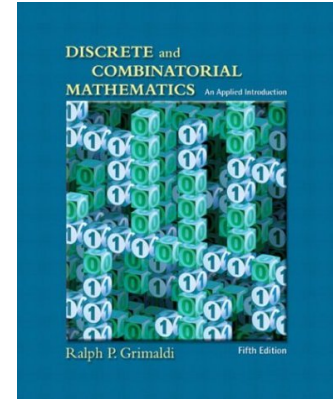
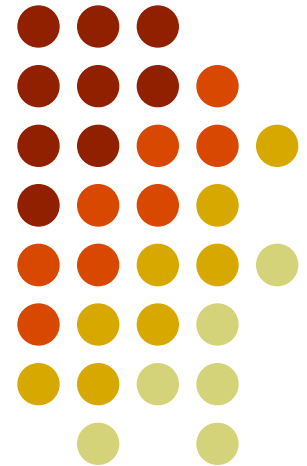


# Discrete Mathematics

## -- Chapter 6: Languages: Finite State Machine



*Hung-Yu Kao (高宏宇)*  
*Department of Computer Science and Information Engineering,*  
*National Cheng Kung University*

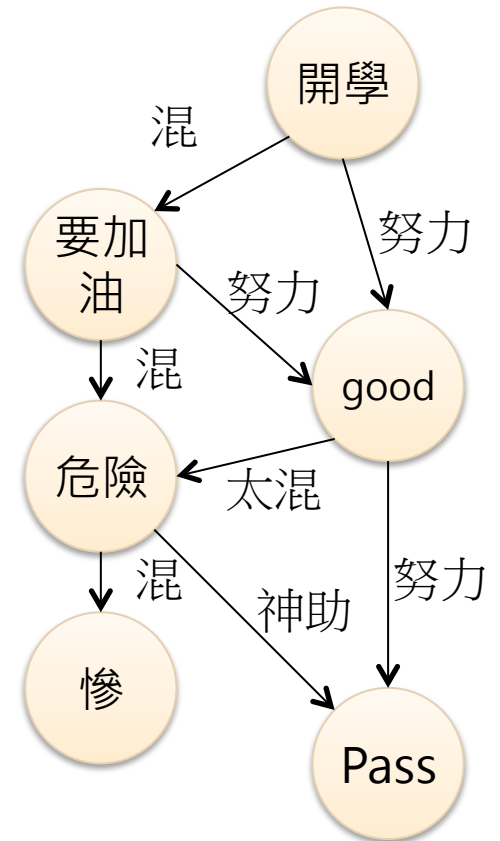
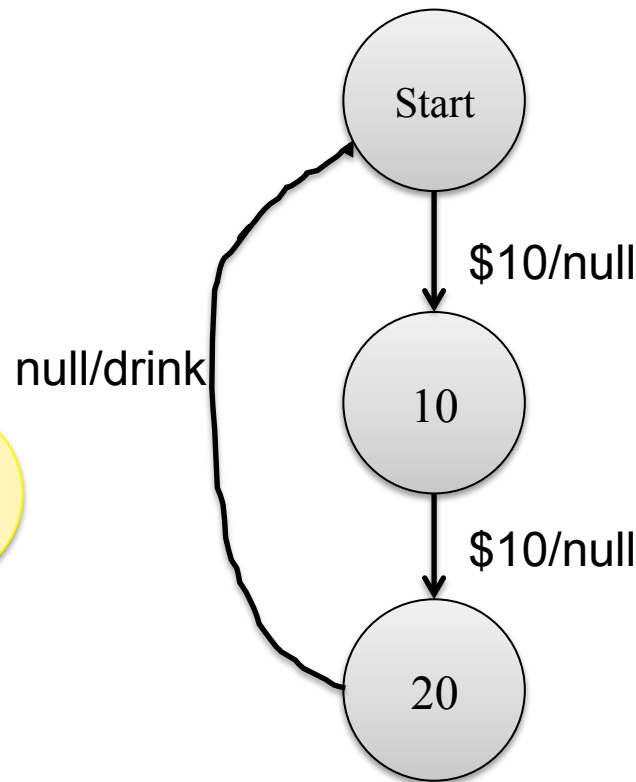
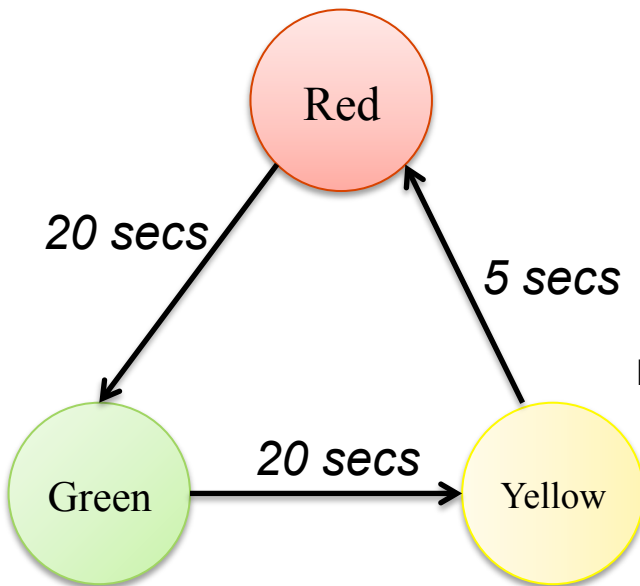




# Outline

- 6.1 Language: The Set Theory of Strings
- 6.2 Finite State Machines: A First Encounter
- 6.3 Finite State Machines: A Second Encounter

# Finite State Machine (有限狀態機)





# Language: The Set Theory of Strings

- A **finite state machine (FSM)**, which is an abstract model, has a finite number of internal states where the machine remembers certain information when it is in a particular state.
- Strings: Sequence of symbols (characters) play a key role in the processing of information by a computer.
- $\Sigma$  denote a nonempty finite set of symbols, collectively called an alphabet. E.g.,  $\Sigma = \{0, 1\}$ ,  $\Sigma = \{a, b, c, d, e\}$ .
- Definition 6.1: If  $\Sigma$  is an alphabet and  $n \in \mathbb{Z}^+$ , we define the powers of  $\Sigma$  recursively as follows:

$$1) \Sigma^1 = \Sigma$$

$$2) \Sigma^{n+1} = \{xy \mid x \in \Sigma, y \in \Sigma^n\}, \text{ where } xy \text{ denotes the juxtaposition of } x \text{ and } y$$



# Language: The Set Theory of Strings

- **Ex 6.1** : Let  $\Sigma$  an alphabet.

