CSCI 361 Lecture 4: Nondeterministic Finite Automata

Shikha Singh

Announcements & Logistics

- **HW 1** due tonight at 10 pm via Gradescope
- Hand in reading assignment #3
- No reading assignment for next lecture:
  - we are still catching up to the readings
- **Thursday's lecture is cancelled**: I'll be at Tapia @ San Diego
  - Planned topic: regular expressions
  - Read from the book and slides
  - Answer HW 2 questions based on reading
- HW 2 will be released today
  - Due next Wed (Sept 25)
- **Questions?**

Overview So Far

- First model of computation: finite automata
- "Expressive"/computational power of finite automata:
  - Solves/recognizes the class of languages: regular languages
- Will look at two "equivalent" models:
  - Non-deterministic finite automata
  - Regular expressions (used to generate regular languages)
  - DFA $\Longleftrightarrow$ NFA $\Longleftrightarrow$ regular expression
- Last segment: will prove some languages are not regular
  - Show recognizes them requires infinite states

Today

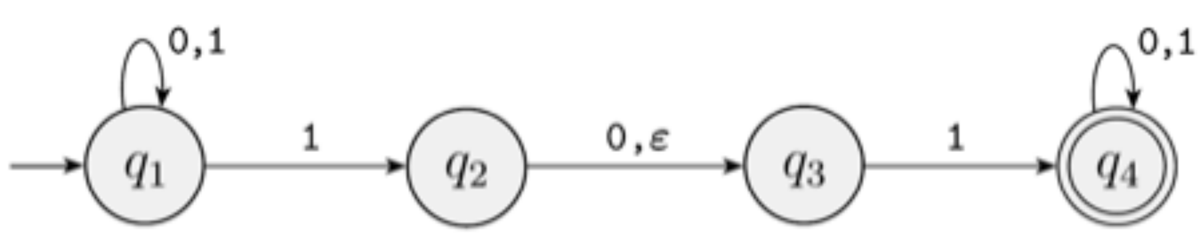
- More practice with designing NFAs
- Plan for Thursday's lecture: show DFA $\Longleftrightarrow$ NFA
  - Instead read "subset construction" from the textbook
  - Answer question on it in HW 2
  - Will review next Tuesday how this equivalence works
- Move on to regular expressions today

Non-deterministic Finite Automaton (NFA)

Formal Definition: NFA

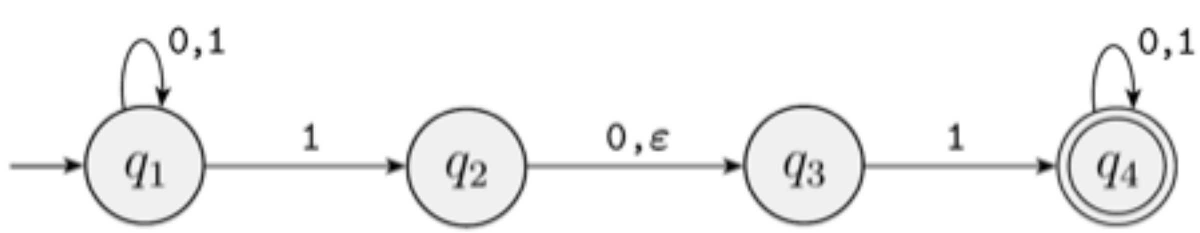
A non-deterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- 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

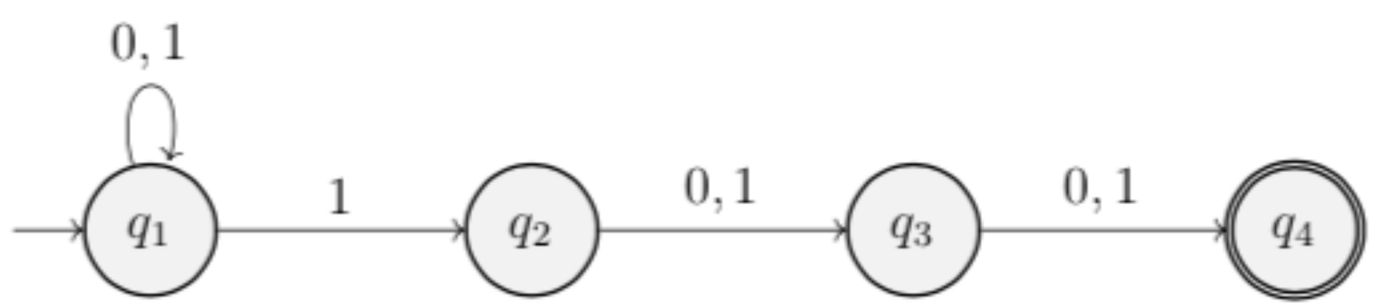
- Let $N = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$. Then N ***accepts*** w if there is a sequence of $r_0, r_1, \ldots, r_n$ in Q such that
  - $r_0 = q_0$
  - $r_{i+1} \in \delta(r_i, w_{i+1})$ for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

