Please check your attendance using Blackboard!

Lecture 2

Finite Automata

COSE215: Theory of Computation

Seunghoon Woo

Fall 2023

Contents

- **Finite Automata**
  - Deterministic Finite Automata (DFA)
  - Nondeterministic Finite Automata (NFA)

Automata

Learning Objectives

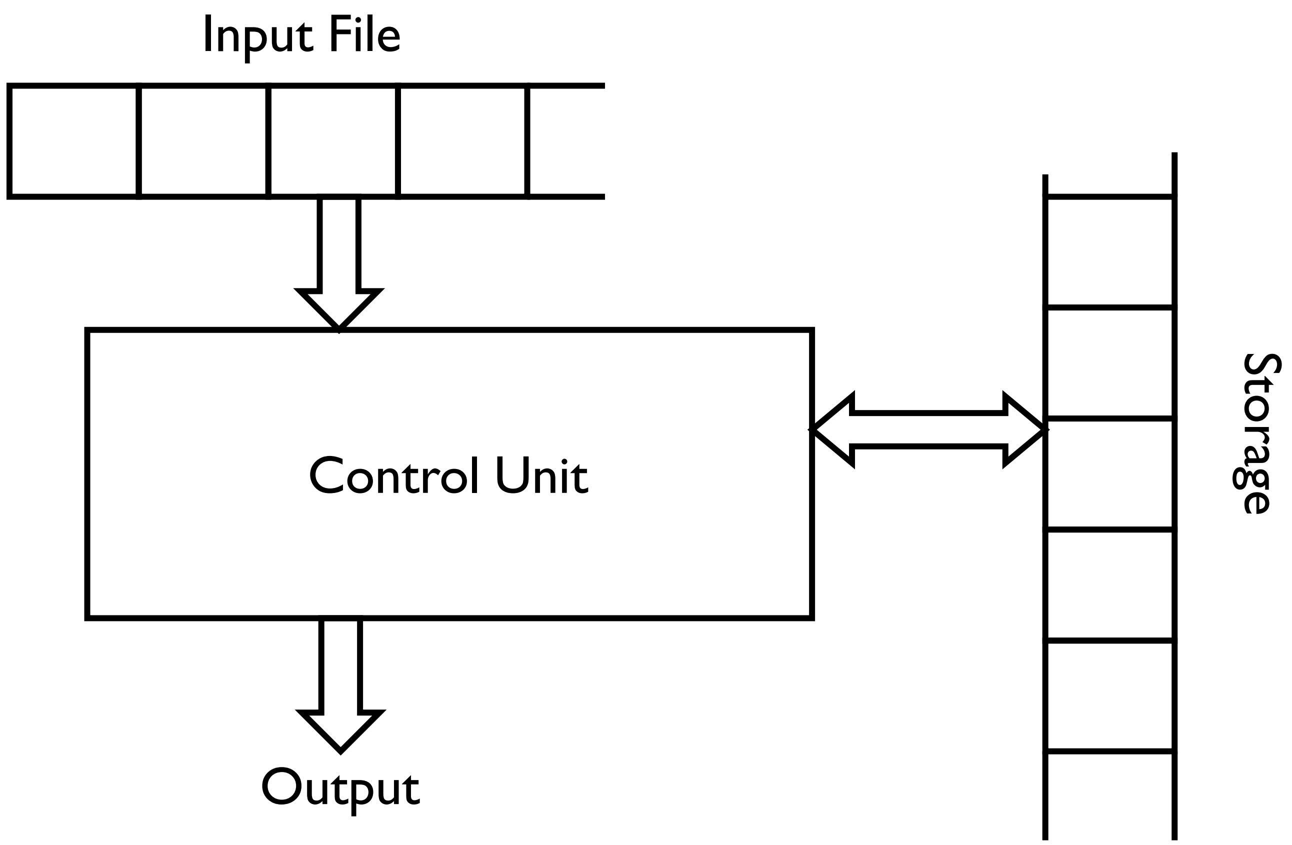
- **Why do we study the Theory of Computation?**
  1. Theory of Computation is a field that deals with theoretical considerations on the principles of operation and computational possibilities of computers
     - ❖ What can computers do?
  2. This helps to develop the ability to model a given problem and the core of all computers and their applications
     - ❖ Model math problems into a form that computers can understand
  3. The ideas we will discuss have some immediate and important applications (e.g., programming languages, compilers, operating systems, security, and AI)

Automata

- **An automaton**
  - An abstract model of a digital computer
- **Every automaton includes some essential features**
  - Reading input (a string over an alphabet)
    - Automaton can read it but not change
  - Producing output
  - Containing a temporal storage
  - Containing a control unit (with a finite number of internal states)

Automata

- **An automaton**

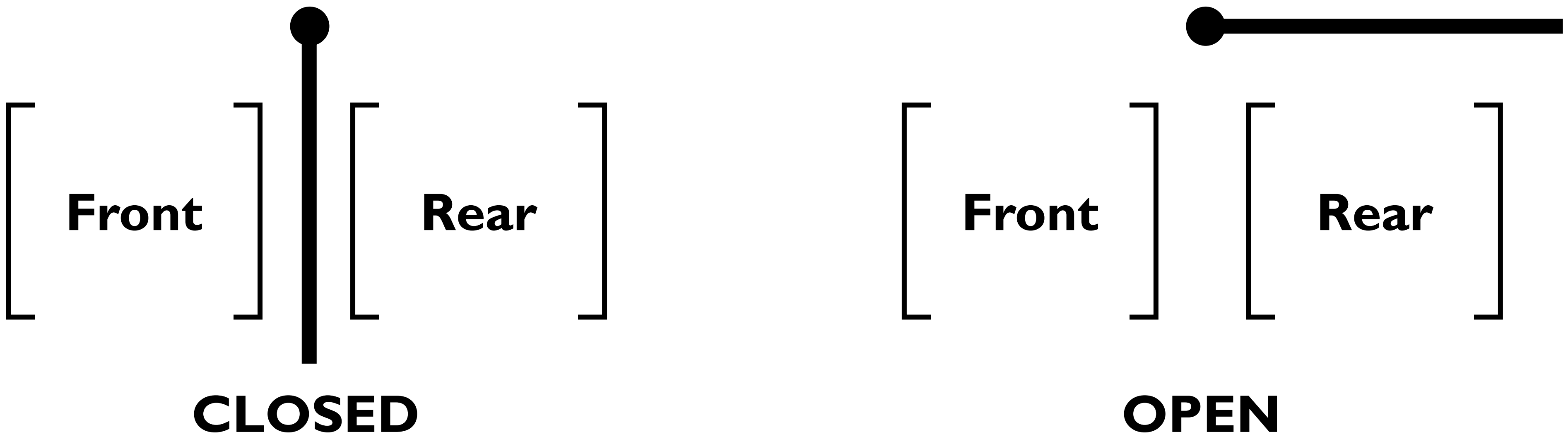


Finite Automata

- **The simplest model: finite automata (finite state machines)**
  - A finite set of internal states (with no other memory)
  - Finite automata can be used in many fields
    - Security, compiler, network protocol, etc.

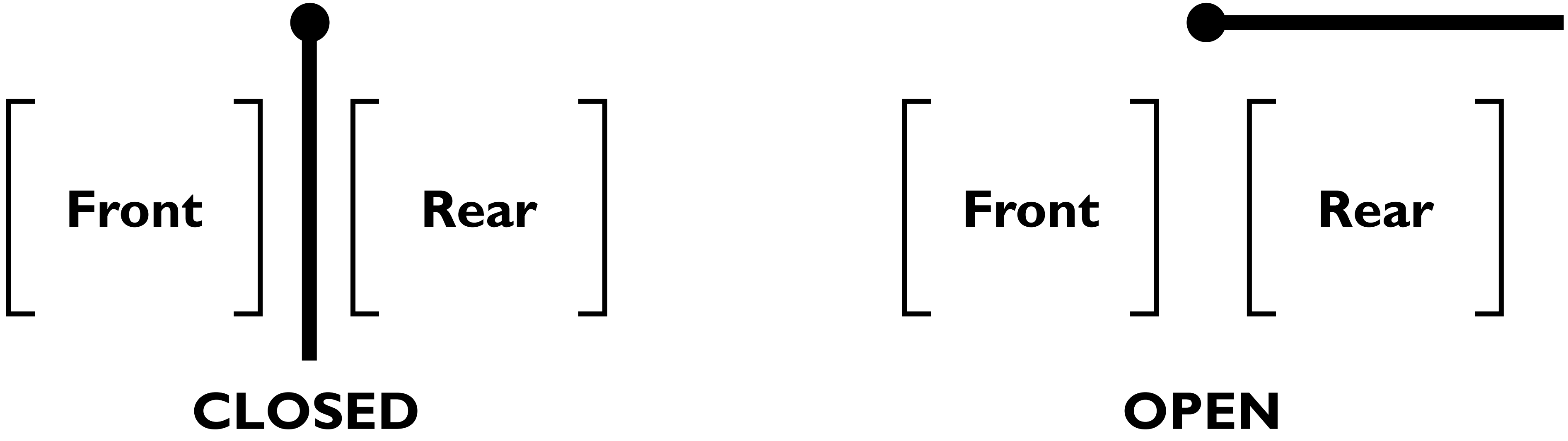
Finite Automata

- **The simplest model: finite state machines (finite automata)**
  - Example: a controller for automatic door



Finite Automata

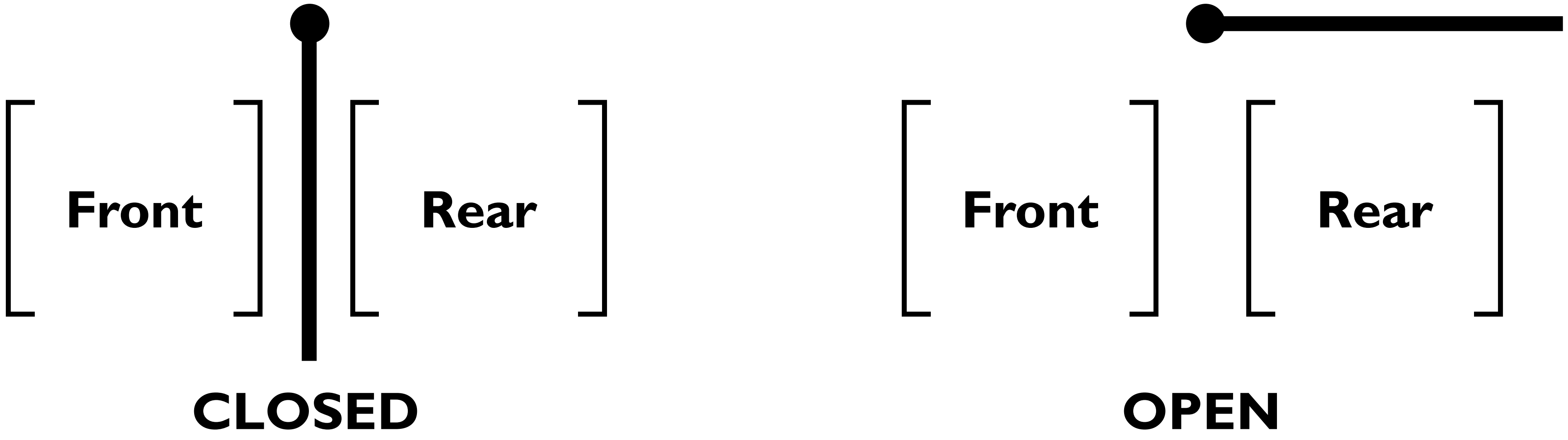
- **The simplest model: finite state machines (finite automata)**
  - Example: a controller for automatic door
    - ❖ If a person is on the **Front,** the door should open
    - ❖ It should remain open long enough to pass all the way through
    - ❖ The door should not strike some standing behind it! (**Rear**)



Finite Automata

- **The simplest model: finite state machines (finite automata)**
  - Example: a controller for automatic door
    - State transition table

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

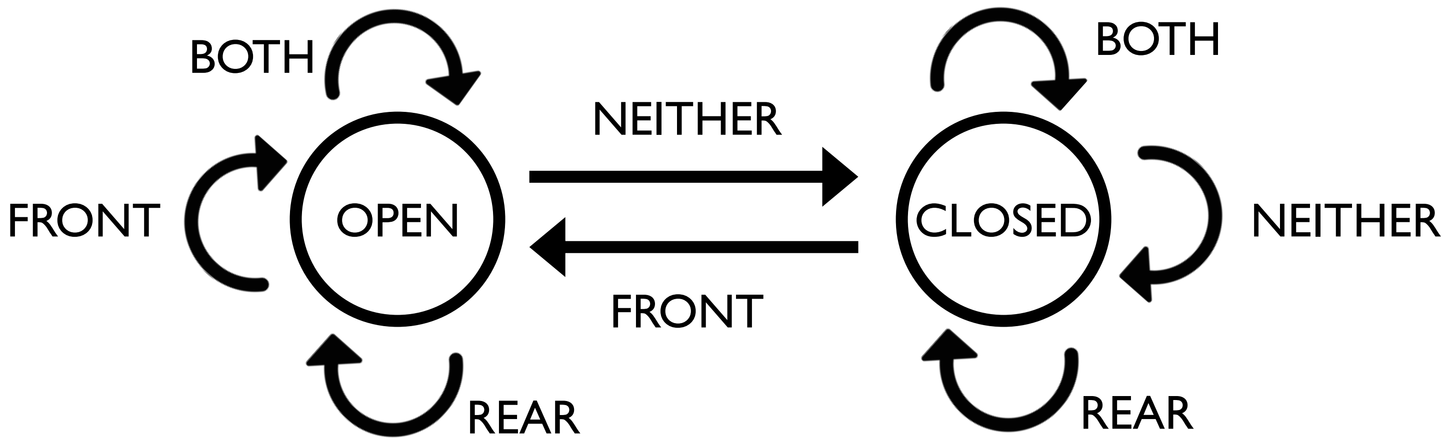


Finite Automata

- **The simplest model: finite state machines (finite automata)**
  - Example: a controller for automatic door
    - ❖ State transition table