If  $n = 2$ ,  $\Sigma^2 = \{xy \mid x \in \Sigma, y \in \Sigma\}$ , e.g.,  $\Sigma = \{0,1\}$ ,  $\Sigma^2 = \{00,01,10,11\}$

$\Sigma = \{a,b,c,d,e\}$ ,  $\Sigma^3$  would contain  $5^3$  three - symbol strings, e.g.,  $aaa,acb,cdd$ , etc.

- Definition 6.2: For an alphabet  $\Sigma$  we define  $\Sigma^0 = \{\lambda\}$ , where  $\lambda$  denotes the empty string, i.e., the string consisting of no symbols taken from  $\Sigma$ .

$$(1) \{\lambda\} \not\subseteq \Sigma \text{ since } \lambda \notin \Sigma$$

$$(2) \{\lambda\} \neq \phi \text{ because } |\{\lambda\}| = 1 \neq 0 = |\phi|$$

- Definition 6.3: If  $\Sigma$  is an alphabet, then

$$(1) \Sigma^+ = \bigcup_{n=1}^{\infty} \Sigma^n = \bigcup_{n \in \mathbb{Z}^+} \Sigma^n$$

$$(2) \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$



# Language: The Set Theory of Strings

- **Ex 6.2** : Let  $\Sigma = \{0,1\}$  the set  $\Sigma^*$  consists of all finite strings of 0's and 1's together with the empty string. (how about  $\Sigma^+$ ?)
- If  $\Sigma = \{\beta, 0, 1, \dots, 9, +, -, /, \}$ , where  $\beta$  denotes the blank (space). Here in  $\Sigma^*$  we find familiar arithmetic expression such as  $(7+5)/(2-3)$ .
- Definition 6.4:  
If  $w_1, w_2 \in \Sigma^+$ ,  $w_1 = x_1 x_2 \cdots x_m$ ,  $w_2 = y_1 y_2 \cdots y_n$   
and  $x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n \in \Sigma$ ,  
then we say  $w_1$  and  $w_2$  are equal ( $w_1 = w_2$ ) if  $m = n$  and  $x_i = y_i$  for all  $i$ .



# Language: The Set Theory of Strings

- Definition 6.5:

Let  $w = x_1x_2 \cdots x_n \in \Sigma^+$ , where  $x_i \in \Sigma$  for  $1 \leq i \leq n$ .

The length of  $w$  is  $n$ , denoted by  $\|w\|$ , and  $\|\lambda\| = 0$ .

- Definition 6.6:

Let  $x, y \in \Sigma^+$ ,  $x = x_1x_2 \cdots x_m$ ,  $y = y_1y_2 \cdots y_n$ ,

and  $x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n \in \Sigma$ .

The concatenation of  $x$  and  $y$ :  $xy = x_1x_2 \cdots x_my_1y_2 \cdots y_n$ .

The concatenation of  $x$  and  $\lambda$ :  $x\lambda = x$ ,  $\lambda x = x$ .

The concatenation of  $\lambda$  and  $\lambda$ :  $\lambda\lambda = \lambda$ .



# Language: The Set Theory of Strings

- Definition 6.7:

$x \in \Sigma^*$ , we define the powers of  $x$  by  $x^0 = \lambda$ ,  $x^1 = x$ ,  $x^2 = xx$ ,  
 $x^3 = xx^2, \dots, x^{n+1} = xx^n, \dots$ , where  $n \in \mathbb{N}$ .

- Definition 6.8:

If  $x, y \in \Sigma^*$  and  $w = xy$ ,

then  $x$  is called a prefix of  $w$ , and if  $y \neq \lambda$ , then  $x$  is to be a proper prefix.

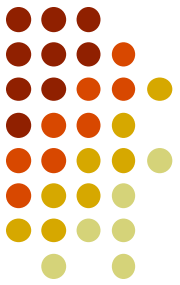
Similarly,  $y$  is called a suffix of  $w$ , it is a proper suffix when  $x \neq \lambda$ .

- Definition 6.9:

If  $x, y, z \in \Sigma^*$  and  $w = xyz$ , then  $y$  is called a substring of  $w$ .

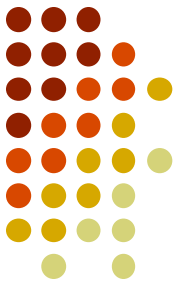
When at least one of  $x$  and  $z$  is different from  $\lambda$ , we call  $y$  a proper substring.





# Language: The Set Theory of Strings

- Definition 6.10:  
For a given alphabet  $\Sigma$ , any subset of  $\Sigma^*$  is called a language over  $\Sigma$ .  
This includes the subset  $\emptyset$ , which we call the empty language.
- Ex 6.9 :
  - With  $\Sigma$  the alphabet of 26 letters, 10 digits, and the special symbols used in a given implementation of C++, the collection of executable programs for that implementation constitutes a language.
  - In the same situation, **each** executable program could be considered a language, as could a particular set of such programs.
- Since **languages** are **sets**, we can form the union, intersection, and symmetric difference of two languages.



# Language: The Set Theory of Strings

- Definition 6.11:

For an alphabet  $\Sigma$  any languages  $A, B \subseteq \Sigma^*$ , the concatenation of  $A$  and  $B$ , denoted  $AB$ , is  $\{ab \mid a \in A, b \in B\}$ .

- Ex 6.10 :

Let  $\Sigma = \{x, y, z\}$ , and let  $A, B$  be the finite languages  $A = \{x, xy, z\}$ ,  $B = \{\lambda, y\}$ .

Then  $AB = \{x, xy, z, xyy, zy\}$  and  $BA = \{x, xy, z, yx, yxy, yz\}$

1)  $|AB| = 5 \neq 6 = |BA|$

2)  $|AB| = 5 \neq 6 = 3 \cdot 2 = |A| |B|$



# Language: The Set Theory of Strings

- Theorem 6.1:

For an alphabet  $\Sigma$ , let  $A, B, C \subseteq \Sigma^*$ .

a)  $A\{\lambda\} = \{\lambda\}A = A$

b)  $(AB)C = A(BC)$

c)  $A(B \cup C) = AB \cup AC$

d)  $(B \cup C)A = BA \cup CA$

e)  $A(B \cap C) \subseteq AB \cap AC$

f)  $(B \cap C)A \subseteq BA \cap CA$

**Proof :**

(f) For  $x \in \Sigma^*$ ,

$$x \in (B \cap C)A \Rightarrow x = yz \text{ where } y \in B \cap C \text{ and } z \in A$$

$$\Rightarrow (x = yz \text{ for } y \in B \text{ and } z \in A) \text{ and } (x = yz \text{ for } y \in C \text{ and } z \in A)$$

$$\Rightarrow x \in BA \text{ and } x \in CA$$

$$\Rightarrow x \in BA \cap CA$$

$$\therefore (B \cap C)A \subseteq BA \cap CA$$

$$B = \{x, xx, y\}, C = \{y, xy\}, A = \{y, yy\}$$
$$xyy \in BA \cap CA, xyy \notin (B \cap C)A$$



# Language: The Set Theory of Strings

- Definition 6.12:

For a given language  $A \subseteq \Sigma^*$  we can construct other languages as follows:

a)  $A^0 = \{\lambda\}$ ,  $A^1 = A$ , and  $n \in \mathbb{Z}^+$ ,  $A^{n+1} = \{ab \mid a \in A, b \in A^n\}$

b)  $A^+ = \bigcup_{n \in \mathbb{Z}^+} A^n$ , the positive closure of  $A$ .

