# Finite Automata

# Summary

- Formal Languages
- Finite Automata
- Languages they recognize
- Examples
- Operations on Languages

# Natural Languages

- **Natural Languages**
  - Spoken languages such as English, French, German, Spanish…
  - Sentences can be broken down into two parts
    - Semantics
      - Meaning of a sentence
    - Syntax
      - Form of a sentence
      - Specifies if a sentence is valid
        - Valid: “the frog writes neatly”
        - Invalid: “swims quickly mathematics”
  - Extremely complicated and difficult to specify all rules of syntax.
    - Syntax may be inconsistent

- Natural languages are not suited for computers
  - Must develop **formal languages** which have well-defined rules of syntax.

# Formal Language Terms

- **Alphabet**
  - Any <u>nonempty</u> finite set
  - Members are called **symbols** of the alphabet
  - Usually designated by capital Greek letters (Δ,Σ,Π,…)

$$\Sigma_1 = \{0,1\}$$

$$\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$$

$$\Gamma = \{0, 1, x, y, z\}$$

- **String**
  - Finite sequence of symbols from an alphabet
  - Empty strings specified by ε

- **Language**
  - Set of strings
  - Can be sorted in either
    - **Lexicographic Order**
      - Same as dictionary order
    - **Shortlex (string) Order**
      - Sorted by string length than alphabetical order

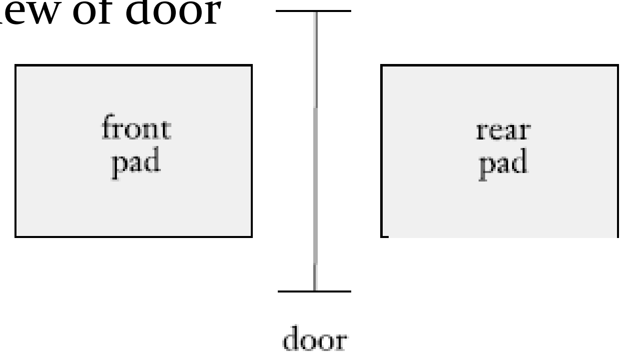
# Finite Automata

- **Finite Automata (FAs)**
  - A model for computation which works well for devices with limited memory
  - One of the simplest types of machines that can recognize patterns (strings).
  - Designed to:
    - Accept some input strings
    - Moves through states and either accepts or rejects the string
    - Recognize a language, which is the set of strings it accepts.
- One machine for strings of all length for a given formal language.

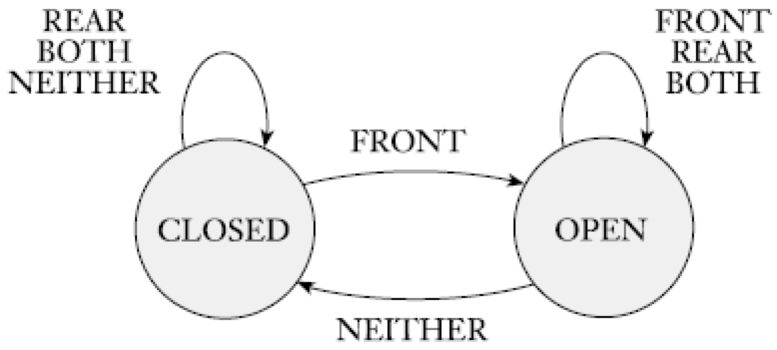
# Finite Automata to Control Devices

- Automatic Swinging Door Controller
  - Two States: “OPEN”, ”CLOSE”
  - Four Input Signals from pads:
    - “FRONT” – person standing on front pad
    - “REAR” – person standing on rear pad
    - “BOTH” – people on standing on both pads
    - “NEITHER” – no one on either pads

