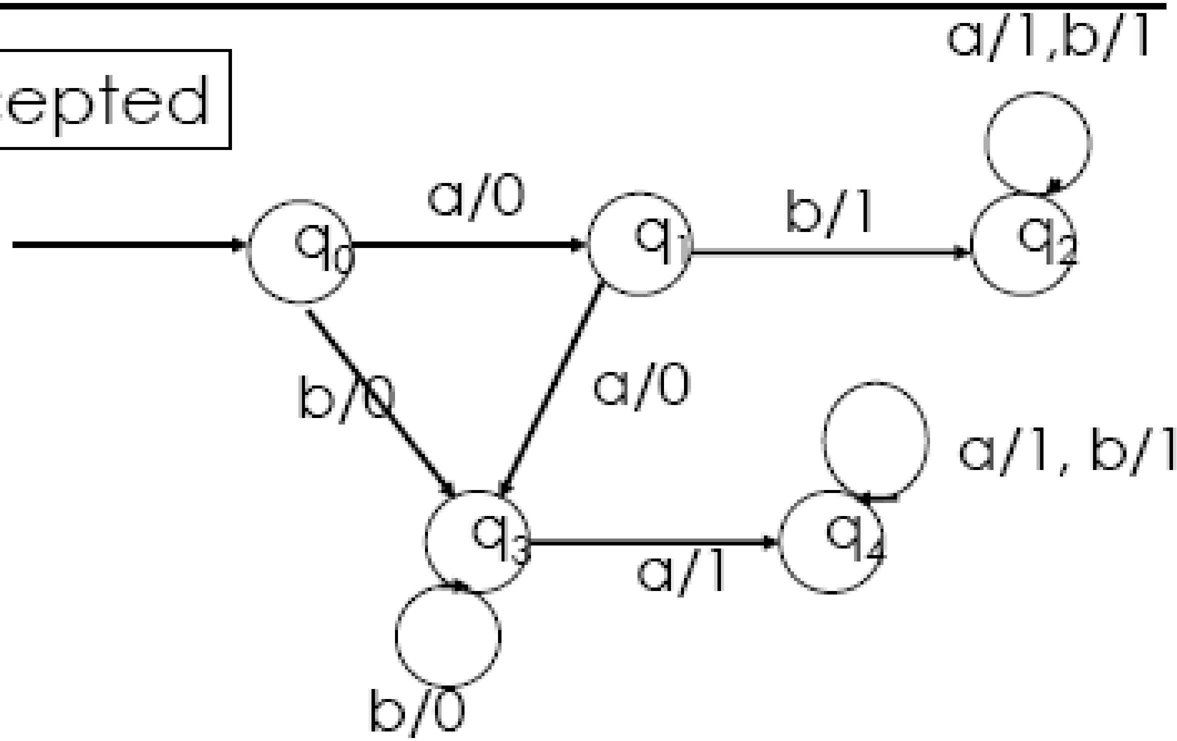
- Is the string *aaa* accepted by M?

aaa

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_3 \xrightarrow{a} q_4$

0 0 1

Output: 1,accepted



example

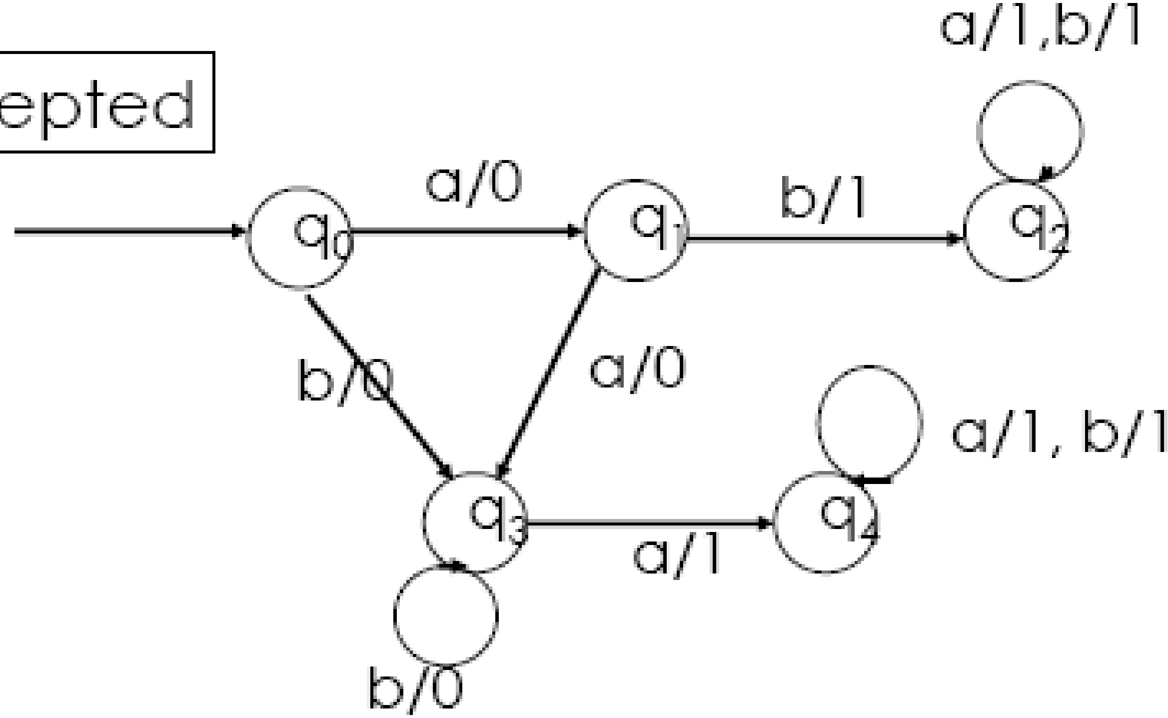
- Which of the strings
  ba, aabbba, bbbb, aaabbbb
  are accepted by M?

ba

$q_0 \xrightarrow{b} q_3 \xrightarrow{a} q_4$

0 1

Output: 1,accepted

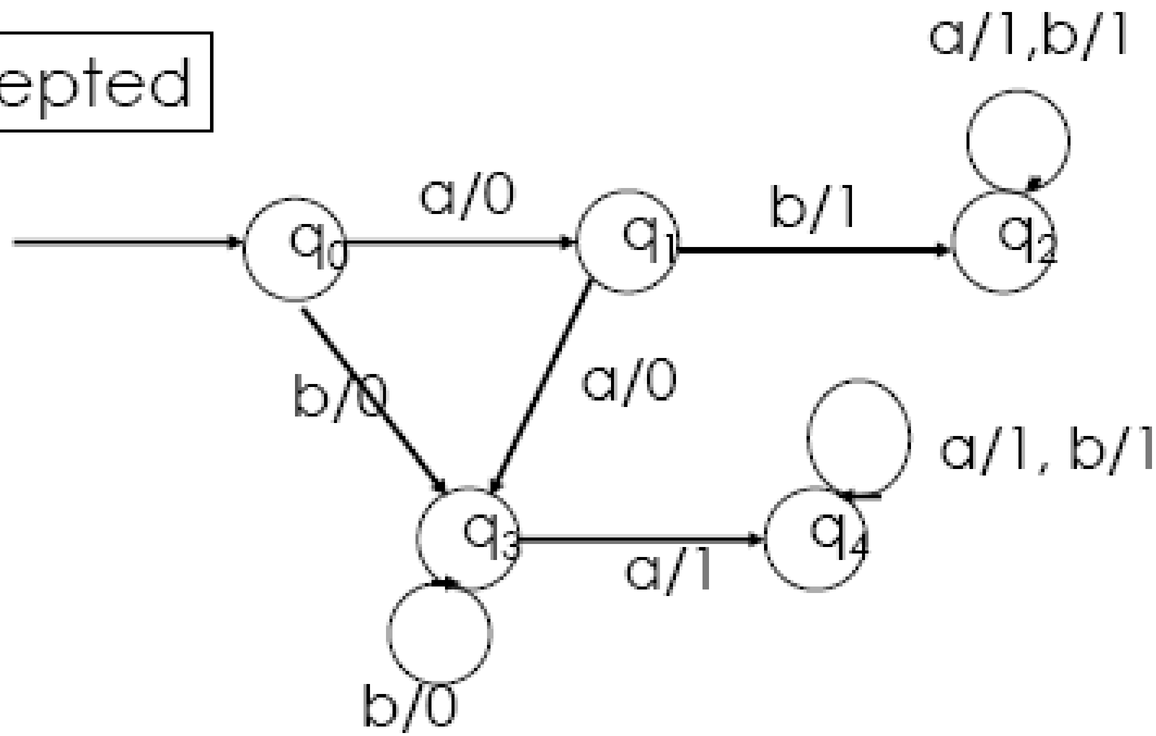


aabbba

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_3 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{a} q_4$$