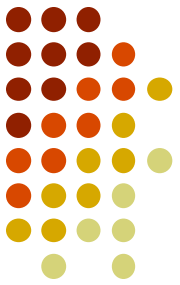
c)  $A^* = A^+ \cup \{\lambda\}$ . The language  $A^*$  is called the Kleene closure of  $A$ .

(in honor of the American logician Stephen Cole Kleene, 1909 - 1994)

- Ex 6.11 :

If  $\Sigma = \{x, y, z\}$ , and  $A = \{x\}$ , then (1)  $A^0 = \{\lambda\}$ ; (2)  $A^n = \{x^n\}$ ,  $n \in \mathbb{N}$ ;

(3)  $A^+ = \{x^n \mid n \geq 1\}$ ; and (4)  $A^* = \{x^n \mid n \geq 0\}$ .



# Language: The Set Theory of Strings

- Lemma 6.1:

Let  $\Sigma$  be an alphabet, with languages  $A, B \subseteq \Sigma^*$ . If  $A \subseteq B$ , then  $A^n \subseteq B^n$

**Proof :** (i)  $n = 1$ ,  $A^1 = A \subseteq B = B^1$

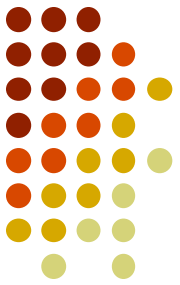
(ii) Assuming the truth for  $n = k$ ,  $A \subseteq B \Rightarrow A^k \subseteq B^k$

(iii) If  $x = x_1x_k \in A^{k+1}$ , i.e.,  $x_1 \in A, x_k \in A^k$ .

$\because A \subseteq B \Rightarrow A^k \subseteq B^k$  (induction hypothesis),  $\therefore x_1 \in B, x_k \in B^k$

$\Rightarrow x = x_1x_k \in BB^k = B^{k+1}$

$\Rightarrow A^{k+1} \subseteq B^{k+1}$



# Language: The Set Theory of Strings

- Theorem 6.2:

For an alphabet  $\Sigma$  and languages  $A, B \subseteq \Sigma^*$ ,

a)  $A \subseteq AB^*$

b)  $A \subseteq B^*A$

c)  $A \subseteq B \Rightarrow A^+ \subseteq B^+$

d)  $A \subseteq B \Rightarrow A^* \subseteq B^*$

e)  $AA^* = A^*A = A^+$

f)  $A^*A^* = A^* = (A^*)^* = (A^*)^+ = (A^+)^*$

g)  $(A \cup B)^* = (A^* \cup B^*)^* = (A^*B^*)^*$

**Proof :**

(g)  $[(A \cup B)^* = (A^* \cup B^*)^*]$

(i)  $A \subseteq A^*, B \subseteq B^* \Rightarrow (A \cup B) \subseteq (A^* \cup B^*)$

$\Rightarrow (A \cup B)^* \subseteq (A^* \cup B^*)^*$  [by (d)]

(ii)  $A, B \subseteq A \cup B \Rightarrow A^*, B^* \subseteq (A \cup B)^*$  [by (d)]

$\Rightarrow (A^* \cup B^*) \subseteq (A \cup B)^*$

$\Rightarrow (A^* \cup B^*)^* \subseteq ((A \cup B)^*)^* = (A \cup B)^*$  [by (d) and (f)]



# Language: The Set Theory of Strings

- Ex 6.14 :

For an alphabet  $\Sigma = \{0,1\}$  consider the languages  $A \subseteq \Sigma^*$ , where each word in  $A$  contains exactly one occurrence of the symbol 0, e.g., 0, 01, 10, 0111, etc. We can define this language  $A$  recursively as follows:

- 1) Our base step tells us that  $0 \in A$
- 2) For the recursive process we want to include in  $A$  the words  $1x$  and  $x1$ , for  $x \in A$ .



# Language: The Set Theory of Strings

- Definition (palindrome):

Given an alphabet  $\Sigma$ , consider  $x = x_1x_2 \cdots x_n$  in  $\Sigma^*$ .

The reversal of  $x$  is denoted  $x^R = x_nx_{n-1} \cdots x_1$ .

We can define the reversal of a string recursively as follows:

1)  $\lambda^R = \lambda$

2) For  $n \in \mathbb{N}$ , if  $x \in \Sigma^{n+1}$ , then we can write  $x = zy$  where  $z \in \Sigma$  and  $y \in \Sigma^n$

--- here, we define  $x^R = (zy)^R = (y^R)z$ .

- **Ex 6.16** : Prove that  $x_1, x_2 \in \Sigma^* \Rightarrow (x_1x_2)^R = x_2^R x_1^R$ .

**Proof** : By mathematical induction

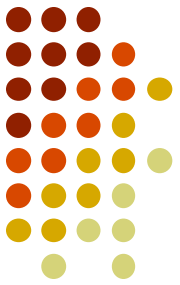
(i)  $\|x_1\| = 0$  → (ii)  $\|x_1\| = k$ ,  $x_1 = \lambda$  and  $(x_1x_2)^R = (\lambda x_2)^R = x_2^R = x_2^R \lambda = x_2^R \lambda^R = x_2^R x_1^R$  because  $\lambda^R = \lambda$

(iii)  $\|x_1\| = k + 1, x_1 = zy_1, \|z\| = 1, \|y_1\| = k$ ,

$$(x_1x_2)^R = (zy_1x_2)^R = (y_1x_2)^R z = x_2^R y_1^R z = x_2^R (zy_1)^R = x_2^R x_1^R.$$

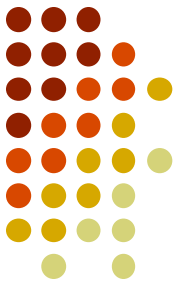


## 6.2 Finite State Machines: A First Encounter



- Example: A vending machine dispenses two flavors of chewing gum: peppermint (P) and spearmint (S).
  - The cost of either flavor is 20c. The machine accepts nickels; dimes, and quarters and returns the necessary change.
  - Mary Jo inserts two nickels and a dime, and press the white button (W) for a package of peppermint-flavored chewing gum.