Top view of door




State Diagram



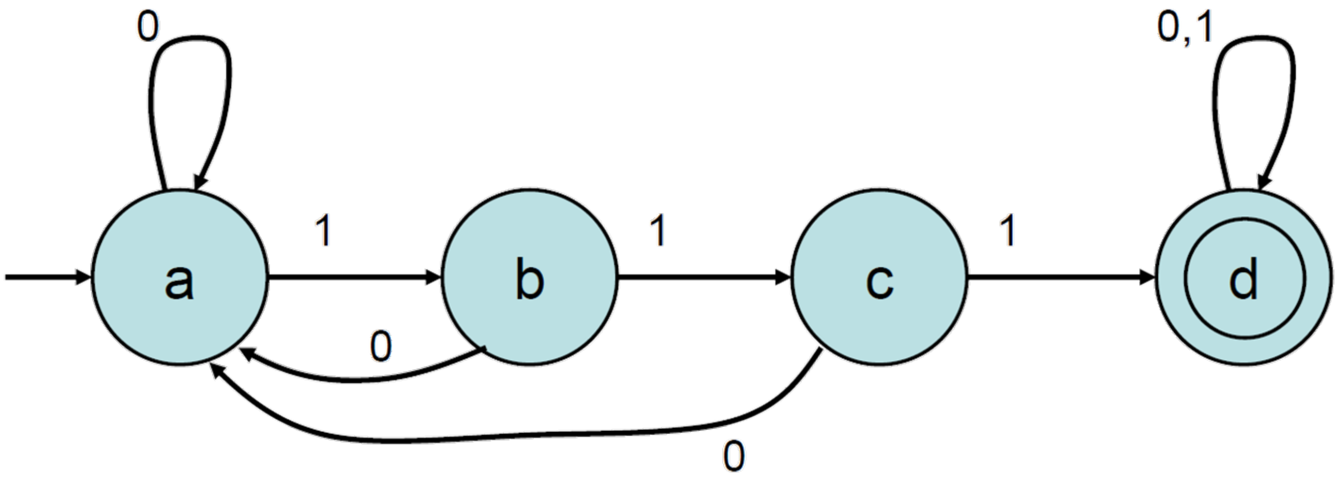

State Transition Table

| | | NEITHER | FRONT | REAR | BOTH |
|---|---|---|---|---|---|
| state | CLOSED | CLOSED | OPEN | CLOSED | CLOSED |
| | OPEN | CLOSED | OPEN | OPEN | OPEN |

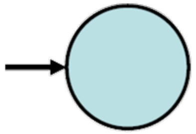
# Finite Automata Diagram

- Directed Multigraph
- String(word) is received at the **start state**
  - Accepted if transitions end at an **accept state**
  - String is rejected as a valid input if not

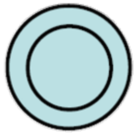
- An FA diagram, machine M



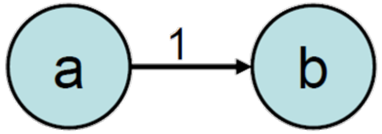

- Conventions:



