Deterministic Finite Automata (DFA)

Asai Asaithambi

Spring 2018

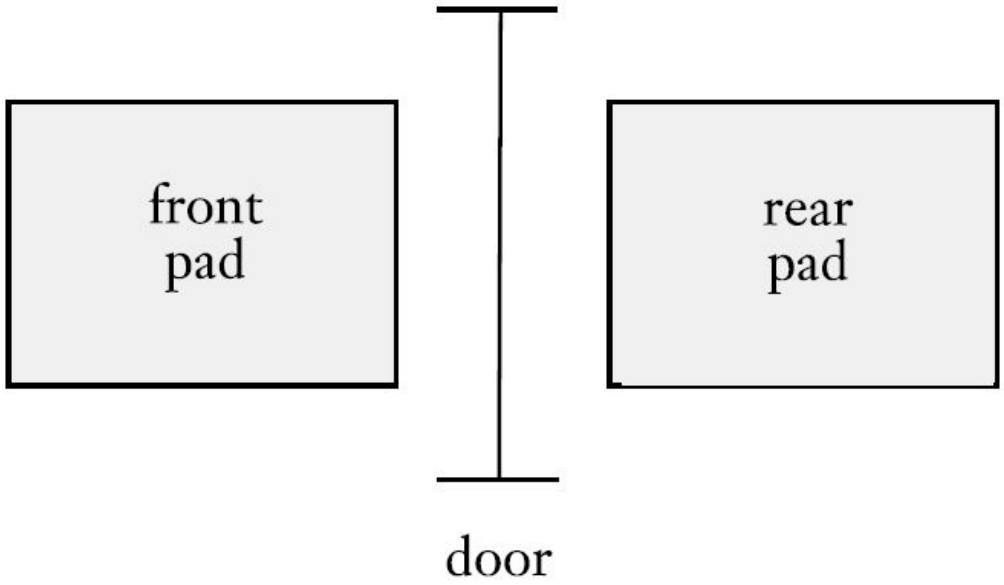
Topic Overview

What does this topic cover?

- Finite State Machines (FSM)
  - Pictorial Representation
  - Formal Definition
  - Computation in a FSM
  - String Recognition/Language
- Regular Language/Operations
  - The union of two regular languages is regular
- Deterministic vs. Nondeterministic

Finite State Machines: General Ideas

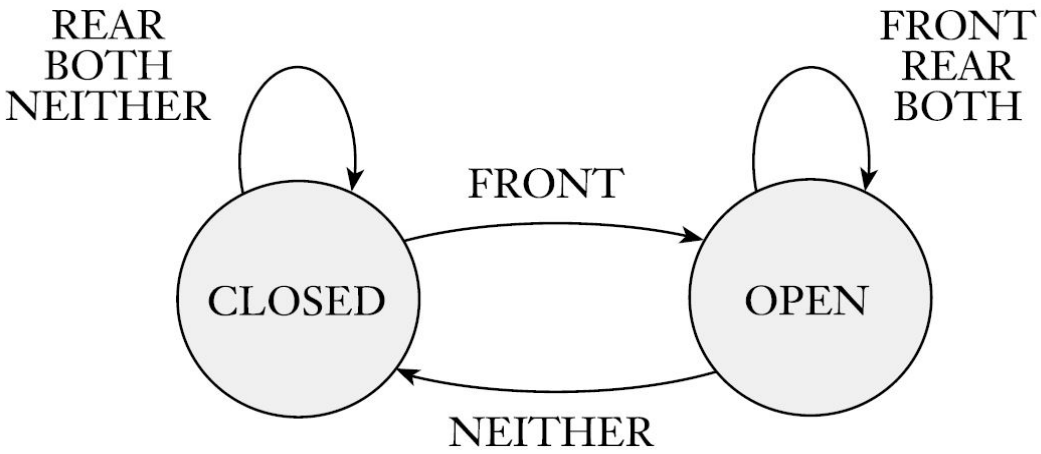
- Simplest model of computation
- A small computer with very limited memory
- "Finite" and small
- An Example: Controller for an automatic (swinging) door
- Two pads: front and rear (of the door)