	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$
<b>State</b>	(1) $s_0$	(4) $s_1$ (5c)	(7) $s_2$ (10c)	(10) $s_3$ (20c)	(13) $s_0$
<b>Input</b>	(2) 5c	(5) 5c	(8) 10c	(11) W	
<b>Output</b>	(3) Nothing	(6) Nothing	(9) Nothing	(12) P	



# Finite State Machines: A First Encounter

- Definition 6.13

A finite state machine is five - tuple  $M = (S, I, O, v, w)$ , where

$S$  = the set of internal states for  $M$ ;

$I$  = the input alphabet for  $M$ ;

$O$  = the output alphabet for  $M$ ;

$v: S \times I \rightarrow S$  is the next state function;

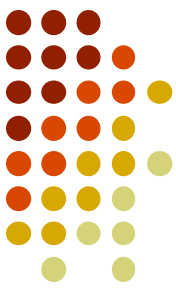
$w: S \times I \rightarrow O$  is the output function;

	$v$		$w$	
	0	1	0	1
$s_0$	$s_0$	$s_1$	0	0
$s_1$	$s_2$	$s_1$	0	0
$s_2$	$s_0$	$s_1$	0	1

- Ex 6.17

Consider the finite state machine  $M = (S, I, O, v, w)$ , where

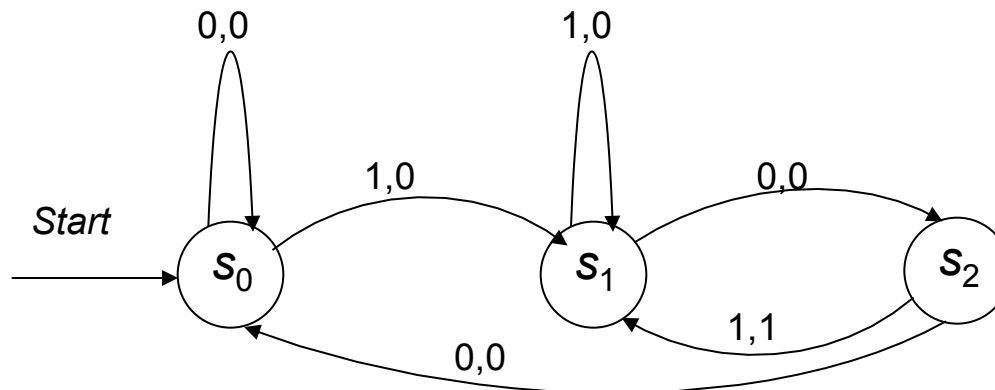
$S = \{s_0, s_1, s_2\}$ ,  $I = O = \{0, 1\}$ , and  $v, w$  are given by the state table.



# Finite State Machines: A First Encounter

- Another representation for finite state machine
  - State diagram
- What is the output string for the input string 1100101101?
  - Input (output) string is an element of  $I^*$  ( $O^*$ ), the Kleene closure of  $I$  ( $O$ ).

State	$s_0$	$v(s_0, 1) = s_1$	$v(s_1, 0) = s_2$	$v(s_2, 1) = s_1$	$v(s_1, 0) = s_2$
Input	1	0	1	0	
Output	$w(s_0, 1) = 0$	$w(s_1, 1) = 0$	$w(s_2, 1) = 1$	$w(s_1, 0) = 0$	

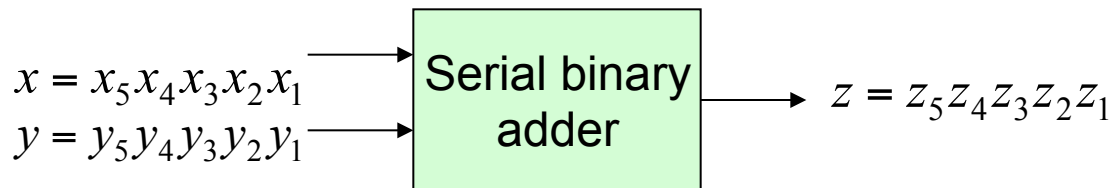


	$v$		$w$	
	0	1	0	1
$s_0$	$s_0$	$s_1$	0	0
$s_1$	$s_2$	$s_1$	0	0
$s_2$	$s_0$	$s_1$	0	1



# Finite State Machines: A First Encounter

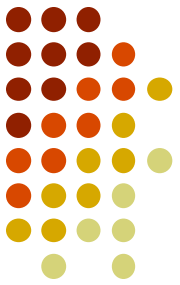
- **Ex 6.19** : A serial binary adder is a finite state machine that we can use to obtain  $x + y$ .
  - E.g.,  $x = x_5x_4x_3x_2x_1 = 00111, y = y_5y_4y_3y_2y_1 = 01101$



$$\begin{array}{r} \phantom{+} x = \phantom{0} \phantom{0} \phantom{1} \phantom{1} \phantom{1} \\ + y = \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \\ \hline z = \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{0} \end{array}$$

+1 +1 +1 +1

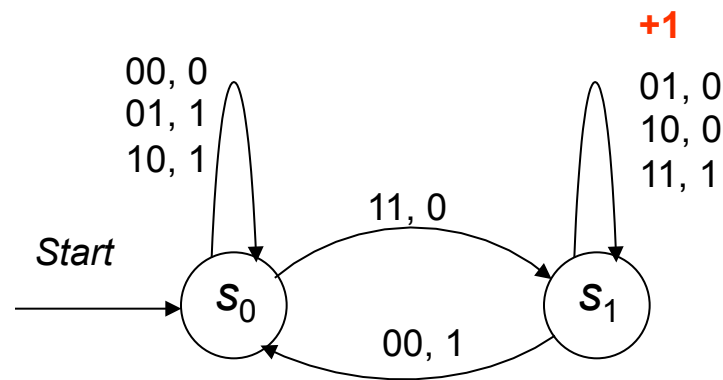
third addition      first addition



# Finite State Machines: A First Encounter

- The serial binary adder is modeled by a finite state machine  $M = (S, I, O, \nu, w)$ .  $S = \{s_0, s_1\}$ , where  $s_i$  indicates a carry of  $i$ ;  $I = \{00, 01, 10, 11\}$ ;  $O = \{0, 1\}$ ; and  $\nu, w$  are given in the following state table.

