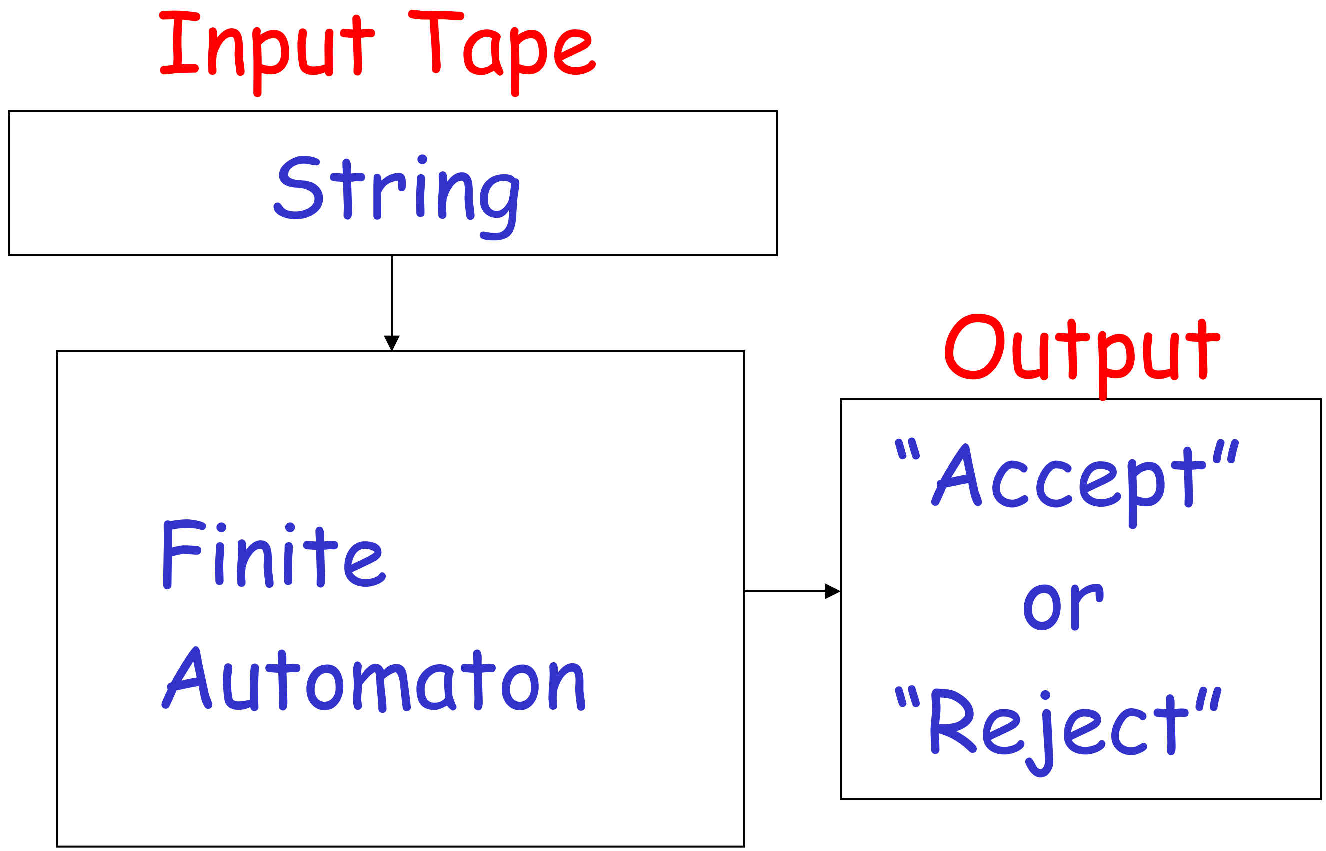
Deterministic

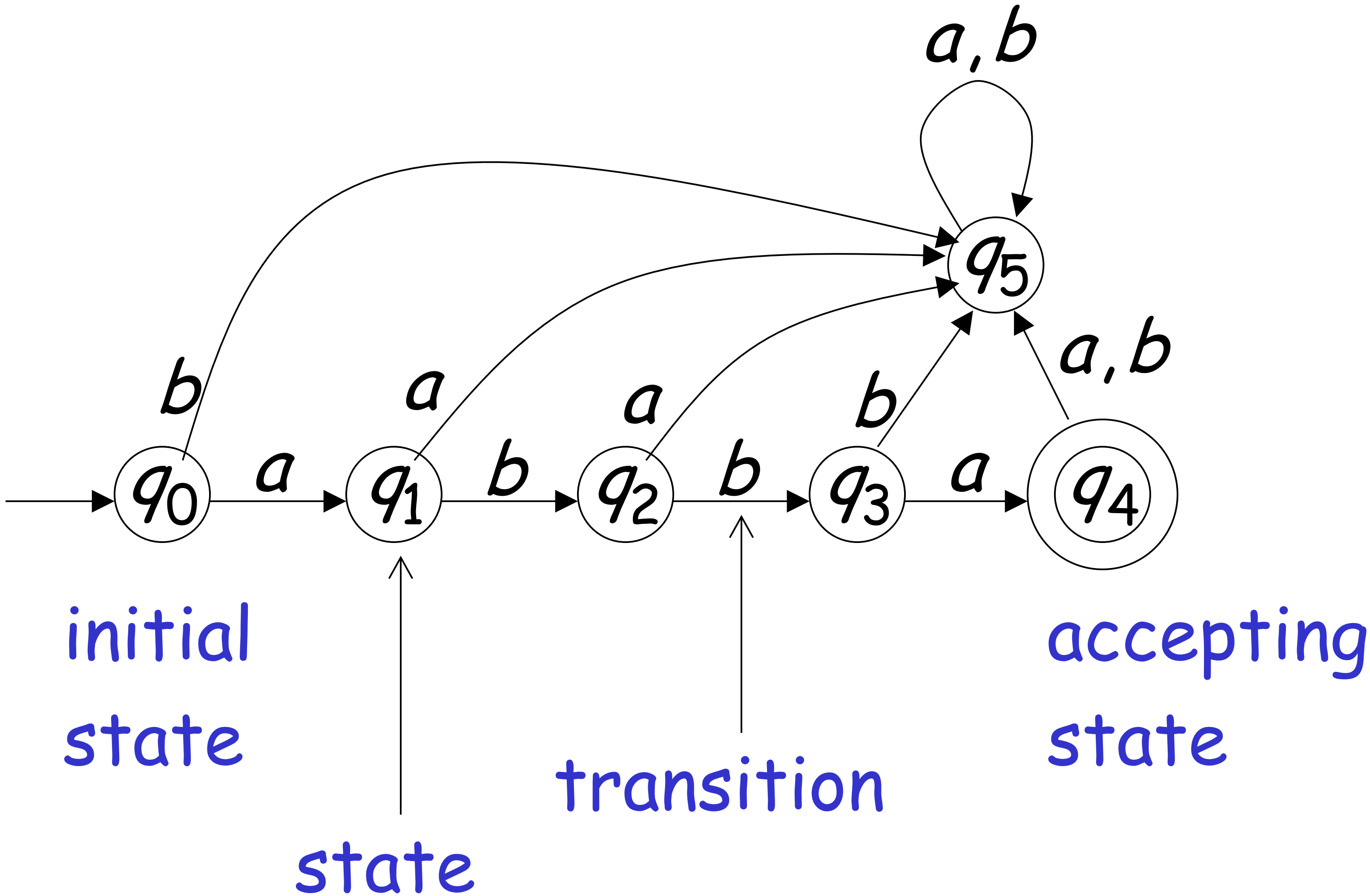
Finite Automata

And Regular Languages

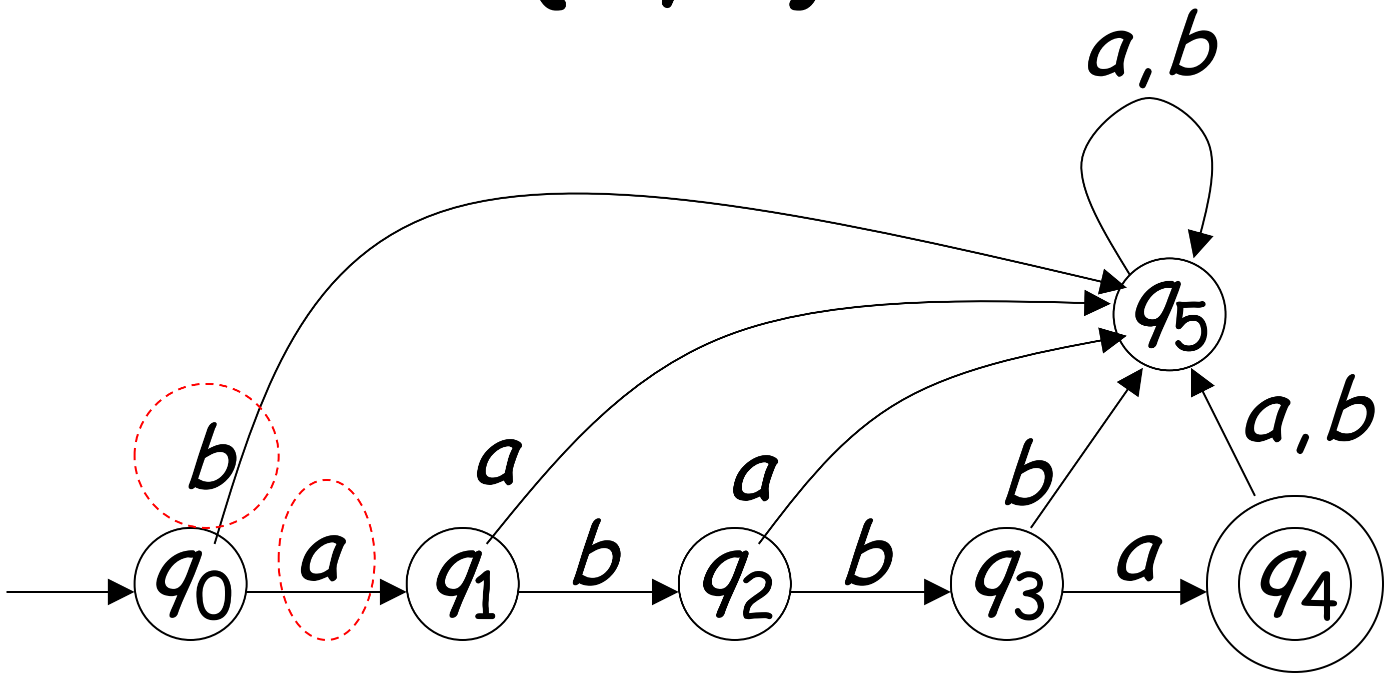
Deterministic Finite Automaton (DFA)



Transition Graph

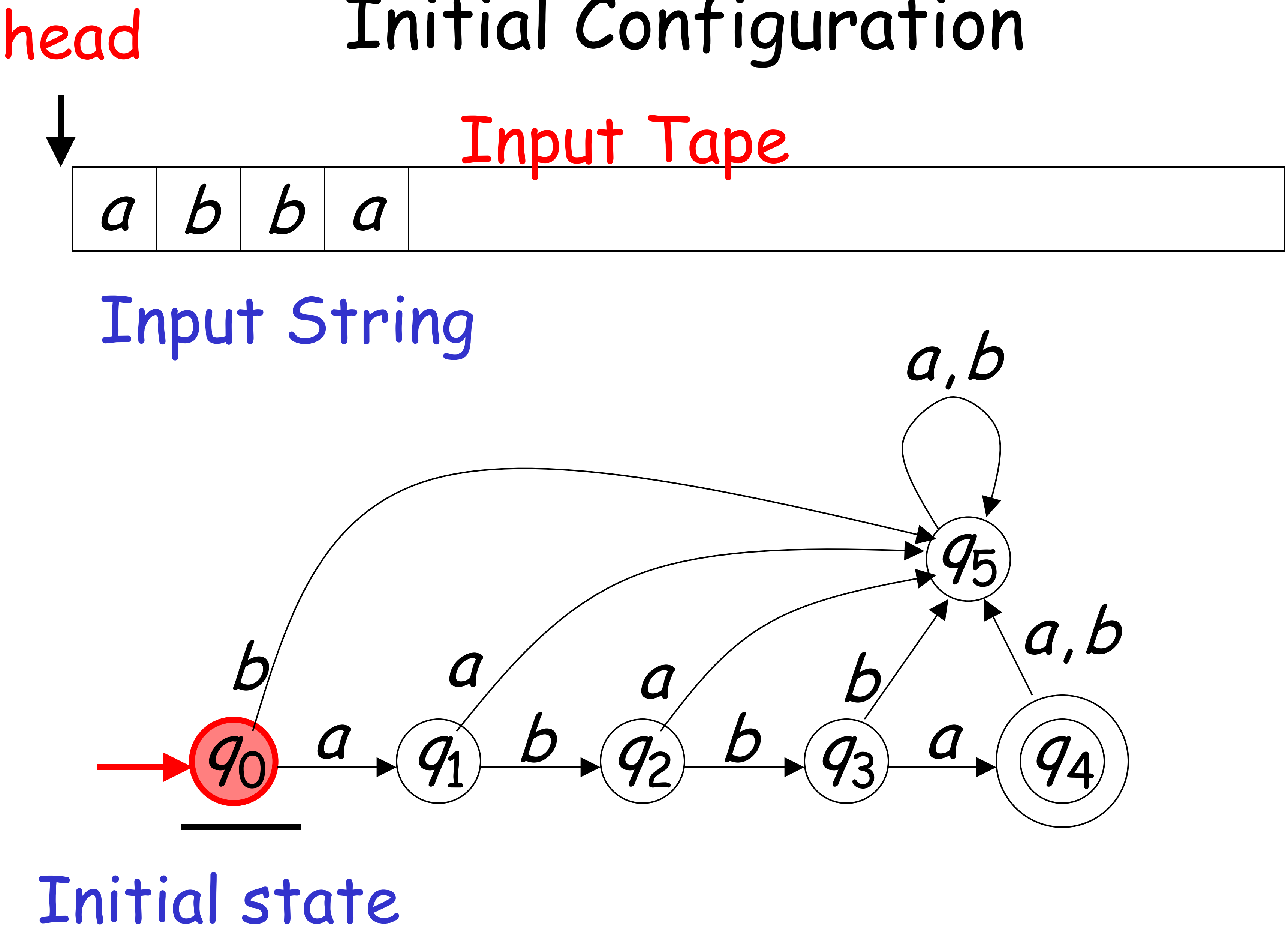


Alphabet $\Sigma = \{a, b\}$

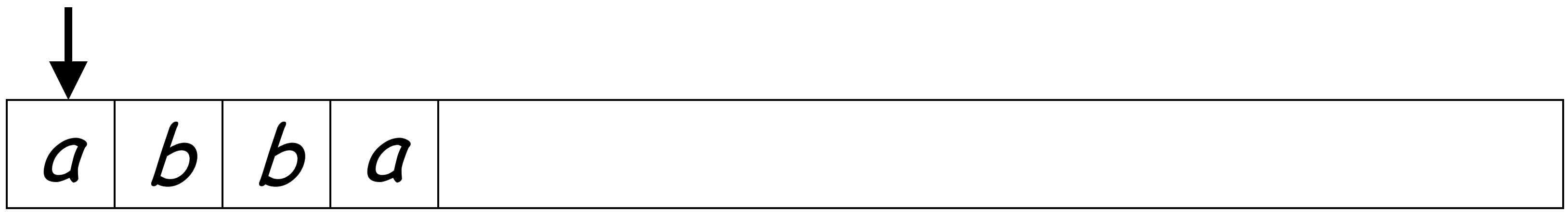


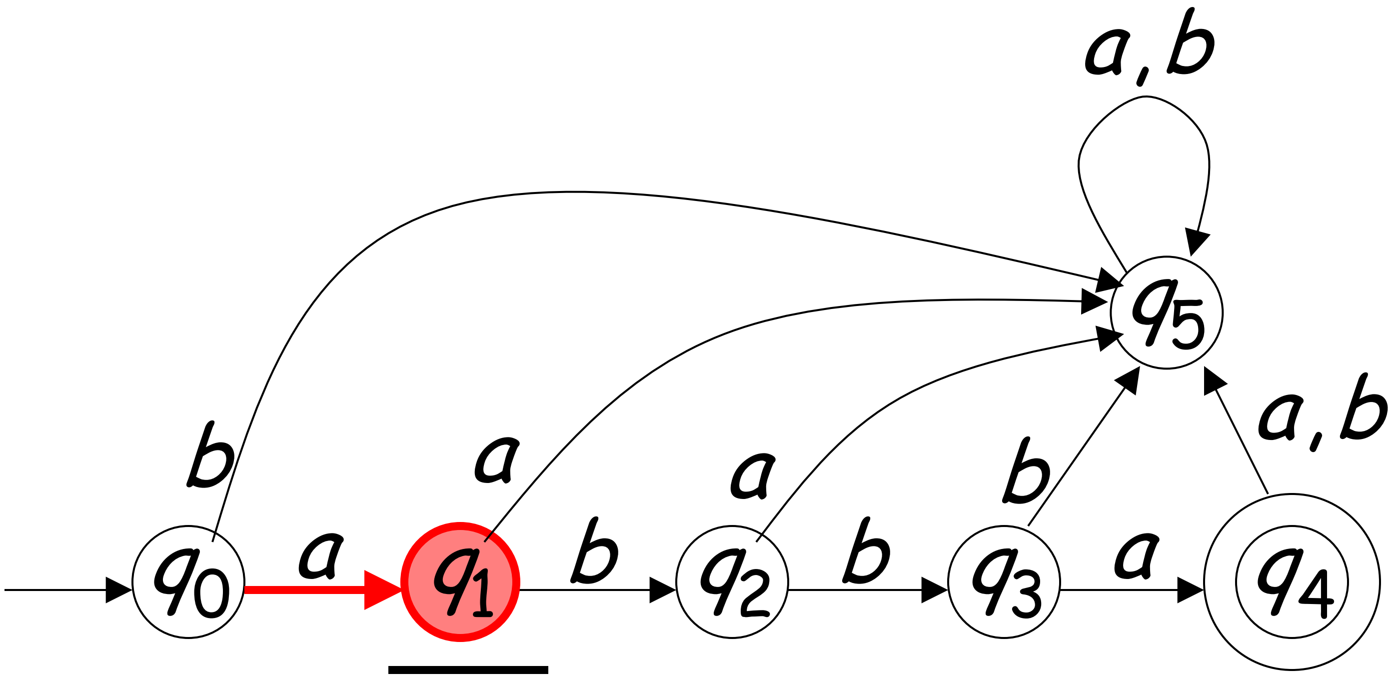
For every state, there is a transition for every symbol in the alphabet