A Swinging Door Controller

- The door is in one of two states: "OPEN" or "CLOSED"
- Four possible input conditions:
  - FRONT (Person standing in front of the doorway)
  - BACK (Person standing to the rear of the doorway)
  - BOTH (People standing in front and to the rear)
  - NEITHER (No one standing on either pad)
- Controller goes from one state to another:
  - Based on input received

State Diagram and Transition Table



---

input signal

state	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

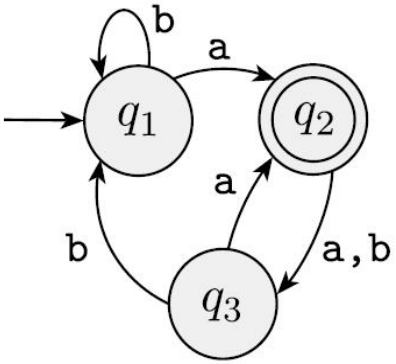
State Diagram/FSM Characteristics

- States drawn as circles (nodes)
- Transitions drawn as arcs (edges)
- Always exactly one starting or initial state
- Final or accepting states (there may be more than one)
- Input will be called the *alphabet* (Σ)
- A FSM can "accept" or "reject" strings
  - Start in the starting state
  - Start at the beginning of string
  - Take transitions based on symbol in the string
  - Process all symbols in the string
  - String "accepted" if you reach one of the final states
  - String "rejected" otherwise.
  - Question: What strings are accepted?

FSM: Formal Definition

5-tuple: $(Q, \Sigma, \delta, q_0, F)$

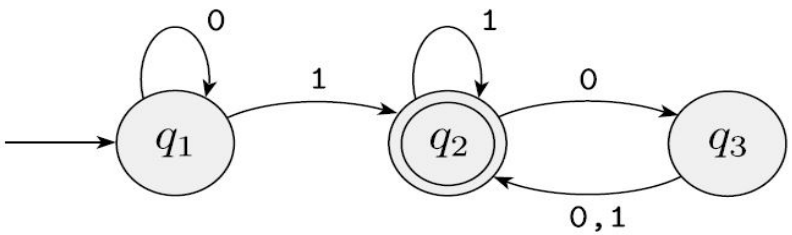
- Q: (Finite) Set of States
- Σ: Alphabet, a (finite) set of symbols
- δ: Transition Function, $\delta : Q \times \Sigma \mapsto Q$
- q_0: unique start or initial state, $q_0 \in Q$
- F: set of final states, $F \subseteq Q$



δ	a	b
q_1	q_2	q_1
q_2	q_3	q_3
q_3	q_2	q_1

$(\{q_1, q_2, q_3\}, \{\mathsf{a}, \mathsf{b}\}, \delta, q_1, \{q_2\})$

FSM: Example Machine M_1



	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

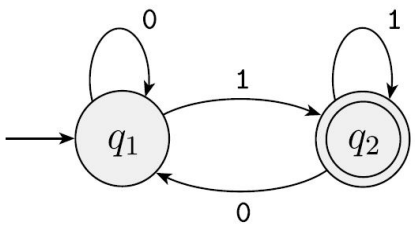
M_1 accepts the language A, where

$$A = L(M_1) = \{w \mid w \text{ contains at least one 1 and an even number of 0s follow the last 1}\}.$$

A is the *language of the machine* M_1.

FSM: Example Machines M_2 and M_3

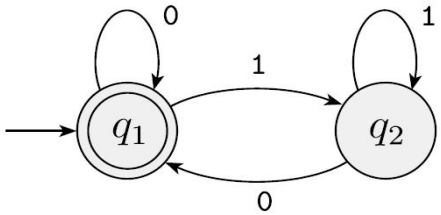
M_2



	0	1
q_1	q_1	q_2
q_2	q_1	q_2.