	$\nu$				$w$			
	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



## 6.3 Finite State Machines: A Second Encounter



- Some additional machines relevant to the design of computer hardware, e.g., sequence recognizer.
- Ex 6.20** : Construct a machine that recognizes each occurrence of the sequence 111.
  - Input: 1110101111, output: 0010000011  $\{0, 1\}^* \{111\}$

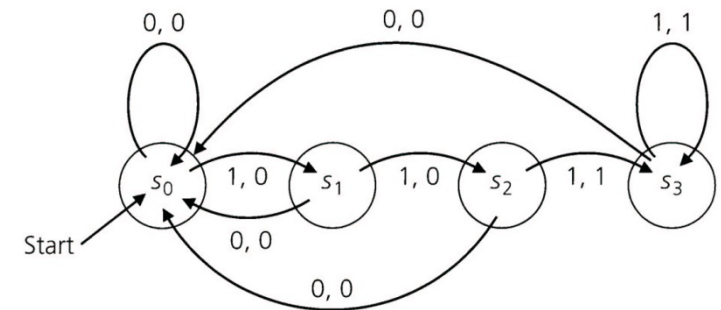
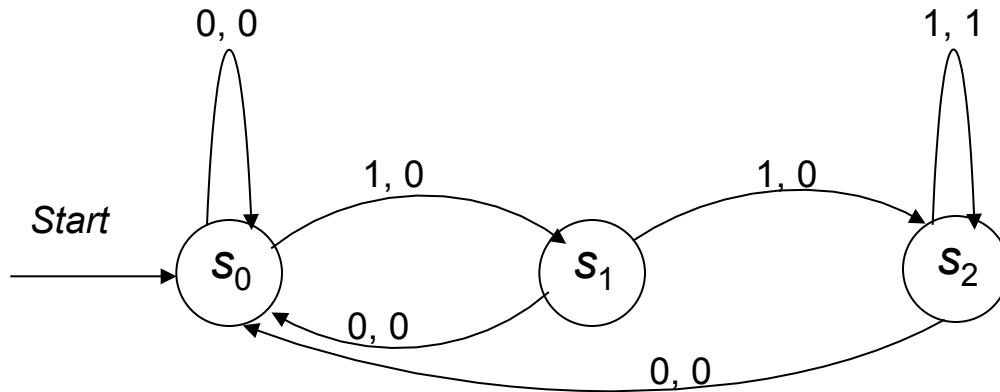


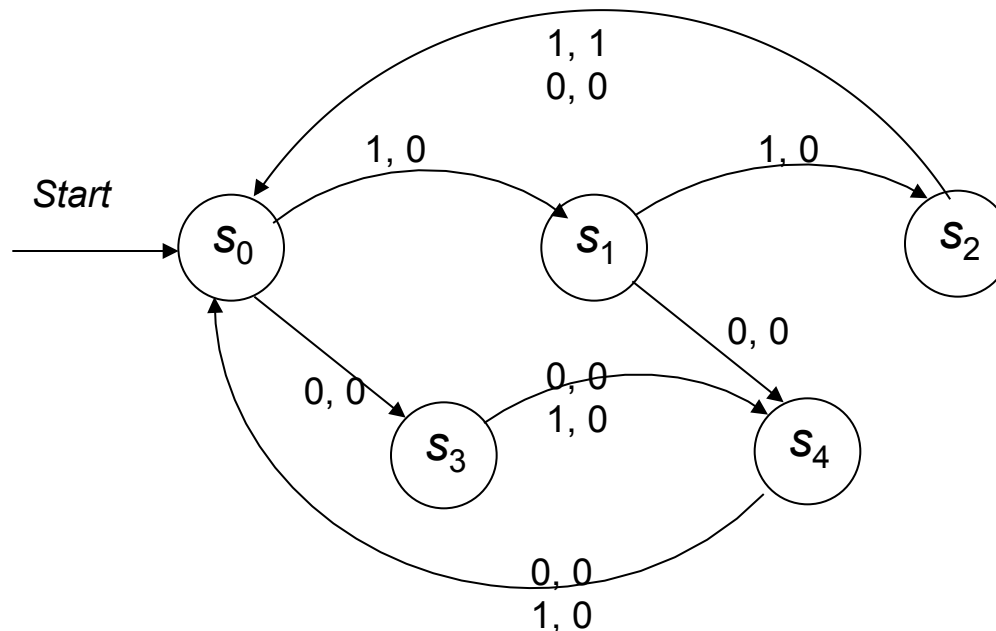
Figure 6.10

*equivalent*

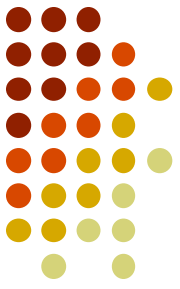
# Finite State Machines: A Second Encounter



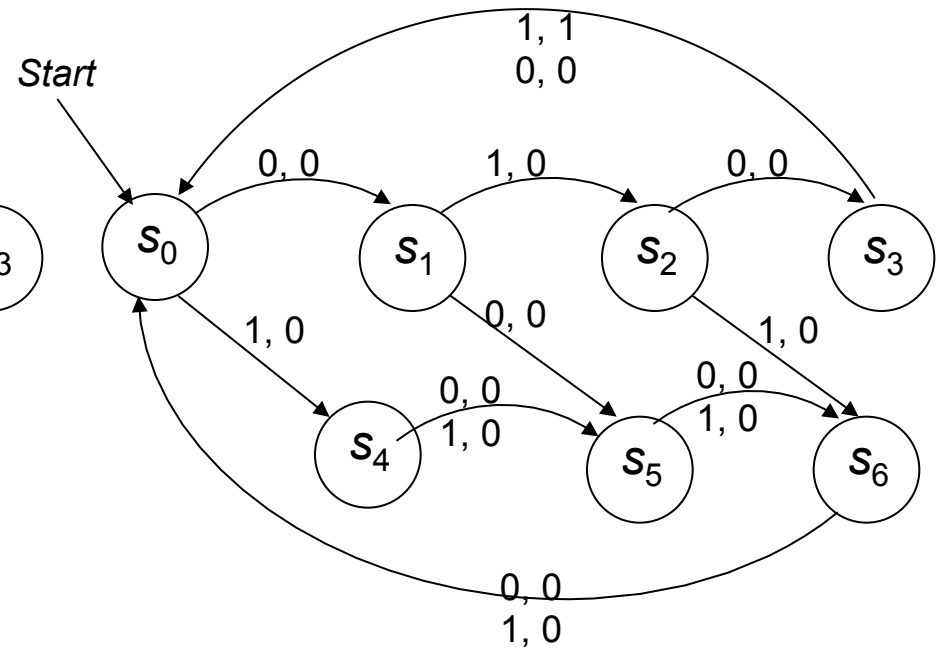
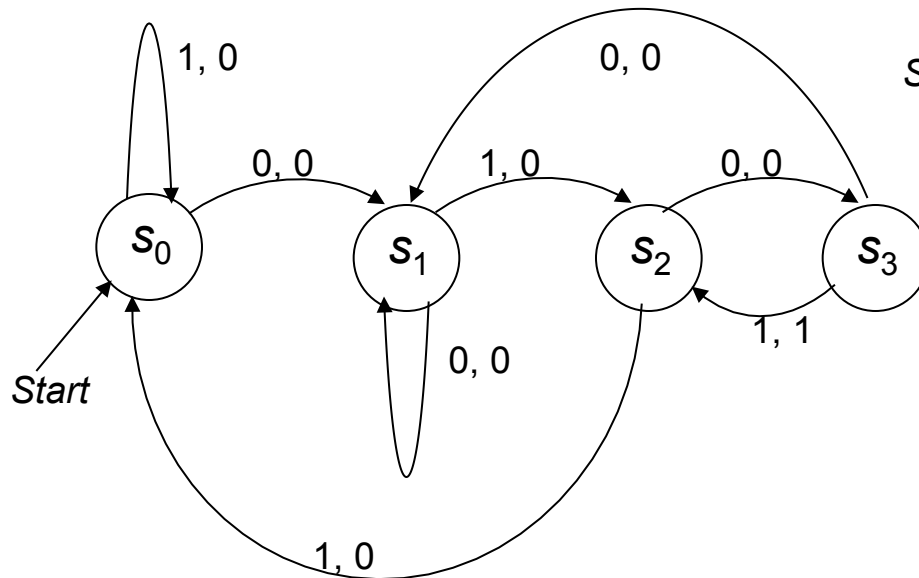
- **Ex 6.21** : Construct a machine that not only recognizes the occurrence of 111, but also recognizes those occurrences that end in a position that is a **multiple of three**.
  - Input: 1110111, output: 0010000, not 0010001
  - Input: 111100111, output: 001000001



