COM2109

Automata

Robert Hierons

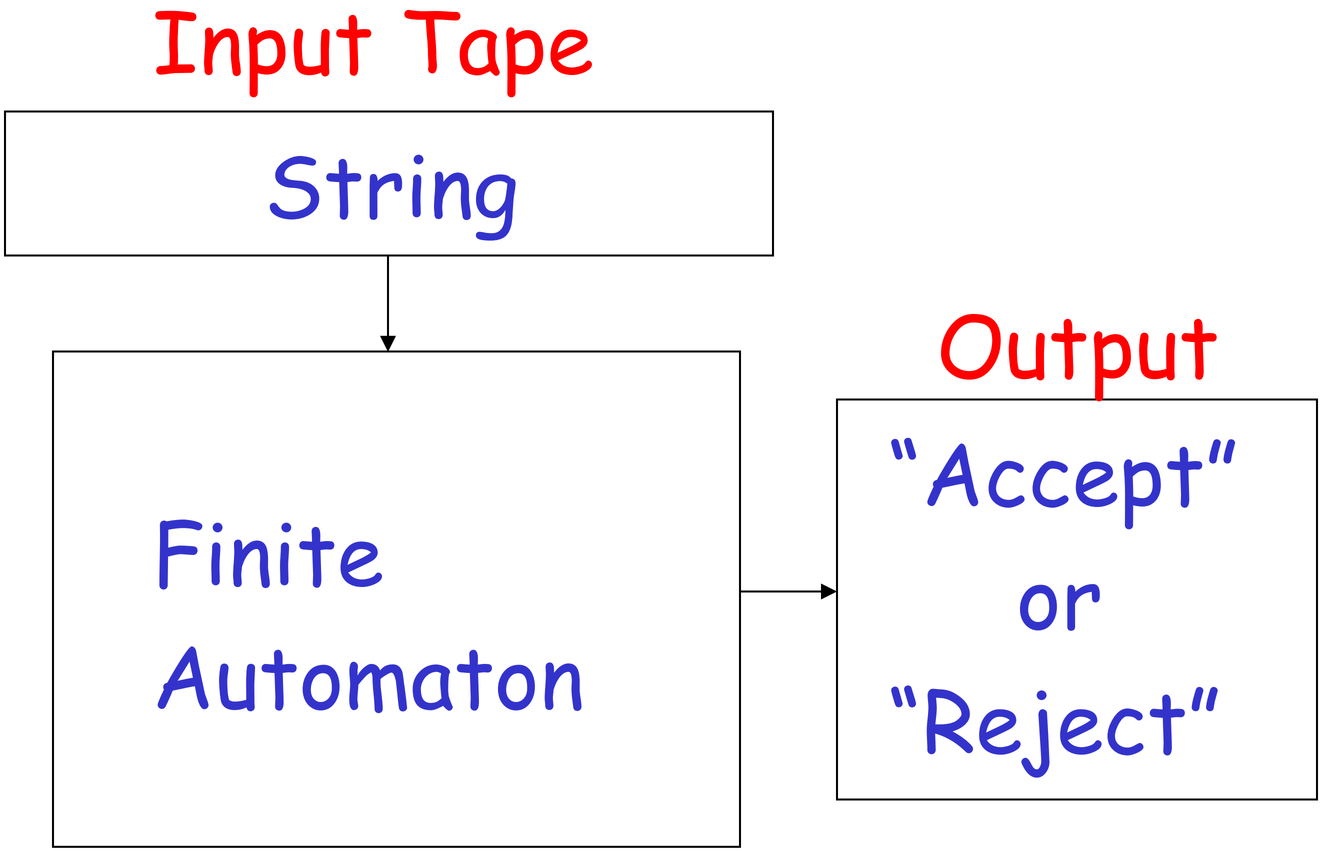
Based on the work of Lucia Specia,
M.S. Moorthy and Costas Busch