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

    - ❖ State transition graph

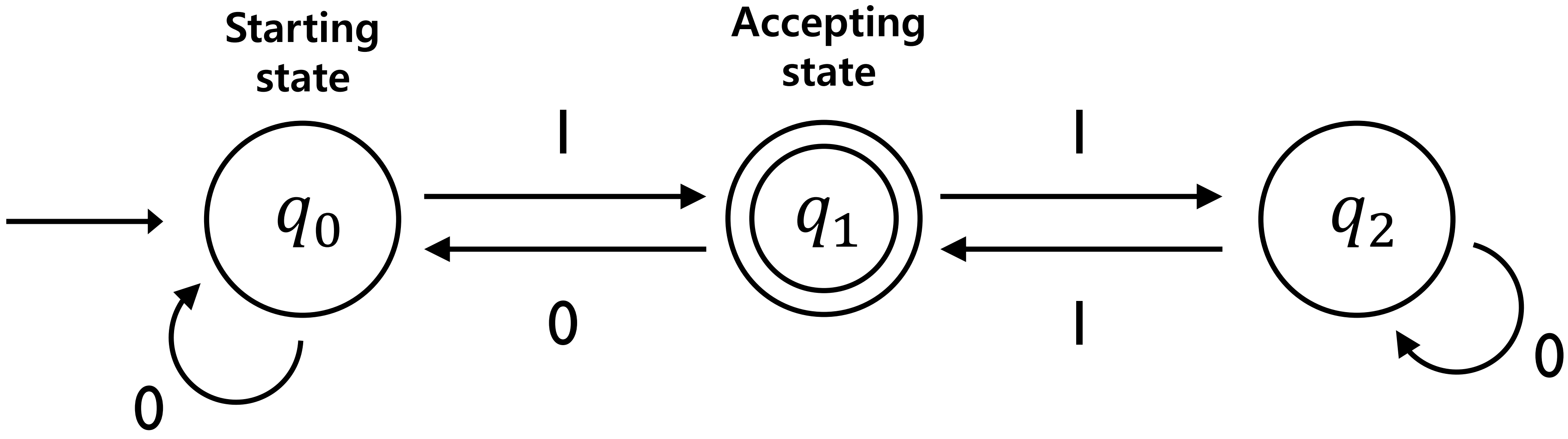


Deterministic Finite Automata (DFA)

- **DFA**
  - Containing a finite number of internal states
    - Including a **starting (initial) state** and **final (accepting) states**
  - Processing an input string, consisting of a sequence of symbols
  - Making transitions for one state to another
    - Depending on the <u>current state</u> and <u>input symbol</u>
  - Producing output
    - Accept or Reject

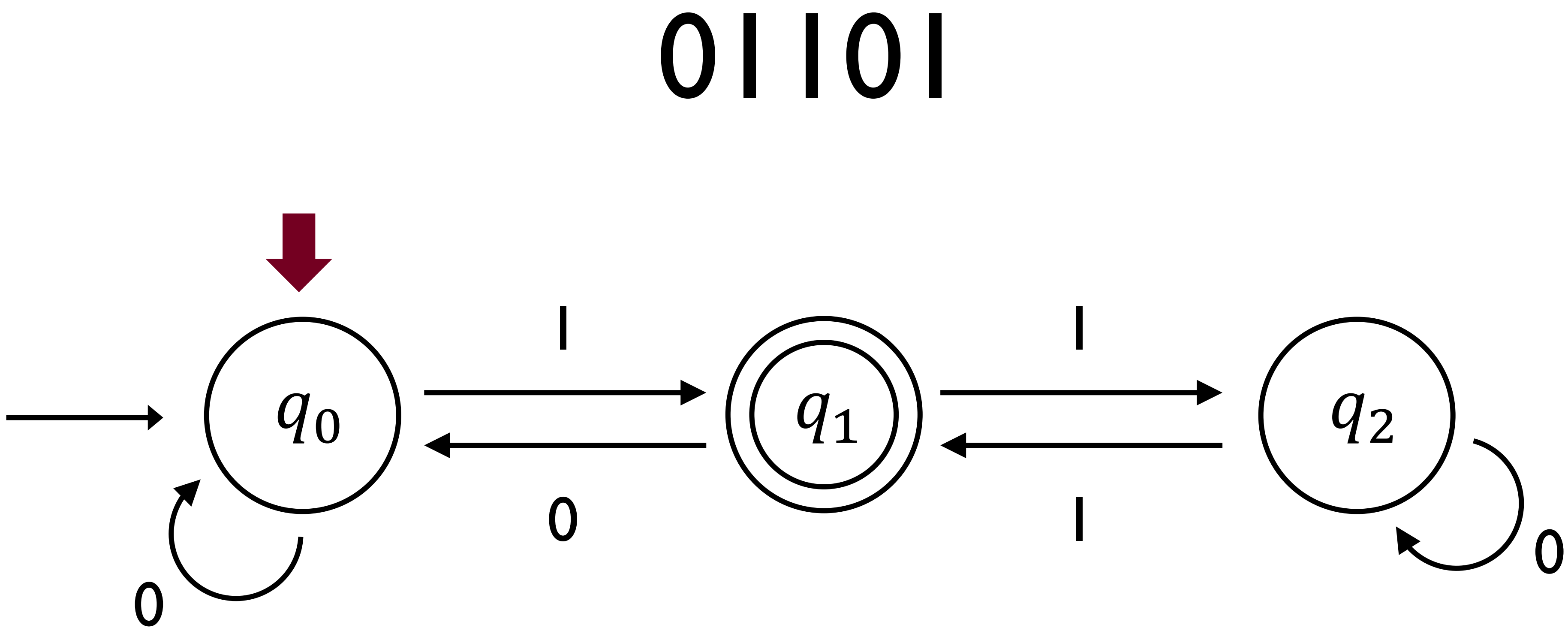
Deterministic Finite Automata (DFA)

- **Example**



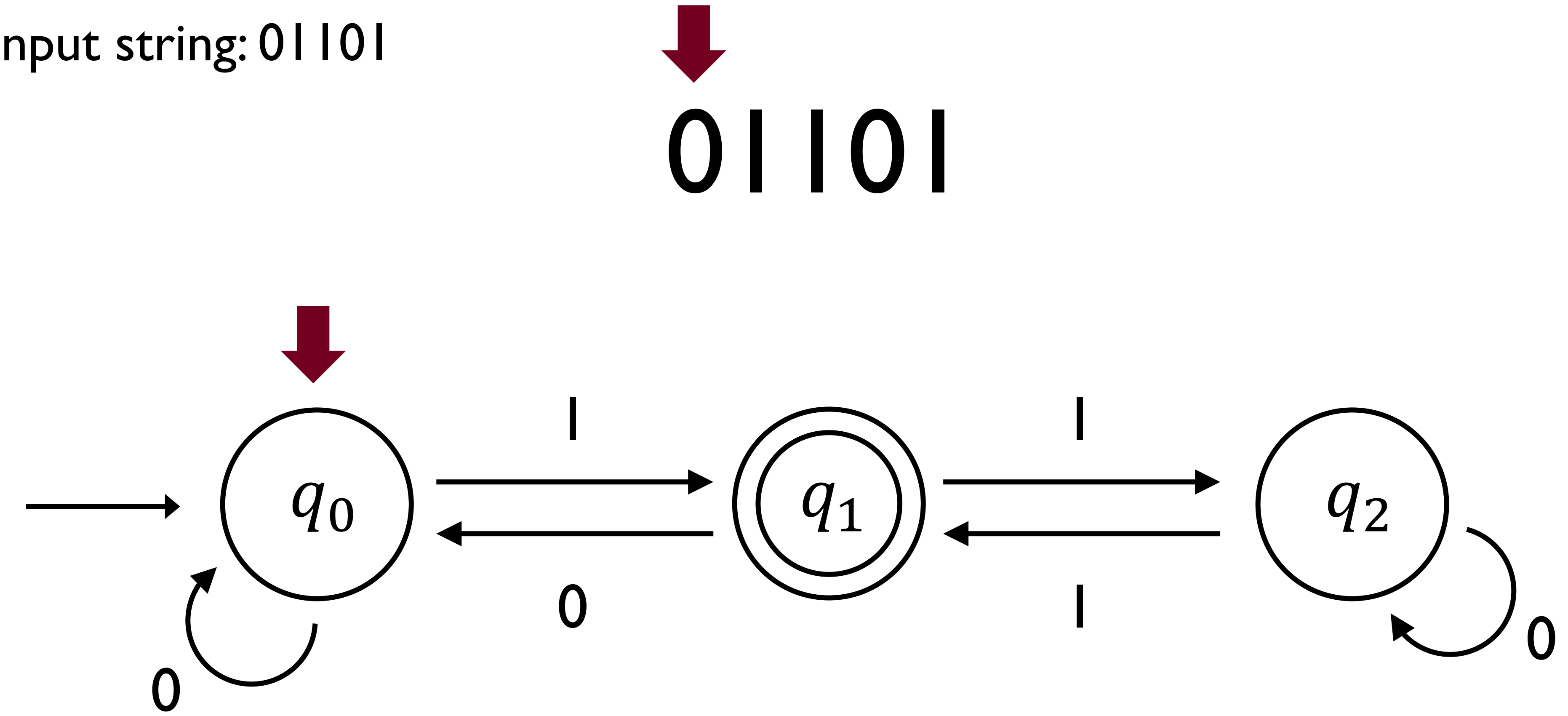
Deterministic Finite Automata (DFA)

- **Example**
  - Input string: 01101



Deterministic Finite Automata (DFA)

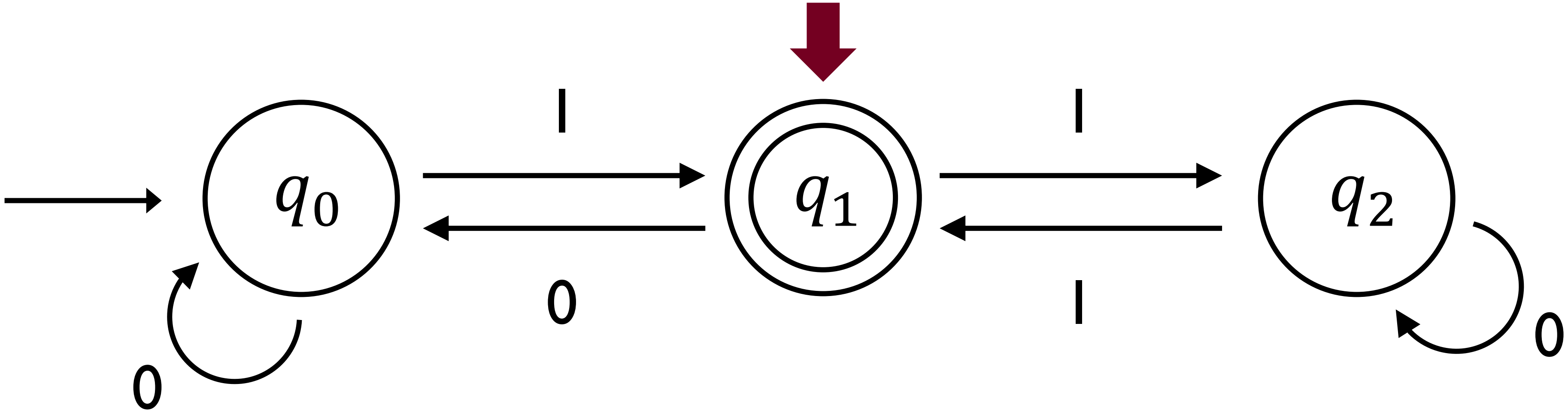
- **Example**
  - Input string: 01101



Deterministic Finite Automata (DFA)

- **Example**
  - Input string: 01101

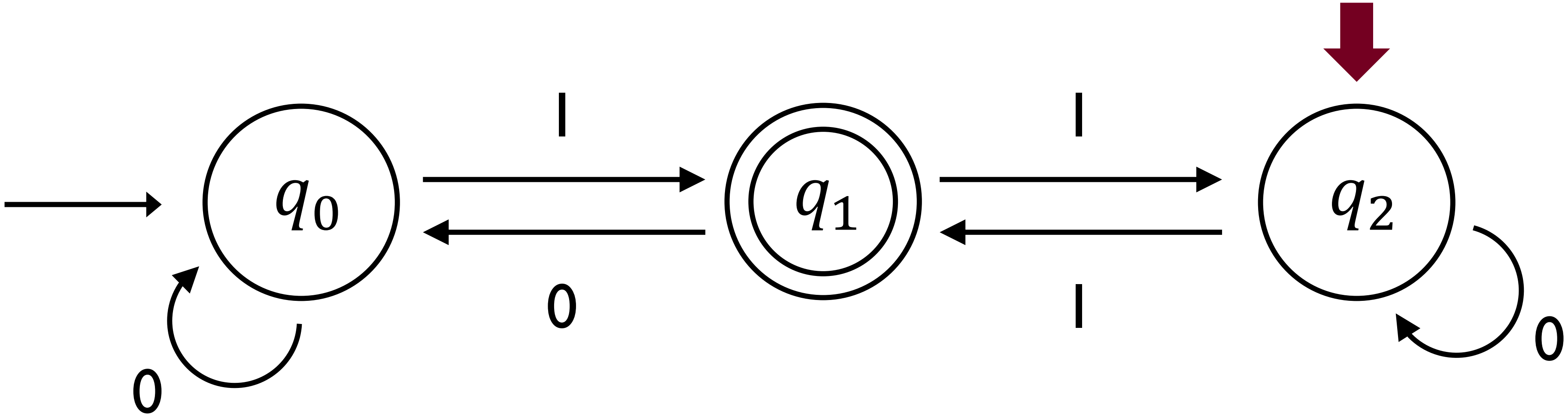
~~0~~1101



Deterministic Finite Automata (DFA)