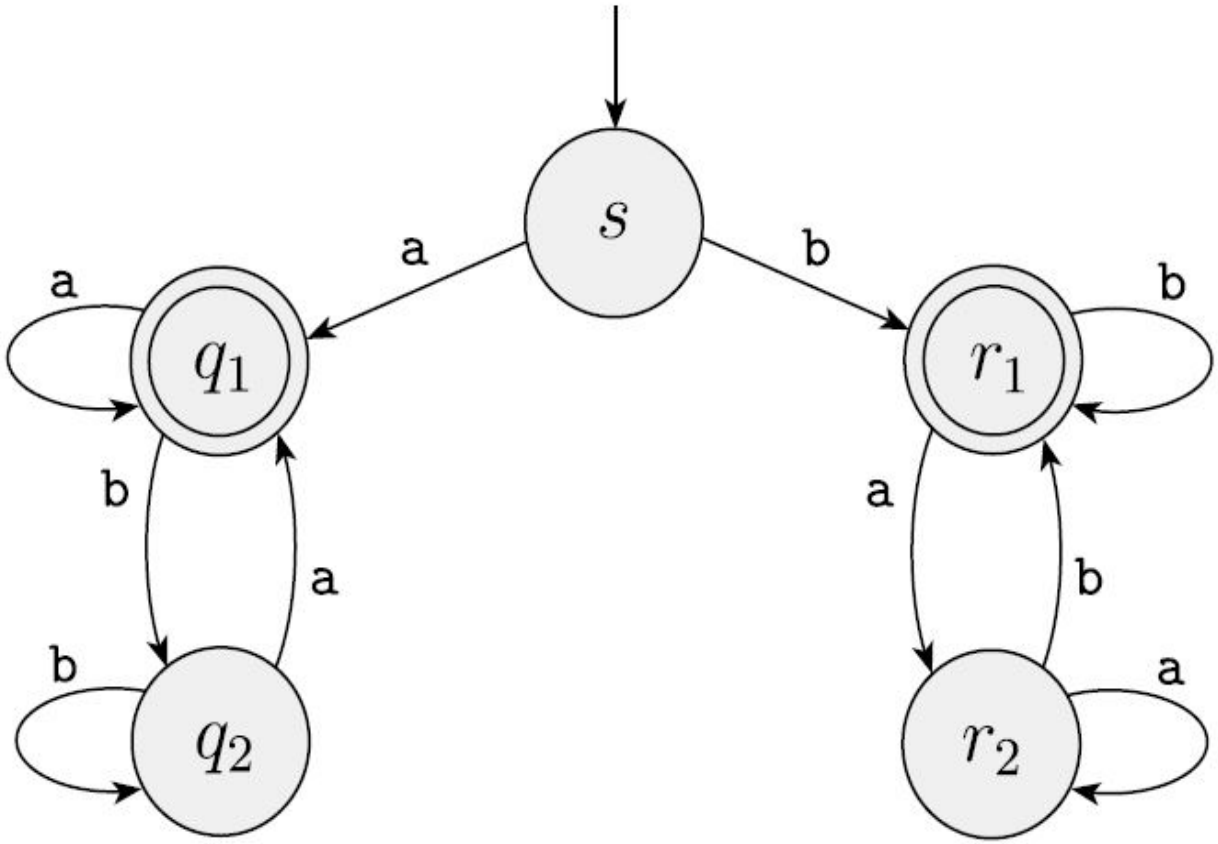
$L(M_2) = \{w \mid w \text{ ends in a } 1\}$

M_3



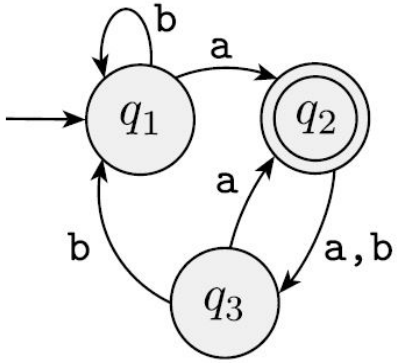
$L(M_3) = \{w \mid w \text{ is the empty string } \varepsilon \text{ or ends in a } 0\}$

FSM: Example Machine M_4



$L(M_4) = \{w \mid w \text{ begins and ends with the same symbol}\}$.