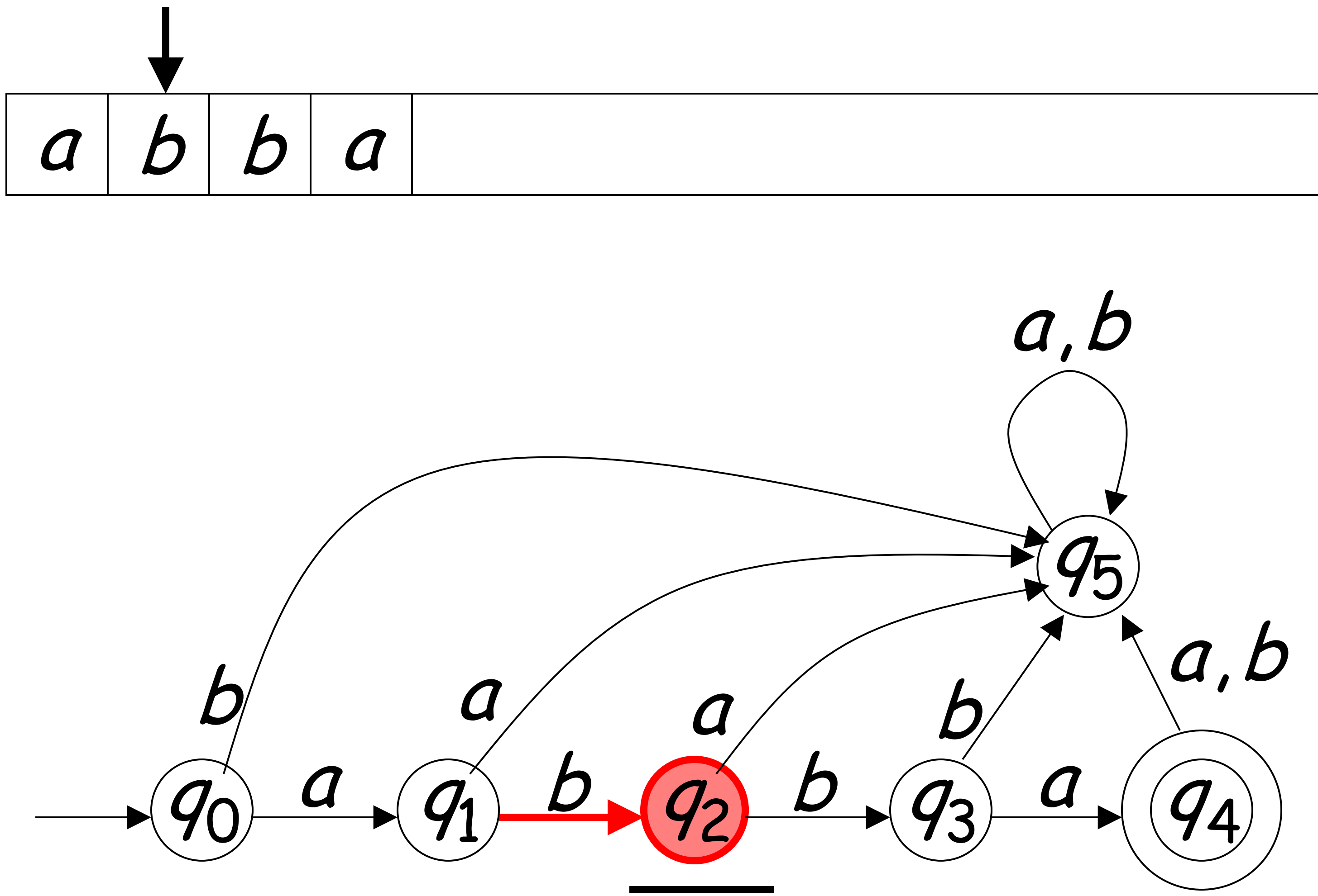
Initial Configuration

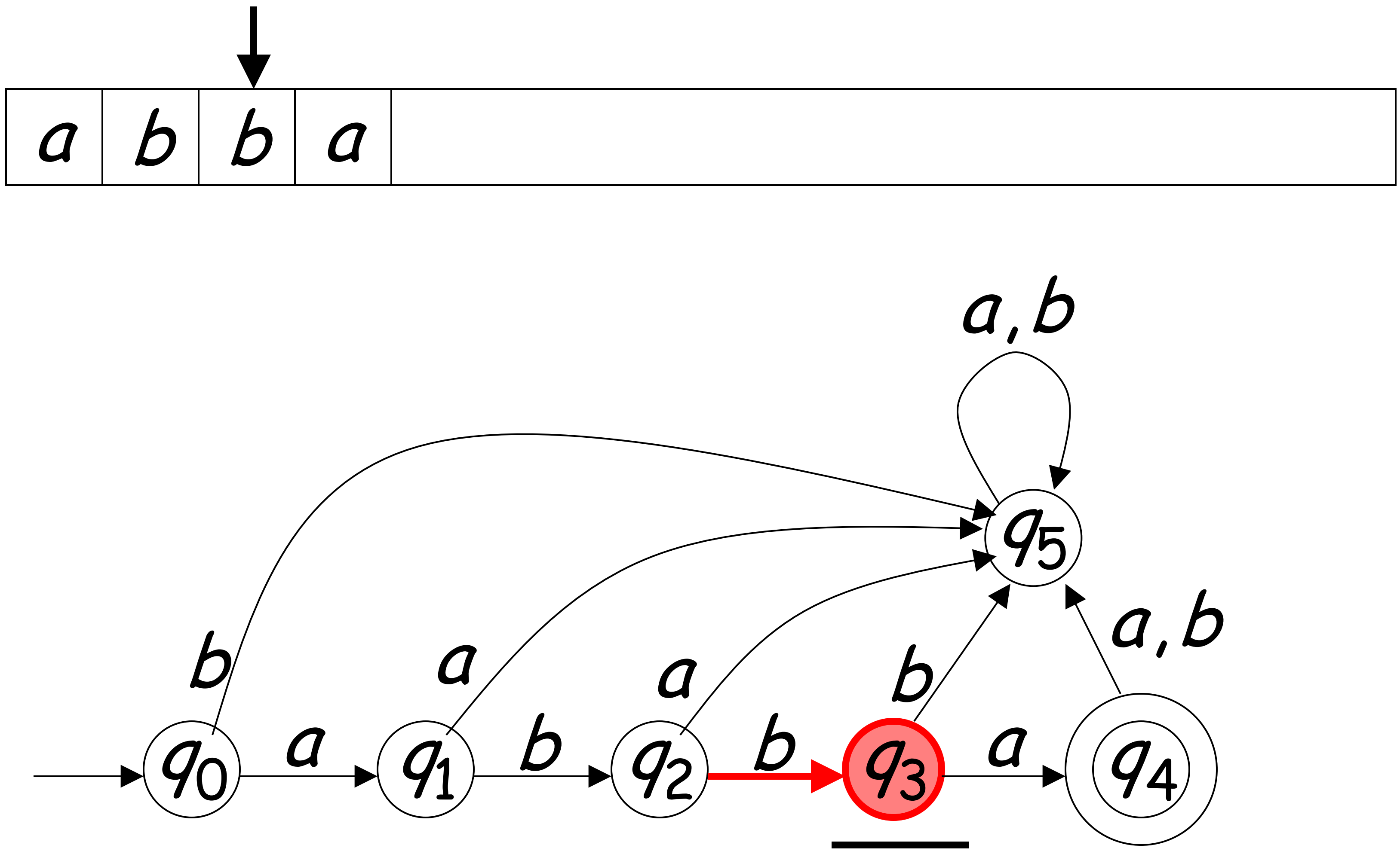


Scanning the Input

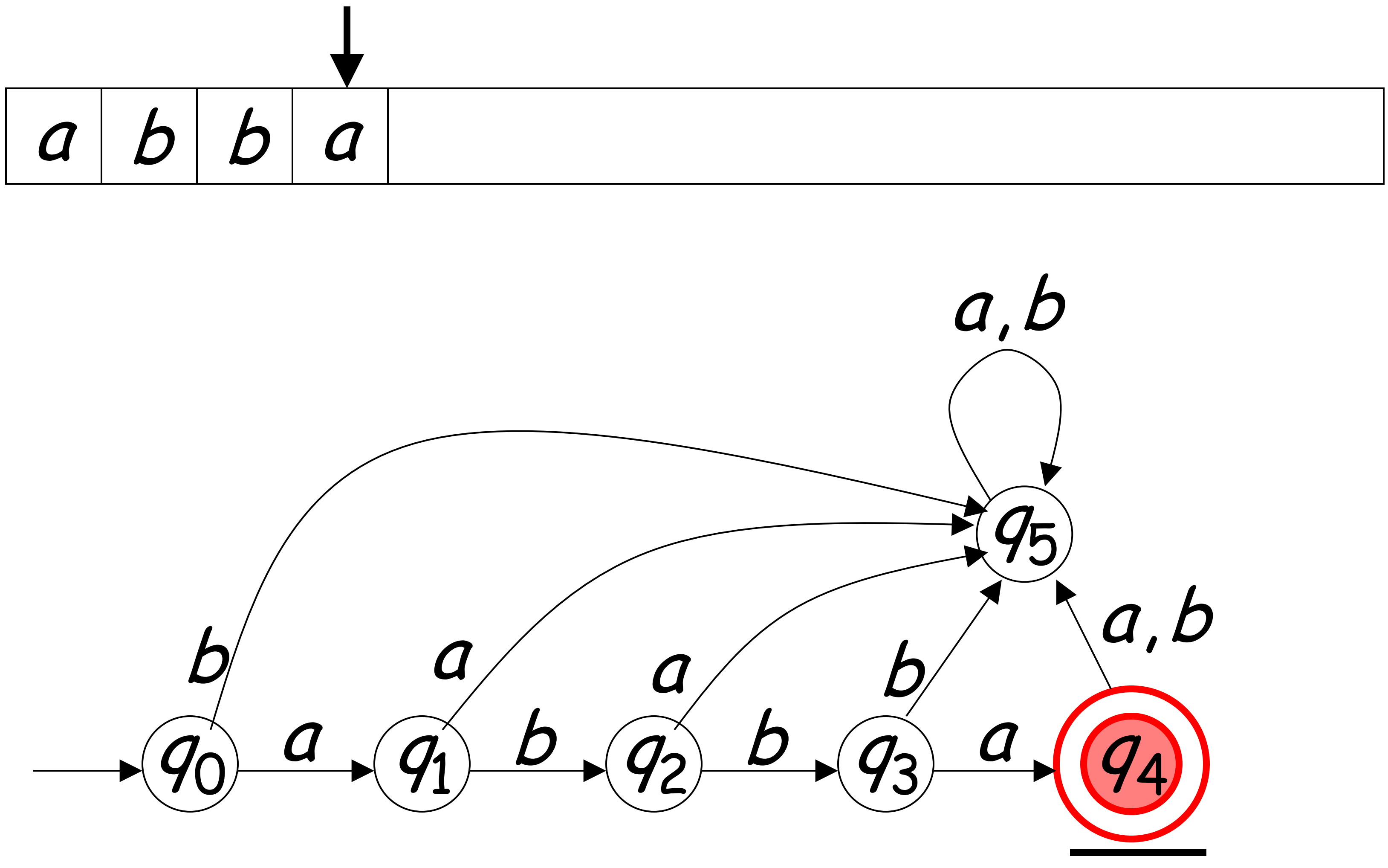





a
b
b
a
a,b
b
a
a
a,b
b
q_0
a
q_1
b
q_2
b
q_3
a
q_4
q_5


a
b
b
a
a, b
b
a
a
b
b
a, b
q0
a
q1
b
q2
b
q3
a
q4
q5

Input finished

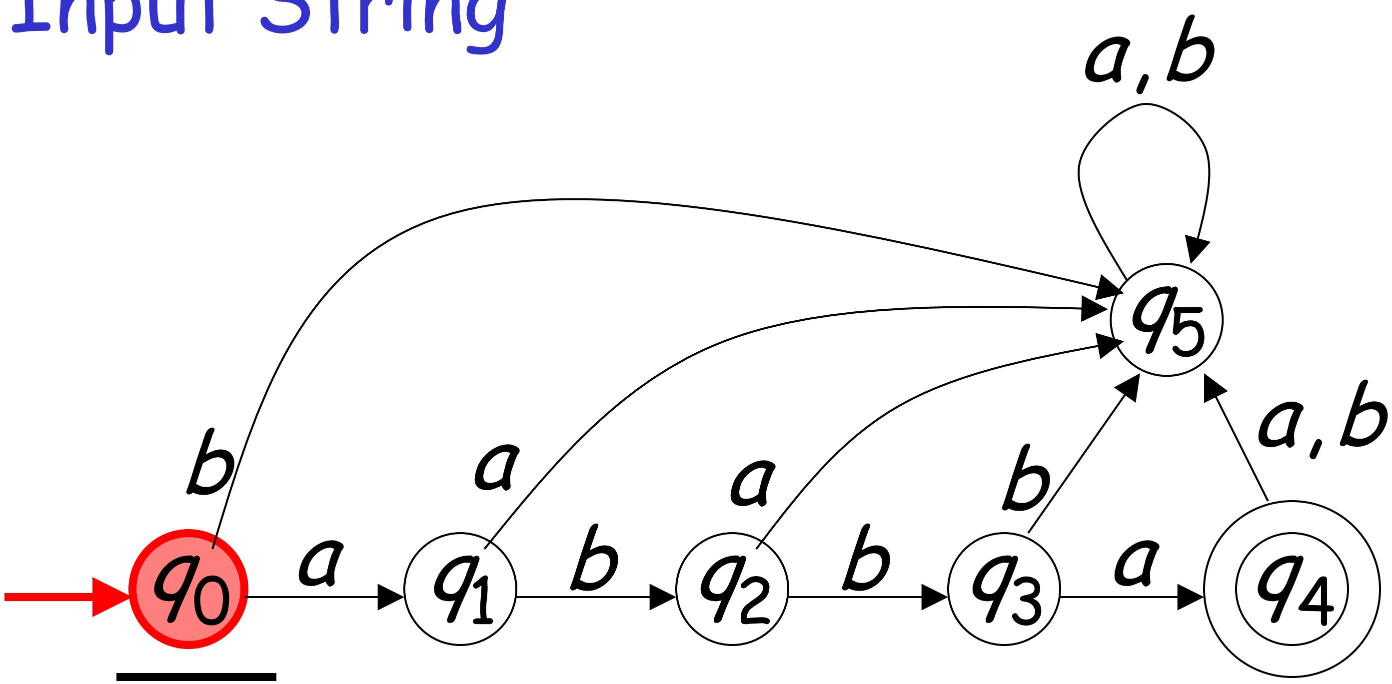


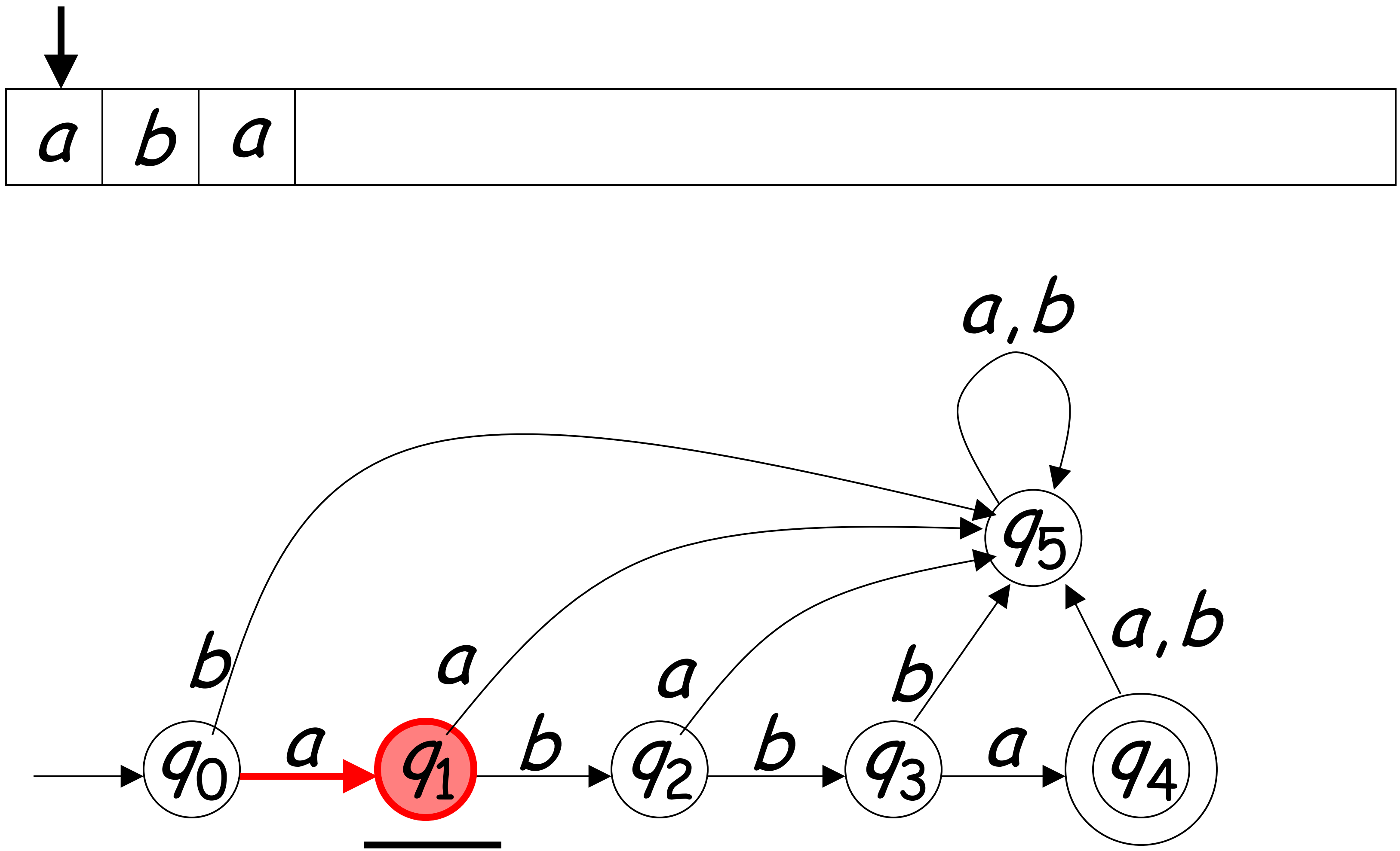
accept

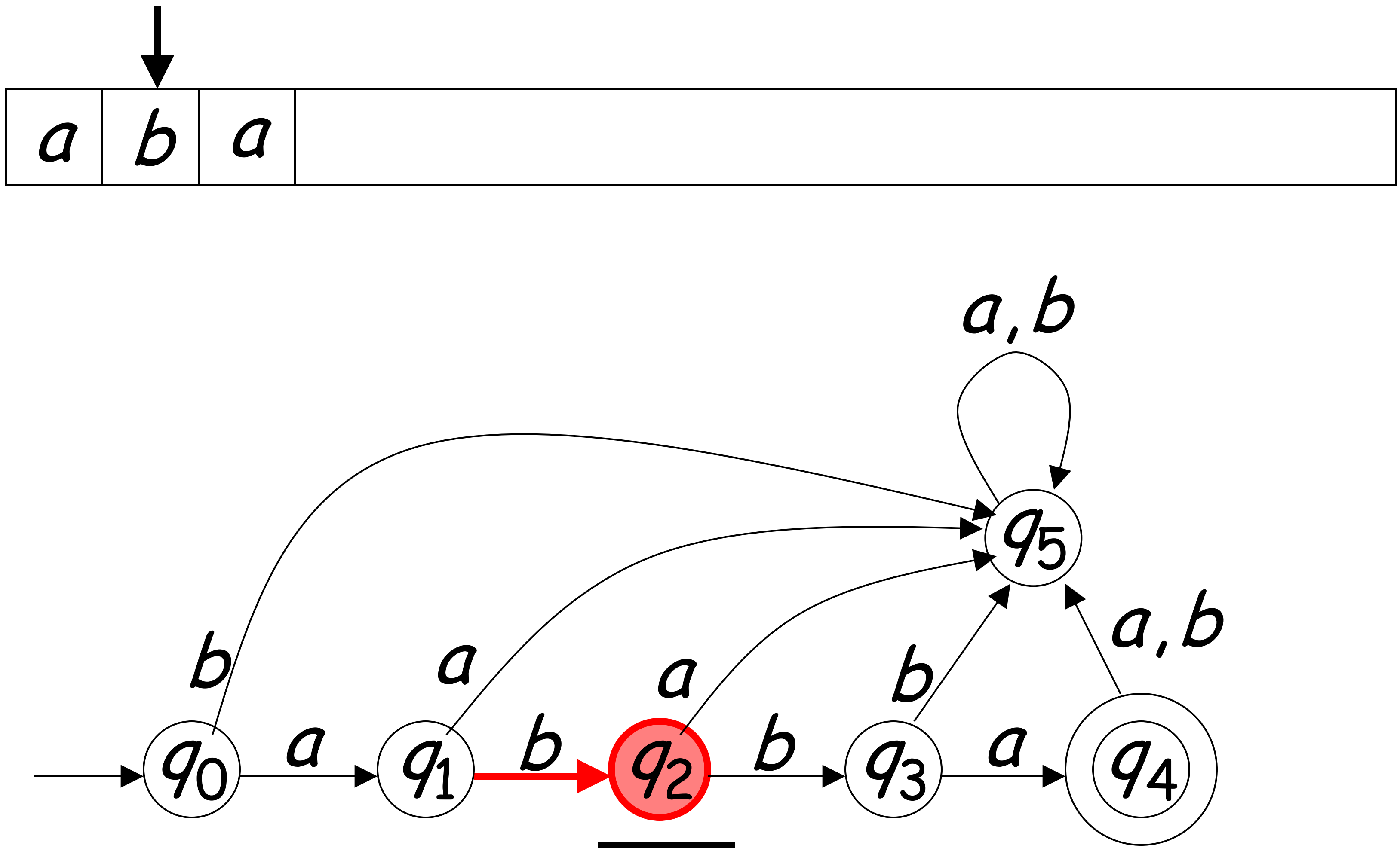
A Rejection Case

a	b	a	

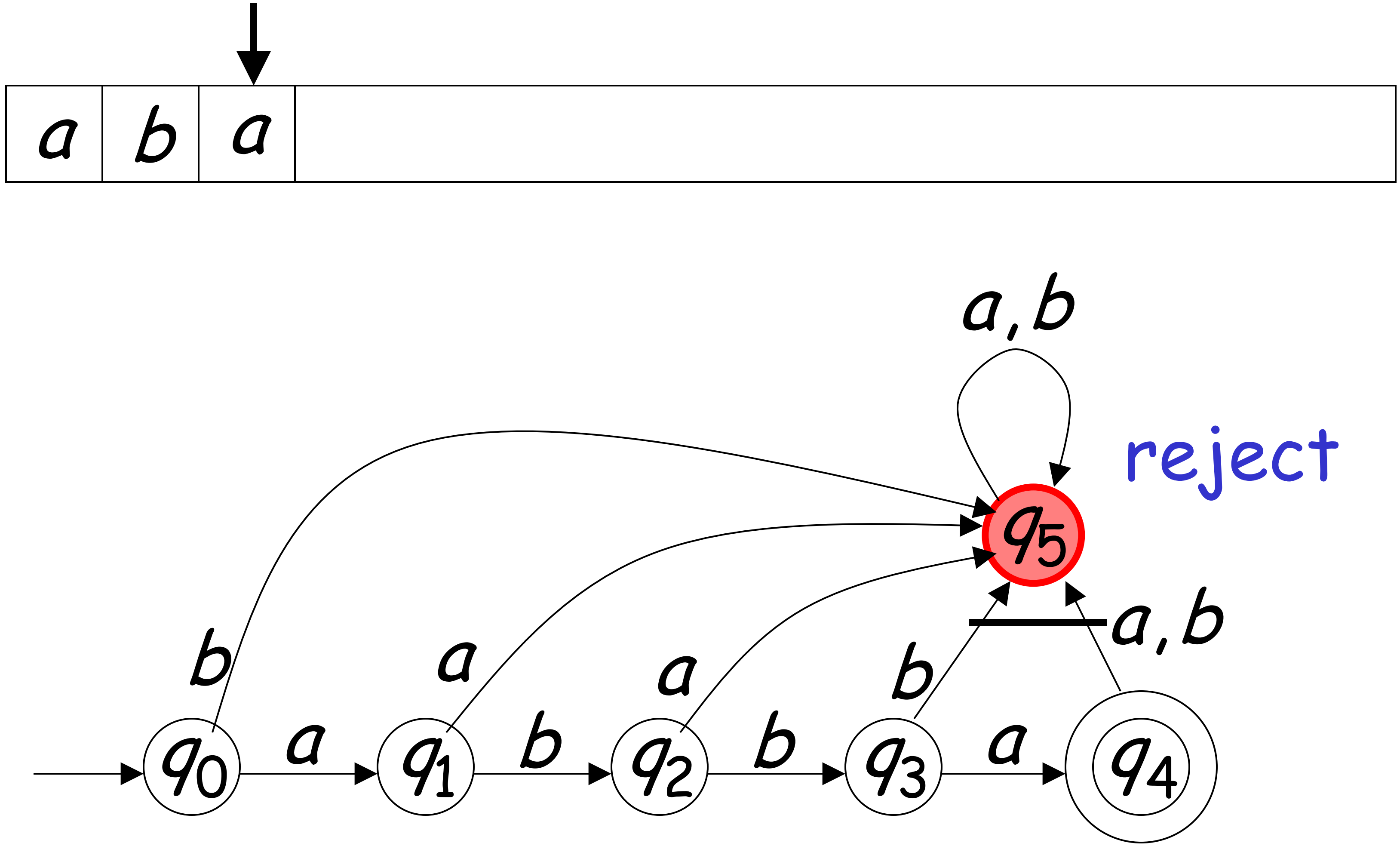
Input String




a
b
a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4


a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

Input finished

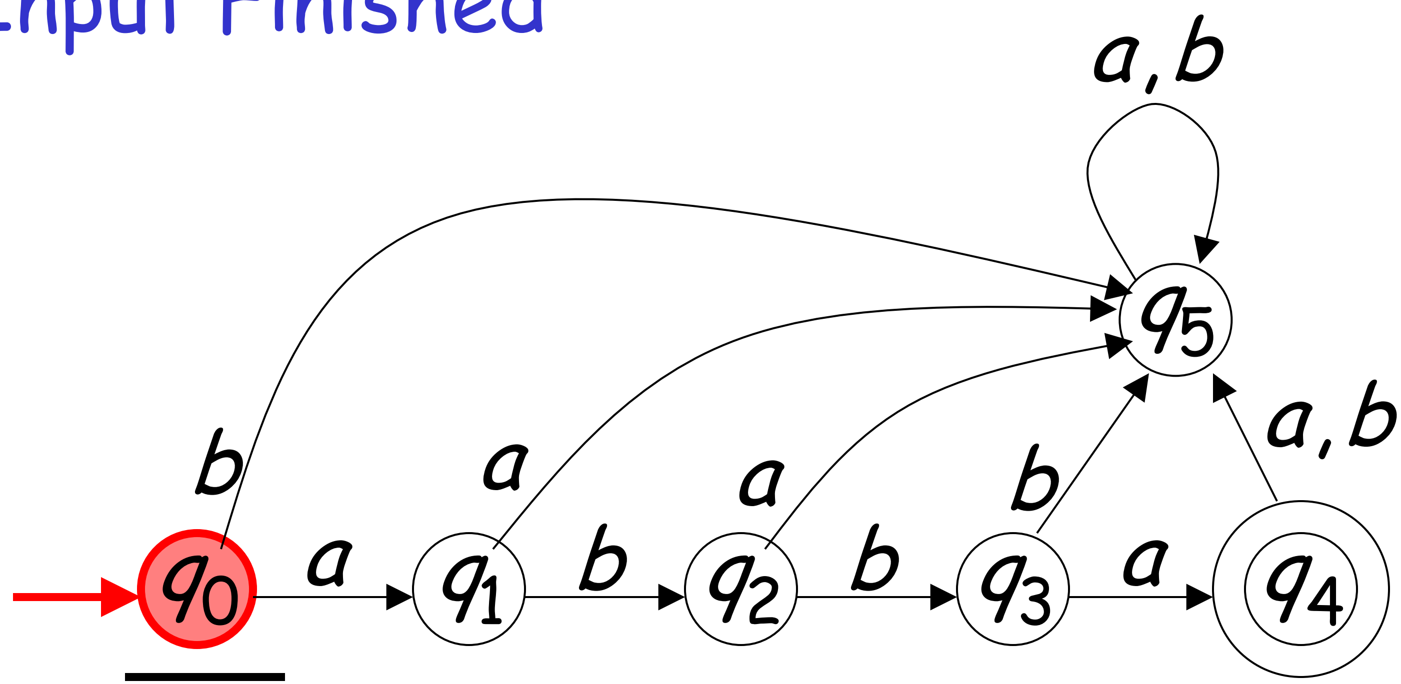


Another Rejection Case

Tape is empty

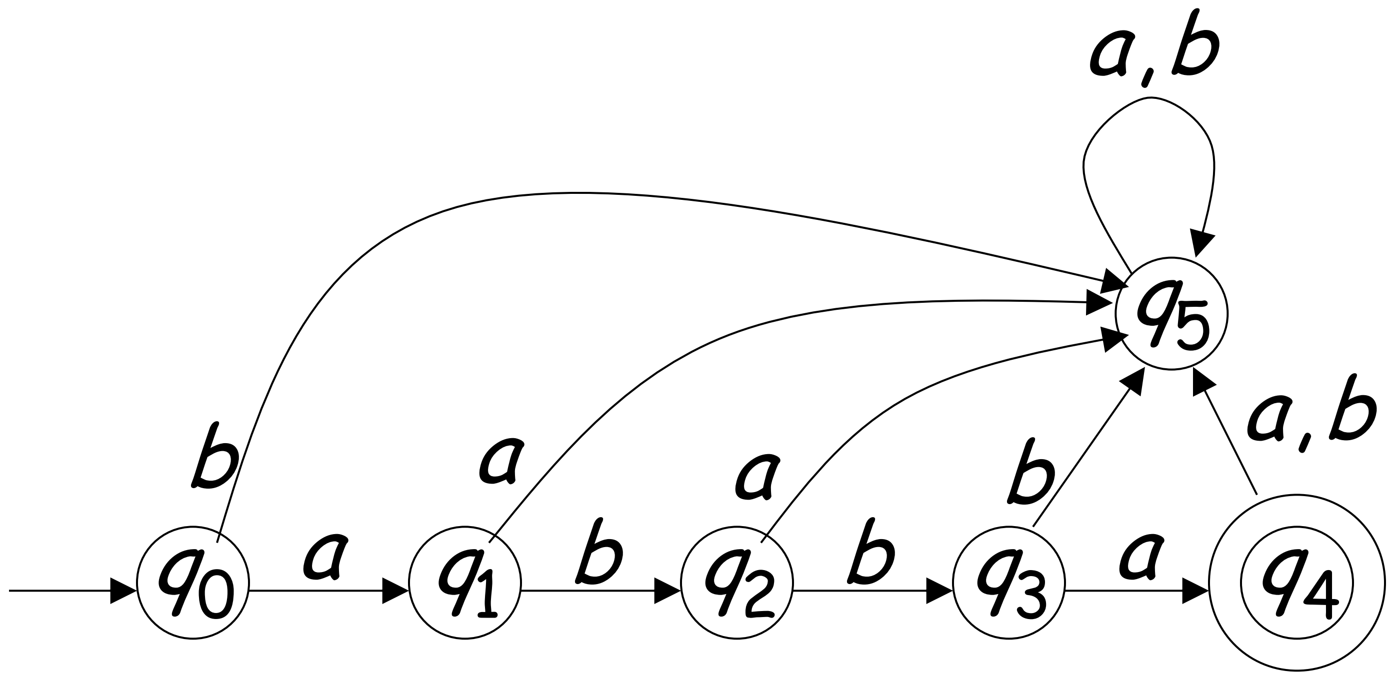
(λ)

Input Finished



reject

Language Accepted: $L = \{abba\}$



To accept a string:

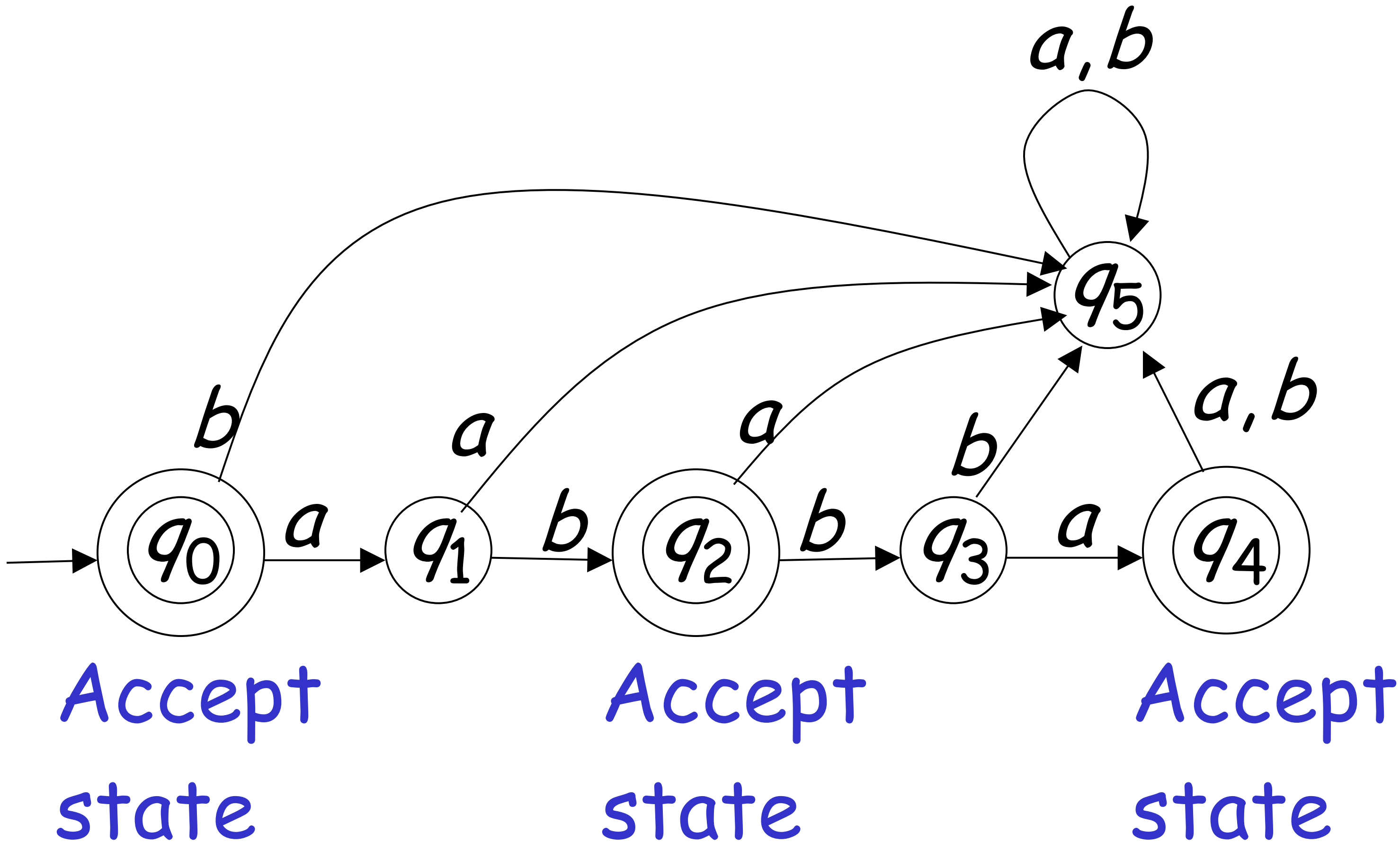
all the input string is scanned
and the last state is accepting

To reject a string:

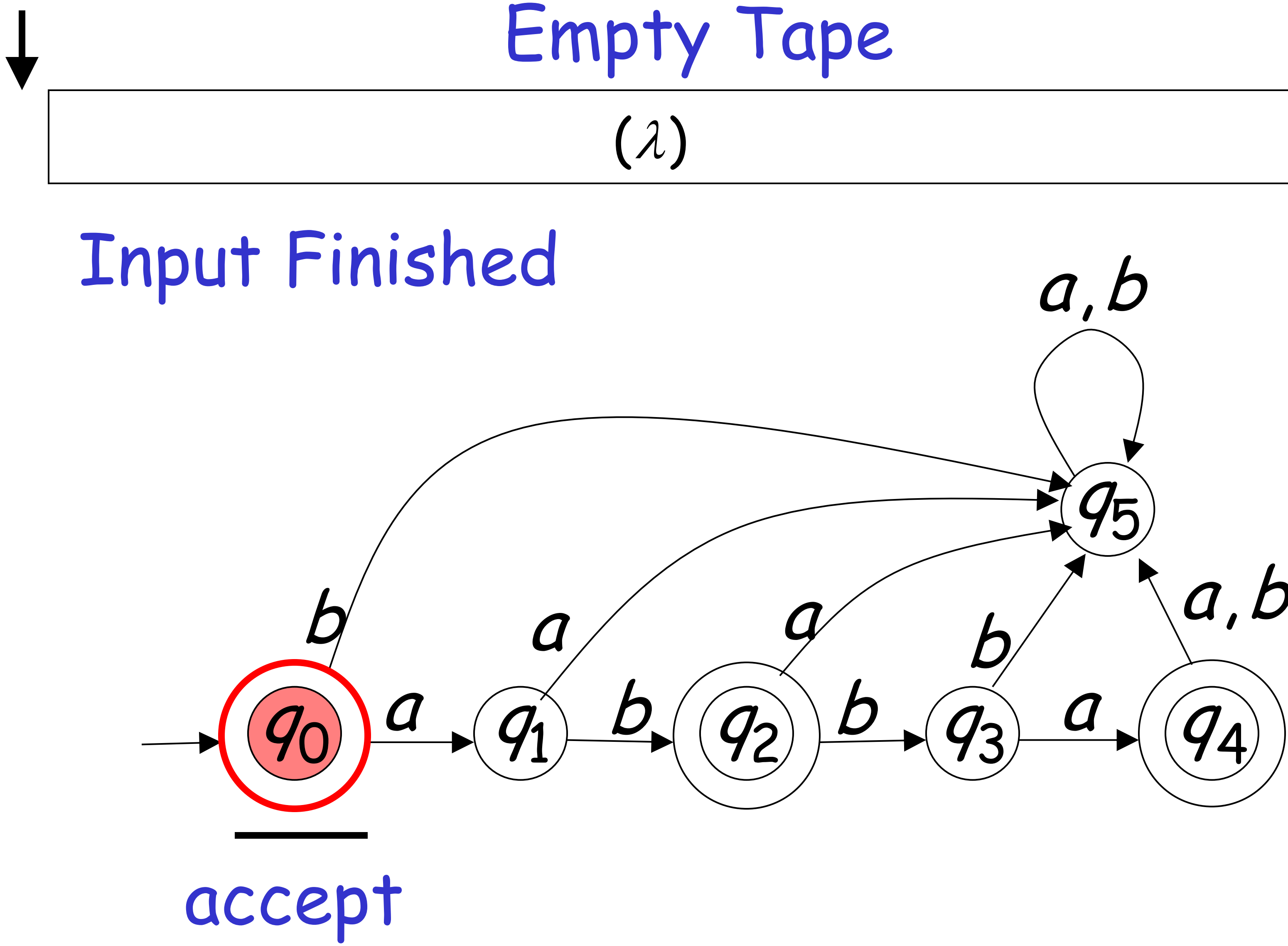
all the input string is scanned
and the last state is non-accepting