0 0 0 0 0 1

Output: 1,accepted

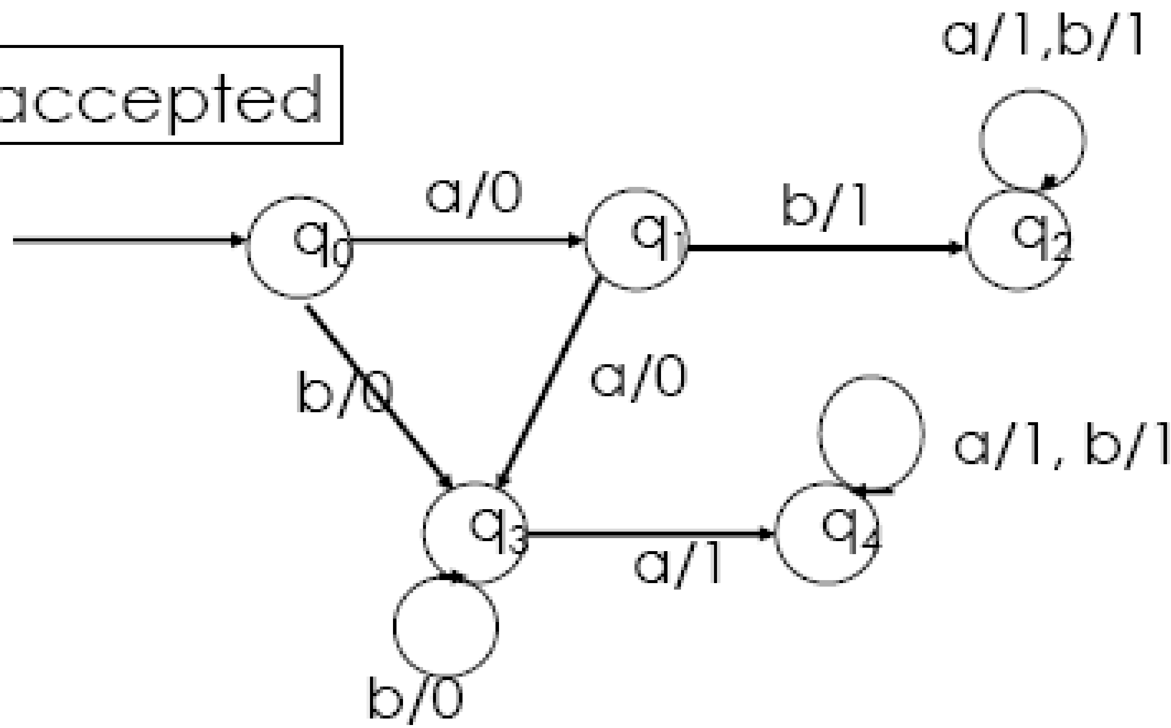


bbbb

$$q_0 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{b} q_3 \xrightarrow{b} q_3$$

0 0 0 0

Output: 0,not accepted

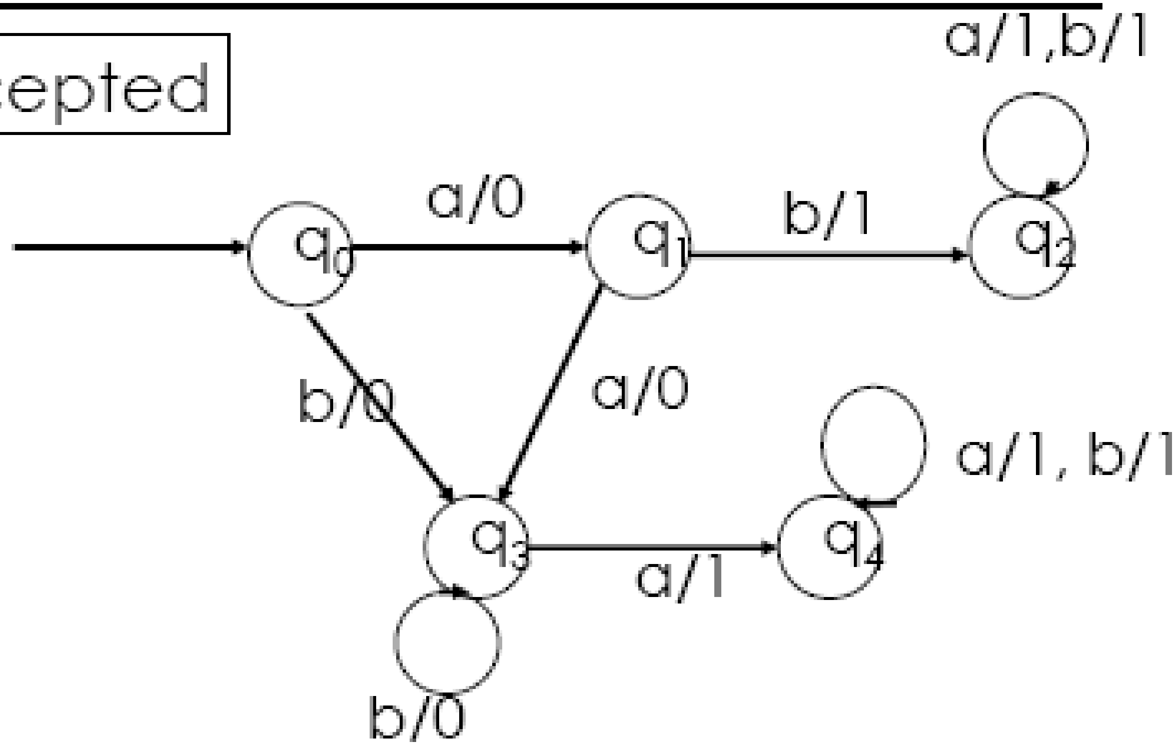


aaabbbb

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_3 \xrightarrow{a} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4 \xrightarrow{b} q_4$$

