Theory of Formal Languages and Automata

Lecture 3

Mahdi Dolati

Sharif University of Technology

Fall 2025

February 17, 2025

Finite Automata

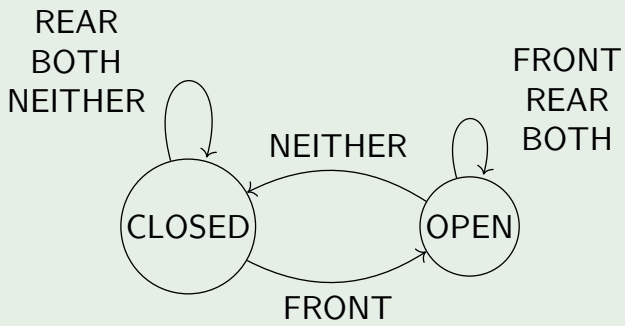
- What is a computer?
  - Computational model: An idealized computer
  - Use different models when considering different features
  - Finite state machine or finite automaton: Simplest model
    - A FA either accept or reject its input string.

Finite Automata

- A computer with extremely limited amount of memory

Example (Automatic Doors)

- Two states = {OPEN, CLOSED}
- Needs 1 bit of memory



- State transition table

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

Finite Automata

- Other examples:
  - Elevator controller
  - Electronic thermostat

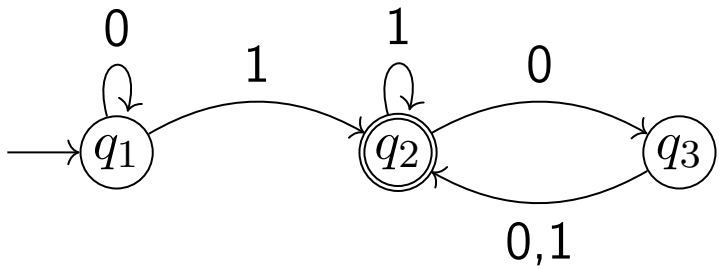
- The states (a finite set) are the memory,
- Starts from a fixed initial state,
- Change state based by reading the input one symbol at a time,
- Accept if is in an accept state after reading the input.
- Standard ways of representation,
- Methodology and terminology helps to design such devices.

Finite Automata

- A simple setting to start
  - Definition
  - Manipulation
  - Power and limitations

Finite Automata

Finite automaton M_1:



- State diagram of M_1
- State: A circle with a name
  - Start state: An arrow points to it
  - Accept state: Two circles
- Input: Strings (consume one-by-one)
  - 1101
- Transitions: arrows (match on the label)
- Output: {accept, reject} after reading last symbol

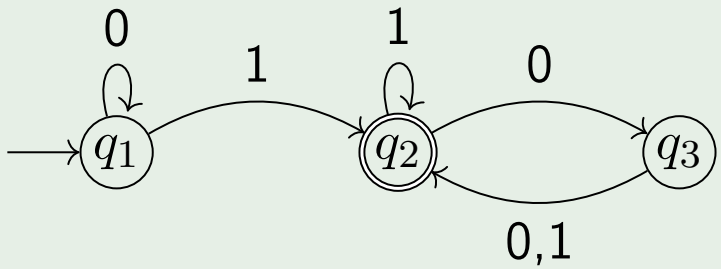
Finite Automata

Definition (Finite Automaton)

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

1. Q: states,
2. Σ: alphabet,
3. δ: $Q \times \Sigma \to Q$ transition function,
4. $q_0 \in Q$: start state,
5. $F \subseteq Q$: accept states.

Example



1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ:

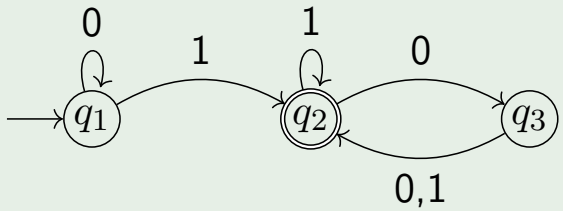
	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. Start state: q_1
5. $F = \{q_2\}$

Finite Automata

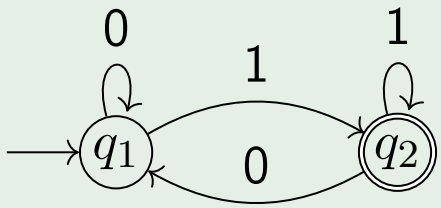
- A: Set of all strings that machine M accepts
  - A is the language of machine M
  - $L(M) = A$
  - M recognizes A
  - M accepts A
- Each machine accepts only one language
  - Can be $\emptyset$