Deterministic

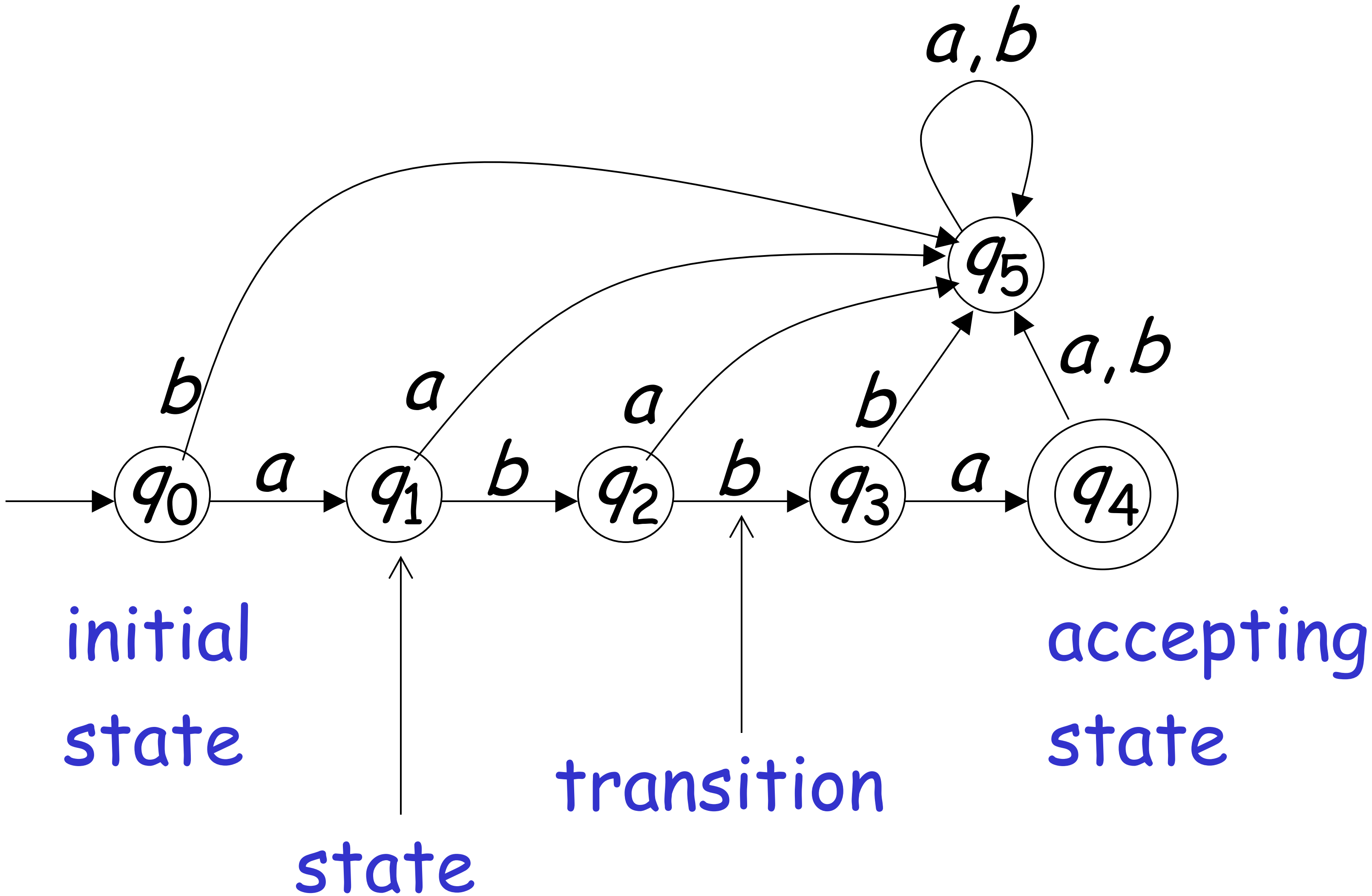
Finite Automata

And Regular Languages