Another Example

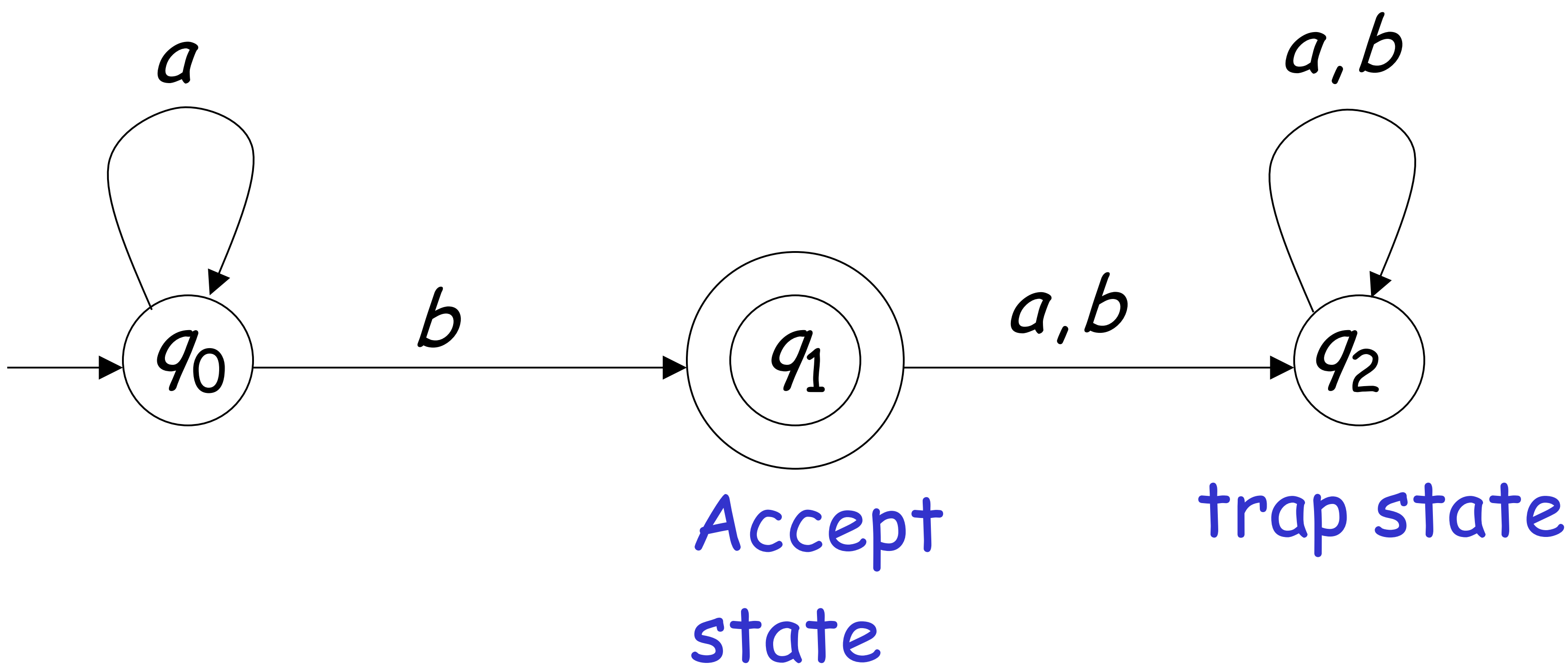
$$L = \{\lambda, ab, abba\}$$

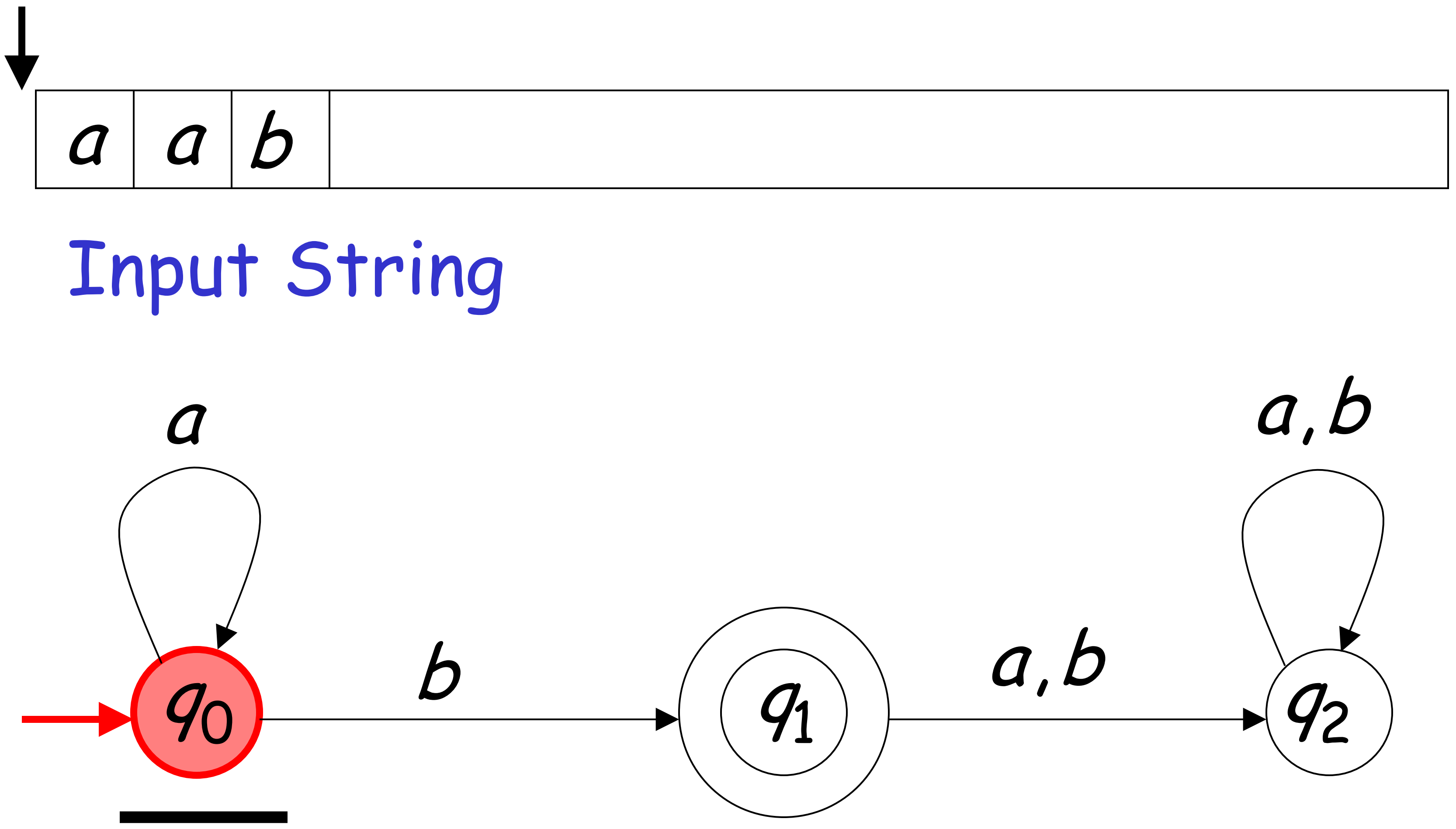


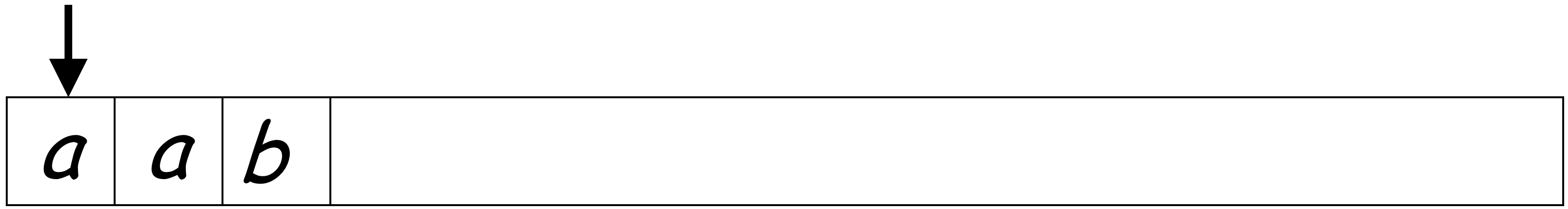
Empty Tape

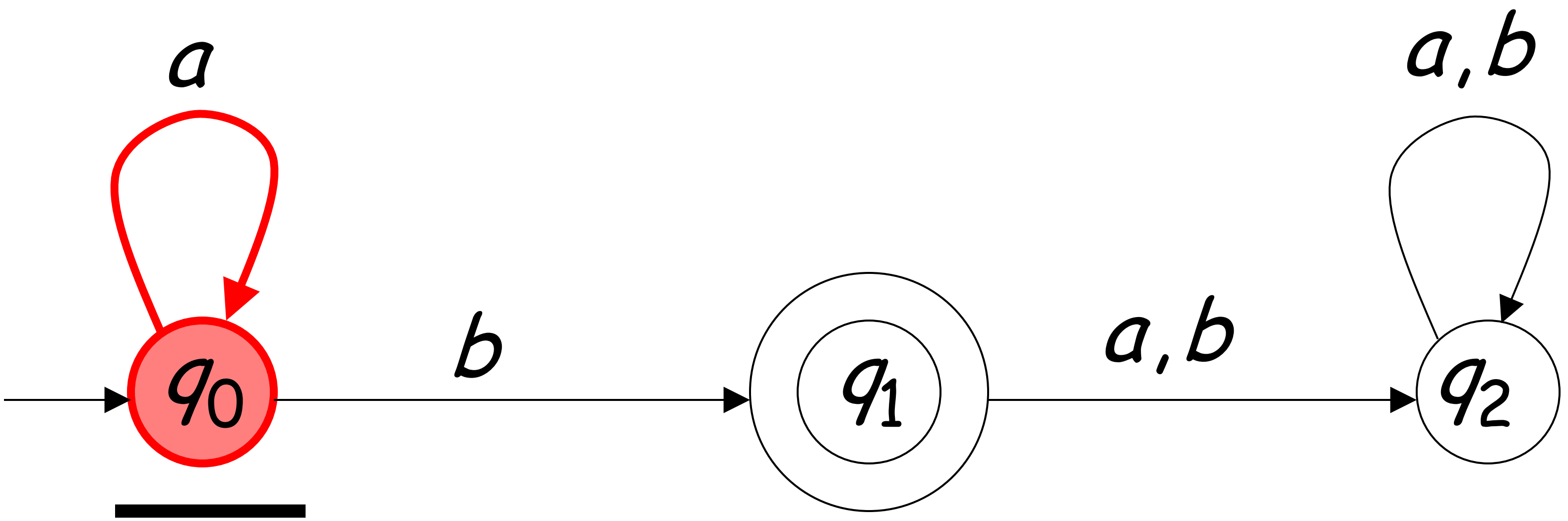


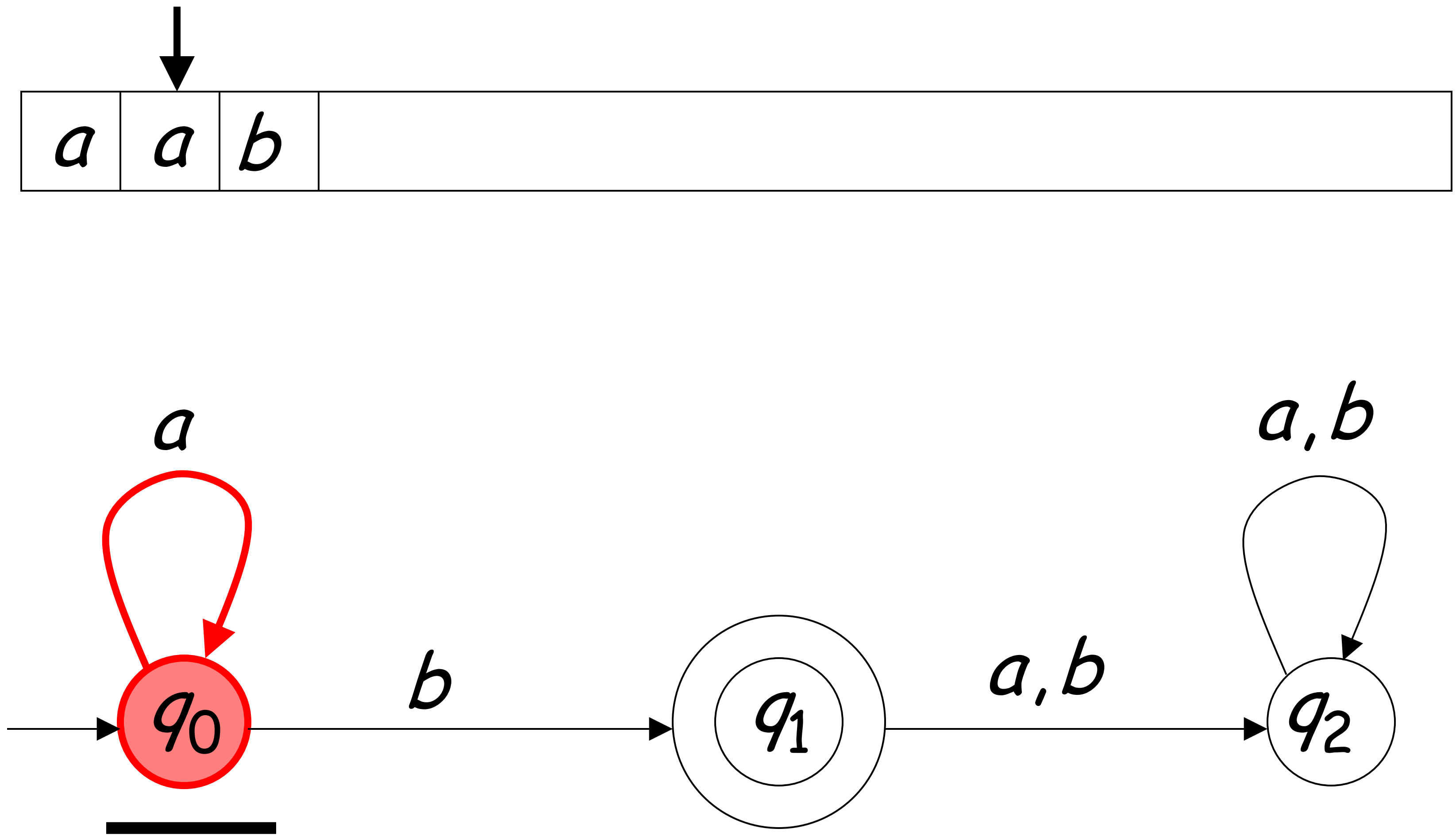
Another Example



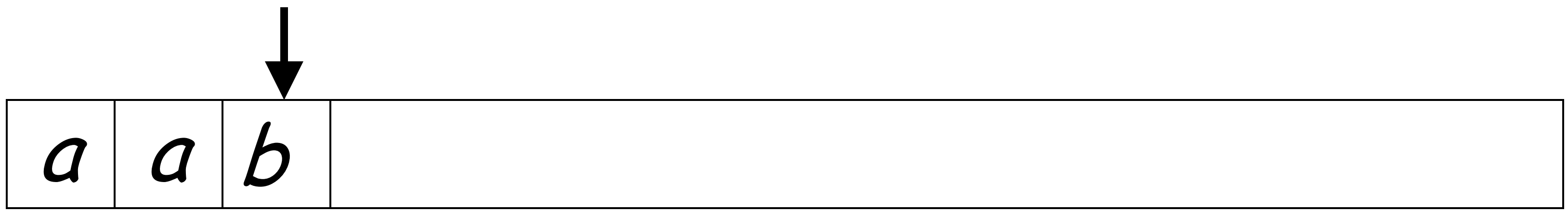

a
a
b
Input String
a
a,b
b
a,b
q0
q1
q2

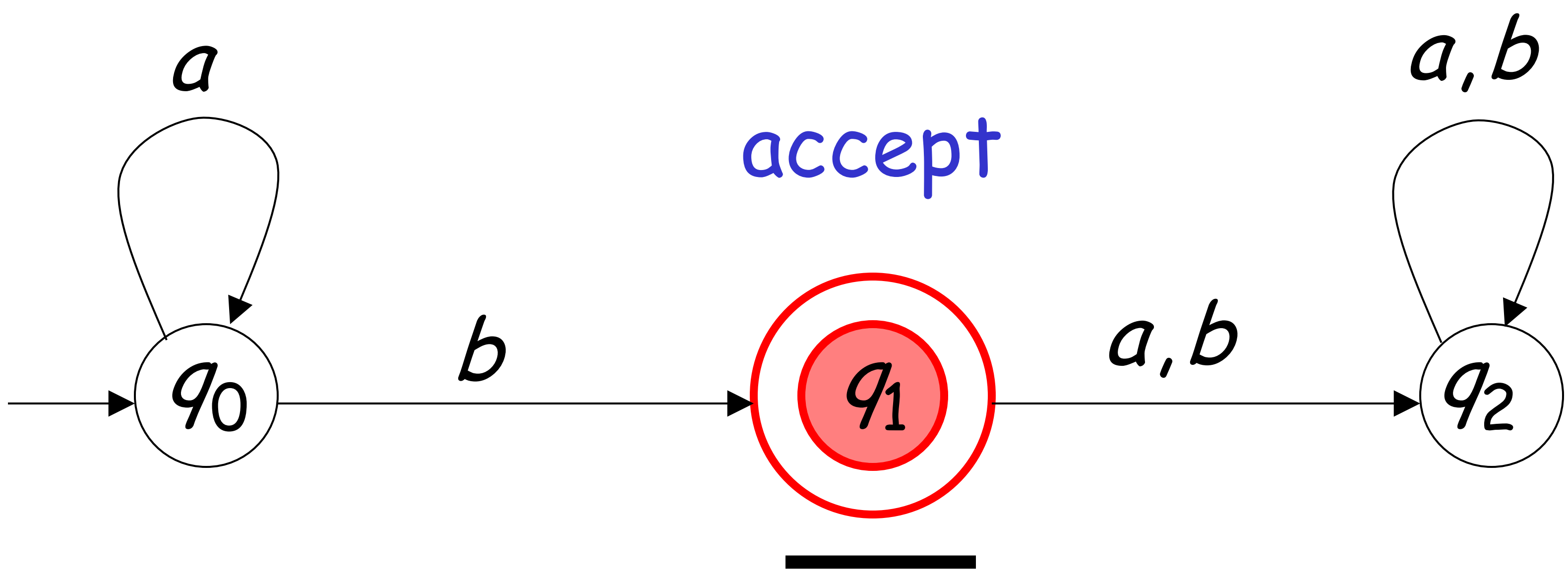

a
a
b


a
b
a,b
a,b
q_0
q_1
q_2

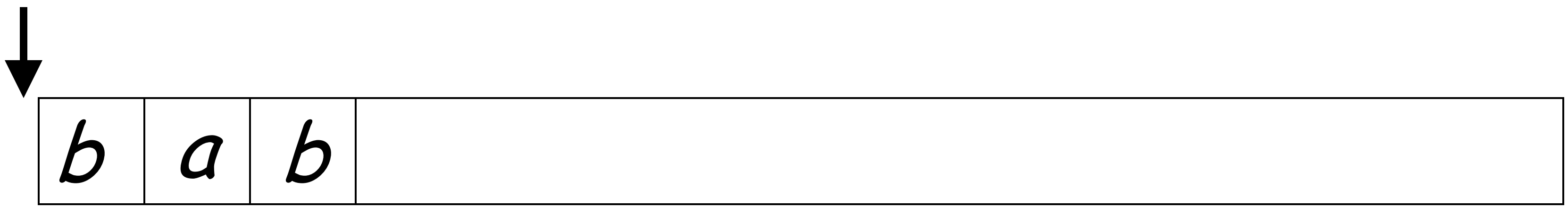

a
a
b
a
b
a, b
a, b
q_0
q_1
q_2

Input finished

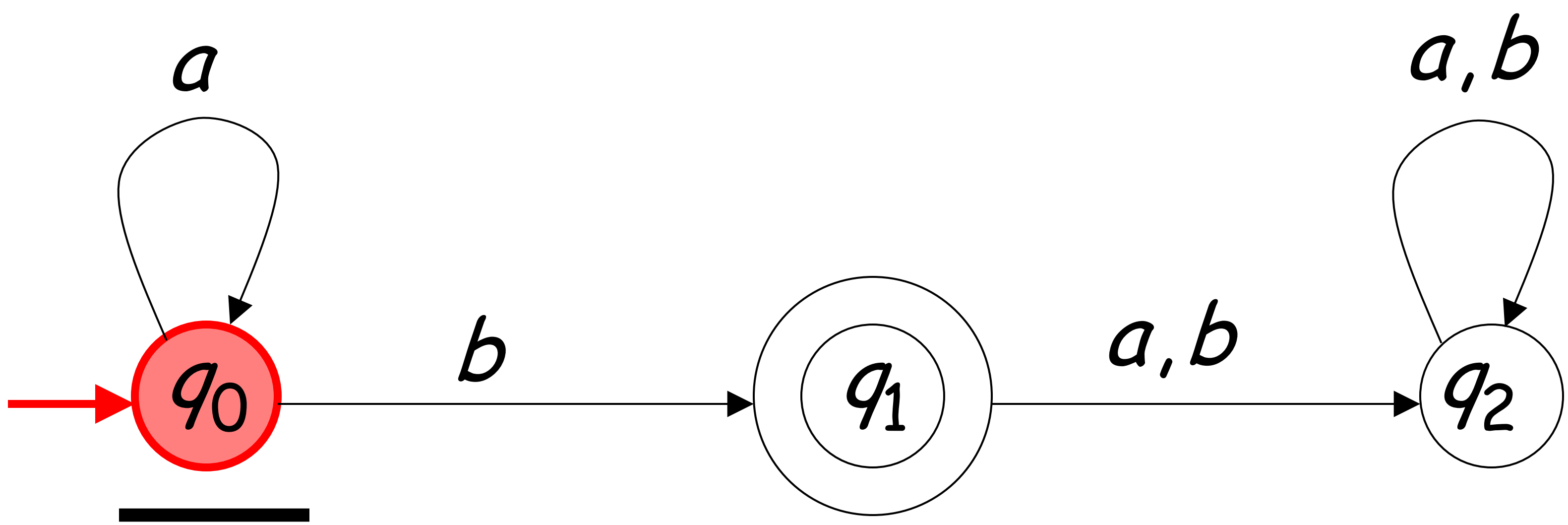


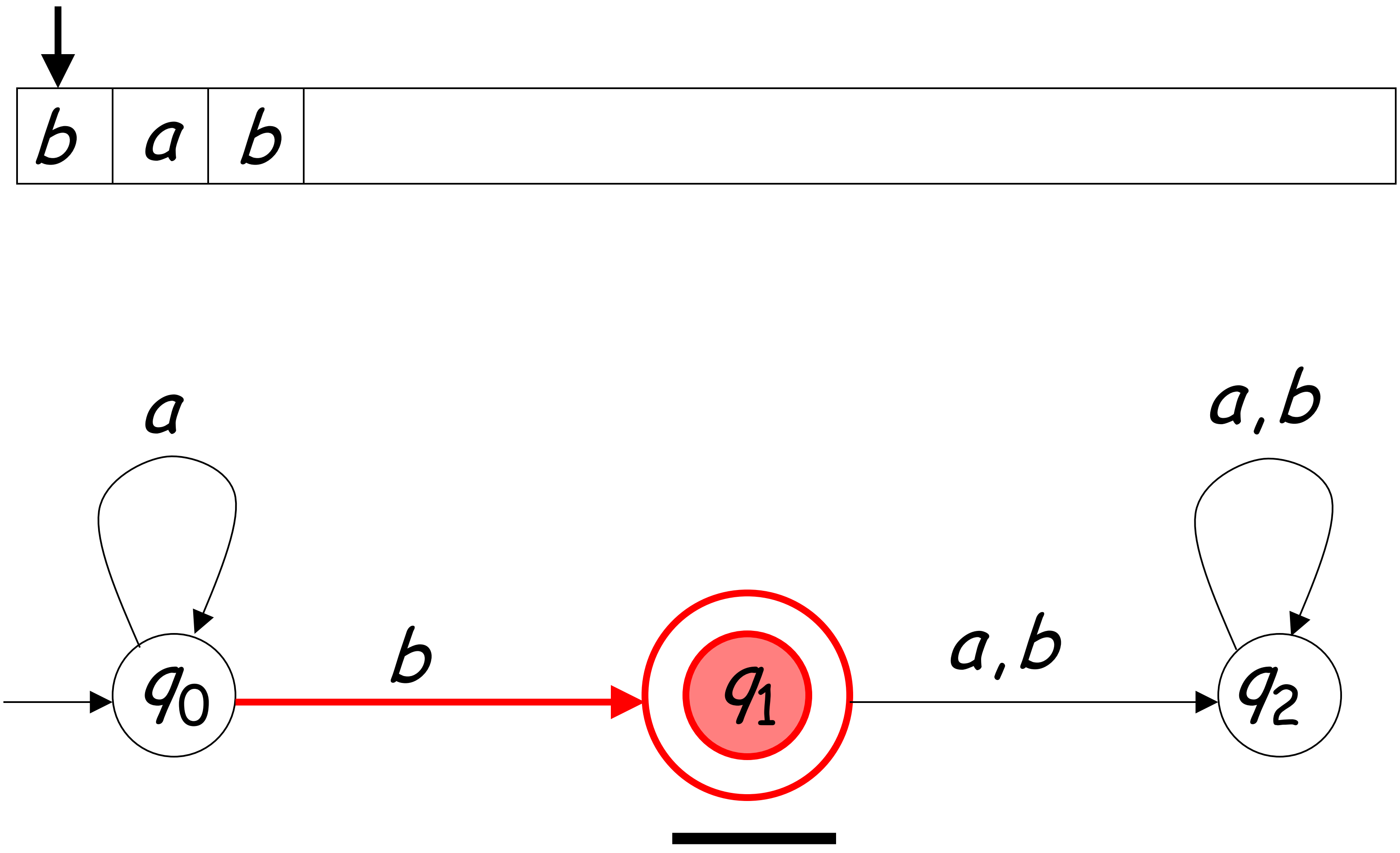