- **Example**
  - Input string: 01101

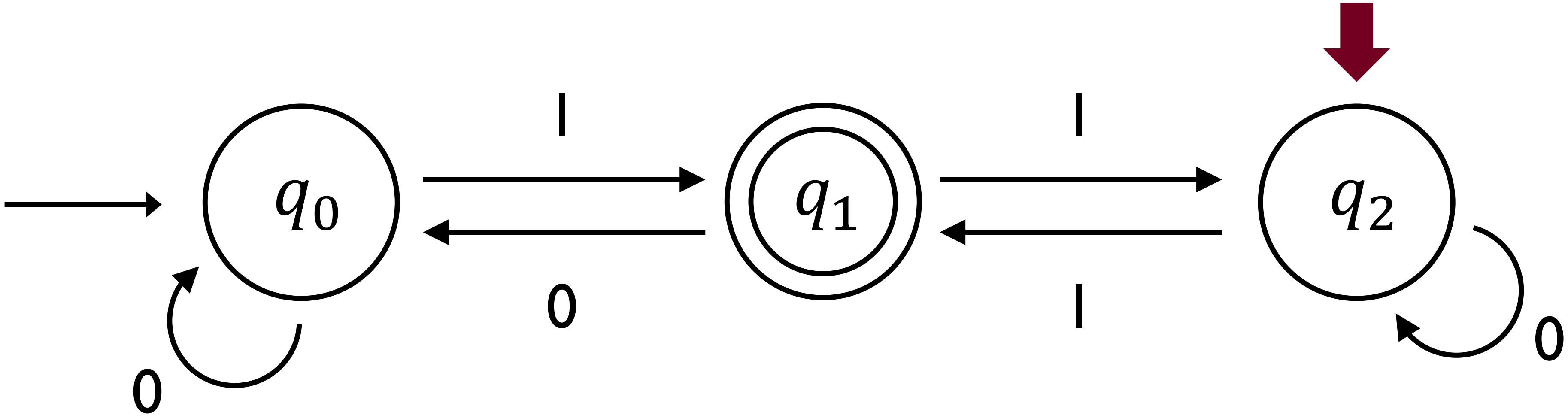
~~01~~101



Deterministic Finite Automata (DFA)

- **Example**
  - Input string: 01101

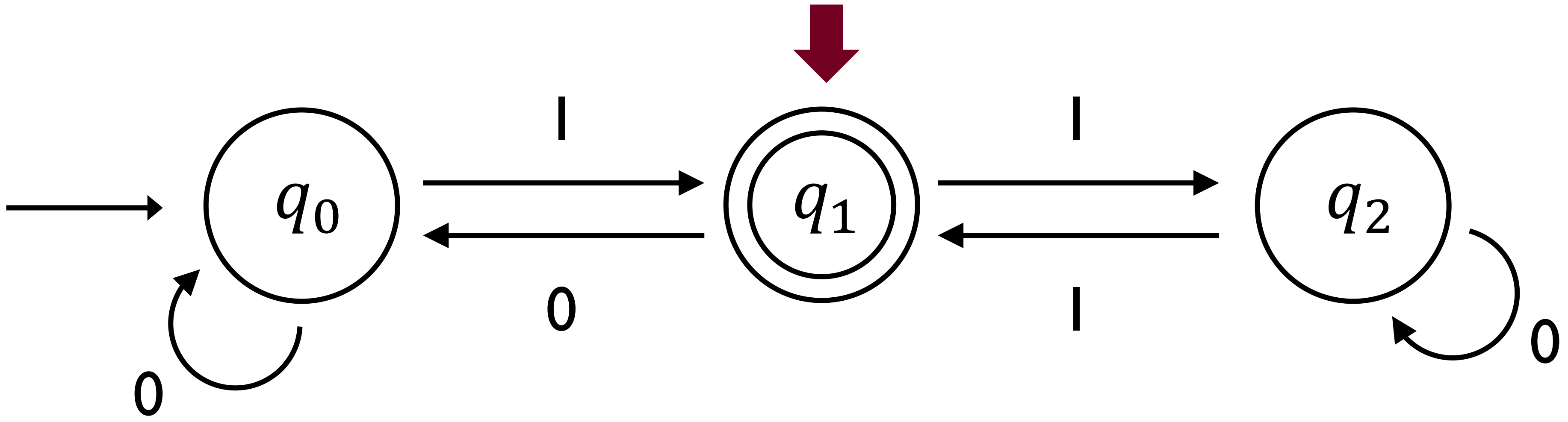
~~011~~01



Deterministic Finite Automata (DFA)

- **Example**
  - Input string: 01101

~~0110~~1 :ACCEPT!



Deterministic Finite Automata (DFA)

- **Definition of DFA**
  - A DFA is defined by 5-tuples

$$M = (Q, \Sigma, \delta, q_0, F)$$

❖ Q is a finite set of **internal states**

❖ Σ is a finite set of **symbols**

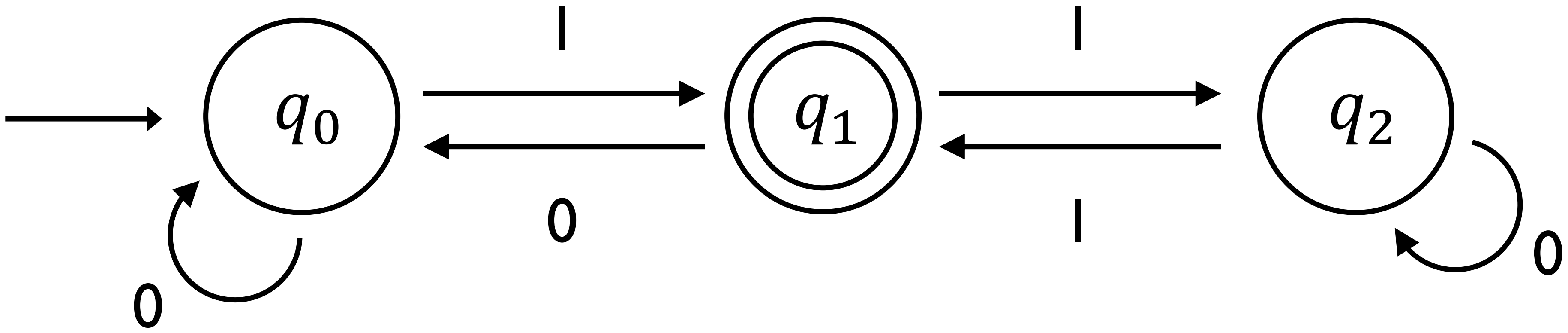
❖ $\delta: Q \times \Sigma \to \mathrm{Q}$ is the **transition function**

- **Every state must have a transition for every symbol**

❖ q_0 is the **initial state** ($q_0 \in Q$)

❖ F is a set of **final states** ($F \subseteq Q$)

Deterministic Finite Automata (DFA)



$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

$\delta(q_0, 0) = q_0$	$\delta(q_1, 0) = q_0$	$\delta(q_2, 0) = q_2$
$\delta(q_0, 1) = q_1$	$\delta(q_1, 1) = q_2$	$\delta(q_2, 1) = q_1$

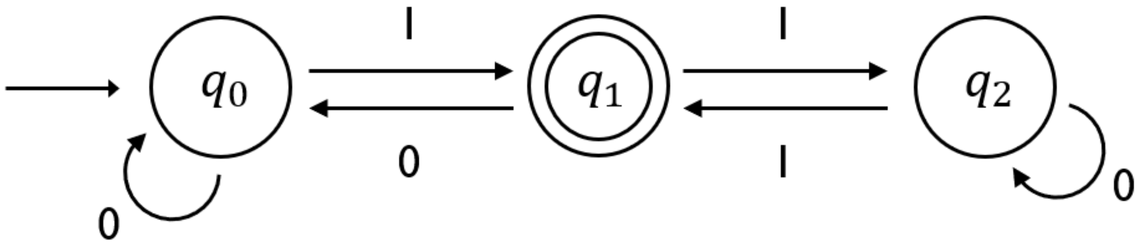
Deterministic Finite Automata (DFA)

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

$\delta(q_0, 0) = q_0$ $\quad$ $\delta(q_1, 0) = q_0$ $\quad$ $\delta(q_2, 0) = q_2$

$\delta(q_0, 1) = q_1$ $\quad$ $\delta(q_1, 1) = q_2$ $\quad$ $\delta(q_2, 1) = q_1$

Transition Graph

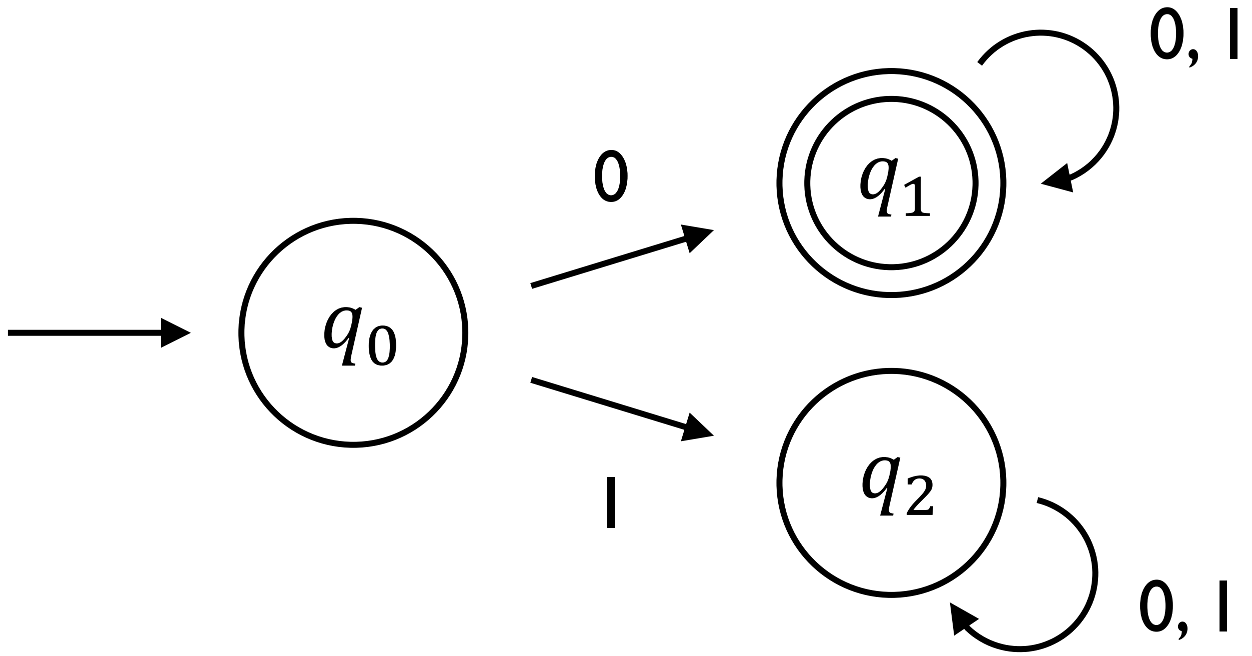


Transition Table

q	0	1
$\rightarrow q_0$	q_0	q_1
$* \, q_1$	q_0	q_2
q_2	q_2	q_1

Deterministic Finite Automata (DFA)

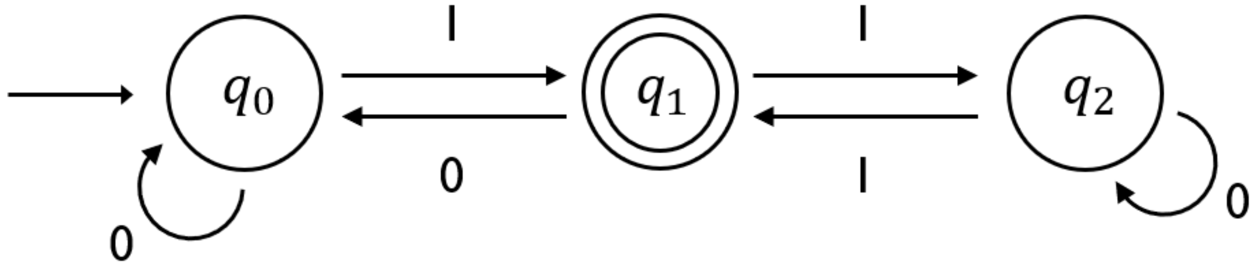
- **Trap state**
  - Trap states in a transition graph



TRAP STATE

Deterministic Finite Automata (DFA)

- **Extended transition function**
  - $\delta^*: Q \times \Sigma^* \rightarrow Q$
  - Connection of multiple transition functions
  - $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$
  - Example
    - ❖ $\delta(q_0, 1) = q_1$ and $\delta(q_1, 1) = q_2$
    - ❖ Then, $\delta^*(q_0, 11) = q_2$



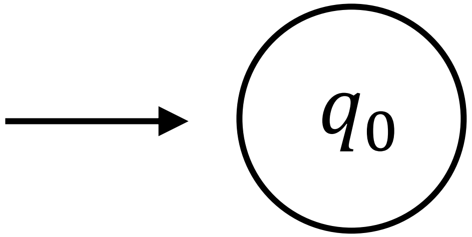
$\delta^*(q_0, 100) = ?$

Deterministic Finite Automata (DFA)

- **Example**
  - Find a DFA that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab
    - ab, abb, ababa, abbaaa, abaaa => Accepted
    - aab, ba, bbba, baabaaa, aabbb => Rejected

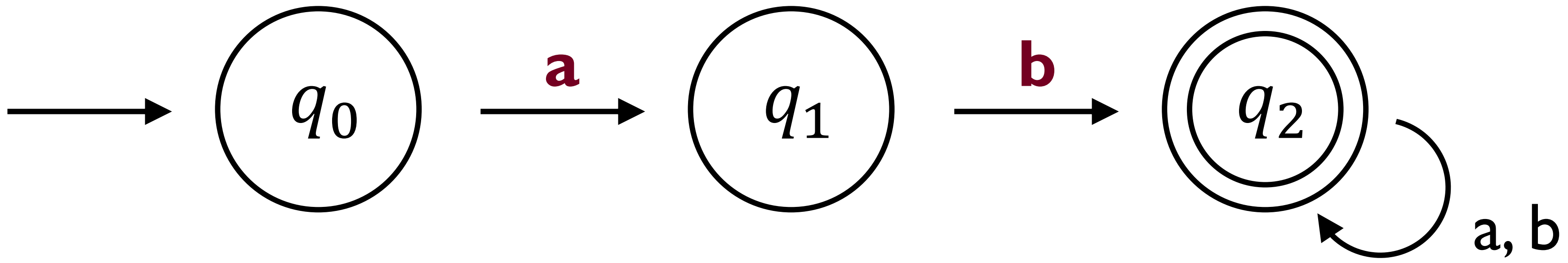
Deterministic Finite Automata (DFA)

- **Example**
  - Find a DFA that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab
    - ab, abb, ababa, abbaaa, abaaa => Accepted
    - aab, ba, bbba, baabaaa, aabbb => Rejected