0 0 1 1 1 1 1

Output: 1,accepted



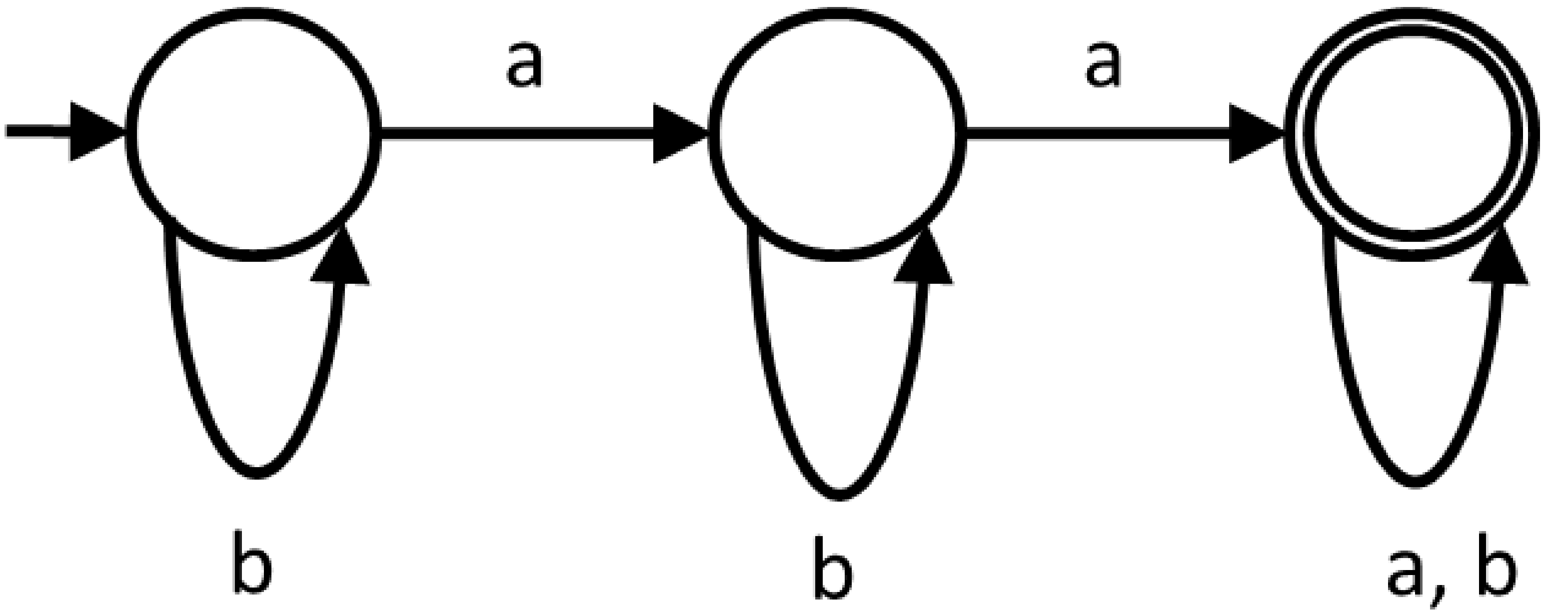
Exercise : Designing Finite Automata

1. Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with at least two a's.

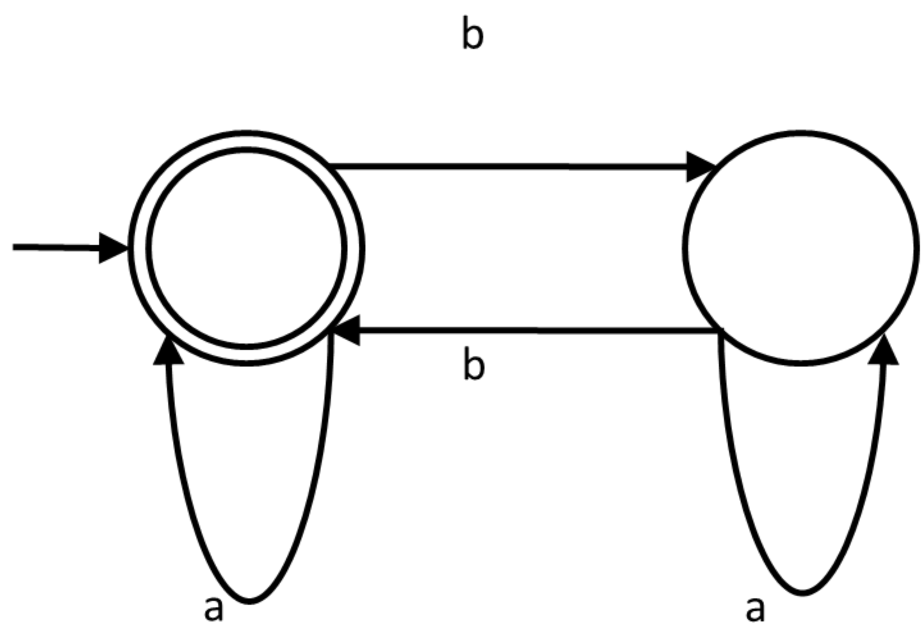
2. Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with an even number of b's.

3. Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with at least two a's and an even number of b's.

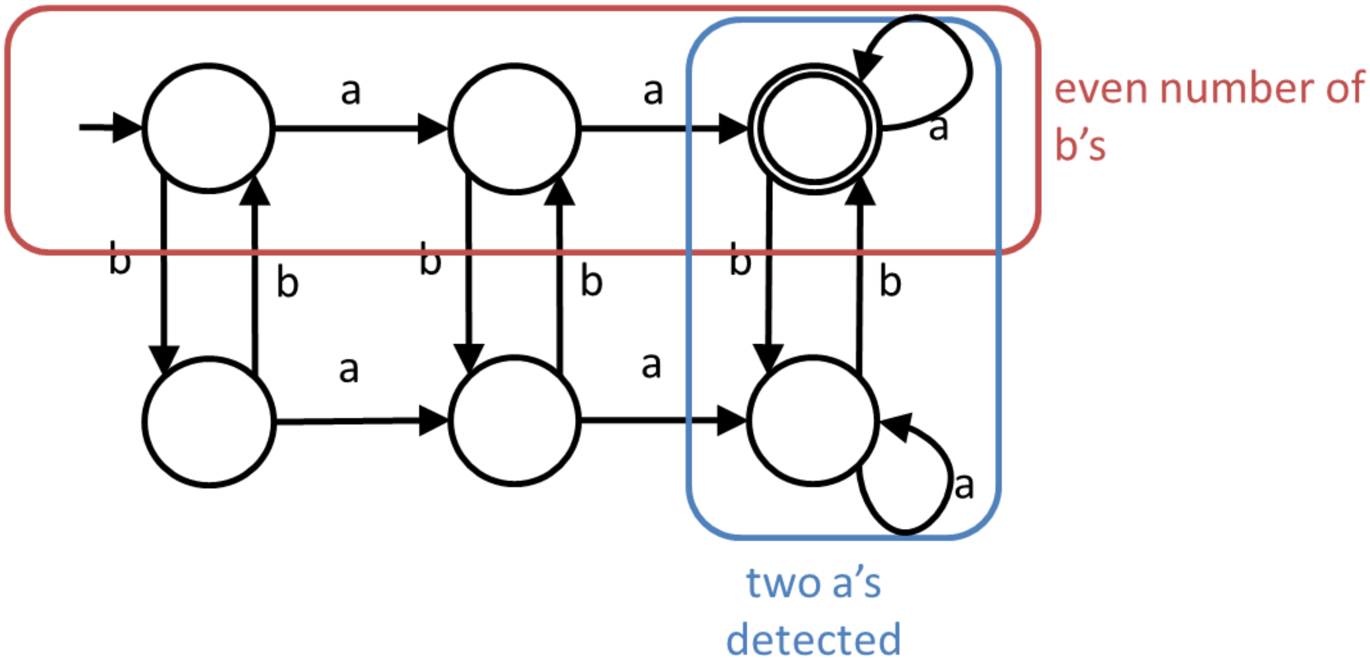
SOLUTION 1 Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with at least two a’s.



SOLUTION 2 Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with an even number of b's.



SOLUTION 3 Design a finite automaton (with input symbols a and b) that accepts the language consisting all sequences with at least two a's and an even number of b's.



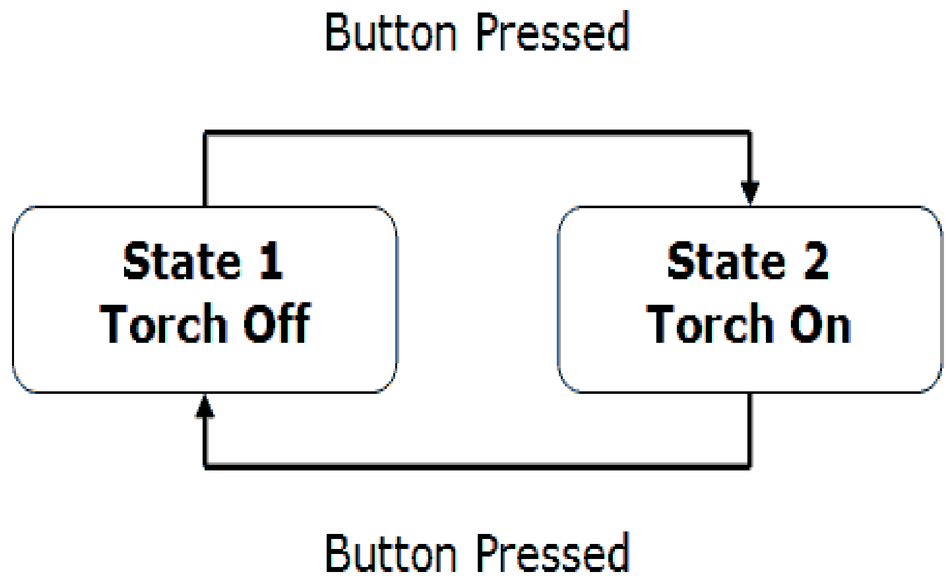
FSM Examples in daily live

- Vending Machines
- Traffic Lights
- Alarm Clock
- Microwave

Each of these devices can be thought of as a reactive system – that is because each of them work by reacting to signals or inputs from the external world

