COMP 2121
DISCRETE MATHEMATICS

Lecture 11

[1]

Finite State Machines

Combinatorial Circuit is characterized by the fact that its output is completely determined by its input/output table (truth table), or, in other words by Boolean function.

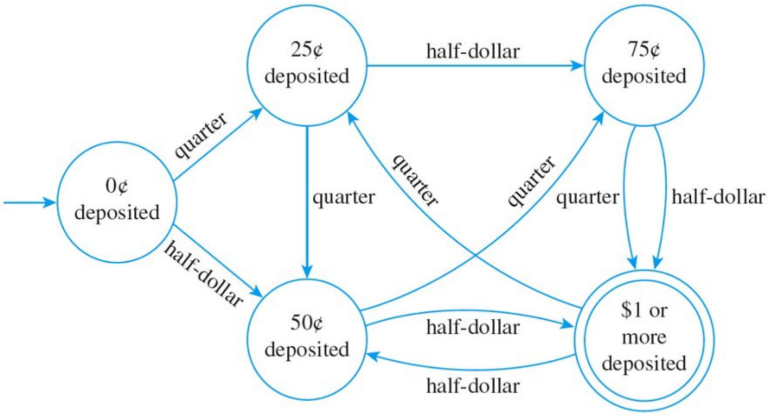
For *Sequential* Circuits one cannot predict the output corresponding to a particular input unless one also knows something about the prior history of the circuit, or, more technically, unless one knows the state the circuit was in before receiving the input. A computer memory circuit is type of the sequential circuit.

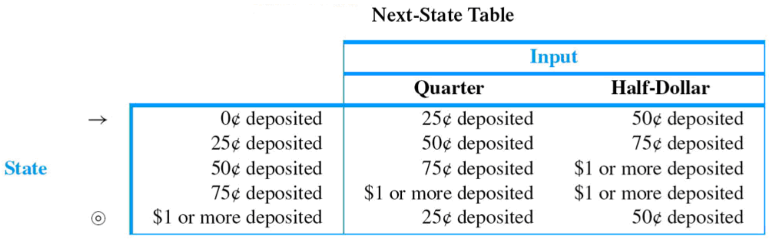
Finite State Machine (FSM) is an idealized machine that embodies the essential idea of sequential circuit. Each piece of input to a finite state machine leads to a change in the state, which in turn affects how subsequent input is processed.

Example 1. A Simple Vending Machine

- A simple vending machine dispenses bottles of juice that cost $1 each.
- The machine accepts quarters and half-dollars only and does not give change.
- As soon as the amount deposited equals or exceeds $1 the machine releases a bottle of juice.
- The next coin deposited starts process over again.

[1] http://www.kryptonitekollectibles.com/images/sce/chhomereatingmachine2.jpg





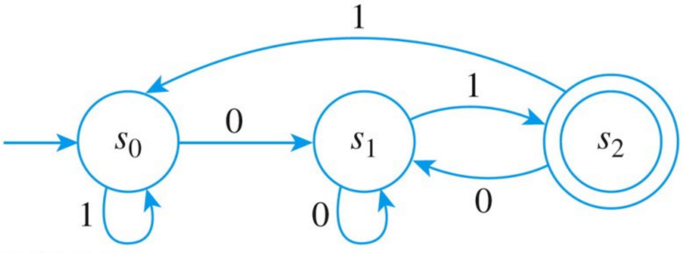
Next-State Table

		Input	
		Quarter	**Half-Dollar**
→	0¢ deposited	25¢ deposited	50¢ deposited
	25¢ deposited	50¢ deposited	75¢ deposited
State	50¢ deposited	75¢ deposited	$1 or more deposited
	75¢ deposited	$1 or more deposited	$1 or more deposited
◎	$1 or more deposited	25¢ deposited	50¢ deposited

A finite state machine A consists of following objects:

- a set I, called input alphabet, of input symbols;
- a set S of states the machine can be in; Set must include
  - one designated state s_0 called initial state;
  - designated set of states called the set of accepting states;
- a next-state function N: S x I → S that associates a "next – state" to each ordered pair consisting of a "current state" and a "current input". For each state s in S and input symbol m in I, N(s, m) is called the state to which A goes if m is input to A when A is in state s.

Example 2. Finite State Machine given by Transition Diagram



What are the states of A?

What are the input symbols of A?

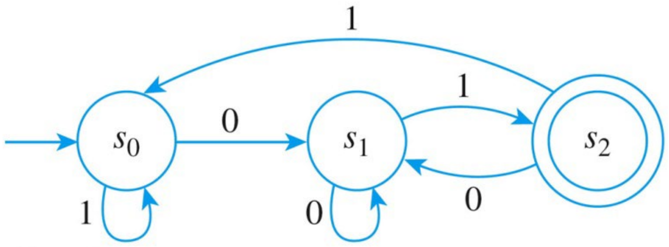
What is the initial state of A?