Start state



Accept state

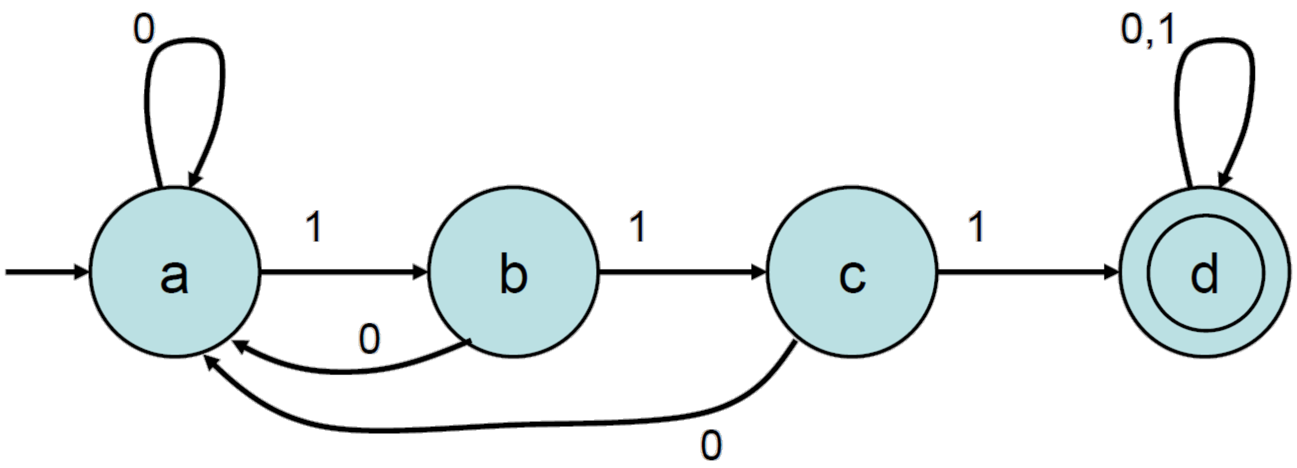



Transition from a to b on input symbol 1.
Allow self-loops

# Example 1

- **Language**, L
  - Any set of strings over some alphabet
- L(M), language recognized by M:
  - {w|wis accepted by finite automata M}
- **Regular** aka **FA-recognizable**
  - A language that is recognized by some finite automaton

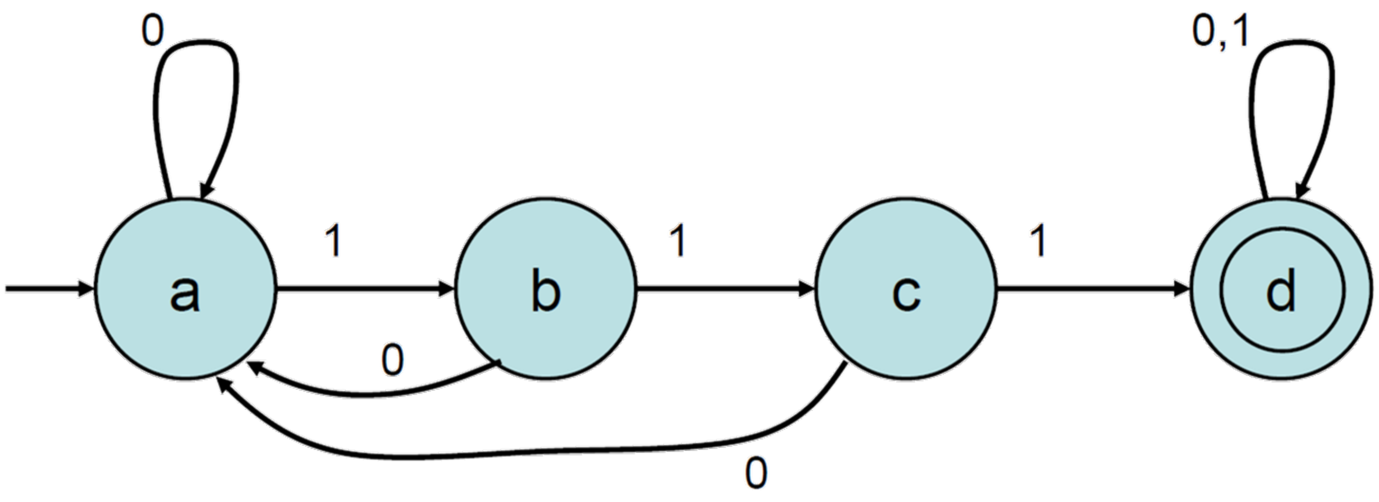
- What is L(M) for Example 1?




- Example computation:
  - Input word w: 1 0 1 1 0 1 1 1 0
  - States: a b a b c a b c d d
- We say that M accepts w, since w leads to d, an accepting state.