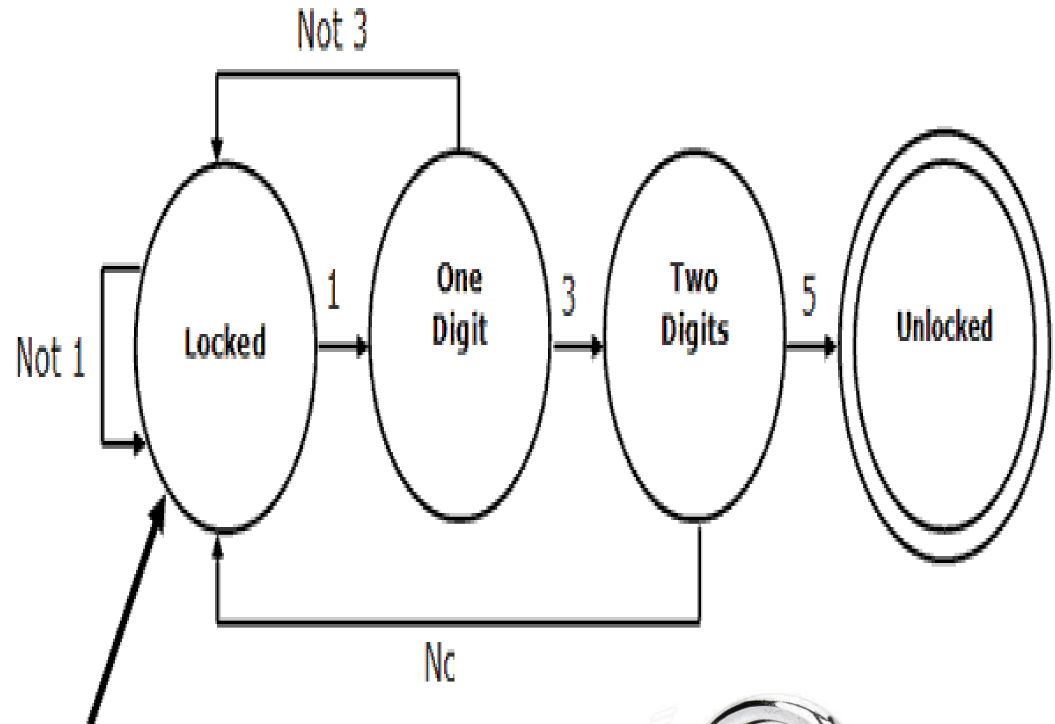
FSM Design - A simple torch

- The boxes are the possible states for the machine.
- The arrows indicate a transition between two states. The pointer indicates the direction of the change.
- Each transition is labelled with the input that caused the transition to occur.



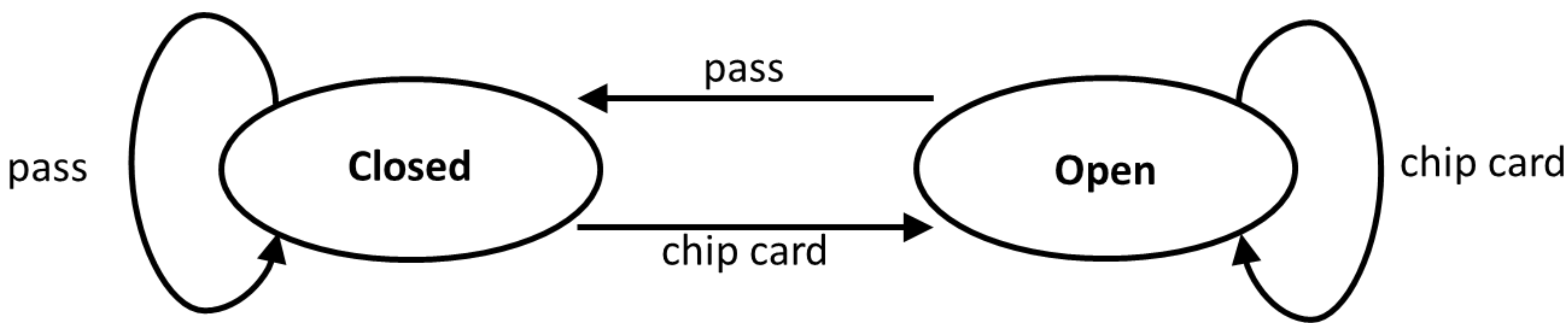
FSM Design - A combination lock

- The arrow pointing into the locked state indicates the initial state.
- The double circle for the unlocked state indicates the goal or accepting state.
- It has a keypad with the digits 0 - 9. You need to enter the combination 1, 3 & 5 to unlock the treasure.



FSM design - The chip card automaton

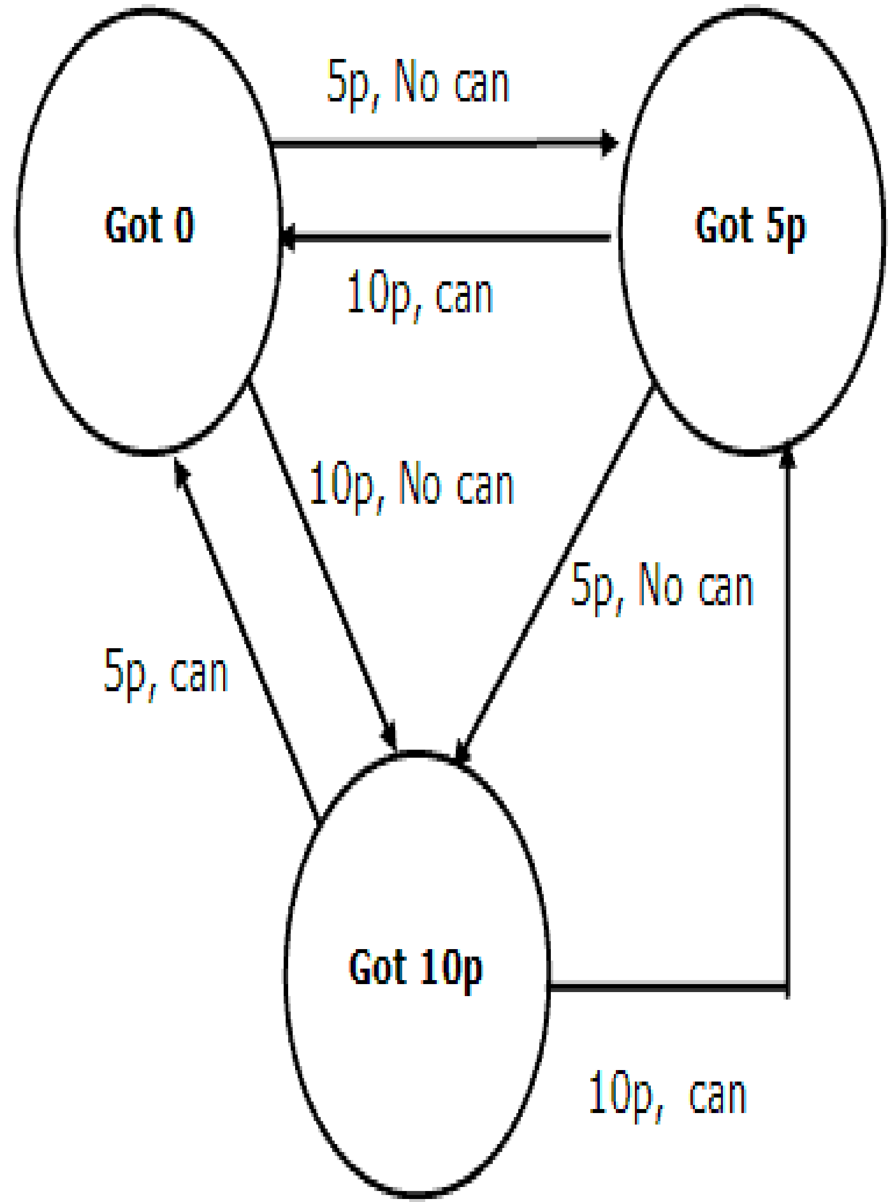
- The gate can be open or closed (i.e., it has two 'observed' states).
- When the gate is closed and the machine detects a valid chip card, it opens.
- When the gate is open and someone passes through, it closes.



