

Theory of Automata

Shakir Ullah Shah

Lecture 4

Languages Associated with Regular Expressions

Regular Languages

- The language generated by any *regular expression* is called a **regular language**.
- It is to be noted that if r_1, r_2 are regular expressions, corresponding to the languages L_1 and L_2 then the languages generated by $r_1 + r_2$, $r_1 r_2$ (or $r_2 r_1$) and r_1^* (or r_2^*) are also regular languages.

Note

- It is to be noted that if L_1 and L_2 are expressed by r_1 and r_2 , respectively then the language expressed by
 1. $r_1 + r_2$, is the language $L_1 + L_2$ or $L_1 \cup L_2$
 2. $r_1 r_2$, is the language $L_1 L_2$, of strings obtained by prefixing every string of L_1 with every string of L_2
 3. r_1^* , is the language L_1^* , of strings obtained by concatenating the strings of L , including the null string.

Rules for language association with R.E's

- Rule 1:
 - The language associated with the regular expression that is just a single letter is that one-letter word alone, and the language associated with Λ is just $\{\Lambda\}$, a one-word language.

Rules for language association with R.E's

- Rule 2:

If r_1 is a regular expression associated with the language L_1 and r_2 is a regular expression associated with the language L_2 , then:

(i) The regular expression $(r_1)(r_2)$ is associated with the product L_1L_2 , that is the language L_1 times the language L_2 :

$$\text{language}(r_1r_2) = L_1L_2$$

Rules for language association with R.E's

- Rule 2:
 - (ii) The regular expression $r_1 + r_2$ is associated with the language formed by the union of L_1 and L_2 :
$$\text{language}(r_1 + r_2) = L_1 + L_2$$
 - (iii) The language associated with the regular expression $(r_1)^*$ is L_1^* , the Kleene closure of the set L_1 as a set of words:
$$\text{language}(r_1^*) = L_1^*$$

Finite Languages Are Regular

Theorem 5:

- If L is a finite language (a language with only finitely many words), then L can be defined by a regular expression.
- In other words, all finite languages are regular.

Finite Languages Are Regular

- ***Proof:***
- Let L be a finite language. To make one regular expression that defines L , we insert plus signs between all the words in L e.g.

$L = \{baa, abbba, bababa\}$ is
 $baa+abbba+bababa$

- Note:
 - This algorithm only works for finite languages because an infinite language would become a regular expression that is infinitely long, which is forbidden.

Finite Automata

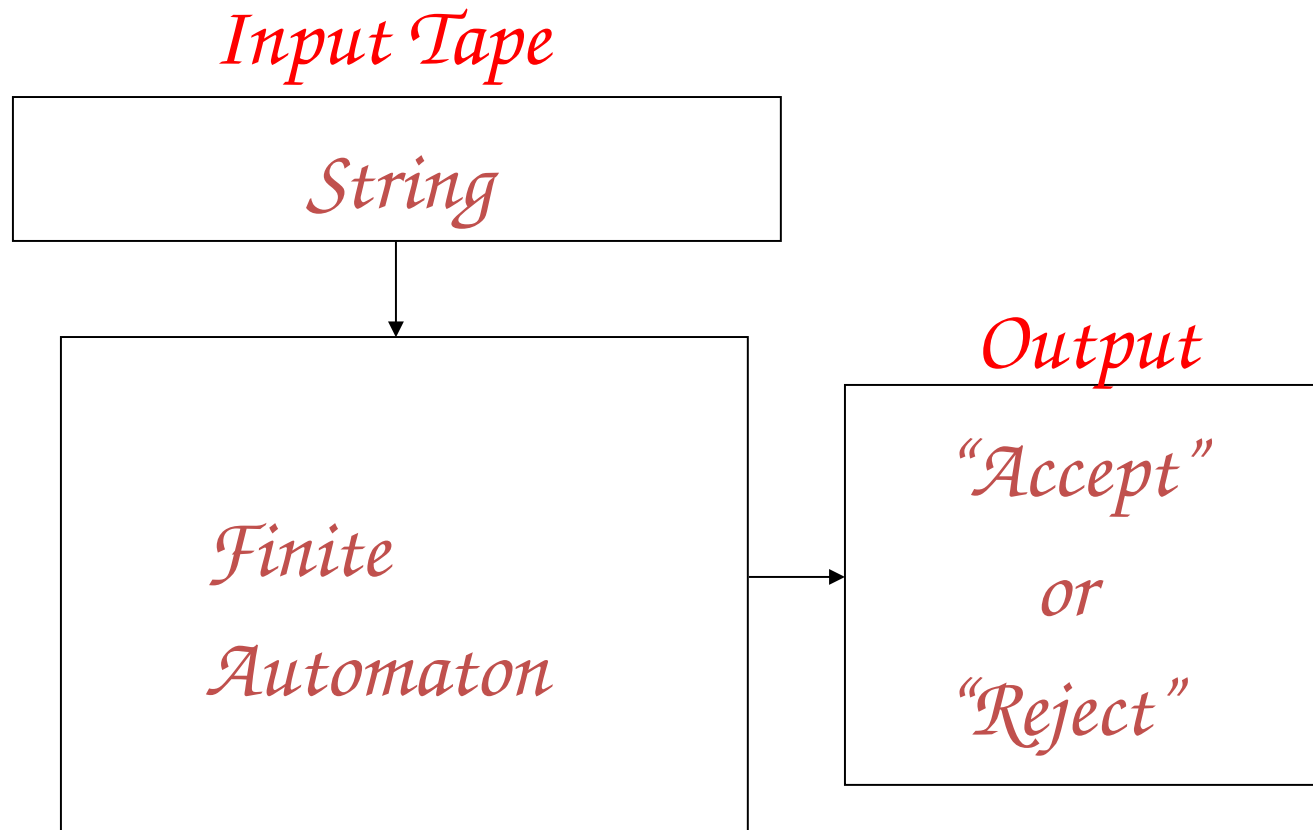
Finite Automata

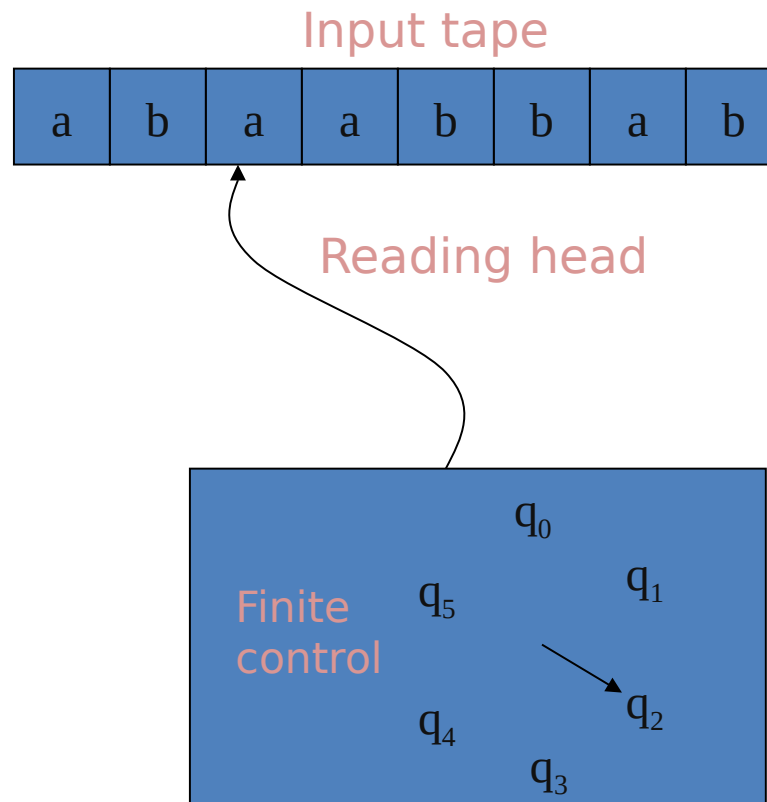
- Also known as **deterministic finite automaton (DFA), Finite State Automata (FSA) or simply FA**
- It is a simple language **recognition device**.
- It is called deterministic because their operation is completely determined by their input.
- Strings are fed into the device by means of an **input tape**, which is divided into squares, with one symbol inscribed in each tape square.

Finite Automata

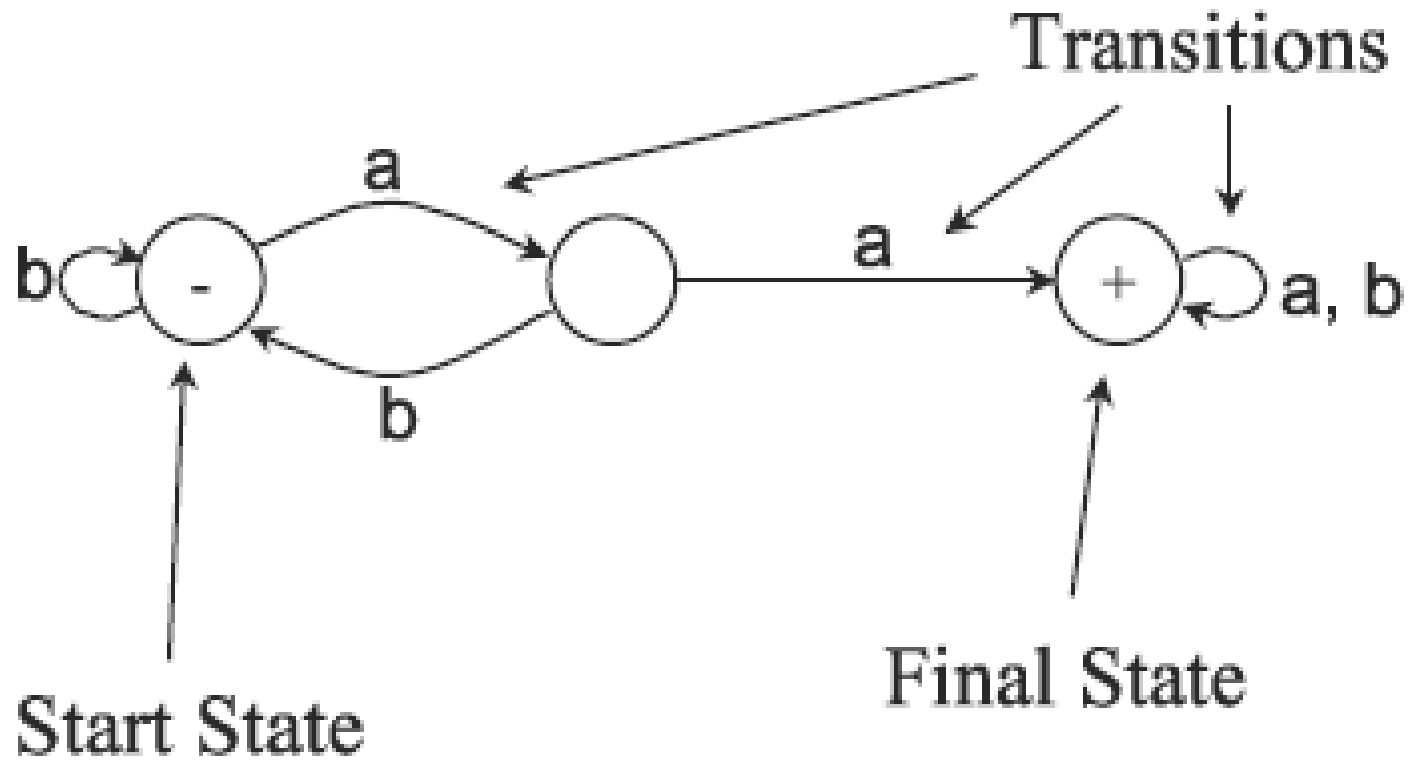
- It is used to determine whether a word does or does not belong to a Regular Language
- User for defining a Regular Language
- Used in Lexical Analyzers