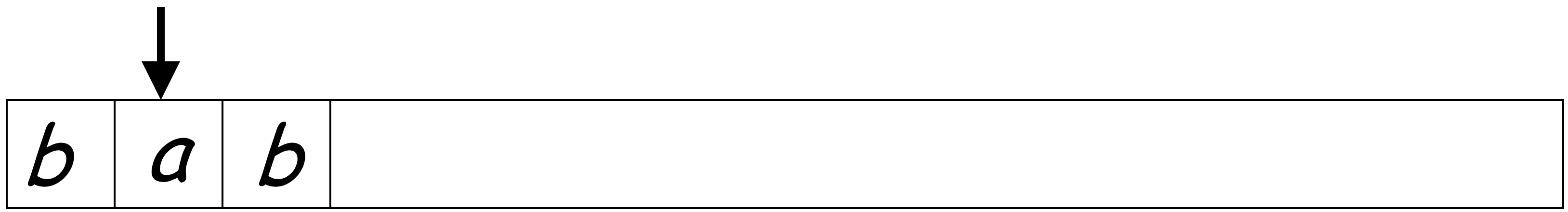
A rejection case

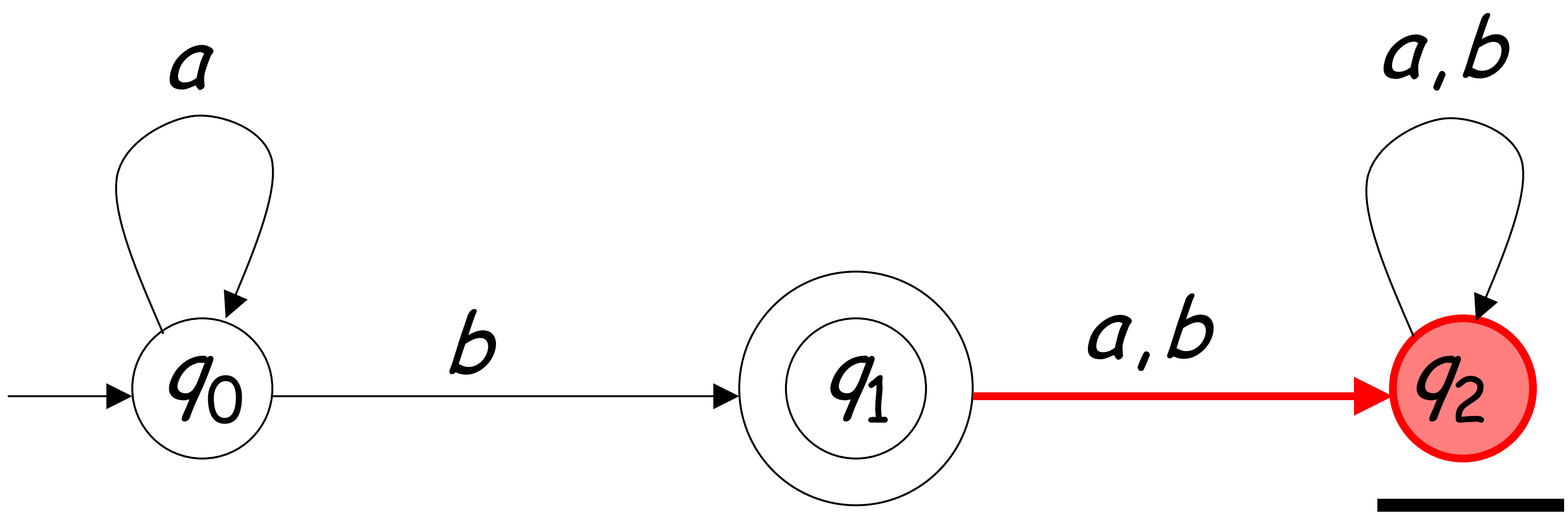


Input String

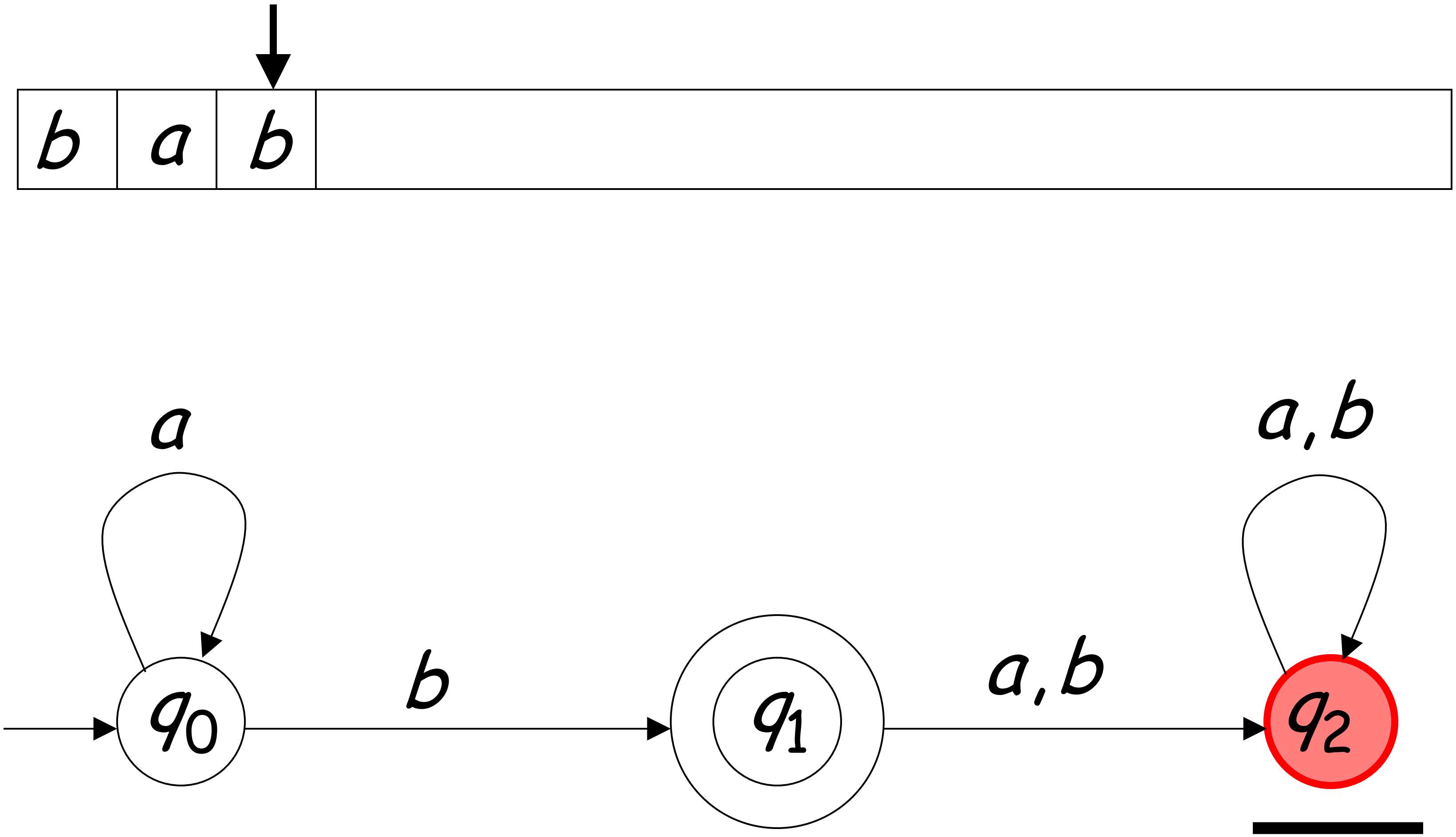



b
a
b
a
a,b
b
a,b
q_0
q_1
q_2


b
a
b

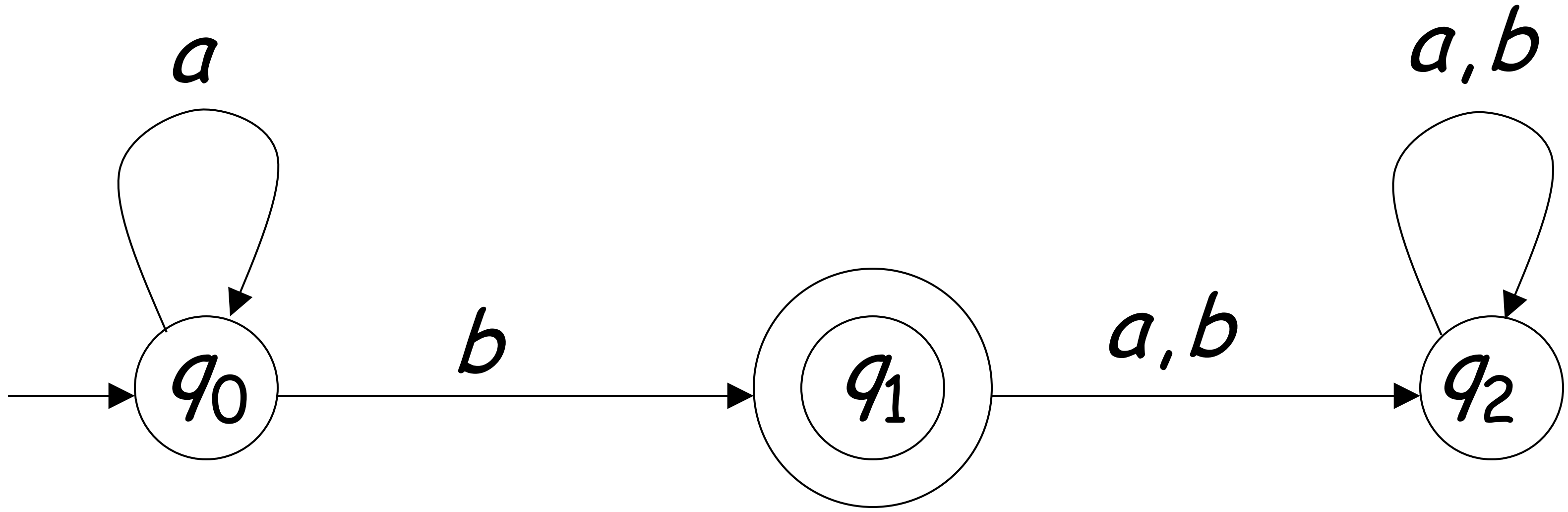

a
a,b
b
a,b
q_0
q_1
q_2

Input finished



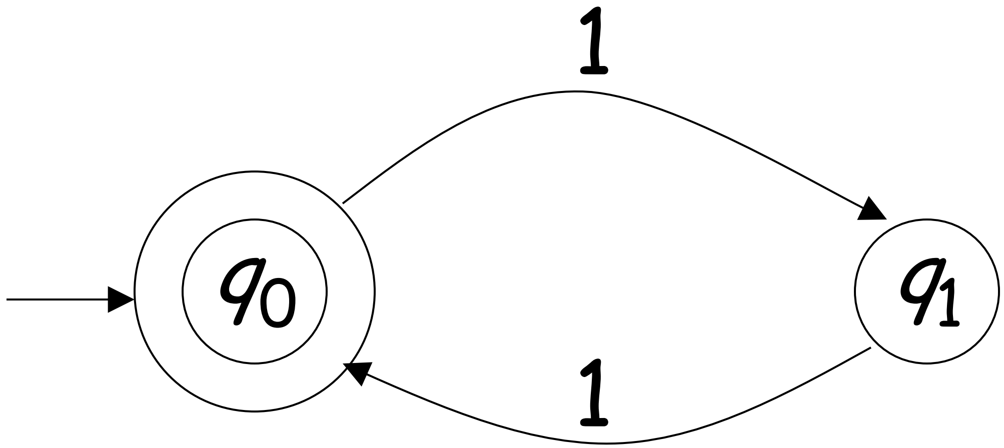
reject

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



Another Example

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



Language Accepted:

$$EVEN = \{x : x \in \Sigma^* \text{ and } x \text{ is even}\}$$
$$= \{\lambda, 11, 1111, 111111, \ldots\}$$