FSM: Example Computations



Processing the string aabb:

$q_1 \xrightarrow{\text{a}} q_2 \xrightarrow{\text{a}} q_3 \xrightarrow{\text{b}} q_1 \xrightarrow{\text{b}} q_1$

q_1 is not a final state;

$\therefore$ aabb is not accepted.

Processing the string abba:

$q_1 \xrightarrow{\text{a}} q_2 \xrightarrow{\text{b}} q_3 \xrightarrow{\text{b}} q_1 \xrightarrow{\text{a}} q_2$

q_2 is a final state;

$\therefore$ abba is accepted.

Formal Definition of Computations

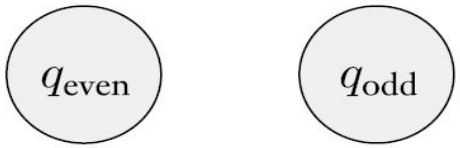
- $M = (Q, \Sigma, \delta, q_0, F)$
- How does M process the string $w = w_1 w_2 \cdots w_n,\ w_i \in \Sigma$
- M starts at state q_0 and receives input w_1
- It reaches the next state as determined by δ
- At this next state, it receives the next input w_2
- In this manner, it reaches the sequence of states
- $r_0, r_1, \cdots, r_n \in Q$ such that:
  - $r_0 = q_0$
  - $r_{i+1} = \delta(r_i, w_{i+1}),\ i = 0, 1, 2, \cdots, n-1$
- w is accepted, or $w \in L(M)$ if $r_n \in F$
- w is not accepted, or $w \notin L(M)$ if $r_n \notin F$

Designing Finite Automata (FA)

- A FA accepts strings, recognizes languages.
- Wish to design a FA that recognizes a given language.
- The designer acts as the automaton.
- Processes strings symbol by symbol.
- After each symbol is "read," needs to decide:
  - Whether the string read "so far" is in the language.
  - Whether the symbol read is "crucial information."
- States serve as "memory"
- As the number of states needs to be "finite,"
  - Needs to remember only a few pieces of information;
  - Exactly what is crucial will depend on the language.

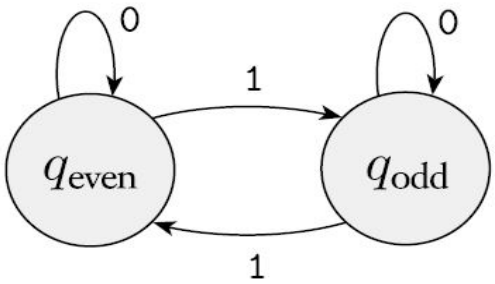
Designing a FA: Example 1

- To recognize all bit strings with an odd number of 1s
- Entire string need not be remembered
- 0s need not be counted or the number of 0s need not be remembered
- Must remember and keep track of whether:
  - the number of 1s read so far is even;
  - the number of 1s read so far is odd;
- This means two states will be needed:
  - even so far;
  - odd so far;



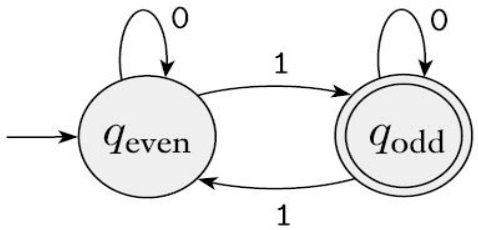
FA Example 1 (cont.)