Deterministic Finite Automaton

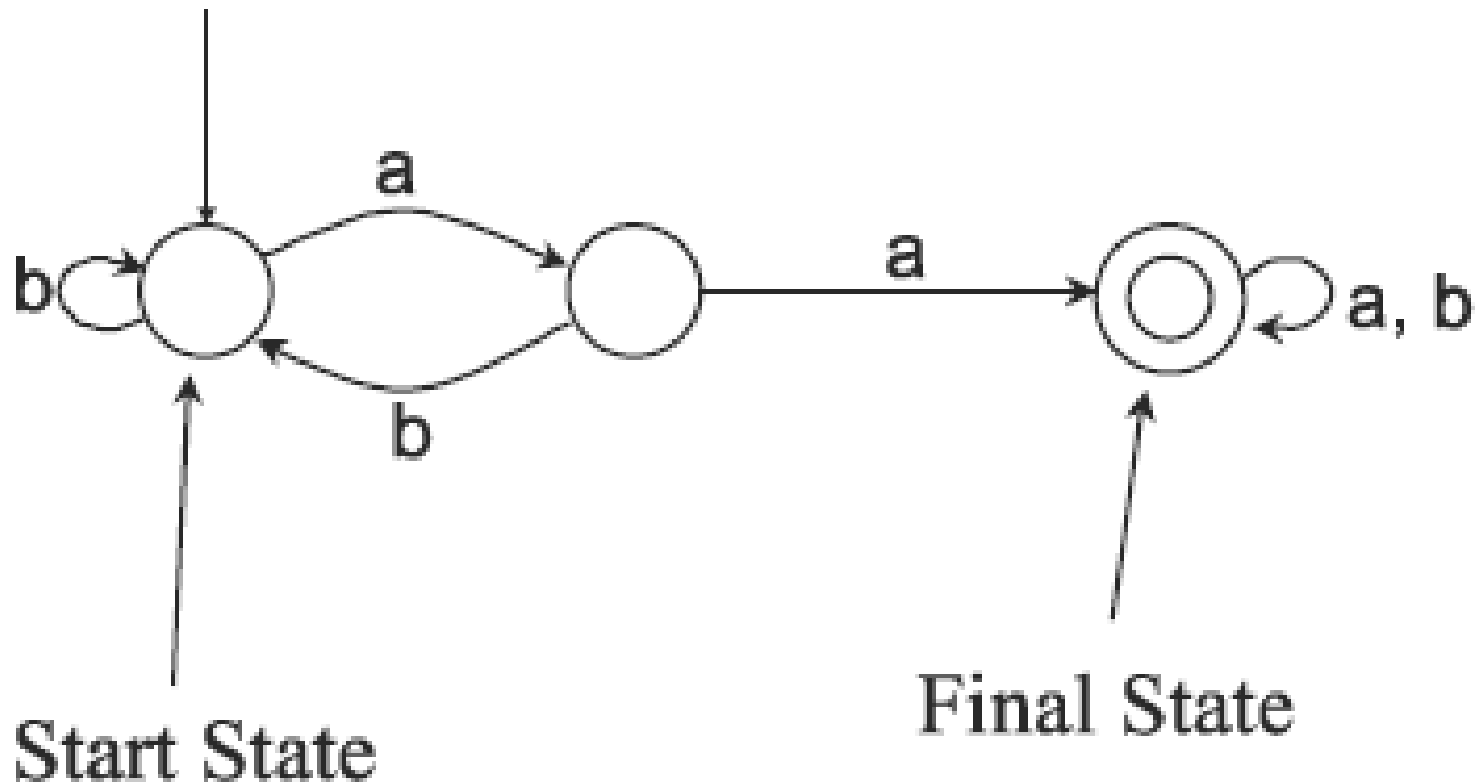




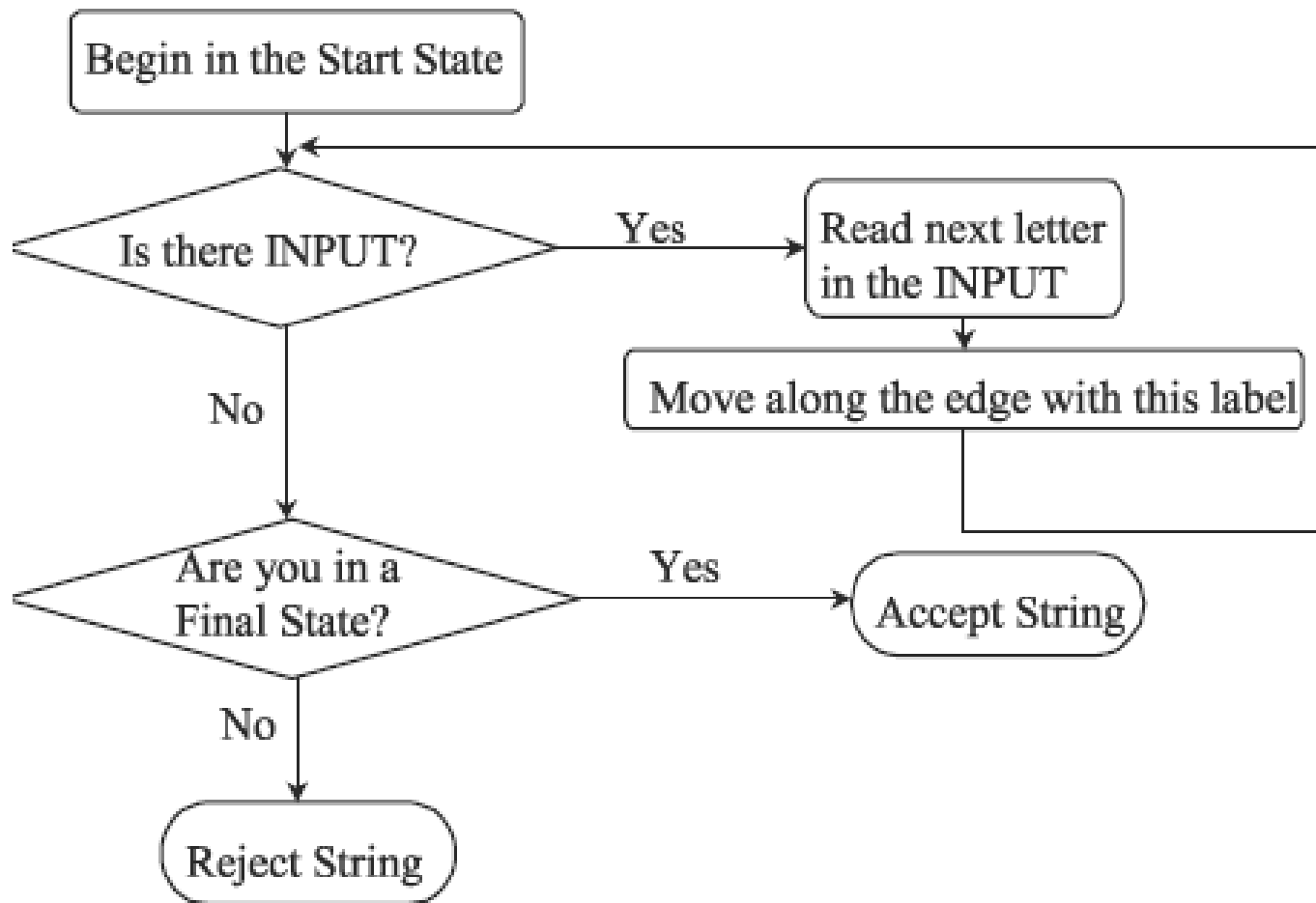
Notation



Alternative Notation



Flow Chart of FA

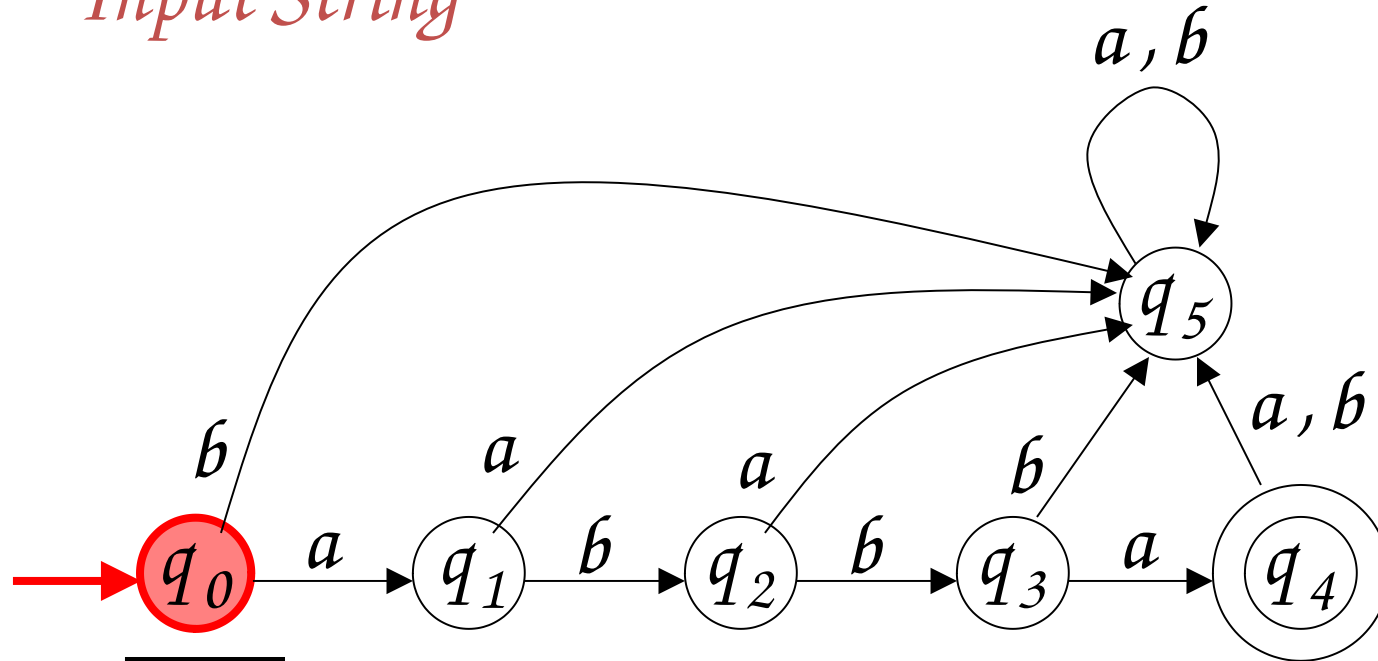


Initial Configuration

head