FSM design – Carbonated water vending machine

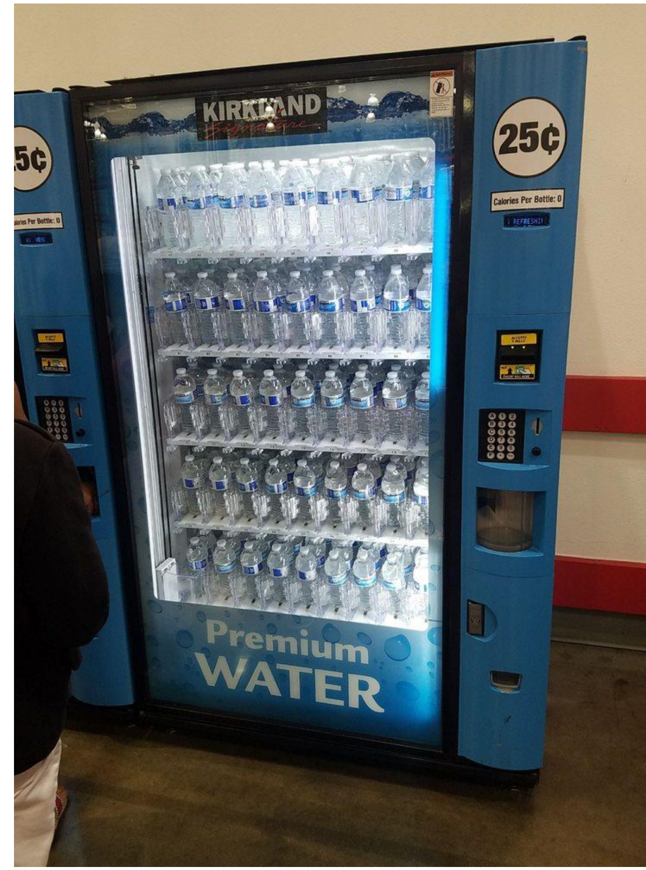
- Costs for each can of carbonated sugar water is 15p.
- When 15p has been inserted into the machine, a can of carbonated sugar water is released.
- Three states : Got 0, Got 5p and Got 10p.
- When the machine has no money entered towards the cost of the can, it is in the state **Got 0**.
- Inserting coinage changes the state.

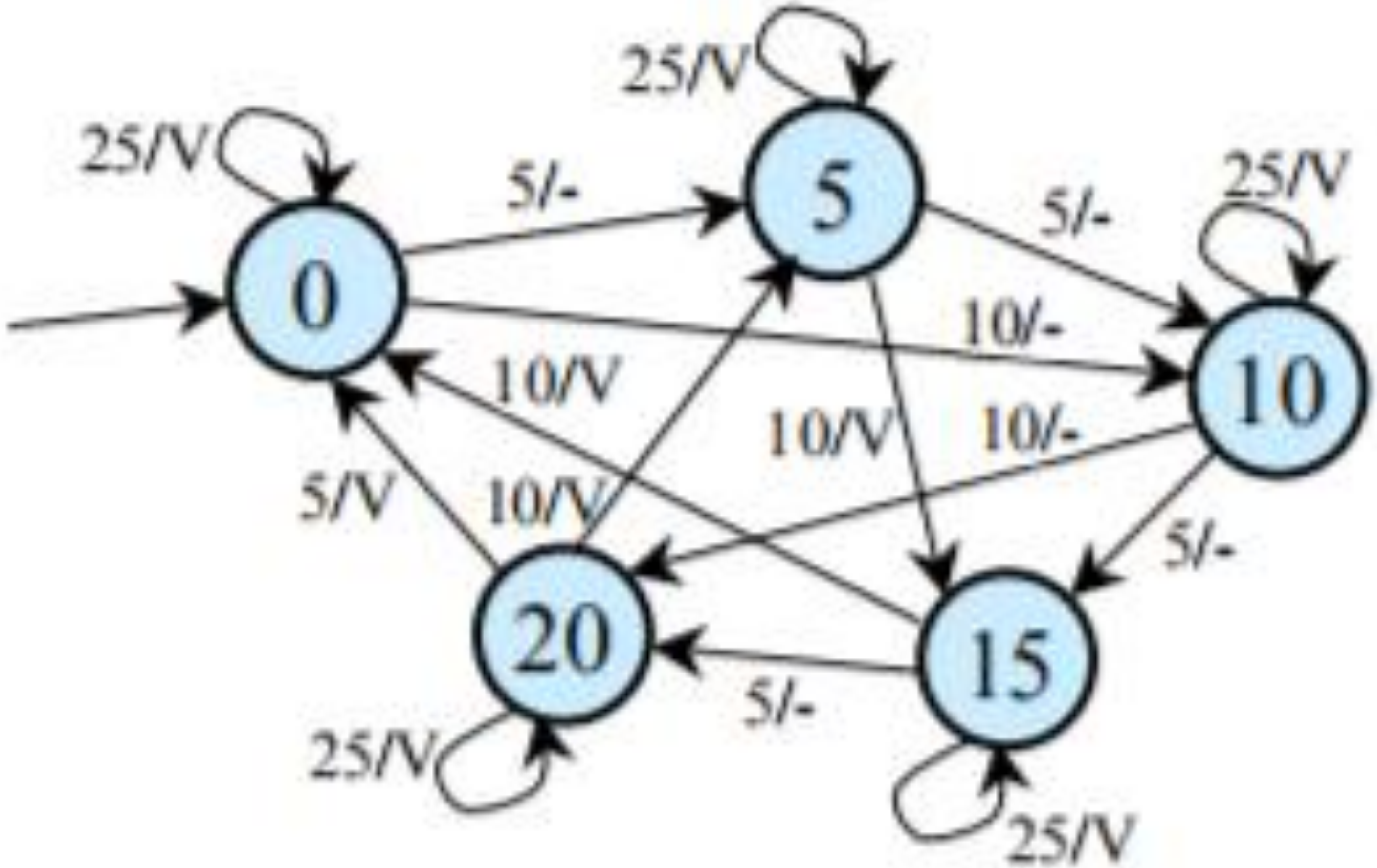
- Machine can provide for the different ways that you can arrive at 15p using the two coins. The machine accepts only 5p and 10p coinage.
- If two 10p coins are inserted, a can is released and the machine moves to the **Got5** state.
- An extension to this machine might be to cater for the release of change after or before the purchase is completed.