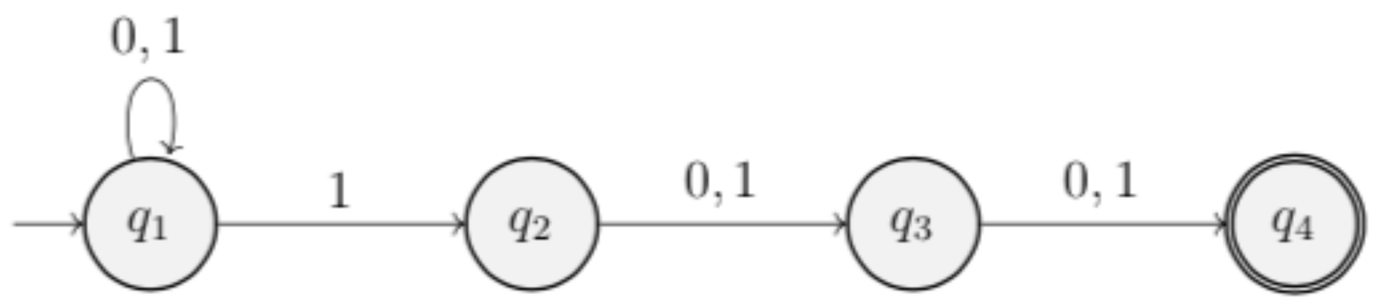
NFA



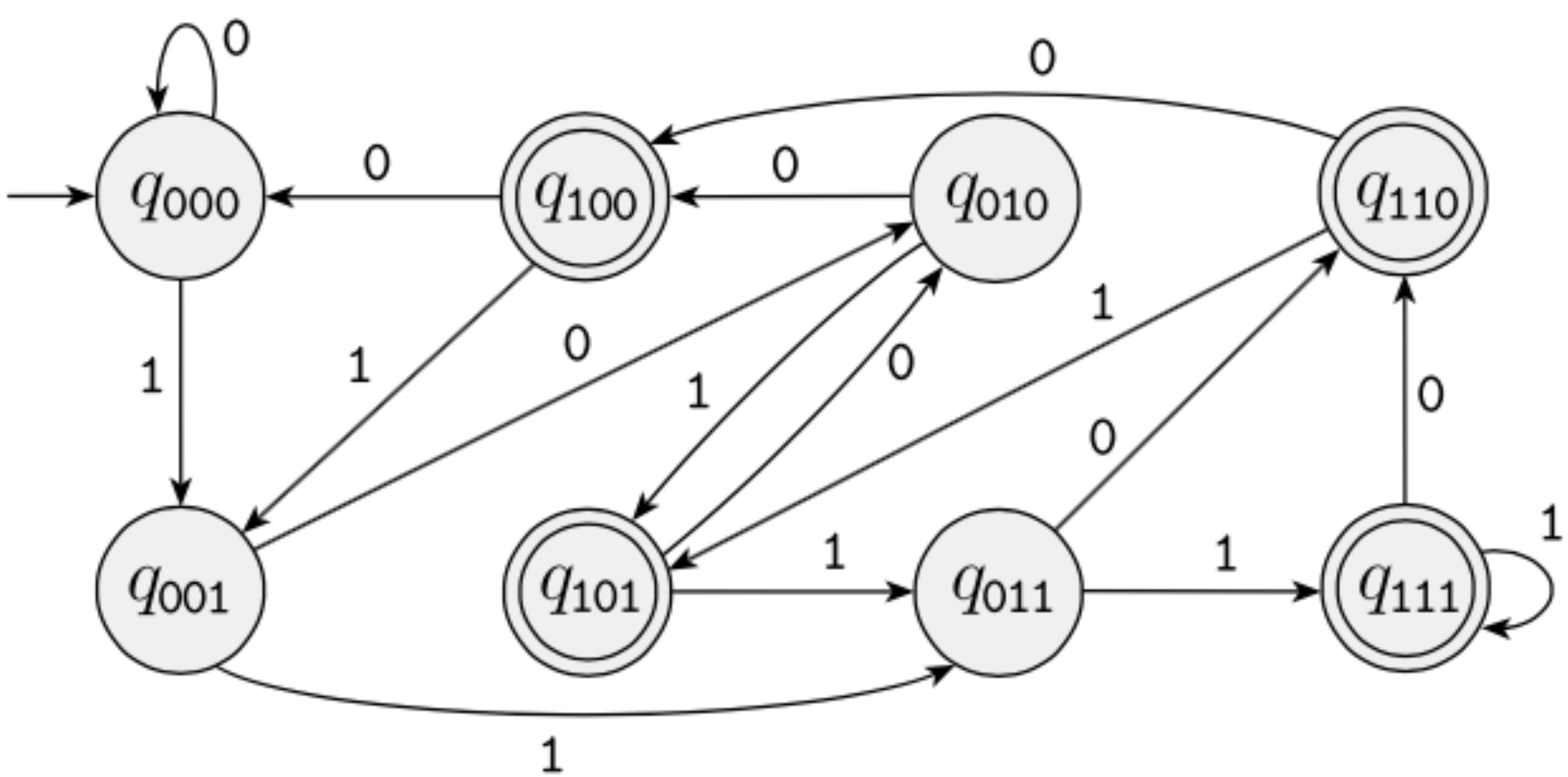
Nondeterminism is Your Friend

- 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

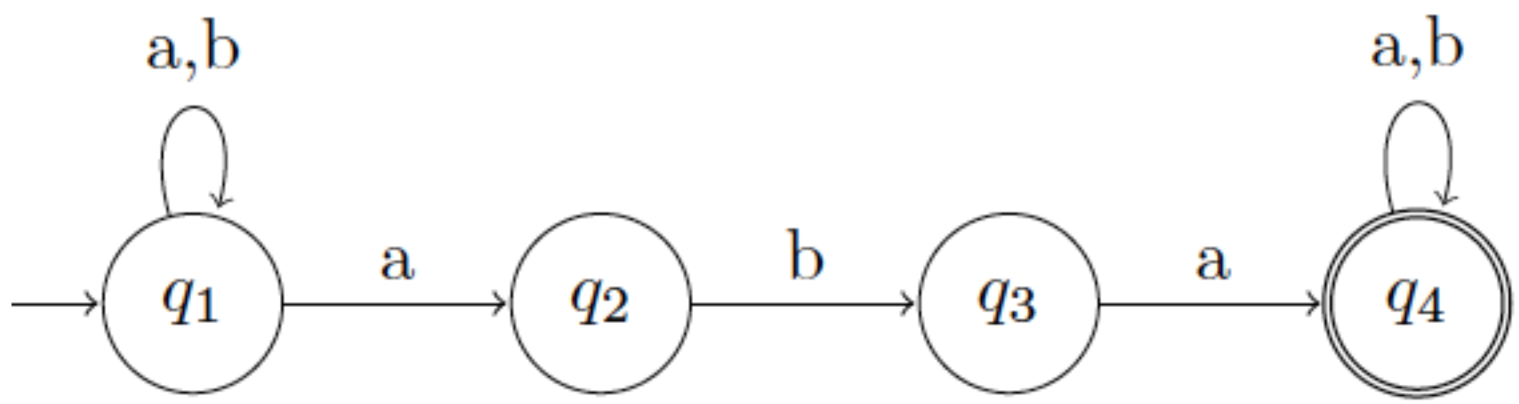


DFA



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 $L(M) = L(N)$

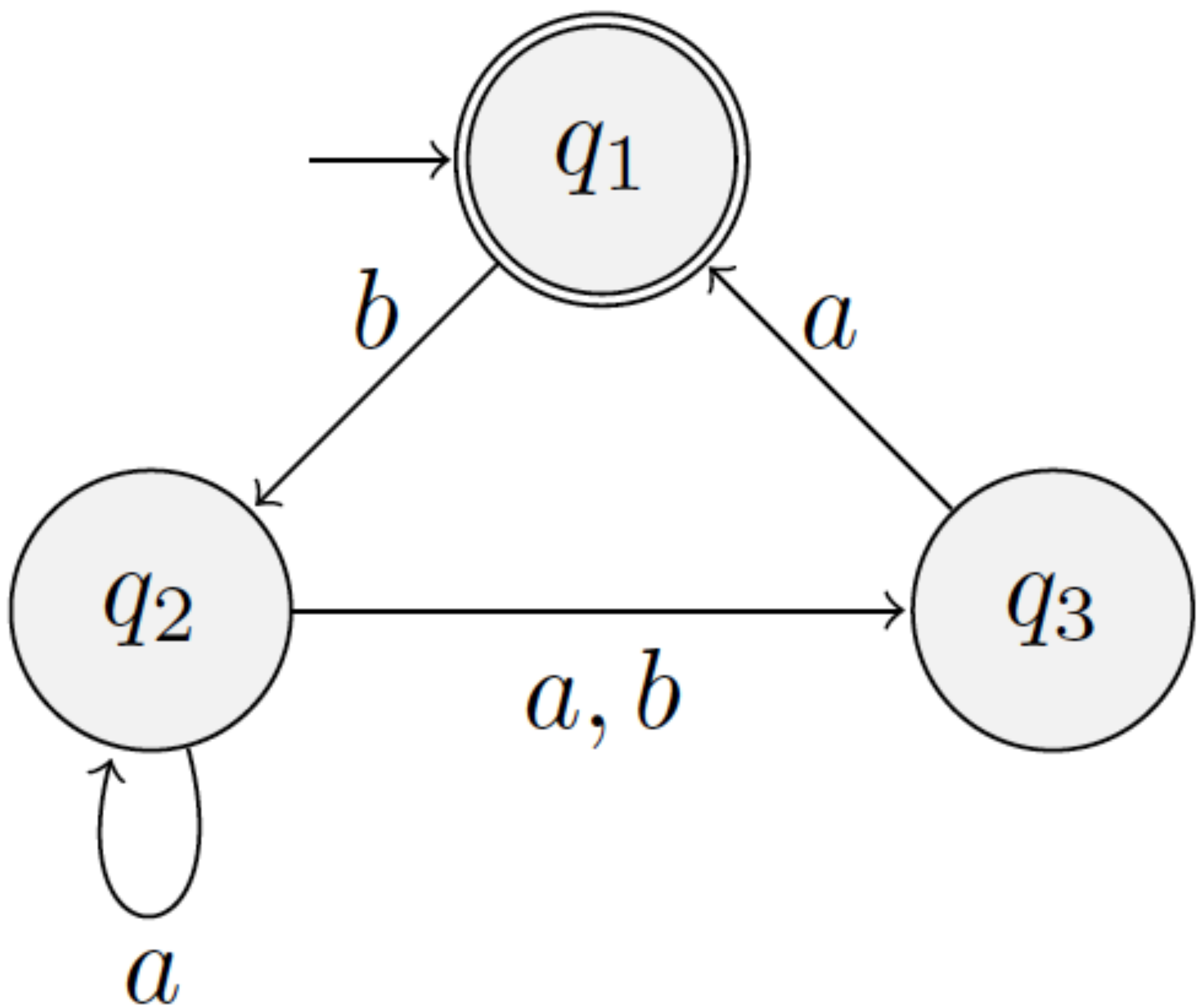
Creating an Equivalent DFA

- **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?

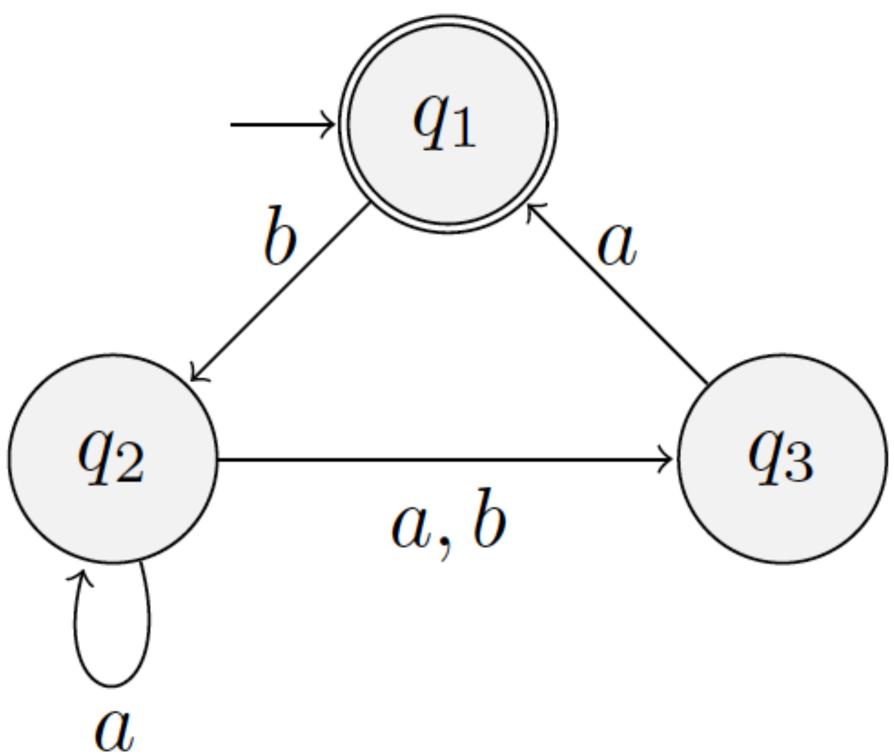