Example



$$A = \{w | w \text{ has at least one } 1 \text{ and} \quad (1)$$
$$\text{even number of zeros follows the last one}\} \quad (2)$$

Example

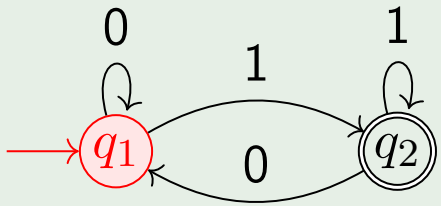


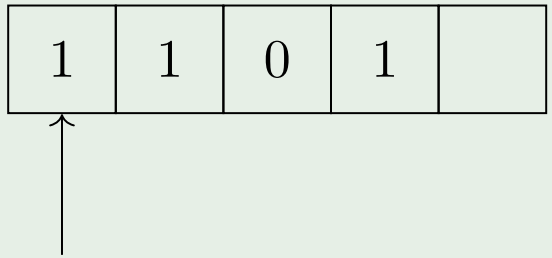
1. $Q = \{q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3. δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

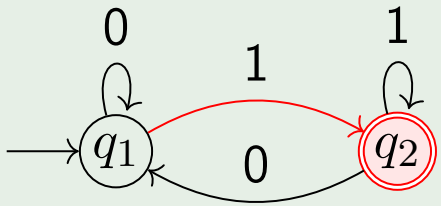
4. Start state: q_1
5. $F = \{q_2\}$

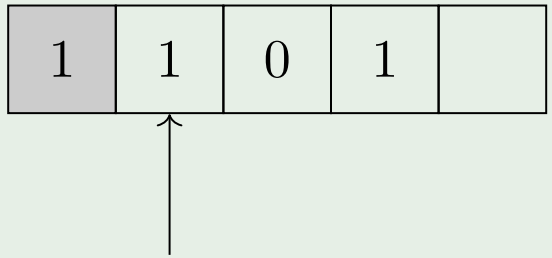
Example (Accept 1101)



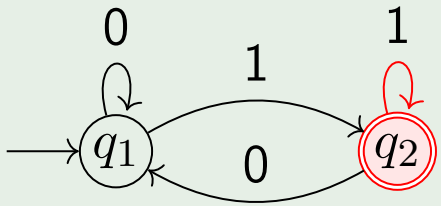


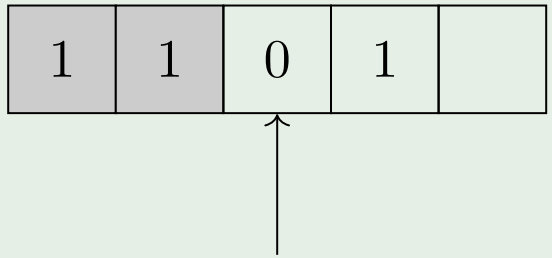
Example (Accept 1101 Cont.)



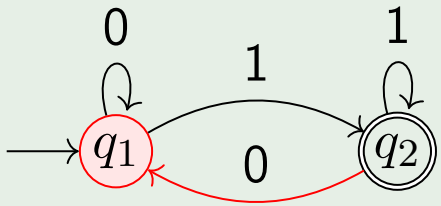


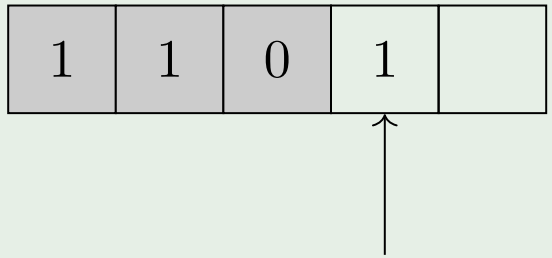
Example (Accept 1101 Cont.)



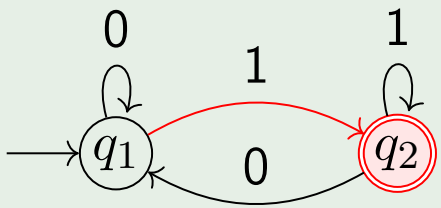


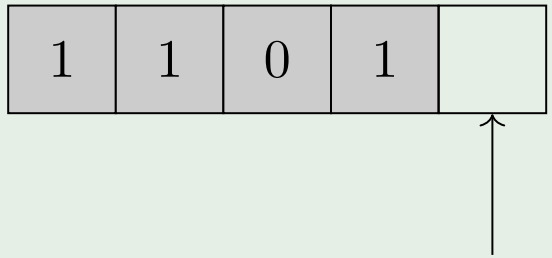
Example (Accept 1101 Cont.)