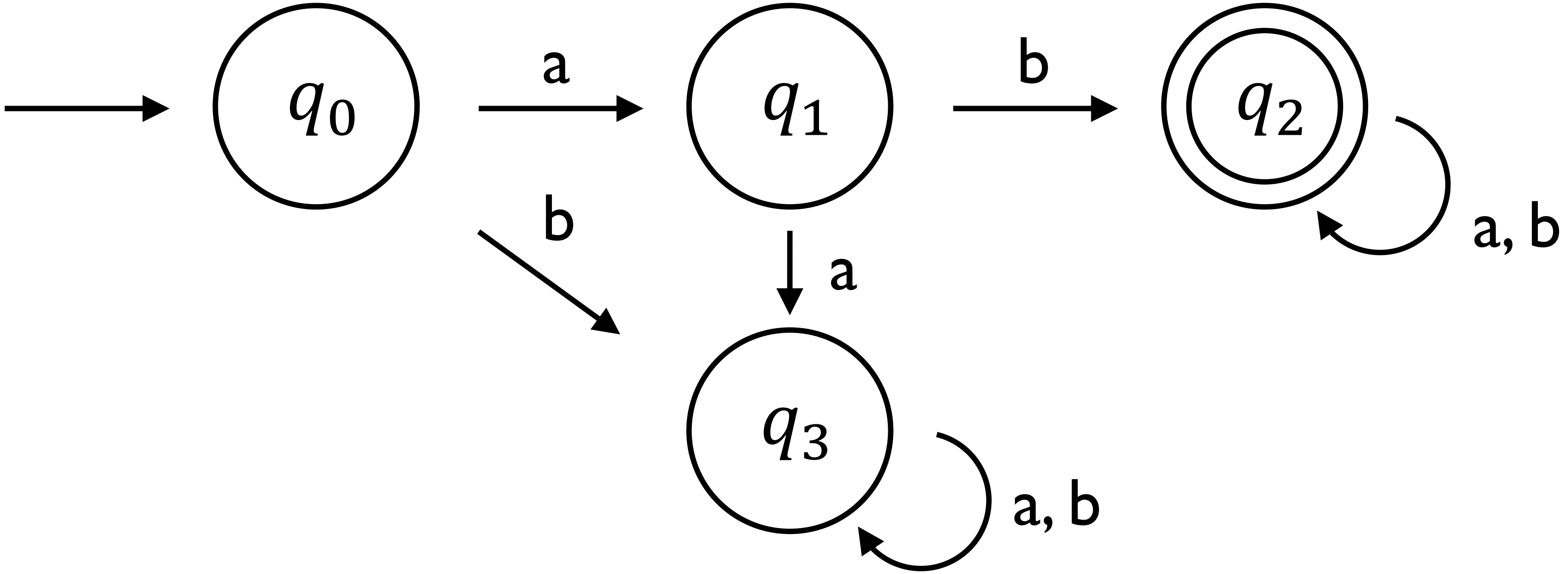
Deterministic Finite Automata (DFA)

- **Example**
  - Find a DFA that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab
    - ❖ ab, abb, ababa, abbaaa, abaaa => Accepted
    - ❖ aab, ba, bbba, baabaaa, aabbb => Rejected



Deterministic Finite Automata (DFA)

- **Example**
  - Find a DFA that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab
    - ab, abb, ababa, abbaaa, abaaa => Accepted
    - aab, ba, bbba, baabaaa, aabbb => Rejected



Deterministic Finite Automata (DFA)

- **Practice**
  - Design a DFA for the language that contains only binary strings (i.e., $\Sigma = \{0, 1\}$) whose bits sum to a multiple of 3
    - ❖ 0, 111, 1011, 1001010111 => Accepted
    - ❖ 1, 101, 1111, 1110000001 => Rejected

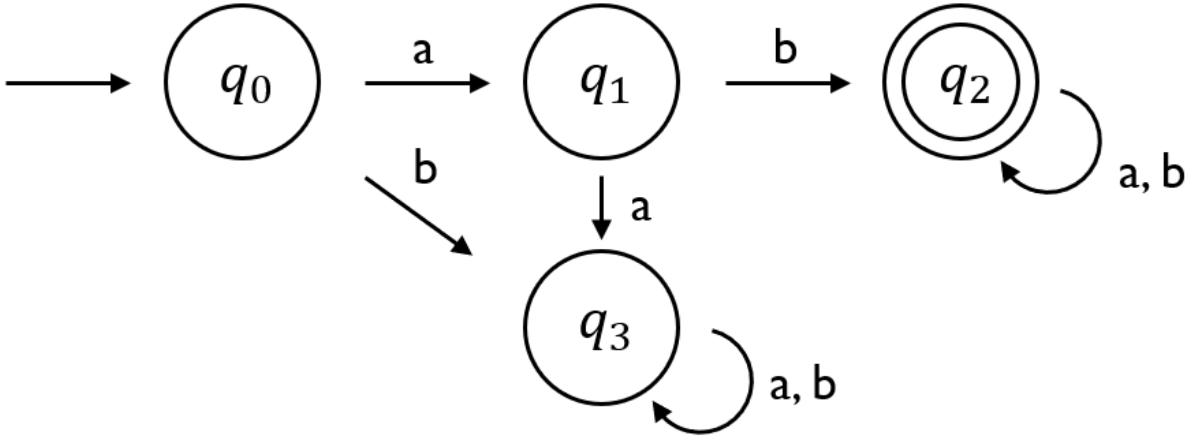
Deterministic Finite Automata (DFA)

- **Acceptance of a language**
  - The language accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$
    => The set of all strings on Σ accepted by M

$$L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \in F\}$$

Deterministic Finite Automata (DFA)

- **Acceptance of a language**
  - ab, abb, ababa, abbaaa, abaaa, $\ldots \in L(M)$
  - aab, ba, bbba, baabaaa, aabbb, $\ldots \notin L(M)$



$$L(M) = \{abw \mid w \in \{a, b\}^*\}$$

Regular Languages

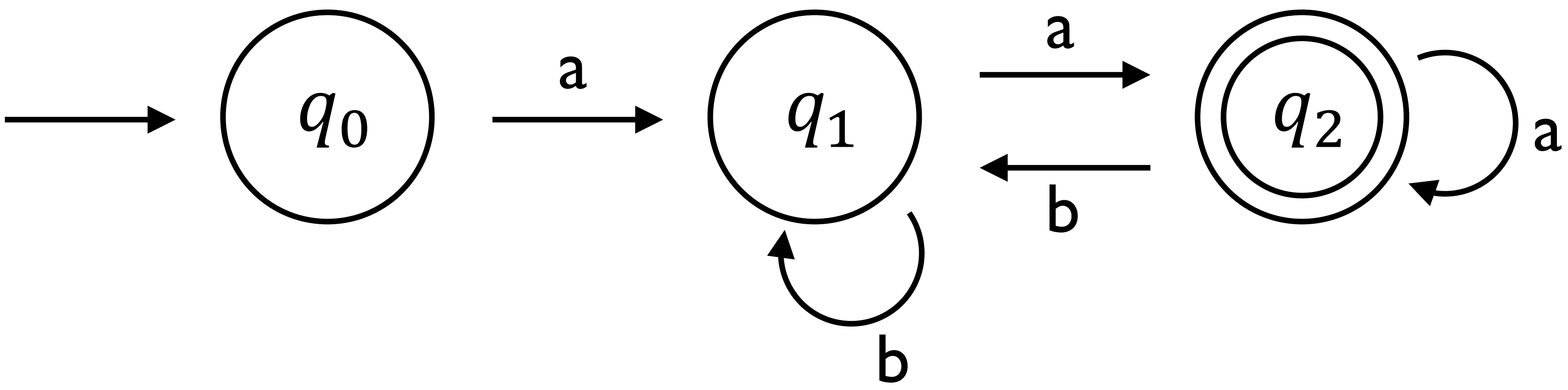
- **Regular language**
  - A language L is called regular if and only if there exists a DFA M such that

$$L = L(M)$$

  - Example
    - Show that the language $L = \{awa : w \in \{a, b\}^*\}$ is regular

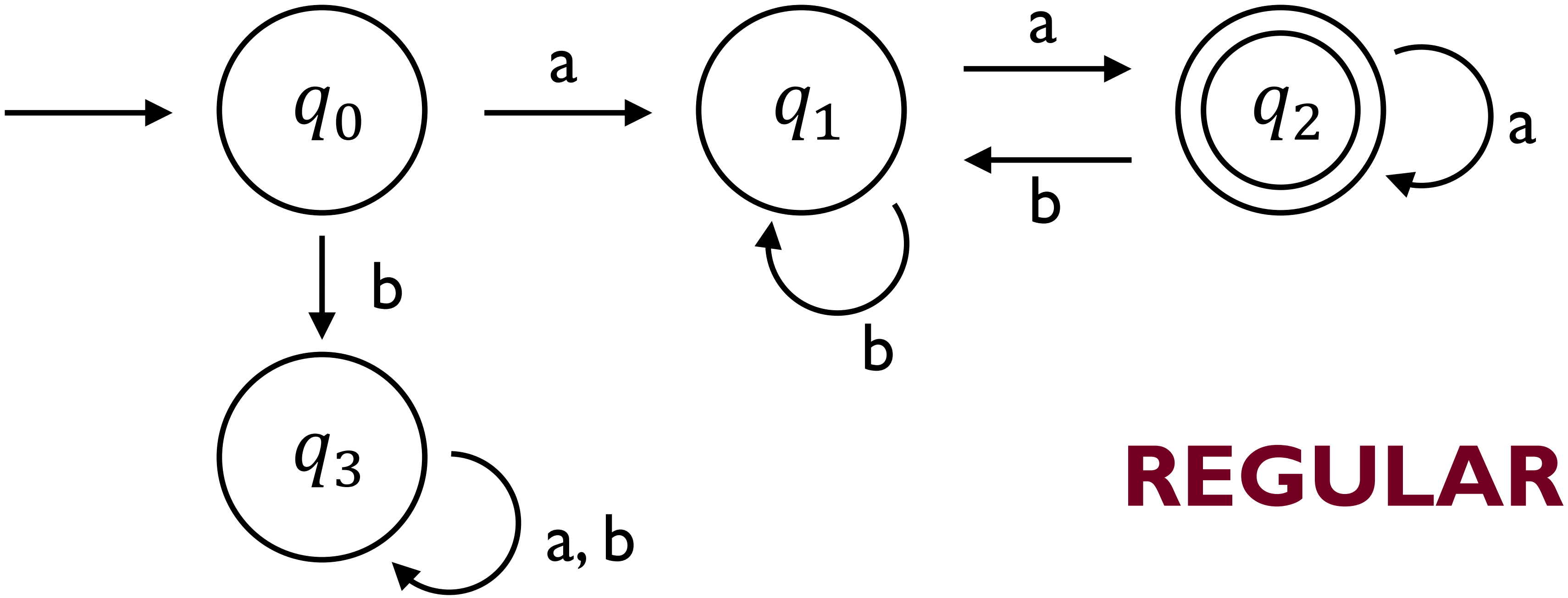
Regular Languages

- **Regular language**
  - Example
    - Show that the language $L = \{awa : w \in \{a, b\}^*\}$ is regular



Regular Languages

- **Regular language**
  - Example
    - Show that the language $L = \{awa: w \in \{a, b\}^*\}$ is regular



REGULAR!

Regular Languages

- **Practice**
  - Example
    - ❖ Show that the language $L = \{w\colon\ |w|\ mod\ 3 = 0\}$ is regular ($\Sigma = \{a, b\}$)

Regular Languages

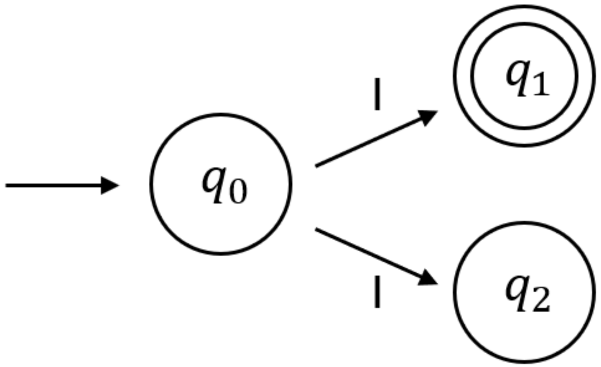
- **Regular language**
  - Example
    - Show that the language $L = \{a^n b^n \mid n \geq 0\}$ is regular

Regular Languages

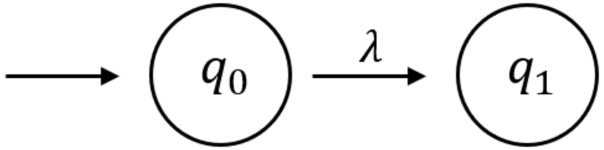
- **Regular language**
  - Example
    - Show that the language $L = \{a^n b^n \mid n \geq 0\}$ is regular
    - We need to construct a DFA M that L = L(M)
    - **But this is impossible!**
      - L is not regular language
      - We will learn how to prove it later in this course

Nondeterministic Finite Automata (NFA)

- **DFA vs NFA**
  - DFA
    - ❖ A unique transition is defined for each state and each input symbol
  - NFA
    - ❖ Multiple or none (λ-transition) transitions possible

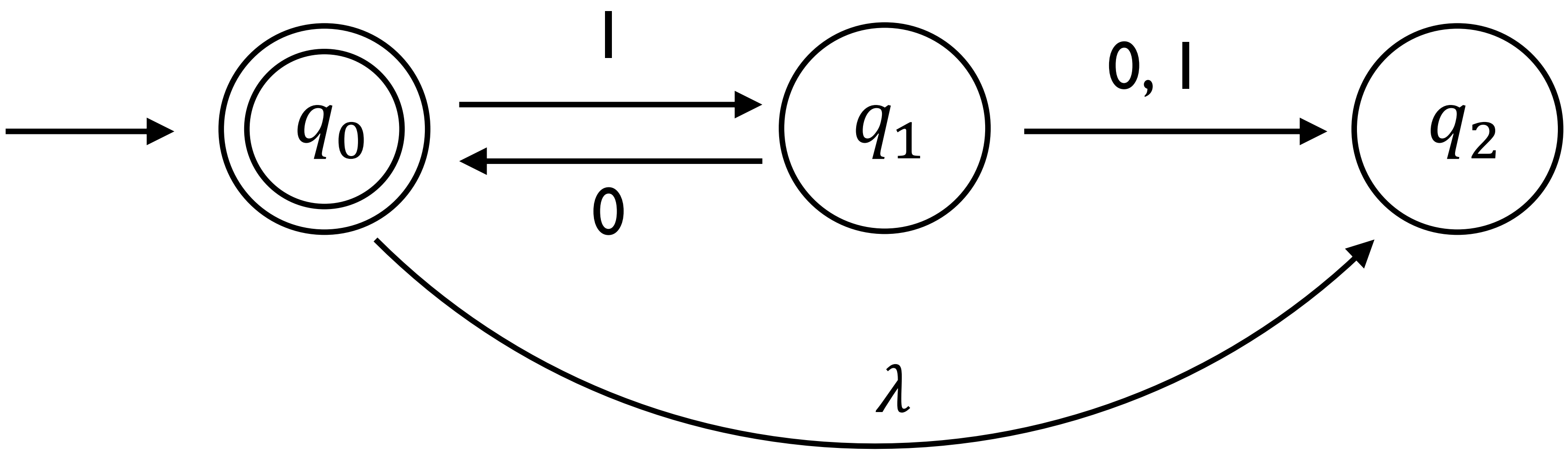


Multiple transitions



λ-transition

Nondeterministic Finite Automata (NFA)