What are the accepting states of A?

Find next-state table for A.

Finding the Language Accepted by a Machine

Example 3.



a) To what state does A go for the following inputs:

- 01
- 0011
- 0101100
- 10101

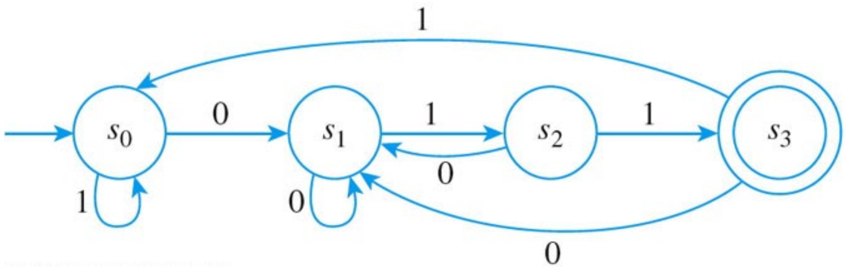
b) Consider the strings that send A to the accepting stage. What is the language accepted by A?

Designing a Finite – State Machine

Example 4. Design a finite – state machine A that accepts the set of all string of 0's and 1's such that the number of 1's in the string is divisible by 3.

Simulating a Finite – State Machine (Implementing FSM)

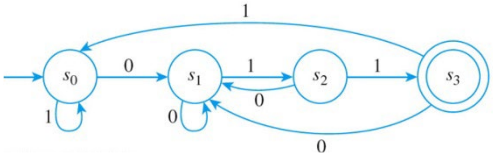
Example 5. The following FSM recognizes strings that end at 011.



Algorithm

Input: string [a string of 0's and 1's which ends with marker e - end of the string]

```
state:=0

symbol := first symbol in the input string

while (symbol =! e)

    if       state = 0 then   if symbol = 0
                                  then state:=1
                                  else state :=0
    else if  state = 1 then   if symbol = 0
                                  then state:=1
                                  else state :=2
    else if  state = 2 then   if symbol = 0
                                  then state:=1
                                  else state :=3
    else if  state = 3 then   if symbol = 0
                                  then state:=1
                                  else state :=0

    symbol := next symbol in the input string

end while
```

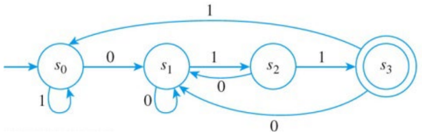


input state	0	1
S_0	S_1	S_0
S_1	S_1	S_2
S_2	S_1	S_3
S_3	S_1	S_0

Output: state

Alternative Algorithm

Input: string [a string of 0's and 1's which ends with marker e - end of the string]



input state	0	1
S_0	S_1	S_0
S_1	S_1	S_2
S_2	S_1	S_3
S_3	S_1	S_0

```
N[0,0]:=1, N[0,1]:=0, N[1,0]:=1, N[1,1]:=2,
N[2,0]:=1 , N[2,1]:=3, N[3,0]:=1, N[3,1]:=0

state: = 0
symbol:= first symbol in the input string

while (symbol != e)

        state:= N[state, symbol]

        symbol:= next symbol in the input string

end while
```

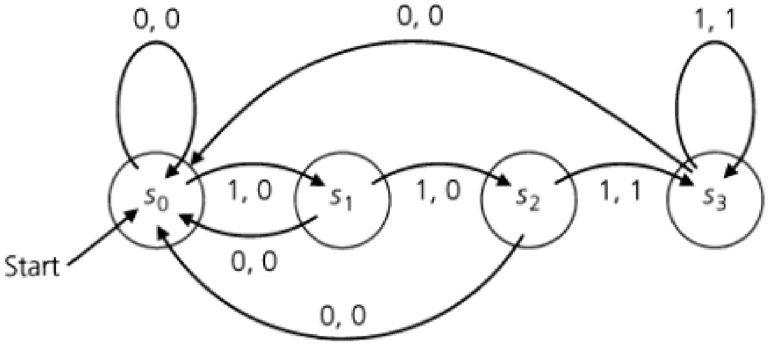
Output: state

Additional Design Problems

Example 6. Design FSM that recognizes strings that a) end with 101, b) contain 101.

Example 7. Design FSM that recognizes strings that contain 101 and an odd number of 0's.

NOTE: Your book does not designate the final state with the double circle. Instead, it outputs 1 when string is accepted (final state) and it outputs 0 when string is not accepted. All we need to do is to check if the last output is 0 or 1.



Supplementary examples from the book: 6.17, 6.18, 6.19, 6.20, 6.21.

Relevant exercises: 6.2: 1, 2, 3, 4, 5, 6, 8, 9.
6.3: 1, 2, 3, 4, 5,