# Finite State Machines: A Second Encounter

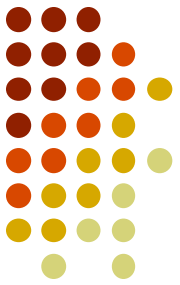


- **Ex 6.22:** Construct a machine that not only recognizes (a) the occurrence of 0101, (b) those occurrences that end in a position that is a multiple of four.
  - Input: 01010100101, output: (a) 00010100001 (b) 00010000000





# Finite State Machines: A Second Encounter



- **Ex 6.23** : Can we construct a finite state machine that recognizes precisely those strings in the language  $A = \{01, 0011, 000111, \dots\} = \{0^i 1^i \mid i \in \mathbb{Z}^+\}$ ?
- Apply the pigeonhole principle

State	$s_0$	$s_1$	$s_2$		$s_i$	$s_{i+1}$		$s_j$	$s_{j+1}$		$s_n$	$s_{n+1}$		$s_{2n}$	$s_{2n+1}$
Input	0	0	0	...	0	0	...	0	0	...	0	1	...	1	1
Output	0	0	0		0	0		0	0		0	1		1	1

State	$s_0$	$s_1$	$s_2$		$s_i$	$s_{j+1}$		$s_n$	$s_{n+1}$		$s_{2n}$	$s_{2n+1}$
Input	0	0	0	...	0	0	...	0	1	...	1	1
Output	0	0	0		0	0		0	1		1	1

# Finite State Machines: A Second Encounter



- **Proof**

$M = \{S, I, O, v, w\}$  can recognize strings in  $A$ , let  $|S| = n \geq 1$ .

Consider  $0^{n+1}1^{n+1}$  in the language  $A$

$\Rightarrow M$  will process  $n+1$  0's by  $n+1$  states  $s_0, s_1, \dots, s_n$

$\therefore |S| = n$ , by pigeonhole principle, exists two states  $s_i = s_j$

$\therefore$  We can remove  $j-i$  columns (see the tables)

$\Rightarrow M$  recognizes  $x = 0^{(n+1)-(j-i)}1^{n+1}$ , where  $(n+1) - (j-i) < n+1$

In fact,  $x \notin A \Rightarrow M$  cannot recognize  $x$

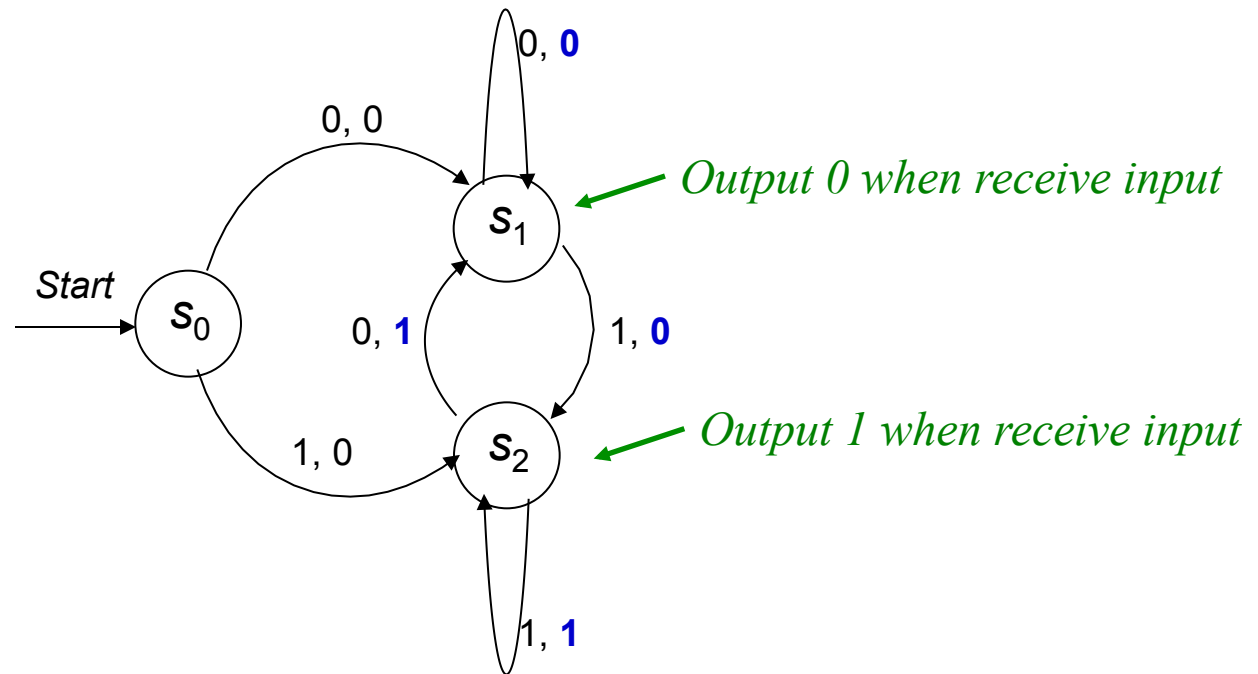
$\therefore$  contradict

$\therefore$  We cannot construct a finite state machine recognizing strings in  $A$ .