Creating an Equivalent DFA

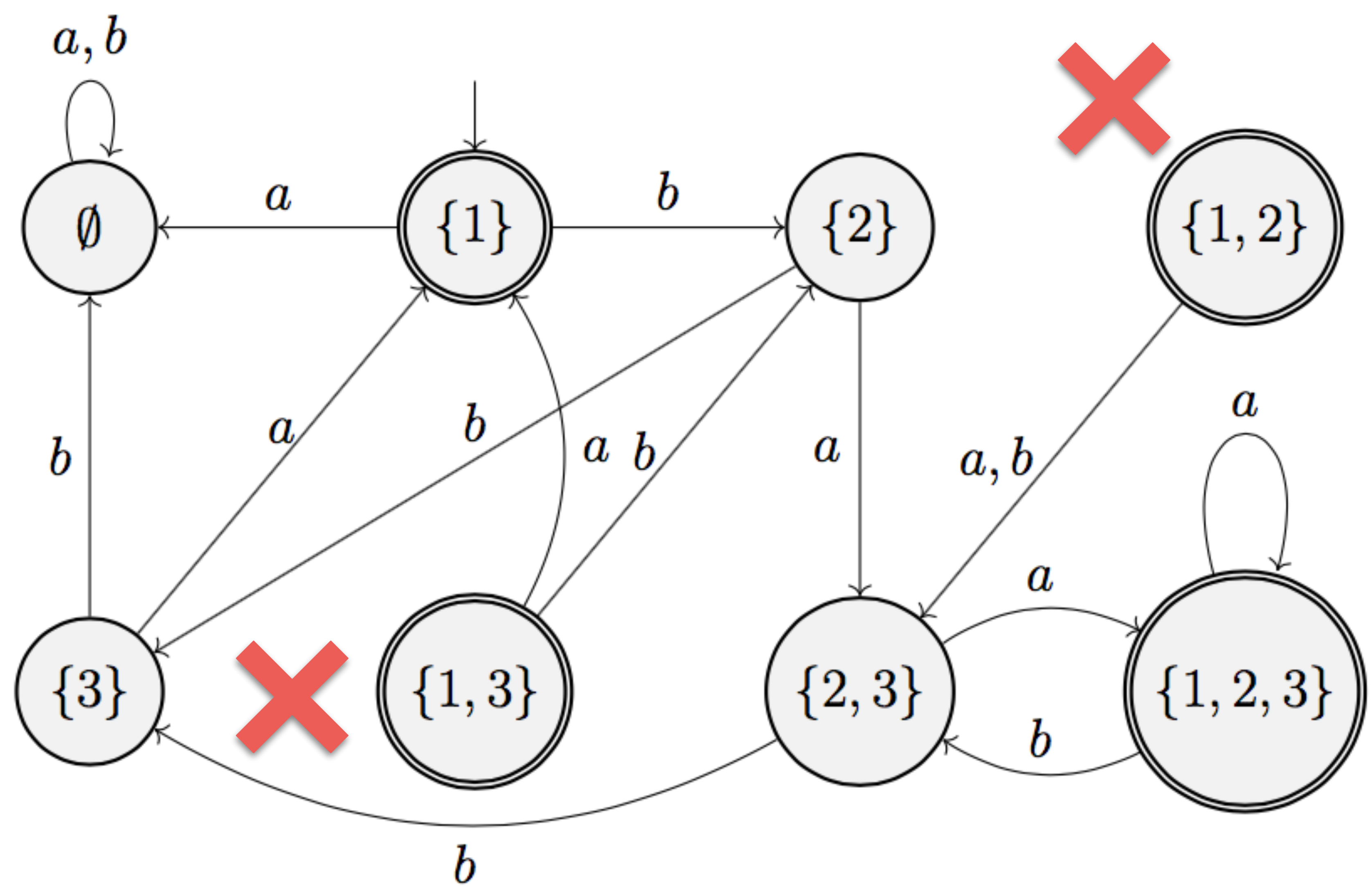
- **Theorem.** Given any NFA $N = (Q, \Sigma, \delta, q, F)$ there exists an equivalent DFA M.
- **Proof.** $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ where
  - $Q_M = \mathscr{P}(Q)$
  - $q_M = \{q\}$
  - $\delta_M(R, a) = \cup_{q \in R}\, \delta(r, a)$ for any $R \in Q_M, a \in \Sigma$
  - $F_M = \{R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

