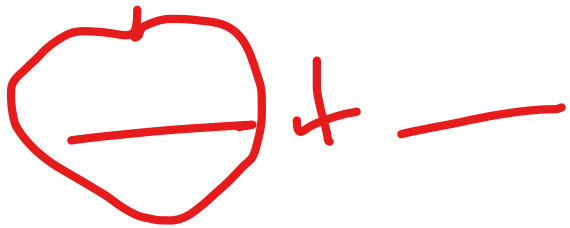
INTRODUCTION TO LANGUAGES AND FINITE AUTOMATA

Maria Thomas

October 6, 2023



Given a task, two questions arise:

1. Can it be carried out using a computer?

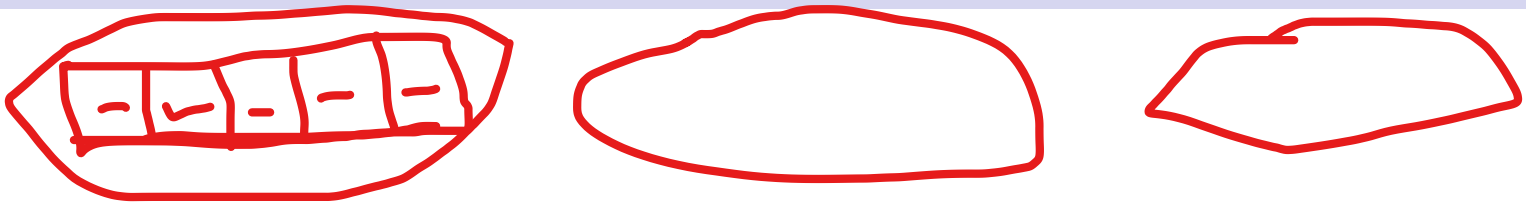
2. How can the task be carried out?

Models of computation are used to help answer these questions.

Types of structures used in models of computation:

- Grammars

Types of structures used in models of computation:



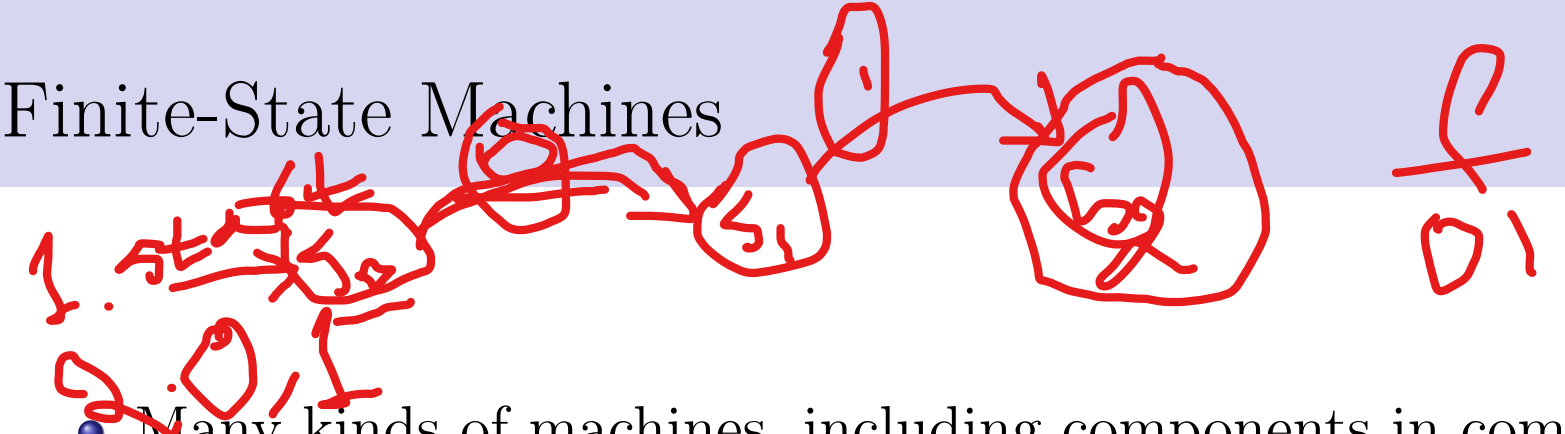
- Grammars
  - Grammars are used to generate the words of a language and to determine whether a word is in a language.
  - Formal languages, which are generated by grammars, provide models for both natural languages, such as English, and for programming languages, such as Pascal, Fortran, Prolog, C, and Java.
  - American linguist Noam Chomsky in the 1950s.

- Finite-state machines

- Finite-state machines
  - Various types of finite-state machines are used in modeling.

- Turing machines

Finite-State Machines



- Many kinds of machines, including components in computers, can be modeled using a structure called a finite-state machine.
- All versions of finite-state machines include a finite set of states
  1. a designated starting state, an input alphabet,
  2. a transition function that assigns a next state to every state
  3. input pair
- Applications in computer science and data networking.
- Spell checking, grammar checking, indexing or searching large bodies of text etc.

Finite-State Machines with Output

- Finite-state machines that produce output.
- We will see how finite-state machines can be used to model a vending machine, a machine that delays input, a machine that adds integers, and a machine that determines whether a bit string contains a specified pattern.

Definition 1

- A finite-state machine $M = (S, I, O, f, g, s_0)$ consists of a finite set S of states, a finite input alphabet I, a finite output alphabet O, a transition function f that assigns to each state and input pair a new state, an output function g that assigns to each state and input pair an output, and an initial state s_0.

Representation

FSM - 6

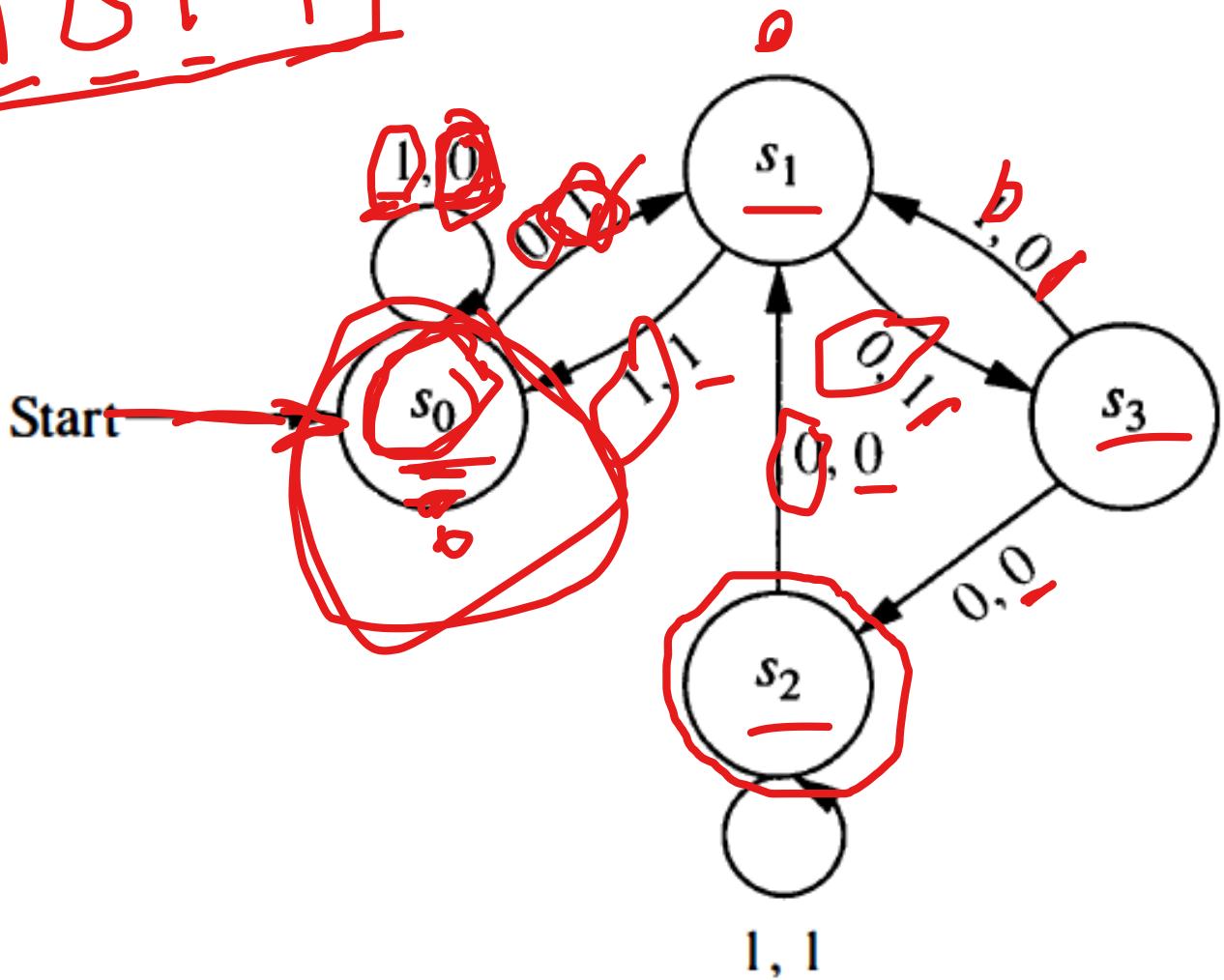
- Let $M = (S, I, O, f, g, s_0)$ be a finite-state machine. We can use a **state table** to represent the values of the transition function f and the output function g for all pairs of states and input.
- Another way to represent a finite-state machine is to use a **state diagram**, which is a directed graph with labeled edges. In this diagram, each state is represented by a circle. Arrows labeled with the input and output pair are shown for each transition.

