

Finite Automata and Regular Expression

Lin Chen

Email: Lin.Chen@ttu.edu

Grader: zulfi.khan@ttu.edu



TEXAS TECH
UNIVERSITY.

DFA, NFA & REG

- We have learned
 - Deterministic Finite Automata
 - Nondeterministic Finite Automata
 - Regular expression

DFA, NFA & REG

- We have learned
 - Deterministic Finite Automata
 - Nondeterministic Finite Automata
 - Regular expression
- We define a language regular if a DFA or NFA recognizes it
- We will show a language is regular if and only if a regular expression describes it.

Equivalent definition

DFA, NFA & REG

- Q: Show a language is regular if and only if a regular expression describes it.

The language defined by regular expression, L_{REG} , equals the language defined by NFA, L_{NFA}

DFA, NFA & REG

- Target: $L_{REG} = L_{NFA}$
 - $L_{REG} \subseteq L_{NFA}$, i.e., any regular expression is recognized by some NFA
 - $L_{NFA} \subseteq L_{REG}$, i.e., any language recognized by some NFA can be described by some regular expression

$$L_{REG} \subseteq L_{NFA}$$

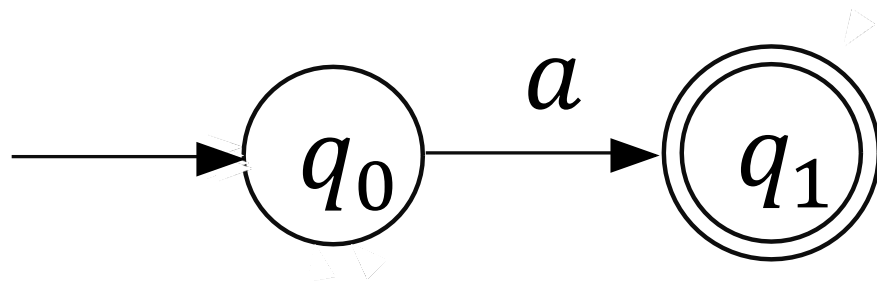
- How do we show $L_{REG} \subseteq L_{NFA}$?
 - For any regular expression r , construct an NFA that accepts exactly the language $L(r)$
 - Regular languages are defined inductively, we prove $L_{REG} \subseteq L_{NFA}$ by induction

Regular expression

- The regular expressions of Σ^* are all strings over $\Sigma \cup \{ (,), \emptyset, \cup, \star \}$ that can be obtained through the following operations:
 - \emptyset and every member of Σ is a regular expression
 - If α and β are regular expressions, then so is $(\alpha\beta)$
 - if α and β are regular expressions, then so is $(\alpha \cup \beta)$
 - if α is a regular expression, then so is α^*
 - Nothing else is a regular expression