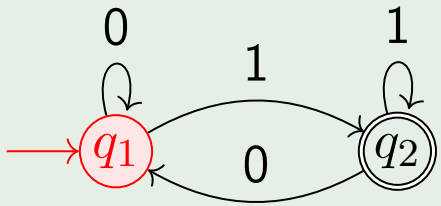
Example (Accept 1101 Cont.)

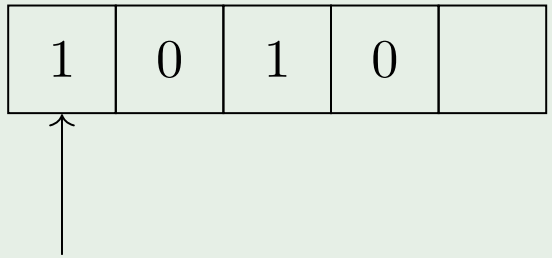




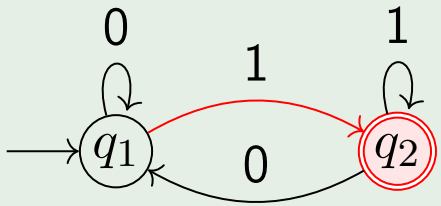
- Output = accept

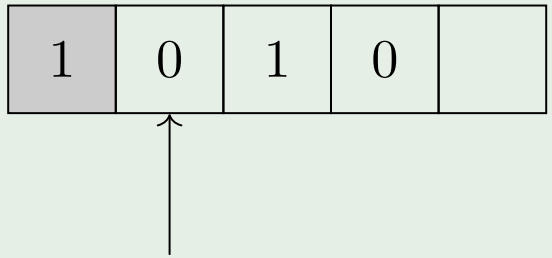
Example (Reject 1010)



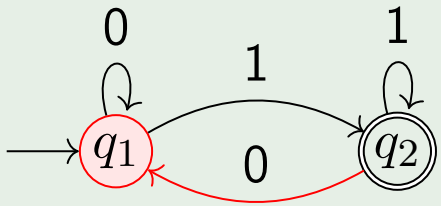


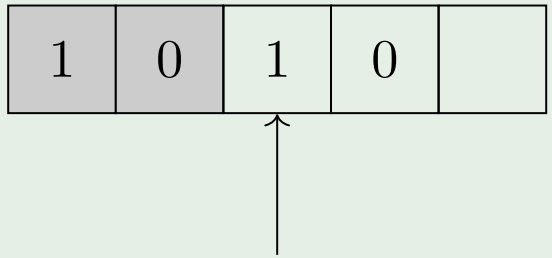
Example (Reject 1010 Cont.)



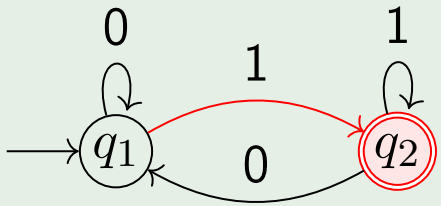


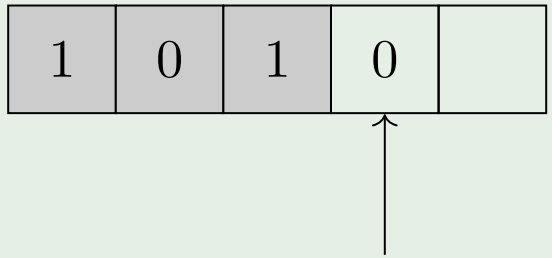
Example (Reject 1010 Cont.)



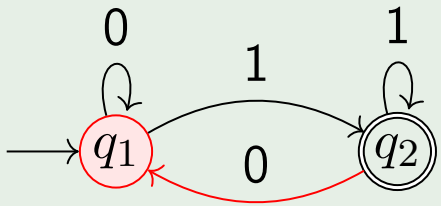


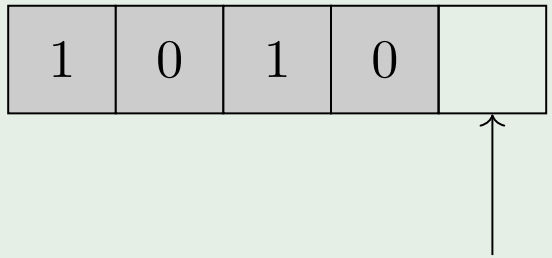
Example (Reject 1010 Cont.)