- 1010, 101010 can be accepted
- 110, 10100 cannot be accepted
- For the case of 10
  - Both q0 and q2 are possible => Accepted

Nondeterministic Finite Automata (NFA)

- **Definition of NFA**
  - A NFA is defined by 5-tuples

$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q is a finite set of **internal states**
- Σ is a finite set of **symbols**
- $\boldsymbol{\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^{Q}}$ is the **transition function**
- q_0 is the **initial state** ($q_0 \in Q$)
- F is a set of **final states** ($F \subseteq Q$)

Nondeterministic Finite Automata (NFA)

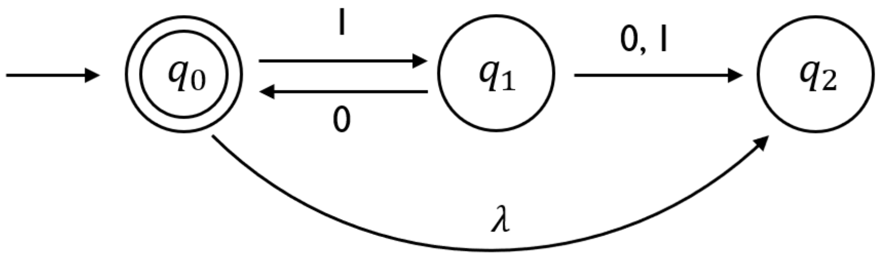
$$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_0\})$$

$\delta(q_0, 0) = \emptyset$ $\delta(q_1, 0) = \{q_0, q_2\}$ $\delta(q_2, 0) = \emptyset$

$\delta(q_0, 1) = \{q_1\}$ $\delta(q_1, 1) = \{q_2\}$ $\delta(q_2, 1) = \emptyset$

$\delta(q_0, \lambda) = \{q_0, q_2\}$ $\delta(q_1, \lambda) = \{q_1\}$ $\delta(q_2, \lambda) = \{q_2\}$

Transition Graph

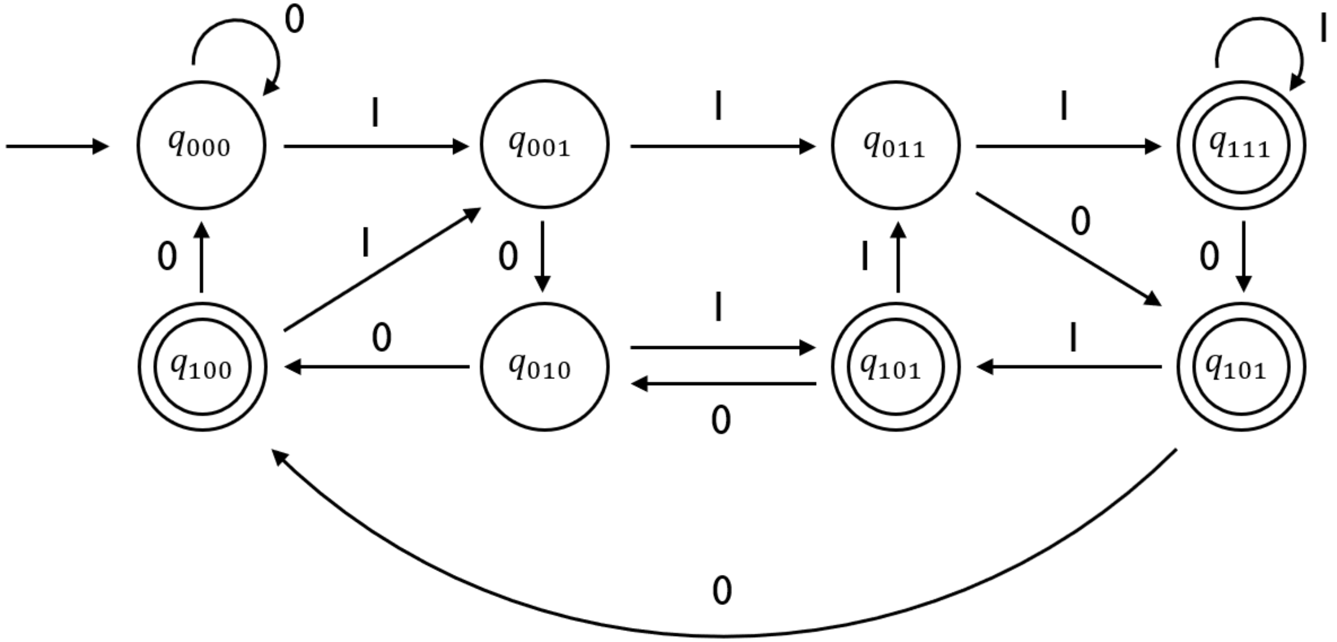


Transition Table

q	0	1	λ
$\rightarrow * \, q_0$	$\emptyset$	$\{q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_0, q_2\}$	$\{q_2\}$	$\{q_1\}$
q_2	$\emptyset$	$\emptyset$	$\{q_2\}$

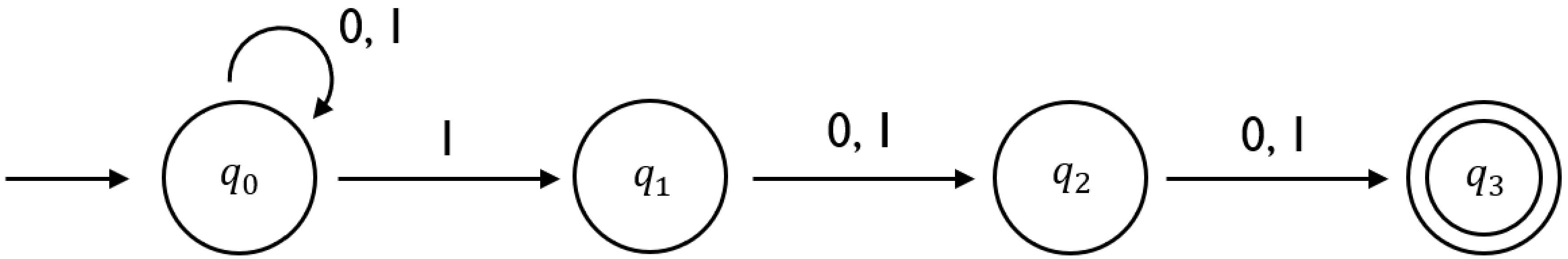
Nondeterministic Finite Automata (NFA)

- **Why NFA is needed?**
  - In certain situations, NFAs can be utilized much more effectively than DFA
  - E.g., FA for accepting strings containing a "1" in third position from the end
    - ❖ DFA



Nondeterministic Finite Automata (NFA)

- **Why NFA is needed?**
  - In certain situations, NFAs can be utilized much more effectively than DFA
  - E.g., FA for accepting strings containing a "1" in third position from the end
    - ❖ NFA



Nondeterministic Finite Automata (NFA)

- **Why NFA is needed?**
  - In certain situations, NFAs can be utilized much more effectively than DFA
  - E.g., FA for accepting strings containing a "1" in third position from the end
    - NFA

Easy to solve a problem and describe a complicated language concisely!

Nondeterministic Finite Automata (NFA)

- **Acceptance of a language**
  - The language accepted by a NFA $M = (Q, \Sigma, \delta, q_0, F)$
    => The set of all strings on Σ accepted by M

$$L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \cap F \neq \emptyset \}$$

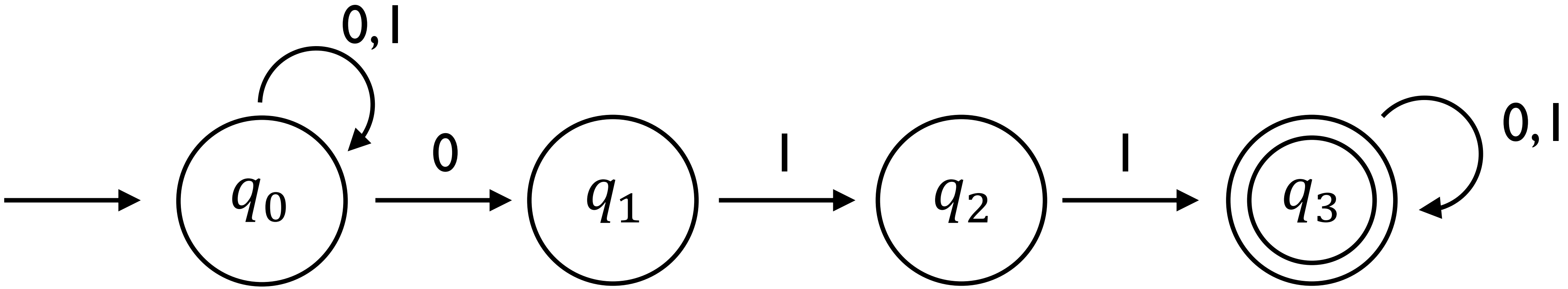
  - Language consists of all strings w
    - ❖For which there is a walk labeled w from the initial state of the transition graph to some final states

Nondeterministic Finite Automata (NFA)

- **Example**
  - Design an NFA for the language $\{w \in \Sigma^* |\ w \text{ contains } 011\}$, where $\Sigma = \{0, 1\}$
    - ❖ 011, 01100, 101100, 1001011100 => Accepted
    - ❖ 1, 11, 101, 11101, 1110000001 => Rejected

Nondeterministic Finite Automata (NFA)

- **Example**
  - Design an NFA for the language $\{w \in \Sigma^* |\ w \text{ contains } 011\}$, where $\Sigma = \{0, 1\}$
    - ❖ 011, 01100, 101100, 1001011100 => Accepted
    - ❖ 1, 11, 101, 11101, 1110000001 => Rejected



Nondeterministic Finite Automata (NFA)

- **Practice**
  - Design an NFA for the language $\{w \in \Sigma^* | \ w \text{ ends with } 00\}$, where $\Sigma = \{0, 1\}$, with three states
    - ❖ 000, 100, 101100, 1001010100 => Accepted
    - ❖ 1, 11, 101, 11101, 1110000001 => Rejected

Equivalence of NFAs and DFAs

- **Every NFA has an equivalent DFA??**
  - Equivalence
    - ❖ Two finite automata, M1 and M2, are said to be equivalent if

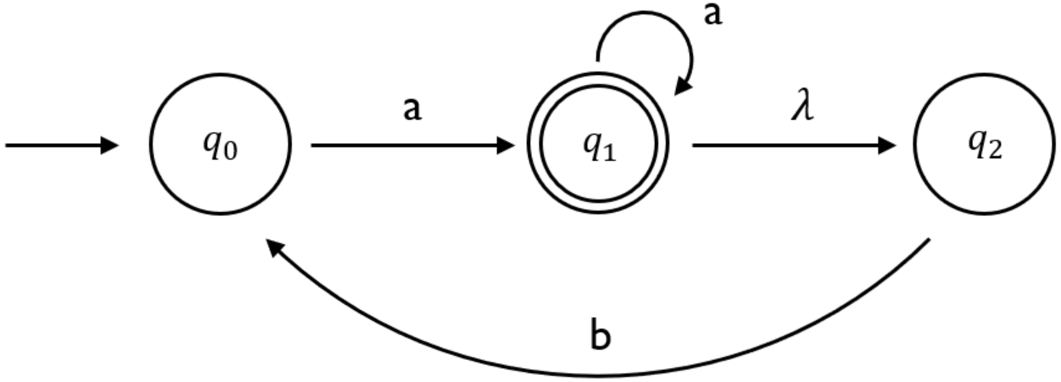
$$L(M_1) = L(M_2)$$

(i.e., They both accept the same language)

Equivalence of NFAs and DFAs

- **NFA (**$N = (Q, \Sigma, \delta, q_0, F)$**) => DFA (**$M = (Q', \Sigma, \delta', {q_0}', F')$**)**
  1. Create a transition table for N

NFA (example)



Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \; q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

Equivalence of NFAs and DFAs

- **NFA ($N = (Q, \Sigma, \delta, q_0, F)$) => DFA ($M = (Q', \Sigma, \delta', q_0', F')$)**

  2. Create the DFA's start state
     - ❖ Set of all possible starting states in the NFA
     - ❖ All states that can be reached from the q_0 by following λ-transition
       - In this case, $\{q_0\}$ will be the starting state

Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \; q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

Equivalence of NFAs and DFAs

- **NFA ($N = (Q, \Sigma, \delta, q_0, F)$) => DFA ($M = (Q', \Sigma, \delta', {q_0}', F')$)**
  3. Create the DFA's transition table
     - Until no new state generated

Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \, q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

Transition table for M

q	a	b
$\rightarrow \{q_0\}$	$\{q_1, q_2\}$	$\emptyset$

Equivalence of NFAs and DFAs

- **NFA (**$N = (Q, \Sigma, \delta, q_0, F)$**) => DFA (**$M = (Q', \Sigma, \delta', q_0{}', F')$**)**

  3. Create the DFA's transition table

     ❖ Until no new state generated

Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \; q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

Transition table for M

q	a	b
$\rightarrow \{q_0\}$	$\{q_1, q_2\}$	$\emptyset$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0\}$

Equivalence of NFAs and DFAs

- **NFA ($N = (Q, \Sigma, \delta, q_0, F)$) => DFA ($M = (Q', \Sigma, \delta', {q_0}', F')$)**
  3. Create the DFA's transition table
     - Until no new state generated

Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \ q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

Transition table for M

q	a	b
$\rightarrow \{q_0\}$	$\{q_1, q_2\}$	$\emptyset$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0\}$
$\emptyset$	$\emptyset$	$\emptyset$