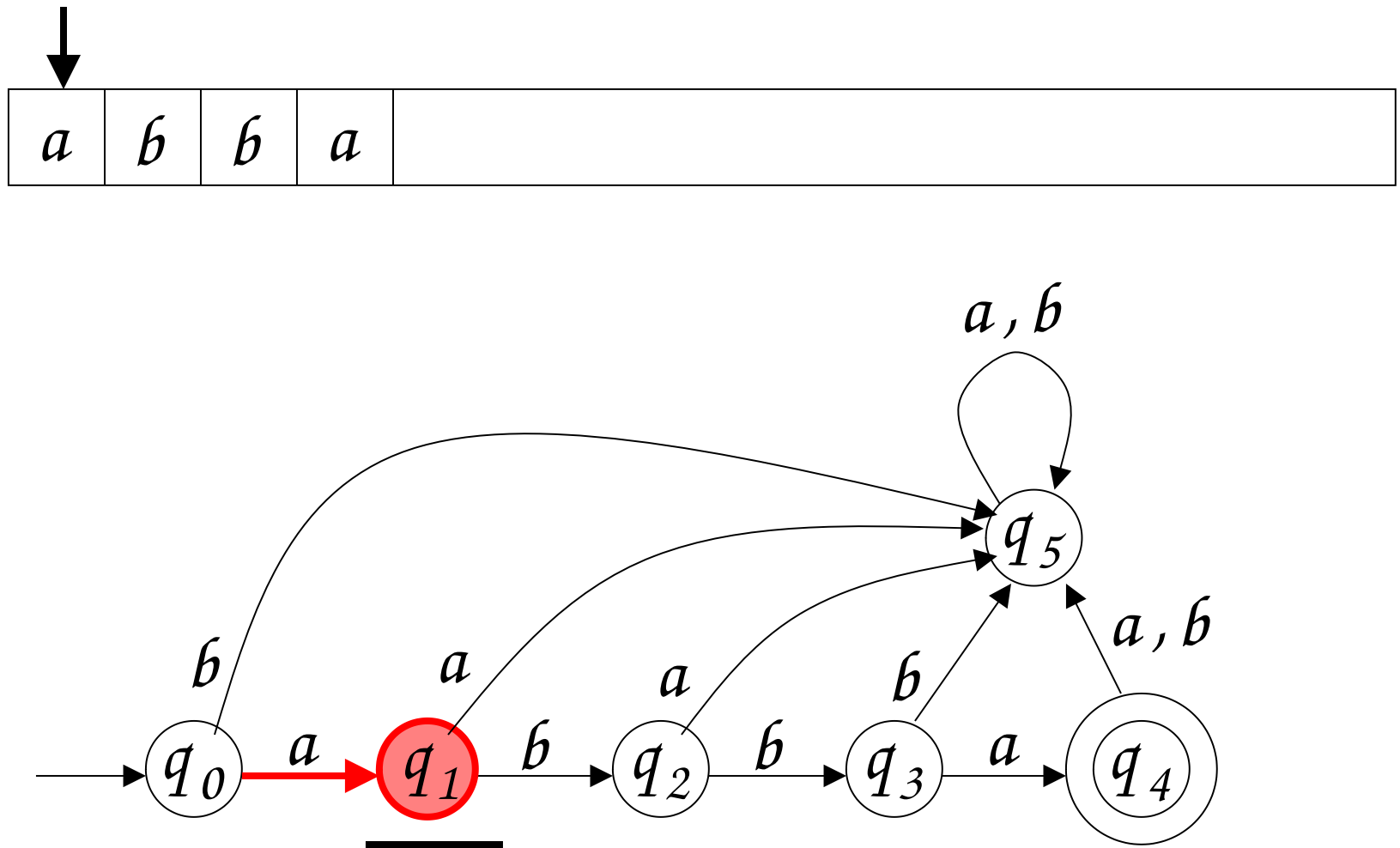


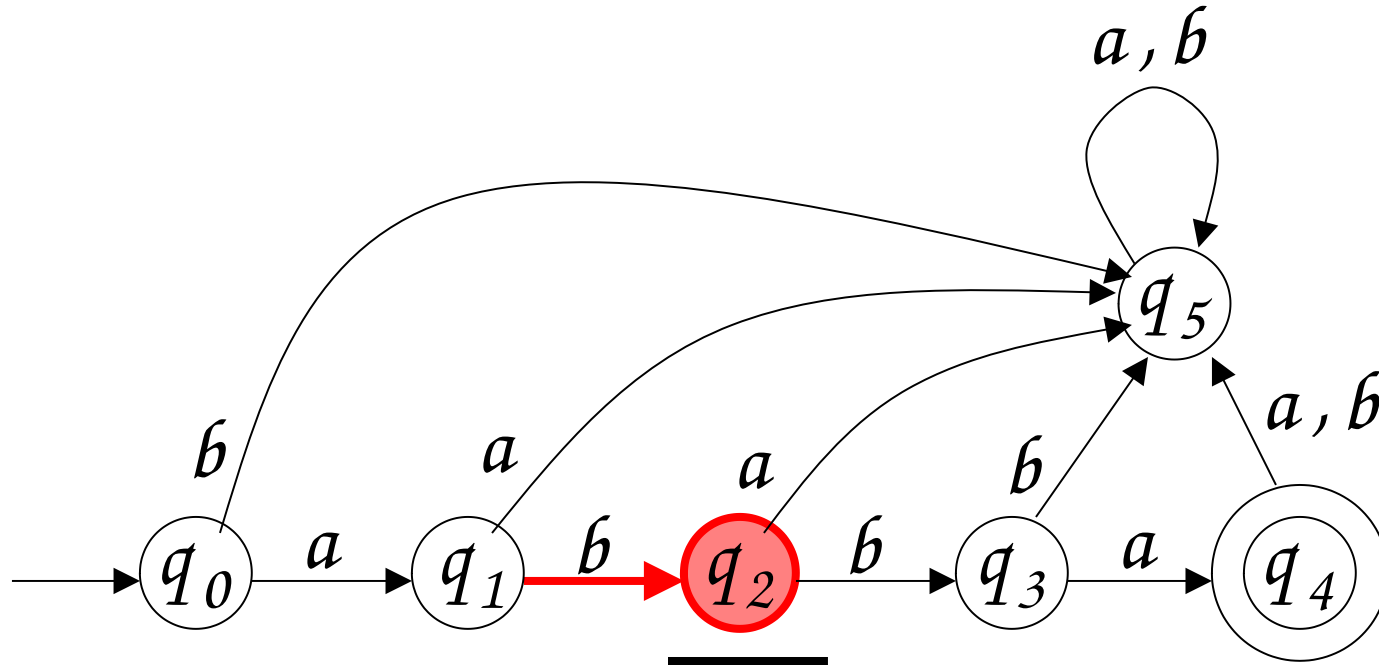
Input String

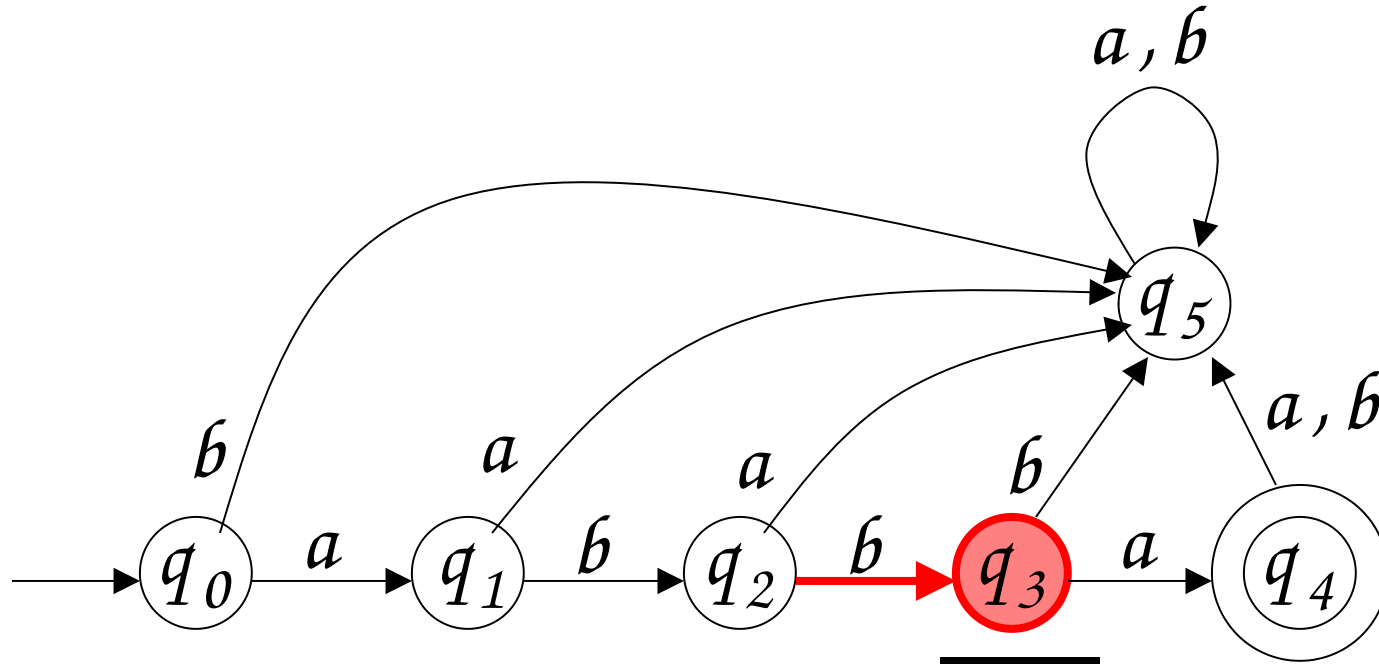


Initial state

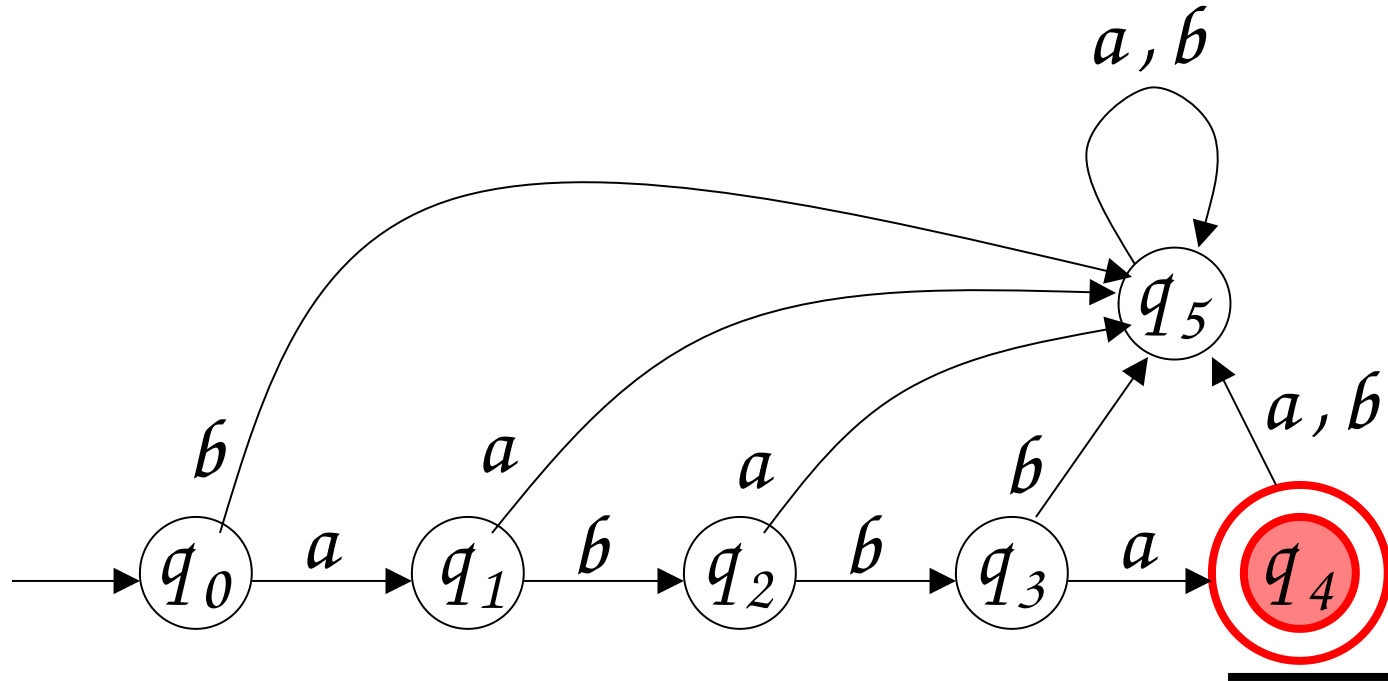
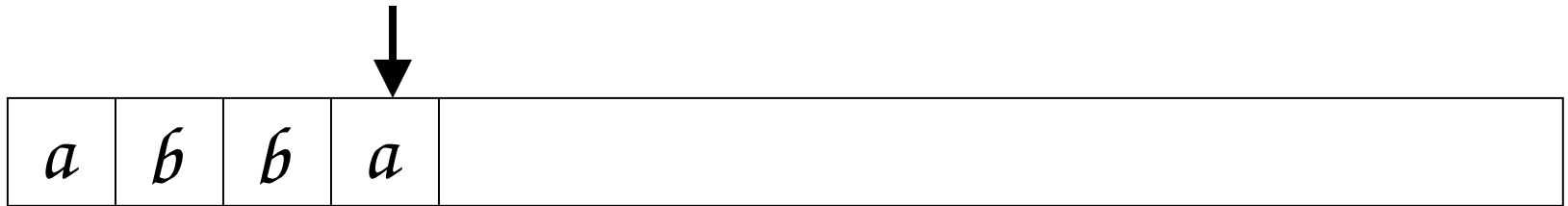
Scanning the Input