- When the FA is in state q_{even}:
  - Reading a 1 takes it to state q_{odd};
  - Reading a 0 keeps it in state q_{even}.
- When the FA is in state q_{odd}:
  - Reading a 1 takes it to state q_{even};
  - Reading a 0 keeps it in state q_{odd}.



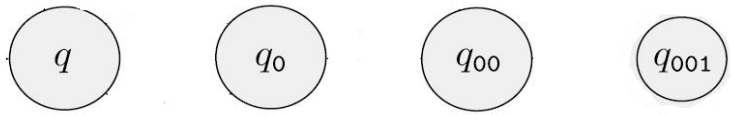
FA Example 1 (finish up)

- Set the start state and the final state
- Which one is the initial state?
  - Initial state: no symbols read yet (zero symbols or ε)
  - Since zero is even, initial state is q_{even}.
- Which one is the final state?
  - Final/Accept state: Odd number of 1s read
  - q_{odd} is set as an accept state.



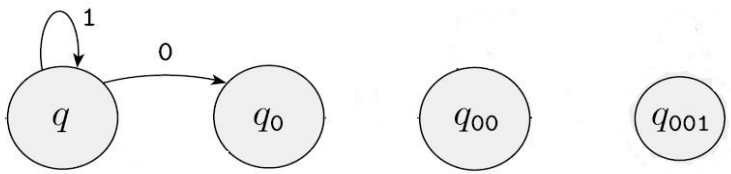
Designing a FA: Example 2

- To recognize all bit strings that contain 001 as a substring.
- Entire string need not be remembered
- There are four possibilities: You have just seen:
  - No symbols as yet;
  - A 0;
  - The pattern 00; and
  - The pattern 001.
- Assign states q, q_0, q_{00}, and q_{001}.



FA Example 2 (cont.)

	0	1
q	q_0	q
q_0		
q_{00}		
q_{001}		

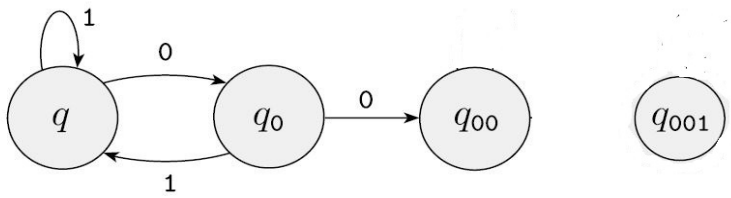


When in state q, reading:

- A 0 will take you to state q_0;
- A 1 will still leave you in state q.

FA Example 2 (cont.)

	0	1
q	q_0	q
q_0	q_{00}	q
q_{00}		
q_{001}		

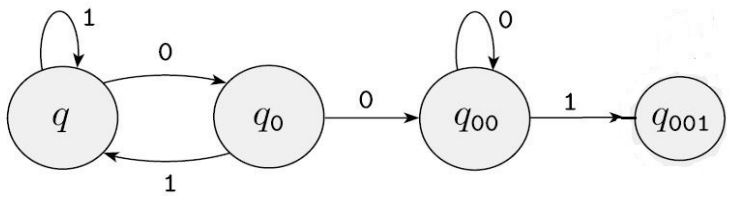


When in state q_0, reading:

- A 0 will take you to state q_{00};
- A 1 will return you to state q.

FA Example 2 (cont.)

	0	1
q	q_0	q
q_0	q_{00}	q
q_{00}	q_{00}	q_{001}
q_{001}		

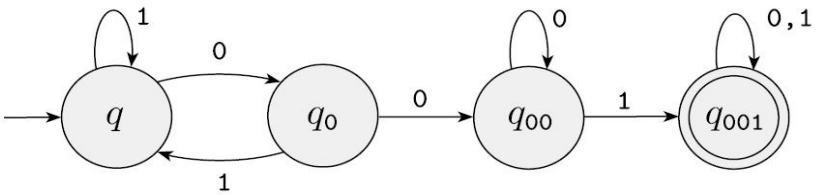


When in state q_{00}, reading:

- A 0 will leave you in state q_{00};
- A 1 will return you to state q_{001}.