Example 1

- The state table shown in Table 2 describes a finite-state machine with $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. The values of the transition function f are displayed in the first two columns, and the values of the output function g are displayed in the last two columns.

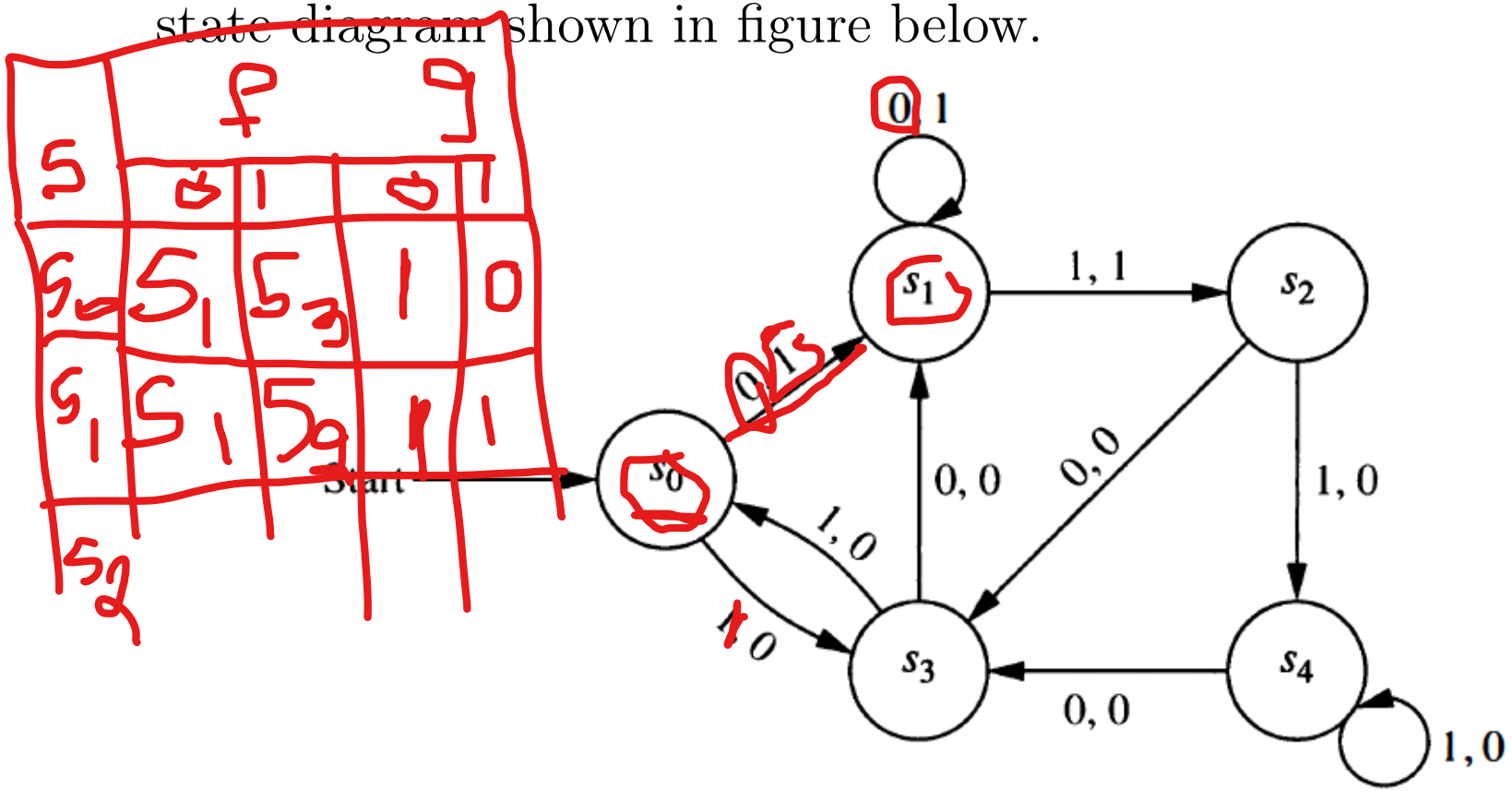
TABLE 2

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


Start
s_0
s_1
s_2
s_3
1, 0
1, 1
0, 1
0, 0
0, 0
1, 1

Example 2

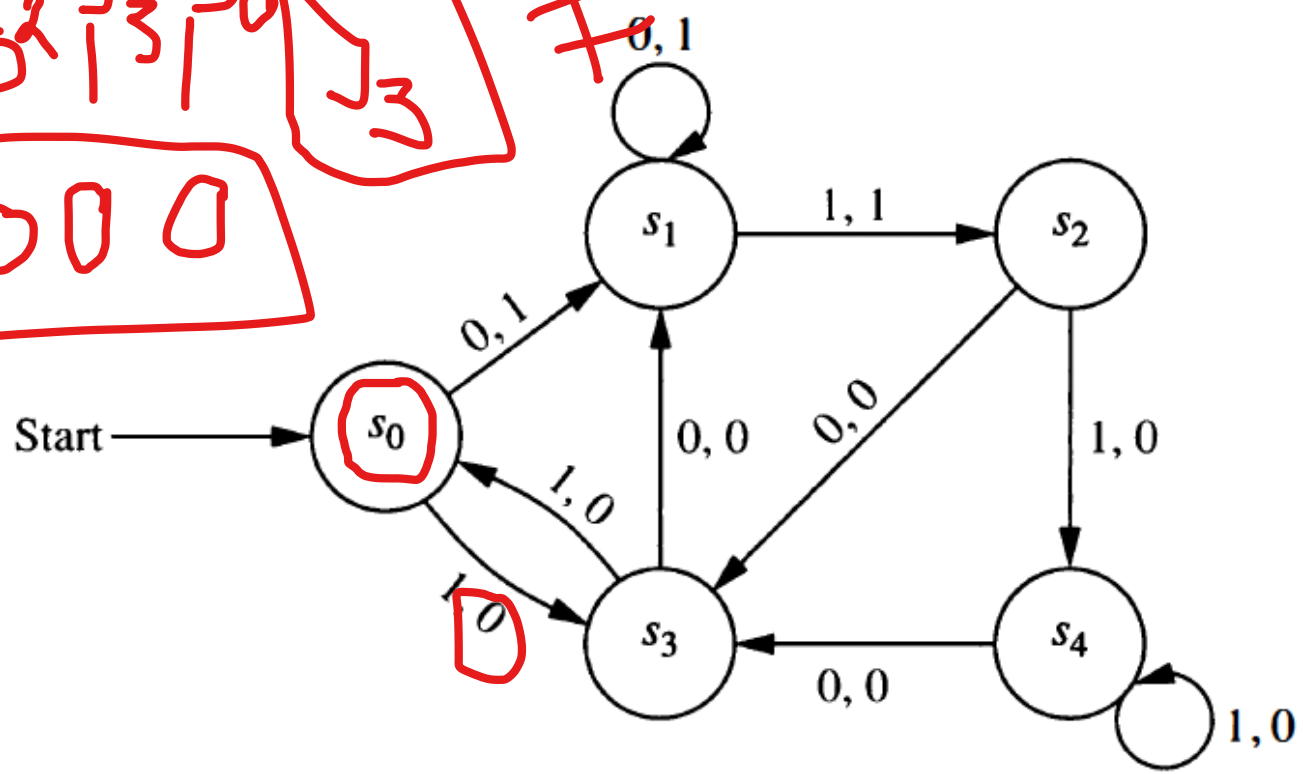
- Construct the state table for the finite-state machine with the state diagram shown in figure below.



State	$\cdot f$		g	
	Input		Input	
	0	1	0	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

Example 3

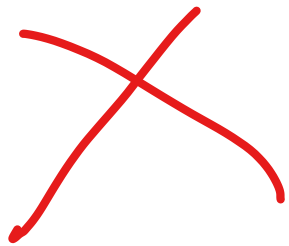
- Find the output string generated by the finite-state machine in if the input string is 101011.



Input	1	0	1	0	1	1	—
State	s_0	s_3	s_1	s_2	s_3	s_0	s_3
Output	0	0	1	0	0	0	—

Definition 2

- Let $M = (S, I, O, f, g, s_0)$ be a finite-state machine and $L \subseteq I^*$. We say that M recognizes(or accepts) L if an input string x belongs to L if and only if the last output bit produced by M when given x as input is a 1.



Example 4

- In a certain coding scheme, when three consecutive 1 s appear in a message, the receiver of the message knows that there has been a transmission error. Construct a finite-state machine that gives a 1 as its current output bit if and only if the last three bits received are all 1s.