Equivalence of NFAs and DFAs

- **NFA ($N = (Q, \Sigma, \delta, q_0, F)$) => DFA ($M = (Q', \Sigma, \delta', {q_0}', F')$)**
  4. Determining the final state of the DFA
     - ❖ Sets of states that contain at least one final state from the NFA

Transition table for N

q	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$	$\{q_0\}$
$* \; q_1$	$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\{q_0\}$	$\{q_2\}$

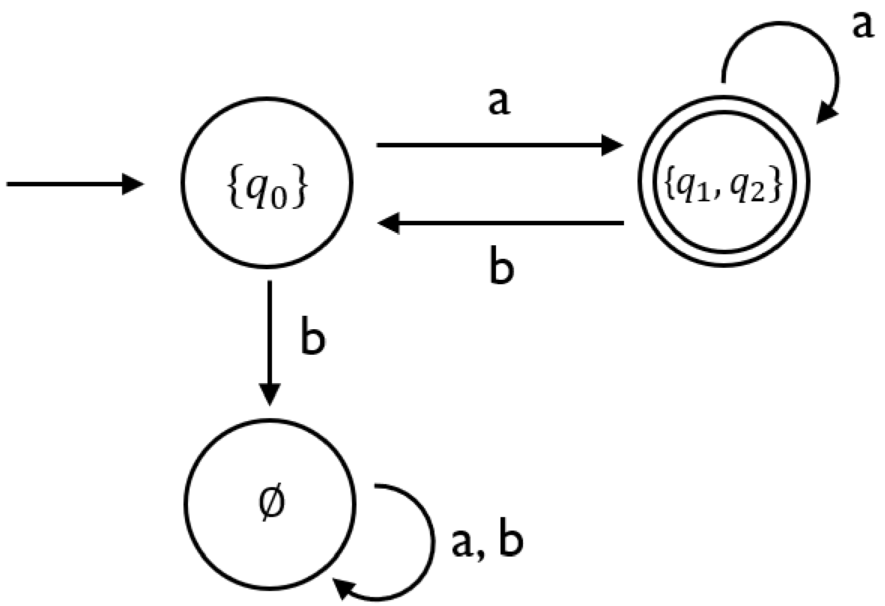
Transition table for M

q	a	b
$\rightarrow \{q_0\}$	$\{q_1, q_2\}$	$\emptyset$
$* \; \{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0\}$
$\emptyset$	$\emptyset$	$\emptyset$

Equivalence of NFAs and DFAs

- **NFA (**$N = (Q, \Sigma, \delta, q_0, F)$**) => DFA (**$M = (Q', \Sigma, \delta', {q_0}', F')$**)**

Transition graph for M



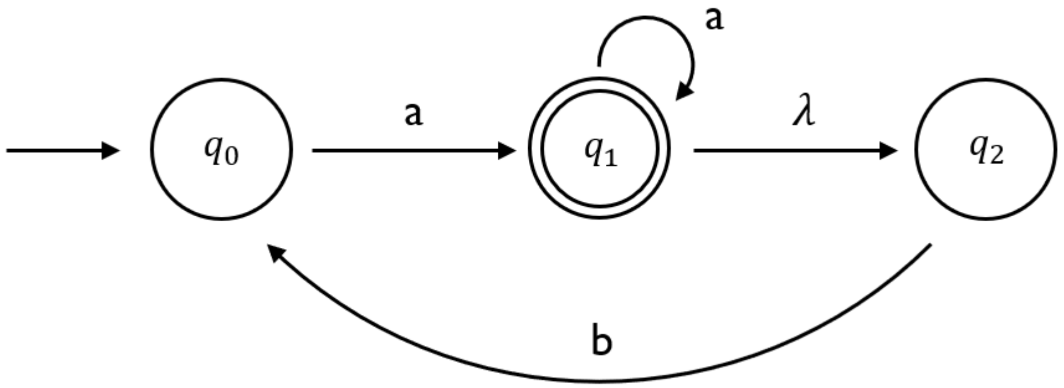
Transition table for M

q	a	b
$\rightarrow \{q_0\}$	$\{q_1, q_2\}$	$\emptyset$
$* \{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0\}$
$\emptyset$	$\emptyset$	$\emptyset$

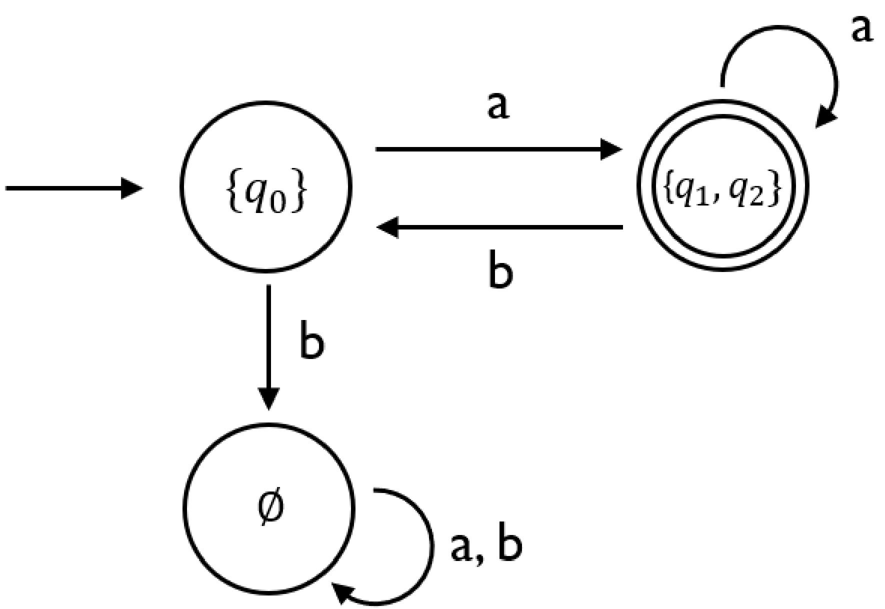
Equivalence of NFAs and DFAs

- **NFA ($N = (Q, \Sigma, \delta, q_0, F)$) => DFA ($M = (Q', \Sigma, \delta', q_0{}', F')$)**

NFA (example)

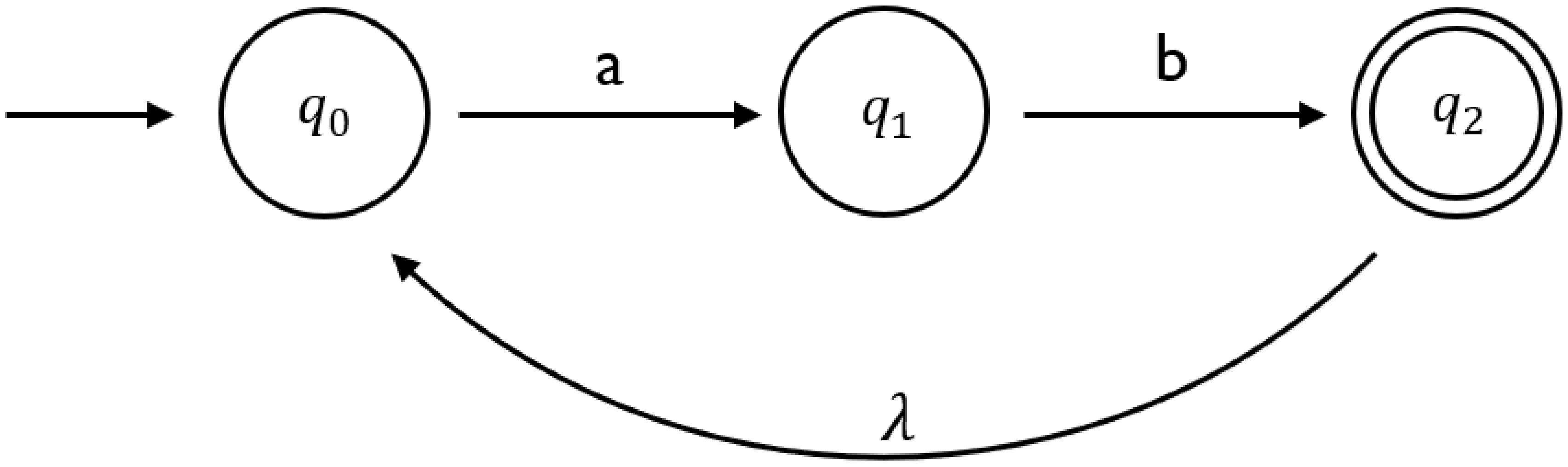


Transition graph for M



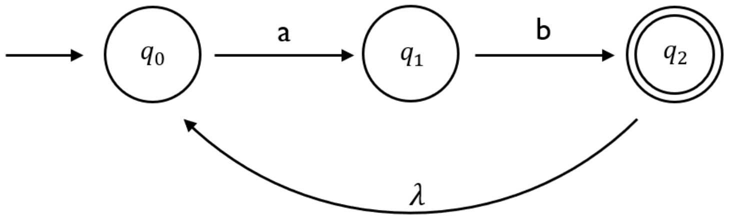
Equivalence of NFAs and DFAs

- **Practice**
  - Converting NFA to DFA ($\Sigma = \{a, b\}$)



Equivalence of NFAs and DFAs

- **Practice**
  - Converting NFA to DFA

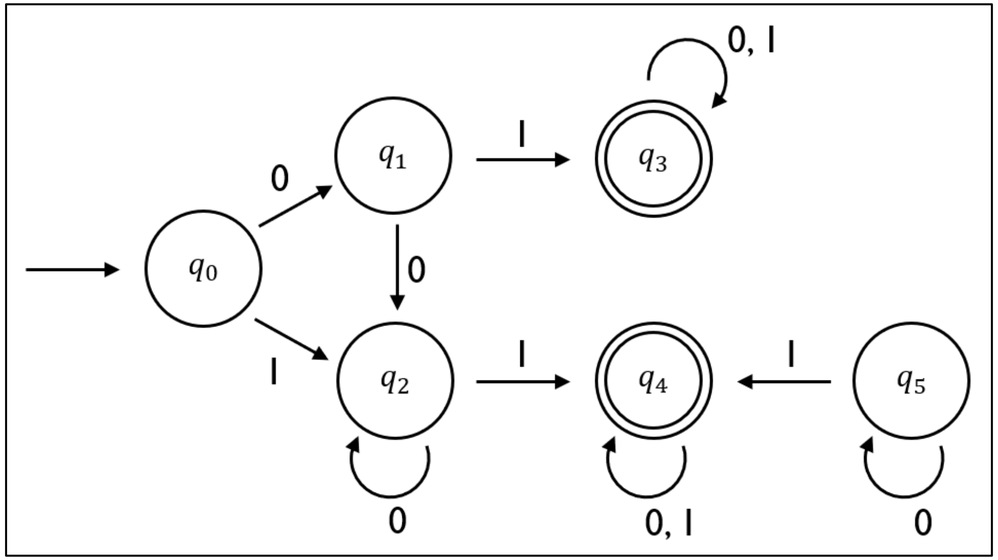


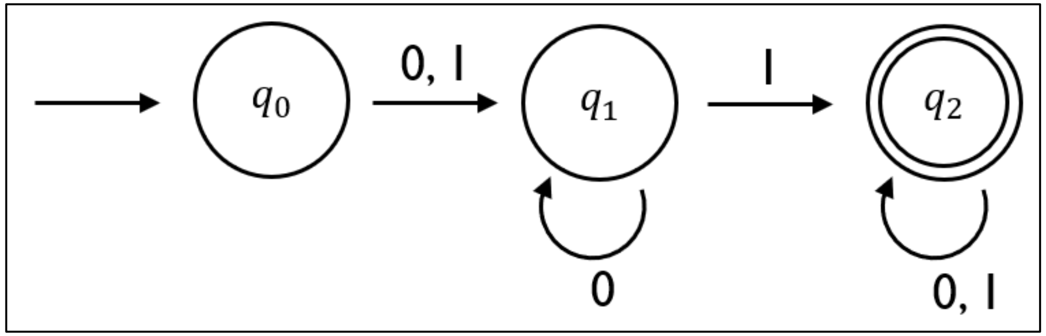
Reduction of the number of states

- **One language can be accepted by many DFAs**
  - One DFA => One language
  - One language => many DFAs

Reduction of the number of states

- **One language can be accepted by many DFAs**
  - One DFA => One language
  - One language => many DFAs





states reachable subsequent to $\delta(q_0, 0)$
= states reachable subsequent to $\delta(q_0, 1)$

Reduction of the number of states

- **Indistinguishable states**
  - Two states p and q of a DFA are called **indistinguishable** if

$$\delta^*(p, w) \in F \text{ implies } \delta^*(q, w) \in F,$$

and

$$\delta^*(p, w) \notin F \text{ implies } \delta^*(q, w) \notin F$$

Reduction of the number of states

- **Indistinguishable states**
  - Two states p and q of a DFA are called **indistinguishable** if

$$\delta^*(p, w) \in F \text{ implies } \delta^*(q, w) \in F,$$

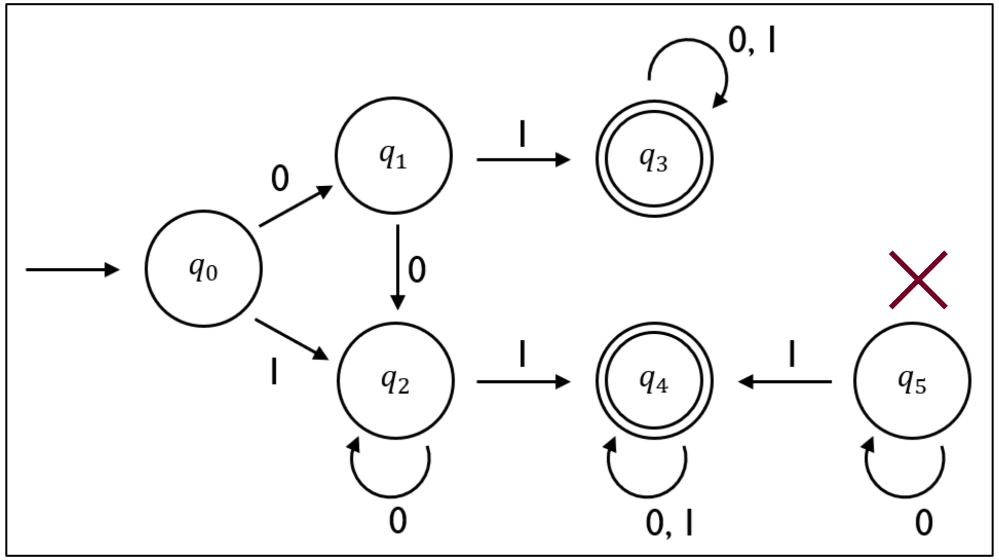
and

$$\delta^*(p, w) \notin F \text{ implies } \delta^*(q, w) \notin F$$

  - Reducing the number of DFA states
    = finding indistinguishable pairs and merging them

