CSCI 361 Lecture 2: Finite Automata

Shikha Singh

Announcements & Logistics

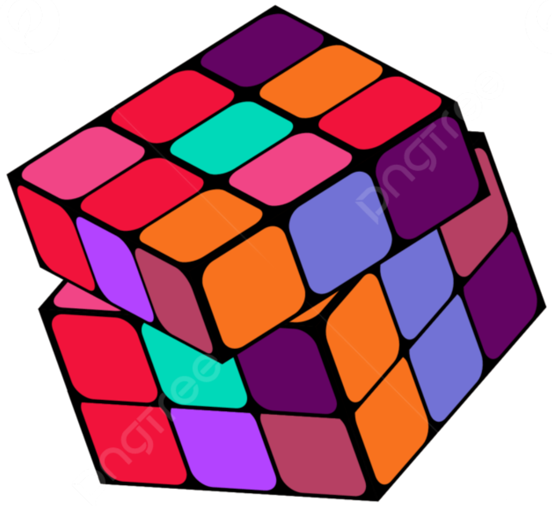
- Hand in **Exercise 1,** pick up **Exercise 2**
- Pick up Lecture 2 Handout
- Assignment 1 due Wed at 10 pm on Gradescope
- Make sure to use the LaTeX template provided (required)
- Midterm dates:
  - October 7 (Tuesday) and Nov 6 (Thursday)
  - Please make it on your calendars
- **Questions?**

Last Time

- Introduced history and overview of theory of computation
- Discussed course logistics and reviewed syllabus
- Defined fundamentals of input/output representation
  - **Alphabet** Σ and set of all strings Σ
  - **Language**: any subset of strings from alphabet, i.e., $L \subseteq \Sigma^*$
  - **Length** of string s (# of symbols)
- All input/output in this course will be **binary strings**, that is, $\Sigma = \{0,1\}$
- Function problem vs decision problem:
  - A function problem is given by $f : \Sigma^* \rightarrow \Sigma^*$
  - A decision problem is given by $f : \Sigma^* \rightarrow \{0,1\}$

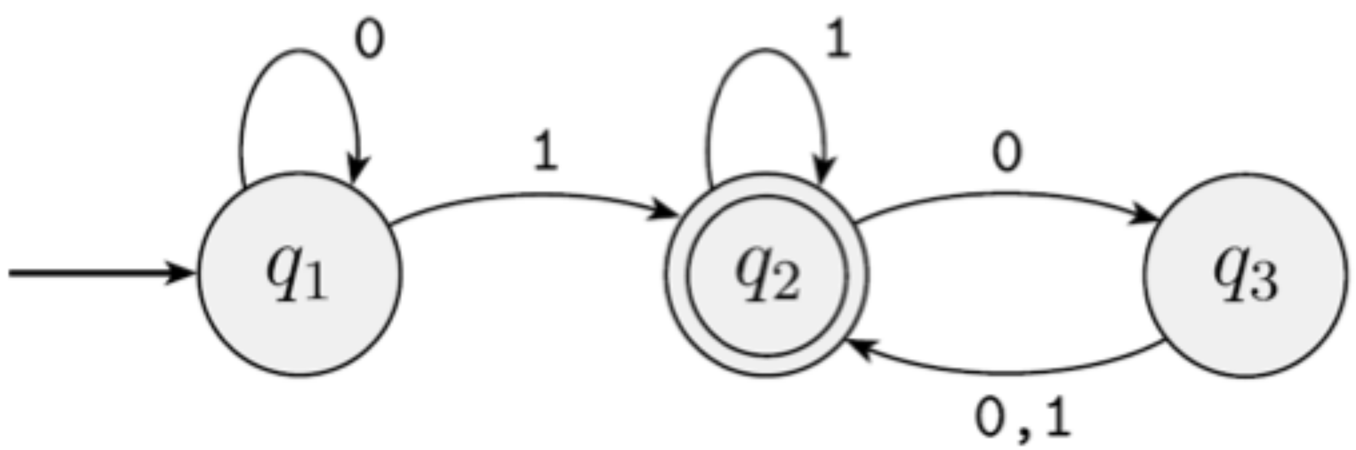
Finite State Automata

Simplest Form of Computation



Deterministic Finite Automata

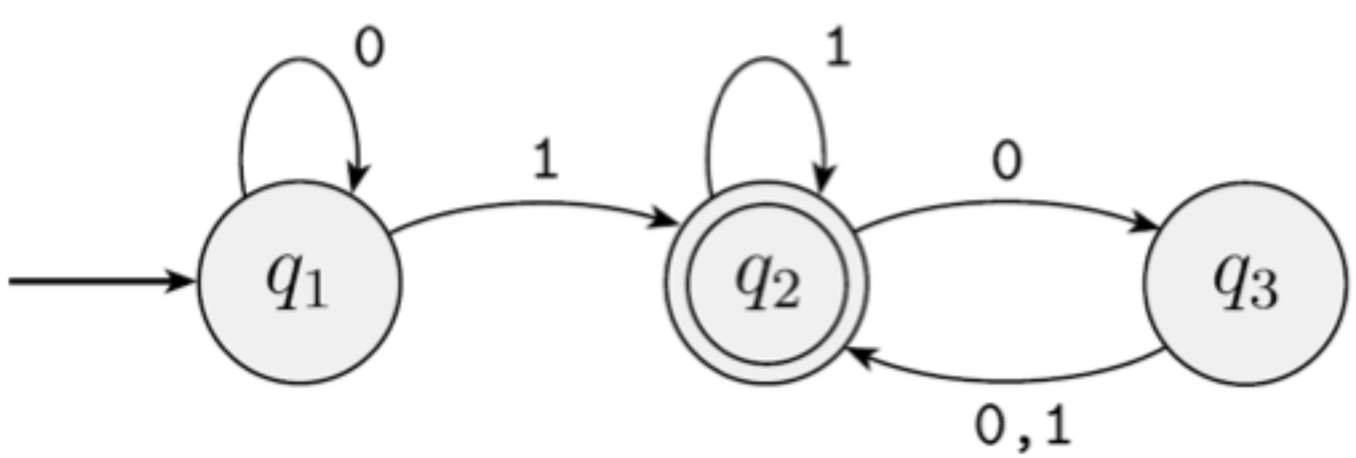
- A **machine** ***recognizes*** a language (akin to listening)
  - If a given input string is in a language, the machine will "accept" (output true), otherwise "reject" (output false)
- **Question.** What language is recognized by this machine?
  - Try some example strings



Definition of a Finite Automaton

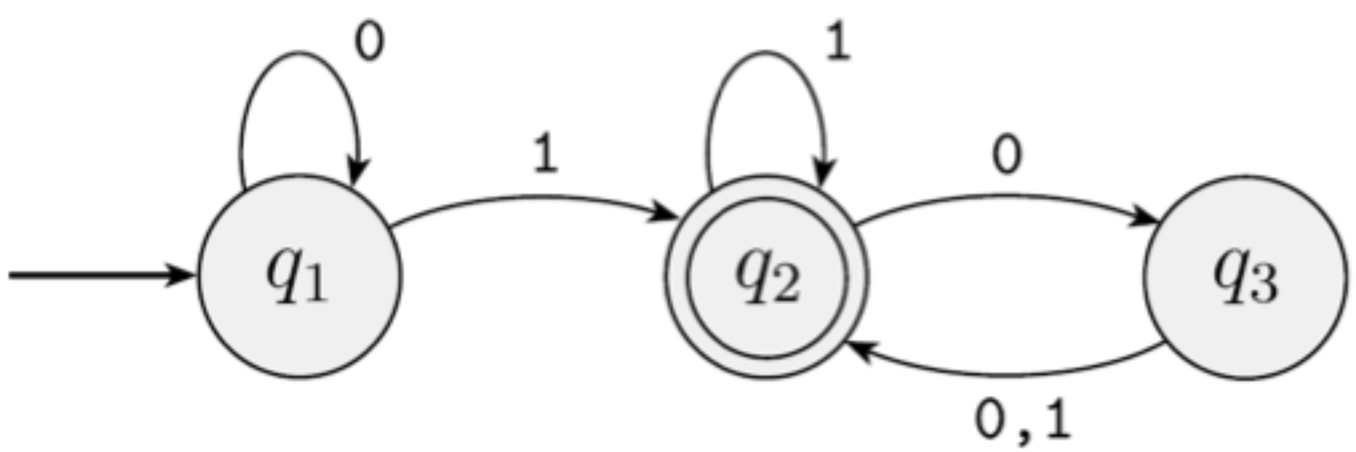
A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set called the states,
- Σ is a finite set called the alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_o \in Q$ is the start state and $F \subseteq Q$ is the set of accept states.

