

Lecture Outline for Finite State Machines

- Main topics
 - Concept of finite state machine (FSM)
 - State table description
 - State graph description
 - Example: Binary Adder
 - Recognition of strings using FSM
 - Regular sets & Kleene's theorem
- Section 8.2 of text (only portions covered in class)

Concept of Finite State Machine

- Finite state machine (FSM) models a computer.
- It has the following characteristics
 - Synchronized by a discrete clock
 - Proceeds in a deterministic manner
 - Takes inputs from a given alphabet
 - Can generate outputs from a given alphabet
 - In one of a finite number of states
 - Next state is determined by current state and input

Definition of FSM

$M = [S, I, O, f_S, f_O]$ is the FSM

1. S is a finite set of states, with a given start state s_0
2. I and O are input and output alphabet
3. $f_S: S \times I \rightarrow S$
4. $f_O: S \rightarrow O$

Example of FSM

- Represented using a state table

Example of FSM

- Represented using state graph
- The two representations are equivalent and one can be converted to the other

Binary Adder

- Adder for summing up two binary integers
- Inputs given as tuples of two elements representing a particular digit
 - Sum(011, 101) is given as (11), (10), (01)
 - Start from the least significant bit
- States of the adder
 - s_0 : Output 0, Carry 0
 - s_1 : Output 0, Carry 1
 - s_2 : Output 1, Carry 0
 - s_3 : Output 1, Carry 1

Binary Adder

1. Input state: s_3 ; Input: (1 1)

- Output state: ?

2. Input state: s_3 ; Input: (1 0)

- Output state: ?

2. Input state: s_1 ; Input: (0 1)

3. Output state: ?

Binary Adder: State Diagram

- How many nodes?
- How many edges?

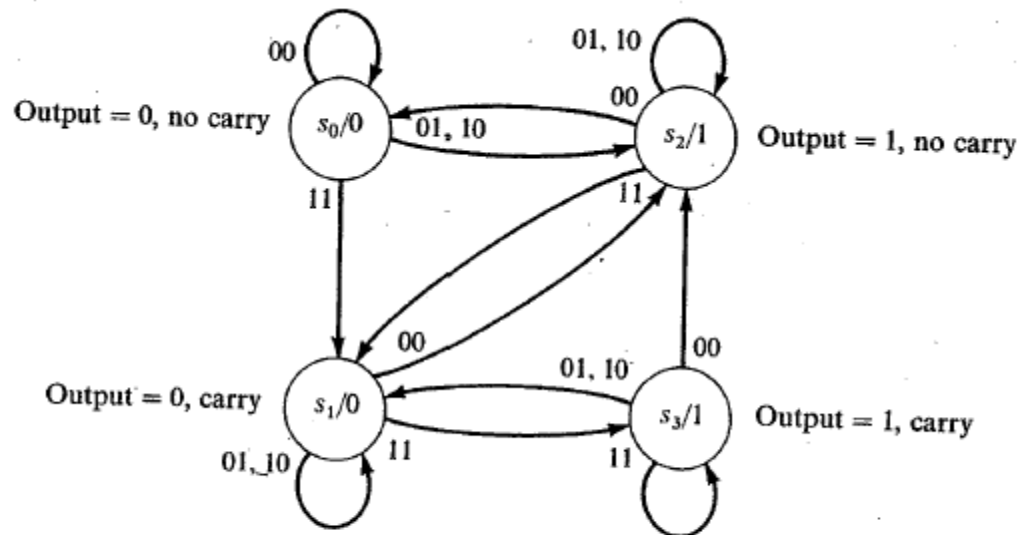


FIGURE 8.3

Binary Adder: Example

- Using the state diagram, compute the following
 - SUM(0110, 0011)
- Be very careful of order of bits

Recognizing Strings

- FSM can be used as *recognizer*
- FSM can output a 1 when the input string matches a certain description
- Example:
 - Even parity: Accept all strings which have an even number of 1's
 - If at t_i , there are an even number of 1's, output a 1 at t_{i+1}
 - Otherwise, output a 0