Example: Equivalent DFA?

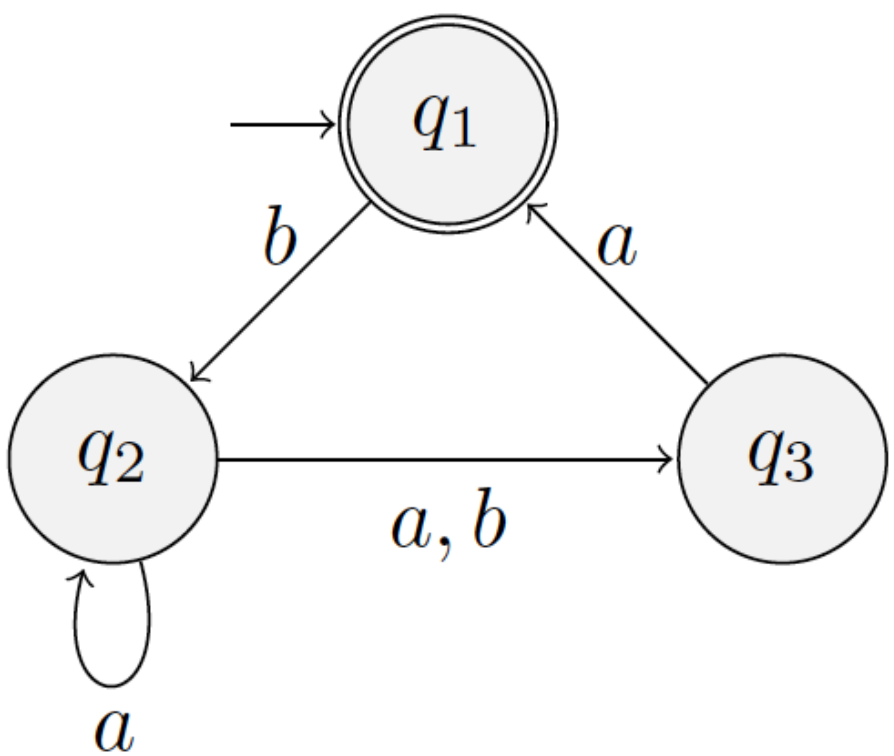


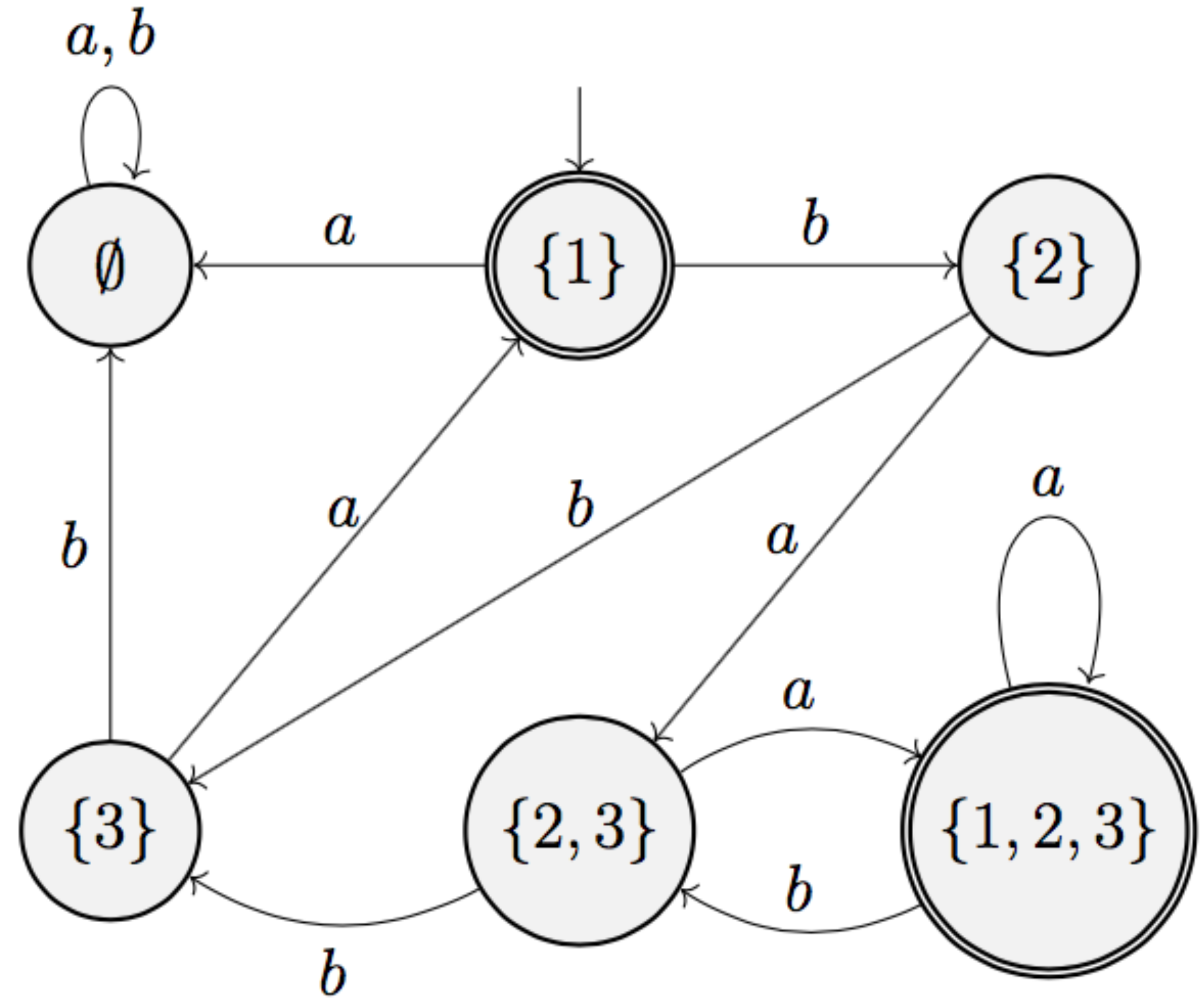
Example: Equivalent DFA?