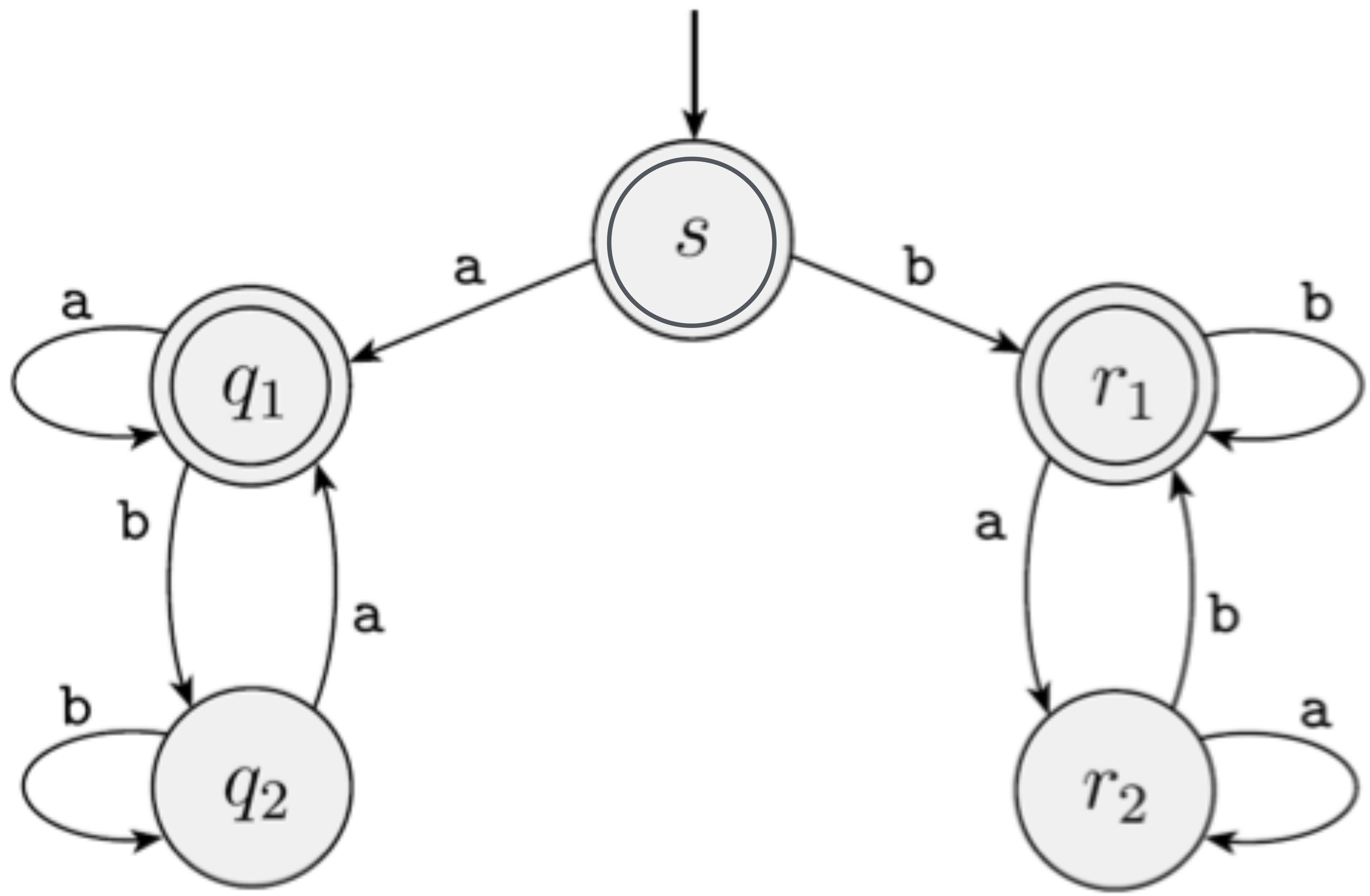


Language of a Machine

- The set of all strings accepted by a finite automaton M is called the language of machine M, and is written $L(M)$.
    - Say M **recognizes** language $L(M)$
- We will define M accepts w more formally
- Intuitive it is the strings on which it reaches an accepting state



What Language?



Automaton Computation

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$. Then M ***accepts*** w if there is a sequence of $r_0, r_1, \ldots, r_n$ in Q such that
  - $r_0 = q_0$
  - $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, 1, \ldots, n-1$ and
  - $r_n \in F$

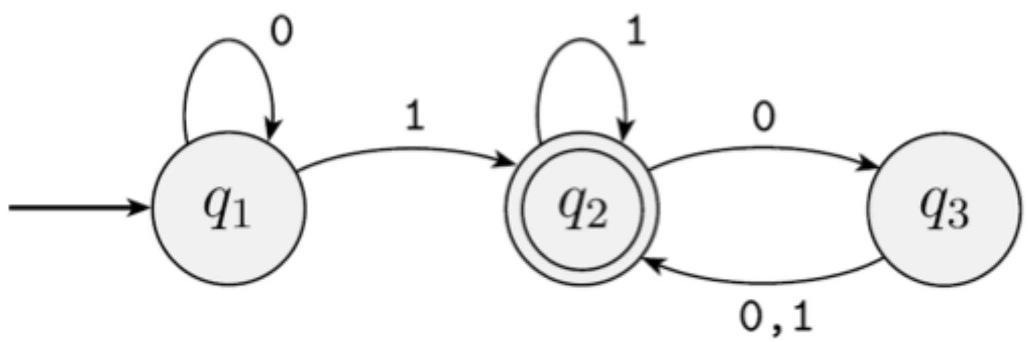
Extended Transition Function

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA
- Transition function $\delta : Q \times \Sigma \to Q$ is often extended to $\delta^* : Q \times \Sigma^* \to Q$ where $\delta^*(q, w)$ is defined as the state the DFA ends up in if it starts at q and reads the string w
- Alternate definition of M accepts $w \iff \delta^*(q_0, w) \in F$

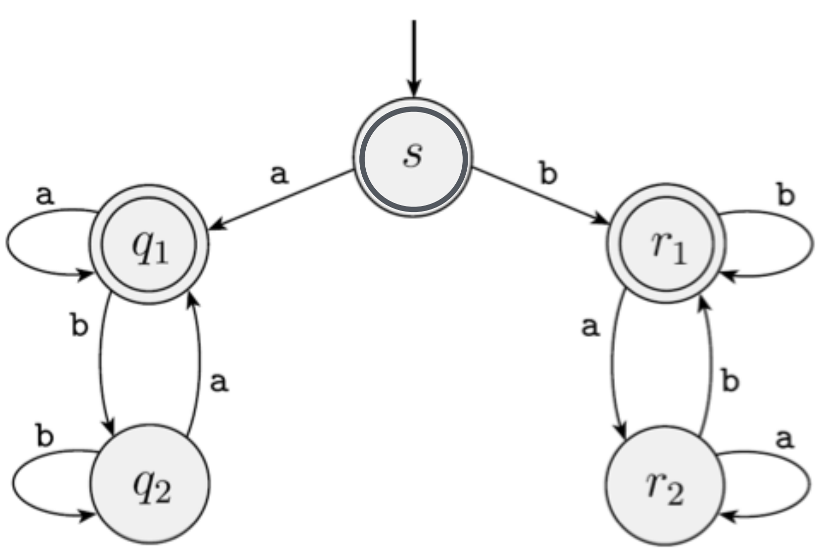
Language of a Machine

- The set of all strings accepted by a finite automaton M is called the language of machine M, and is written $L(M)$.
    - Say M **recognizes** language $L(M)$

M_1



M_2



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

$L(M_2) = \{w \mid w \in \{a, b\}^*$ that starts and ends with the same symbol$\}$

Regular Languages

- **Definition**. A language is called a **regular** language if some deterministic finite automaton recognizes it.
- Thus, to show a language L is regular, we must design a DFA M that recognizes it, that is, $L(M) = L$
    - M accepts $w \iff w \in L$

Class Exercise: Practice with DFAs

- Show that the following languages are regular by drawing the state diagram of a DFA that recognizes it:
- $\{w \in \{0,1\}^* \mid w$ contains an even number of 1s $\}$
- $\{w \in \{0,1\}^* \mid w$ ends in 01 $\}$
- $\{w \in \{a,b\}^* \mid w$ contains the substring aba $\}$

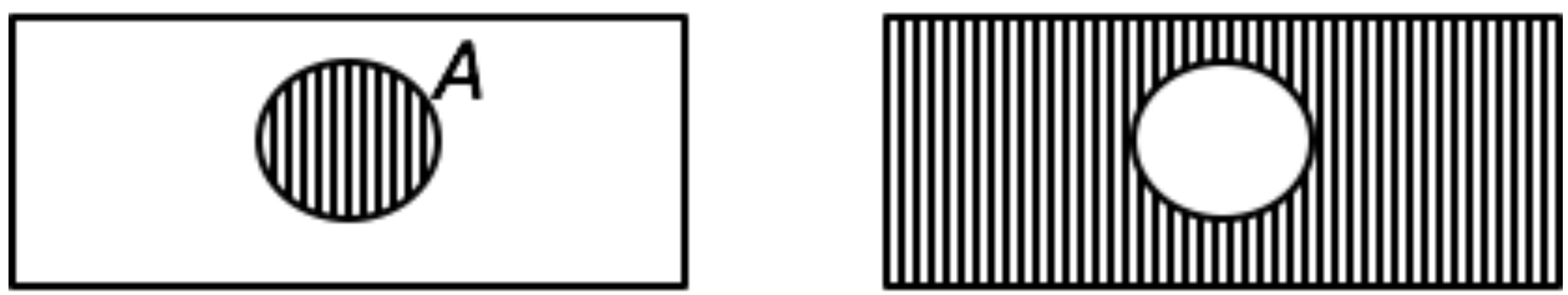
How About These Languages?