# Language of Finite Automata M

- Only strings of containing a substring of 111 ends at an accept state
- L(M) is the set of all strings that contain a 111 substring
- {0,1}* specifies set of all strings that contain symbols 0 and 1

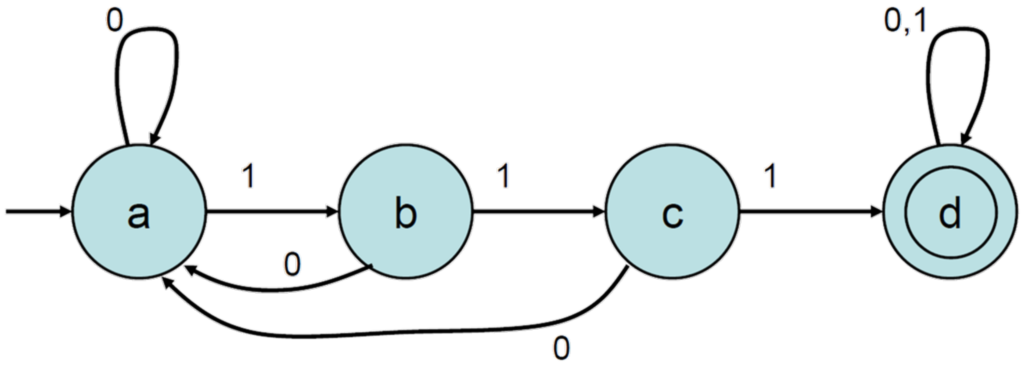



$L(M) = \{w \in \{0,1\}^* \mid w \text{ contains } 111 \text{ as a substring}\}$

# Formal Definition of an FA

- An FA can be formally defined as a 5-tuple $(Q,\Sigma,\delta,q_o,F)$, where:
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite set (alphabet) of input symbols
  - $\delta: \underbrace{Q \text{ x } \Sigma}_{\text{The arguments of } \delta \text{ are a state and an alphabet symbol.}} \rightarrow \underbrace{Q}_{\text{The result is a state.}}$ is the **transition function**
  - $q_o \in Q$, is the start state
  - $F \subseteq Q$, set of accept states