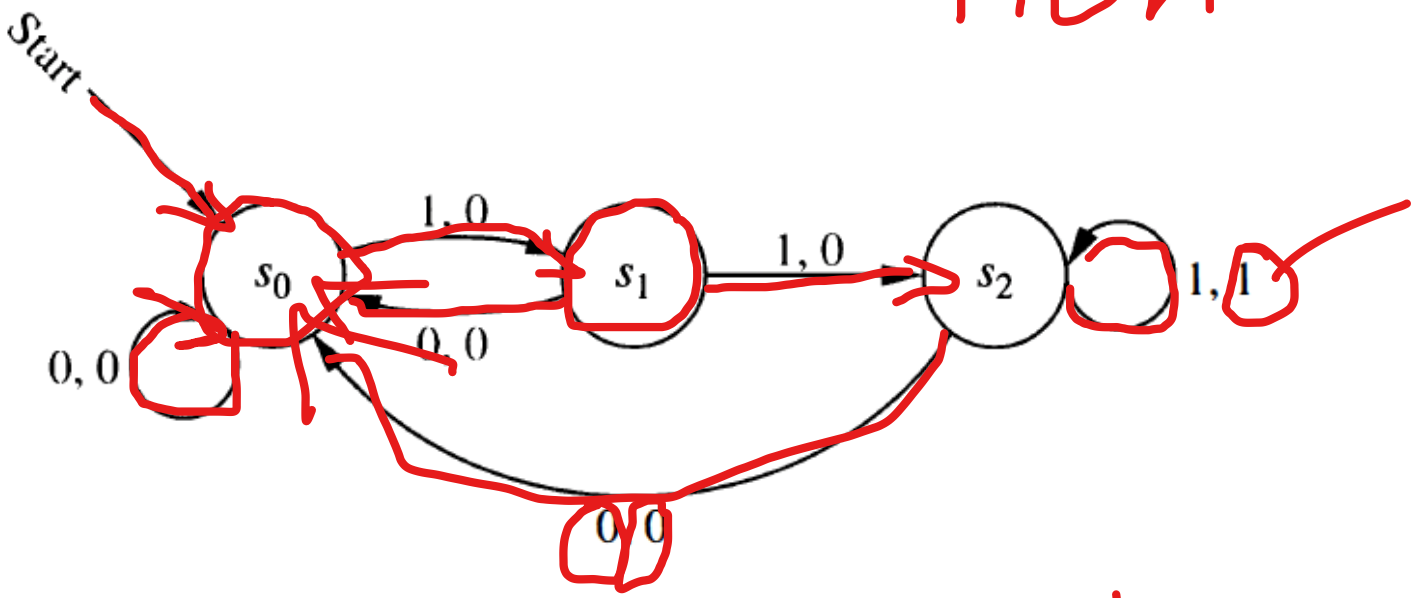
Example 4

- Three states are needed in this machine:
- The start state s_0 remembers that the previous input value, if it exists, was not a 1.
- The state s_1 remembers that the previous input was a 1, but the input before the previous input, if it exists, was not a 1.
- The state s_2 remembers that the previous two inputs were 1s.

Example 4 Ctd.

Input 1: 1011

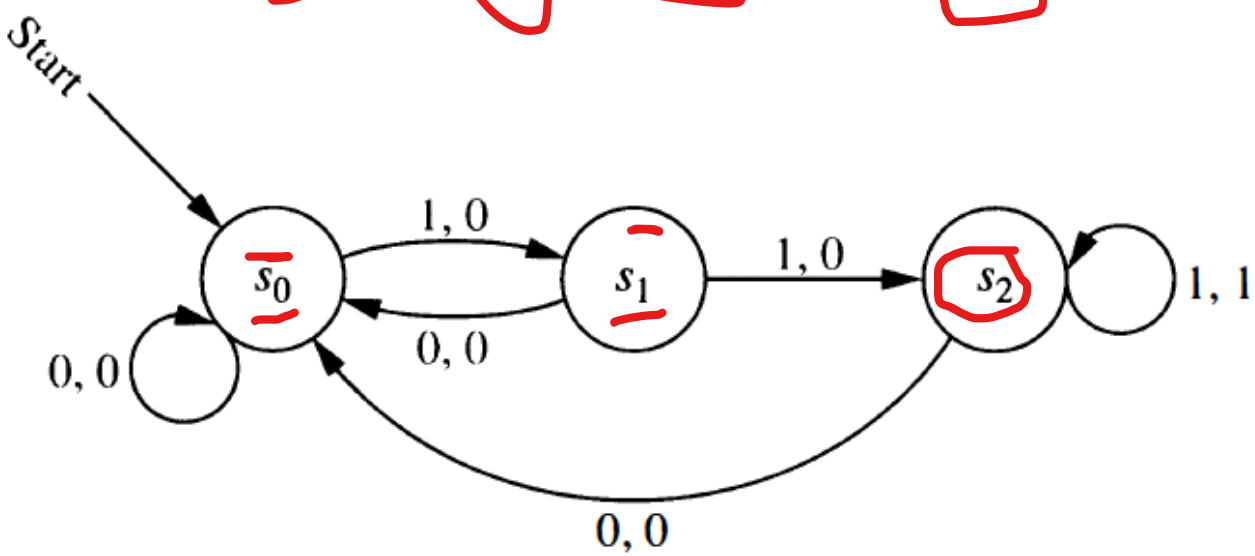
Input 2: 0100111



Example 4 Ctd.

Input 1: 1011

Input 2: 0100111



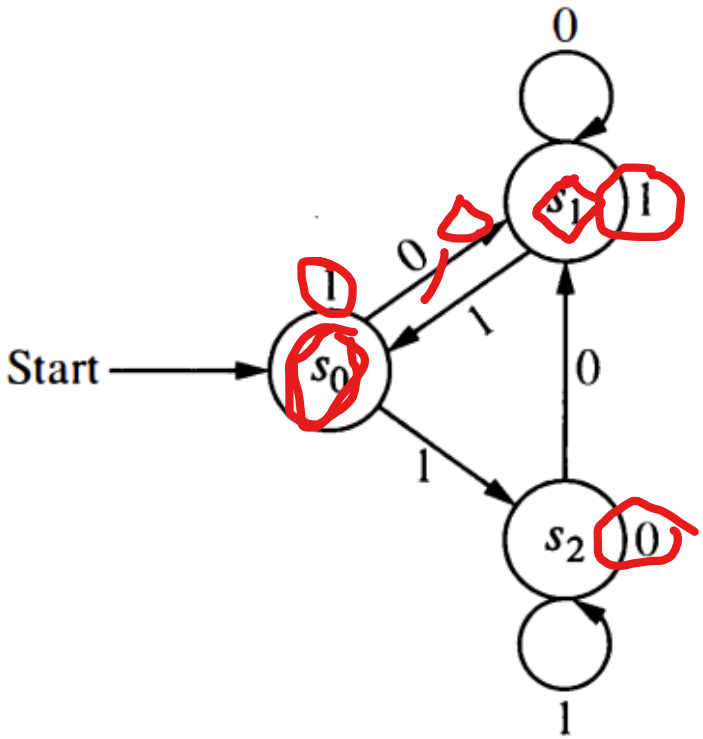
The final output bit of the finite-state machine we constructed here is 1 if and only if the input string ends with 111 .

Types of Finite-State Machines

- Mealy machines: outputs correspond to transitions between states.
- Moore machine: output is determined only by the state.

Example of Moore Machine

Construct the state table for the Moore machine with the state diagram shown here.



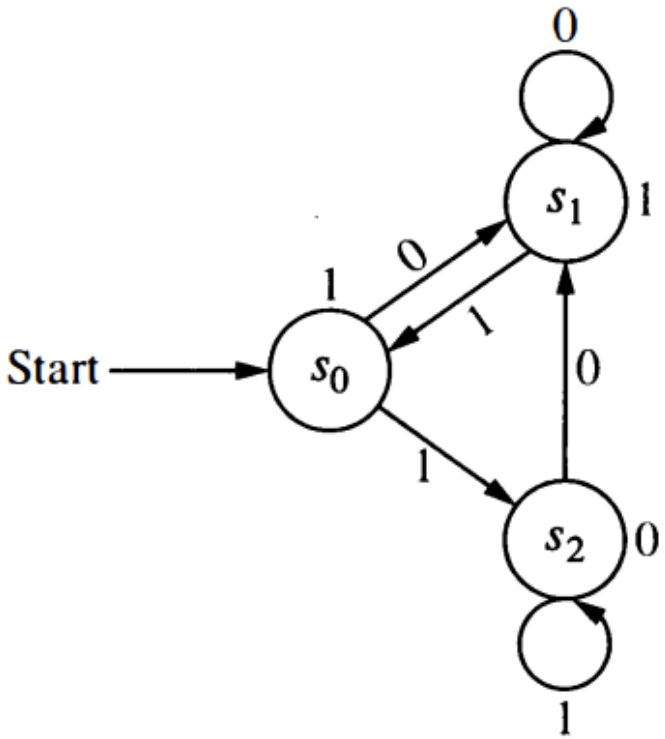
Example of Moore Machine

	f		
	Input		
State	0	1	g
s_0	s_1	s_2	1
s_1	s_1	s_0	1
s_2	s_1	s_2	0

Example of Moore Machine

Inputs: a) 0101

b) 111111



Strings

- Sequences of the form $a_1, a_2, ..., a_n$ or $a_1 a_2 ... a_n$ are called strings.
- The length of the string S is the number of terms in this string.
- The **empty string**, denoted by λ, is the string that has no terms. The empty string has length zero.