- Any similarities?
  - $L_4 = \{w \in \{0,1\}^* \mid w$ contains an **odd number** of 1s $\}$
  - $L_5 = \{w \in \{0,1\}^* \mid w$ **does not end in** $01\}$
  - $L_4 = \{w \in \{a,b\}^* \mid w$ **does not contain** the substring aba $\}$

Regular Operations

Building New Languages From Old

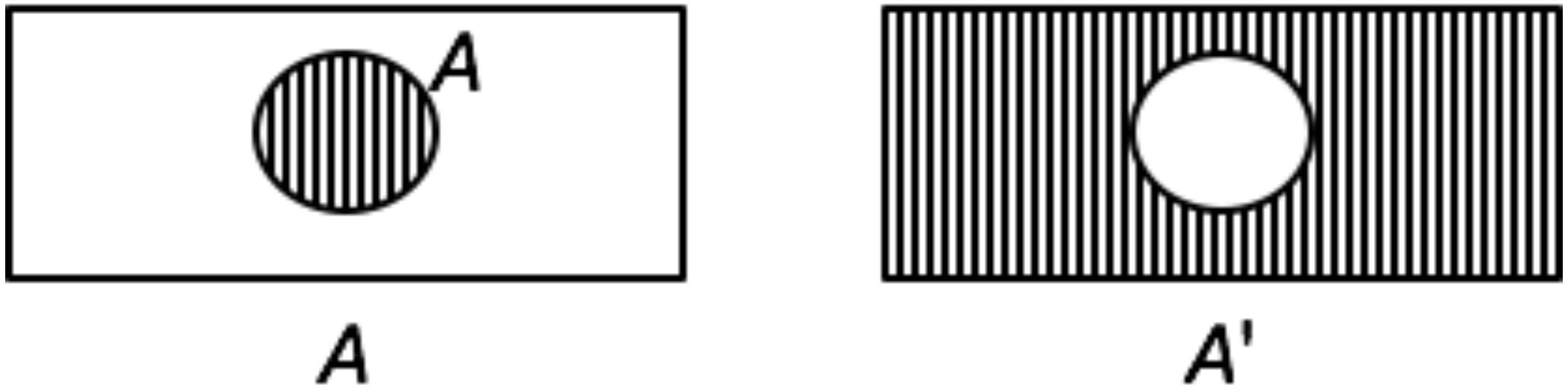
- Let A be a language on Σ
- Complement of A, denoted $\overline{A} = \{w \in \Sigma^* \mid w \notin A\}$



A A'

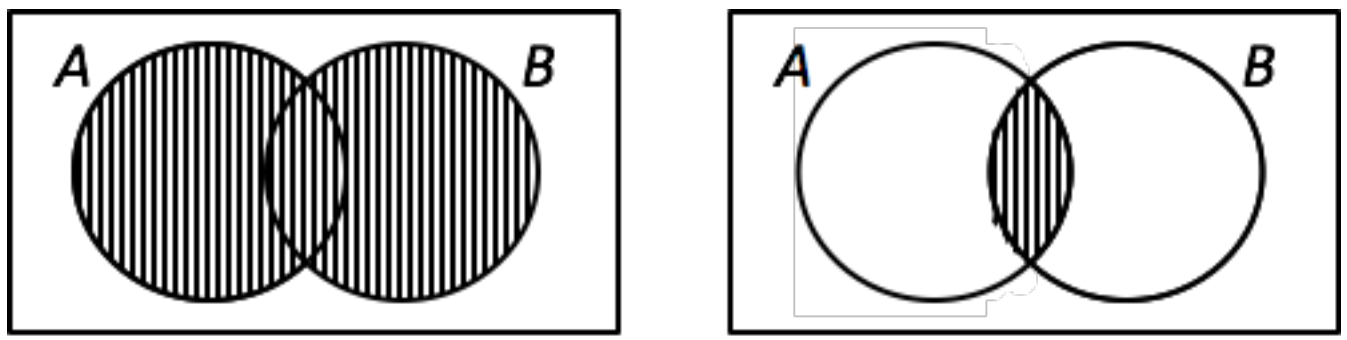
Closed Under Complement

- **Theorem.** The class of regular languages is closed under the complement operation.



Union and Intersection

- Let A and B be regular languages over Σ.
- Is $A \cup B$ regular?   Is $A \cap B$ regular?



$A \cup B$ $A \cap B$

Closed Under Intersection

Theorem. The class of regular languages is closed under the intersection operation.

Closed Under Union

Theorem. The class of regular languages is closed under the union operation.

Concatenation

- Let A and B be languages over Σ.
- **Definition.** Concatenation of A and B, denoted $A \circ B$ is defined as

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

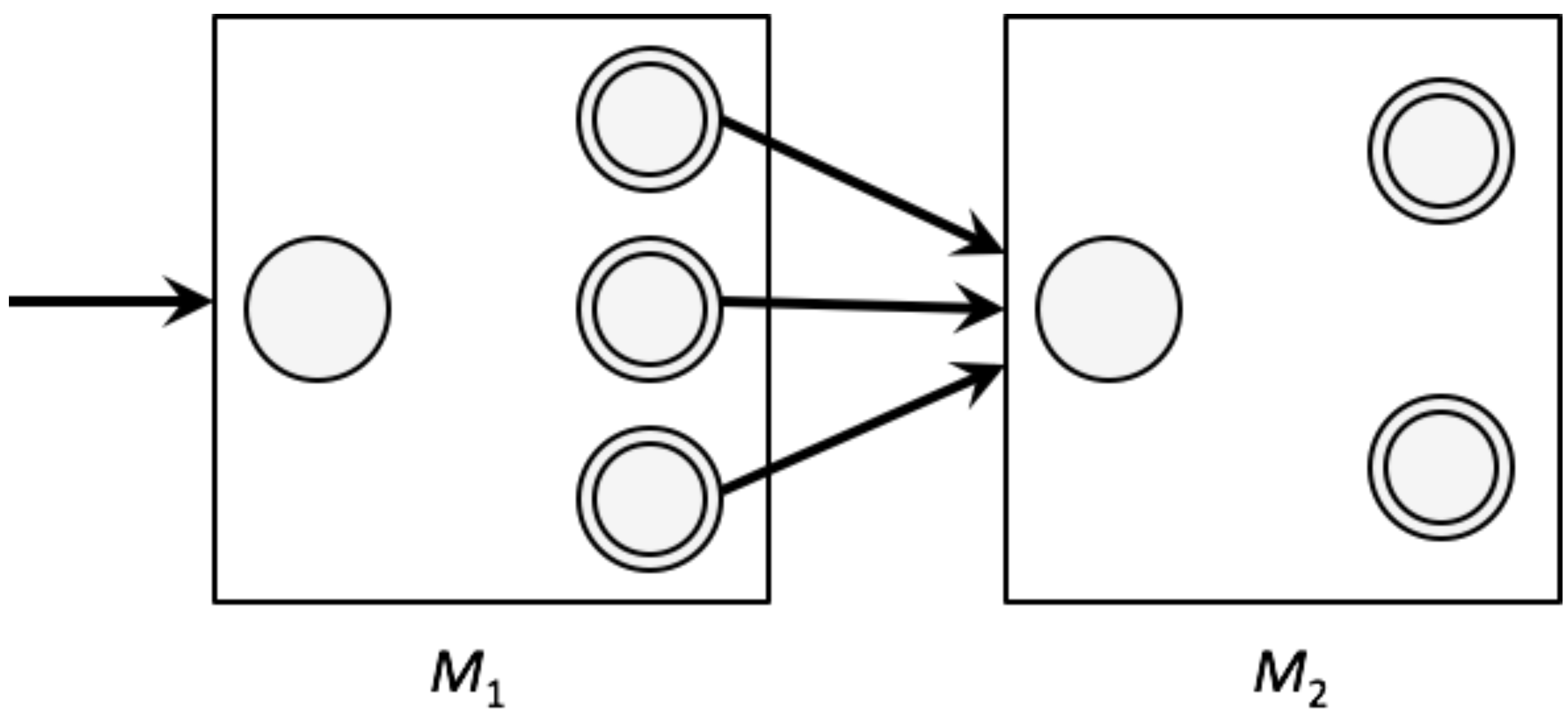
- **Question**. Are regular languages closed under concatenation?