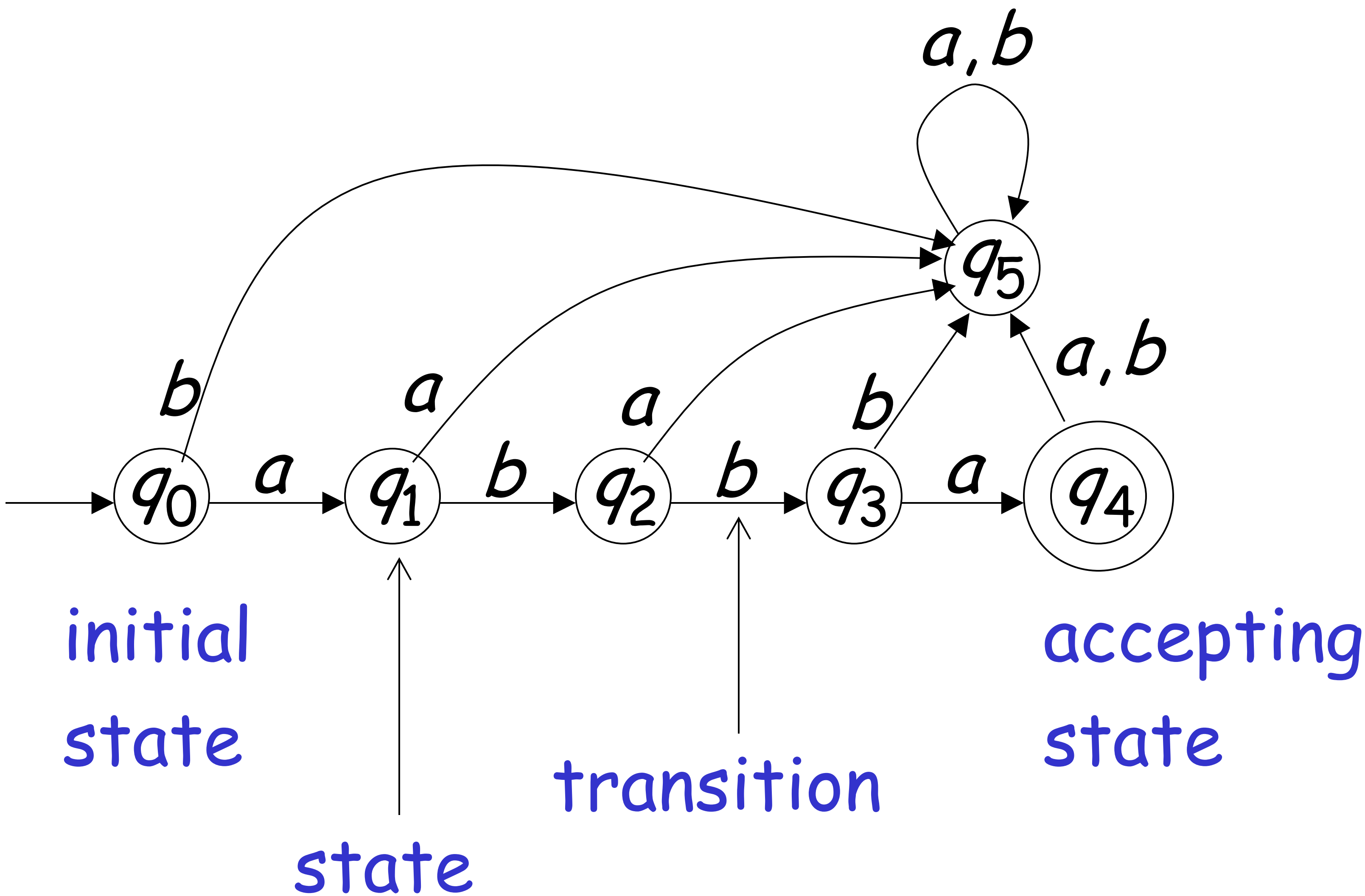
Deterministic Finite Automaton (DFA)