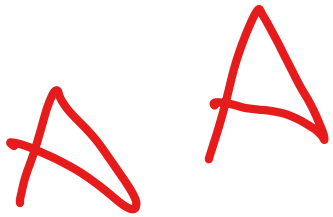
Vocabulary

- A vocabulary (or alphabet) V is a finite, nonempty set of elements called symbols.
- A word (or sentence) over V is a string of finite length of elements of V.
- The empty string or null string, denoted by λ, is the string containing no symbols.
- The set of all words over V is denoted by V^*. A language over V is a subset of V^*.

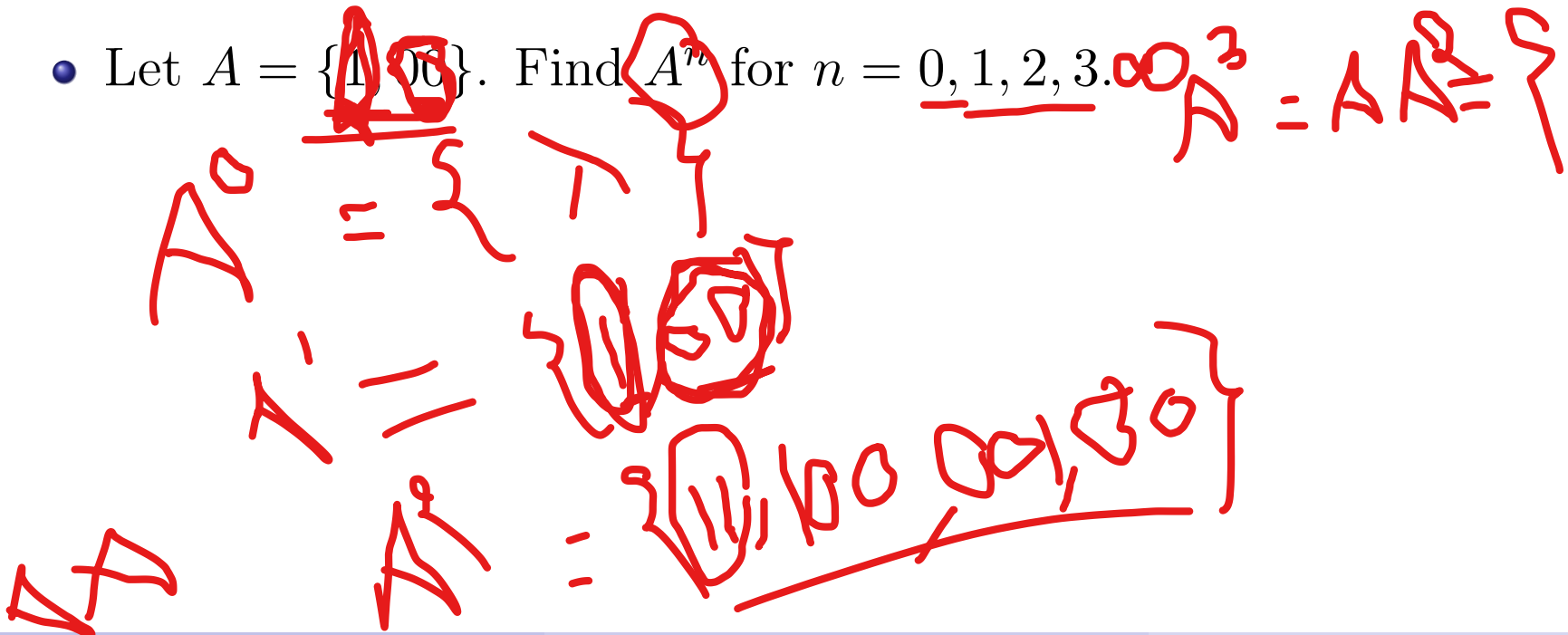
Definition: Concatenation

- Suppose that A and B are subsets of V^*, where V is a vocabulary. The concatenation of A and B, denoted by AB, is the set of all strings of the form xy, where x is a string in A and y is a string in B.

Example



- Let $A = \{0, 11\}$ and $B = \{1, 10, 110\}$. Find AB and BA.
- Let $A = \{1, 00\}$. Find A^n for $n = 0, 1, 2, 3$.



Definition: Kleene closure of A

- Suppose that A is a subset of V^*. Then the Kleene closure of A, denoted by A^*, is the set consisting of concatenations of arbitrarily many strings from A. That is, $A^* = \bigcup_{k=0}^{\infty} A^k$.

- Example: What are the Kleene closures of the sets $A = \{0\}$, $B = \{0, 1\}$, and $C = \{11\}$?

Solution

- The Kleene closure of A is the concatenation of the string 0 with itself an arbitrary finite number of times. Hence,
  $A^* = \{0^n : n = 0, 1, 2, 3...\}$

- The Kleene closure of B is the concatenation of an arbitrary number of strings, where each string is either 0 or 1. This is the set of all strings over the alphabet $V = \{0, 1\}$.

- The Kleene closure of C is the concatenation of the string 11 with itself an arbitrary number of times. Hence, C^* is the set of strings consisting of an even number of 1s.
  That is, $C^* = \{1^{2n} | n = 0, 1, 2, ...\}$.

Finite-State Automata

- A finite-state machine with no output is called finite-state automaton.
- A finite-state automaton $M = (S, I, f, s_0, F)$ consists of a finite set S of states, a finite input alphabet I, a transition function f that assigns a next state to every pair of state and input (so that $f : S \times I \to S$), an initial or start state s_0, and a subset F of S consisting of final (or accepting states).
- We can represent finite-state automata using either state tables or state diagrams. Final states are indicated in state diagrams by using **double circles**.

Exercise

- Construct the state diagram for the finite-state automaton $M = (S, I, f, s_0, F)$, where $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$, and the transition function f is given in Table 1.

TABLE 1

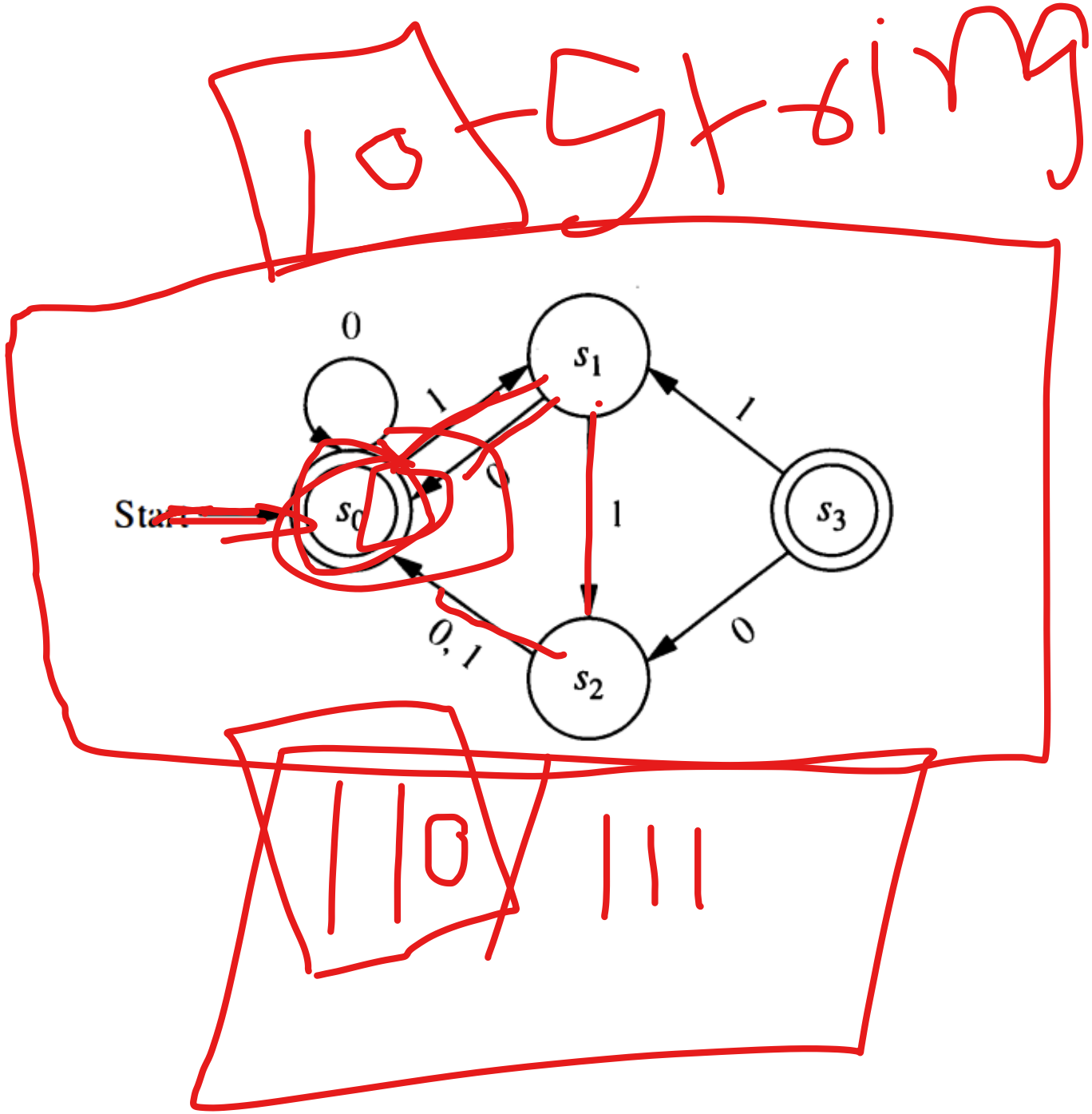
	f	
	Input	
State	**0**	**1**
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1

Exercise

- Construct the state diagram for the finite-state automaton $M = (S, I, f, s_0, F)$, where $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$, and the transition function f is given in Table 1.