Intuition: Closed Under Concatenation

- Let A and B be languages over Σ.
- **Definition.** Concatenation of A and B, denoted $A \circ B$ is defined as

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

- **Question**. Are regular languages closed under concatenation?

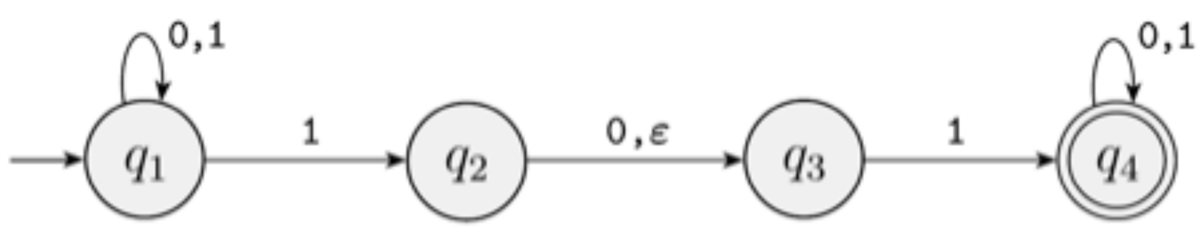


Non-deterministic Finite Automaton (NFA)

Formal Definition: NFA

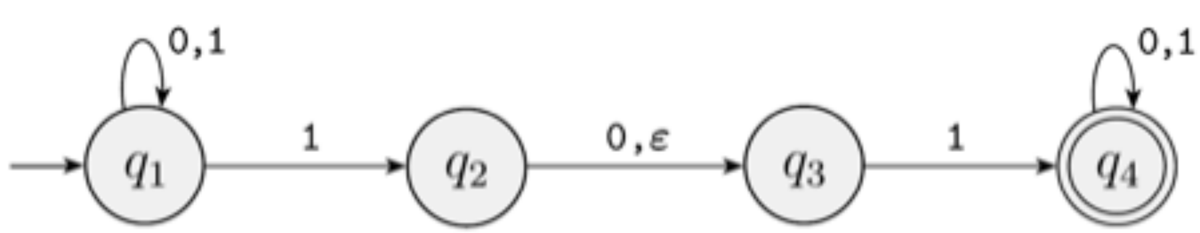
A non-deterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set called the **states**,
- Σ is a finite set called the **alphabet**,
- $\delta : Q \times \Sigma_\varepsilon \to \mathscr{P}(Q)$ is the transition function, where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- $q_o \in Q$ is the **start** state and $F \subseteq Q$ is the set of **accept** states.



NFA Computation

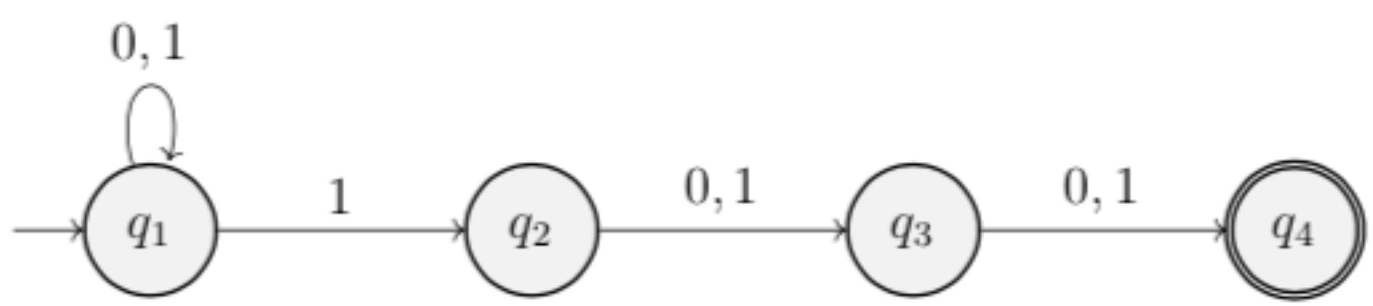
- Let $N = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$. Then N ***accepts*** w if there is a sequence of $r_0, r_1, \ldots, r_n$ in Q such that
  - $r_0 = q_0$
  - $r_{i+1} \in \delta(r_i, w_{i+1})$ for $i = 0, 1, \ldots, n-1$ and
  - $r_n \in F$



Nondeterminism is Your Friend

- Build an NFA to recognize the following language:
- $L = \{w \mid w \in \{0,1\}^*$ and has a 1 in the 3rd position from the end$\}$

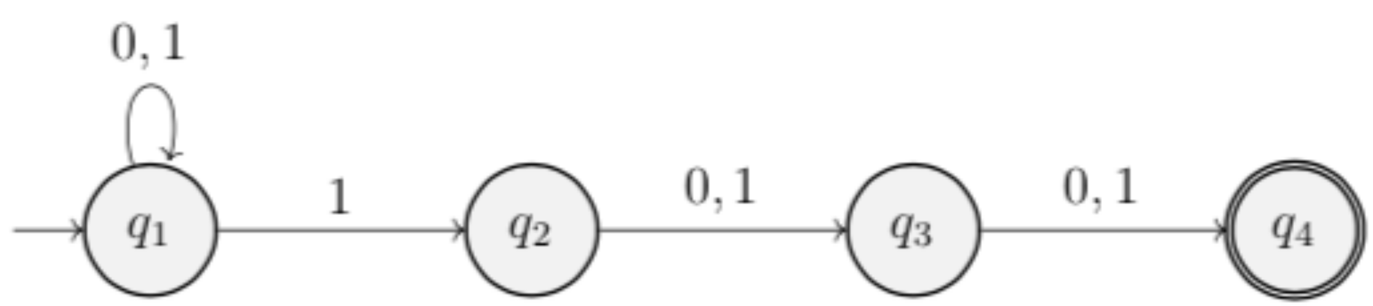
NFA



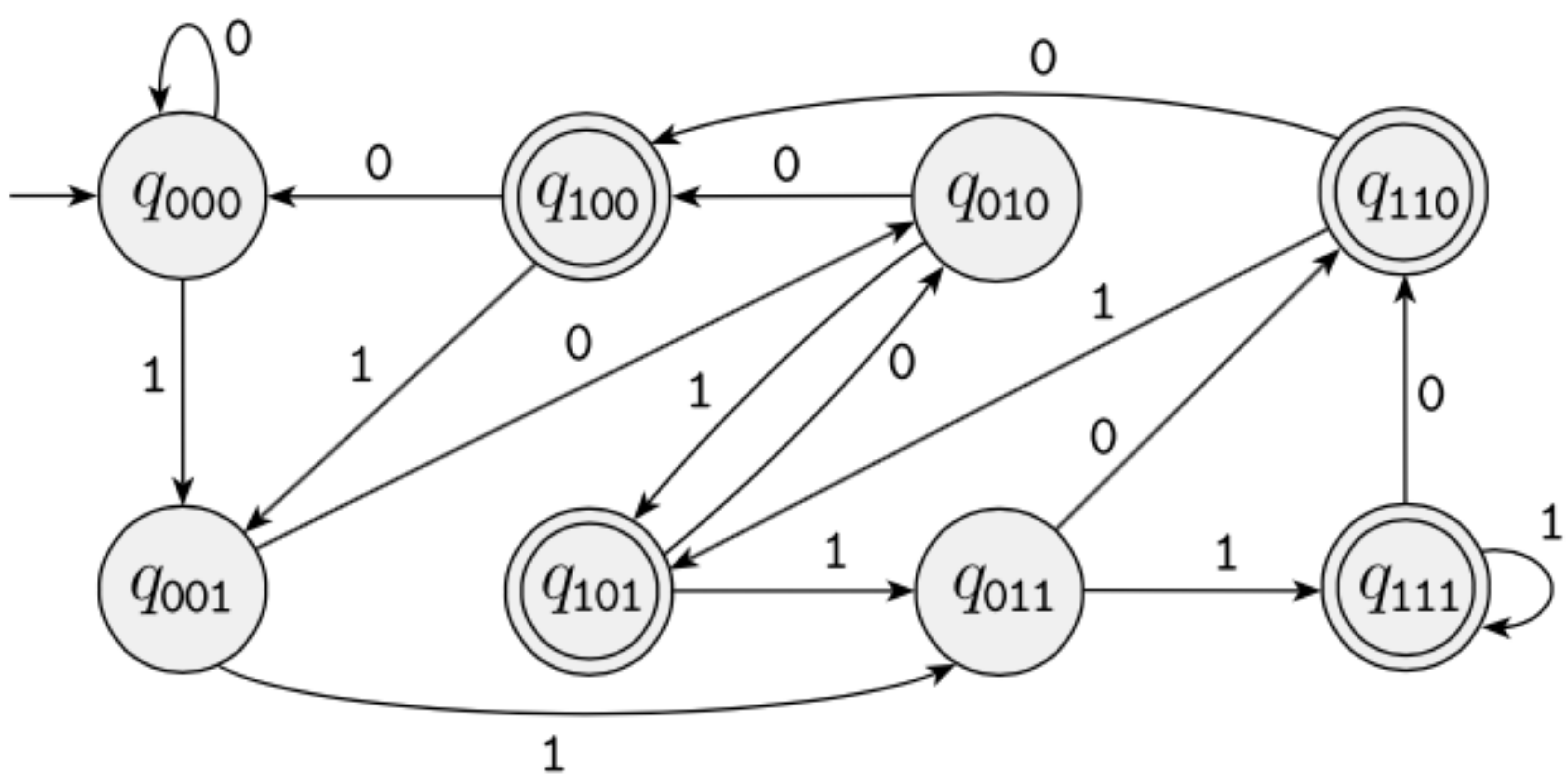
Nondeterminism is Your Friend

- Build an NFA to recognize the following language:
- $L = \{w \mid w \in \{0,1\}^*$ and has a 1 in the 3rd position from the end$\}$