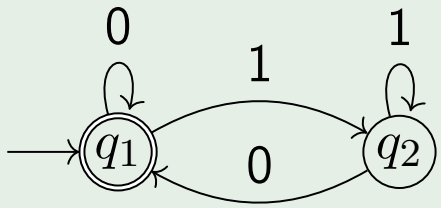
Example (Reject 1010 Cont.)





- Output = reject
- $L(M_2) = \{w | w \text{ ends in a } 1\}$

Example (Language of a DFA)



$$A = \{w | w \text{ is the empty string } \varepsilon \text{ or ends in a zero}\} \tag{3}$$

Formal definition of computation:

- Finite automaton's computation:

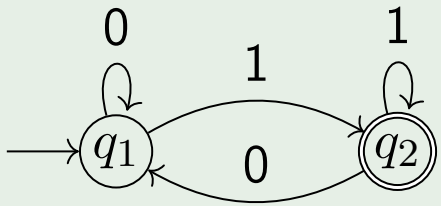
Definition

$M = (Q, \Sigma, \delta, q_0, F)$
$w = w_1 w_2 \dots w_n$
M accepts w if a sequence of states $r_0, \dots, r_n$ exists:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \dots, n-1$, and
3. $r_n \in F$.

- M recognizes A: $A = \{w | M \text{ accepts } w\}$

Example (Reject 1010 Cont.)



Input	States$(r_0, \ldots, r_n)$	Output
ε	q_1	REJECT
1	q_1, q_2	ACCEPT
000	q_1, q_1, q_1, q_1	REJECT
011	q_1, q_1, q_2, q_2	ACCEPT
1110	q_1, q_2, q_2, q_2, q_1	REJECT

Definition (Regular Language)

We call a language regular language if there exists a finite automaton that recognizes it.

Finite Automata

Prove that following languages are regular ($\Sigma = \{0, 1\}$):

- $L_1 = \emptyset$
- $L_2 = \Sigma^*$
- $L_3 = \{w \mid w \text{ starts and ends with } 0\}$
- $L_4 = \{w \mid w \text{ length of } w \text{ is not a multiple of } 3\}$

The Regular Operations

Theory of computation:

- Properties of finite automata and regular languages
- Design automata to recognize some languages
- Prove non-regularity
- Manipulation:
  - Objects: Languages
  - Operations: Manipulate languages

Definition (Regular Operations)

A and B are regular languages:

- Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
- Concatenation: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$
- Star: $A^* = \{x_1 x_2 \dots x_k | k \geq 0 \text{ and } x_i \in A\}$

The Regular Operations

Example

$\Sigma = \{a, b, \ldots, z\}$

$A =$ {good, bad}, $B =$ {boy, girl}

- $A \cup B =$ {good, bad, boy, girl}
- $A \circ B =$ {goodboy, goodgirl, badboy, badgirl}
- $A^* =$ {ε, good, bad, goodgood, goodbad
  badbad, badgood, goodgoodgood, goodgoodbad}

The Regular Operations

- Closed under some operation: The result is in the collection of inputs
- Example:
  - Multiplication
  - Division
- Collection of regular languages is closed under:
  - Complement
  - Union
  - Concatenation
  - Star

The Regular Operations

Theorem

The complement of a regular language is a regular languages.

Proof idea:

- Consider a regular language A,
- Since A is regular, there exists a DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes A,
- Construct a new DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that recognizes $\overline{A}$,
- M_1 and M_2 consume a string similarly, however, when M_1 accepts a string M_2 rejects it and vice versa.

The Regular Operations

Proof.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be a DFA such that $L(M_1) = A$.
Construct $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$:

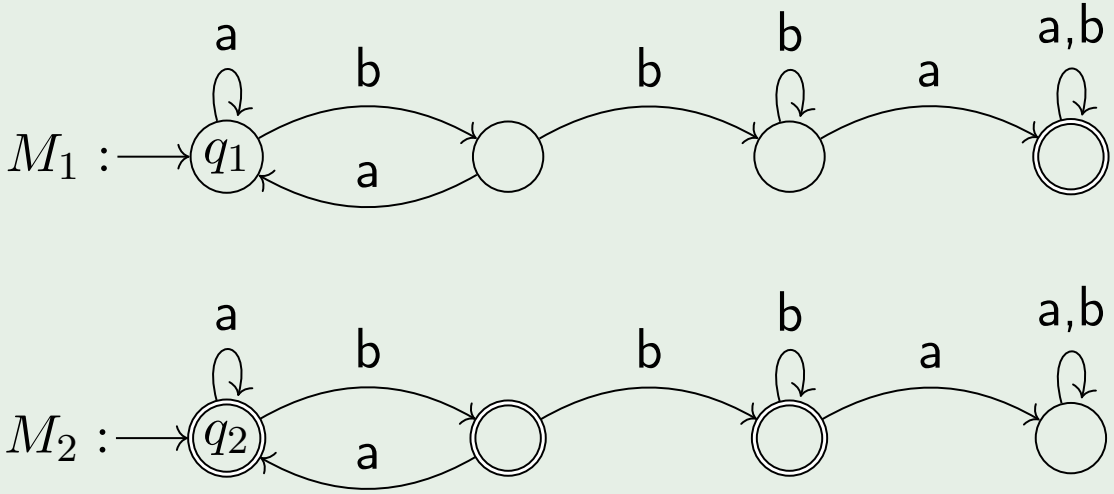
1. $Q_2 = Q_1$
2. Σ
3. $\delta_2 = \delta_1$
4. $q_2 = q_1$
5. $F_2 = Q_1 - F_1$.

□

- Formal correctness proof?

The Regular Operations

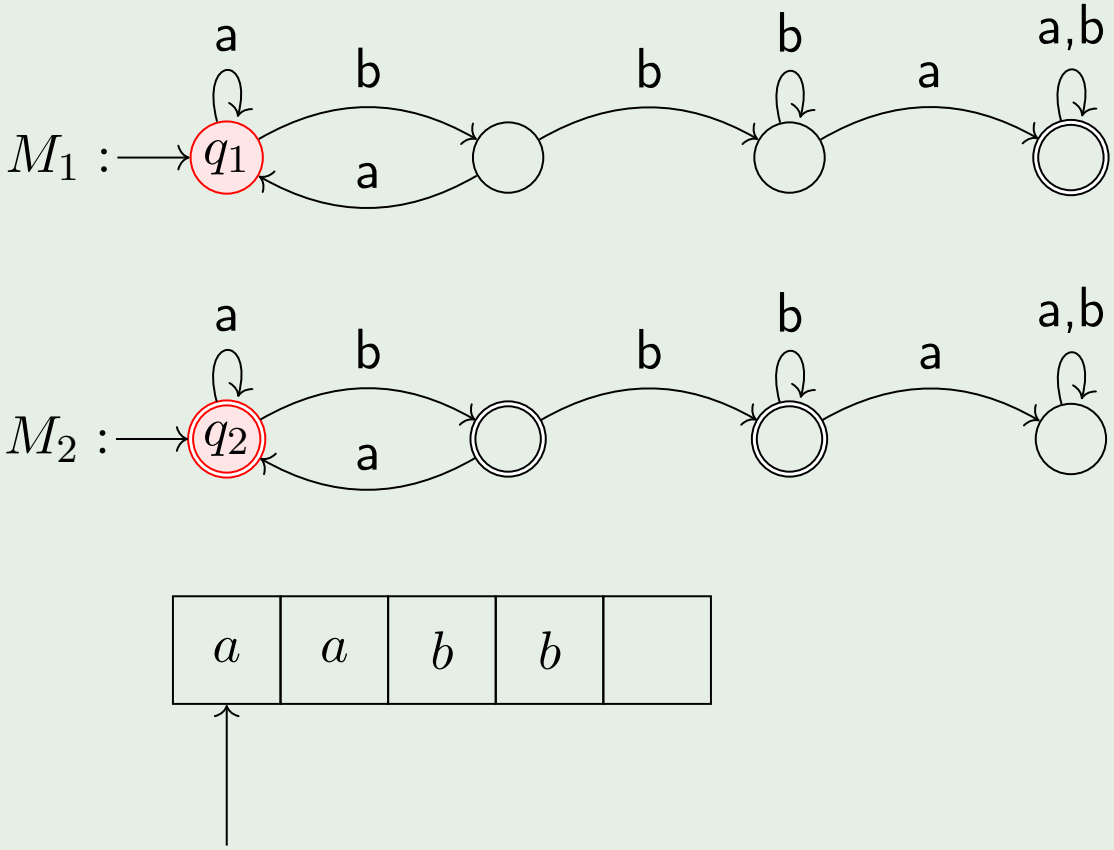
Example (Substring Complement)