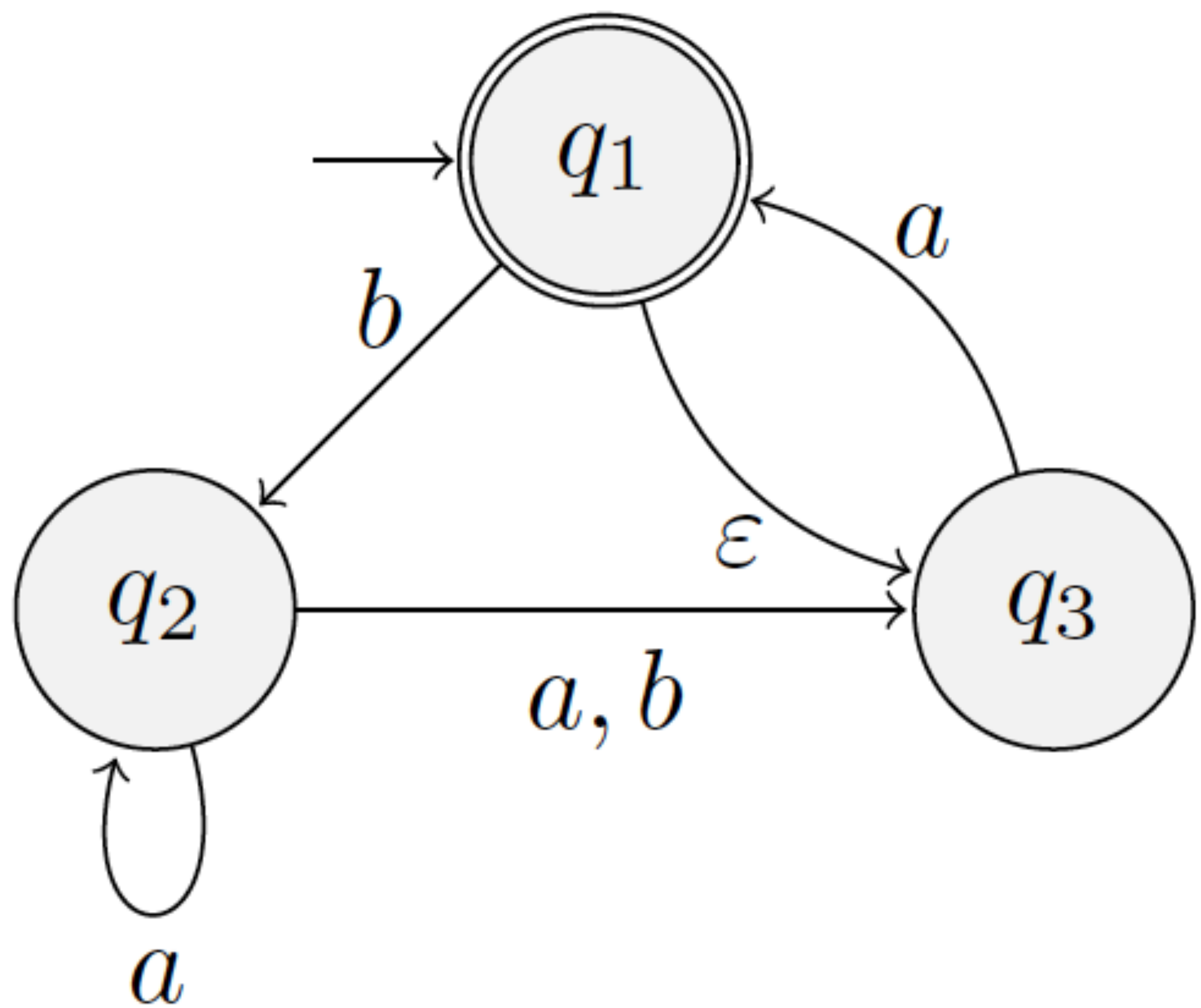


Example: Equivalent DFA?





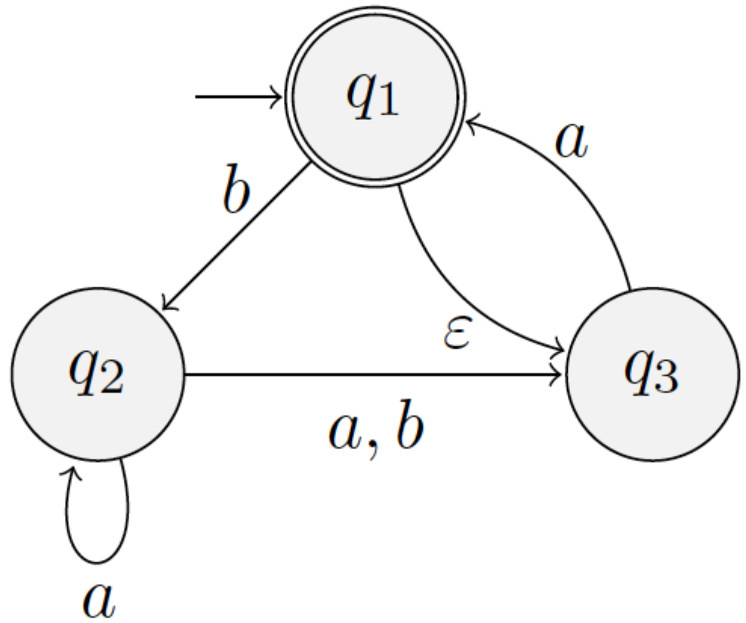
What about ε transitions?