Transition Graph

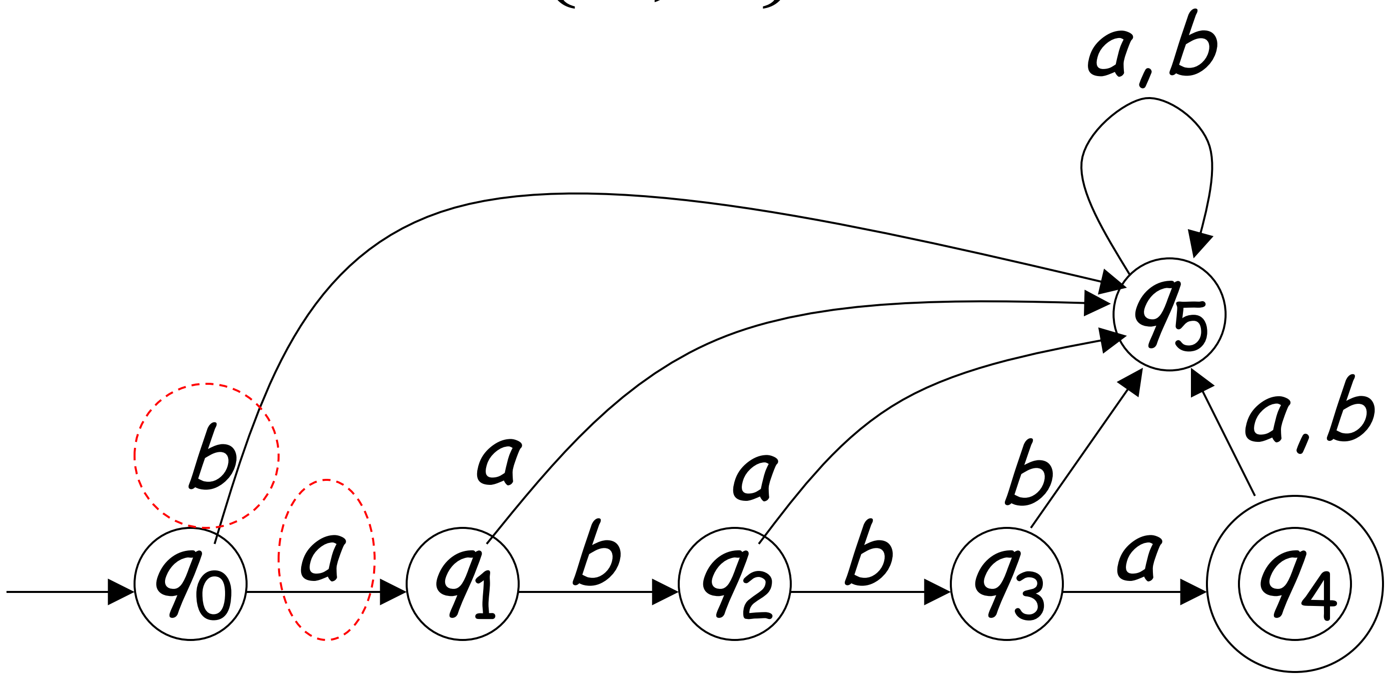


Transition Graph



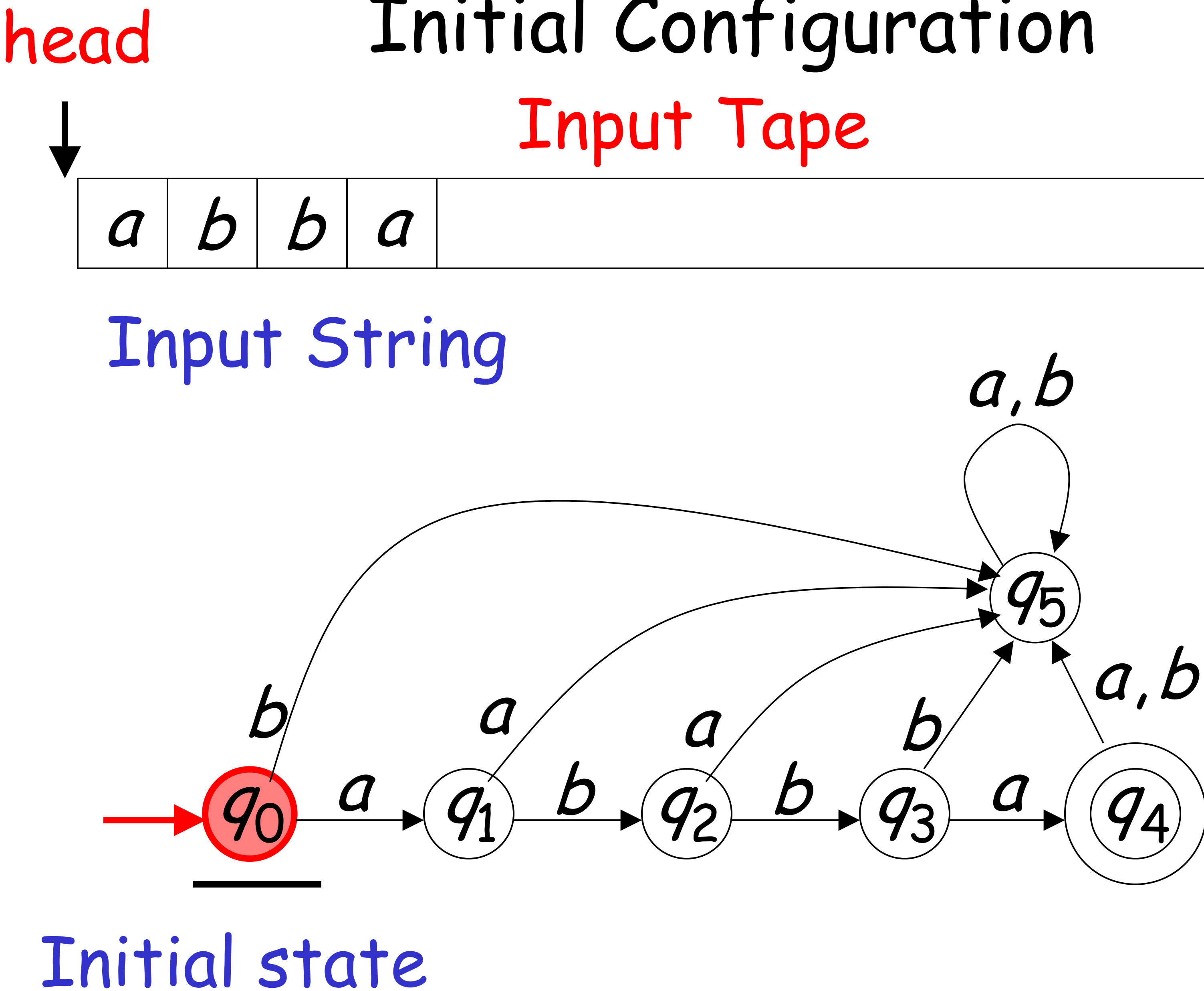