NFA

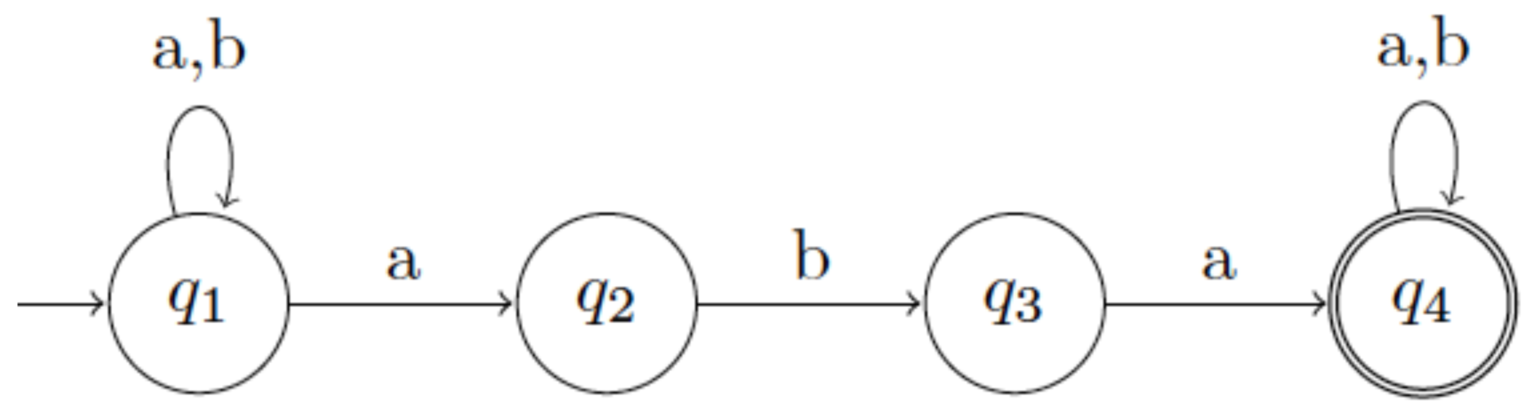


DFA



Another Example

- What is the language recognized by this NFA?



DFA $\Longleftrightarrow$ NFA
Equivalence

Equivalence

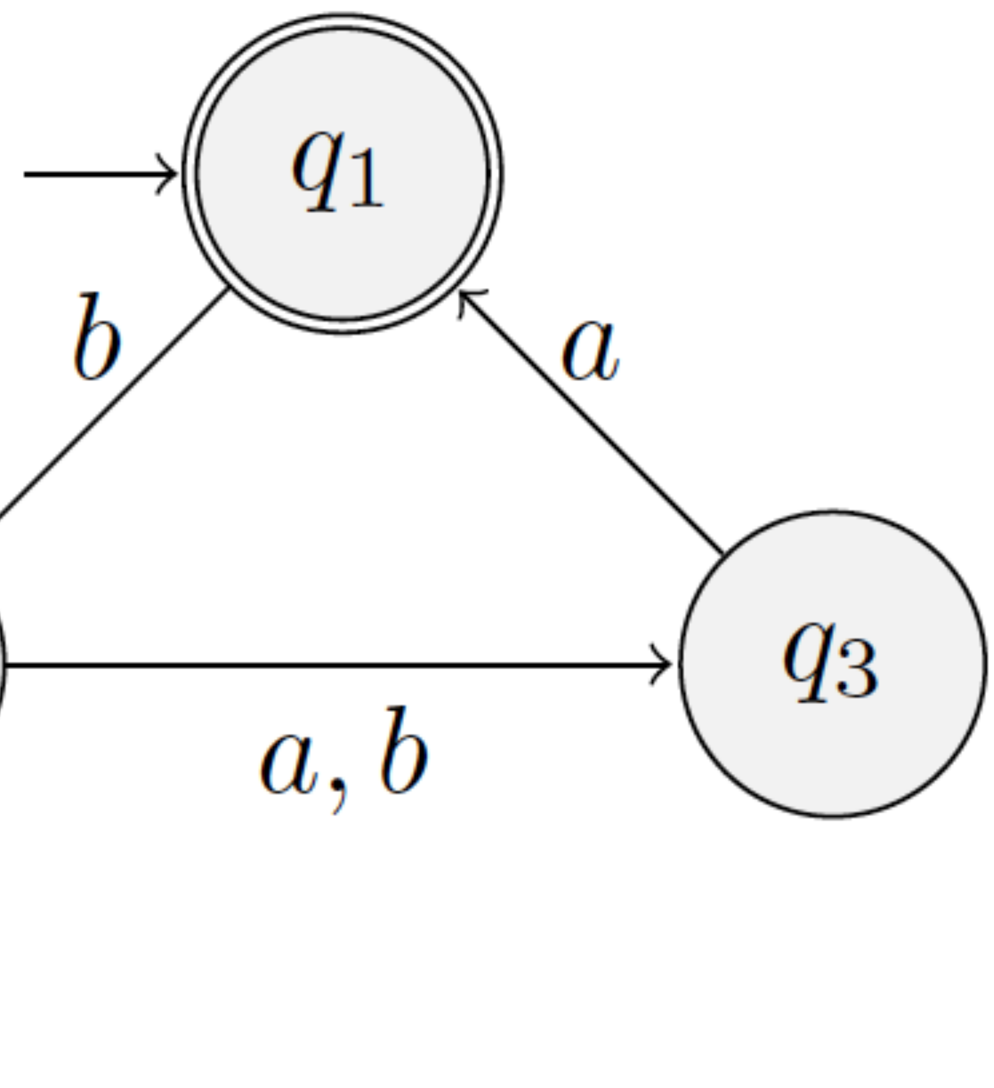
- **Definition.** Two machines are equivalent if they recognize the same language.

- **Theorem.** Given any NFA N there exists an equivalent DFA M and vice versa.
  - One direction is easy: every DFA is also an NFA by definition.
  - Need to show can construct a DFA M such that $L(M) = L(N)$