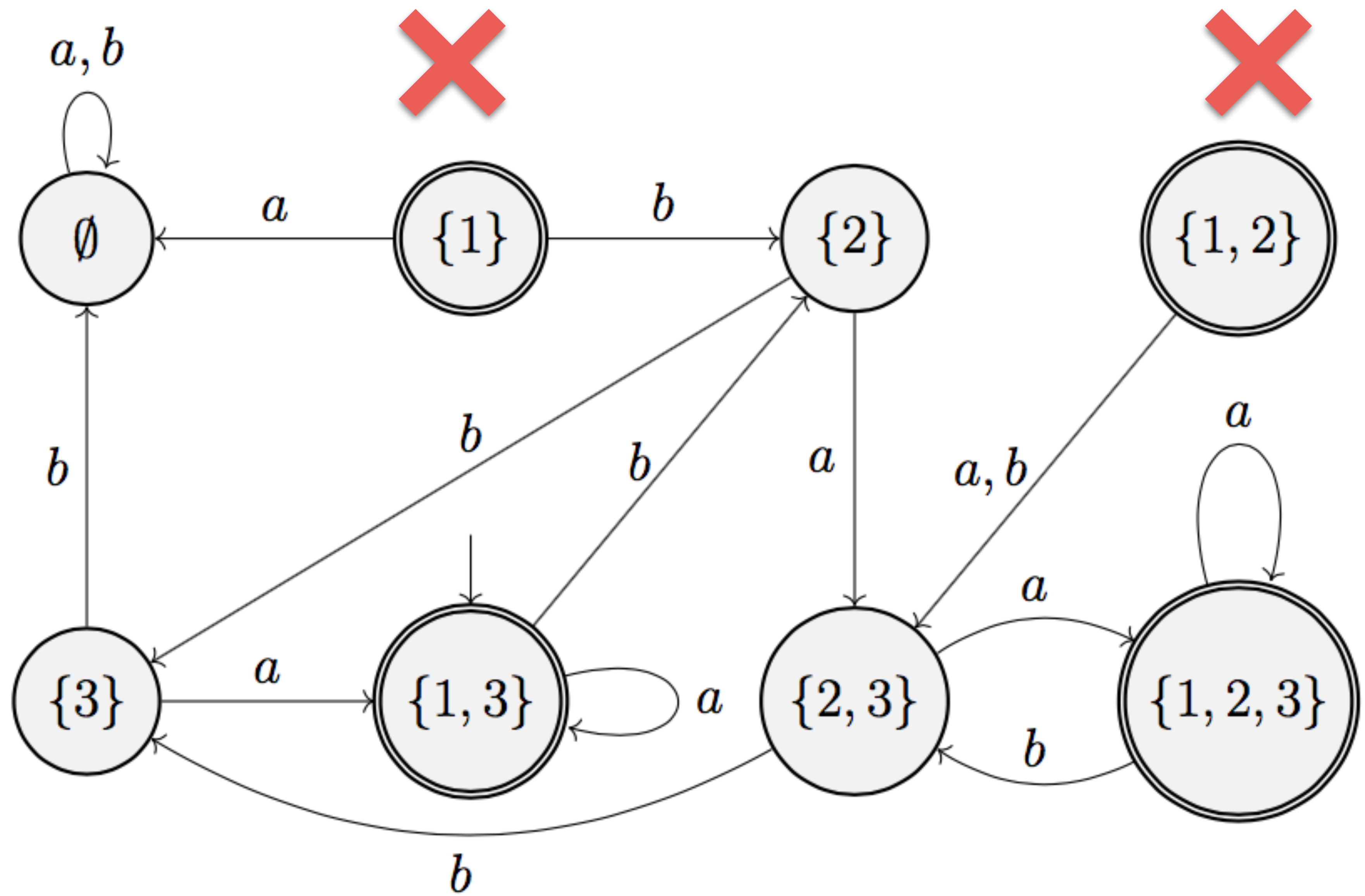


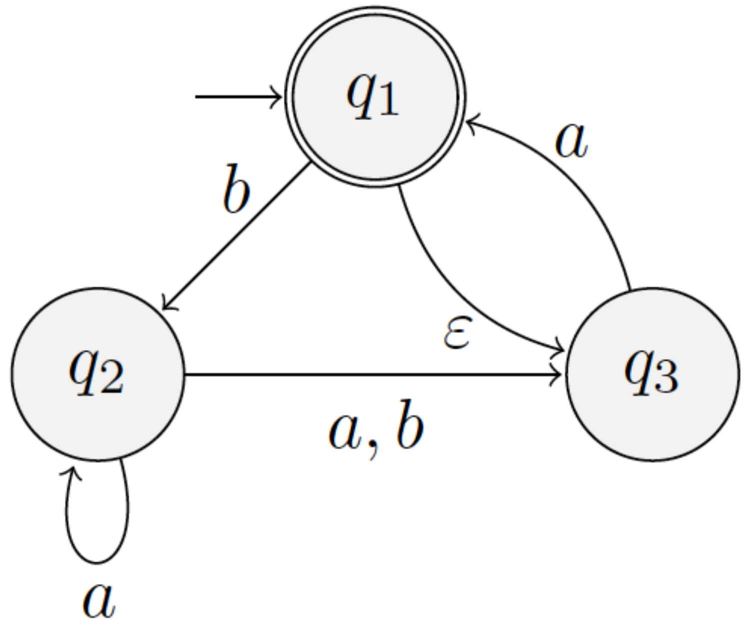
Creating an Equivalent DFA

- **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) $E(Q) = \{q \in Q \mid q$ can reached from any state in R along zero or more ε transitions $\}$
  - Notice that $R \subseteq E(Q)$ and $E(Q) \in Q_M$
- Now we can define the start state of M as: $q_M = E(\{q\})$
- Transition function $\delta(R, a) = \cup_{r \in Q} E(\delta(r, a))$ for any $R \in Q_M, a \in \Sigma$