FSM Design – Water vending machine

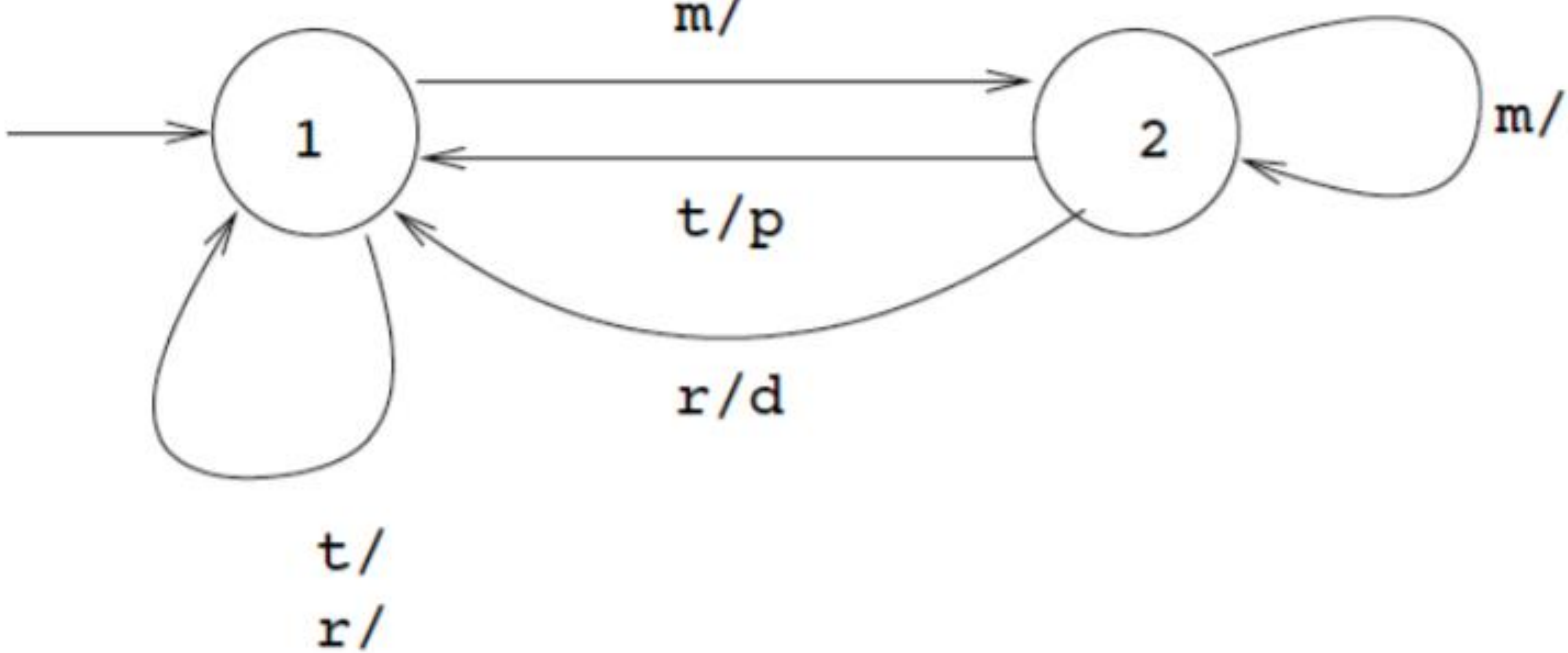
- Assume that all drinks are 25 cents.
- Machine receives coins only, allows user to select drink when the required amount has been met.
- If amount deposited is over 25 cents, retain the remainder.
- Assume that coin insertions are synchronized with the clock (i.e. implicit no-op transition if no coins are present).
- States : amount of money deposited (0, 5, 10, 15, 20).
- Transitions : coin being inserted (5, 10, 25).
- Output: “V” (deliver drink), “–” (no drink)




25/V
5/-
25/V
5
5/-
25/V
0
10/-
10/V
10
10/V
10/-
5/V
10/V
5/-
20
15
5/-
25/V
25/V

FSM Design – Ticket machine

- In the case of a parking ticket machine, it will not print a ticket when you press the button unless you have already inserted some money.
- Thus the response to the print button depends on the previous history of the use of the system.
- Inputs : inserting money (m), requesting ticket (t), requesting refund (r).
- Non-empty set of states : unpaid (1), paid (2).
- Outputs : print ticket (p), deliver refund (d)


m/
1
2
m/
t/p
r/d
t/
r/

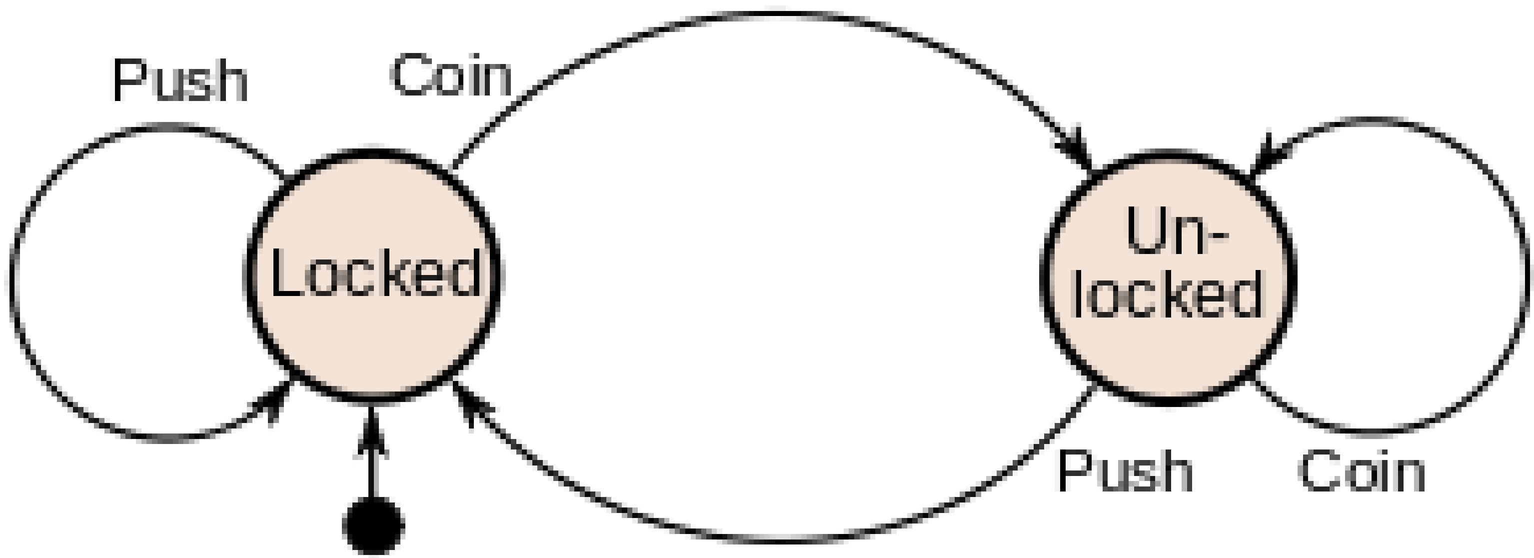
FSM Design – Turnstile machine

- Used to control access to subways and amusement park rides, is a gate with three rotating arms at waist height, one across the entryway.
- Two possible states: Locked and Unlocked. Two possible inputs that affect its state: putting a coin in the slot (coin) and pushing the arm (push).
- In the **locked** state, pushing on the arm has no effect; no matter how many times the input push is given, it stays in the locked state. Putting a coin in – that is, giving the machine a coin input – shifts the state from Locked to Unlocked.
- In the **unlocked** state, putting additional coins in has no effect; that is, giving additional coin inputs does not change the state. However, a customer pushing through the arms, giving a push input, shifts the state back to Locked.

State transition table

Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlocks the turnstile so that the customer can push through.
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When the customer has pushed through, locks the turnstile.

State diagram



FSM Design - Candy Machine

- Consider a vending machine that
  - Accepts nickels (5 cents), dimes (10 cents), and quarters (25 cents), crediting the amount.
  - If the total credit is more than 25, it returns the difference so only 25 cents credit remains.
  - Dispenses a candy bar if the candy button is pushed and there is 20 cents credit.
  - Dispenses a candy bar and returns 5 cents if the candy button is pushed and there is 25 cents credit.
  - Dispenses a soda if the soda button is pushed and there is 25 cents credit.