FA Example 2 (finish up)

	0	1
q	q_0	q
q_0	q_{00}	q
q_{00}	q_{00}	q_{001}
q_{001}	q_{001}	q_{001}



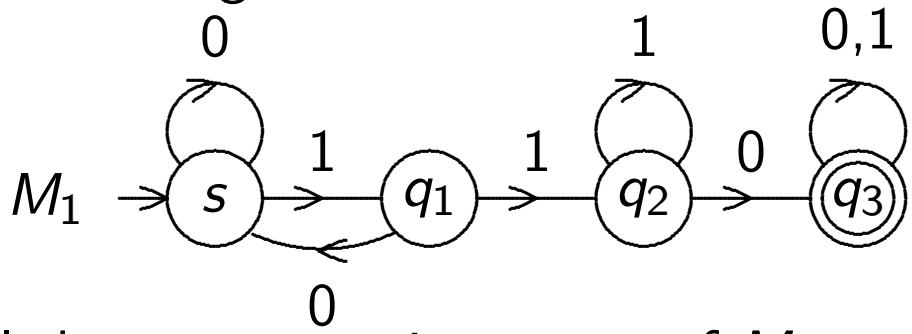
When in state q_{001}, reading:

- A 0 or 1will leave you in state q_{001};
- A 1 will leave you in state q_{001}.

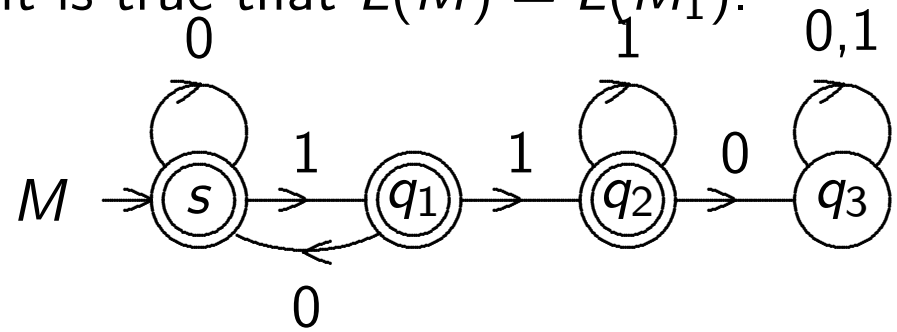
q is the start state and q_{001} is the accept state.

Designing a FA: Example 3

To construct a machine M that recognizes all bit strings not containing 110 as a substring, we construct M_1 that recognizes all bit strings containing 110 as a substring.



Then, by making all the non-accepting states of M_1 accepting states, and the accepting states of M_1 as non-accepting states, we obtain the desired machine M. Then, it is true that $L(M) = \overline{L(M_1)}$.



Regular Languages and Operations

- Regular Languages
  - We say that a language is *regular*, if:
    some finite automaton recognizes it.
  - The languages we have seen so far are regular.
- Regular Operations: Let A and B be languages.
- Then we define:
  - Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
  - Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
  - Star: $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Regular Operations Example

Let the alphabet Σ be the standard 26 letters $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. If $A = \{\mathtt{good}, \mathtt{bad}\}$ and $B = \{\mathtt{boy}, \mathtt{girl}\}$, then

$$A \cup B = \{\mathtt{good}, \mathtt{bad}, \mathtt{boy}, \mathtt{girl}\},$$

$$A \circ B = \{\mathtt{goodboy}, \mathtt{goodgirl}, \mathtt{badboy}, \mathtt{badgirl}\}, \text{ and}$$

$$A^* = \{\varepsilon, \mathtt{good}, \mathtt{bad}, \mathtt{goodgood}, \mathtt{goodbad}, \mathtt{badgood}, \mathtt{badbad}, \mathtt{goodgoodgood}, \mathtt{goodgoodbad}, \mathtt{goodbadgood}, \mathtt{goodbadbad}, \ldots\}.$$