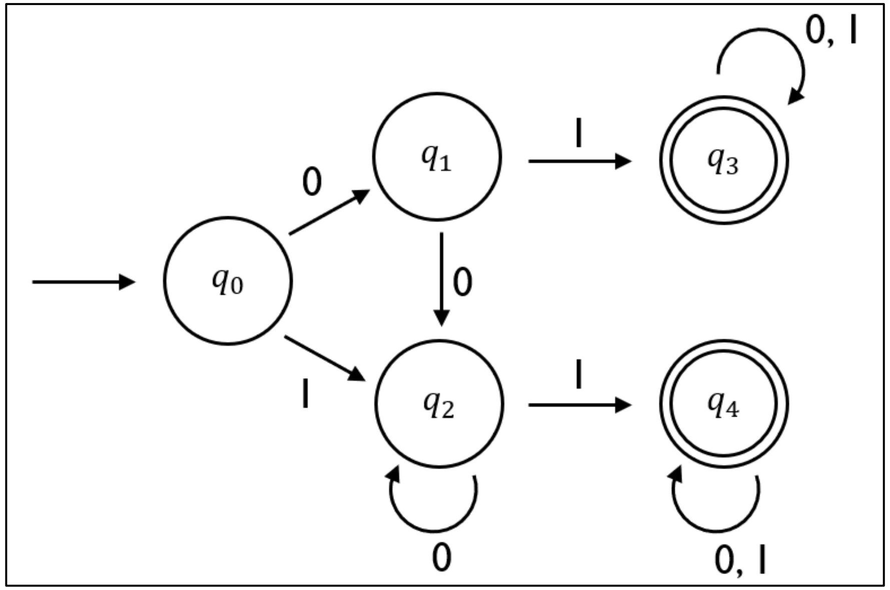
Reduction of the number of states

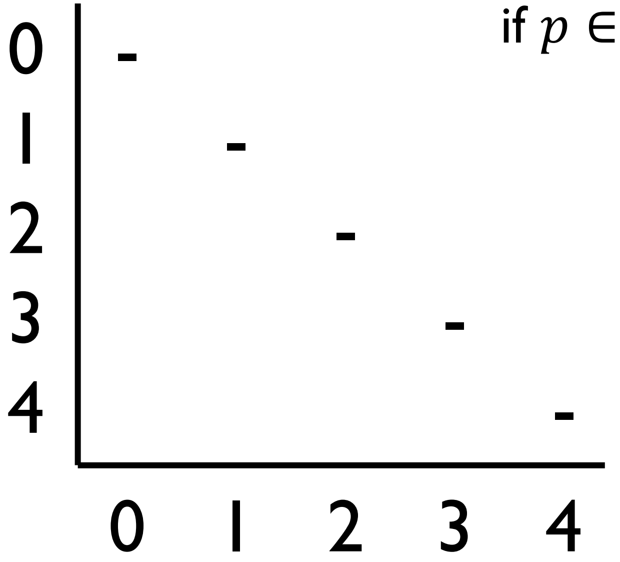
- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state



Reduction of the number of states

- **Finding and merging indistinguishable pairs**
    1. Remove all inaccessible states, where no path exists from the initial state
    2. Construct a grid of pairs of states

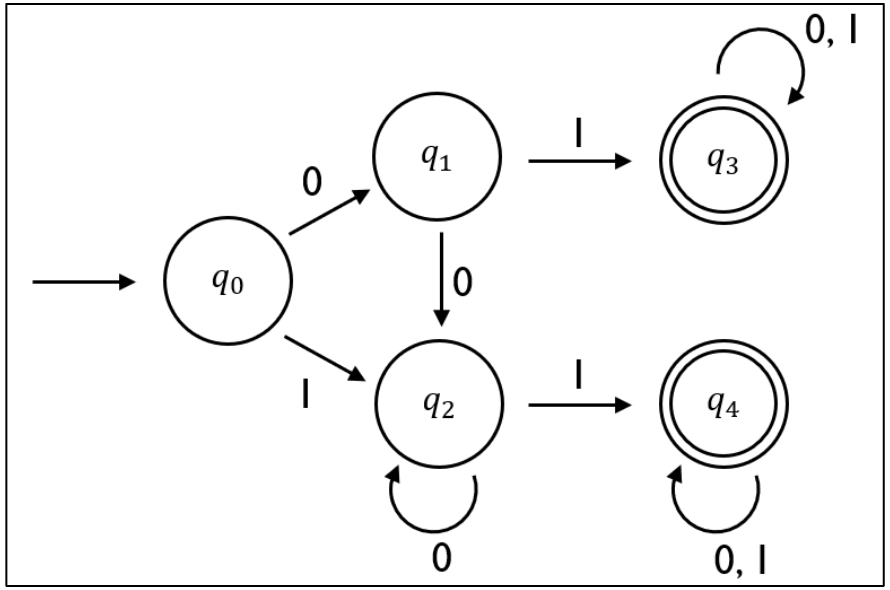




For a pair (p, q),
if $p \in F$ and $q \notin F$ (or vice versa),
(p, q) is distinguishable

Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state
  2. Construct a grid of pairs of states



	0	1	2	3	4
0	-				
1	?	-			
2	?	?	-		
3	X	X	X	-	
4	X	X	X	?	-

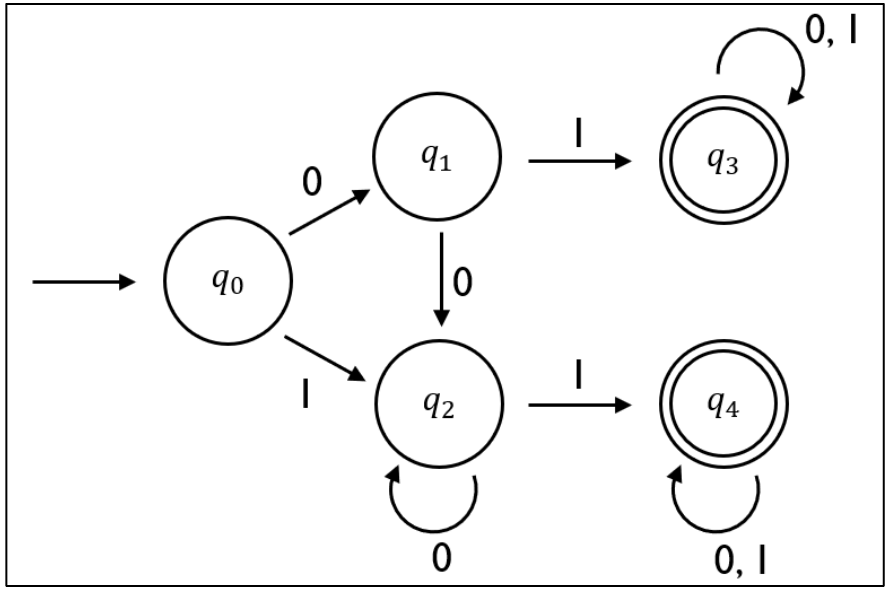
For a pair (p, q),
if $p \in F$ and $q \notin F$ (or vice versa),
(p, q) is distinguishable

Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state
  2. Construct a grid of pairs of states

For all pairs (p, q) and all $a \in \Sigma$, compute $\delta(p, a) = p_a$ and $\delta(q, a) = q_a$.

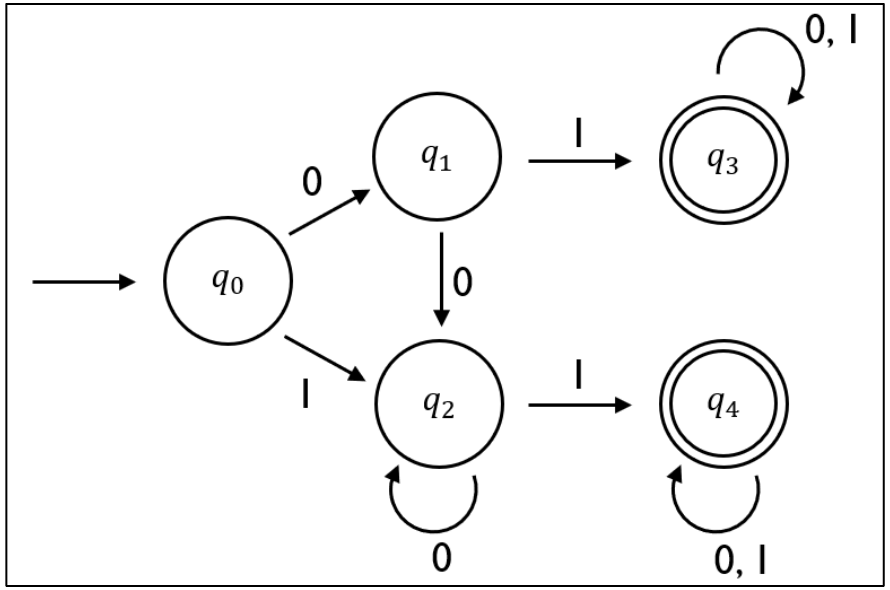
If the pair (p_a, q_a) is distinguishable, then (p, q) is distinguishable



	0	1	2	3	4
0	-				
1	?	-			
2	?	?	-		
3	X	X	X	-	
4	X	X	X	?	-

Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state
  2. Construct a grid of pairs of states

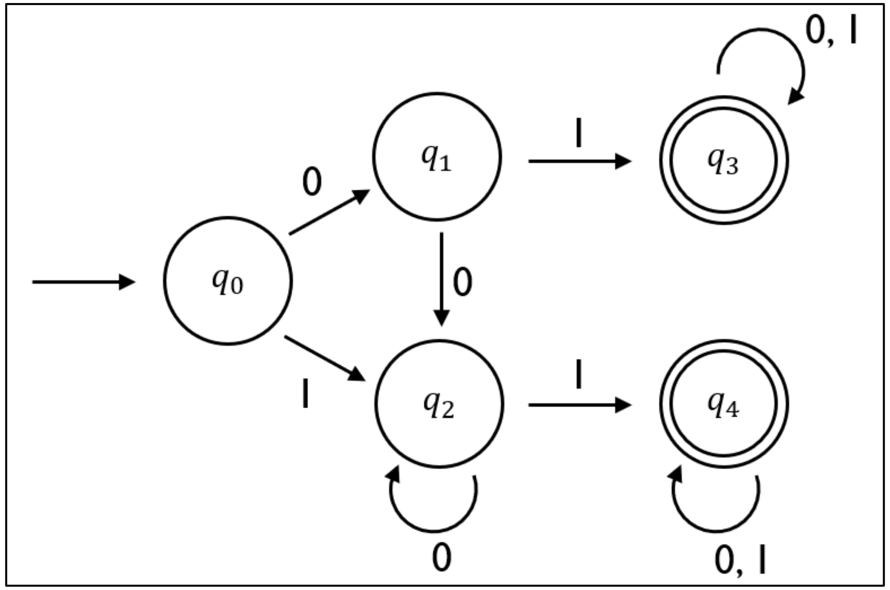


	0	1	2	3	4
0	-				
1	?	-			
2	?	?	-		
3	X	X	X	-	
4	X	X	X	?	-

{0, 1} given 0 => {1, 2}
{0, 1} given 1 => **{2, 3}**
{0, 2} given 0 => {1, 2}
{0, 2} given 1 => **{2, 4}**
{1, 2} given 0 => {2, 2}
{1, 2} given 1 => {3, 4}
{3, 4} given 0 => {3, 4}
{3, 4} given 1 => {3, 4}

Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state
  2. Construct a grid of pairs of states

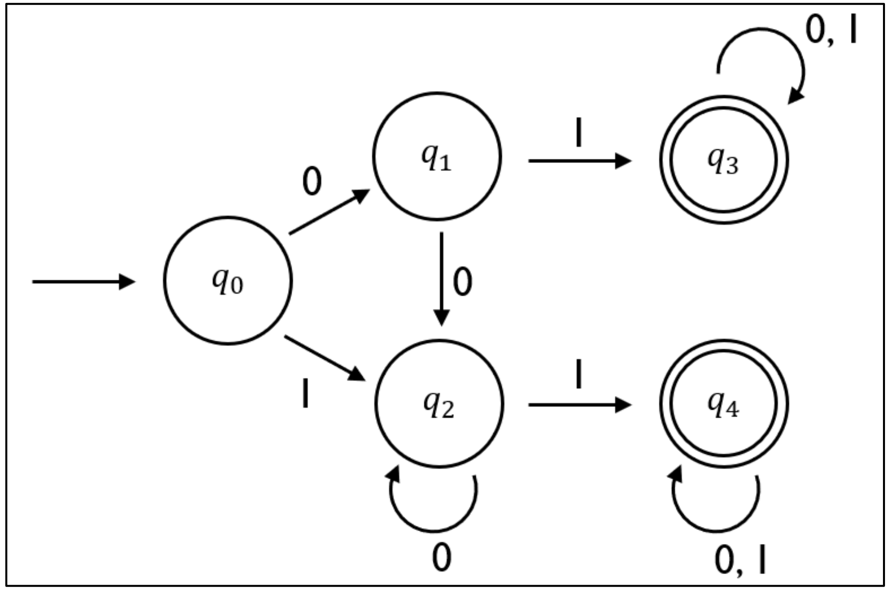


	0	1	2	3	4
0	-				
1	X	-			
2	X	O	-		
3	X	X	X	-	
4	X	X	X	O	-

{0, 1} given 0 => {1, 2}
{0, 1} given 1 => **{2, 3}**
{0, 2} given 0 => {1, 2}
{0, 2} given 1 => **{2, 4}**
{1, 2} given 0 => {2, 2}
{1, 2} given 1 => {3, 4}
{3, 4} given 0 => {3, 4}
{3, 4} given 1 => {3, 4}