$L(M_1) = \{w \mid bba \text{ is a substring of } w\}$
$L(M_2) = \{w \mid bba \text{ is not a substring of } w\}$

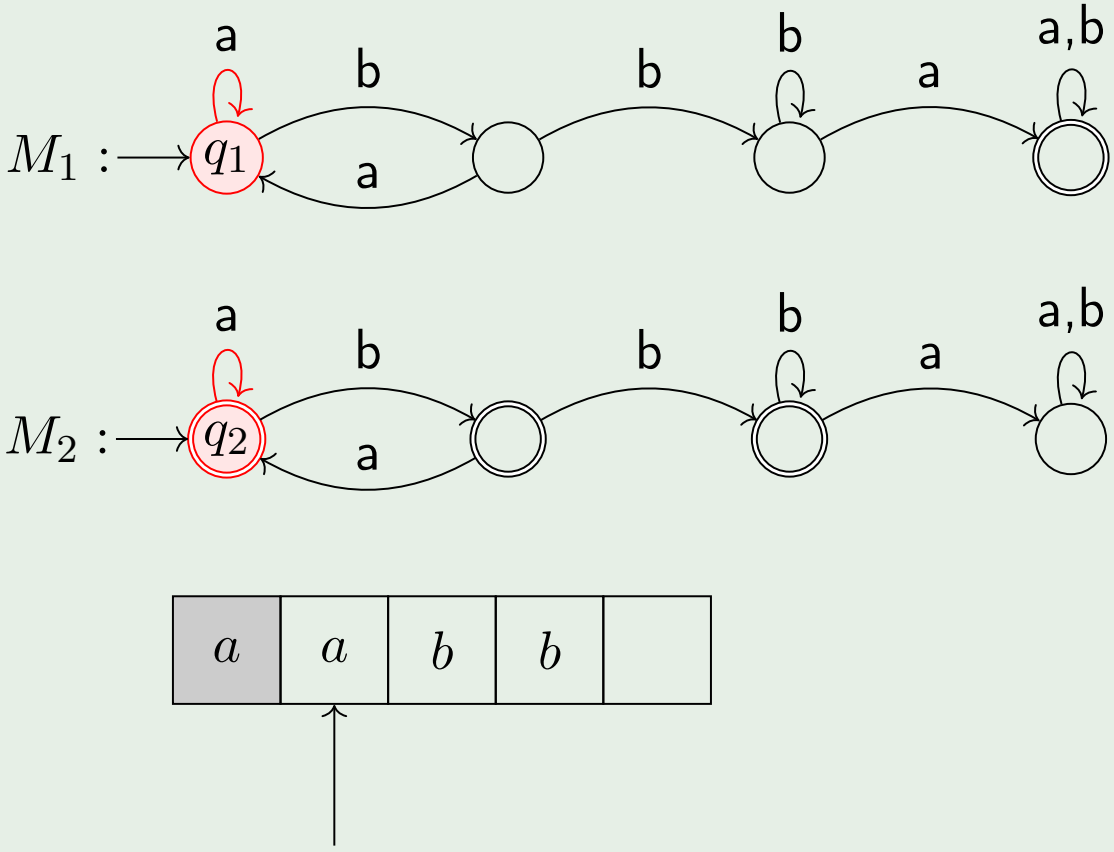
The Regular Operations

Example (Input=aabb)