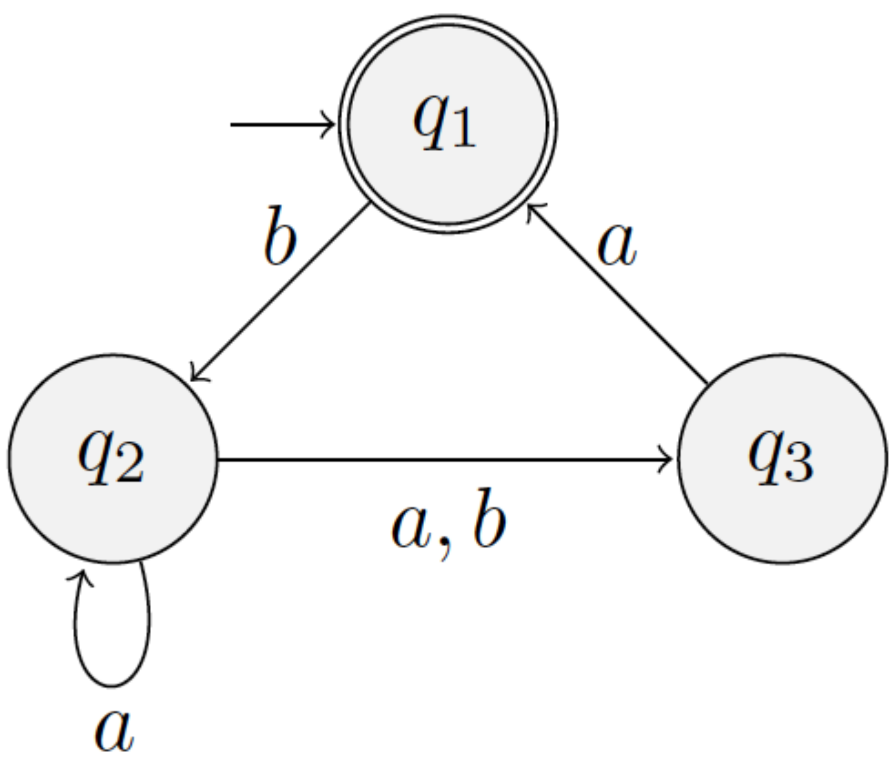
Creating an Equivalent DFA

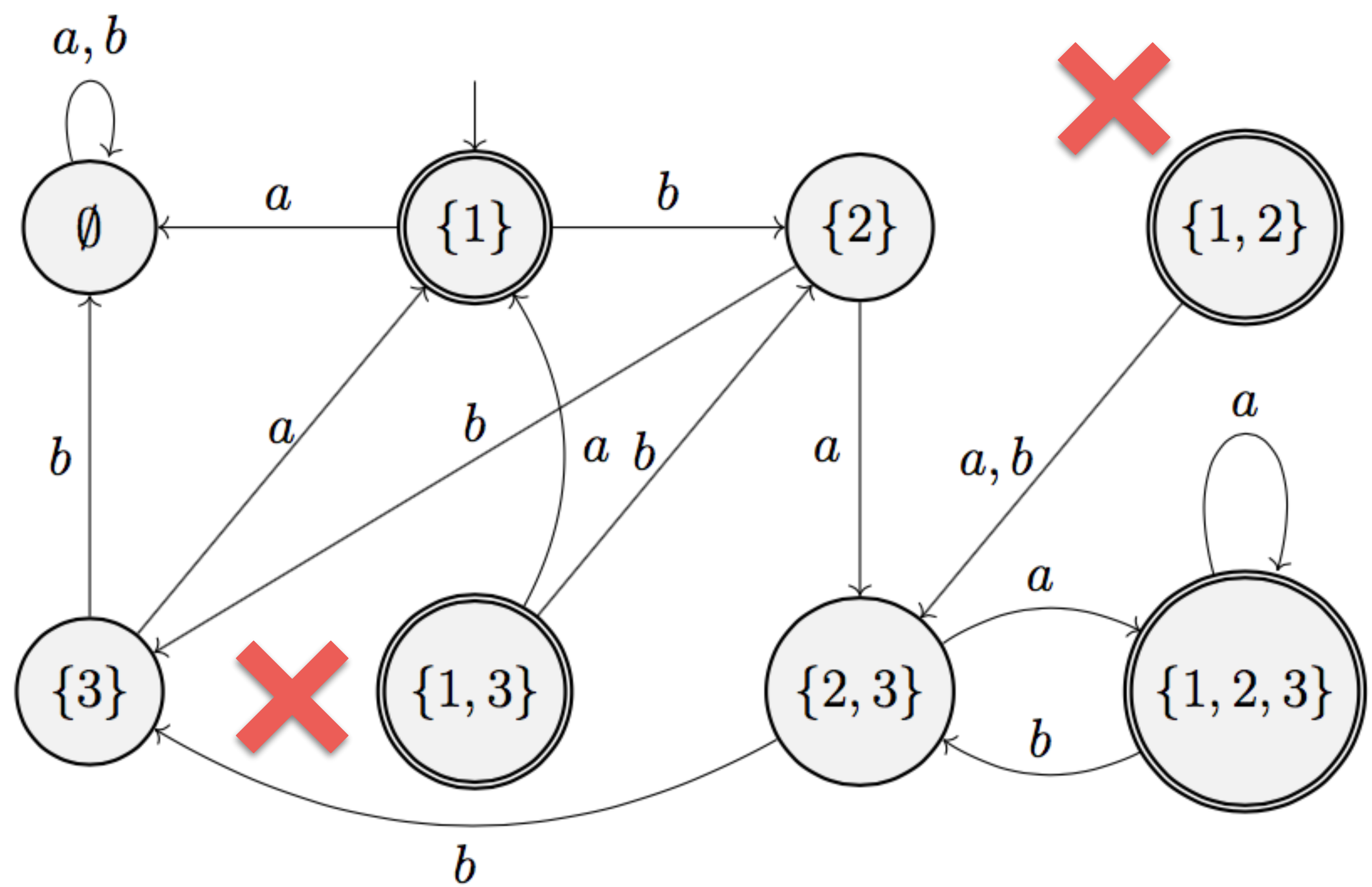
- **Theorem.** Given any NFA $N = (Q, \Sigma, \delta, q, F)$ there exists an equivalent DFA M.
- **Proof outline:** M "simulates" N by having a larger state space
  - If N has k states, M will have 2^k states to account for any possible subset of N's states
- In particular, $Q_M = \mathscr{P}(Q)$
- First, let's ignore ε transitions
- How can M simulate N?

Example: Equivalent DFA?



Example: Equivalent DFA?