TABLE 1

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


0
1
1
0
Start
s_0
s_1
s_2
s_3
1
0, 1
0

EXTENDING THE TRANSITION FUNCTION

- The transition function I of a finite-state machine $M = (S, I, f, s_0, F)$ can be extended so that it is defined for all pairs of states and strings.
  i.e. f can be extended to a function $f : S \times I^* \to S$.

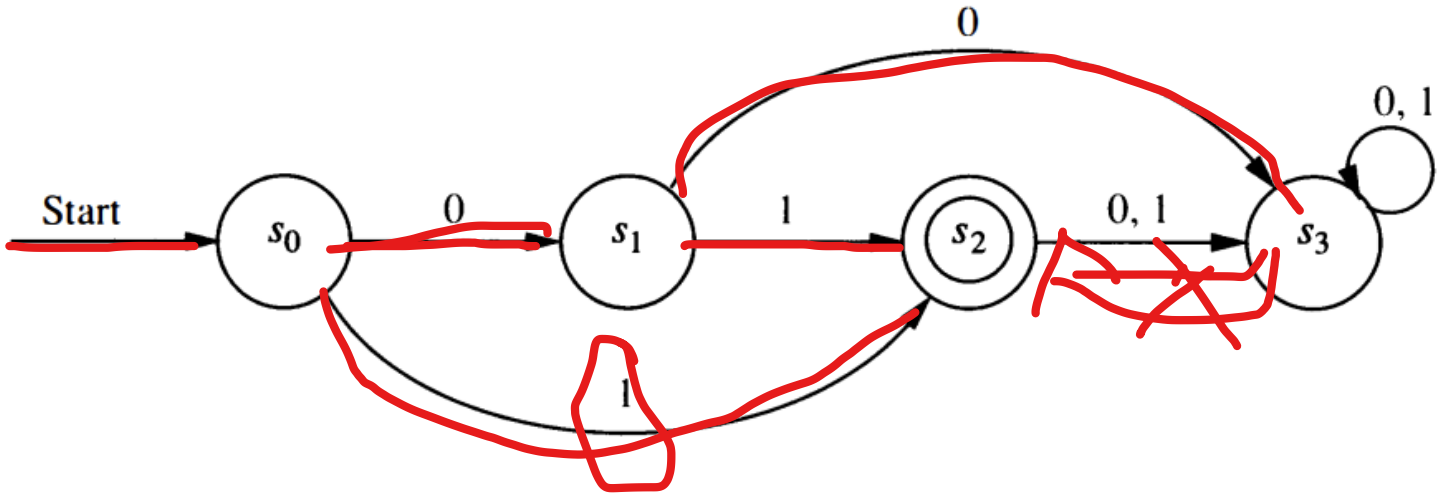
- Let $x = x_1x_2...x_k$ be a string in I^*. Then $f(s_1, x)$ is the state obtained by using each successive symbol of x , from left to right, as input, starting with state s_1. From s_1 we go on to state $s_2 = f(s_1, x_1)$, then to state $s_3 = f(s_2, x_2)$, and so on, with $f(s_1, x) = f(s_k, x_k)$.

Language Recognition by Finite-State Machines

- A string x is said to be recognized or accepted by the machine $M = (S, I, f, s_0, F)$ if it takes the initial state s_0 to a final state. i.e. $f(s_0, x)$ is a state in F.

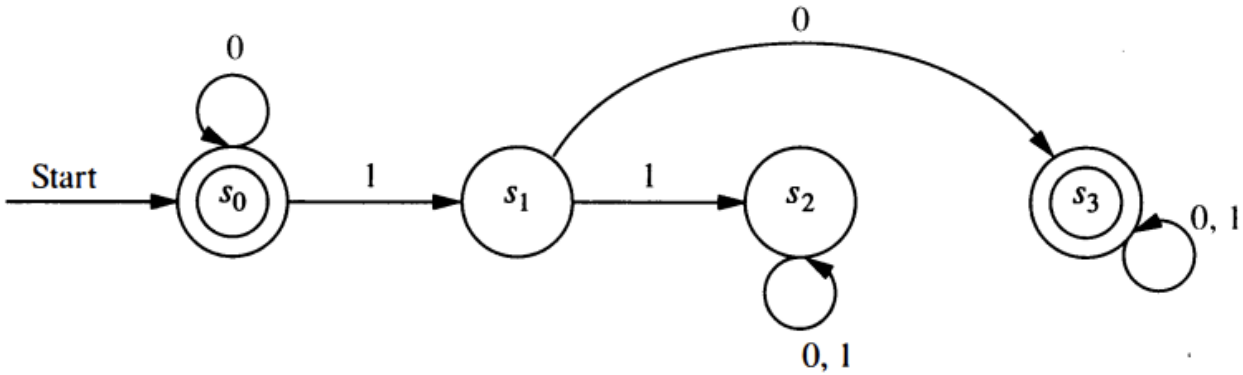
- Two finite-state automata are called equivalent if they recognize the same language.

Determine the languages recognized by the finite-state automaton



Ans: $L(M) = \{0, 01\}$

Determine the languages recognized by the finite-state automaton

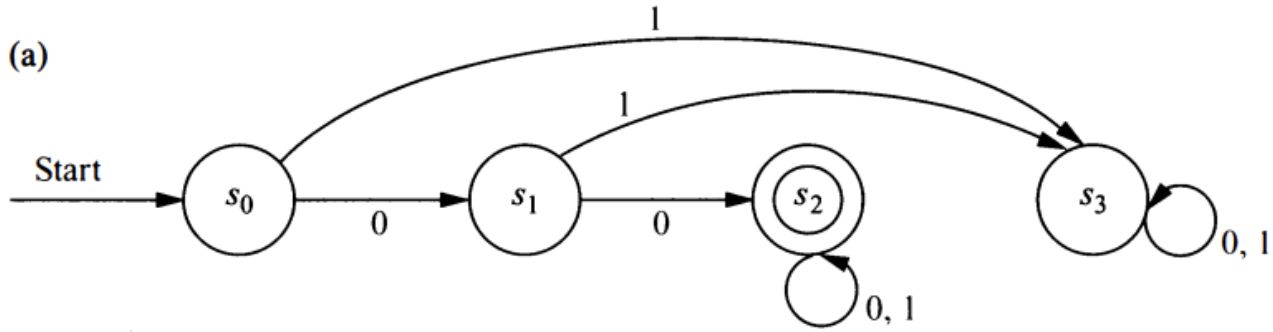


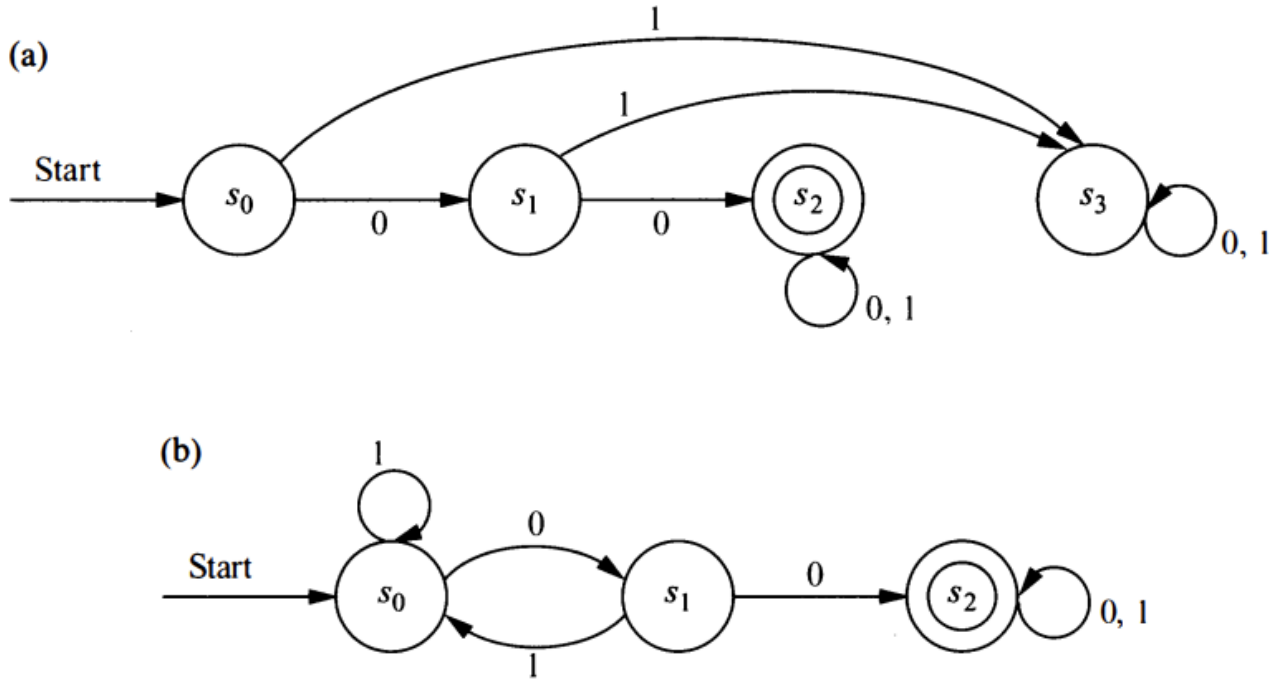
Ans: $L(M) = \{0^n, 0^n 10x : n = 0, 1, 2, ...and$x is any string$\}$