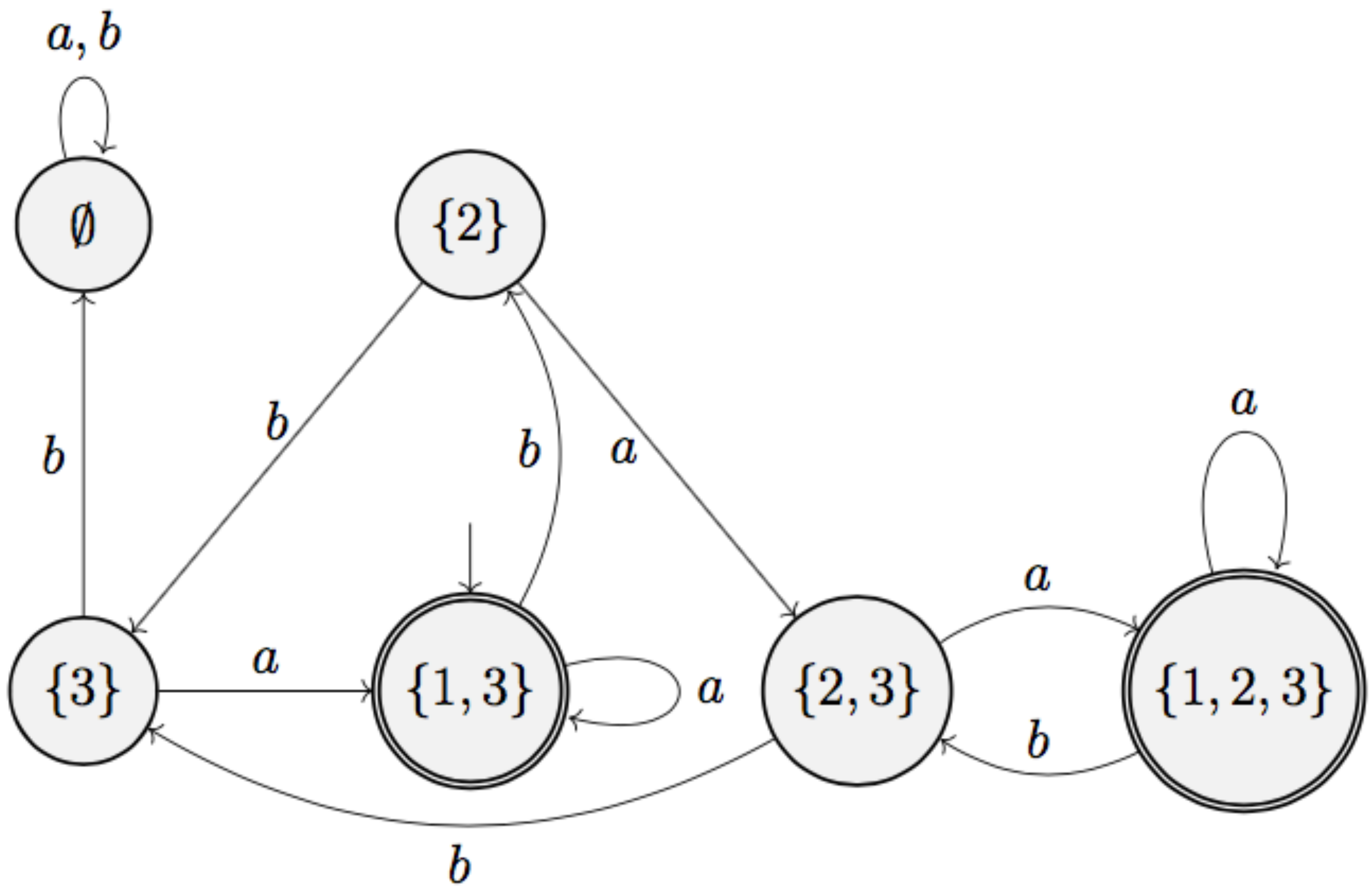
Equivalent DFA







Equivalent DFA



Alternate Definition of Regular Languages

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

Concatenation

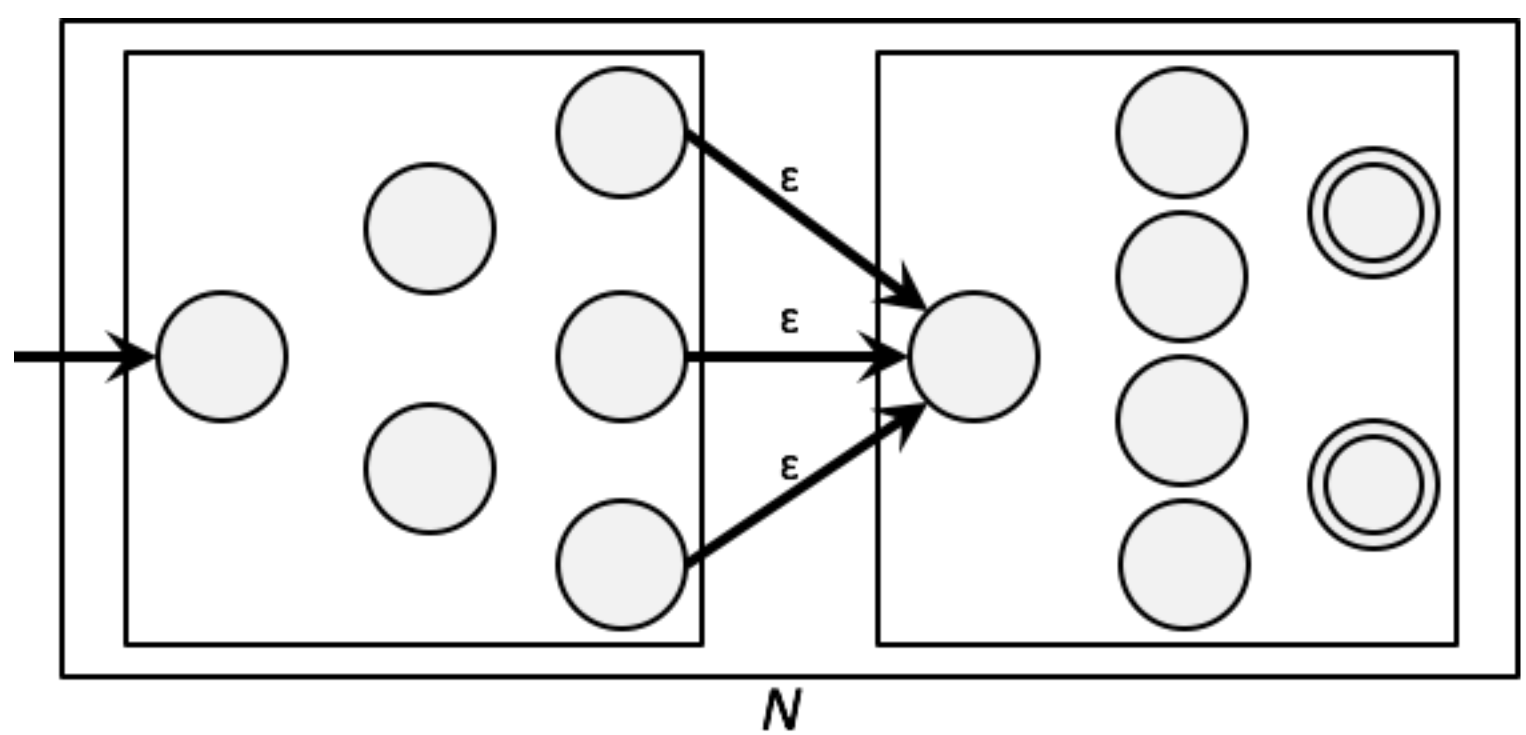
- Let A and B be languages over Σ.
- **Definition.** Concatenation of A and B, denoted $A \circ B$ is defined as

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

- **Theorem**. Regular languages are closed under concatenation.