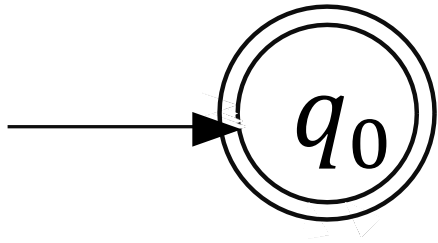
$$L_{REG} \subseteq L_{NFA}$$

- Base case
 - $L(r) = \{a\}, a \in \Sigma$



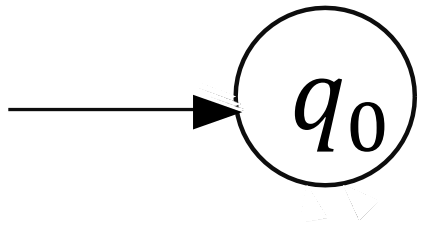
$$L_{REG} \subseteq L_{NFA}$$

- Base case
 - $L(r) = \{e\}$



$$L_{REG} \subseteq L_{NFA}$$

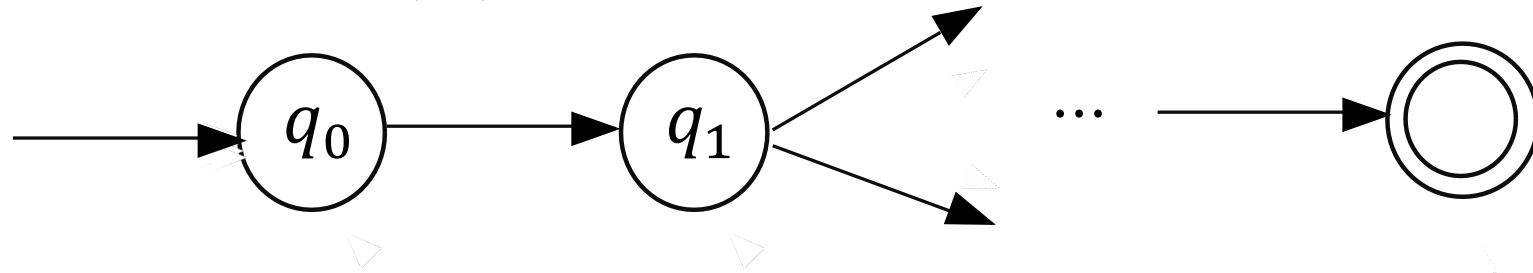
- Base case
 - $L(r) = \emptyset$



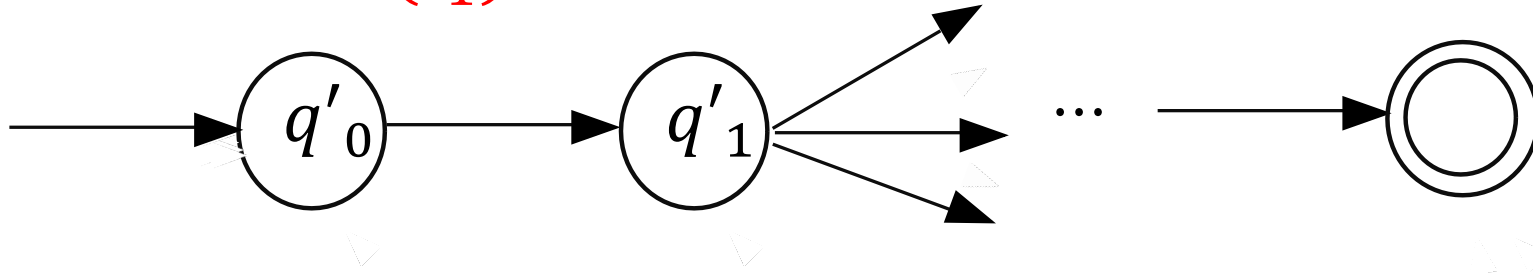
$$L_{REG} \subseteq L_{NFA}$$

- Recursive definition of Regular Language
 - $L(r) = L(r_1 r_2)$

NFA for $L(r_1)$



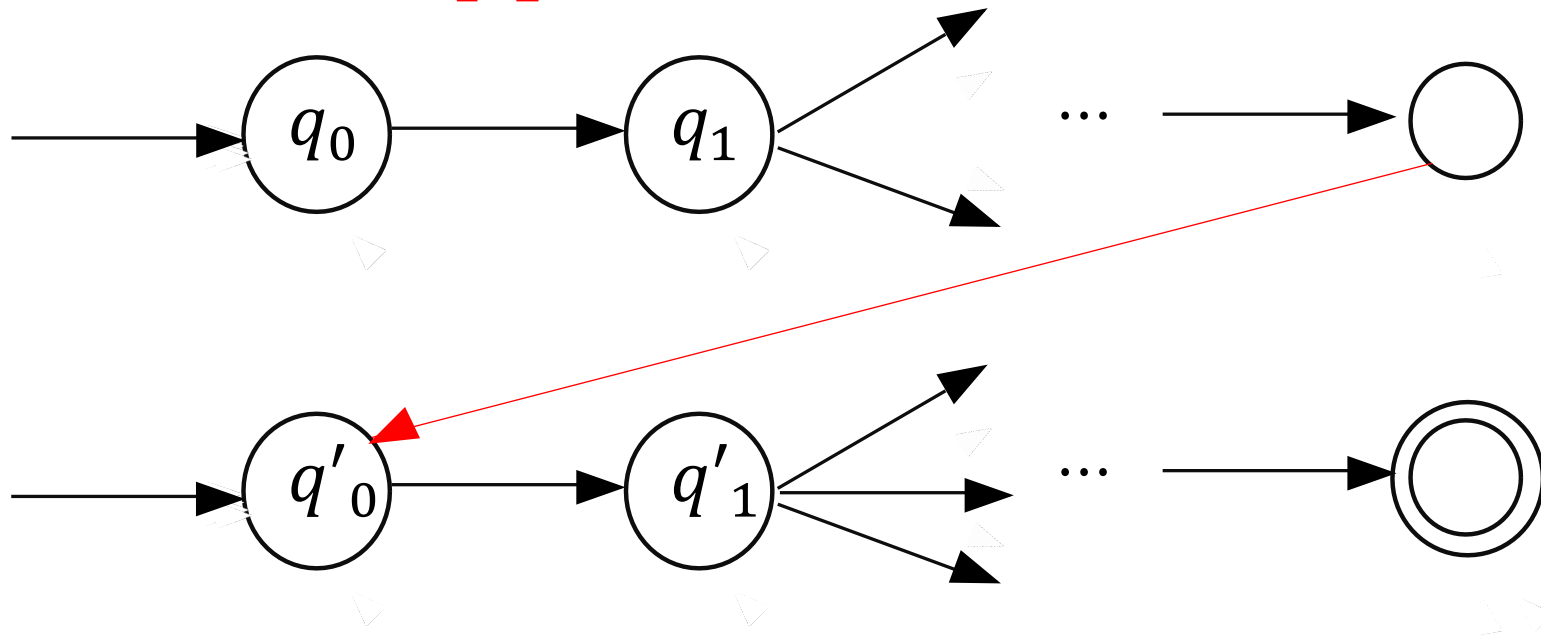
NFA for $L(r_1)$



$$L_{REG} \subseteq L_{NFA}$$

- Recursive definition of Regular Language
 - $L(r) = L(r_1 r_2)$

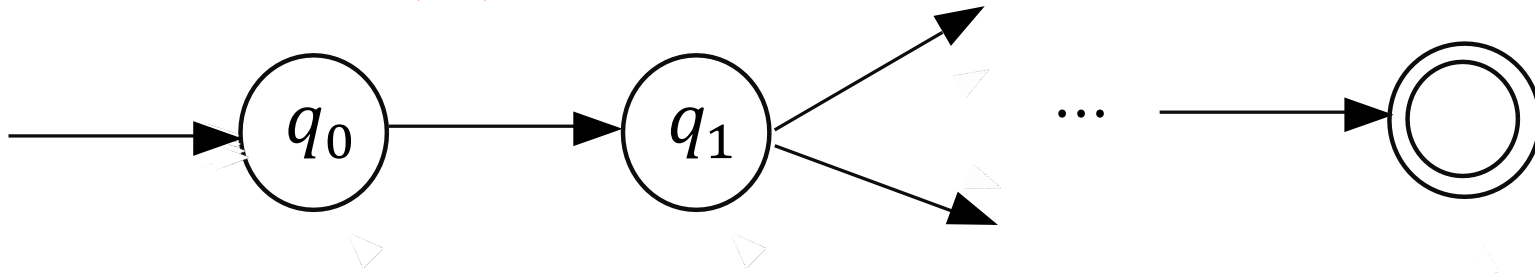
NFA for $L(r_1 r_2)$