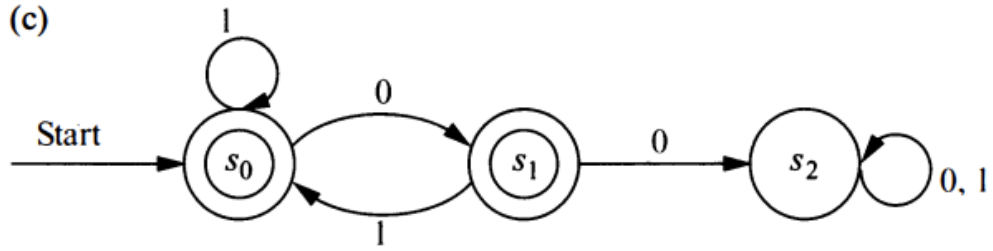
Deterministic Finite-State Automaton

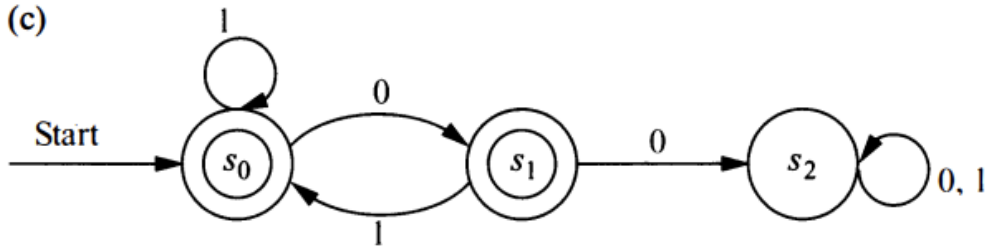
Deterministic finite-state automaton is a finite-state machine in which for each pair of state and input value there is a unique next state given by the transition function.

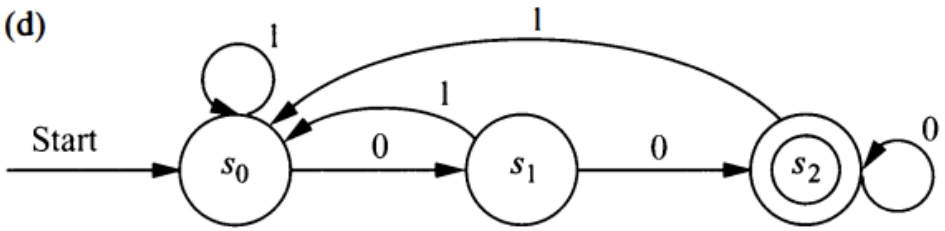
- Construct deterministic finite-state automata that recognize each of these languages.
  a. the set of bit strings that begin with two 0s
  b. the set of bit strings that contain two consecutive 0s
  c. the set of bit strings that do not contain two consecutive 0s
  d. the set of bit strings that end with two 0s
  e. the set of bit strings that contain at least two 0s

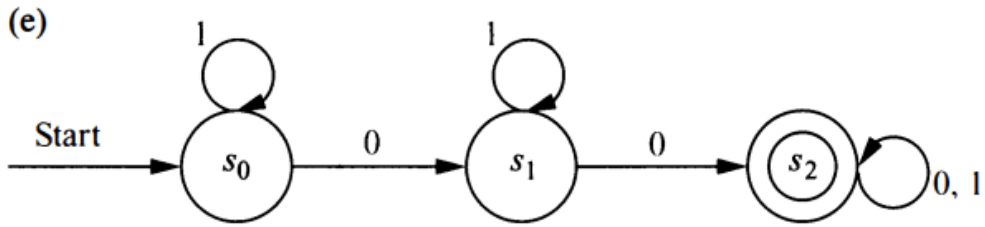

(a)
Start
s_0
s_1
s_2
s_3
0
0
1
1
0, 1
0, 1


(a)
Start
s_0
s_1
s_2
s_3
1
1
0
0
0, 1
0, 1
(b)
Start
s_0
s_1
s_2
1
0
1
0
0, 1

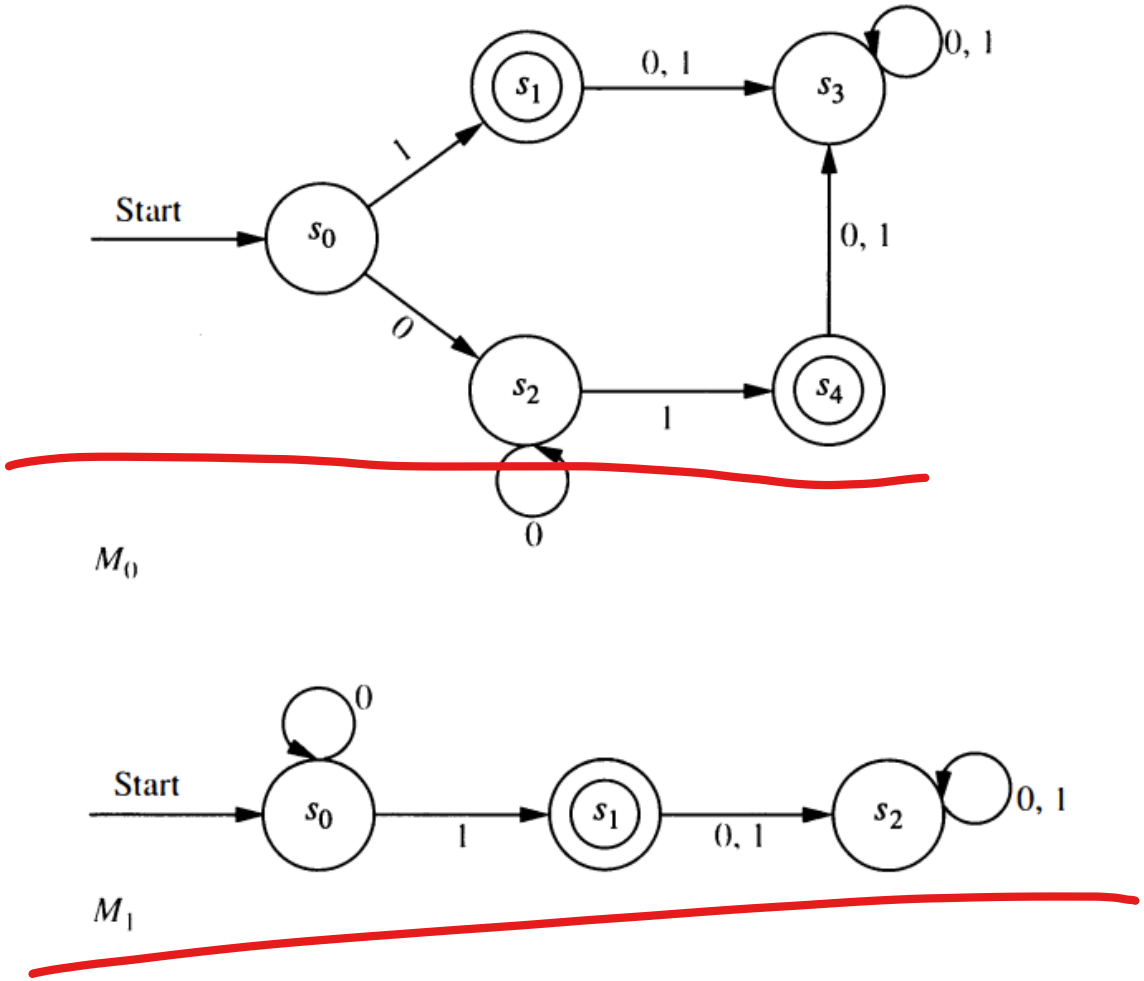

(c)
1
0
Start
s_0
s_1
0
s_2
0, 1
1


(c)
1
0
Start
s_0
s_1
0
s_2
0, 1
1


(d)
1
1
1
Start
s_0
0
s_1
0
s_2
0


(e)
1
1
Start
s_0
0
s_1
0
s_2
0, 1

- Show that the two finite-state automata M_0 and M_1 in the figures are equivalent.



Solution: Let x be any string.