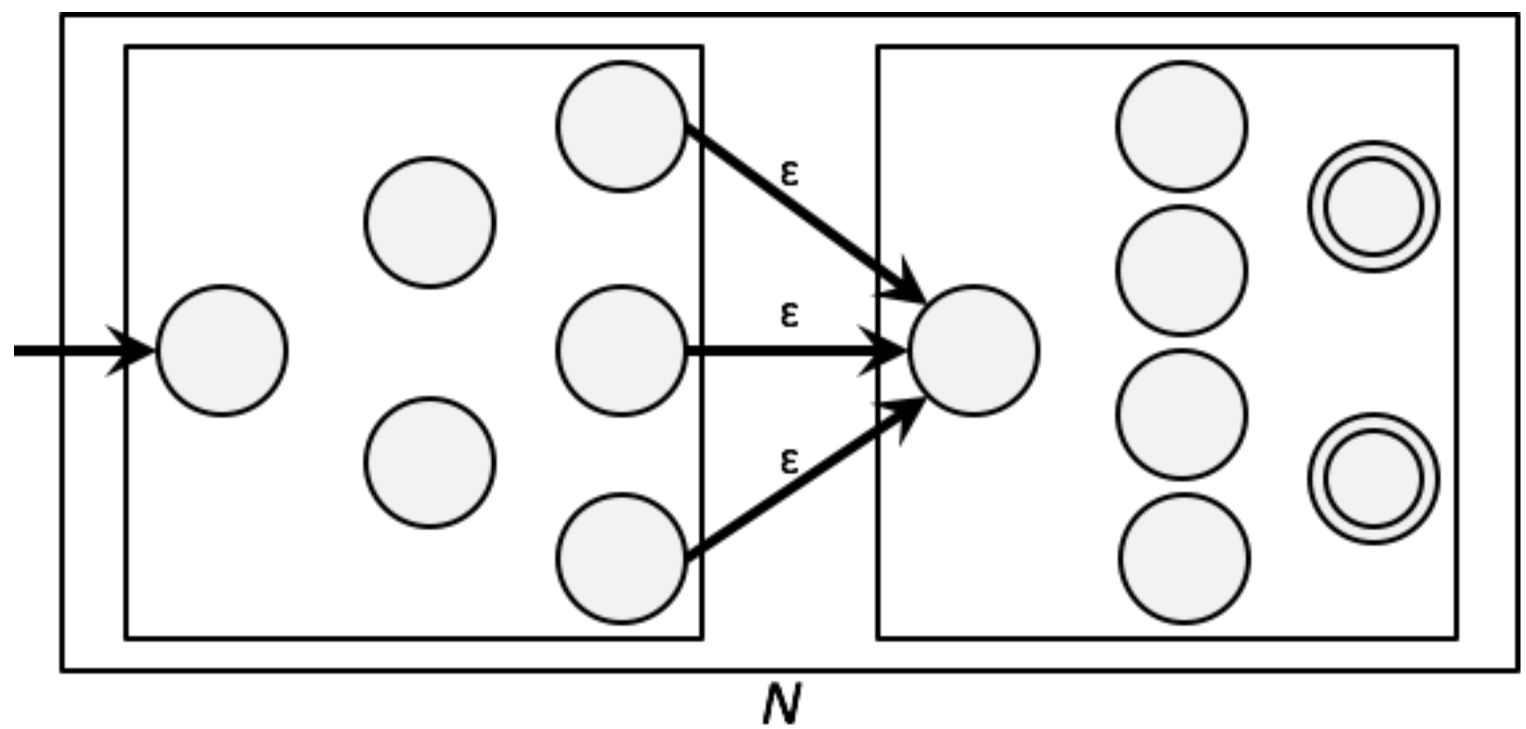
Closed Under Concatenation

- **Theorem**. The class of languages are closed under concatenation.



Closed Under Concatenation

- **Theorem**. The class of languages are closed under concatenation.



Closed Under Concatenation

- **Theorem**. The class of languages are closed under concatenation.
- Proof. Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be the NFA for L_1 and
  $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be the NFA for L_2
- Construct NFA $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $L_1 \circ L_2$
  - $Q = Q_1 \cup Q_2$
  - $q_0 = q_1$
  - $F = F_2$
  - $$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

Kleene Star

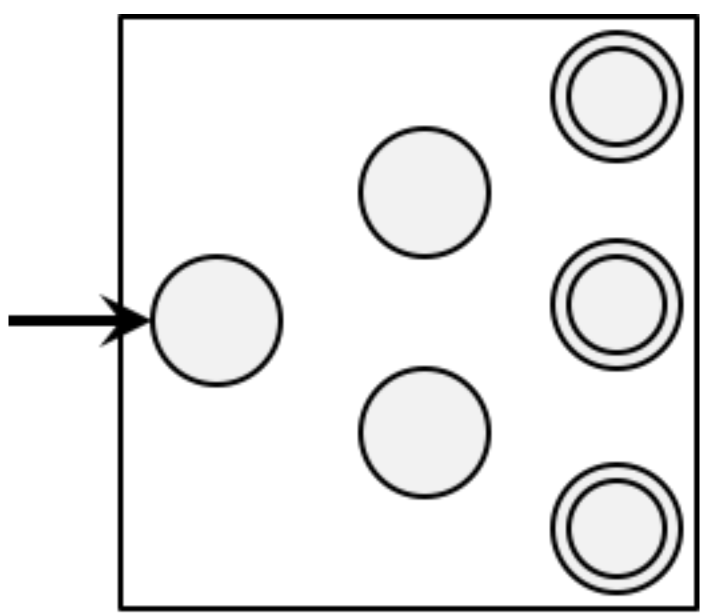
- Let A be a language on Σ
- Definition. Kleene star of A, denoted A^* is defined as:

$$A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and each } w_i \in A\}$$

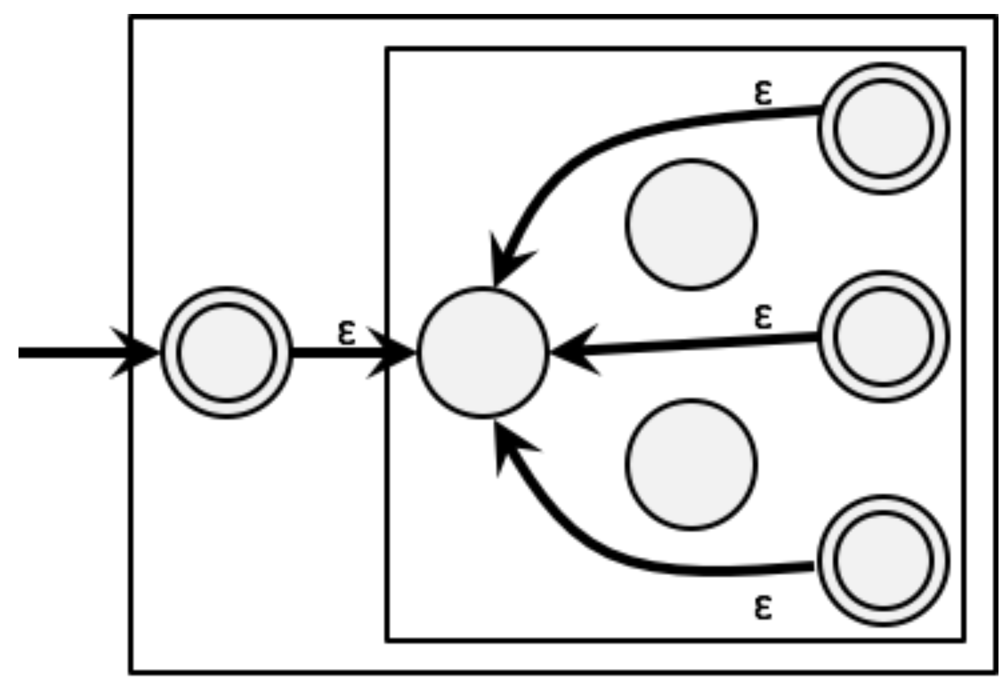
- **Example**. Suppose $L_1 = \{01, 11\}$, what is L^*?
- **Question**. Are regular languages closed under Kleene star?

Kleene Star

- **Theorem.** The class of regular languages is closed under Kleene star.



N_1



N

Closed Under Kleene Star

- **Theorem**. The class of languages are closed under Kleene star.
- Proof. Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be the NFA for L_1
- Construct NFA $N = (Q, \Sigma, \delta, q_0, F)$ to recognize L_1^*
  - $Q = Q_1 \cup \{q_0\}$ (add a new start state)
  - $F = F_1 \cup \{q_0\}$
  - $$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Not All Languages are Regular

- Intuition about regular languages:
  - DFA only has finitely many states, say k
  - Any string with at least k symbols: some DFA state is visited more than once
    - DFA "loops" on long enough strings
  - Can only recognize languages with such nice "regular" structure
- Will see general techniques for showing that a language is not regular
- Classic example of a language that is not regular:
  - $\{w = 0^n 1^n \mid n \geq 0\}$ (equal number of 0s and 1s)