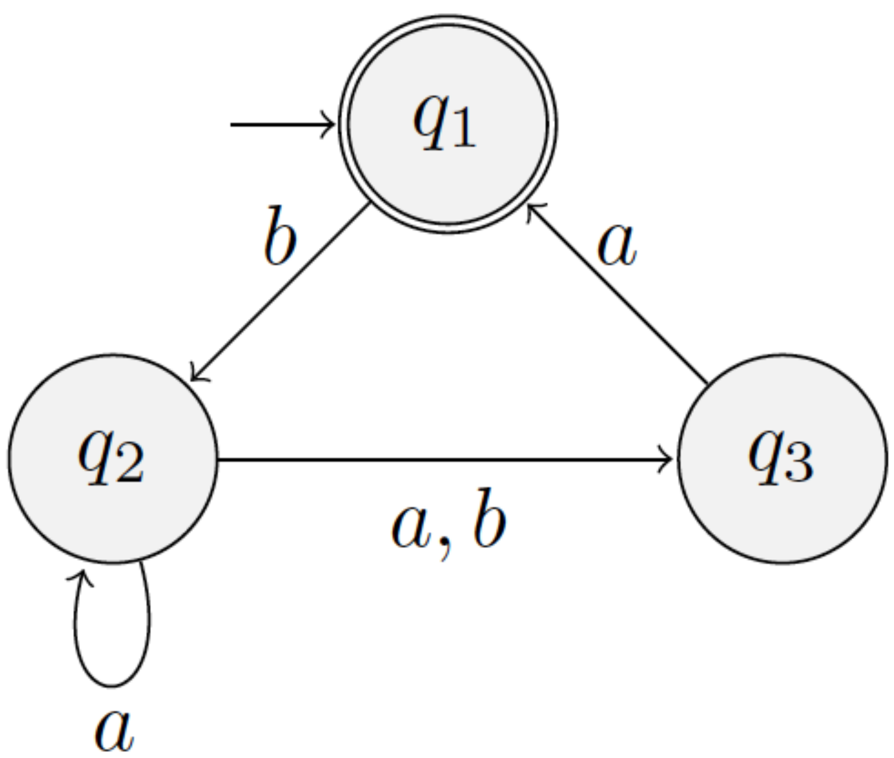


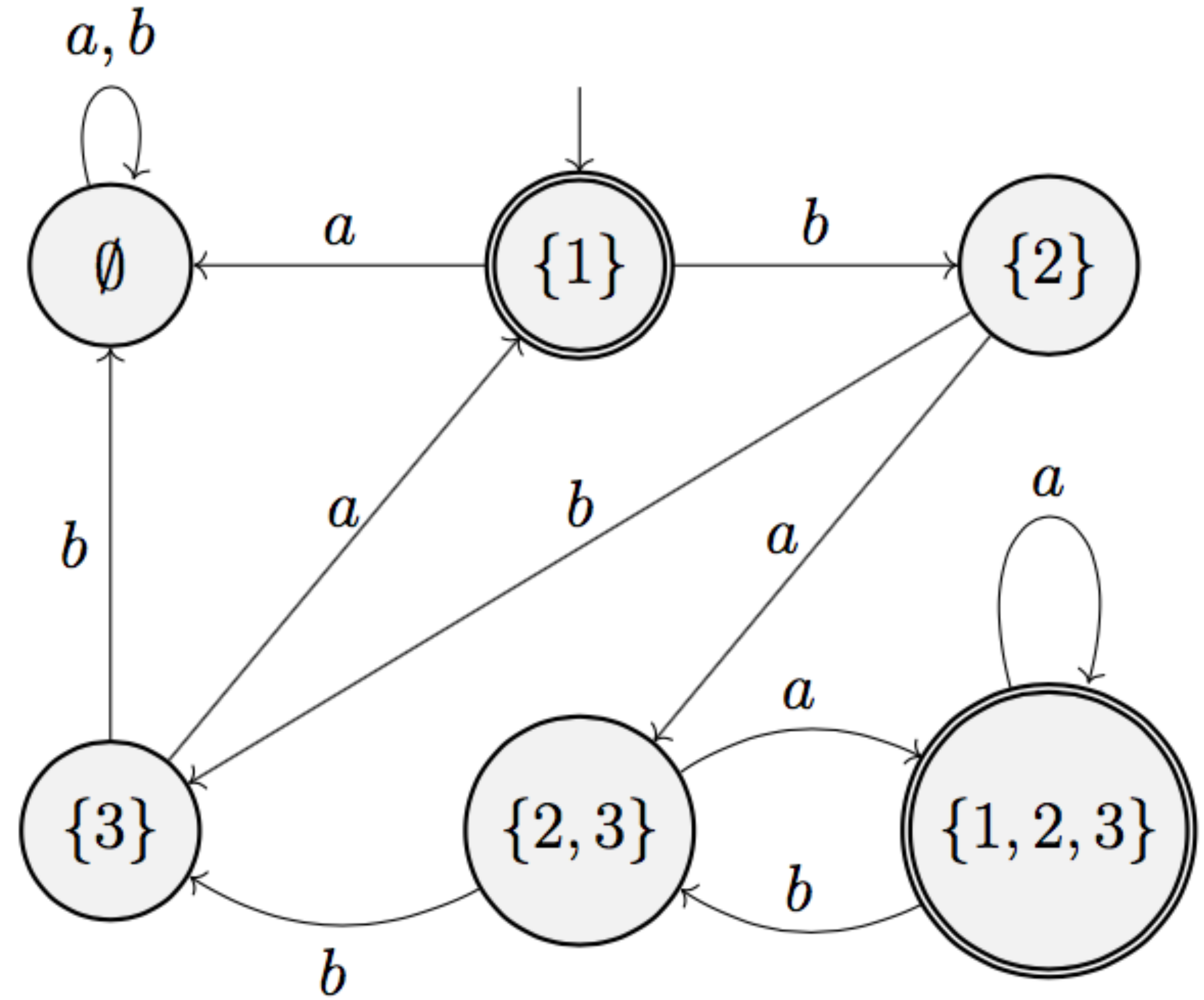


Creating an Equivalent DFA

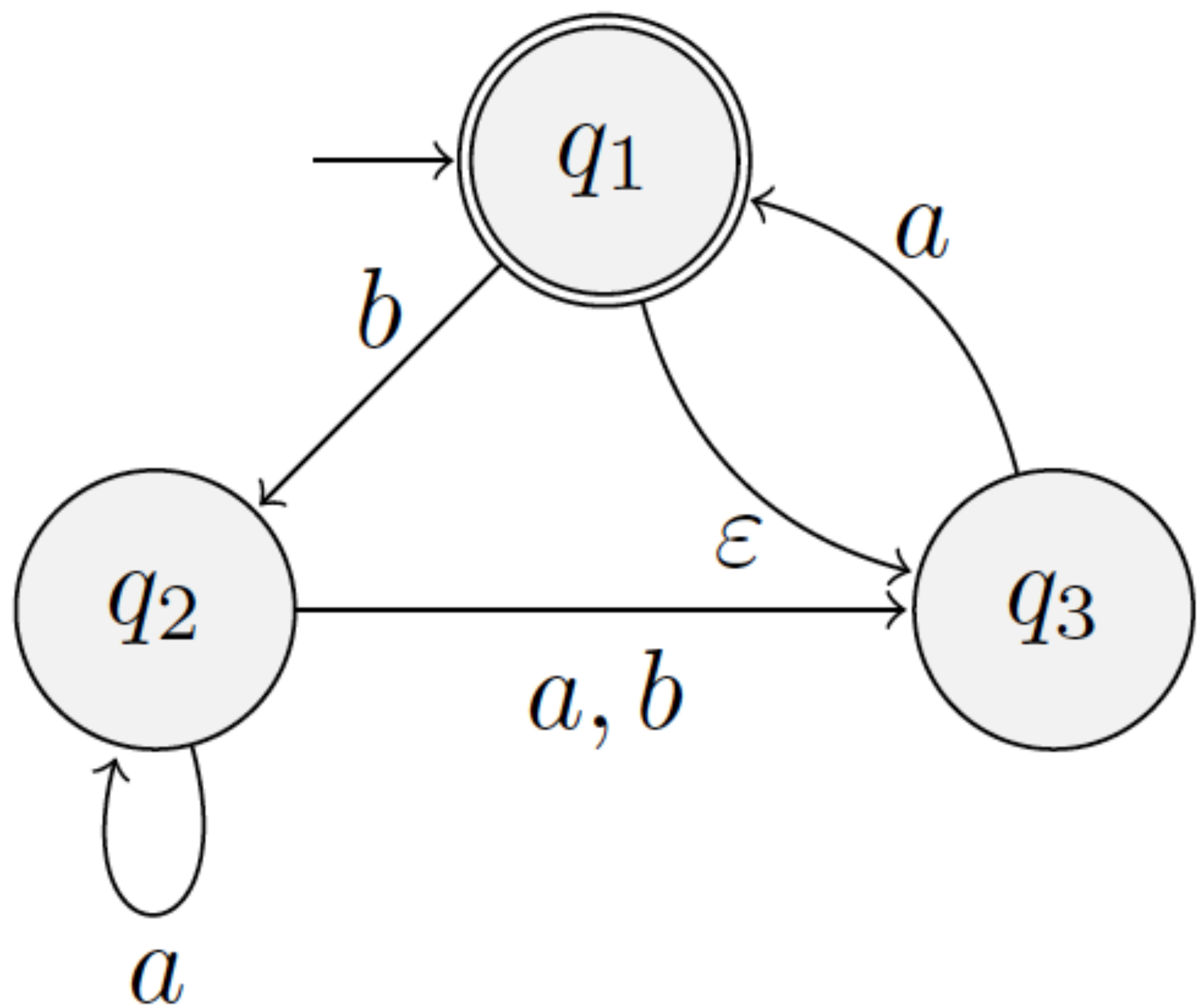
- **Theorem.** Given any NFA $N = (Q, \Sigma, \delta, q, F)$ there exists an equivalent DFA M.
- **Proof.** $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ where
  - $Q_M = \mathscr{P}(Q)$
  - $q_M = \{q\}$
  - $\delta_M(R, a) = \cup_{q \in R}\, \delta(r, a)$ for any $R \in Q_M, a \in \Sigma$
  - $F_M = \{R \in Q \mid R \cap F \neq \emptyset\}$ (any "set" of states that contains an accept state of N)
- ***Correctness:*** $w \in L(N) \iff w \in L(M)$

Example: Equivalent DFA?





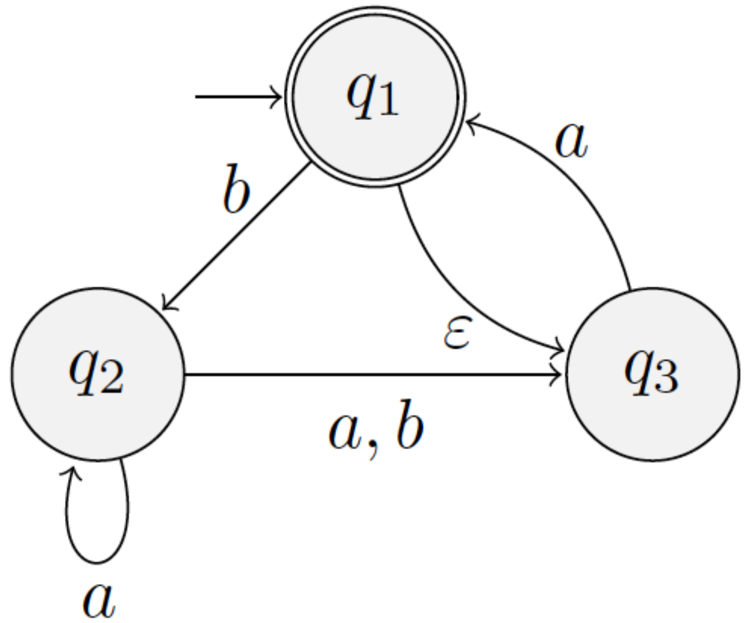
What about ε transitions?