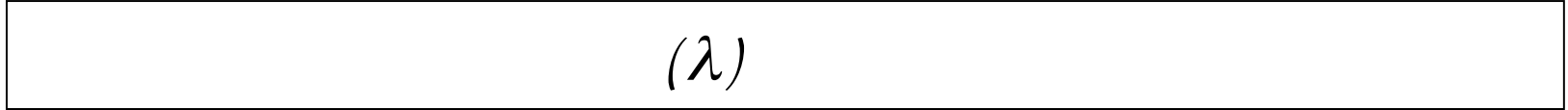
Input finished



accept

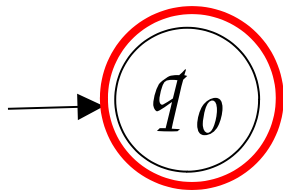


Empty Tape



(λ)

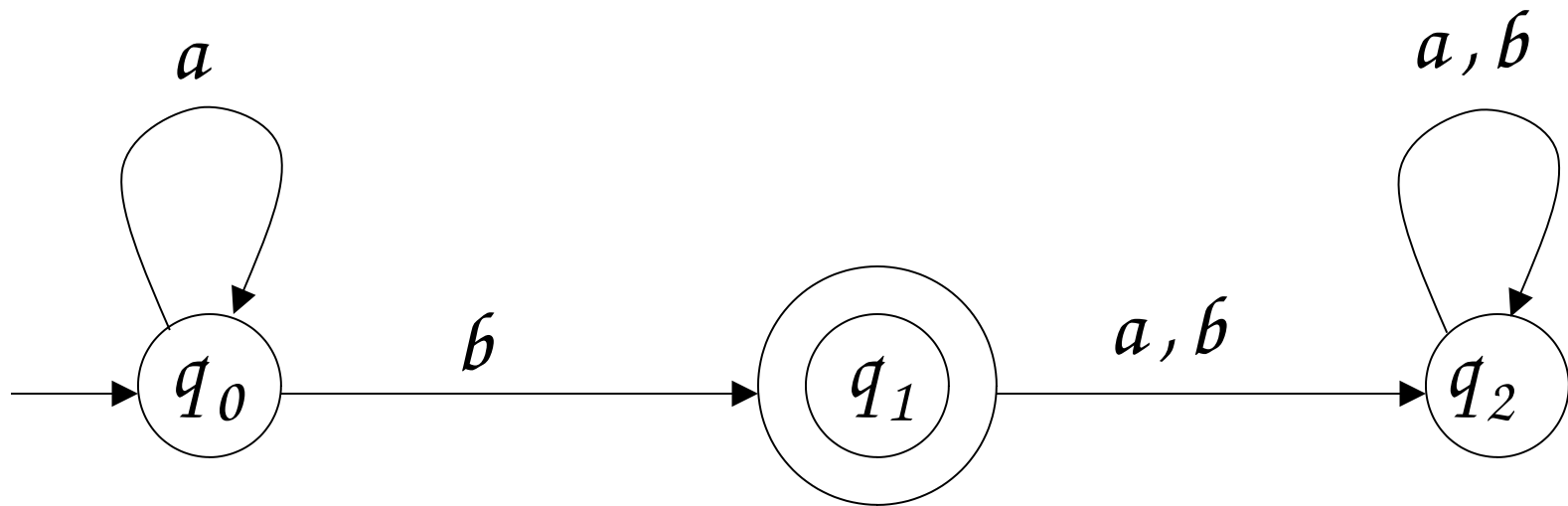
Input Finished



accept

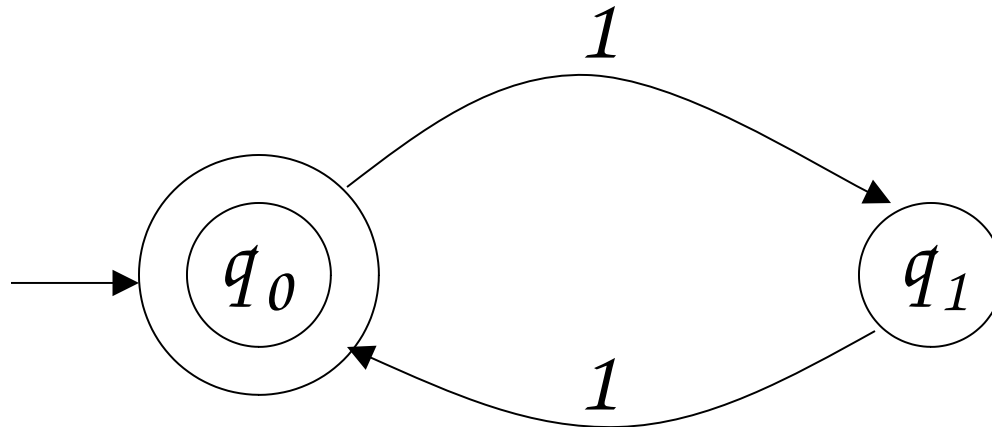
Language Accepted:

$$\mathcal{L} = \{a^n b : n \geq 0\}$$



Another Example

Alphabet: $\Sigma = \{1\}$



Language Accepted:

$$\begin{aligned} \text{EVEN} &= \{x : x \in \Sigma^* \text{ and } x \text{ is even}\} \\ &= \{\lambda, 11, 1111, 111111, \dots\} \end{aligned}$$

Formal Definition

Deterministic Finite Automaton (DFA)

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

Q : *set of states*

Σ : *input alphabet*

δ : *transition function*

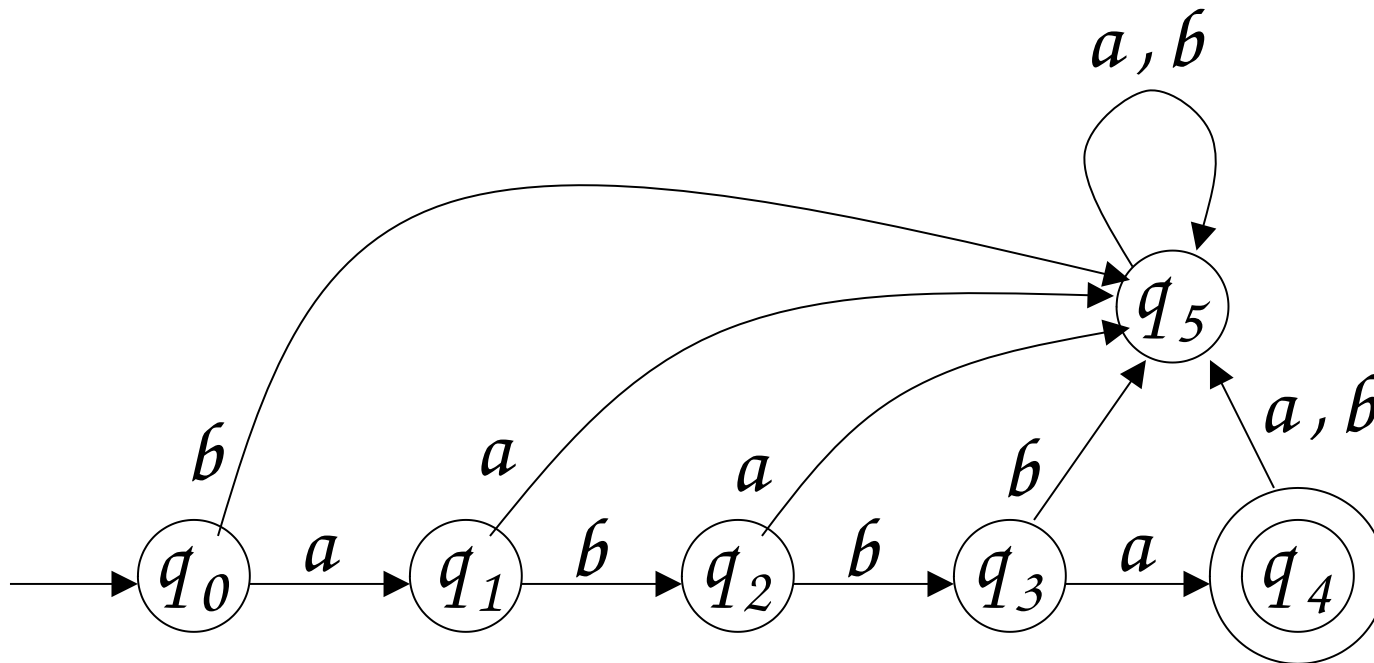
q_0 : *initial state (one)*

F : *set of accepting states*

Set of States Q

Example

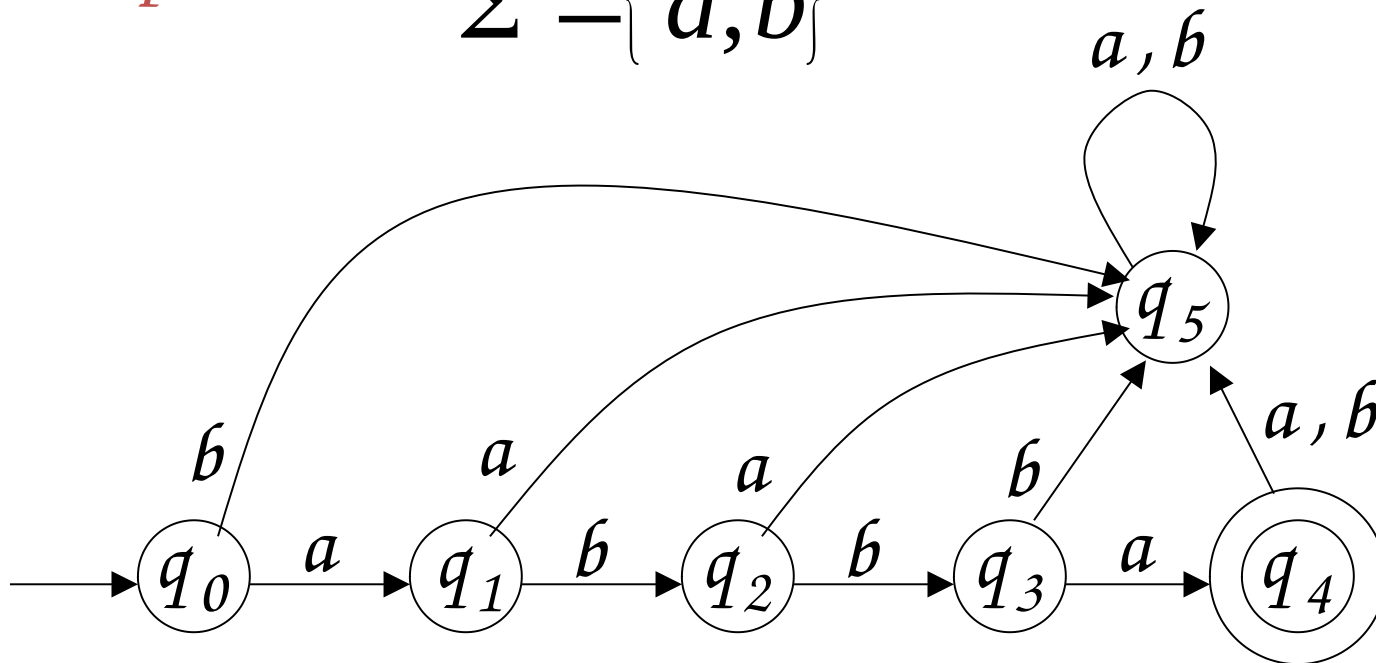
$$Q = \{ q_0, q_1, q_2, q_3, q_4, q_5 \}$$