Formal Definition

Deterministic Finite Automaton (DFA)

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

Q : set of states

Σ : input alphabet $\quad \lambda \notin \Sigma$

δ : transition function

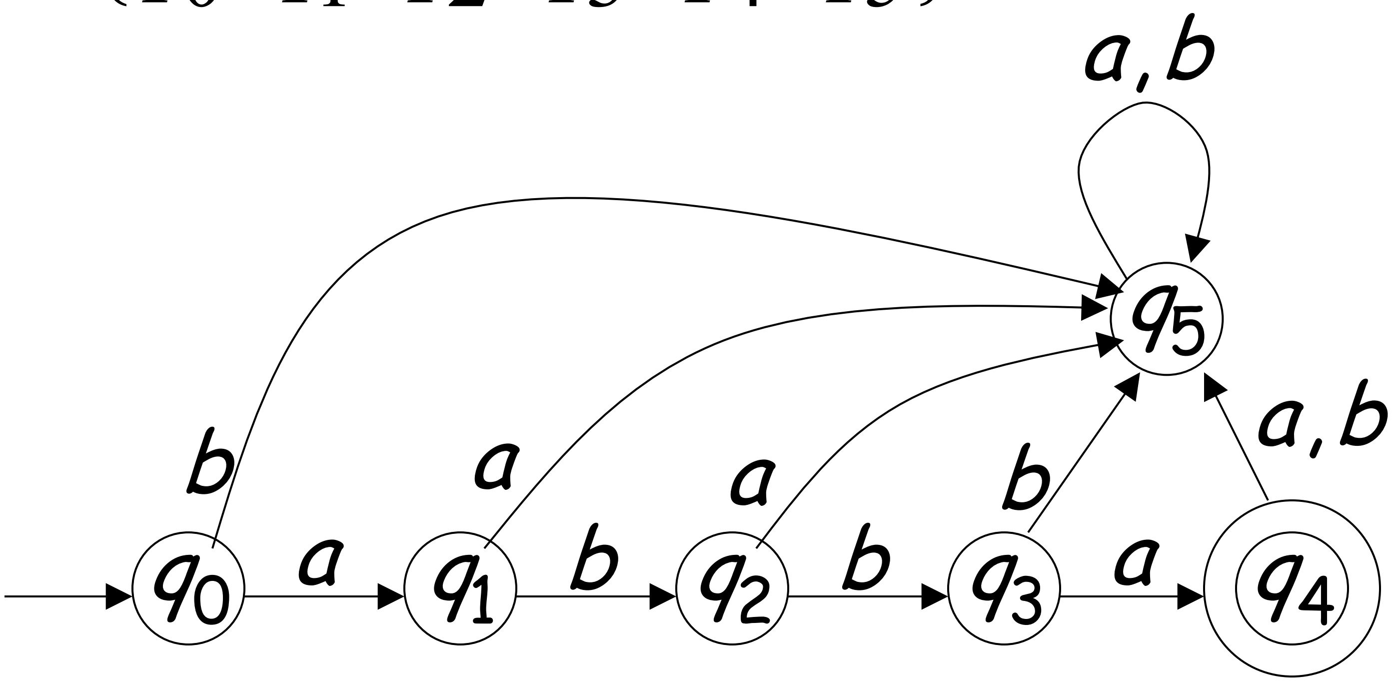
q_0 : initial state

F : set of accepting states

Set of States Q

Example

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

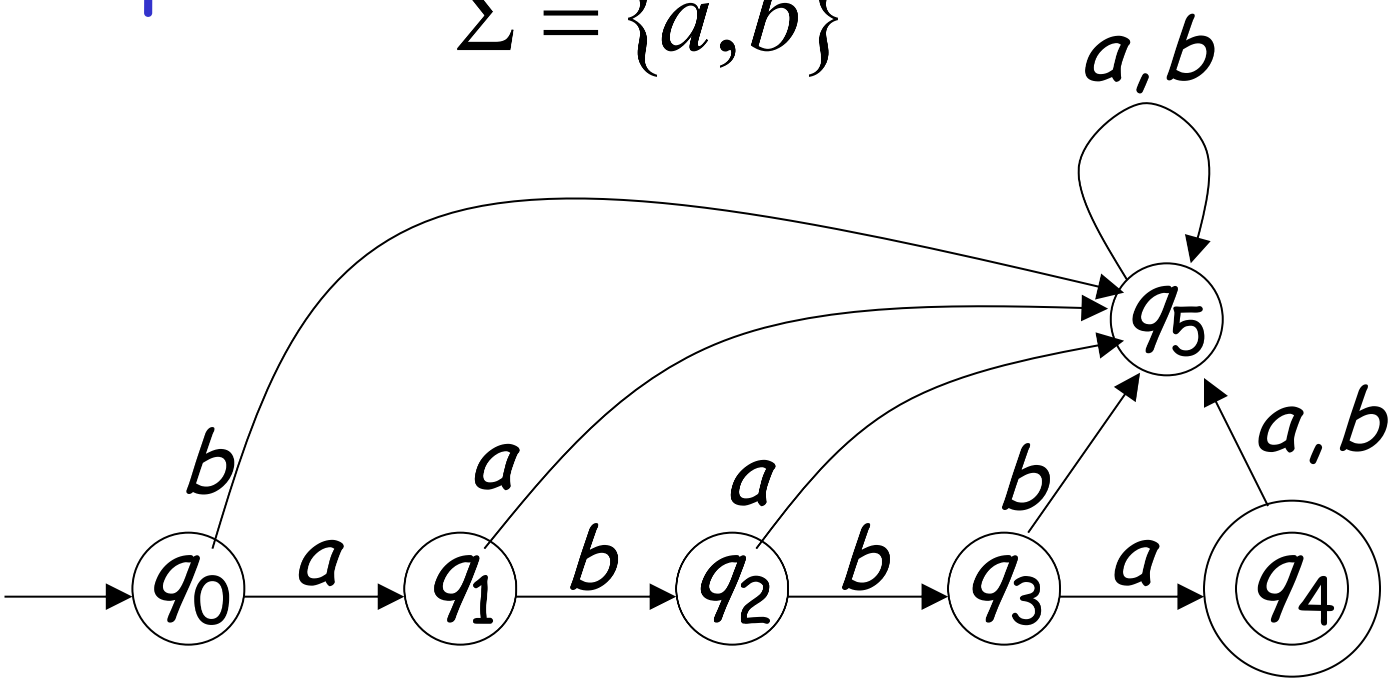


Input Alphabet Σ

$\lambda \notin \Sigma$:the input alphabet never contains λ

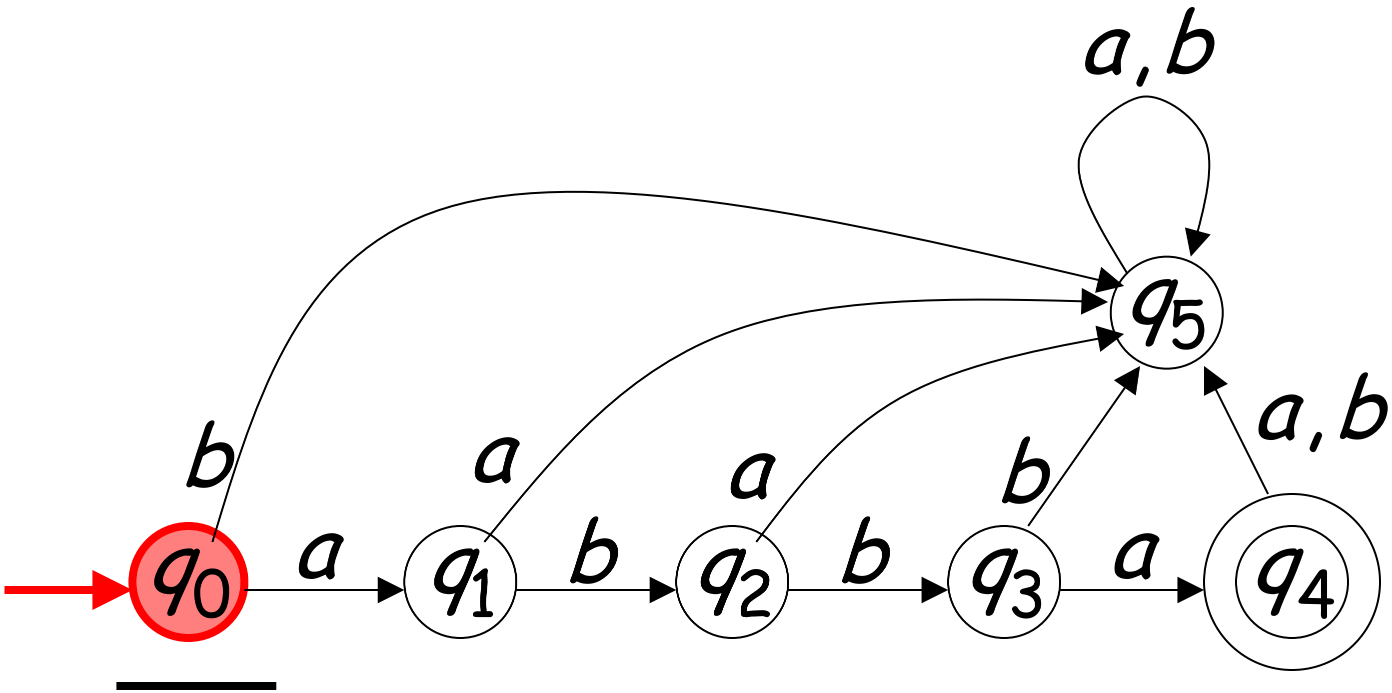
Example

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



Initial State q_0

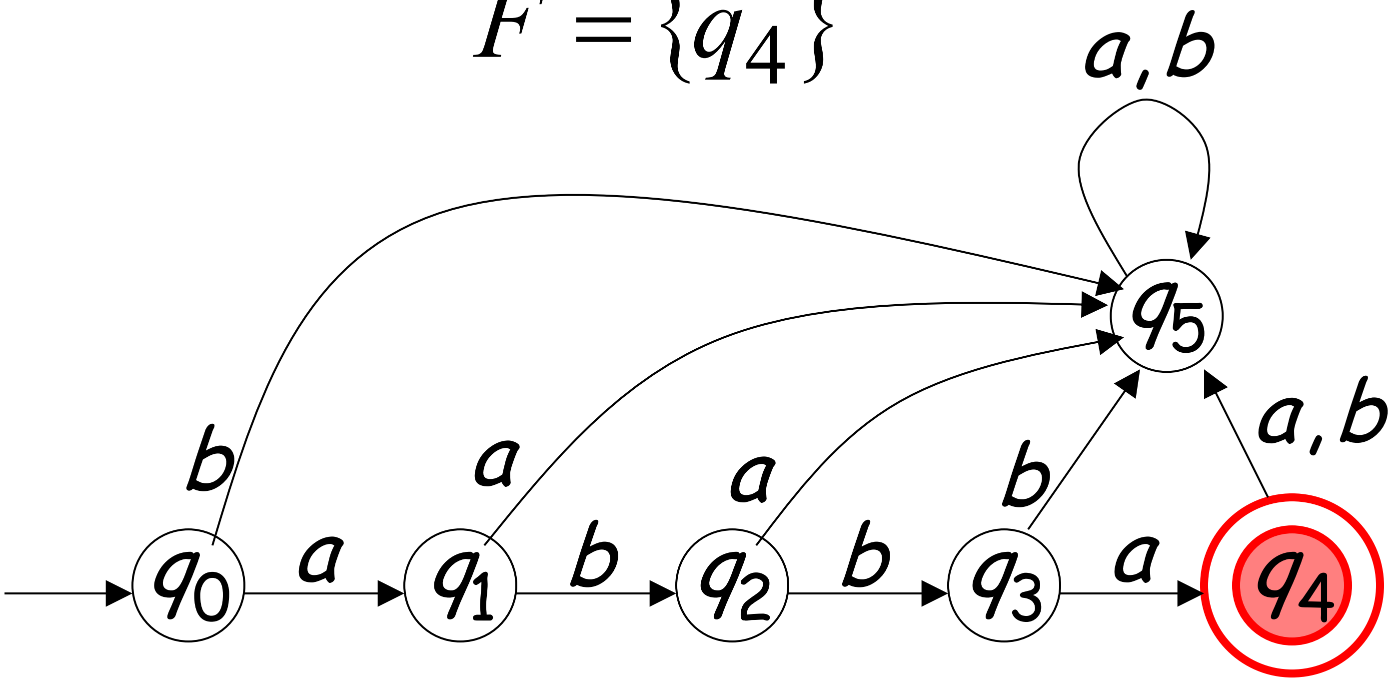
Example



Set of Accepting States $F \subseteq Q$

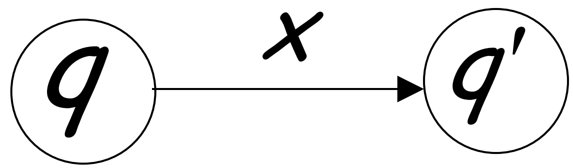
Example

$$F = \{q_4\}$$



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

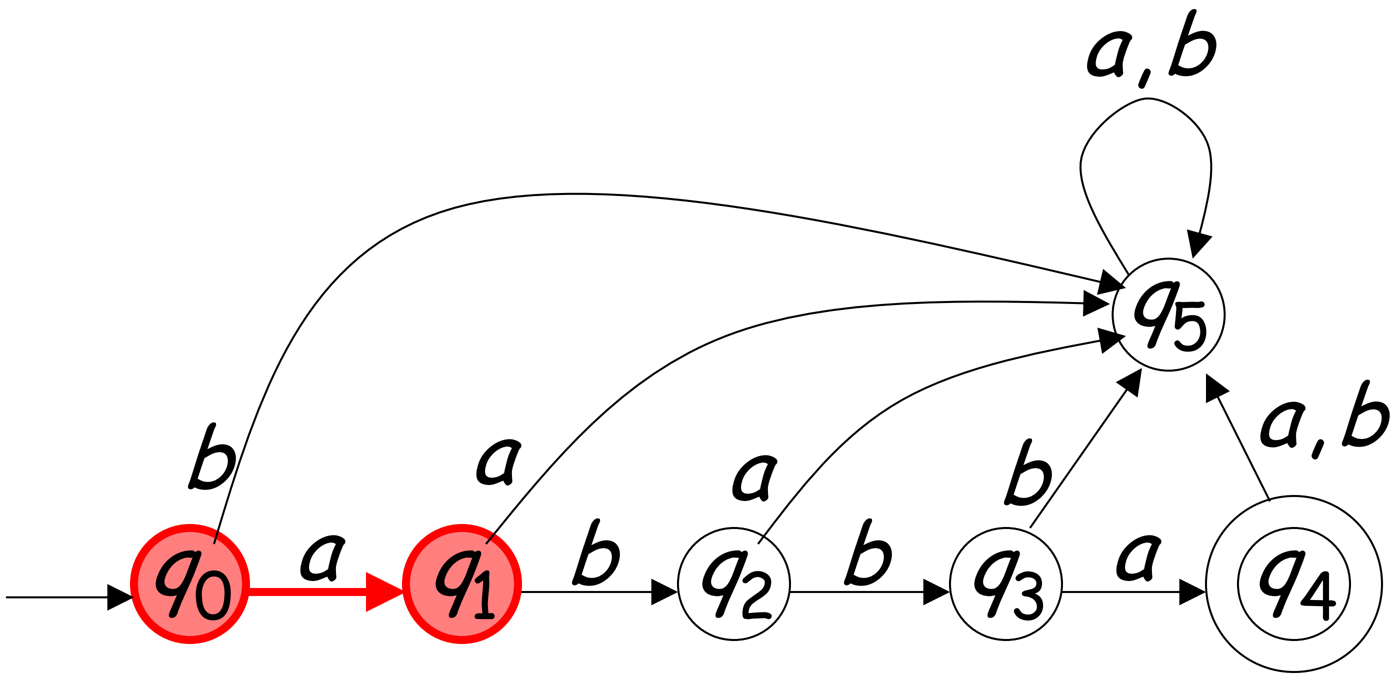
$$\delta(q, x) = q'$$



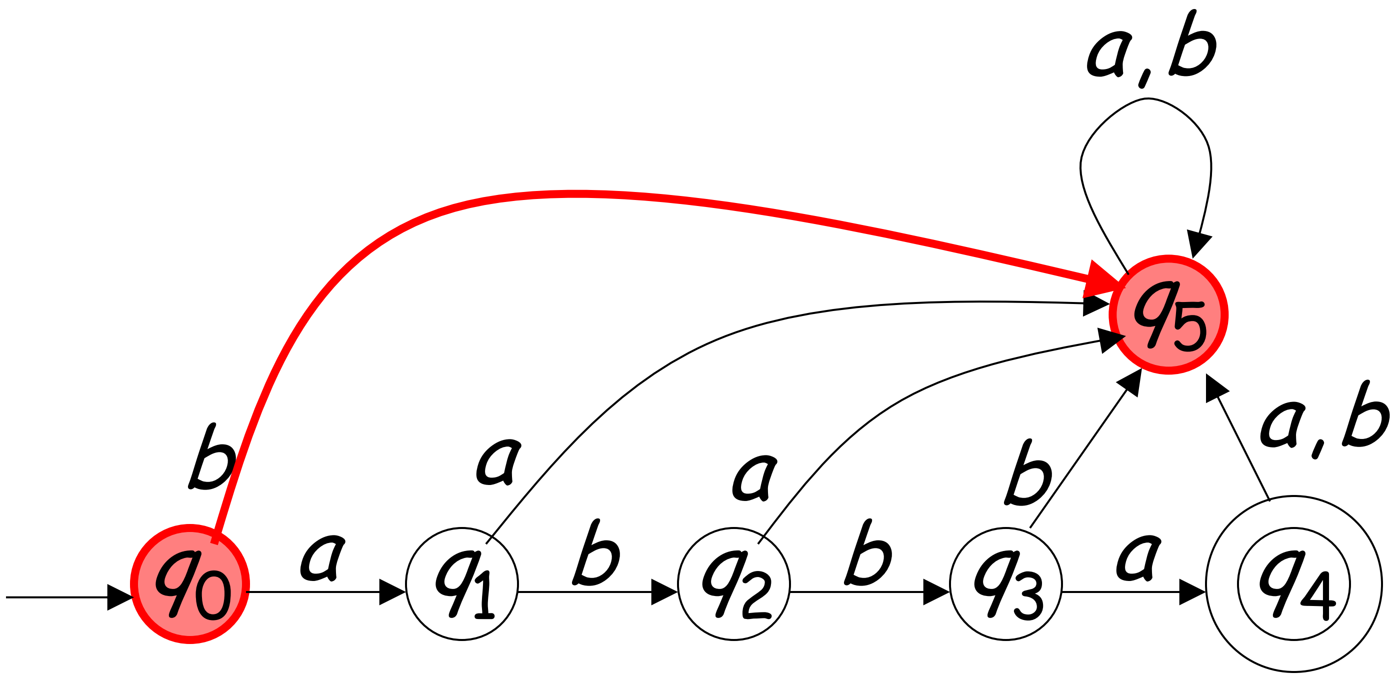
Describes the result of a transition from state q with symbol x