For M_0 to recognise x, x must reach one of the final stages, s_1 or s_4, from the starting stage s_0. Now, the only string that reaches s_1 from s_0 is 1.

The strings that will reach s_4 from s_0 should reach s_2 and then from s_2 to s_4.

The strings from s_0 to reach s_4 should start with 0, followed by a 1 or start with 0 followed by more additional 0s, which keep the machine in state s_2, followed by a 1 to reach the final stage, s_4.

All other strings takes the machine from s_0 to a state that is not final.

$\therefore$

$$L(M_0) = \{1, 0^n1 : n = 1, 2, ...\} \tag{1}$$

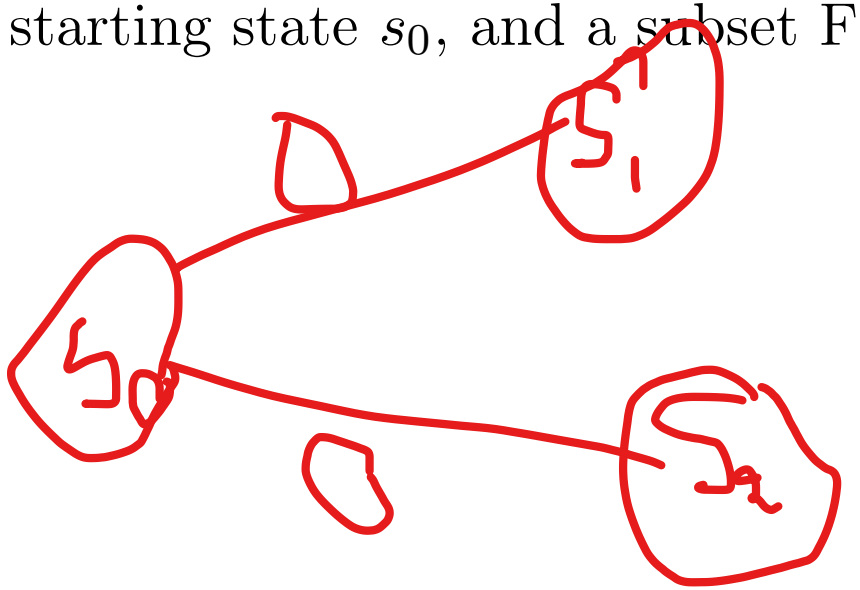
Solution ctd: For a string x to be recognized by M_1, x must take the machine from s_0 to the final state, i.e. s_1.
If x is just 1, it reaches s_1 from s_0.
If x starts with a number of 0s, which leave the machine in state s_0, followed by a 1, which takes it to the final state s_1. A string of all 0s is not recognized because it leaves the machine in state s_0, which is not final. All strings that contain a start with 1 followed by a 0 are not recognized because it will reach s_2, which is not final. $\therefore$

$$L(M_1) = \{1, 0^n1 : n = 1, 2, ...\} \tag{2}$$

From (1) and (2) we can conclude that the finite state machines M_0 and M_1 recognise the same language. Hence by the definition of equivalent machines, M_0 and M_1 are equivalent.

Nondeterministic Finite-State Automaton

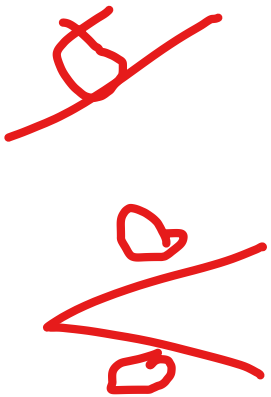
A nondeterministic finite-state automaton $M = (S, I, f, s_0, F)$ consists of a set S of states, an input alphabet I, a transition function f that assigns a set of states to each pair of state and input (so that $f : S \times I \to P(S)$), a starting state s_0, and a subset F of S consisting of the final states.



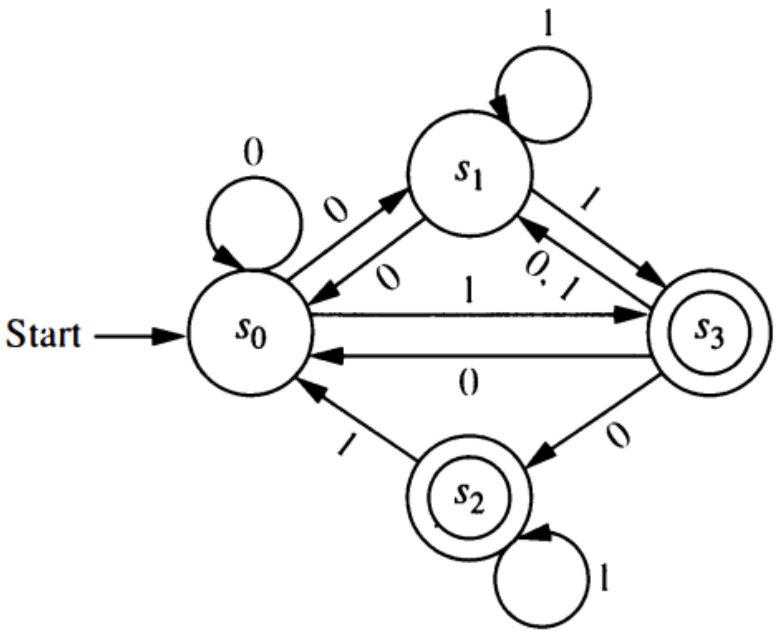
Example

Find the state diagram for the nondeterministic finite-state automaton with the state table shown below.

	f	
	Input	
State	**0**	**1**
s_0	s_0, s_1	s_3
s_1	s_0	s_1, s_3
s_2		s_0, s_2
s_3	s_0, s_1, s_2	s_1

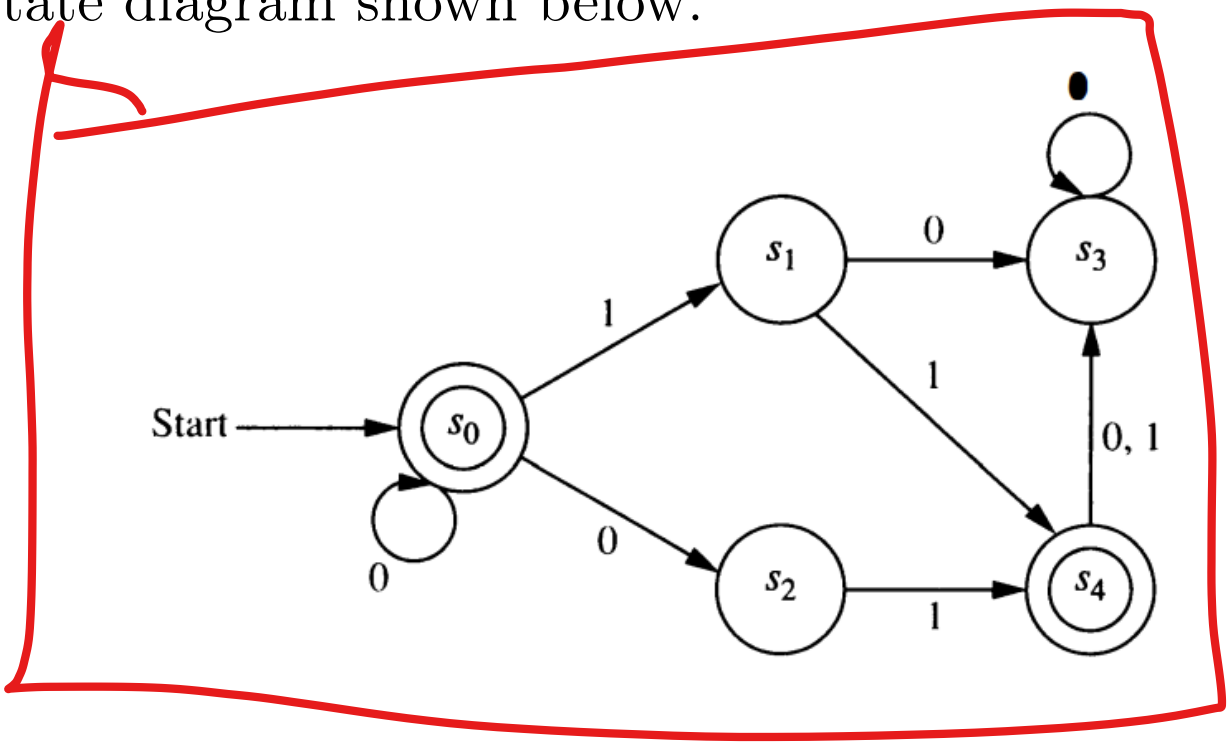


Example



Example

Find the state table for the nondeterministic finite-state automaton with the state diagram shown below.