Input Alphabet Σ

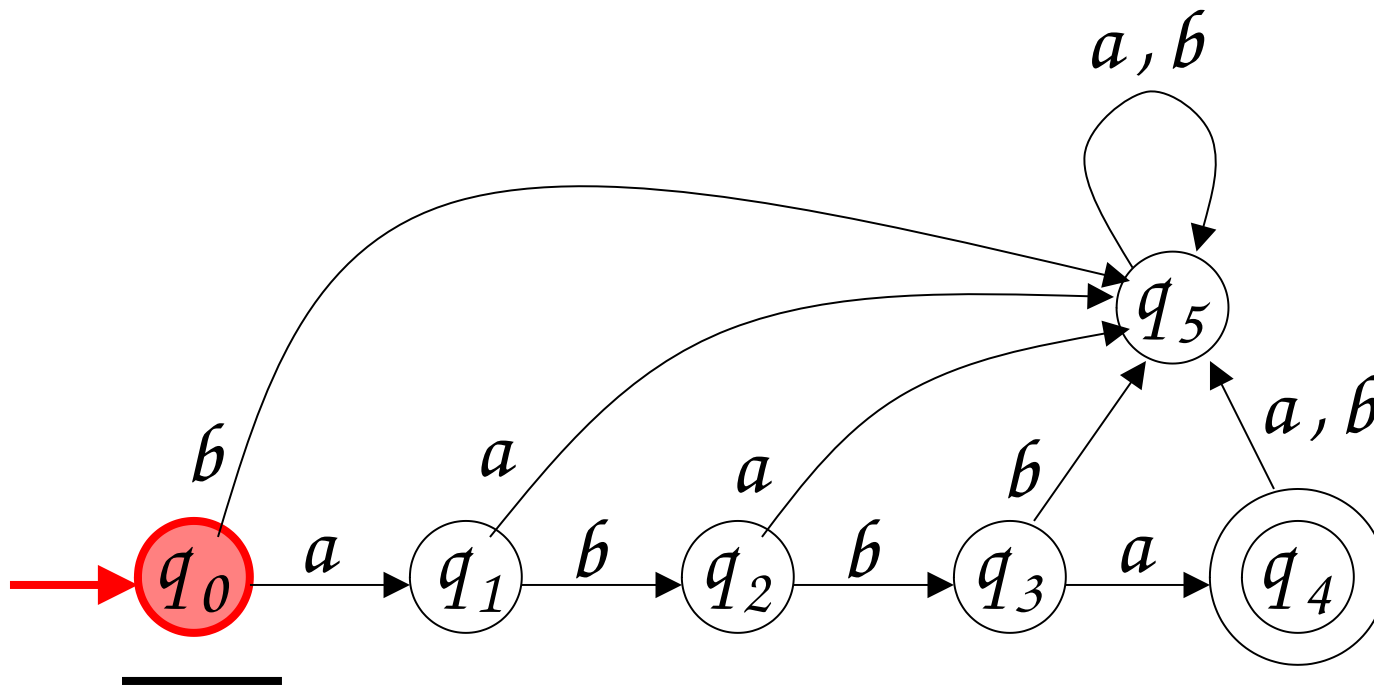
Example

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



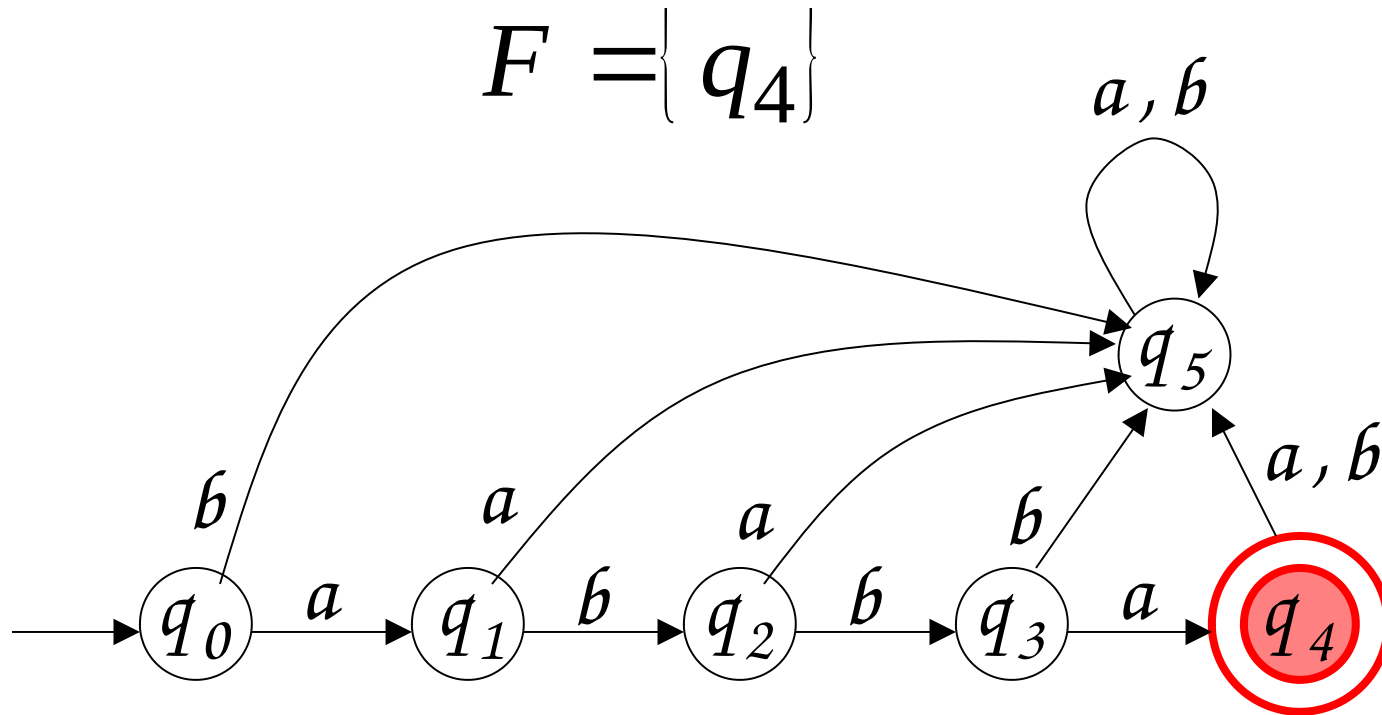
Initial State q_0

Example



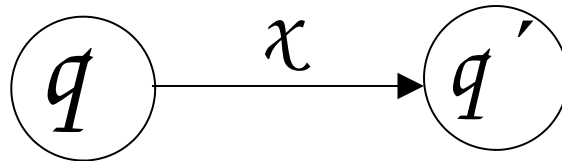
Set of Accepting States $\mathcal{F} \subseteq Q$