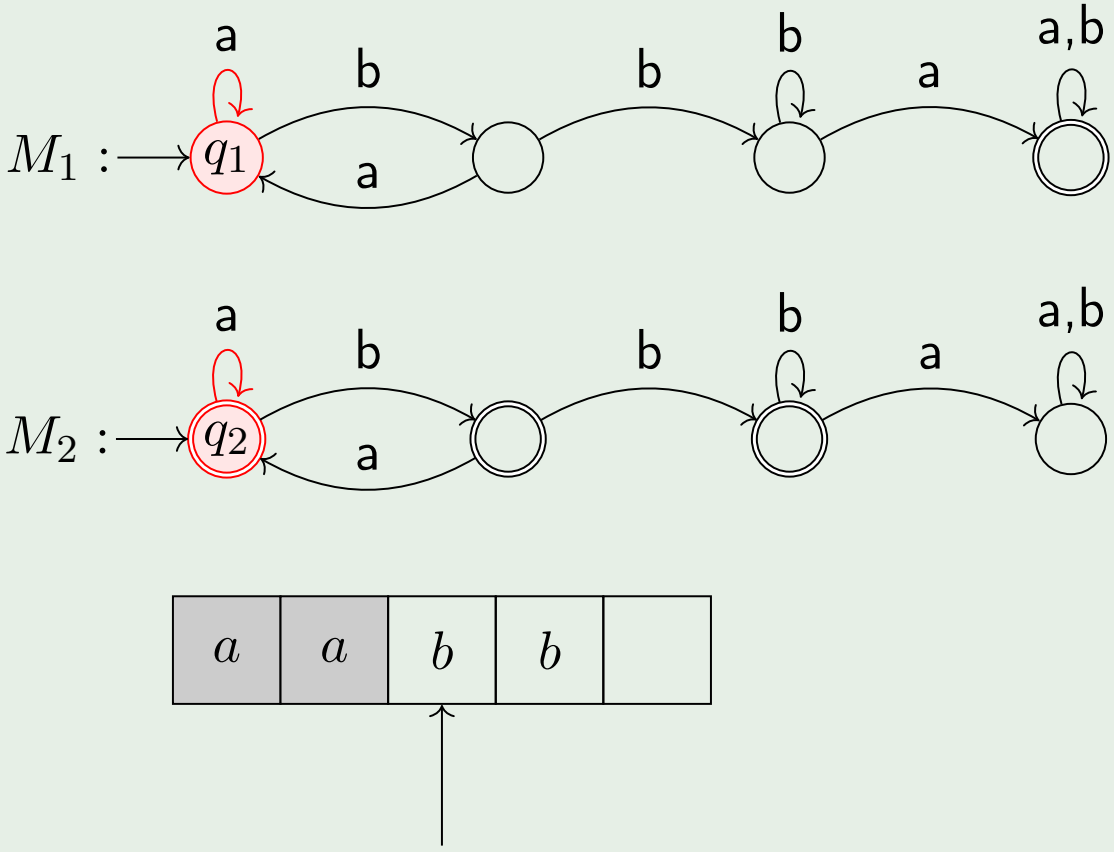
The Regular Operations

Example (Input=aabb Cont.)