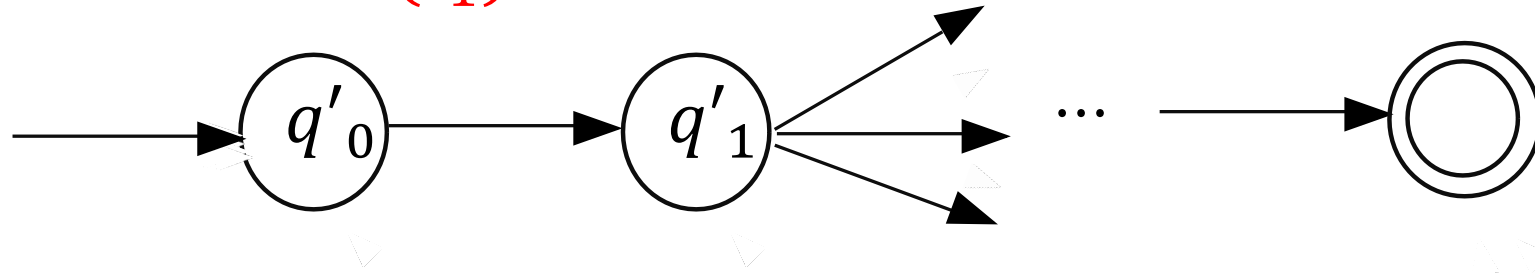
$$L_{REG} \subseteq L_{NFA}$$

- Recursive definition of Regular Language
 - $L(r) = L(r_1 \cup r_2)$

NFA for $L(r_1)$



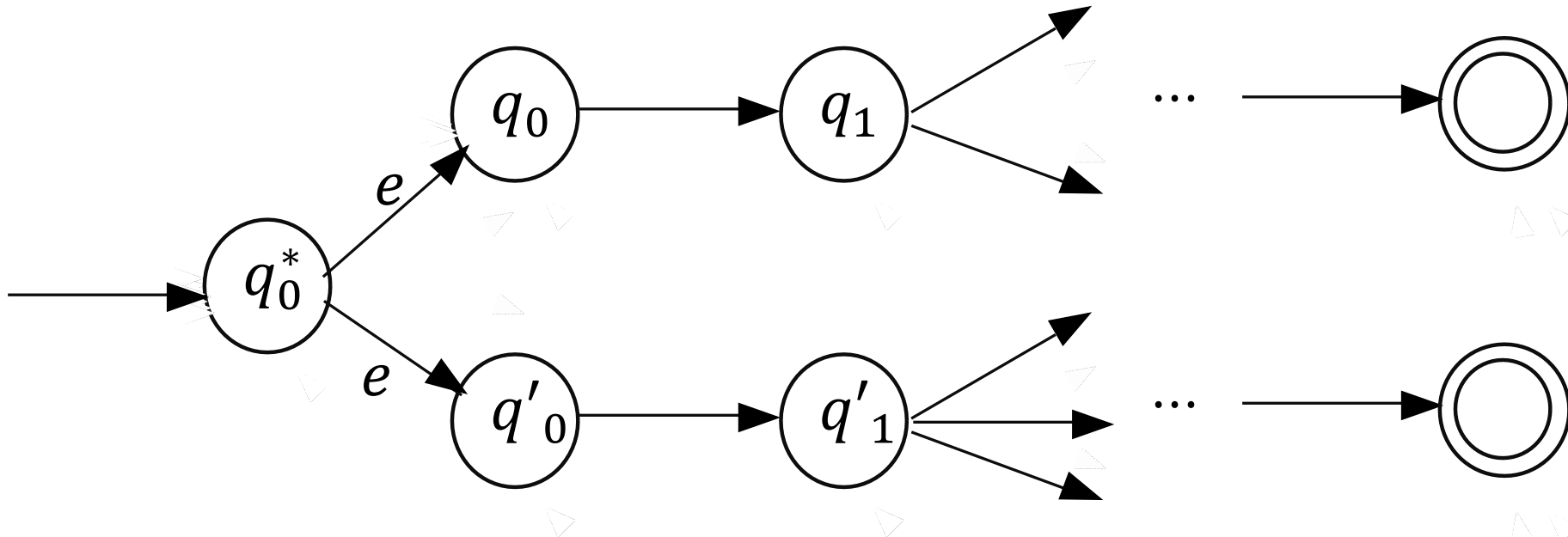
NFA for $L(r_1)$



$$L_{REG} \subseteq L_{NFA}$$

- Recursive definition of Regular Language
 - $L(r) = L(r_1 \cup r_2)$

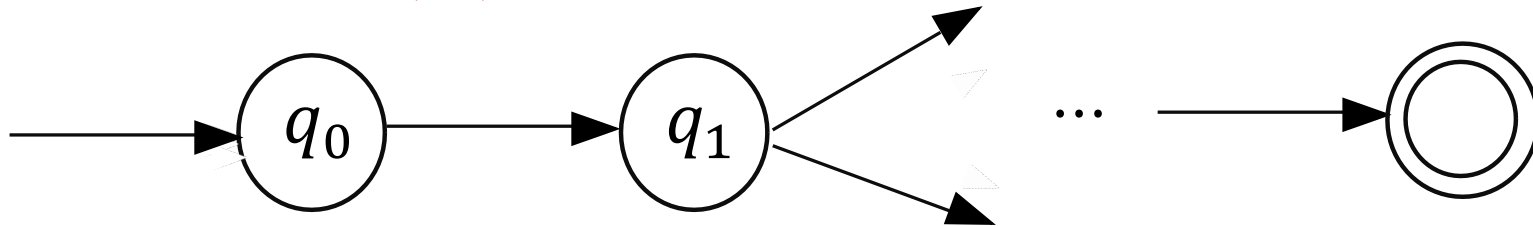
NFA for $L(r_1 \cup r_2)$



$$L_{REG} \subseteq L_{NFA}$$

- Recursive definition of Regular Language
 - $L(r) = L(r_1^*)$

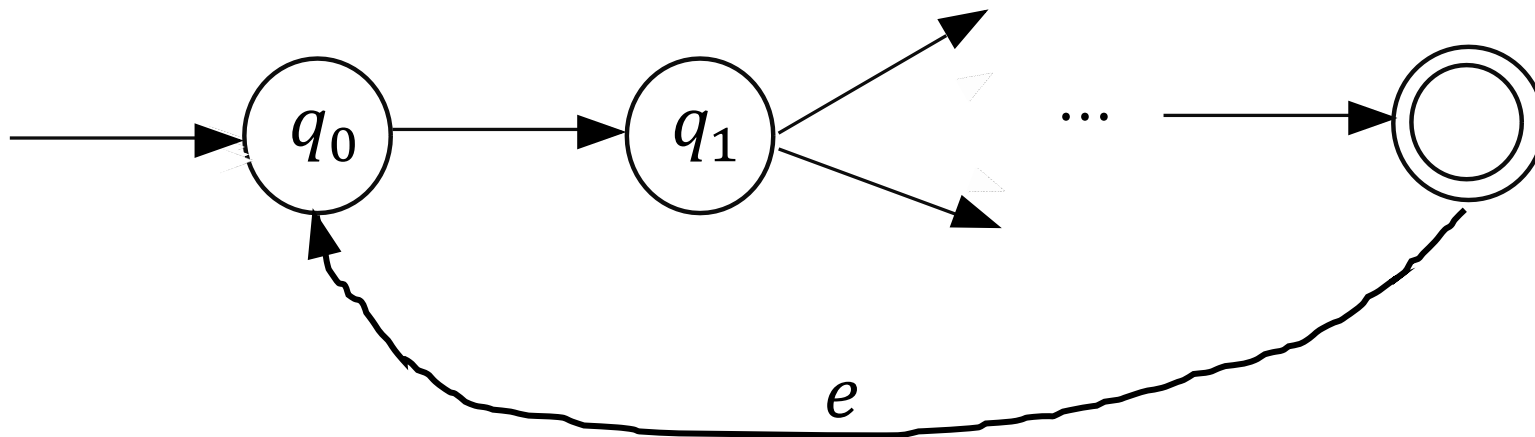
NFA for $L(r_1)$



$$L_{REG} \subseteq L_{NFA}$$

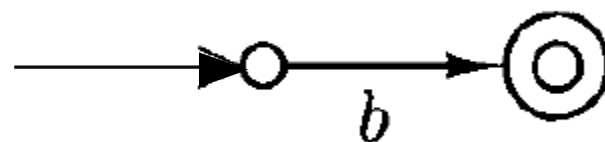
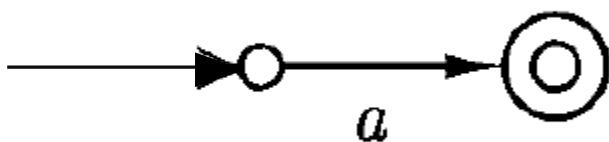
- Recursive definition of Regular Language
 - $L(r) = L(r_1^*)$