Example



Transition Function $\delta : Q \times \Sigma \rightarrow Q$

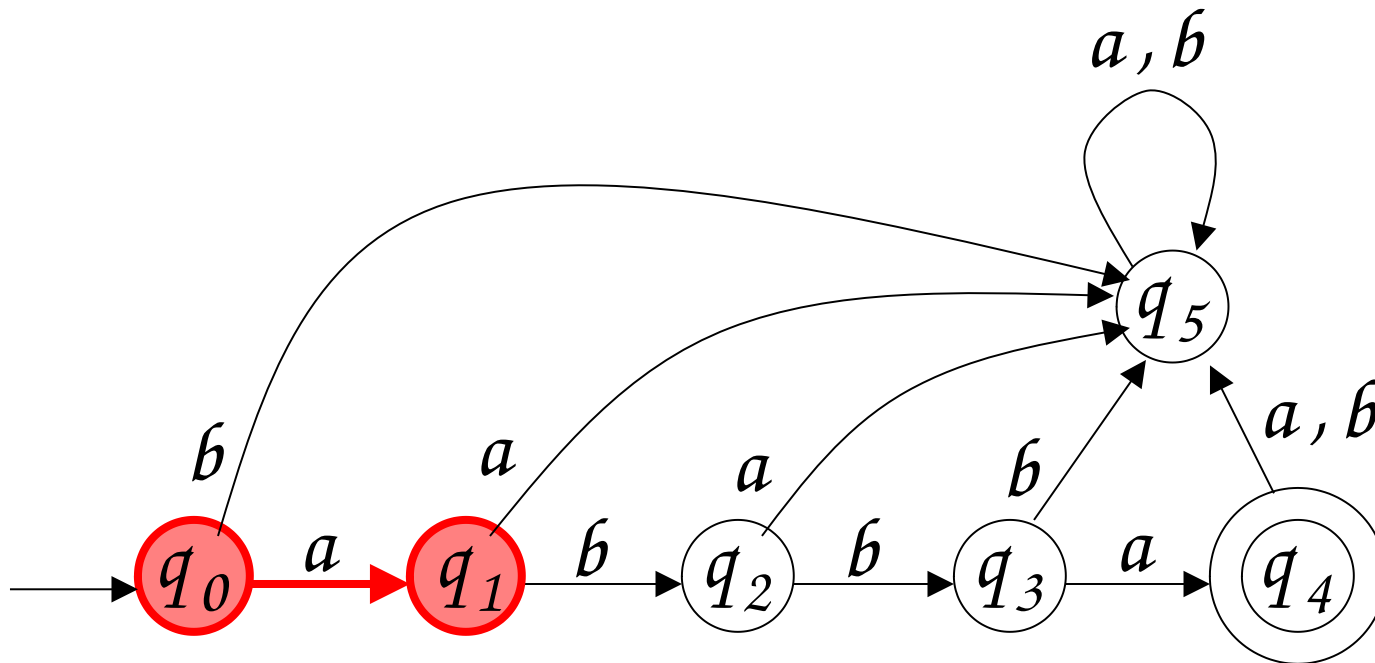
$$\delta(q, x) = q'$$



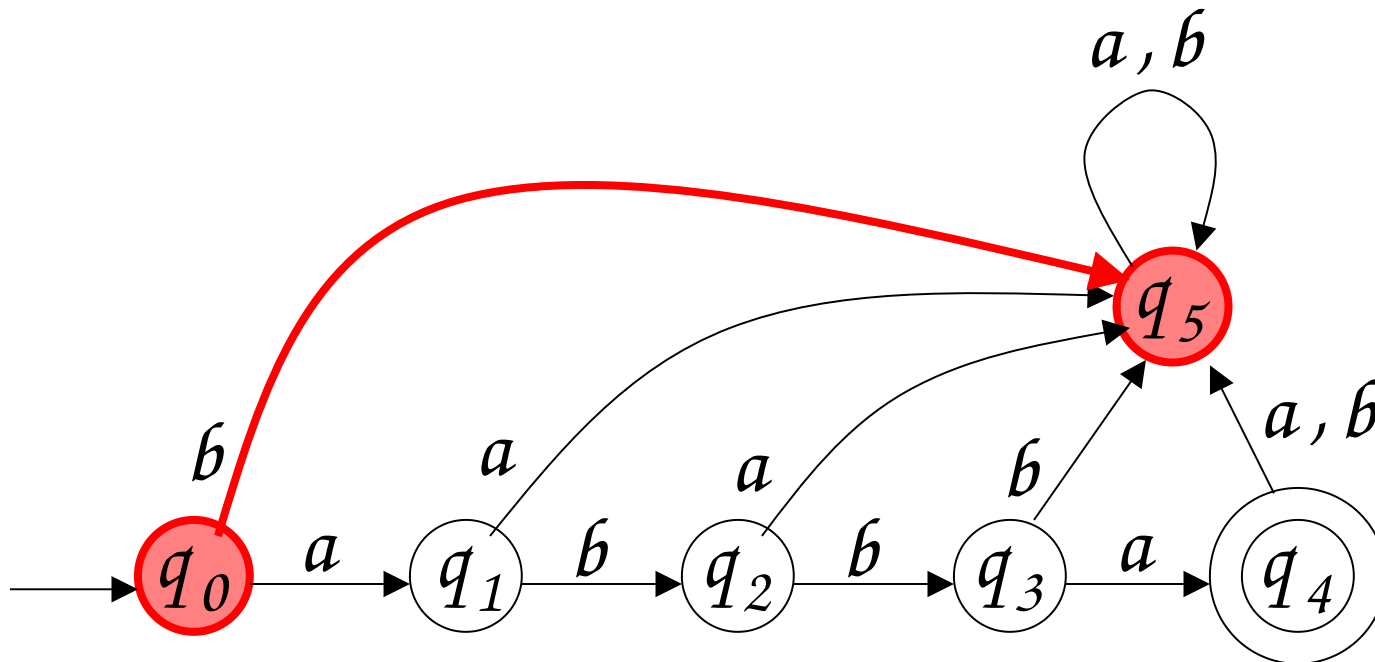
*Describes the result of a transition
from state q with symbol x*

Example:

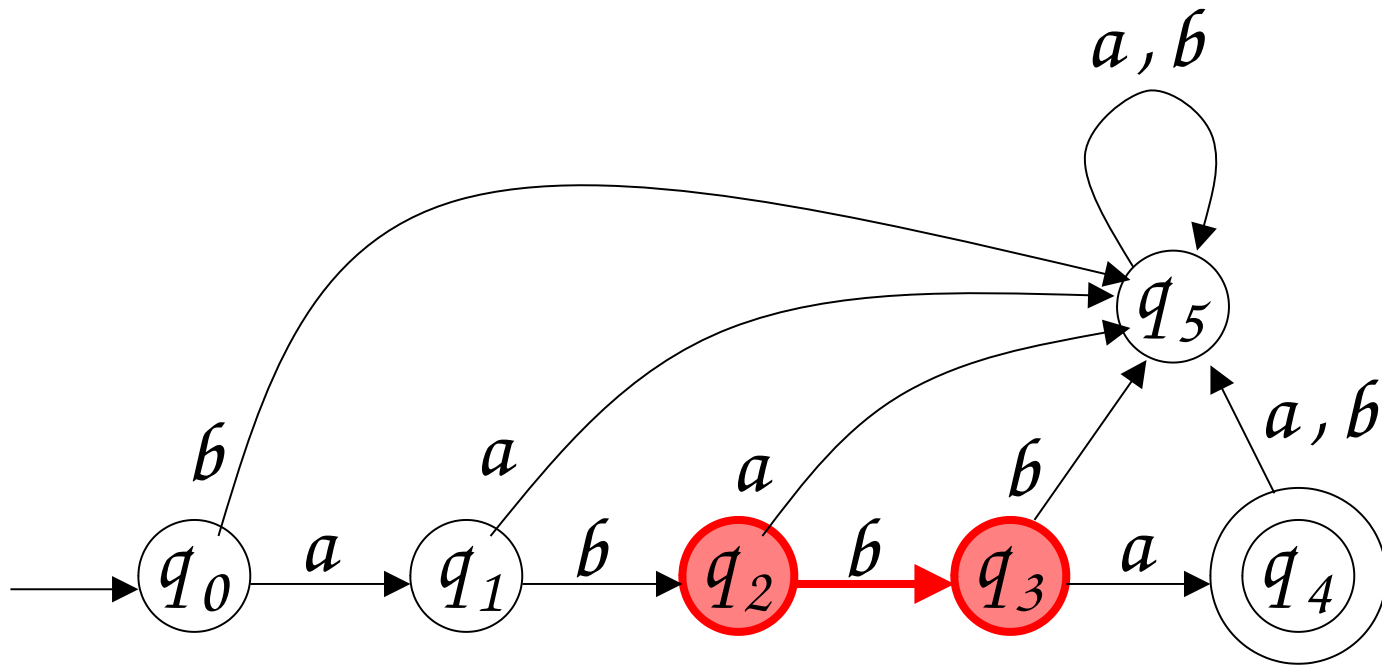
$$\delta(q_0, a) = q_1$$



$$\delta(q_0, b) = q_5$$



$$\delta(q_2, b) = q_3$$

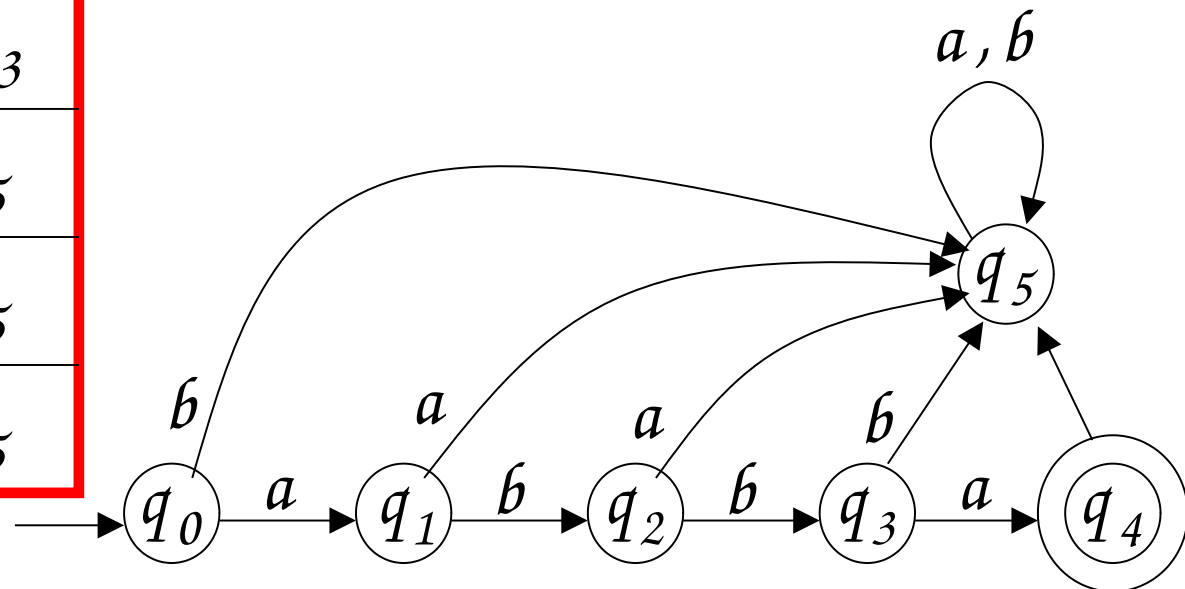


Transition Table for δ

symbols

states

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5



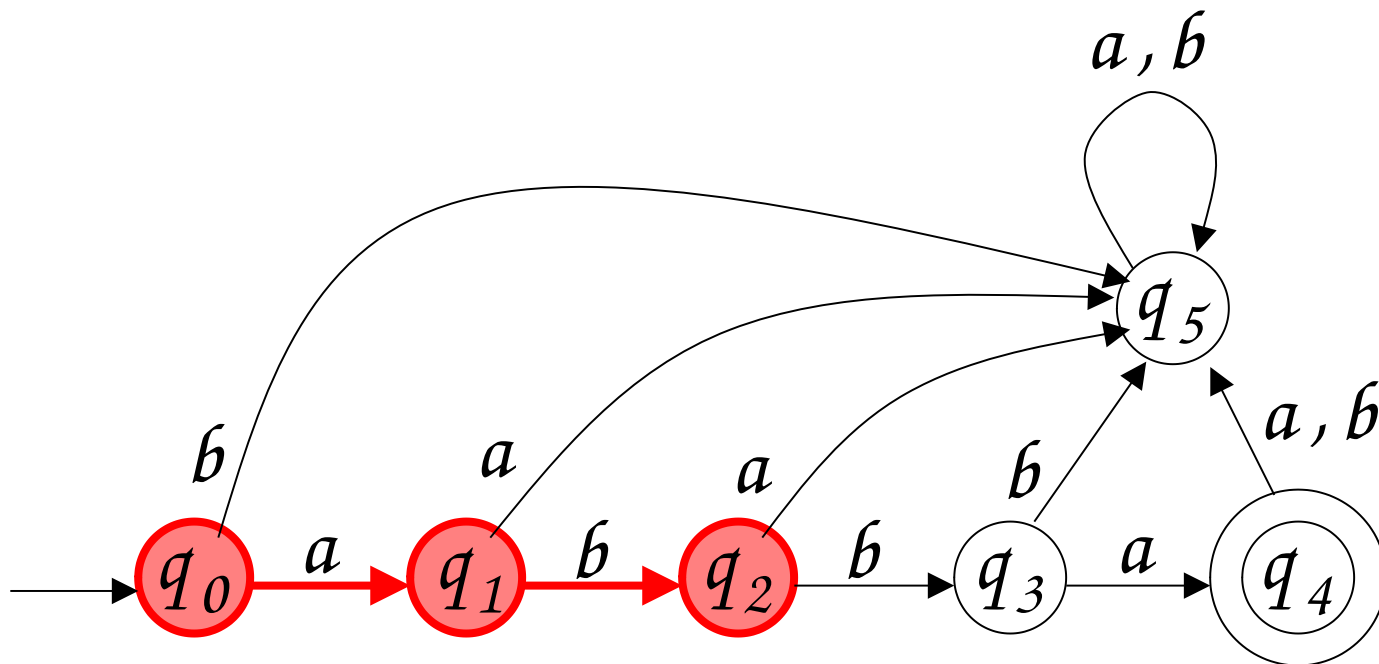
Extended Transition Function

$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

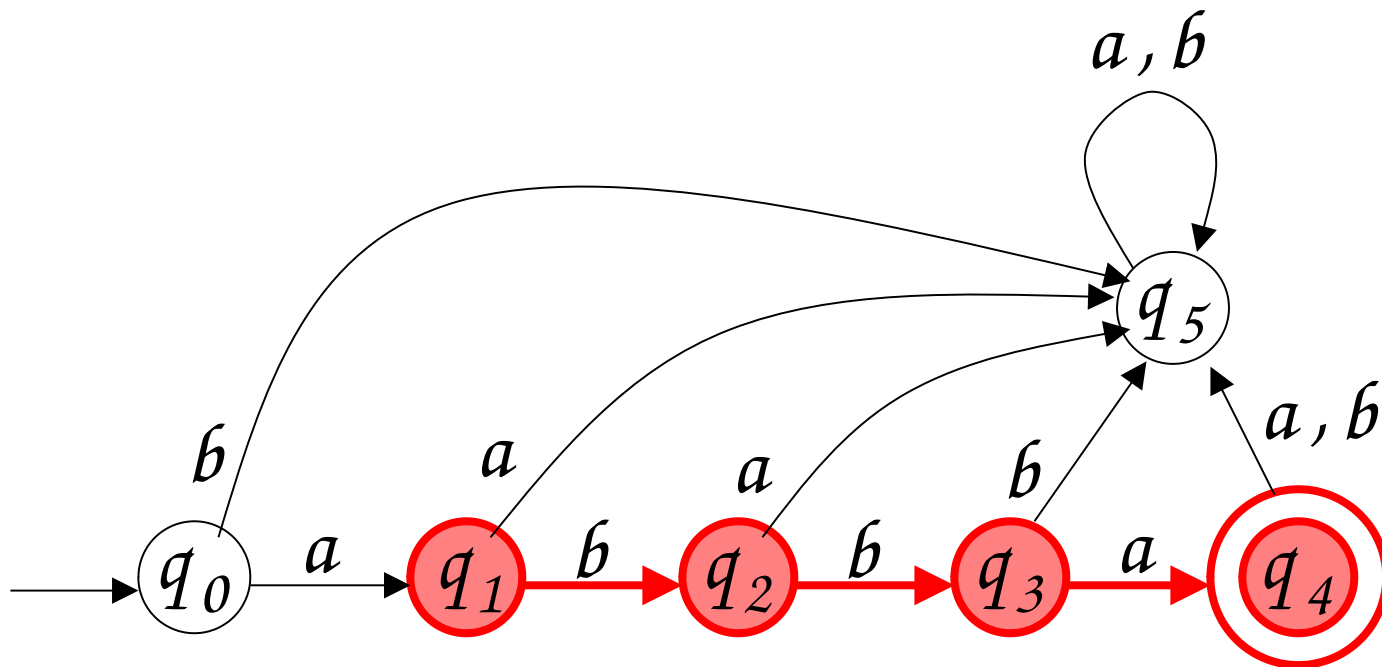
$$\delta^*(q, w) = q'$$

*Describes the resulting state
after scanning string w from state q*

Example: $\delta^*(q_0, ab) = q_2$



$$\delta^*(q_1, bba) = q_4$$



Special case:

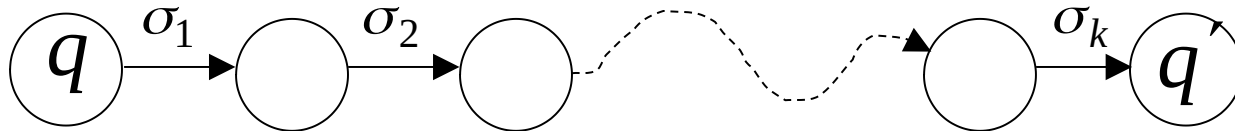
for any state q

$$\delta^*(q, \lambda) = q$$

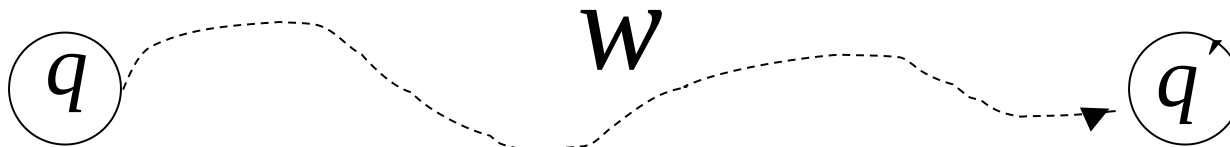
In general: $\delta^*(q, w) = q'$

implies that there is a walk of transitions

$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



states may be repeated



Language Accepted by DFA

Language of DFA \mathcal{M}

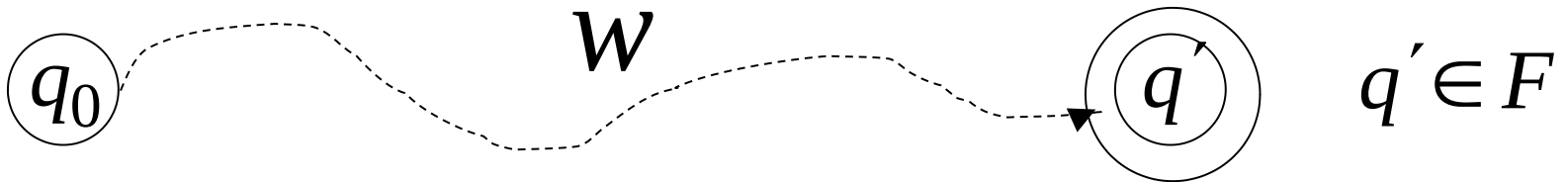
*it is denoted as $\mathcal{L}(\mathcal{M})$ and contains
all the strings accepted by \mathcal{M}*

*We say that a language \mathcal{L}'
is accepted (or recognized)
by DFA \mathcal{M} if $\mathcal{L}(\mathcal{M}) = \mathcal{L}'$*

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

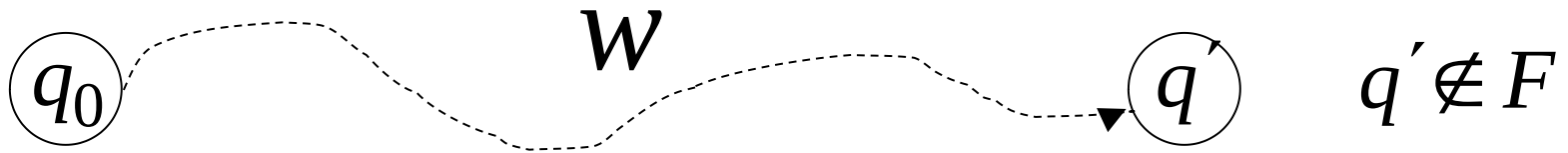
Language accepted by M :

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



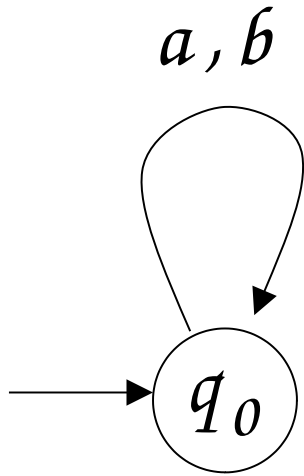
Language rejected by \mathcal{M} :

$$\overline{\mathcal{L}(\mathcal{M})} = \left\{ w \in \Sigma^* : \delta^*(q_0, w) \notin \mathcal{F} \right\}$$