Also called a finite state machine

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

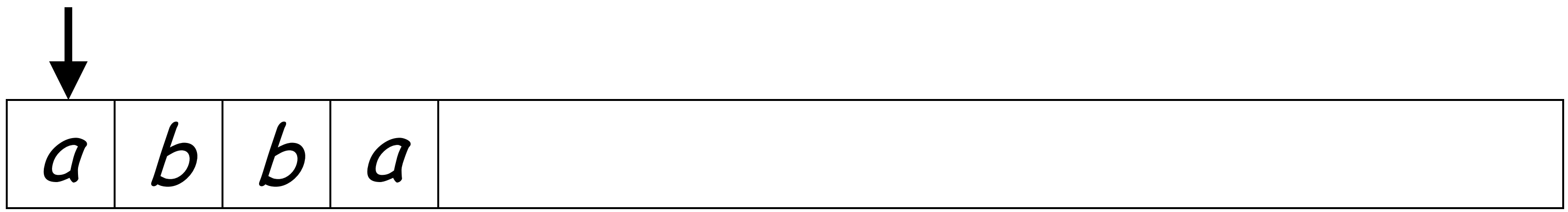


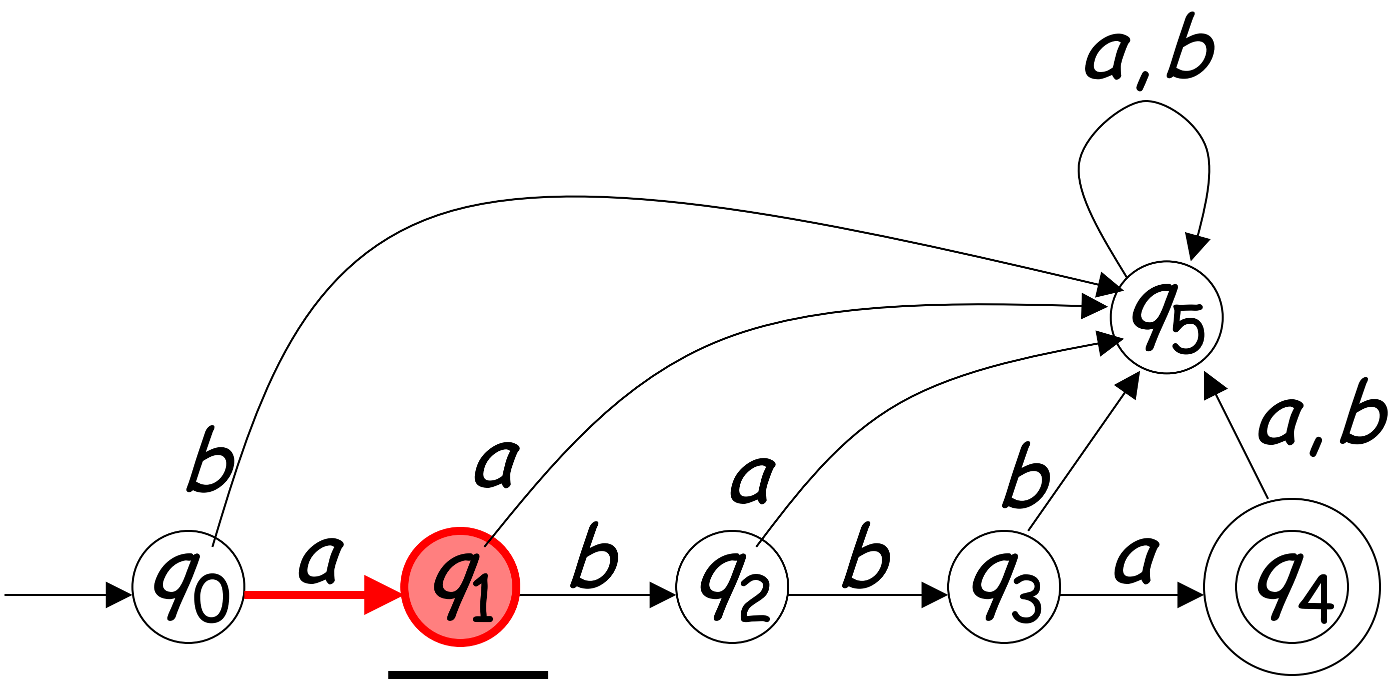
For every state, there is a transition for every symbol in the alphabet