NFA for $L(r_1^)$*



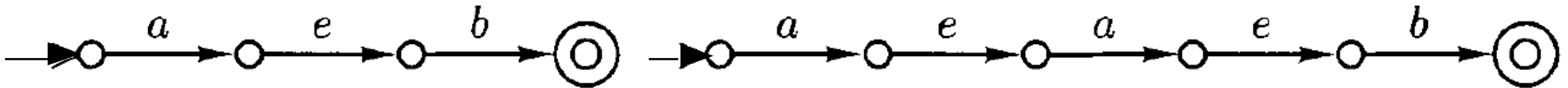
$$L_{REG} \subseteq L_{NFA}$$

- Example
 - a, b



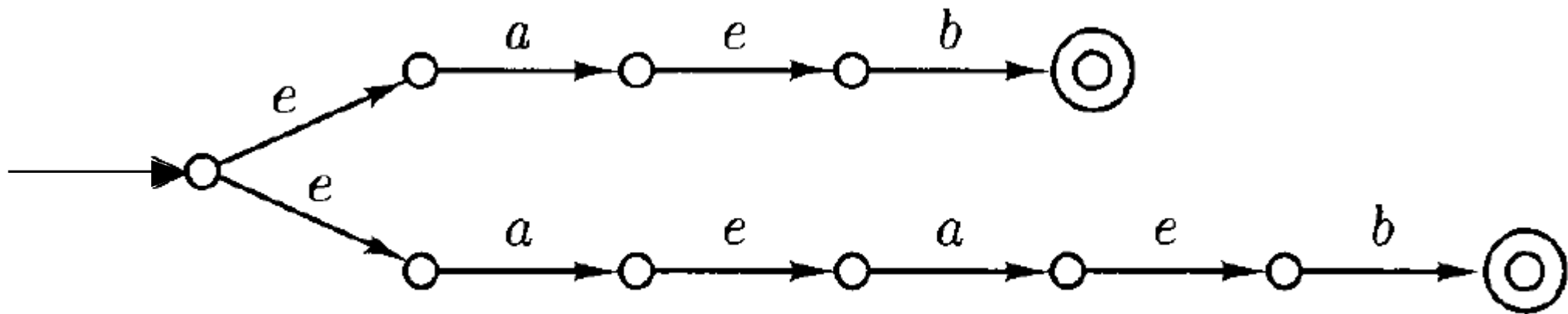
$$L_{REG} \subseteq L_{NFA}$$

- Example
 - ab, aab



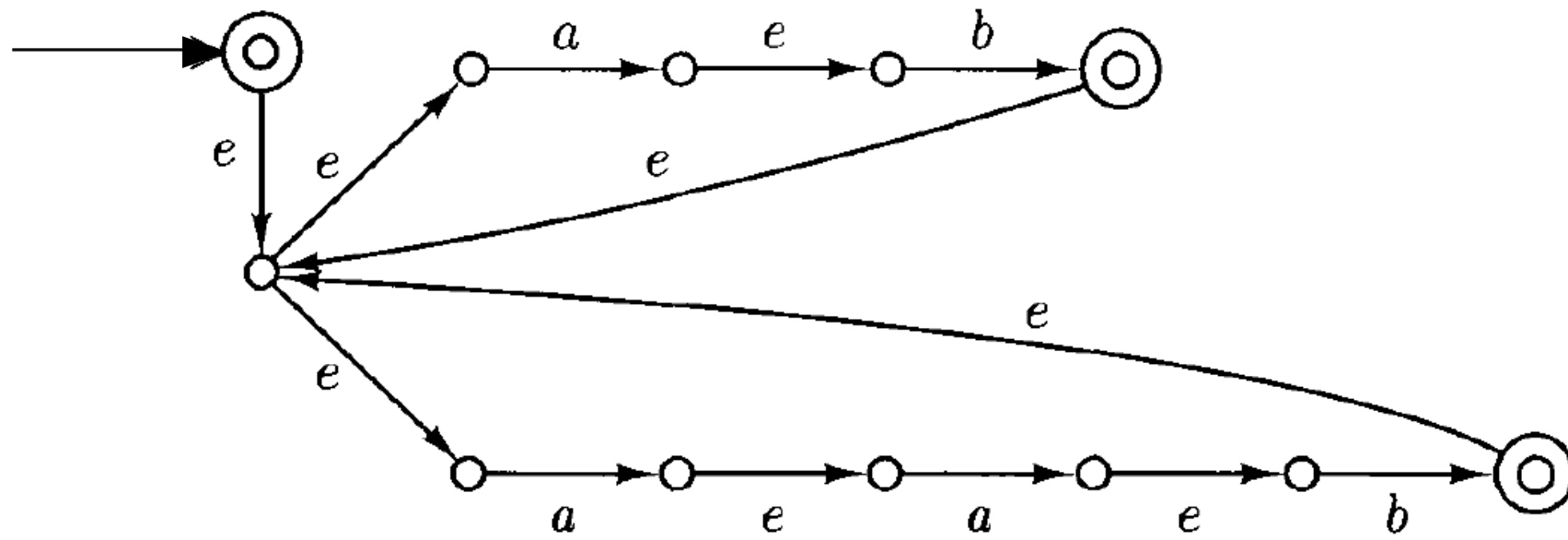
$$L_{REG} \subseteq L_{NFA}$$

- Example
 - $ab \cup aab$



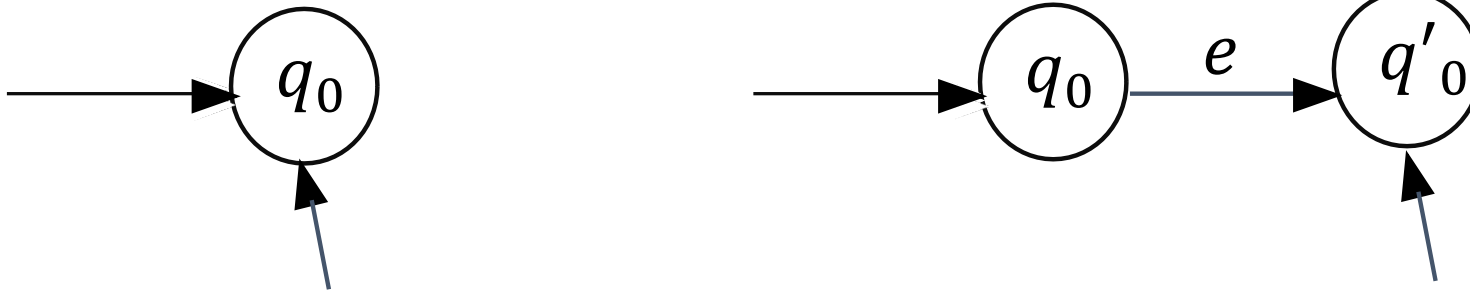
$$L_{REG} \subseteq L_{NFA}$$

- Example
 - $(ab \cup aab)^*$



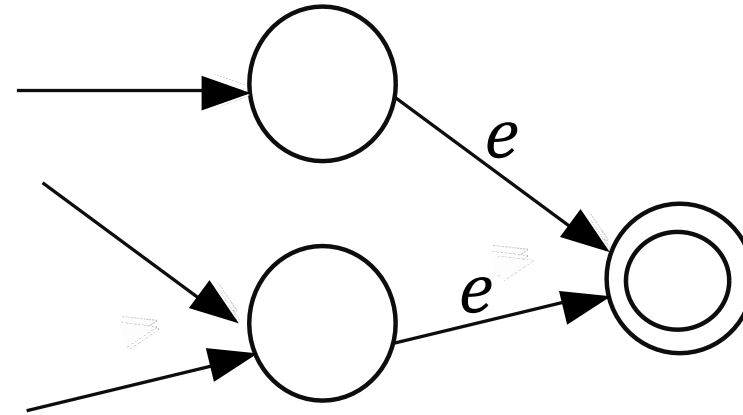
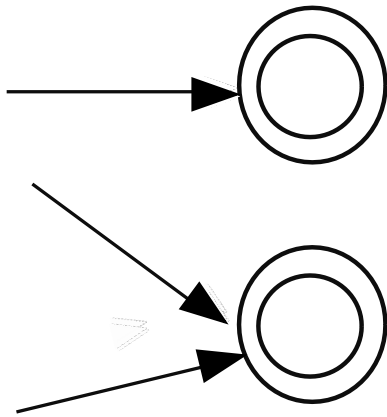
$$L_{NFA} \subseteq L_{REG}$$

- Start with a specialized NFA
 - No transition into the start state



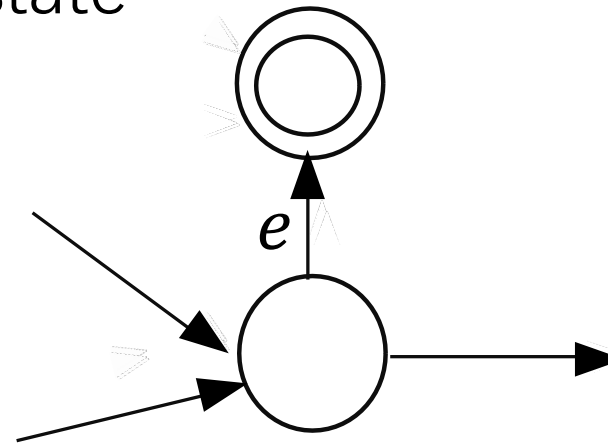
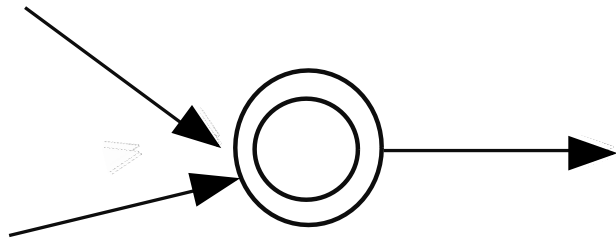
$$L_{NFA} \subseteq L_{REG}$$

- Start with a specialized NFA
 - No transition into the start state
 - Single final state



$$L_{NFA} \subseteq L_{REG}$$

- Start with a specialized NFA
 - No transition into the start state
 - Single final state
 - No transition out of the final state



$$L_{NFA} \subseteq L_{REG}$$

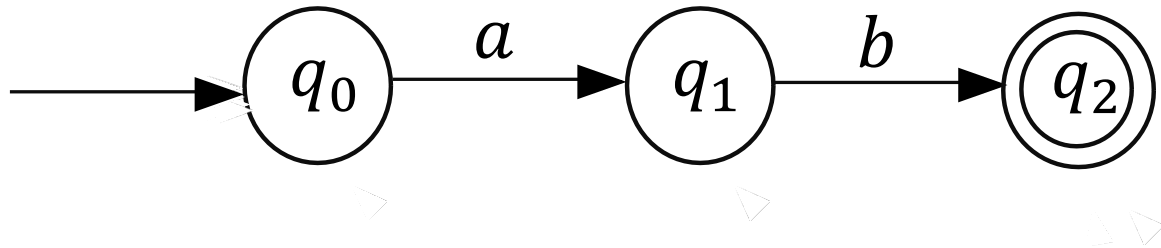
- Transitions will be labeled with regular expressions
- If there is a transition from state q_1 to state q_2 labeled with regular expression r , then any string generated by r can move the machine from q_1 to q_2
- Remove states, relabeling transitions so that the language defined by the machine does not change

$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
- Use example to illustrate the basic idea (example≠proof!)

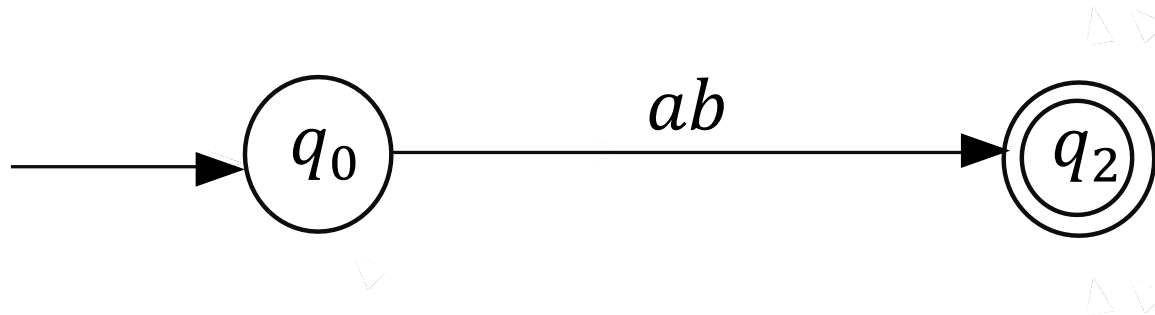
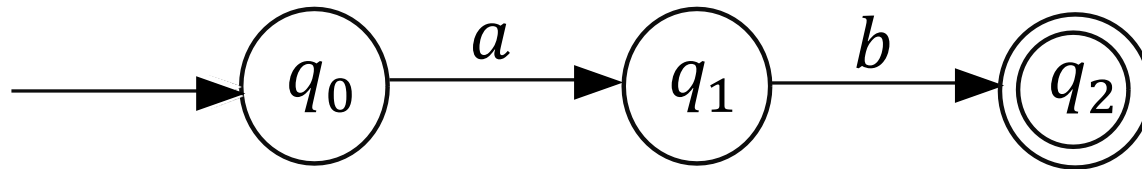
$$L_{NFA} \subseteq L_{REG}$$