More on Regular Languages and Operations

- The class of regular languages is *closed* under regular operations.
- Let Σ be the alphabet.
- Let A_1 and A_2 be regular languages over Σ. Then:
  - $A_1 \cup A_2$ is regular.
  - $A_1 \circ A_2$ is regular.
  - A_1^* and A_2^* are regular.
- To show that $A_1 \cup A_2$ is regular:
  - We let M_1 and M_2 be FAs with: $L(M_1) = A_1$ and $L(M_2) = A_2$
  - We construct a FA M, with $L(M) = A_1 \cup A_2$

Constructing FA for $A_1 \cup A_2$

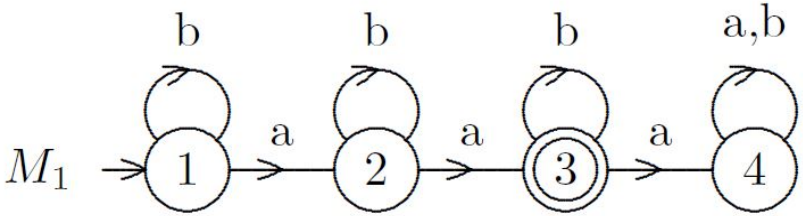
- Let $A_1 = L(M_1)$, with $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
- Let $A_2 = L(M_2)$, with $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- Let
  - $L(M) = A_1 \cup A_2 = L(M_1) \cup L(M_2)$, with
  - $M = (Q, \Sigma, \delta, q_0, F)$.
- Then:
  - $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$
  - Σ is the same as for both M_1 and M_2
  - δ for M is defined using δ_1 and δ_2:

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

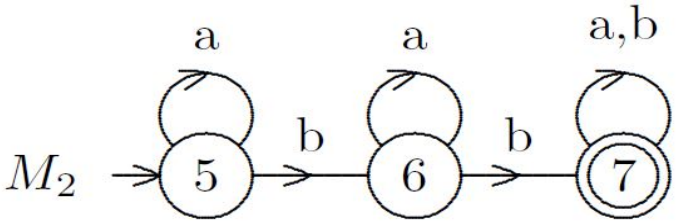
  - $q_0 = (q_1, q_2)$
  - $F = F_1 \times Q_2 \cup F_2 \times Q_1$
  - Note: For $A_1 \cap A_2$, we set $F = F_1 \times F_2$

FAs for Two Languages A_1 and A_2

- Let $A_1 = L(M_1) = \{w \mid w \text{ contains exactly two a's}\}$.