# Formal Definition of Example 1

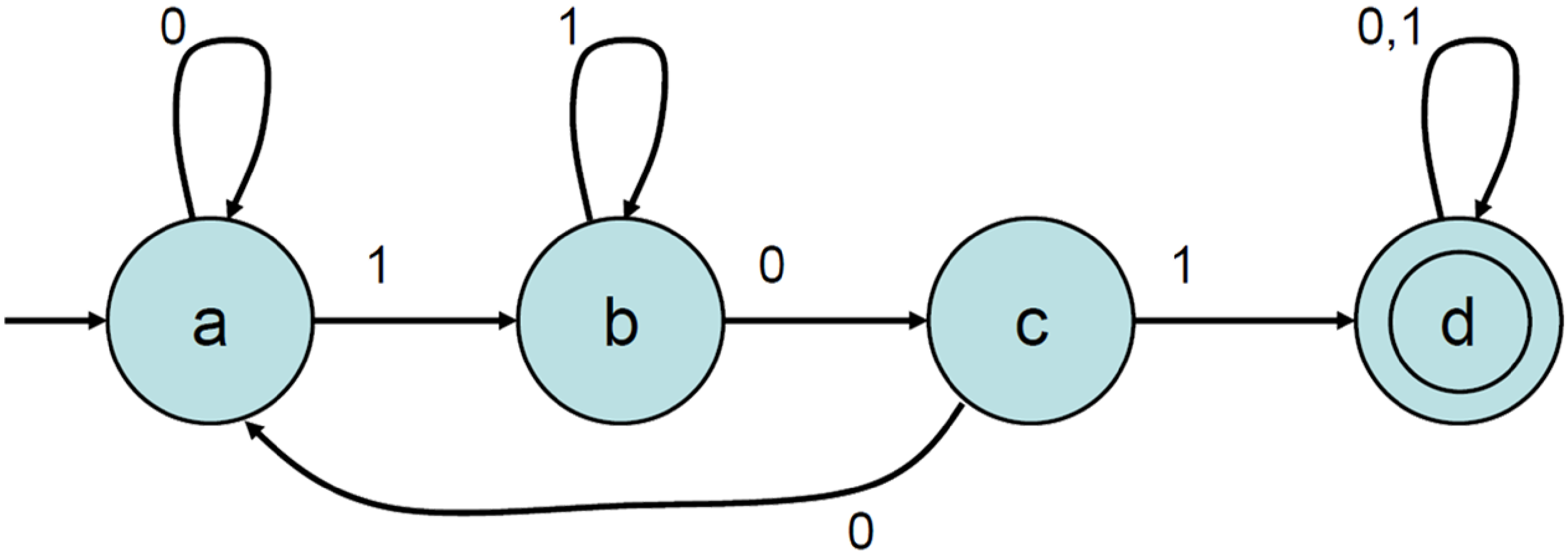



- List all possible states
  - $Q = \{a,b,c,d\}$
- List all symbols in the alphabet
  - $\Sigma = \{0,1\}$
- Specify state transition function with diagram or table
  - $\delta$ by table:
    - Rows represent current state
    - Columns current signal
    - Elements represent new state being mapped to.
- Specify start state
  - $q_0 = a$
- List all accept states
  - $F = \{d\}$

| | 0 | 1 |
|---|---|---|
| a | a | b |
| b | a | c |
| c | a | d |
| d | d | d |