- Example:



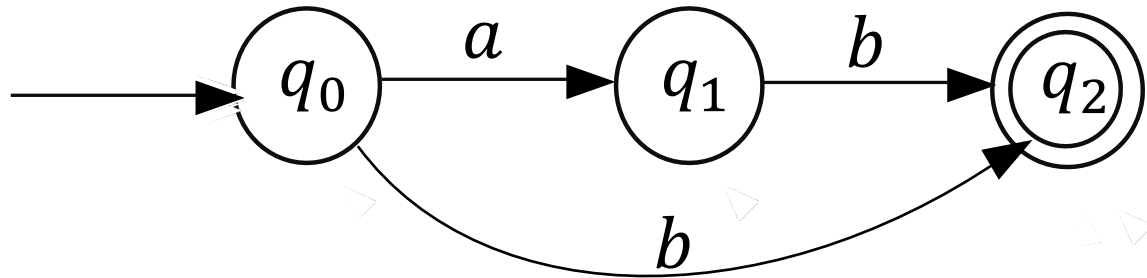
$$L_{NFA} \subseteq L_{REG}$$

- Example:
 - Remove state q_1



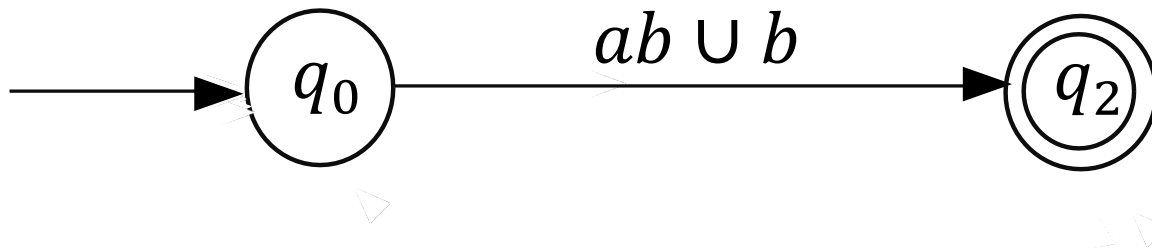
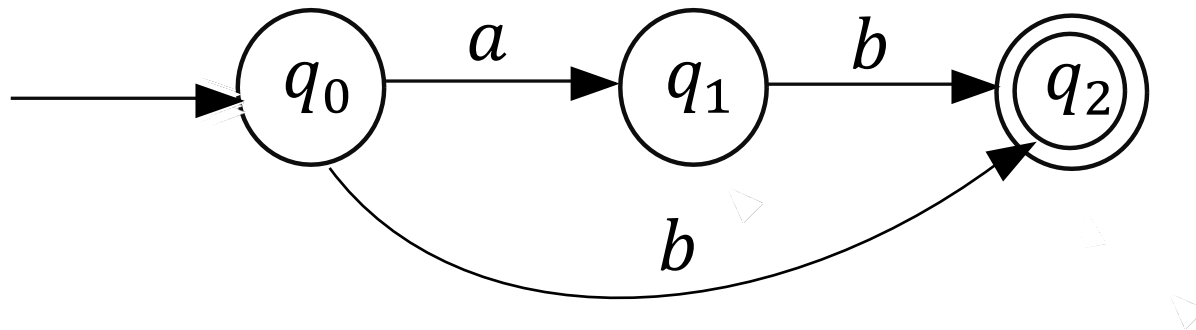
$$L_{NFA} \subseteq L_{REG}$$

- Example:



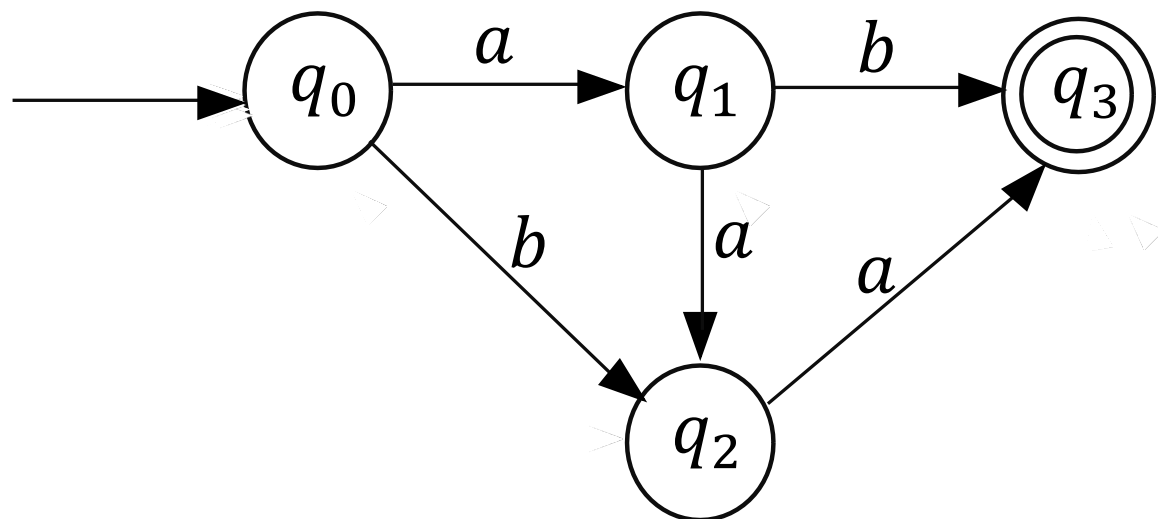
$$L_{NFA} \subseteq L_{REG}$$

- Example:
 - Remove state q_1



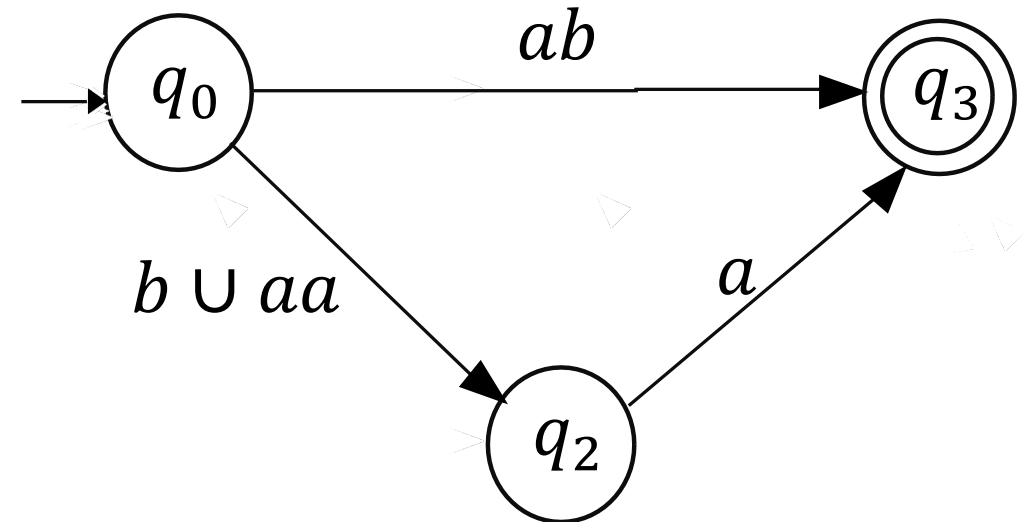
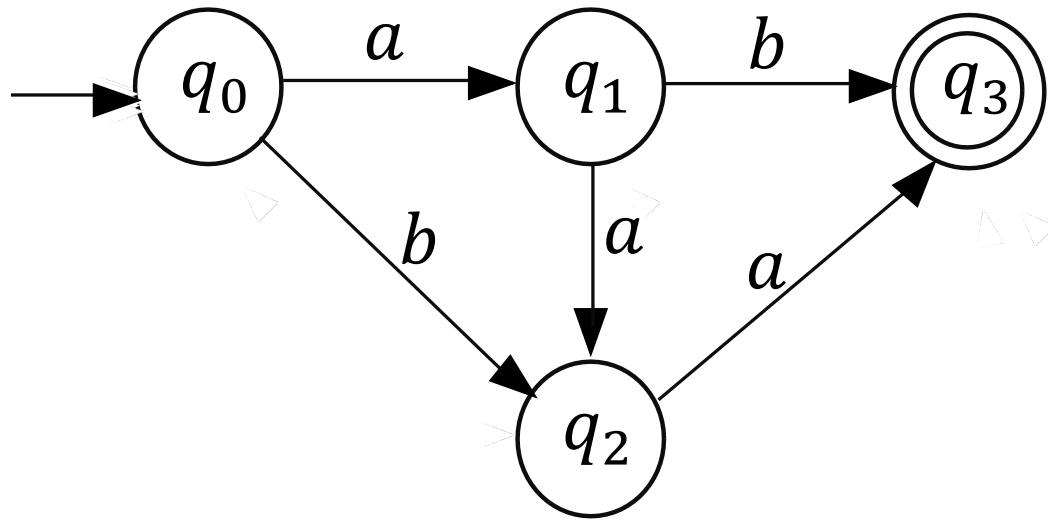
$$L_{NFA} \subseteq L_{REG}$$