# Finite State Machines: A Second Encounter



- One-unit delay machine
  - **Ex 6.24**: If  $x = x_1x_2\dots x_m$ , then  $w(s_0, x) = 0x_1x_2\dots x_{m-1}$ .

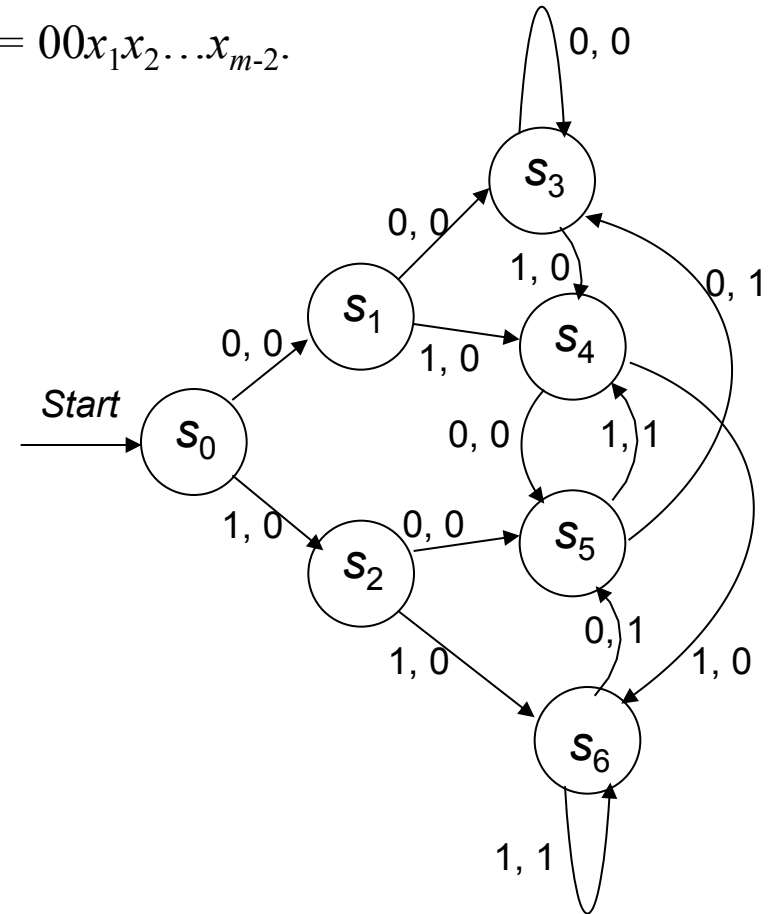
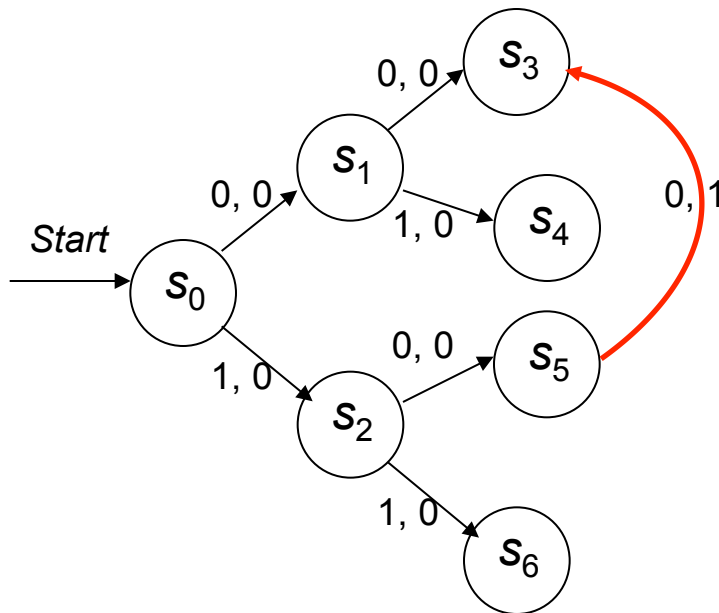


# Finite State Machines: A Second Encounter



- Two-unit delay machine

- Ex 6.25**: If  $x = x_1x_2\dots x_m$ , then  $w(s_0, x) = 00x_1x_2\dots x_{m-2}$ .



# Finite State Machines: A Second Encounter

- Definition 6.14

$M = \{S, I, O, v, w\}$  is a finite state machine

a)  $s_i, s_j \in S, s_j$  is reachable from  $s_i$

if  $s_i = s_j$  or  $v(s_i, x) = s_j$  (e.g.,  $s_3$  is reachable from  $s_0, s_1, s_2$  in the Fig.)

b)  $s$  is transient if  $v(s, x) = s$  for  $x \in I^*$  implies  $x = \lambda$  (e.g.,  $s_2$  in the Fig.) (出去回不來)

c)  $s$  is a sink if  $v(s, x) = s$  for all  $x \in I^*$  (e.g.,  $s_3$  in the Fig.)

d)  $S_1 \subseteq S, I_1 \subseteq I$ , if  $v_1 = v|_{S_1 \times I_1} : S_1 \times I_1 \rightarrow S$ ,

$w_1 = w|_{S_1 \times I_1}, M_1 = \{S_1, I_1, O, v_1, w_1\}$  is a submachine of  $M$

e) Strongly connected if for any  $s_i, s_j \in S, s_j$  is reachable from  $s_i$