Candy Machine States

- The vending machine can be in different states based on the amount of money that has been credited to the user.
- Change is returned after 25 cents, and all coins are multiples of 5.
- Thus, the machine can be in the following states:
  - 0 cents credit (state S_0)
  - 5 cents credit (state S_1)
  - 10 cents credit (state S_2)
  - 15 cents credit (state S_3)
  - 20 cents credit (state S_4)
  - 25 cents credit (state S_5)

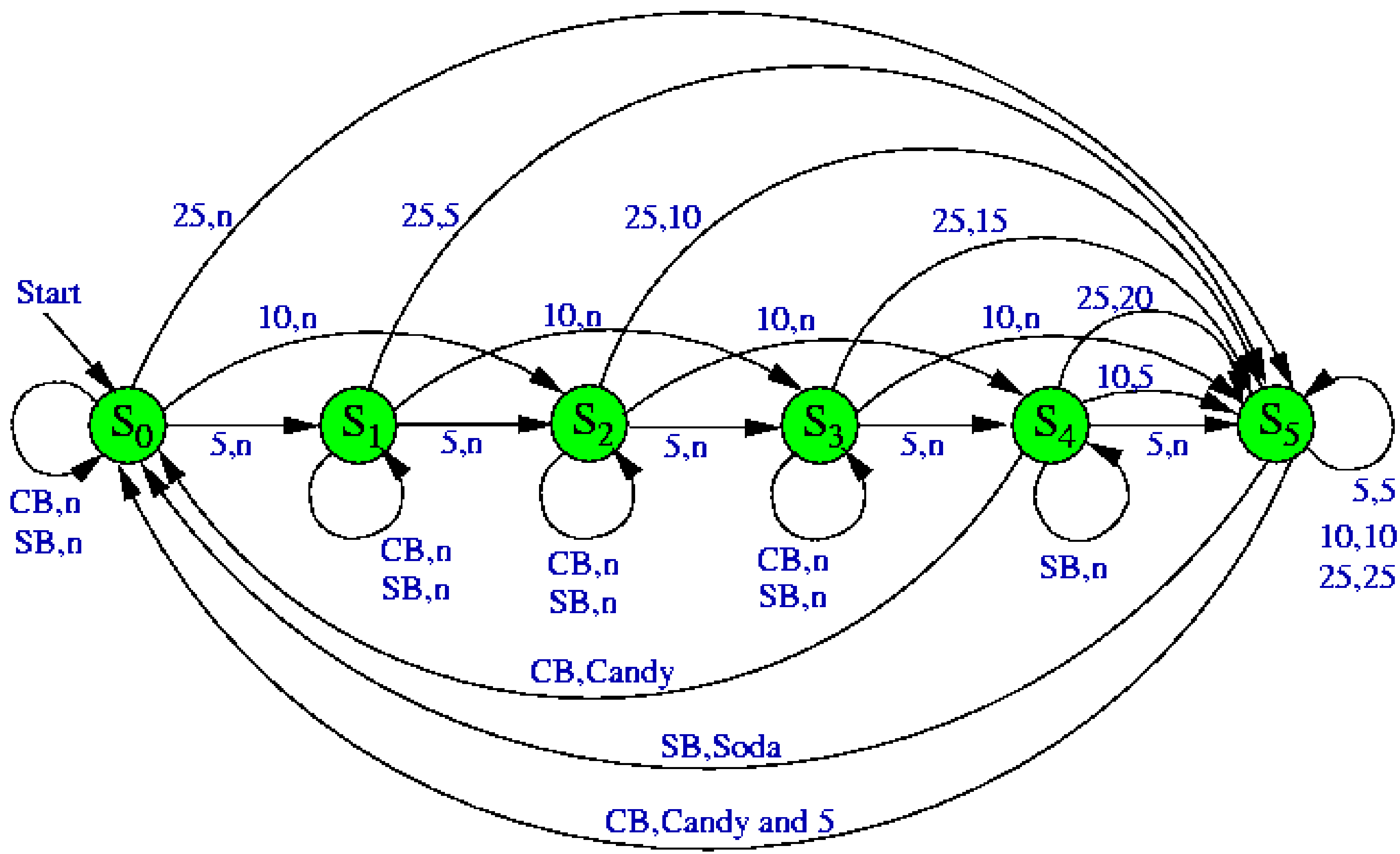
Candy Machine Input/Output

- The machine can accept the following inputs
  - A dime (10 cents) inserted
  - A nickel (5 cents) inserted
  - A quarter (25 cents) inserted
  - Candy button pushed (CB)
  - Soda button pushed (SB)
- The machine has the following possible outputs
  - A dime (10 cents) returned
  - A nickel (5 cents) returned
  - A quarter (25 cents) returned
  - A candy bar (C) dispensed
  - A soda (S) dispensed
  - Nothing (n) is returned or dispensed

Candy Machine FSM

- We now have enough information to construct our finite-state machine.
- For each possible input and each possible state, we need to know what to output (if anything) and what state the machine should go to.
- For instance:
  - If the machine is in state S_3 (15 cents credit) and
  - a quarter (25 cents) is input
  - the machine should transition into state S_5 (25 cents credit) and
  - 15 cents (a dime and nickel) should be output.
- We can construct a state diagram and/or a state table by considering every possible input on every possible state.

Candy Machine State Diagram



Candy Machine State Table

	Next State					Output				
	Input					Input				
State	5	10	25	CB	SB	5	10	25	CB	SB
S_0	S_1	S_2	S_5	S_0	S_0	n	n	n	n	n
S_1	S_2	S_3	S_5	S_1	S_1	n	n	5	n	n
S_2	S_3	S_4	S_5	S_2	S_2	n	n	10	n	n
S_3	S_4	S_5	S_5	S_3	S_3	n	n	15	n	n
S_4	S_5	S_5	S_5	S_0	S_4	n	5	20	Candy	n
S_5	S_5	S_5	S_5	S_0	S_0	5	10	25	Candy,5	Soda

FSM Definition

- **Definition:** A *finite-state machine* is a 6-tuple $M=(S, I, O, f, g, S_0)$ where
  - S is a finite set of *states*
  - I is a finite *input alphabet*
  - O is a finite *output alphabet*
  - $f: S \times I \rightarrow S$ is a *transition function* from each state-input pair to a state
  - $g: S \times I \rightarrow O$ is a *output function* from each state-input pair to an output
  - S_0 is the *initial state*

FSM Representations

- As we have already seen, there are two common ways of representing finite-state machines
  - A *state table* is used to represent a finite-state machine by giving the values of the functions *f* and *g*.
  - A *state diagram* is a directed graph representation of a finite-state machine.

State Tables

- A *state table* is organized as follows

The rows are indexed by the states.

The columns are split into two groups:

- The first half are indexed by the *inputs*
- The entries in the table give the value of the function *f* – that is the *new states*

- The second half are also indexed by the *inputs*
- The entries in the table give the value of the function *g* – that is, the *outputs*.

	Next State					Output				
	Input					Input				
State	5	10	25	CB	SB	5	10	25	CB	SB
S_0	S_1	S_2	S_5	S_0	S_0	n	n	n	n	n
S_1	S_2	S_3	S_5	S_1	S_1	n	n	5	n	n
S_2	S_3	S_4	S_5	S_2	S_2	n	n	10	n	n
S_3	S_4	S_5	S_5	S_3	S_3	n	n	15	n	n
S_4	S_5	S_5	S_5	S_0	S_4	n	5	20	Candy	n
S_5	S_5	S_5	S_5	S_0	S_0	5	10	25	Candy,5	Soda

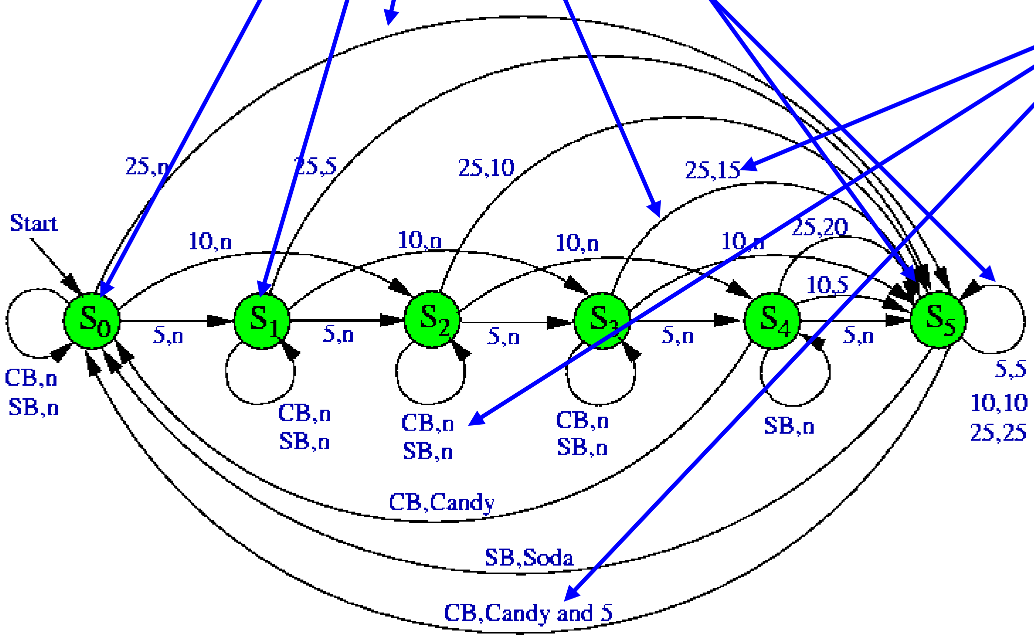
State Diagram

- A *state diagram* is organized as follows
  - The *nodes* in the graph represent the states.