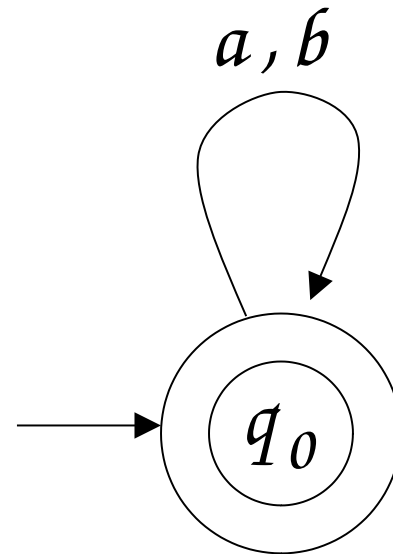
More DFA Examples

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



$$\mathcal{L}(\mathcal{M}) = \{ \}$$

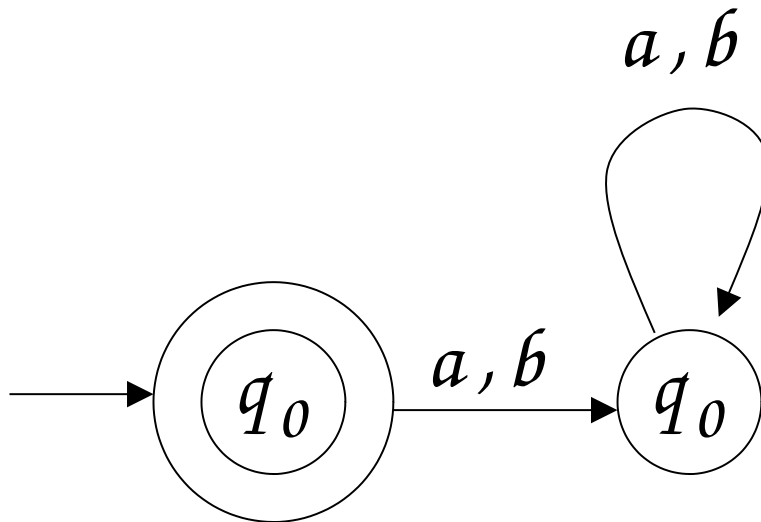
Empty language



$$\mathcal{L}(\mathcal{M}) = \Sigma^*$$

All strings

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

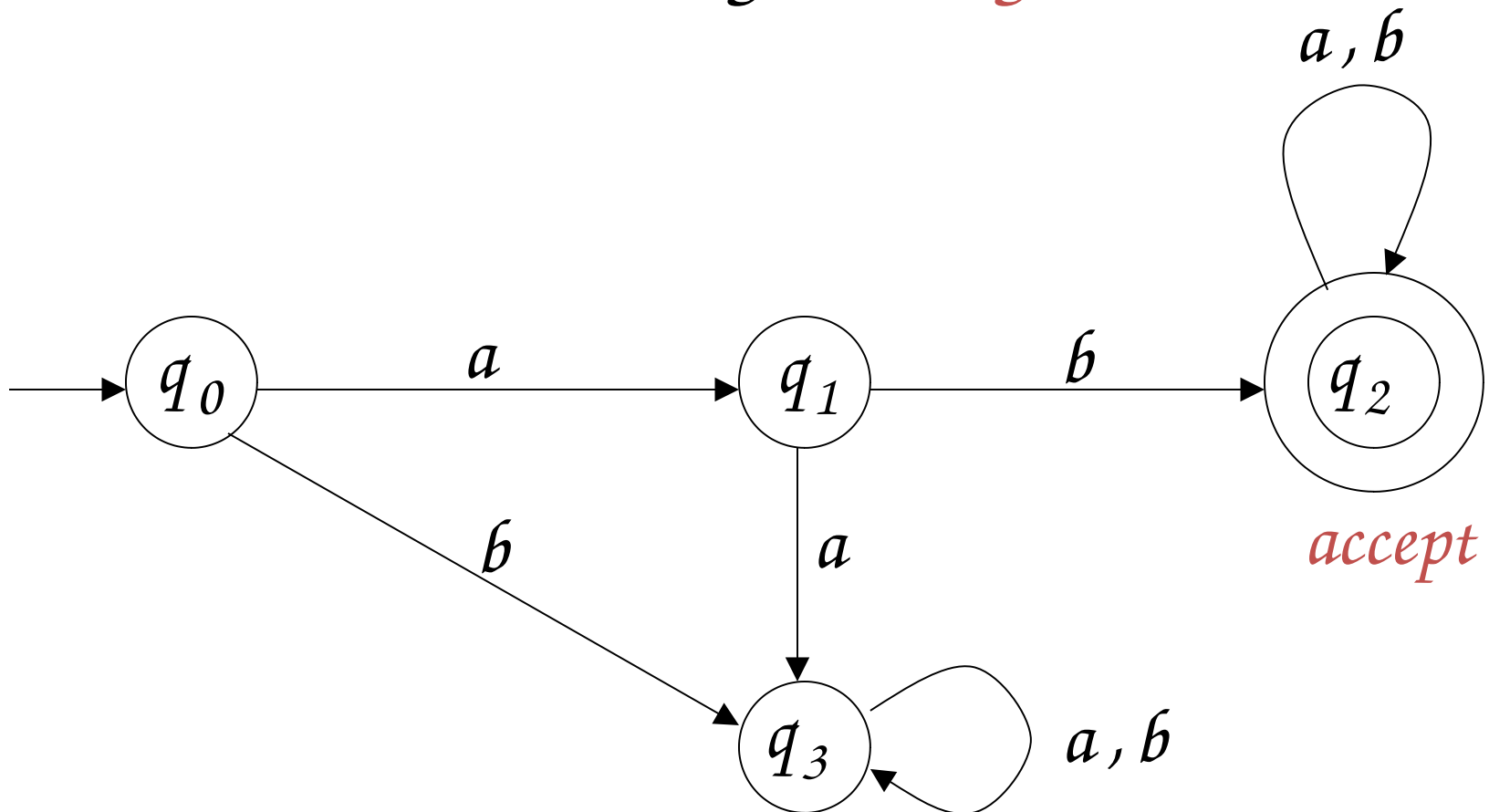


$$\mathcal{L}(\mathcal{M}) = \{\lambda\}$$

Language of the empty string

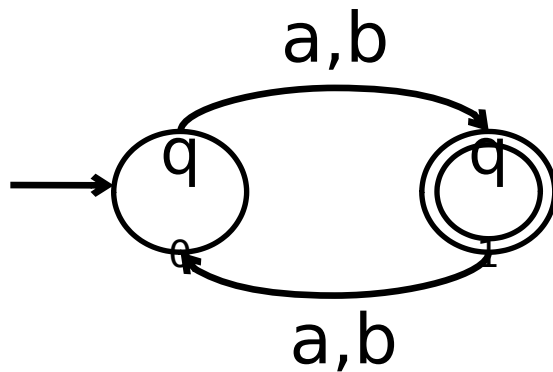
$$\Sigma = \{a, b\}$$

$L(M)$ = { all strings with prefix ab }
 = { all strings that *begins with ab* }

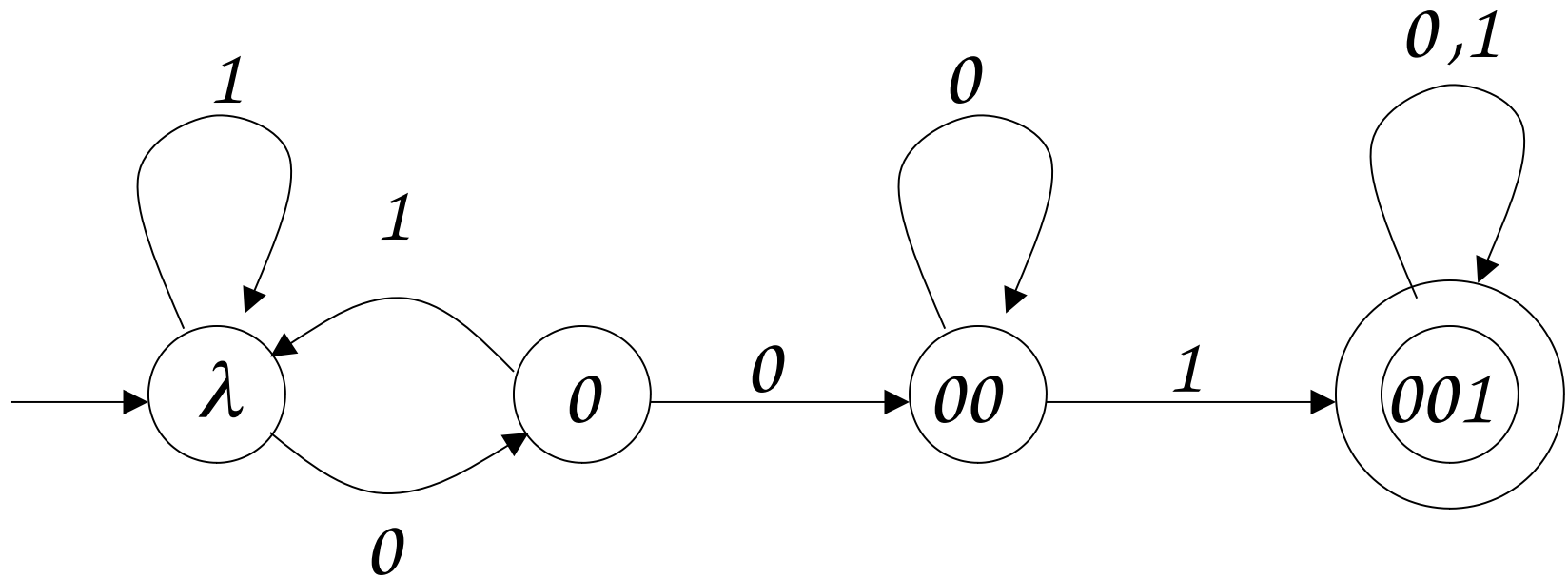


Infinite Languages

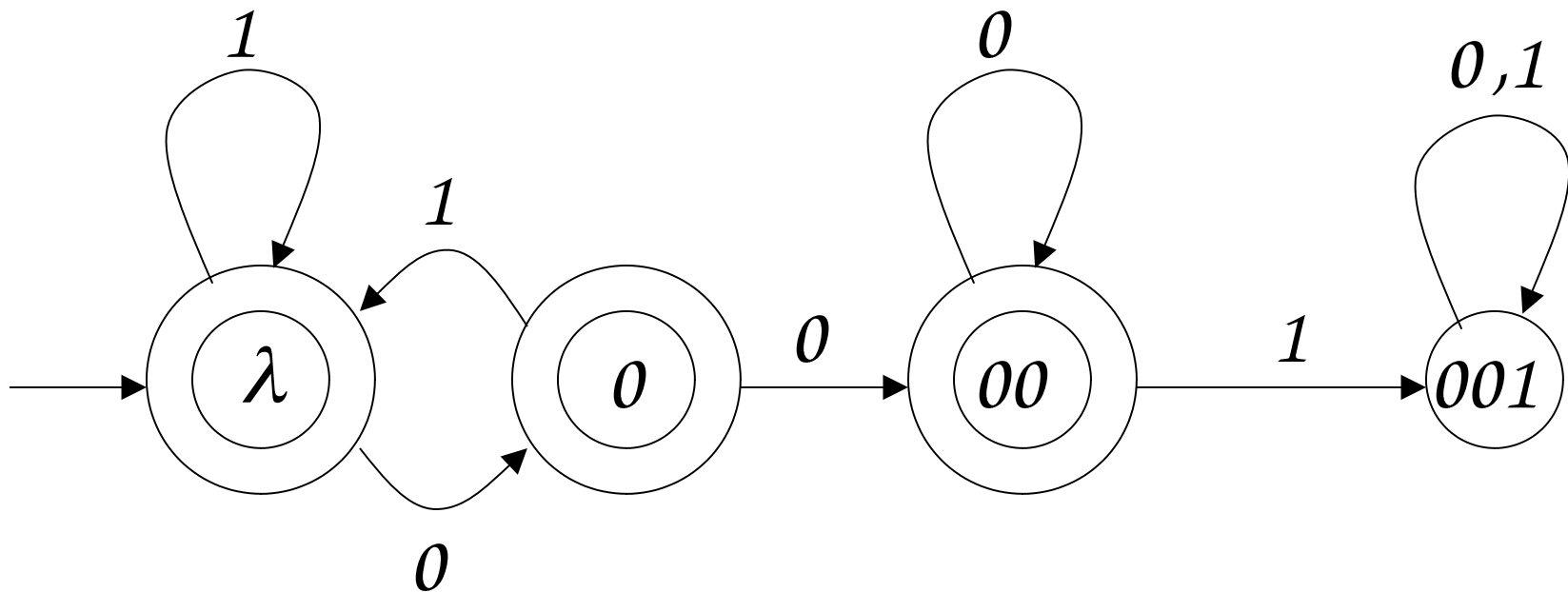
- *Example:* $\Sigma = \{a,b\}$
 $L = \{s : |s| \text{ is odd}\}$

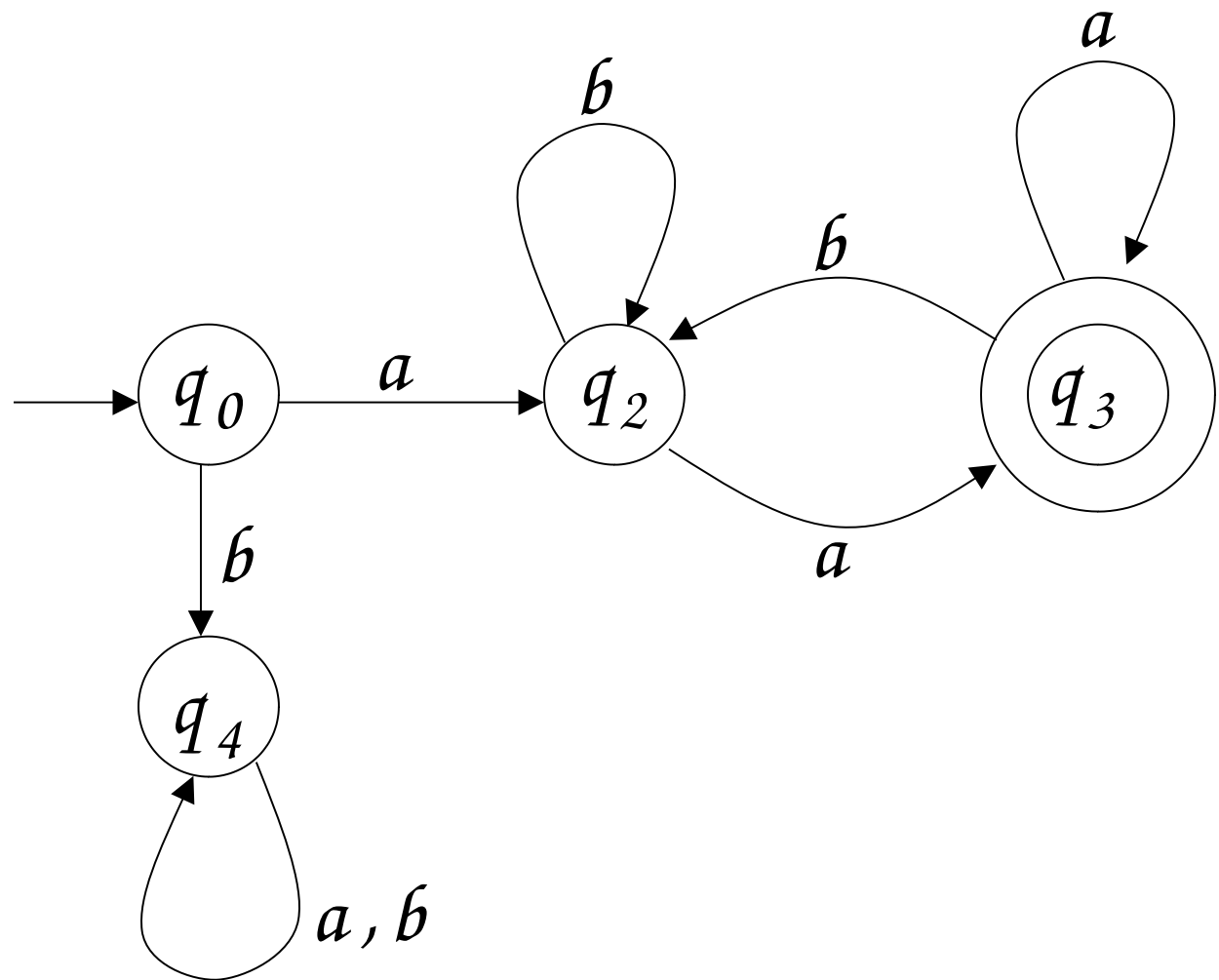


$$\mathcal{L}(\mathcal{M}) = \{ \text{all binary strings containing} \\ \text{substring } 001 \}$$



$\mathcal{L}(\mathcal{M}) = \{ \text{all binary strings without} \\ \text{substring } 001 \}$





$$\mathcal{L}(\mathcal{M}) = \left\{ awa : w \in \{a, b\}^* \right\}$$