- Example:



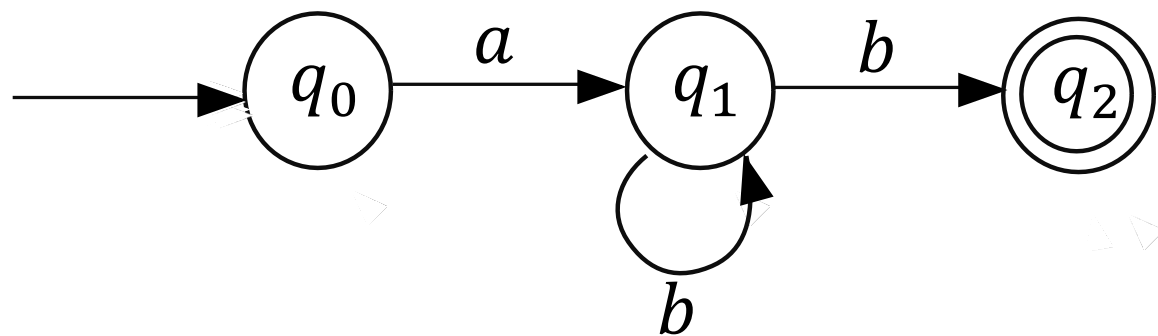
$$L_{NFA} \subseteq L_{REG}$$

- Example:
 - Remove state q_1



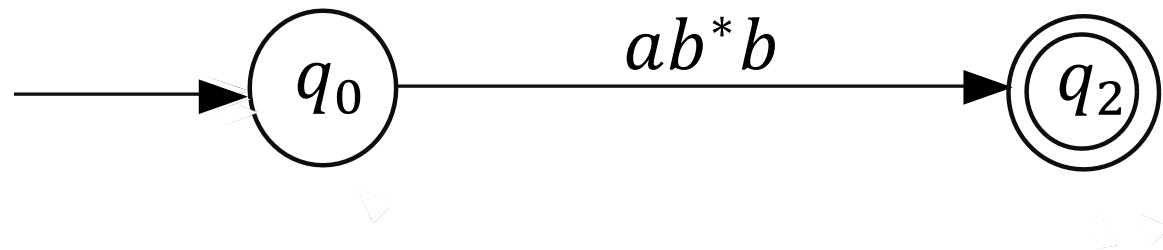
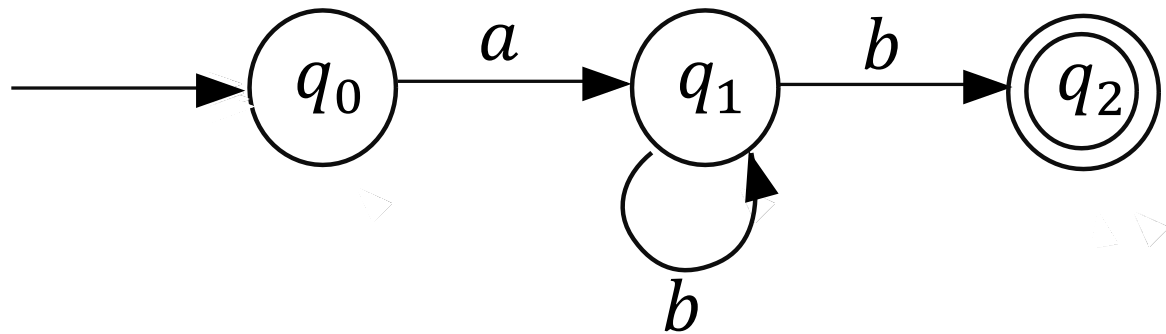
$$L_{NFA} \subseteq L_{REG}$$

- Example:



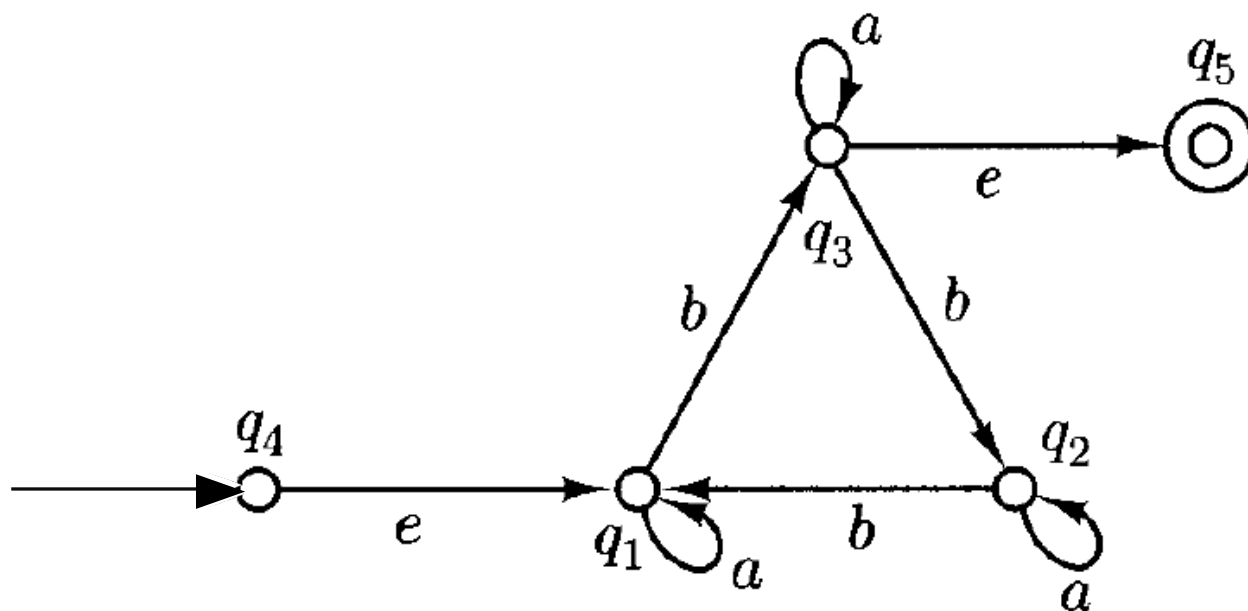
$$L_{NFA} \subseteq L_{REG}$$

- Example:
 - Remove state q_1



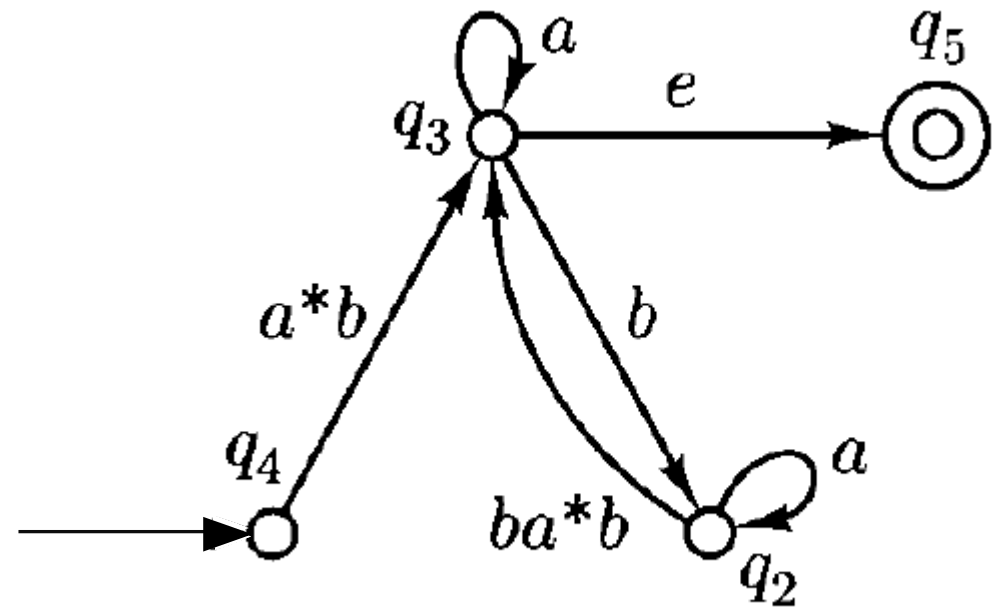
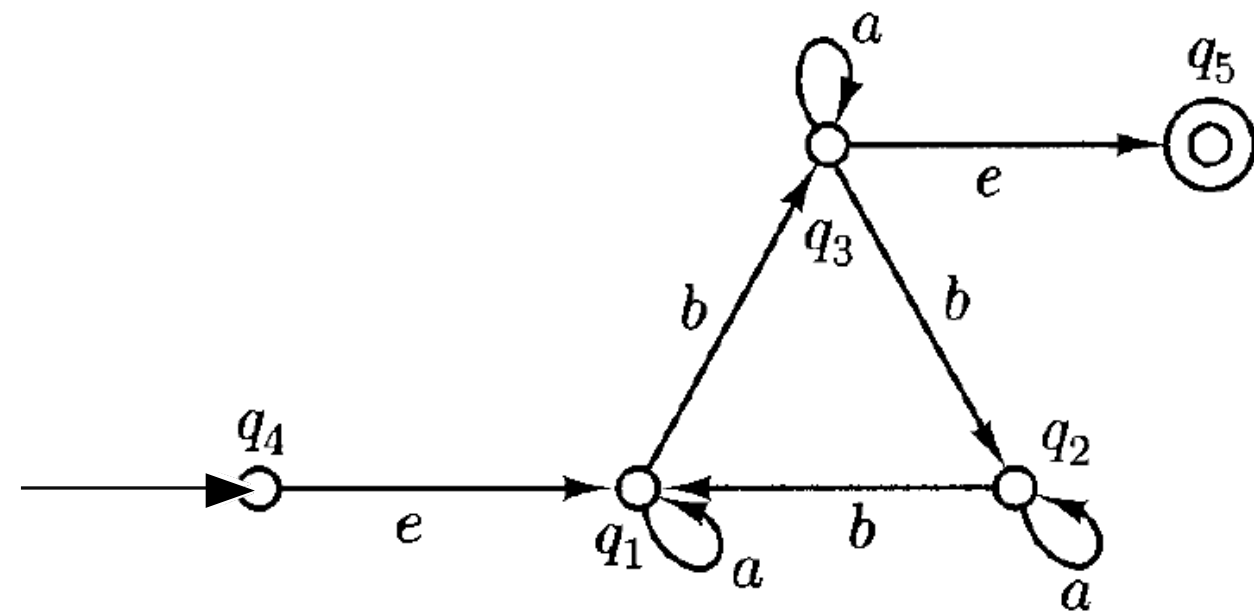
$$L_{NFA} \subseteq L_{REG}$$

- More example:



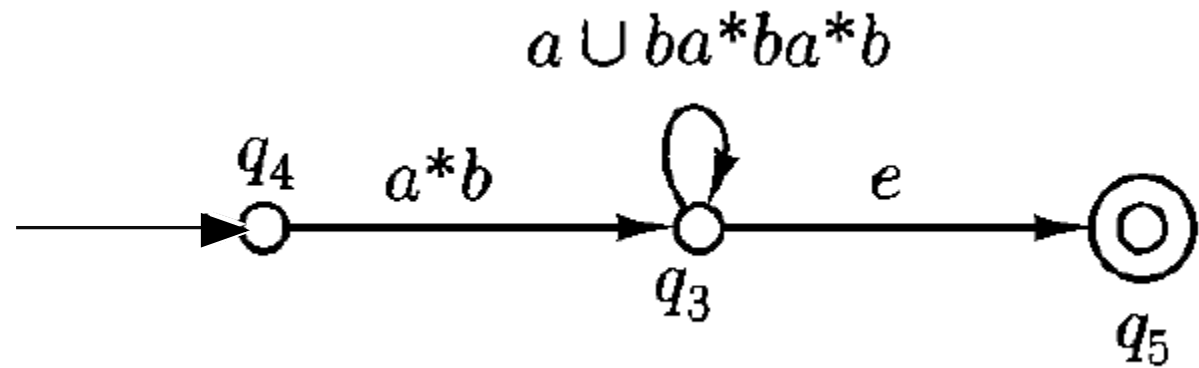
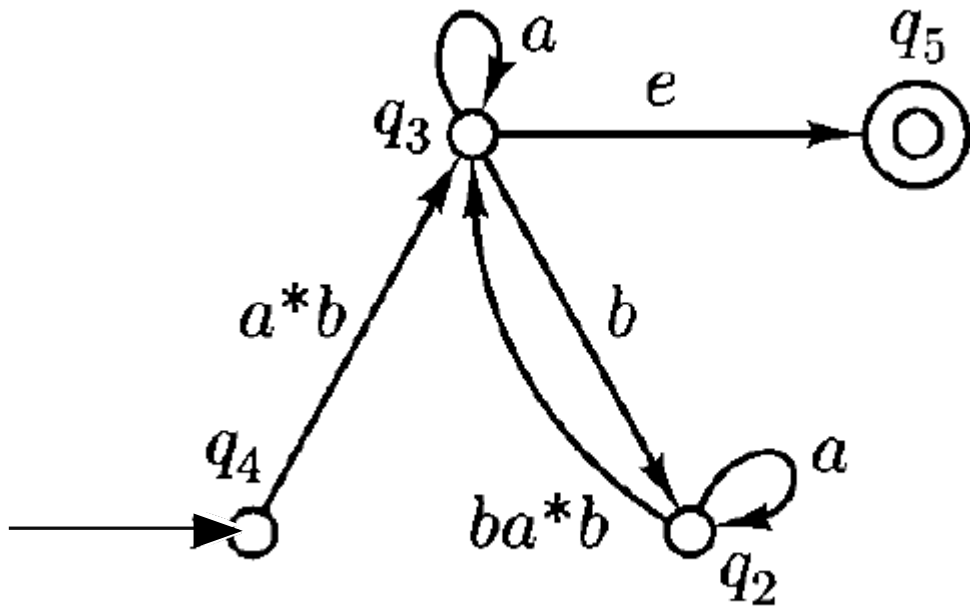
$$L_{NFA} \subseteq L_{REG}$$

- More example:



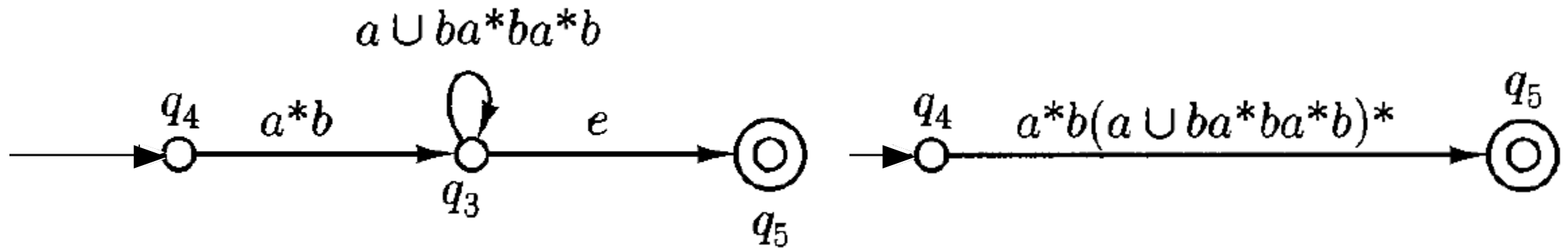
$$L_{NFA} \subseteq L_{REG}$$

- More example:



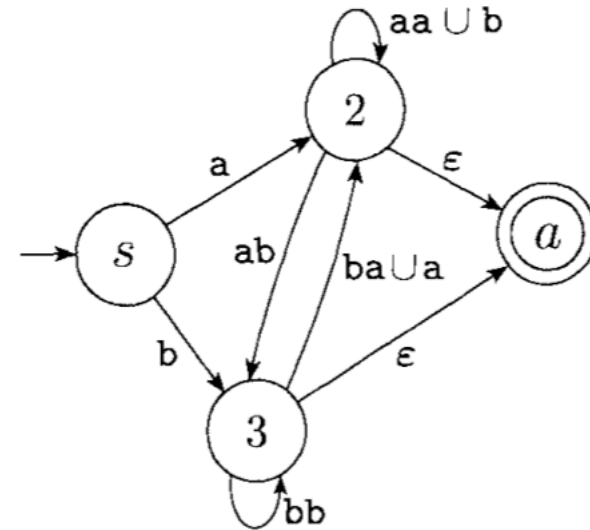
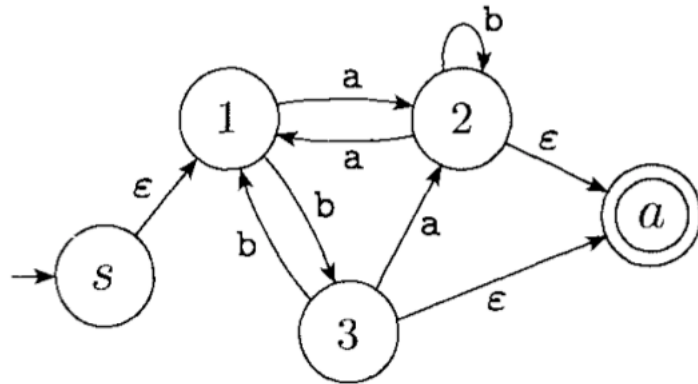
$$L_{NFA} \subseteq L_{REG}$$

- More example:



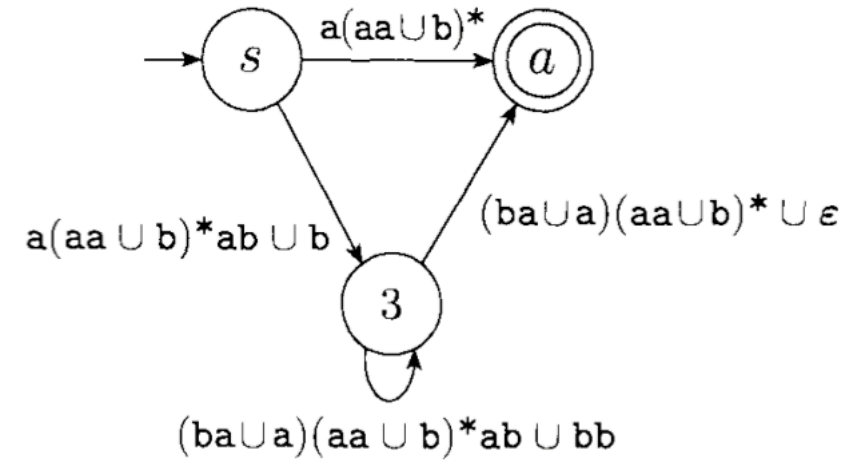
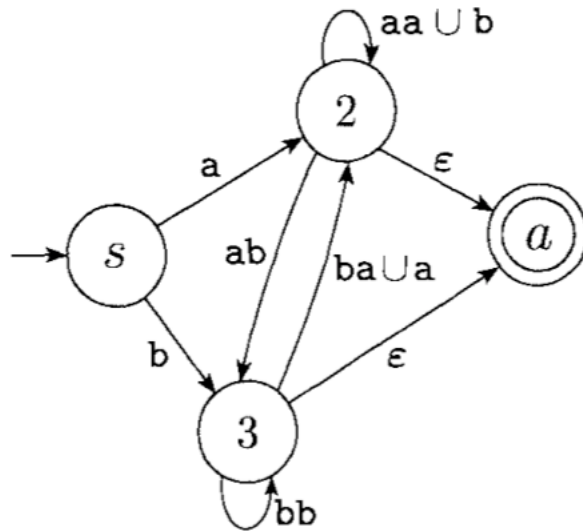
$$L_{NFA} \subseteq L_{REG}$$

- More example:



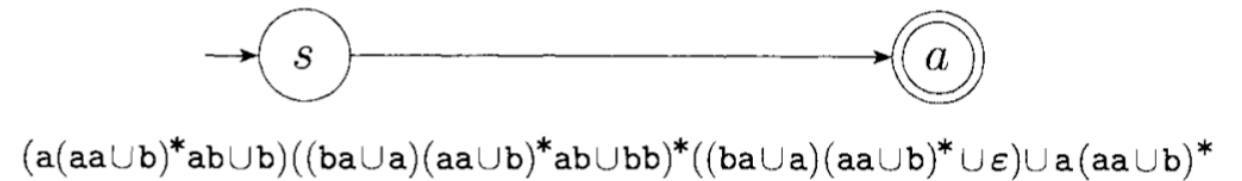
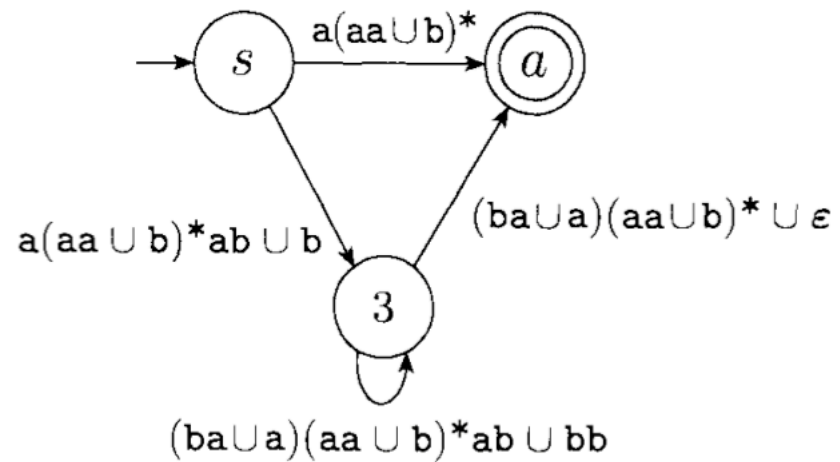
$$L_{NFA} \subseteq L_{REG}$$

- More example:



$$L_{NFA} \subseteq L_{REG}$$

- More example:



$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
 - Let $K = \{q_1, q_2, \dots, q_n\}$ and $s = q_1$
 - Define $R(i, j, k)$ as the set of strings that can drive M from q_i to q_j without passing any state numbered $k + 1$ or higher

$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
 - Let $K = \{q_1, q_2, \dots, q_n\}$ and $s = q_1$
 - Define $R(i, j, k)$ as the set of strings that can drive M from q_i to q_j without passing any state numbered $k + 1$ or higher
- What is the language accepted by M

$$L(M) = \bigcup \{R(1, j, n) : q_j \in F\}$$

$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
 - Let $K = \{q_1, q_2, \dots, q_n\}$ and $s = q_1$
 - Define $R(i, j, k)$ as the set of strings that can drive M from q_i to q_j without passing any state numbered $k + 1$ or higher
- What is the language accepted by M
- We prove $R(i, j, k)$ is regular for any i, j, k

$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
 - Let $K = \{q_1, q_2, \dots, q_n\}$ and $s = q_1$
 - Define $R(i, j, k)$ as the set of strings that can drive M from q_i to q_j without passing any state numbered $k + 1$ or higher
- What is the language accepted by M

$$L(M) = \bigcup \{R(1, j, n) : q_j \in F\}$$

- We prove $R(i, j, k)$ is regular for any i, j, k

$$L_{NFA} \subseteq L_{REG}$$

- Proof by induction on the “length” of the path
 - Let $K = \{q_1, q_2, \dots, q_n\}$ and $s = q_1$
 - Define $R(i, j, k)$ as the set of strings that can drive M from q_i to q_j without passing any state numbered $k + 1$ or higher
- What is the language accepted by M

$$L(M) = \bigcup \{R(1, j, n) : q_j \in F\}$$

- We prove $R(i, j, k)$ is regular for any i, j, k

$$R(i, j, k) = R(i, j, k-1) \cup R(i, k, k-1)R(k, k, k-1)^*R(k, j, k-1)$$