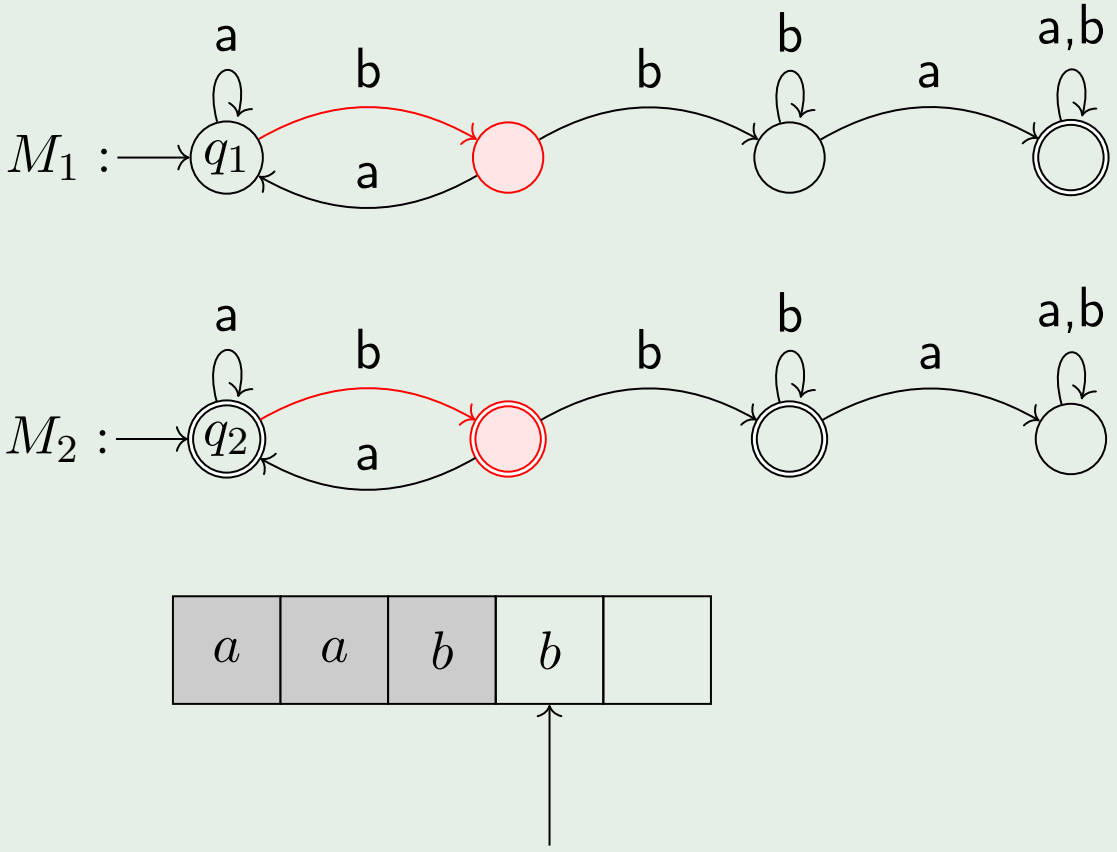
The Regular Operations

Example (Input=aabb Cont.)



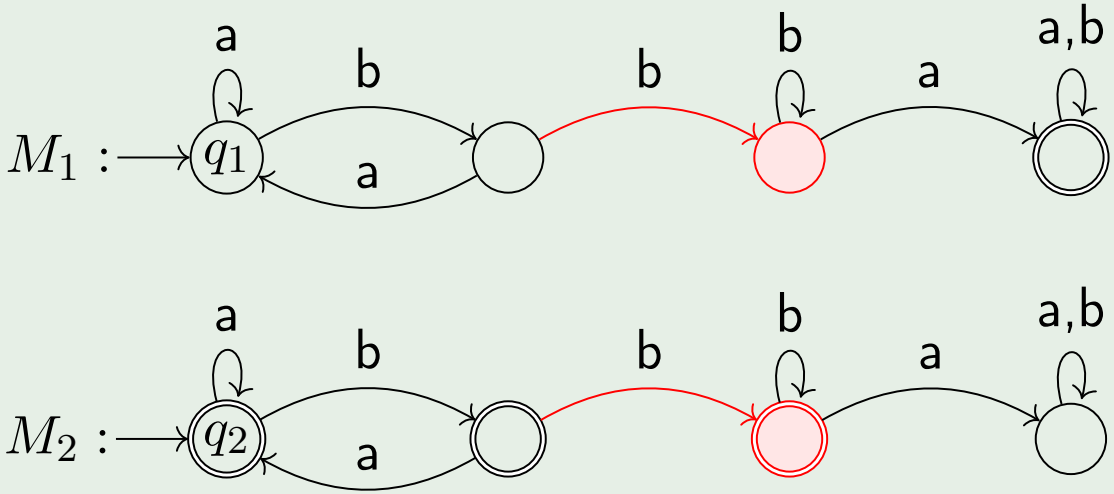
The Regular Operations

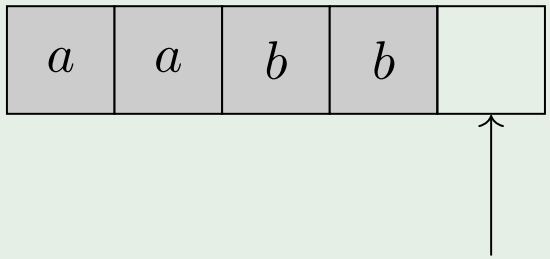
Example (Input=aabb Cont.)



The Regular Operations

Example (Input=aabb Cont.)





Output of M_1 = reject

Output of M_2 = accept

The Regular Operations

Theorem

The class of regular languages is closed under the union operation. In other words, A_1 and A_2 are regular $\rightarrow$ $A_1 \cup A_2$ is regular.

Proof idea:

- There exists a finite automaton M_1 that recognizes A_1
- There exists a finite automaton M_2 that recognizes A_2
- We construct a finite automaton M that recognizes $A_1 \cup A_2$
- Can M simulate M_1 and then M_2 on the input and accept if either of simulations accept?
  - We can't rewind the input tape!
- Simulate the machines simultaneously
  - Keep track of both machines
  - Possible with finite memory

The Regular Operations

Proof.

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
Construct $M_1 = (Q, \Sigma, \delta, q_0, F)$:

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. Σ
3. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
4. $q_0 = (q_1, q_2)$
5. $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}$

□

- Formal correctness proof?

The Regular Operations

- Prove the closure under concatenation?
- Break the string into two pieces that are accepted by M_1 and M_2, respectively
- Where to break the string?
- New technique: Nondeterminism