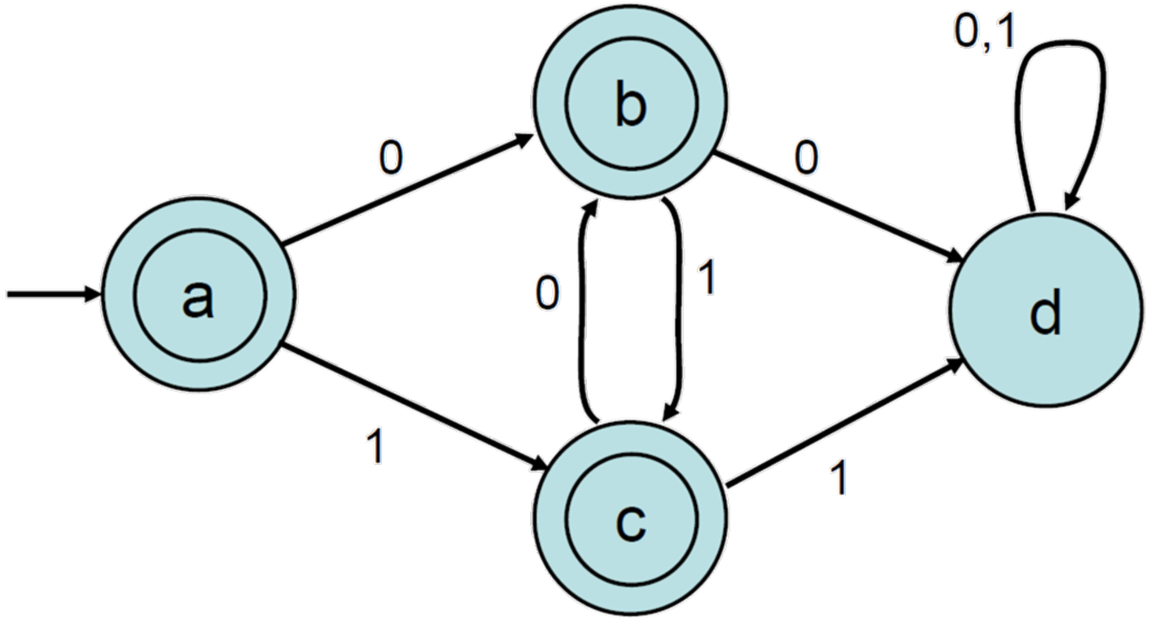
# Example 2: Different Substring

- Design an FA M with L(M)={w∈{0,1}*|w contains 101 as a substring}



# Example 3: Trap State

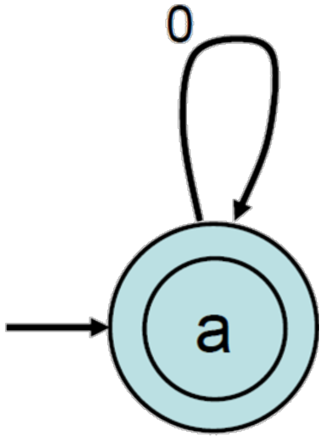
- L(M)={w∈{0,1}*|w doesn't contain either 00 or 11 as a substring}



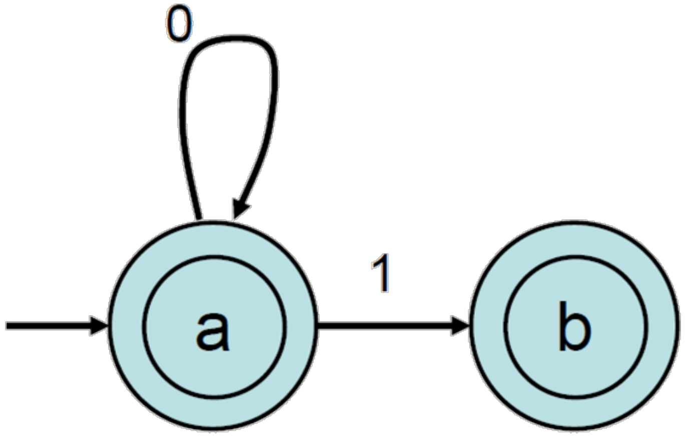

- State d is a **trap state**
  - A nonaccepting state that can't leave
    - String is rejected as it is impossible to be accepted
  - Sometime some arrows are omitted
    - By convention, they go to a trap state

# Example 4: Building Diagram

- $L(M)=\{w\in\{0,1\}^*|\text{all nonempty blocks of 1s in } w \text{ have odd length}\}$
  - E.g., ε, 100111000011111, or any number of zeros
  - Initial zeros don't matter, so start with:

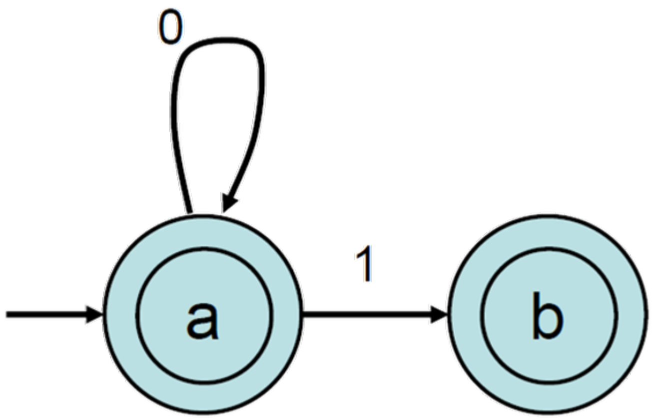


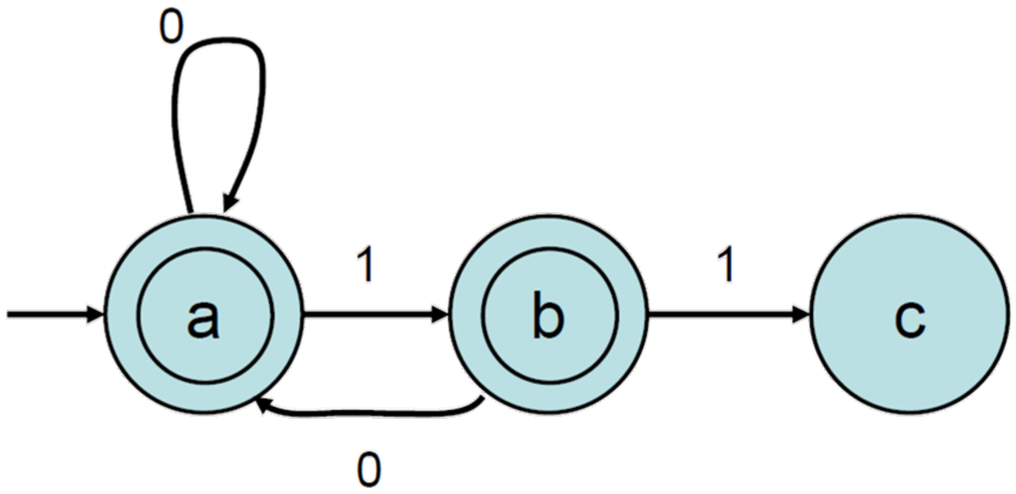

- Then 1 also leads to an accepting state, but it should be a different one, to "remember" that the string ends in one 1



# Example 4 : Building Diagram

- L(M)={w∈{0,1}*|all nonempty blocks of 1s in w have odd length}
- From b:
  - 0 can return to a, which can represent either ε, or any string that is OK so far and ends with 0
  - 1 should go to a new nonaccepting state, meaning "the string ends with two 1s"