Reduction of the number of states

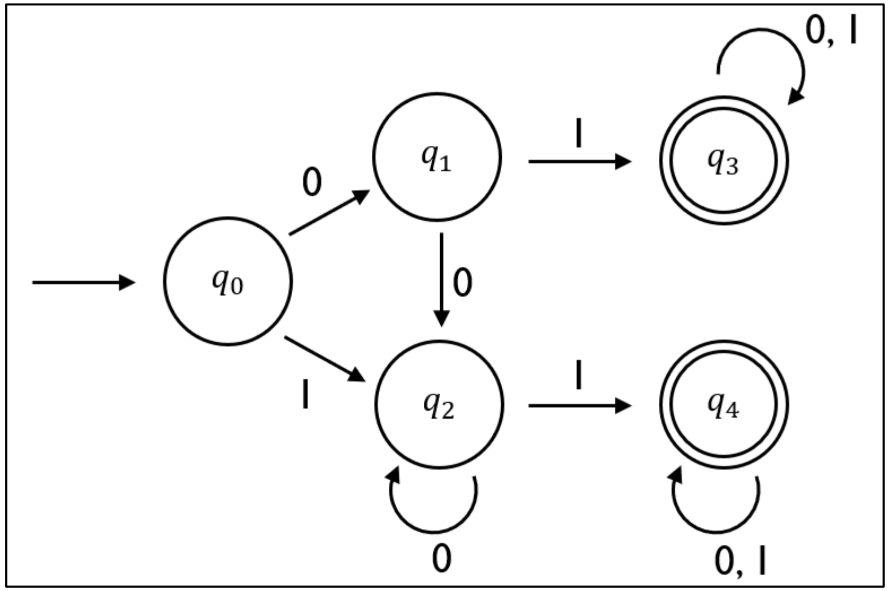
- **Finding and merging indistinguishable pairs**
  1. Remove all inaccessible states, where no path exists from the initial state
  2. Construct a grid of pairs of states
     - ❖ $\{q_1, q_2\}$ and $\{q_3, q_4\}$ are indistinguishable!
  3. Construct a new DFA
     - ❖ Indistinguishable states => a single state
     - ❖ We have three states
       - $\{q_0\}, \{q_1, q_2\}$, and $\{q_3, q_4\}$



Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  3. Construct a new DFA
     - ❖ We have three states
       - $\{q_0\}, \{q_1, q_2\}$, and $\{q_3, q_4\}$

$$\delta(q_0, 0) = q_1$$
$$\delta(q_0, 1) = q_2$$
$$\downarrow$$
$$\delta'(\{q_0\}, 0) = \delta'(\{q_0\}, 1) = \{q_1, q_2\}$$



Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  3. Construct a new DFA
     - ❖We have three states
       - $\{q_0\}, \{q_1, q_2\}$, and $\{q_3, q_4\}$

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_2$

↓

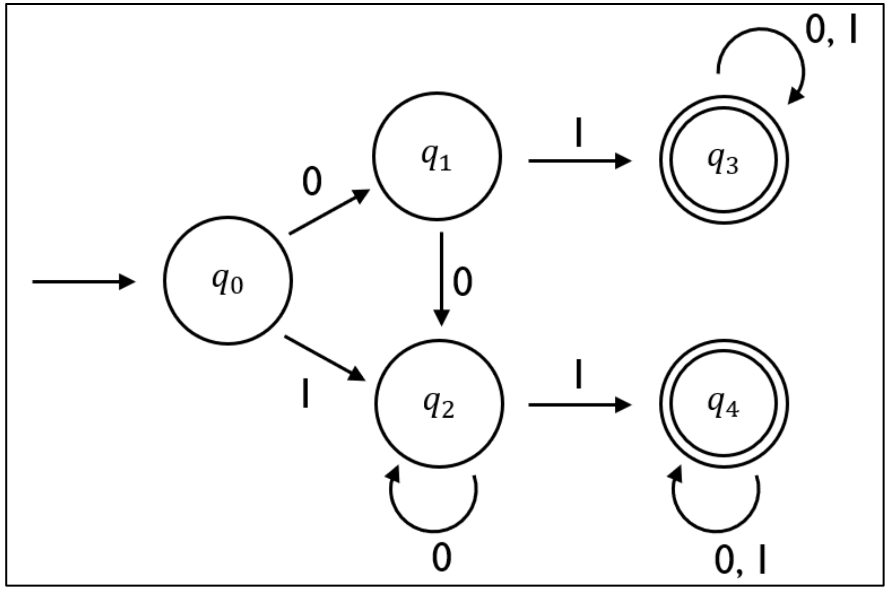
$\delta'(\{q_0\}, 0)$
$= \delta'(\{q_0\}, 1)$
$= \{q_1, q_2\}$

$\delta(q_1, 0) = q_2$

$\delta(q_2, 0) = q_2$

↓

$\delta'(\{q_1, q_2\}, 0)$
$= \{q_1, q_2\}$



Reduction of the number of states

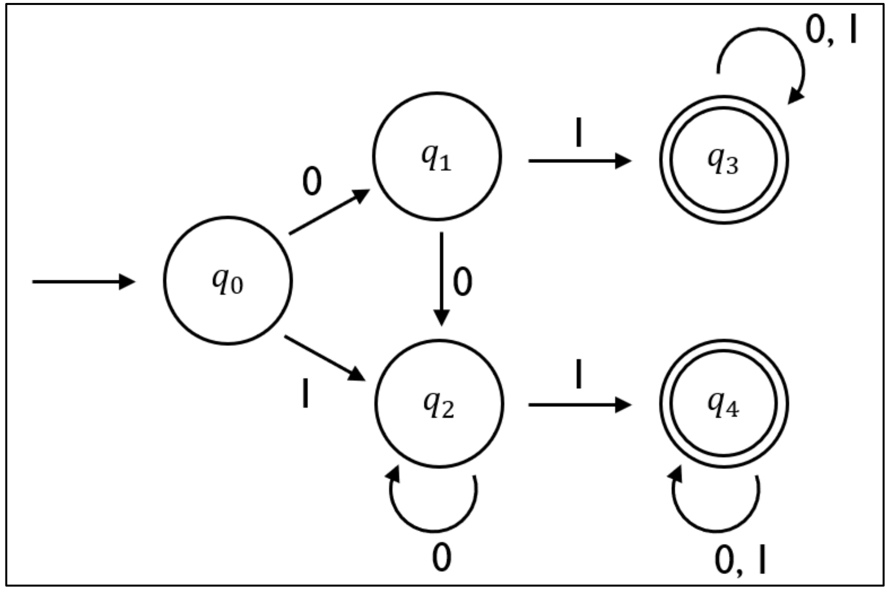
- **Finding and merging indistinguishable pairs**

3. Construct a new DFA

❖ We have three states

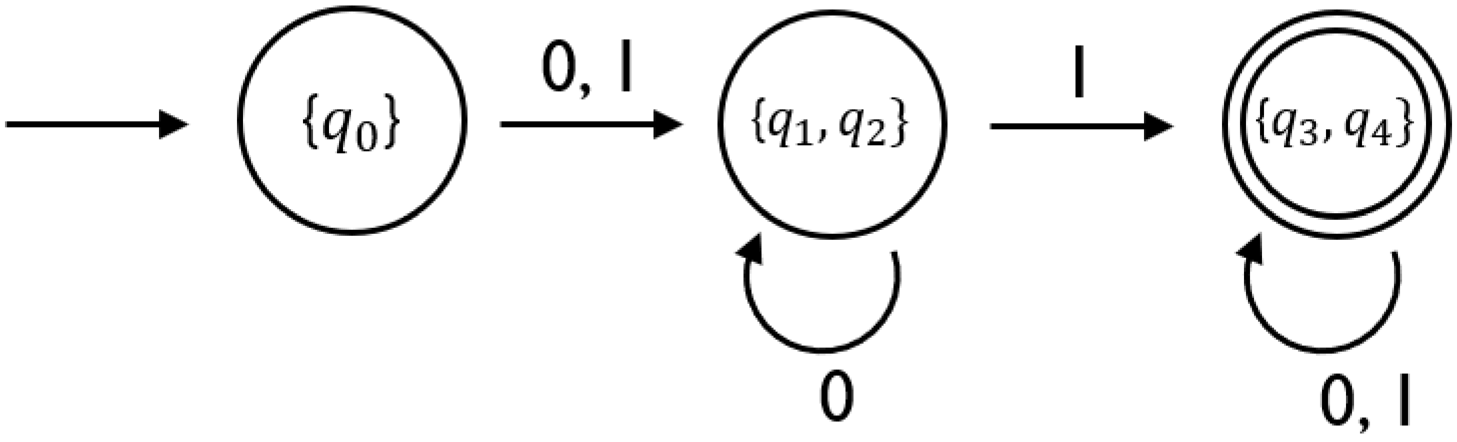
- $\{q_0\}, \{q_1, q_2\}$, and $\{q_3, q_4\}$

$\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_2$ $\Downarrow$ $\delta'(\{q_0\}, 0)$ $= \delta'(\{q_0\}, 1)$ $= \{q_1, q_2\}$	$\delta(q_1, 0) = q_2$ $\delta(q_2, 0) = q_2$ $\Downarrow$ $\delta'(\{q_1, q_2\}, 0)$ $= \{q_1, q_2\}$	$\delta(q_1, 1) = q_3$ $\delta(q_2, 2) = q_4$ $\Downarrow$ $\delta'(\{q_1, q_2\}, 1)$ $= \{q_3, q_4\}$	$\delta'(\{q_3, q_4\}, 0)$ $= \delta'(\{q_3, q_4\}, 1)$ $= \{q_3, q_4\}$



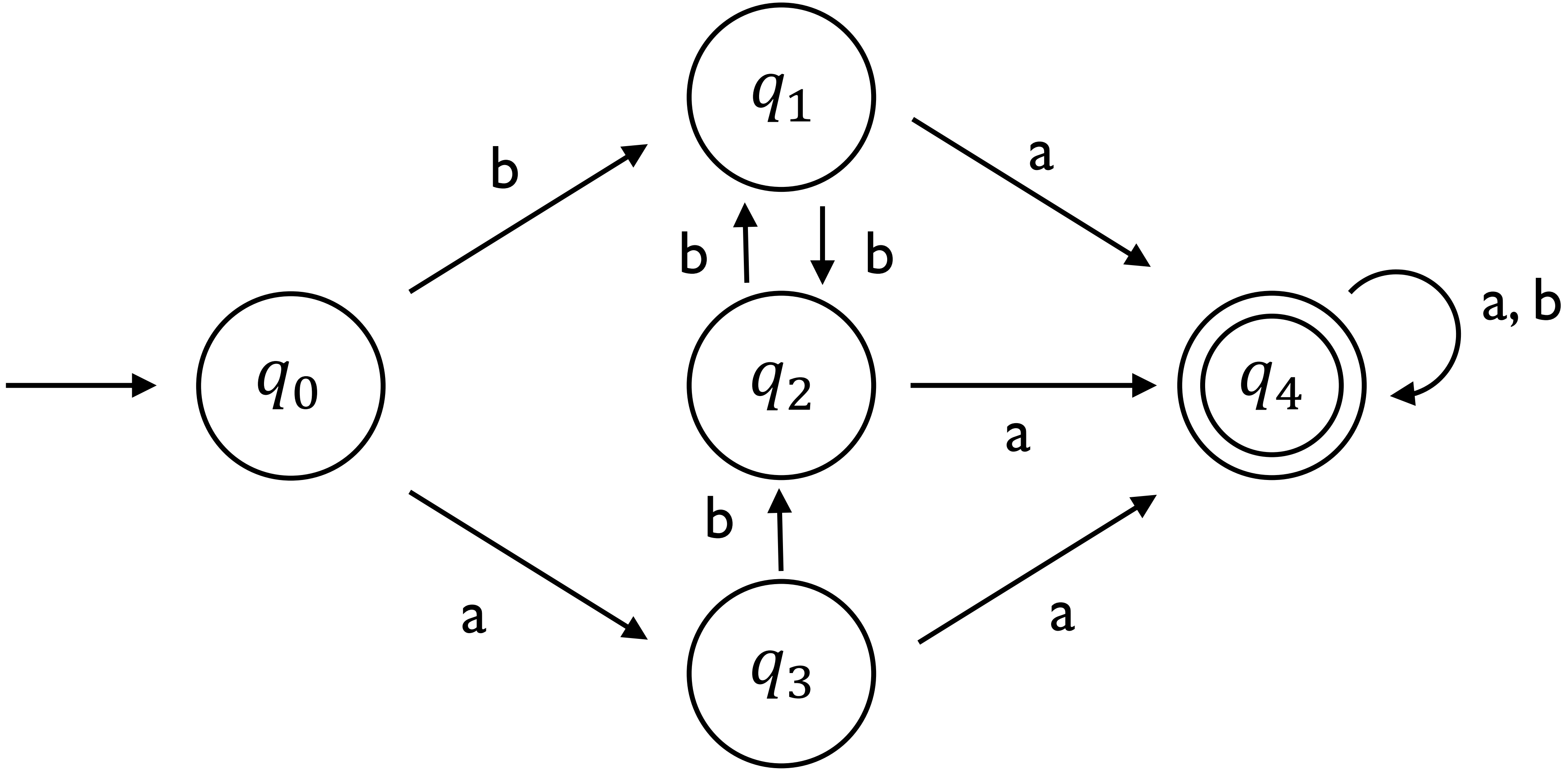
Reduction of the number of states

- **Finding and merging indistinguishable pairs**
  3. Construct a new DFA



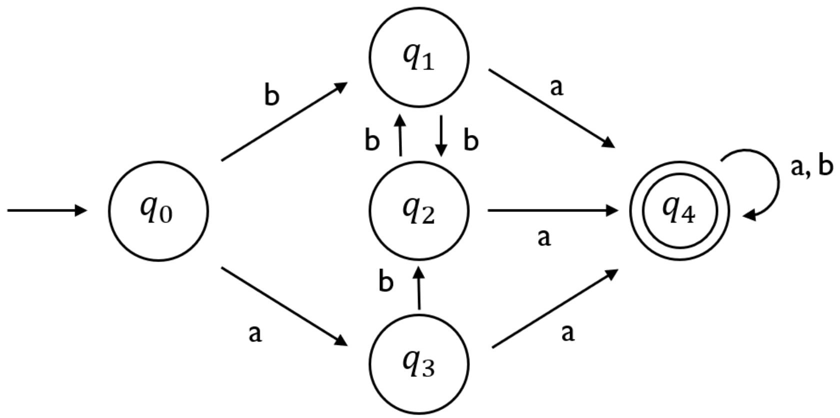
Reduction of the number of states

- **Practice ($\boldsymbol{\Sigma = \{a, b\}}$)**



Reduction of the number of states

- **Practice**



Next Lecture

- **Regular Languages and Regular Grammars**