The *edges* in the graph represent the *transitions*.

An *edge* (S_i, S_j) occurs if some *input* causes a transition from S_i to S_j

Each *edge* is labeled with a pair *(x,y)*, where



- *x* is the *input* which (along with the state) causes the transition
- *y* is the *output* triggered by the state-input pair.

Example: Unit Delay

- In some electronic devices, it is necessary to use a *unit-delay machine*.
- That is, whatever is input into the machine should be output from the machine, but delayed by a specific amount of time.
- For instance, given a string of binary numbers $x_1, x_2, \ldots, x_n$, the machine should produce the string $0, x_1, x_2, \ldots, x_{n-1}$.
- We want to use a finite state machine to model the behavior of a *unit-delay machine*.
- What should a state in this machine represent?
- One possibility is that a state represents the last input bit.
- Thus we need a state for “1” and a state for “0”
- We also need start state.

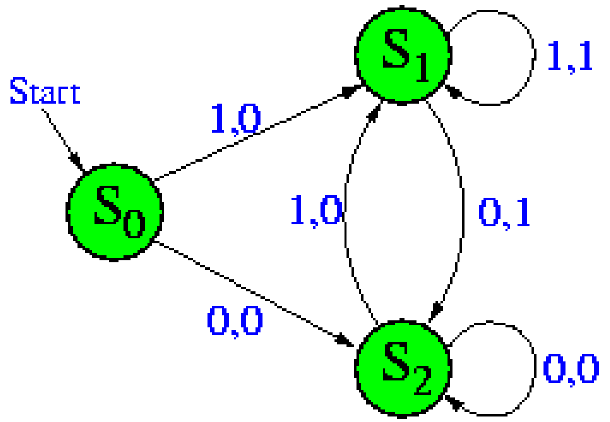
Unit Delay States

- We will use the following states (that memorize last bit; except S_0)
  - State S_0 is the start state
  - State S_1 occurs if the previous input was 1
  - State S_2 occurs if the previous input was 0
- We can easily construct the state table for the unit delay machine by realizing that
  - When the input is 0, we always transition to state S_2
  - When the input is 1, we always transition to state S_1
  - When we are in state S_1 we always output 1 (since the previous input was 1)
  - When we are in state S_2 we always output 0 (since the previous input was 0)
  - When we are in state S_0 we always output 0 (since we always output 0 first)

Unit Delay State Table/Diagram

- Here is the state table and state diagram based on our previous observations.

	Next State		Output	
	Input		Input	
State	0	1	0	1
S_0	S_2	S_1	0	0
S_1	S_2	S_1	1	1
S_2	S_2	S_1	0	0



What's output of 101011?

FSM Design – Binary adder

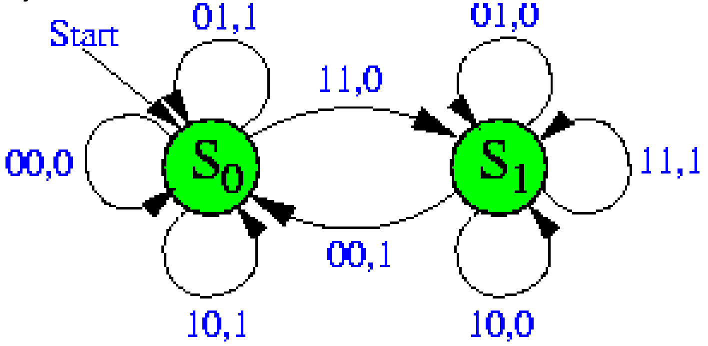
- We want to construct a finite state machine that will add two numbers.
- The input is two binary numbers, $(x_n \ldots x_1 x_0)_2$ and $(y_n \ldots y_1 y_0)_2$
- At each step, we can compute (x_i+y_i) starting with (x_0+y_0).
  - If $(x_i+y_i)=0$, we output 0.
  - If $(x_i+y_i)=1$, we output 1.
  - If $(x_i+y_i)=2$, we have a problem.
- The problem is we need a carry bit.
- In fact, our computation needs to know the carry bit at each step (so we compute $x_i+y_i+c_i$ at each step), and be able to give it to the next step.
- We can take care of this by using states to represent the carry bit.

Binary Adder States

- We will use the following states
  - State S_0 occurs if the carry bit is 0
  - State S_1 occurs if the carry bit is 1
- Since when we begin the computation, there is no carry, we can use S_0 as the start state,
- So, how does which state we are in affect the output?
- If we are in state S_0 (we have a carry of 0)
  - If $(x_i+y_i)=0$, we output 0, and stay in state S_0
  - If $(x_i+y_i)=1$, we output 1, and stay in state S_0
  - If $(x_i+y_i)=2$, we output 0, and go to state S_1
- If we are in state S_1 (we have a carry of 1)
  - If $(x_i+y_i+1)=1$, we output 1, and go to state S_0
  - If $(x_i+y_i+1)=2$, we output 0, and stay in state S_1
  - If $(x_i+y_i+1)=3$, we output 1, and stay in state S_1

Binary Adder State Table/Diagram

- From the previous observations, we can construct the state table and state diagrams
- for the binary adder



	Next State				Output			
	Input				Input			
State	00	01	10	11	00	01	10	11
S_0	S_0	S_0	S_0	S_1	0	1	1	0
S_1	S_0	S_1	S_1	S_1	1	0	0	1

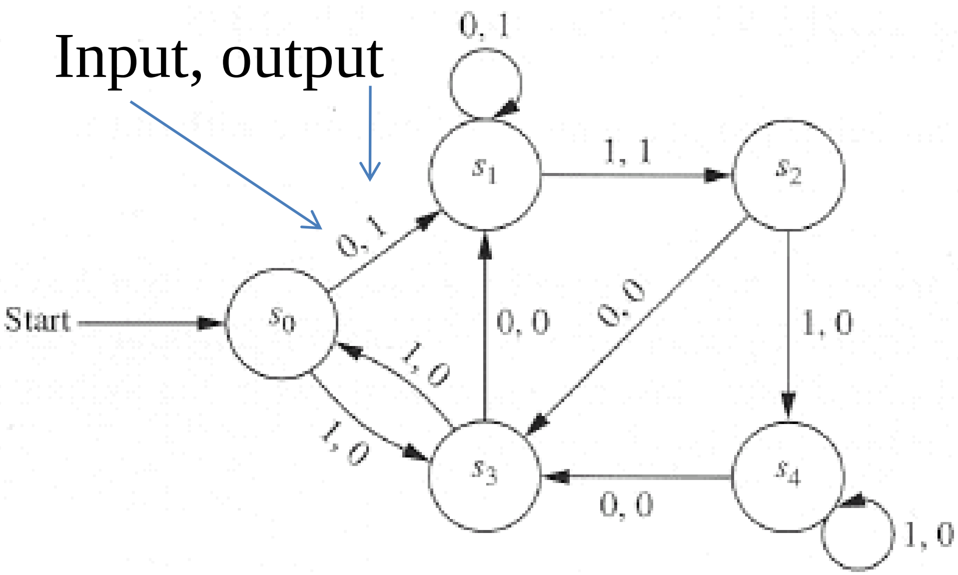


FIGURE 3 **A Finite-State Machine.**

TABLE 3

State	*f* Input 0	*f* Input 1	*g* Input 0	*g* Input 1
s_0	s_1	s_3	1	0
s_1	s_1	s_2	1	1
s_2	s_3	s_4	0	0
s_3	s_1	s_0	0	0
s_4	s_3	s_4	0	0

Output of 101011?

001000