Recognizing Strings: Another Example

- Construct a FSM for the bitwise AND of two binary numbers

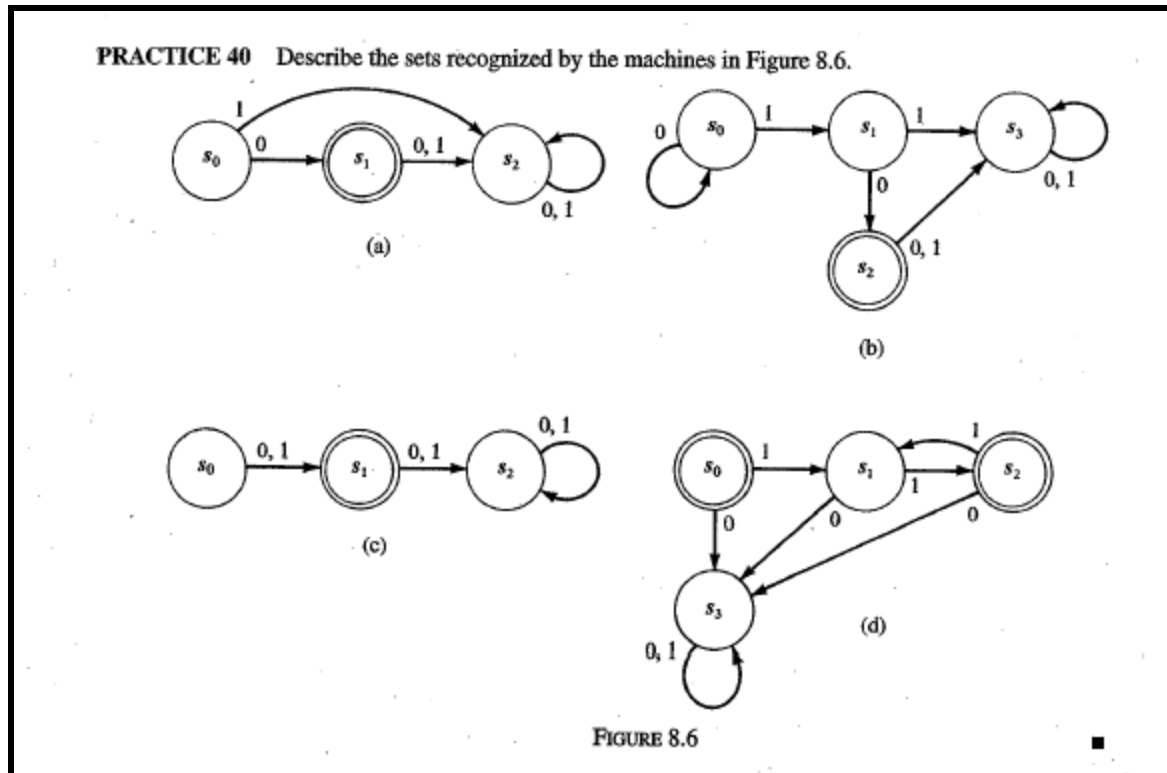
Formalizing recognition of strings

- There is one starting state
- There may be multiple final states where the FSM outputs a 1, i.e., accepts the string
- Consider a FSM $M=[S, I, O, f_S, f_O]$
- M is said to recognize $\sigma \subseteq I^*$ if
 - Beginning in start state s_0 , M processes an input string α and ends in a final state if and only if $\alpha \in \sigma$

Regular Expression

- a^* : 0 or more occurrences of a
- a^+ : 1 or more occurrences of a
- $a?$: 0 or 1 occurrence of a
- $[a-c]$: {a, b, c}
- $[a-z, A-Z]$: all alphabetical characters, lower or upper case
- λ : Empty string
- $a . b$: Concatenation of two strings a and b

What strings are recognized by the following FSMs?



Regular expressions

- You are given an input alphabet I
- A **regular expression** over I is one of the following
 - The symbols \emptyset and λ
 - The symbol i for any $i \in I$
 - If A and B are regular expressions, so are AB , $A \vee B$, A^*
- A **regular set** is a set represented by a regular expression if the following are true
 - \emptyset represents the empty set
 - λ represents the set $\{\lambda\}$ with the empty string
 - i represents the set $\{i\}$
 - Define (AB) , $(A \vee B)$, $(A)^*$

Example of regular expressions

1. $0^* \vee 10$

2. $(0 \vee 1)^*$

3. $0 \vee 1^*$

4. Regular expressions for the FSM's introduced in Practice 40?

Equivalent regular expressions

- There may be multiple ways of representing the same regular expression
 - $(11)^* = \lambda \vee (11)^+$
- No efficient algorithm that can decide if two regular expressions are equivalent has not been found

Kleene's Theorem

1. If a set is a regular set, then you can build a FSM to recognize strings in the set
 2. For a given FSM, you can write the strings that it recognizes as a regular set
- Counter-example: $\{0^m1^m \mid m \geq 0\}$

Construct an FSM that recognizes strings

1. The set of bit strings that start with two 0s.
2. The set of bit strings that contain two consecutive 0s.
3. The set of bit strings that do not contain two consecutive 0s.
4. The bit strings $\{0^*1\}$