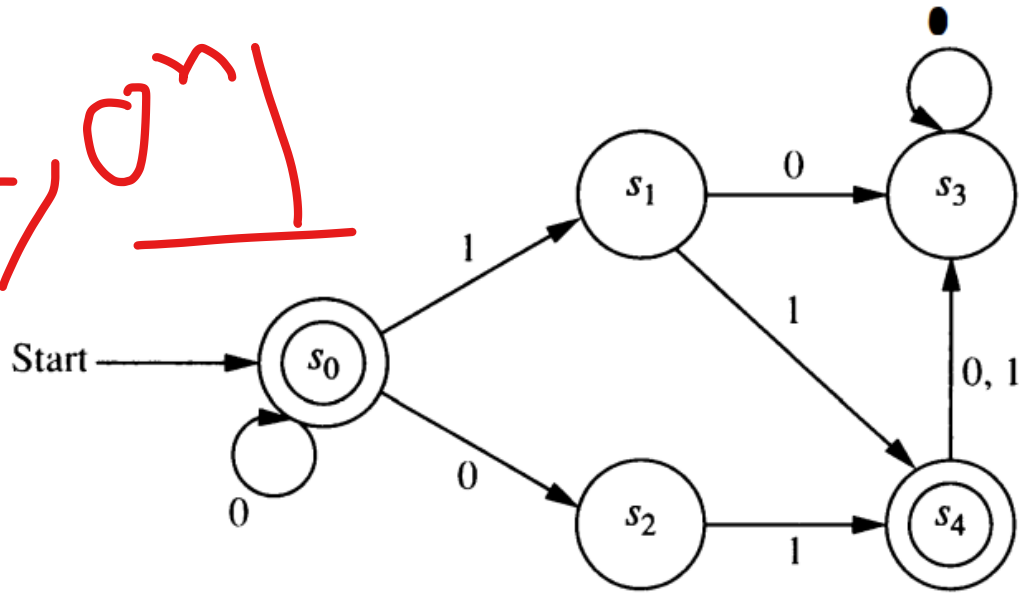


Example

State	f	
	Input	
	0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2		s_4
s_3	s_3	
s_4	s_3	s_3

Example

Find the language recognized by the nondeterministic finite-state automaton, M_2 .

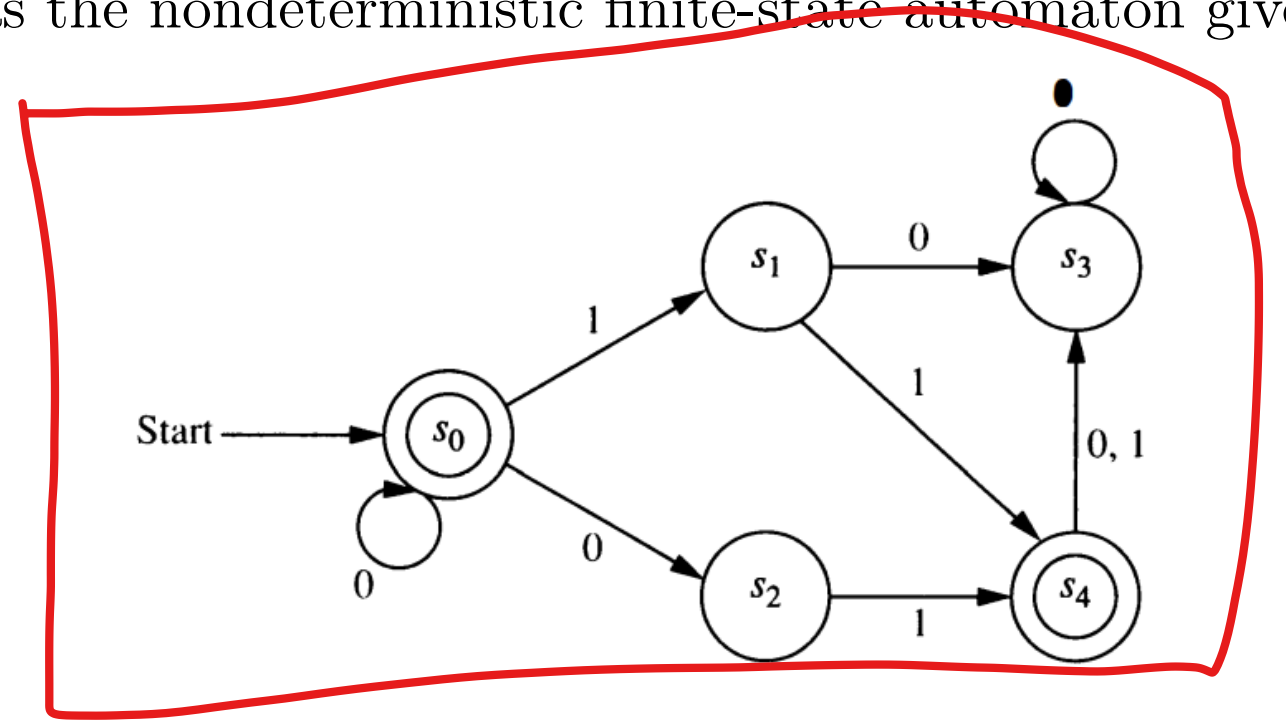


Theorem

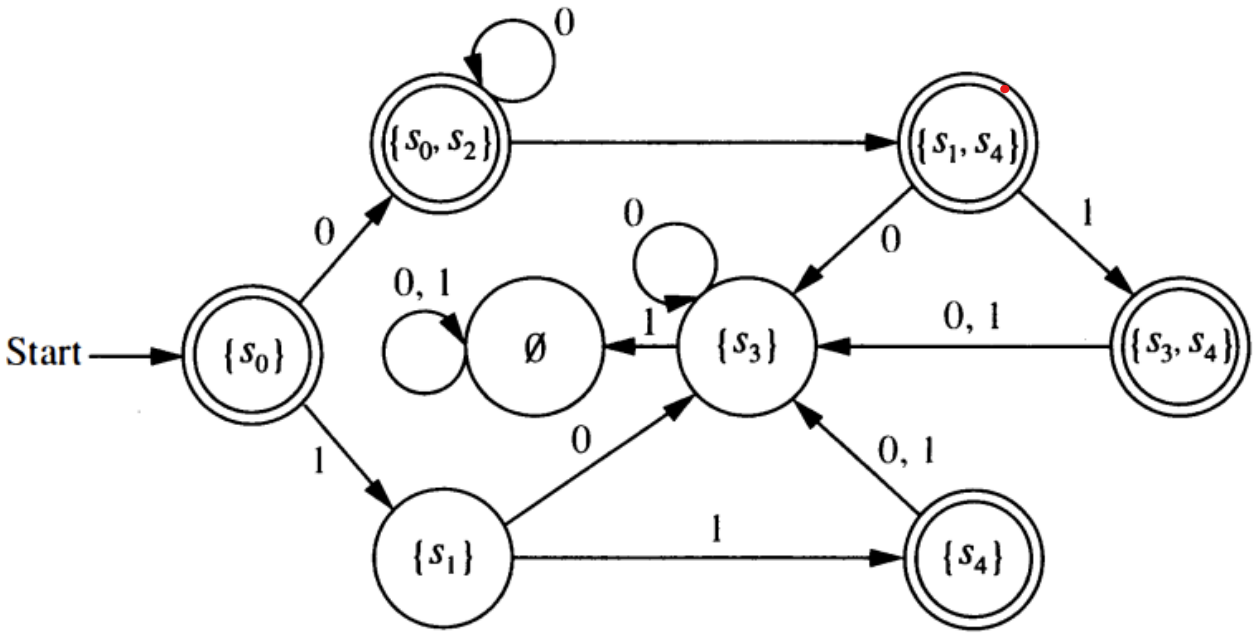
If the language L is recognized by a nondeterministic finite-state automaton M_0, then L is also recognized by a deterministic finite-state automaton M_1.

Example

Find a deterministic finite-state automaton that recognizes the same language as the nondeterministic finite-state automaton given below.

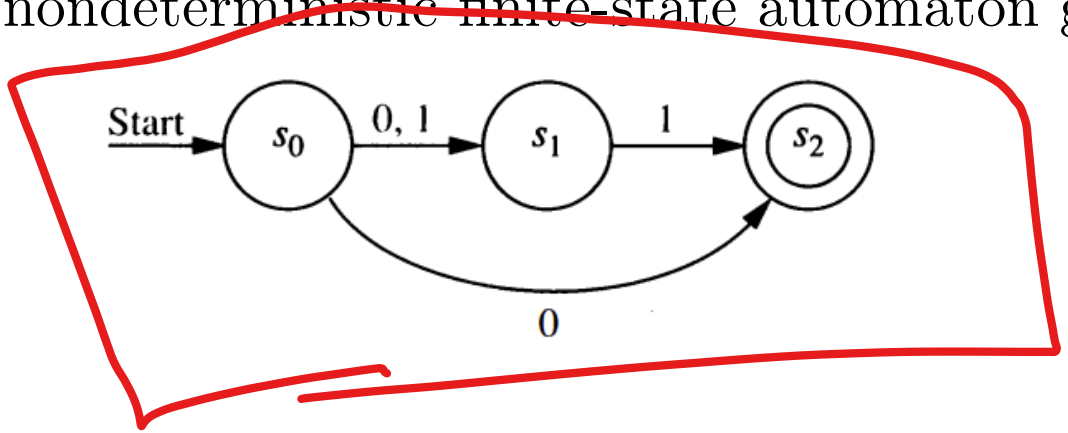


Solution



Example

Find a deterministic finite-state automaton that recognizes the same language as the nondeterministic finite-state automaton given below.



Example

Find a deterministic finite-state automaton that recognizes the same language as the nondeterministic finite-state automaton given below.