Example:

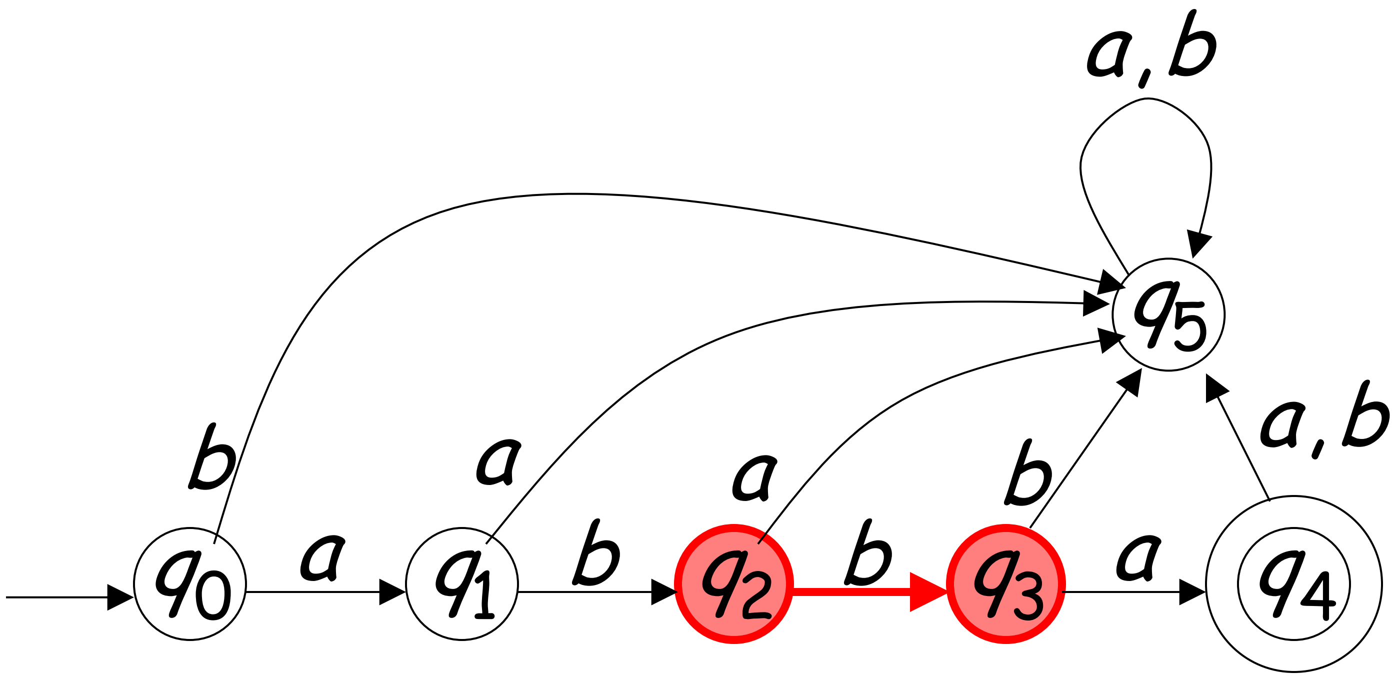
$$\delta(q_0, a) = q_1$$



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



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

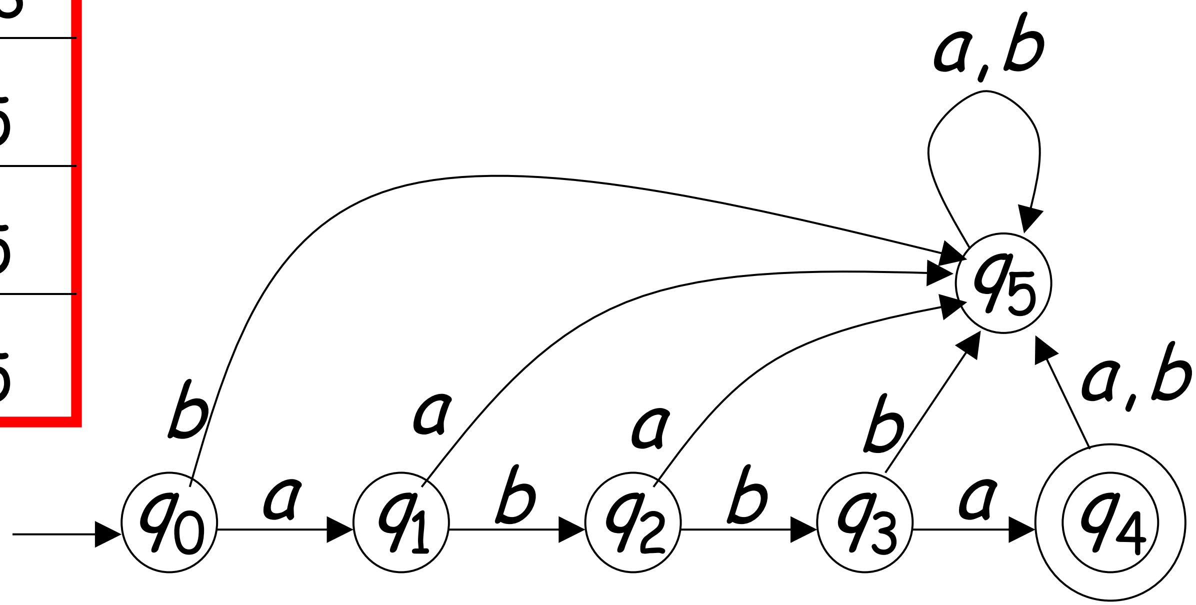


Transition Table for δ

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



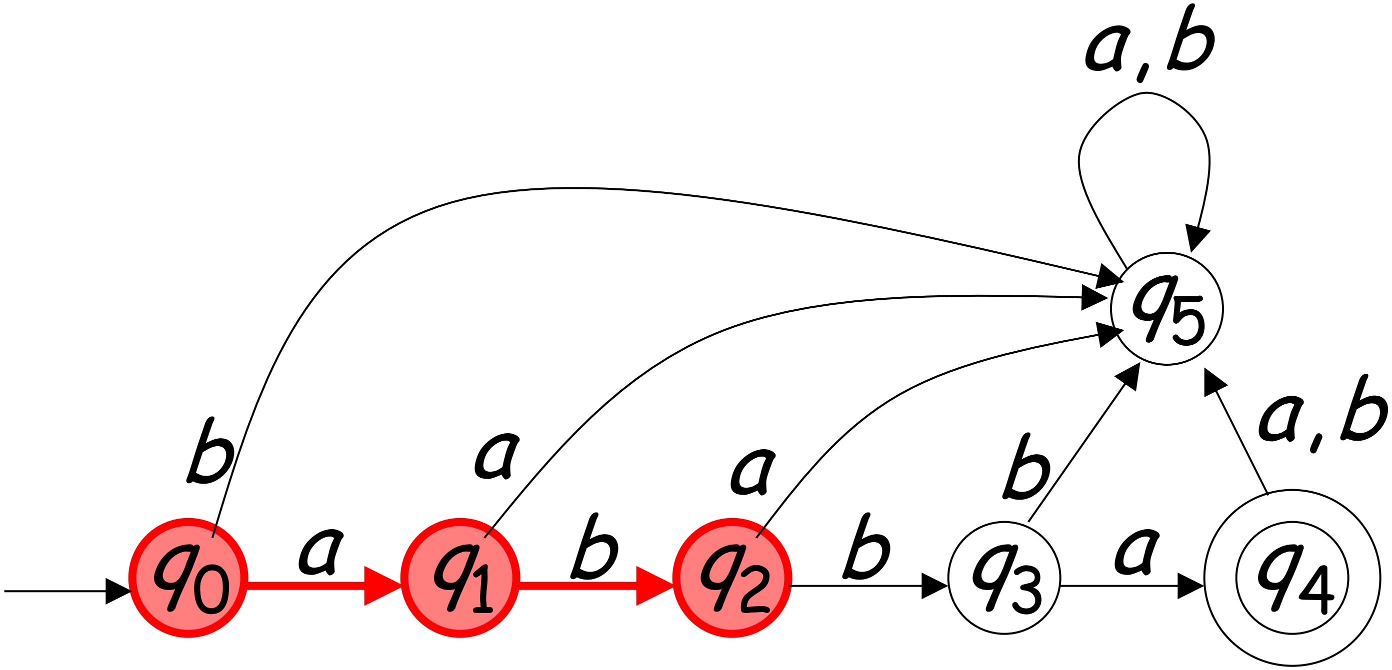
Extended Transition Function

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

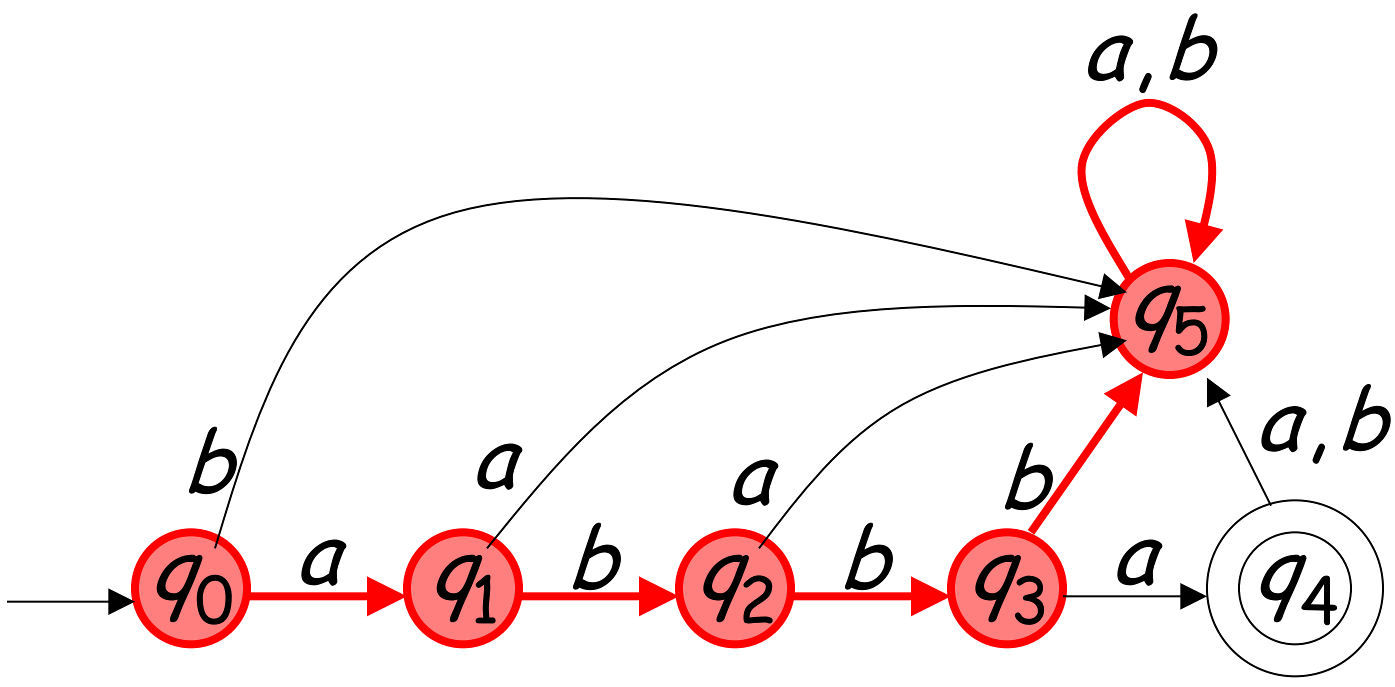
$$\delta^*(q, w) = q'$$

Describes the resulting state
after scanning string w from state q

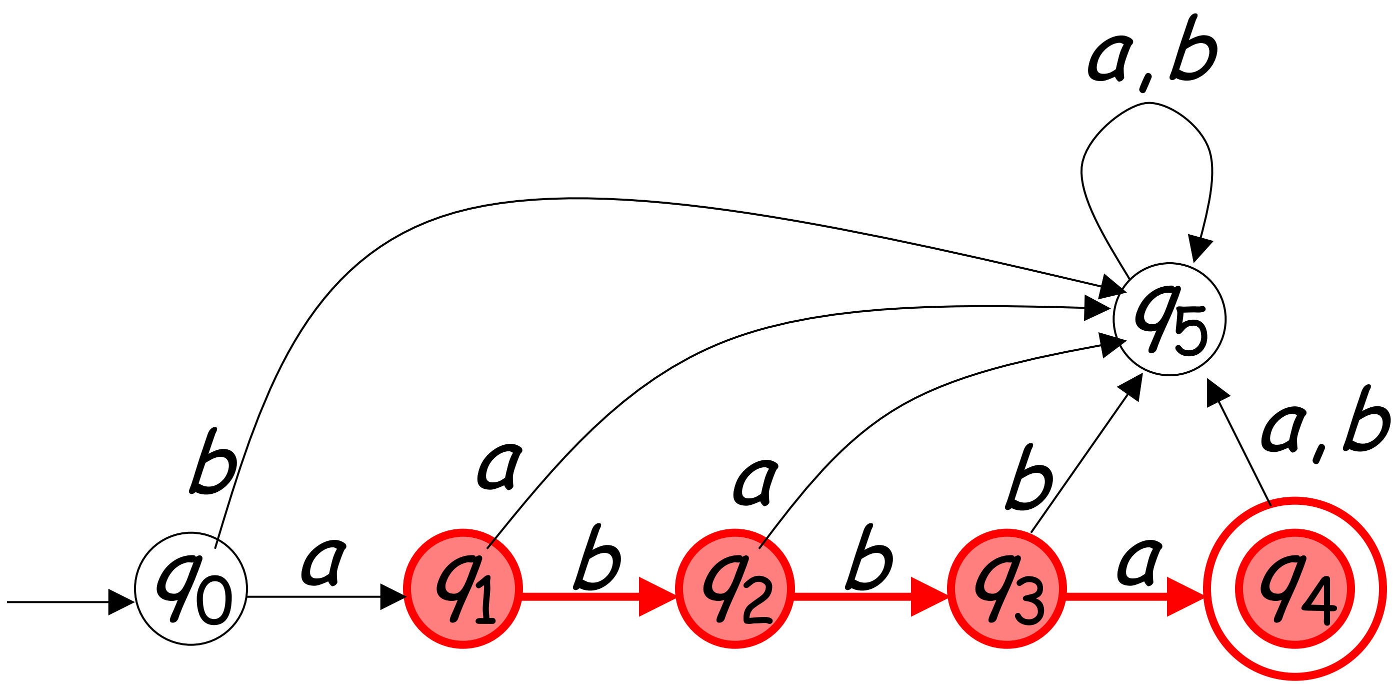
Example: $\delta^*(q_0, ab) = q_2$



$$\delta^*(q_0, abbbaa) = q_5$$



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



Special case:

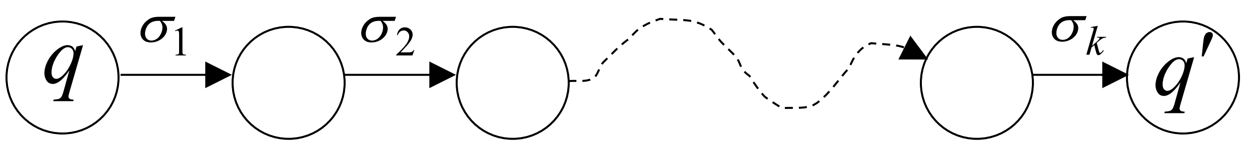
for any state q

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

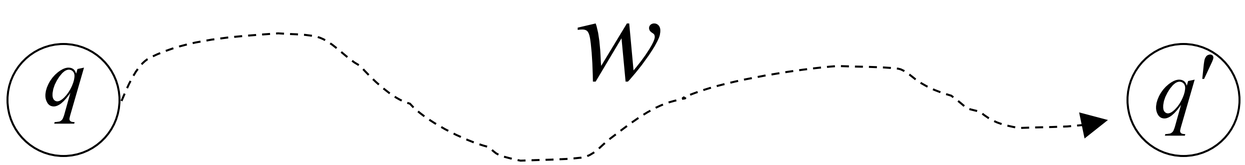
In general: $\delta^*(q,w) = q'$

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 M :