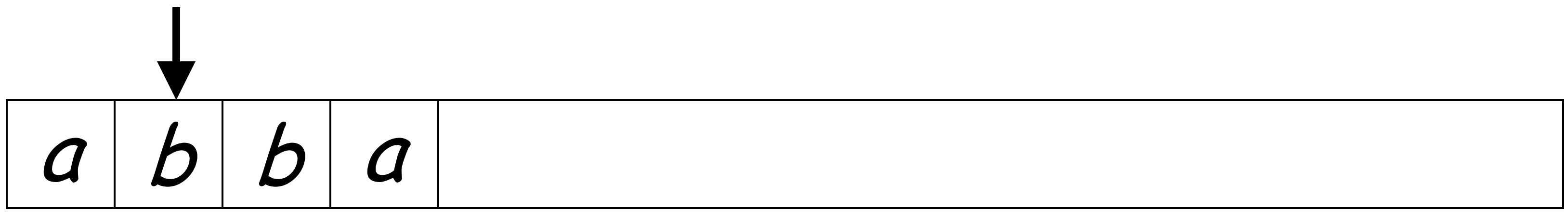
Initial Configuration

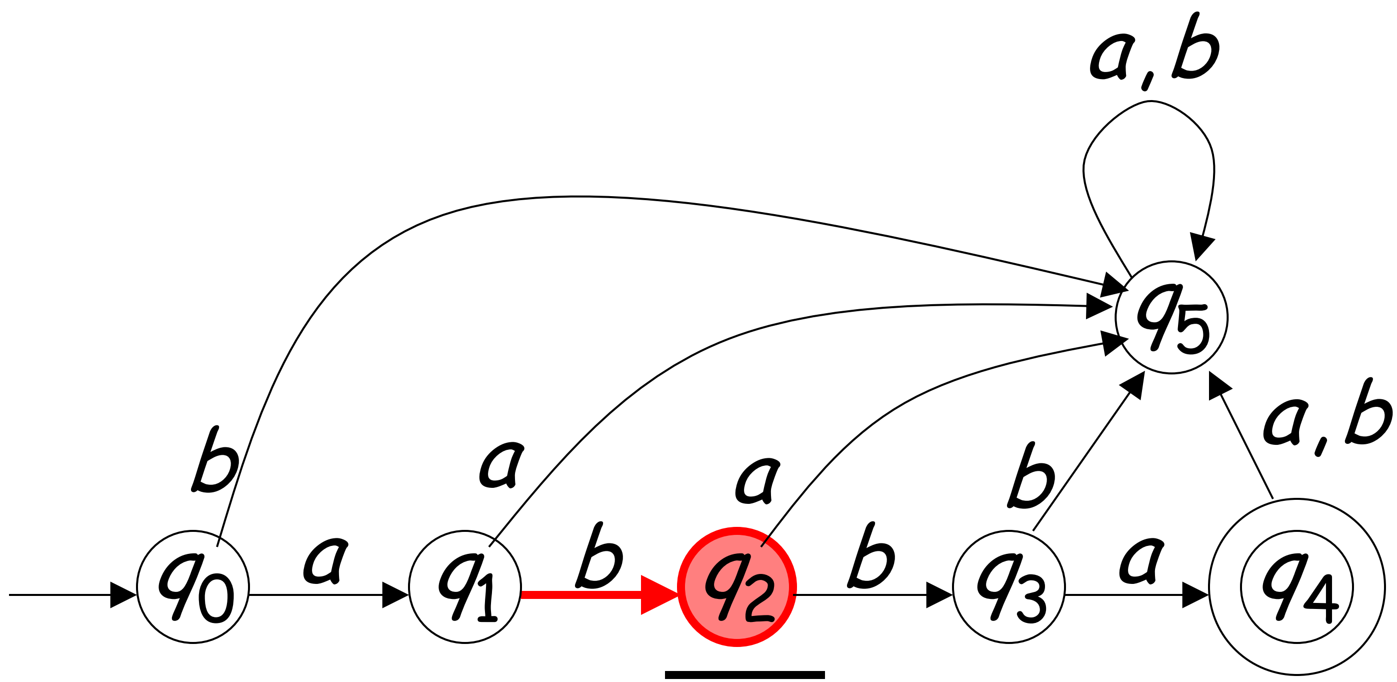