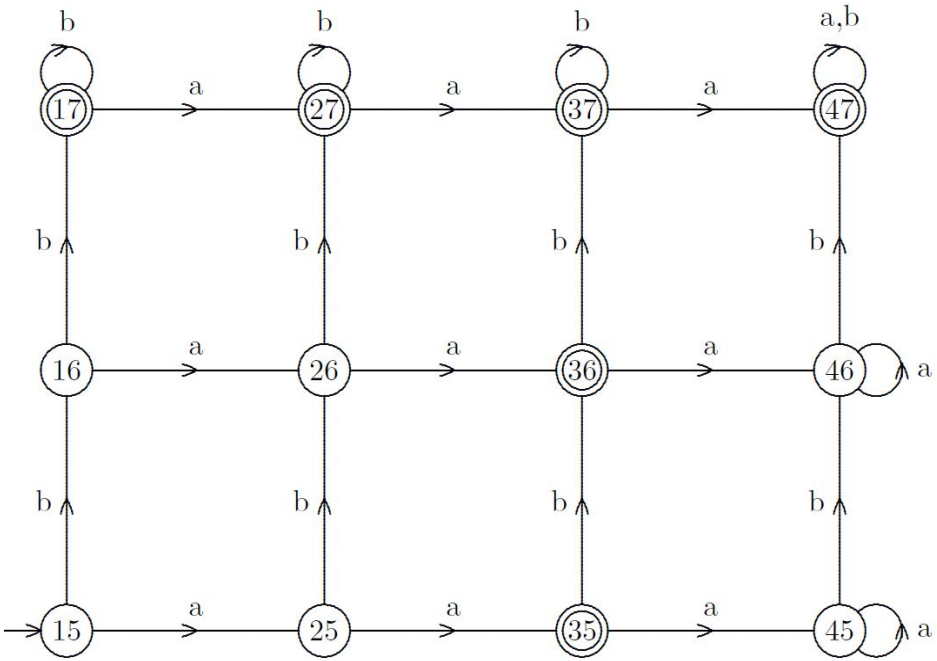


- Let $A_2 = L(M_2) = \{w \mid w \text{ contains at least two b's}\}$.



FA to Accept $A_1 \cup A_2$

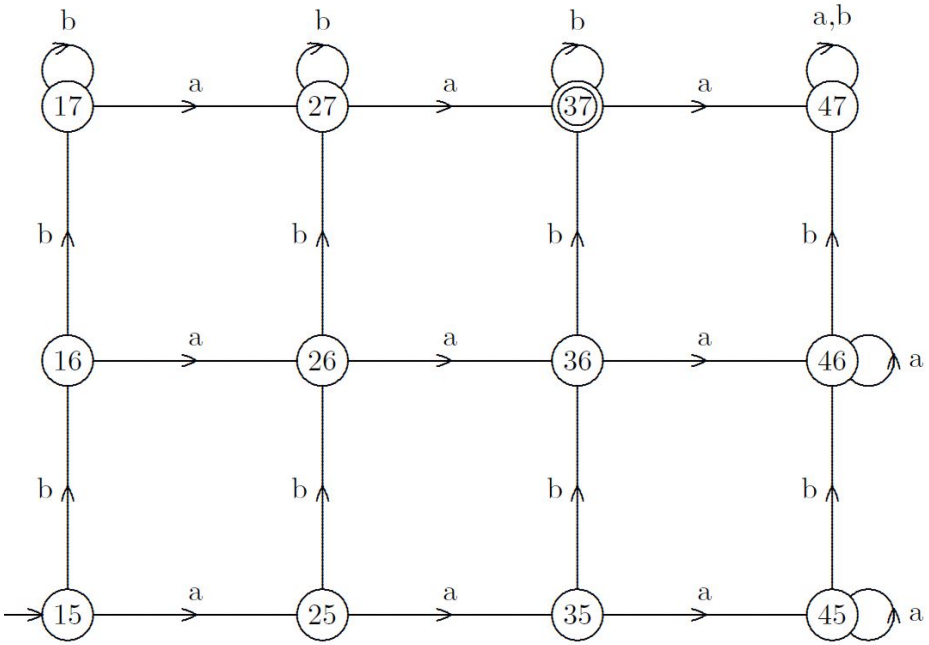
- State (m, n) labeled mn in the picture.



- $A_1 \cup A_2 = \{w \mid w$ has exactly two a's or at least two b's$\}$

FA to Accept $A_1 \cap A_2$

- State (m, n) labeled mn in the picture.



- $A_1 \cup A_2 = \{w \mid w$ has exactly two a's and at least two b's$\}$

Deterministic vs. Nondeterministic

- We need the concept of nondeterminism to show that regular languages are closed under the concatenation and star operations,
- A FA in which the next state is uniquely defined for every state and input pair is called a deterministic finite automaton (DFA). We have considered only DFAs so far.
- A FA in which the next state may be a set of states instead of a unique state is called a nondeterministic finite automaton (NFA). The next state may be the empty set ($\emptyset$) as well.
- Additionally, transitions in a NFA may be labeled with ε (the empty string) as well. Note that the empty string is not part of the alphabet, but ε transitions may occur in NFAs.
- The study of NFAs is our next topic in the course.