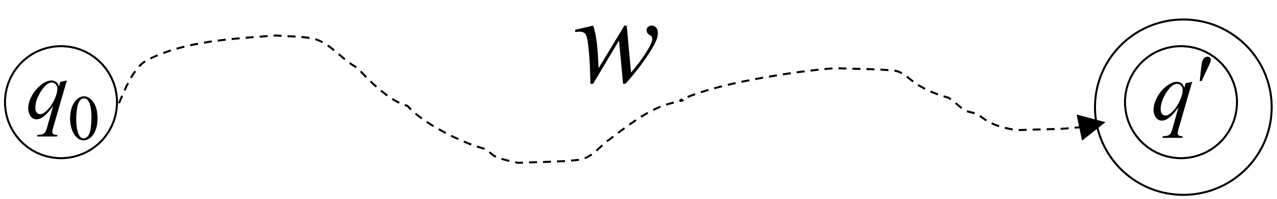
it is denoted as $L(M)$ and contains all the strings accepted by M

We say that a language L' is accepted (or recognized) by DFA M if $L(M) = L'$

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

Language accepted by M:

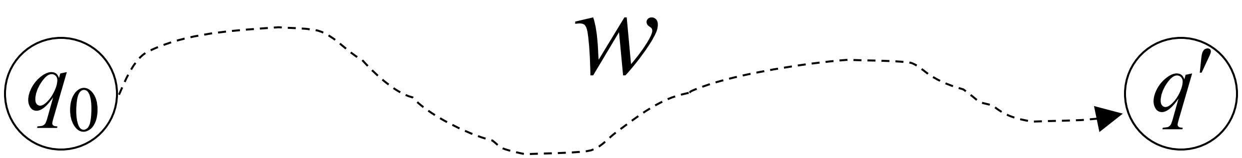
$$L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$$



$q' \in F$

Language rejected by M:

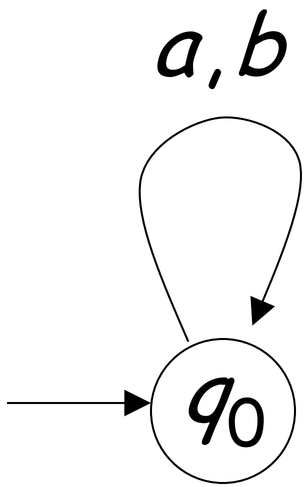
$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$



$q' \notin F$

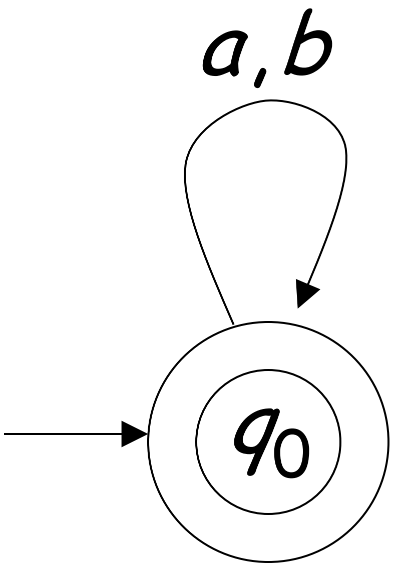
More DFA Examples

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



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

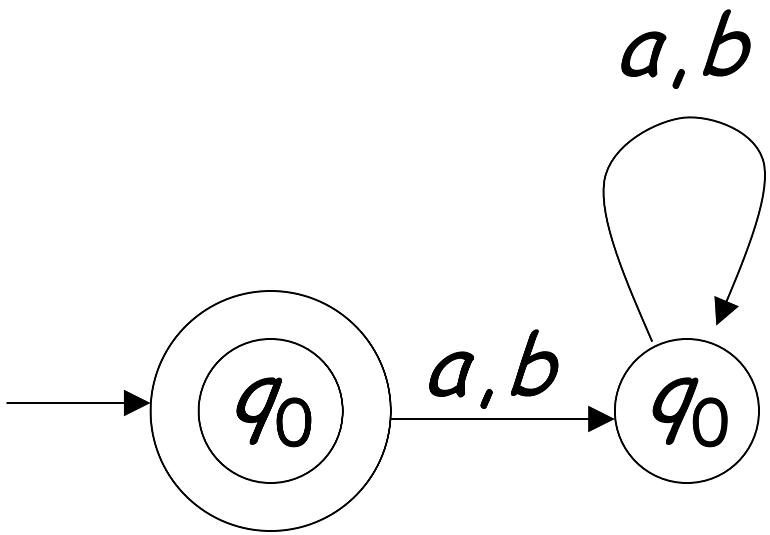
Empty language



$$L(M) = \Sigma^*$$

All strings

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

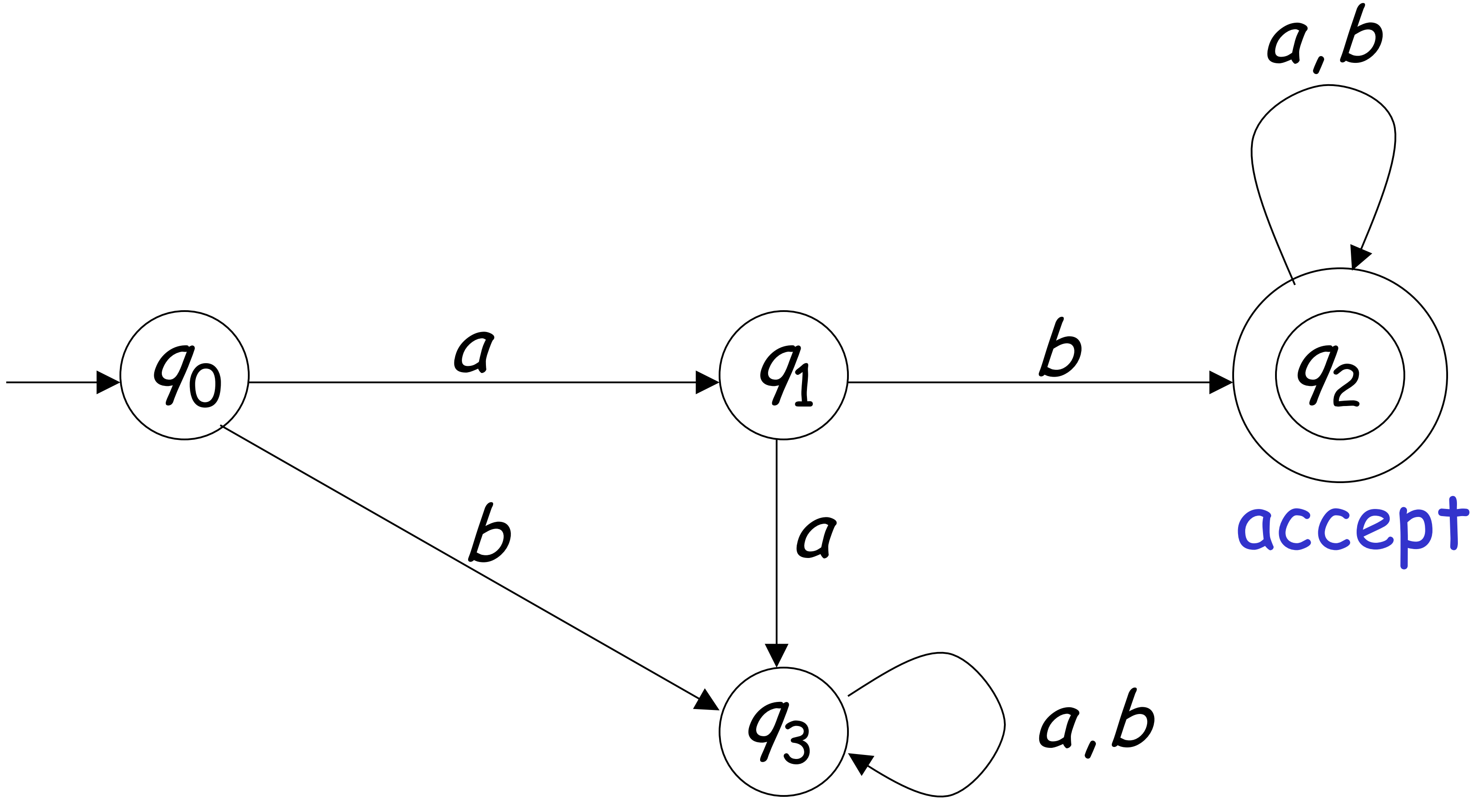


$$L(M) = \{\lambda\}$$

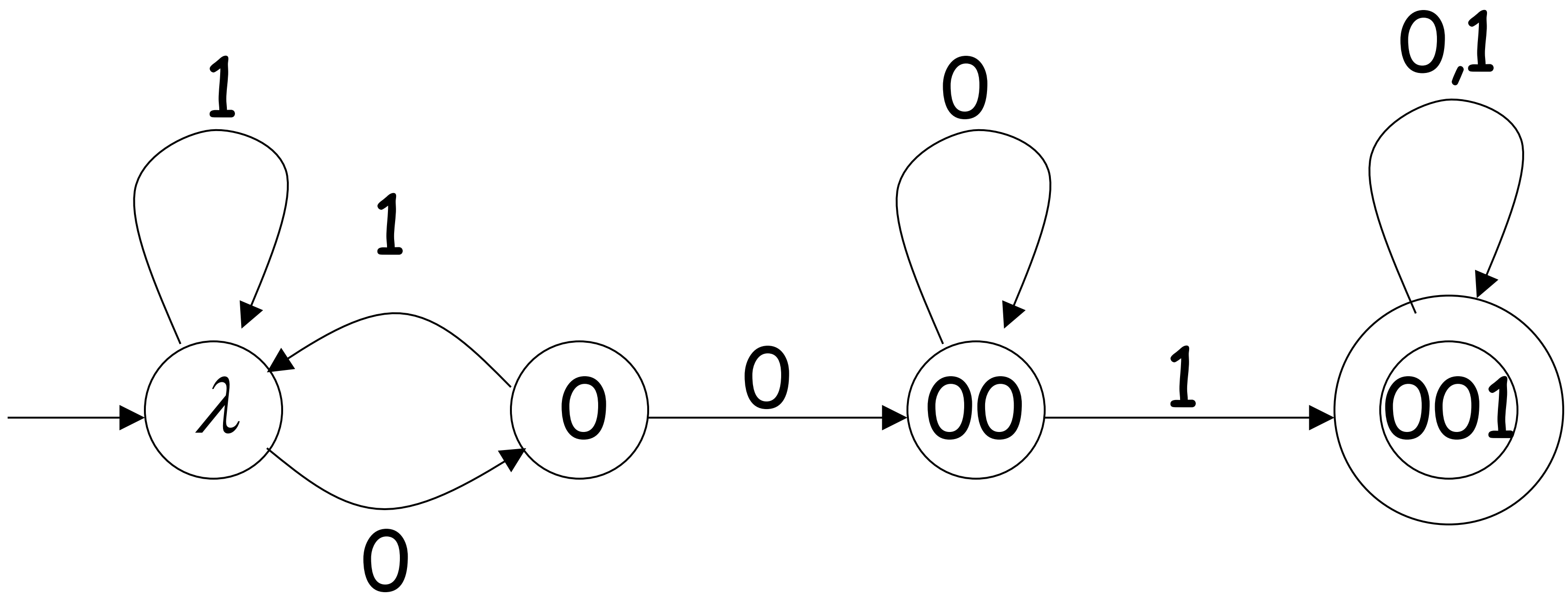
Language of the empty string

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

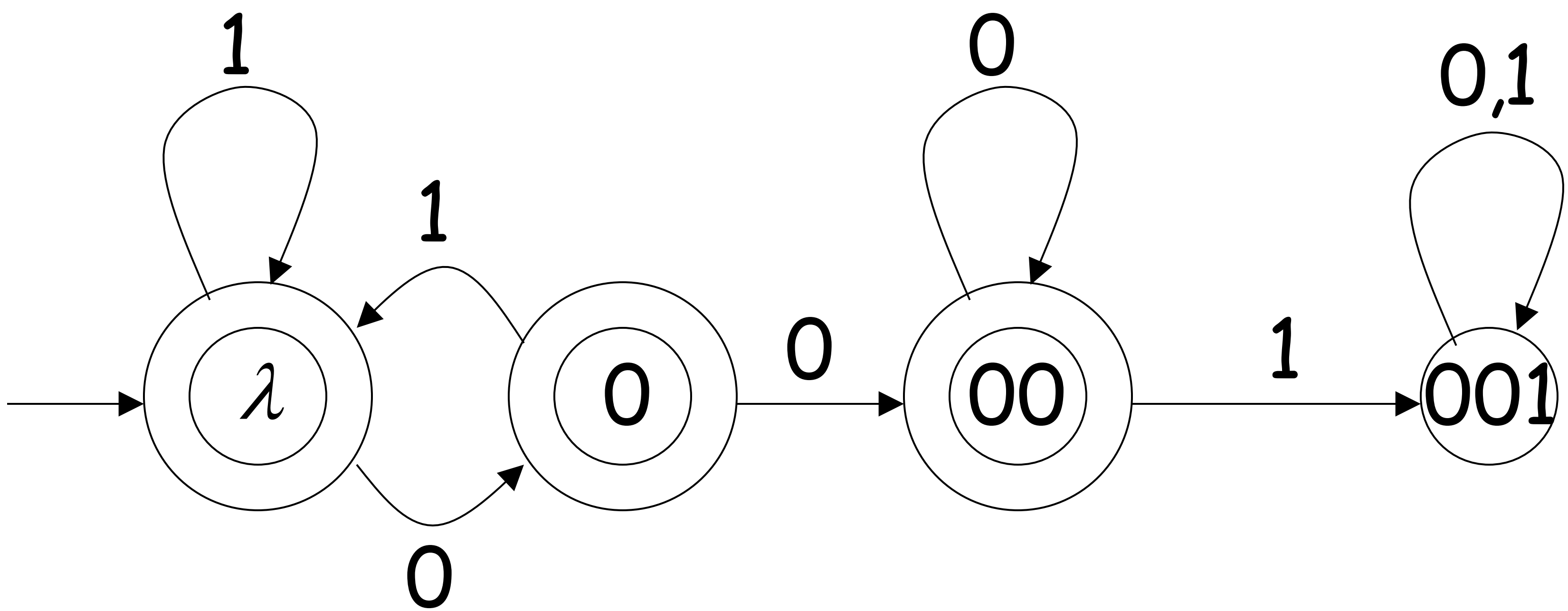
$L(M)$ = { all strings with prefix ab }



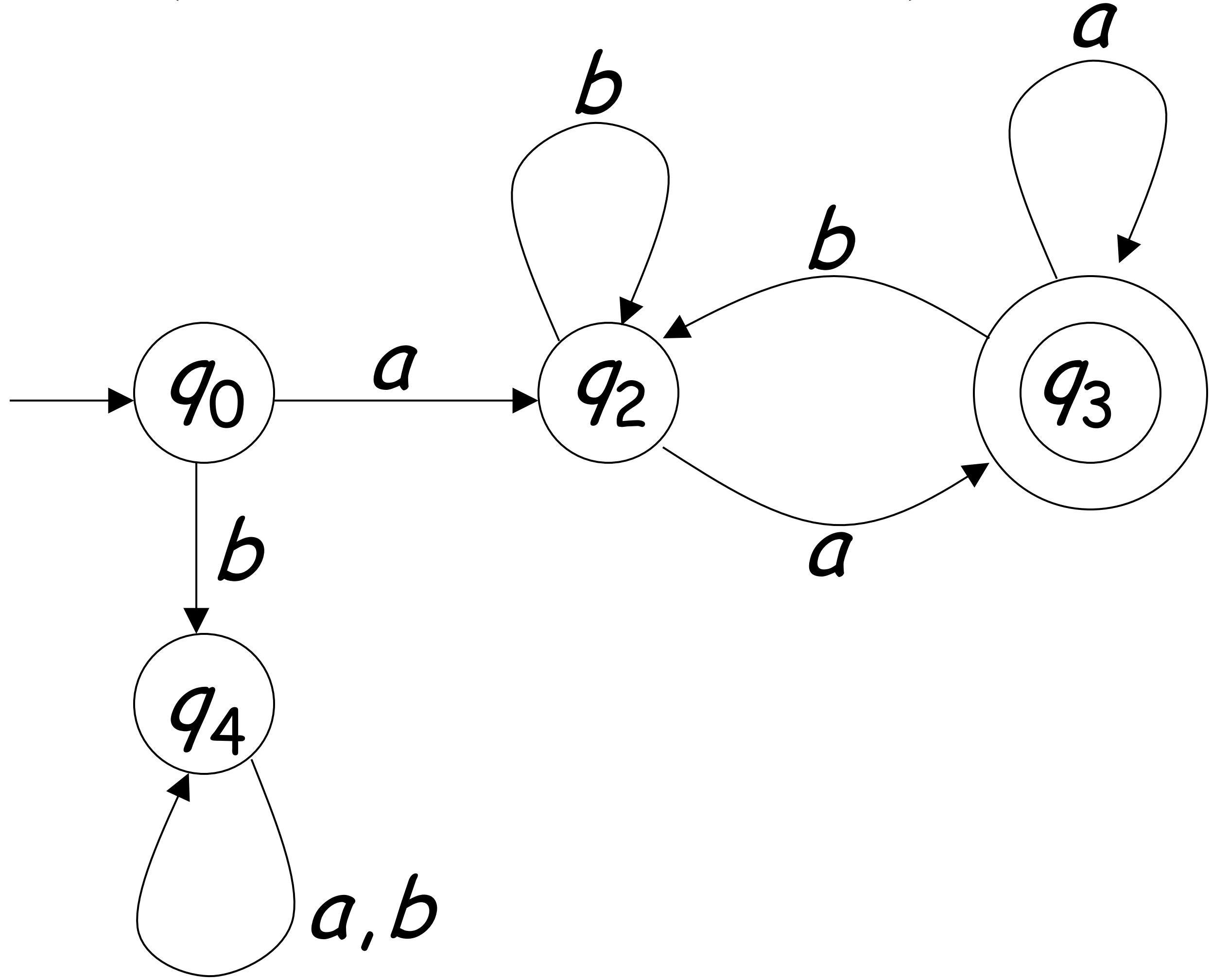
$L(M)$ = { all binary strings containing substring 001 }



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



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



Regular Languages

Definition:

A language L is regular if there is a DFA M that accepts it ($L(M) = L$)

The languages accepted by all DFAs form the family of regular languages

Example regular languages:

$\{abba\}$ $\{\lambda, ab, abba\}$

$\{a^n b : n \geq 0\}$ $\{awa : w \in \{a,b\}^*\}$

{ all strings in {a,b}* with prefix ab }

{ all binary strings without substring 001}

$\{x : x \in \{1\}^*$ and x is even$\}$

$\{\ \}$ $\{\lambda\}$ $\{a,b\}^*$

There exist automata that accept these languages (see previous slides).

There exist languages which are <u>not</u> Regular:

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

$$ADDITION = \{x + y = z \ : x = 1^n, y = 1^m, z = 1^k, \; n + m = k\}$$

There is no DFA that accepts these languages

(we will prove this in a later class)