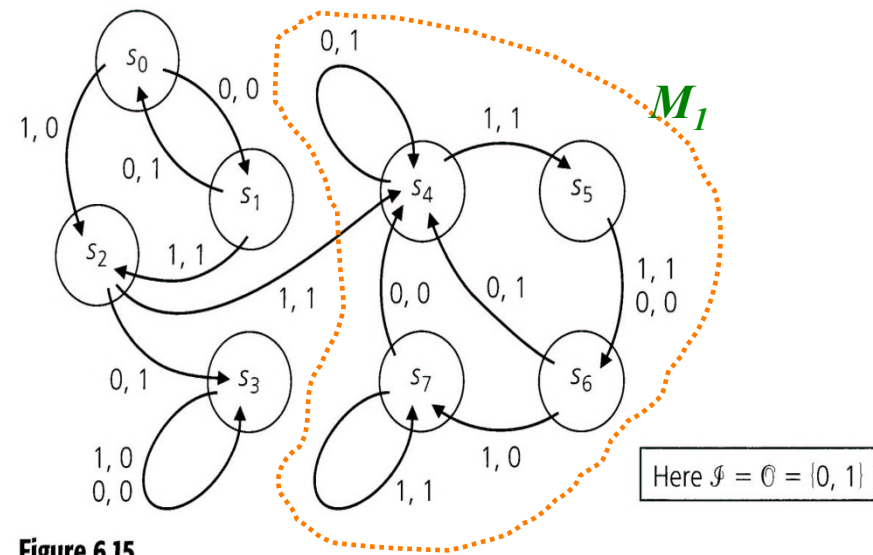
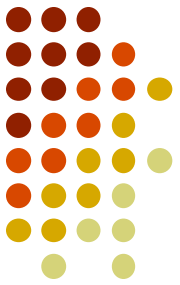


Figure 6.15



# Finite State Machines: A Second Encounter

- Definition 6.15

For a finite state machine  $M$ , let  $s_i, s_j$  be two distinct states.

$x \in I^+$  is called a transfer (transition) sequence from  $s_i$  to  $s_j$  if

a)  $v(s_i, x) = s_j$

b)  $y \in I^+$  with  $v(s_i, y) = s_j \Rightarrow \|y\| \geq \|x\|$

- Ex 6.26** : Find a transfer sequence from  $s_0$  to  $s_2$  in the state table.

- $x = 0000$

	$v$		$w$	
	0	1	0	1
$s_0$	$s_6$	$s_1$	0	1
$s_1$	$s_5$	$s_0$	0	1
$s_2$	$s_1$	$s_2$	0	1
$s_3$	$s_4$	$s_0$	0	1
$s_4$	$s_2$	$s_1$	0	1
$s_5$	$s_3$	$s_5$	1	1
$s_6$	$s_3$	$s_6$	1	1

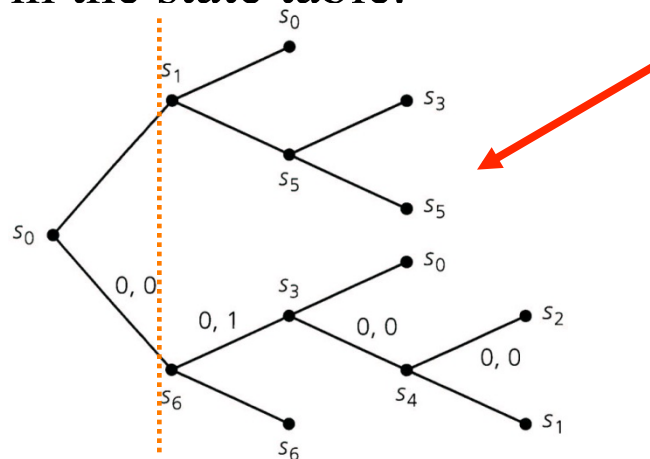
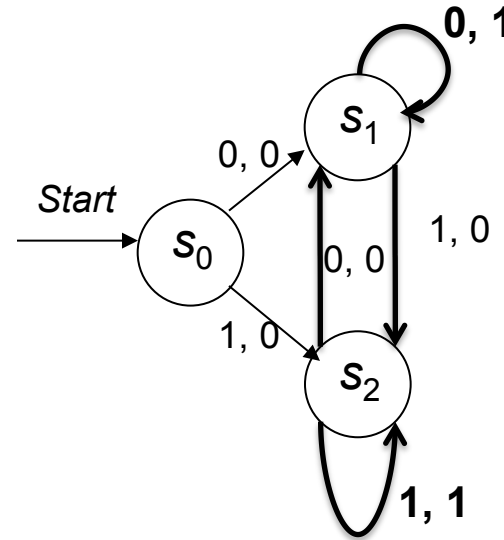
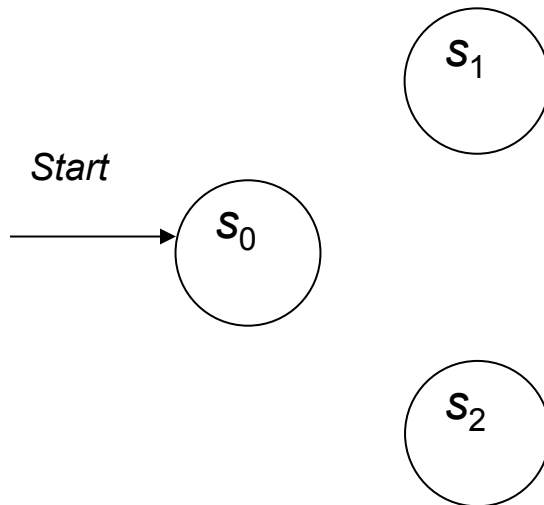


Figure 6.16



## Practice (6.3-3)

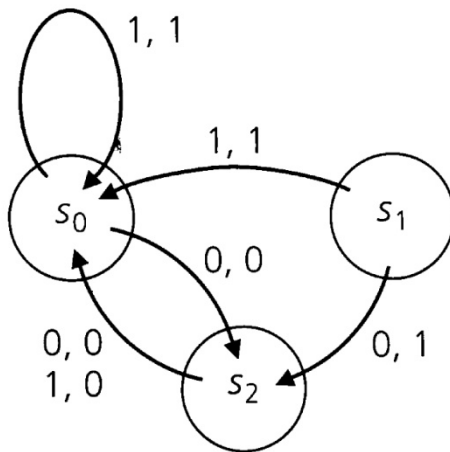
- Construct a state diagram for a finite state machine with  $I=O=\{0, 1\}$  that recognizes all strings in the language  $\{0, 1\}^*\{00\} \cup \{0, 1\}^*\{11\}$





# Practice (Sppl.-5)

- Let  $M$  be the finite state machine in the following figure. For states  $S_i, S_j$ , where , let  $O_{ij}$  denote the set of all nonempty output strings that  $M$  can produce as it goes from state  $S_i$ , to state  $S_j$ , e.g.,  $O_{20} = \{0\} \{1, 00\}^*$ . Find  $O_{02}, O_{22}, O_{11}$ , and  $O_{10}$ .



$$O_{00} = \{1, 00\}^* - \{\lambda\}$$

$$O_{02} = \{1, 00\}^* \{0\}$$

$$O_{22} = \{0\} \{1, 00\}^* \{0\}$$

$$O_{11} = \Phi$$

$$O_{10} = \{1\} \{1, 00\}^* \cup \{10\} \{1, 00\}^*$$