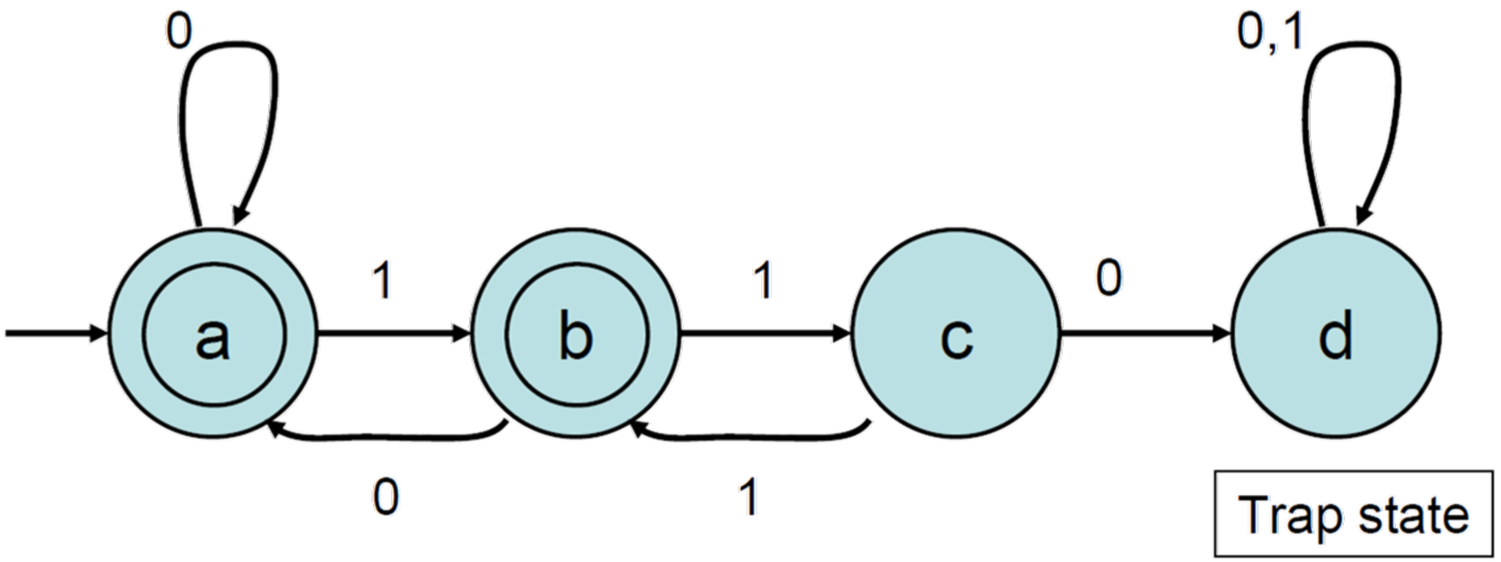



- Note: c isn't a trap state
  - We can accept some extensions

# Example 4 : Building Diagram

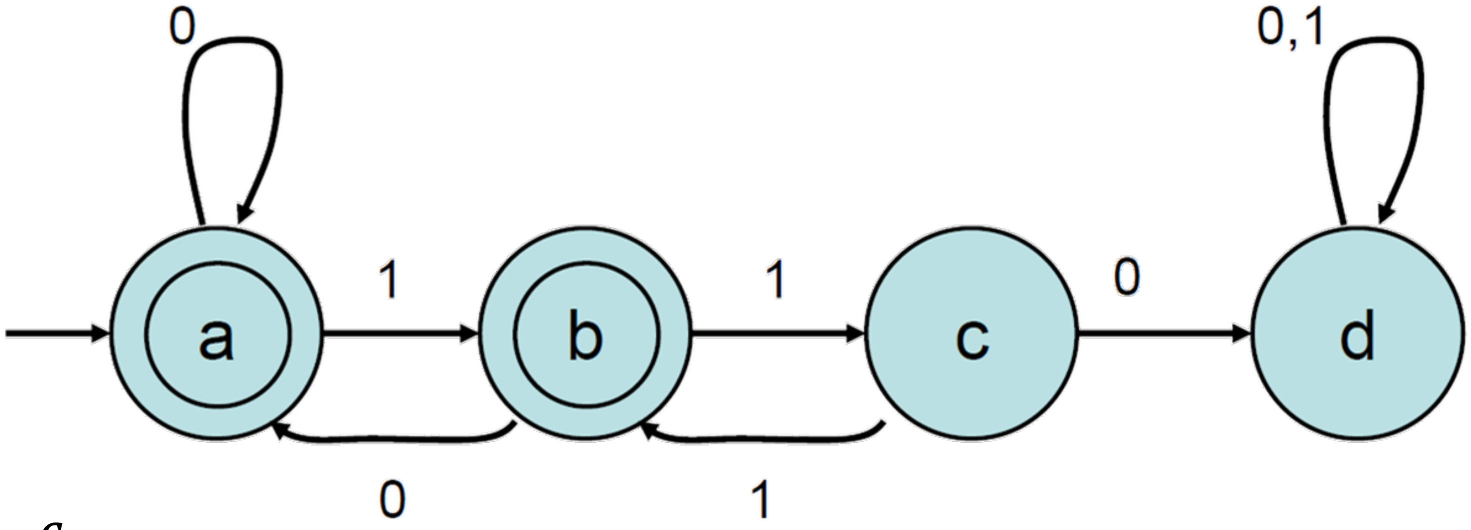
- L(M)={w∈{0,1}*|all nonempty blocks of 1s in w have odd length}




- From c:
  - 1 can lead back to b, since future acceptance decisions are the same if the string so far ends with any odd number of 1s
    - Reinterpret b as meaning " ends with an odd number of 1s"
    - Reinterpret c as "ends with an even number of 1s"
  - 0 means we must reject the current string and all extensions

# Example 4 : Building Diagram

- L(M)={w∈{0,1}*|all nonempty blocks of 1s in w have odd length}




- Meanings of states:
  - a: Either ε, or contains no bad block (even block of 1s followed by 0) so far and ends with 0
  - b: No bad block so far, and ends with odd number of 1s
  - c: No bad block so far, and ends with even number of 1s
  - d: Contains a bad block

# Example 5

- $L(M) = EQ = \{w \in \{0,1\}^* \mid w$ contains an equal number of zeros and ones$\}$

- No FA recognizes this language
  - Not a regular language

- Reasoning
  - Machine must “remember” how many zeros and ones it has seen, or at least the difference between these numbers

  - Since these numbers (and the difference) could be anything, there can’t be enough states to keep track

  - So the machine will sometimes get confused and give a wrong answer