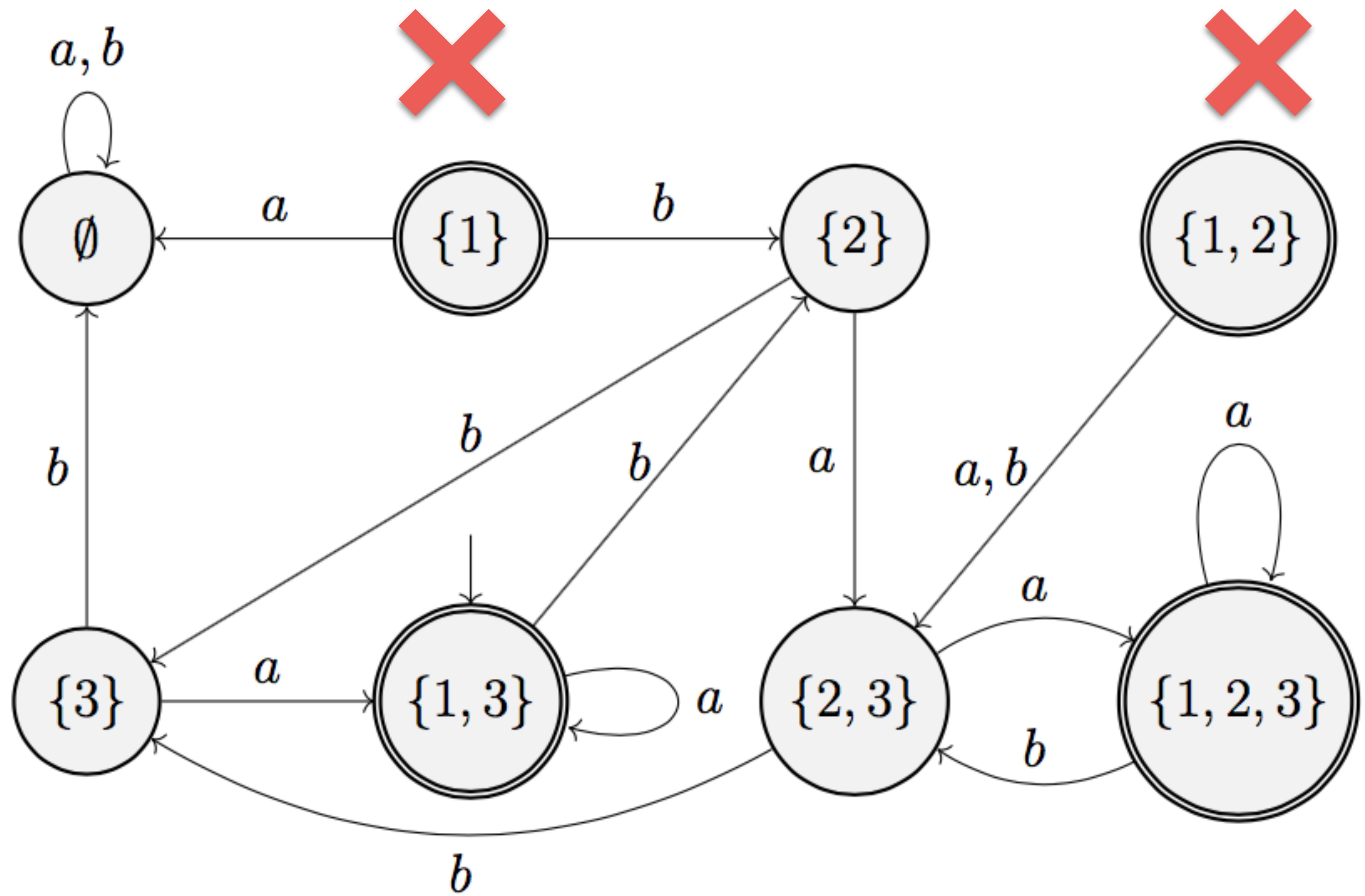


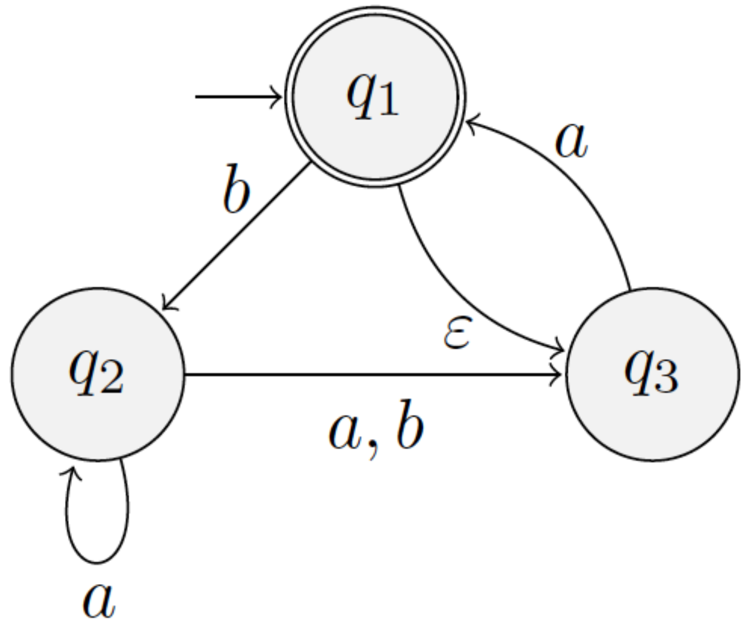
Creating an Equivalent DFA

- **Theorem.** Given any NFA $N = (Q, \Sigma, \delta, q, F)$ there exists an equivalent DFA M.
- **Proof.** $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ where $Q_M = \mathscr{P}(Q)$ and $F_M = \{R \in Q \mid R \cap F \neq \emptyset\}$ as before.
- **Definition.** (ε-closure) $E(R) = \{q \in Q \mid q$ can reached from any state in R along zero or more ε transitions $\}$
  - Notice that $R \subseteq E(R)$ and $E(R) \in Q_M$
- Now we can define the start state of M as: $q_M = E(\{q\})$
- Transition function $\delta(R, a) = \cup_{r \in R} E(\delta(r, a))$ for any $R \in Q_M, a \in \Sigma$

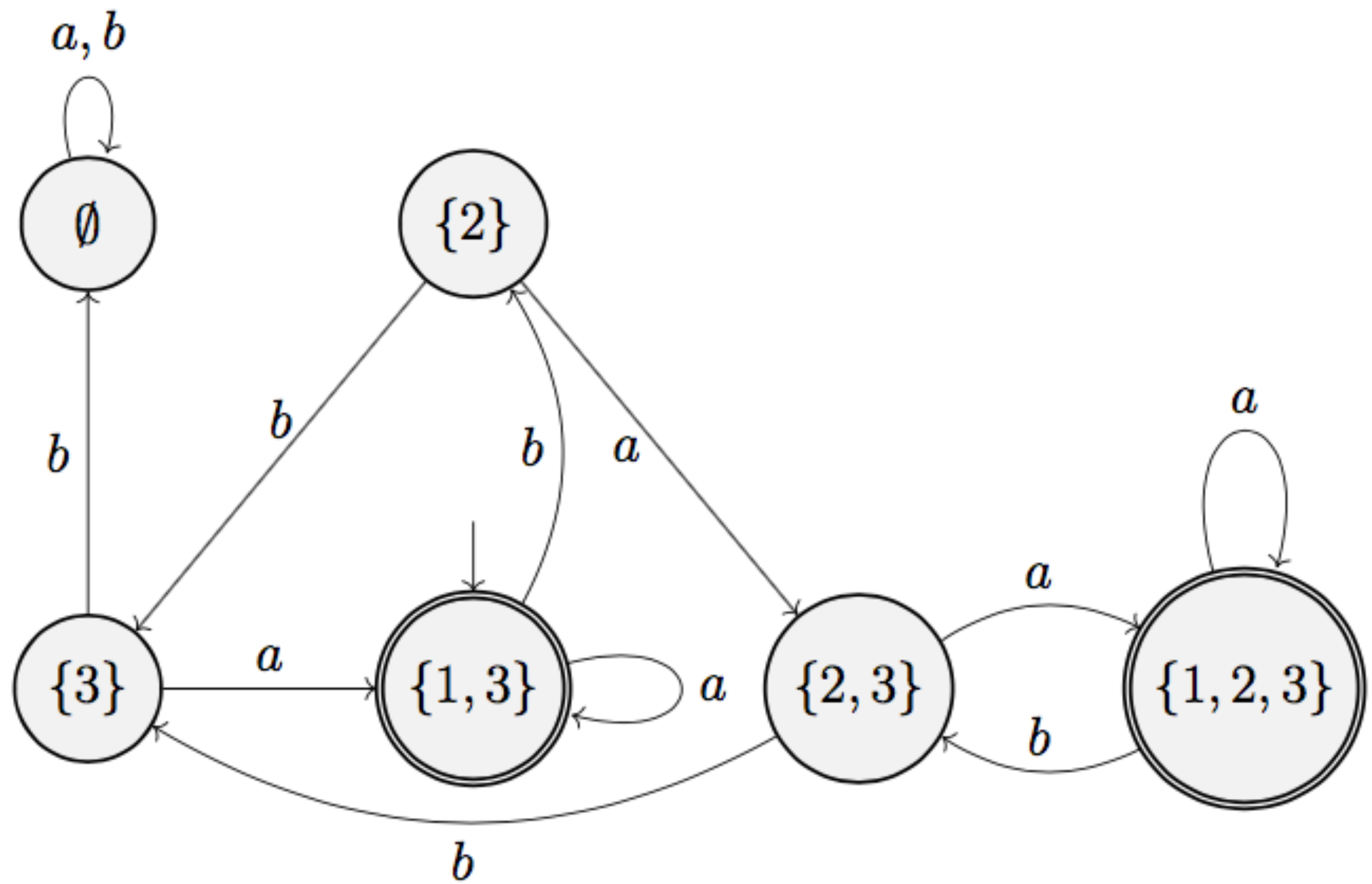
Equivalent DFA







Equivalent DFA



Alternate Definition of Regular Languages

- **Corollary.** A language is regular iff some NFA recognizes it.