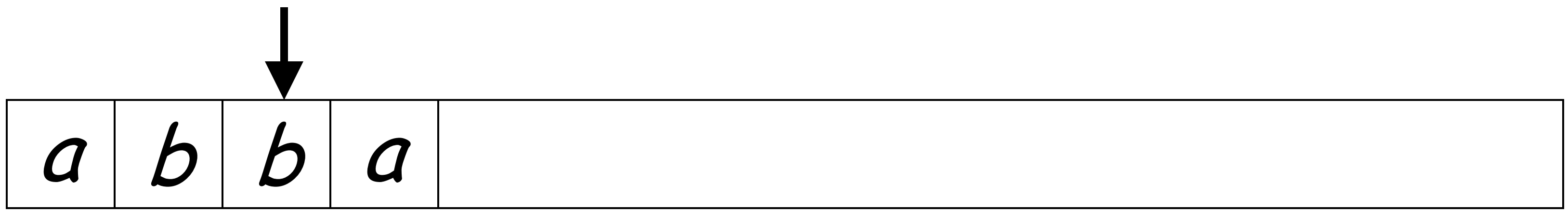
Scanning the Input

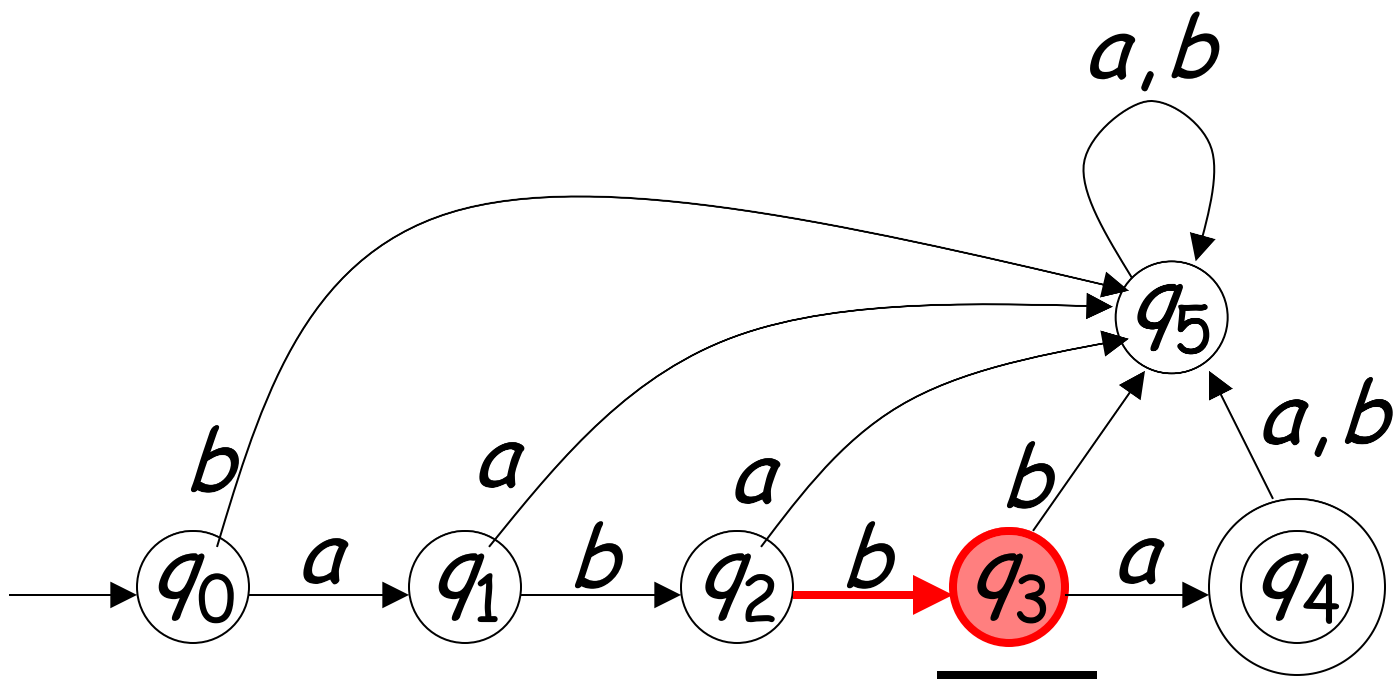




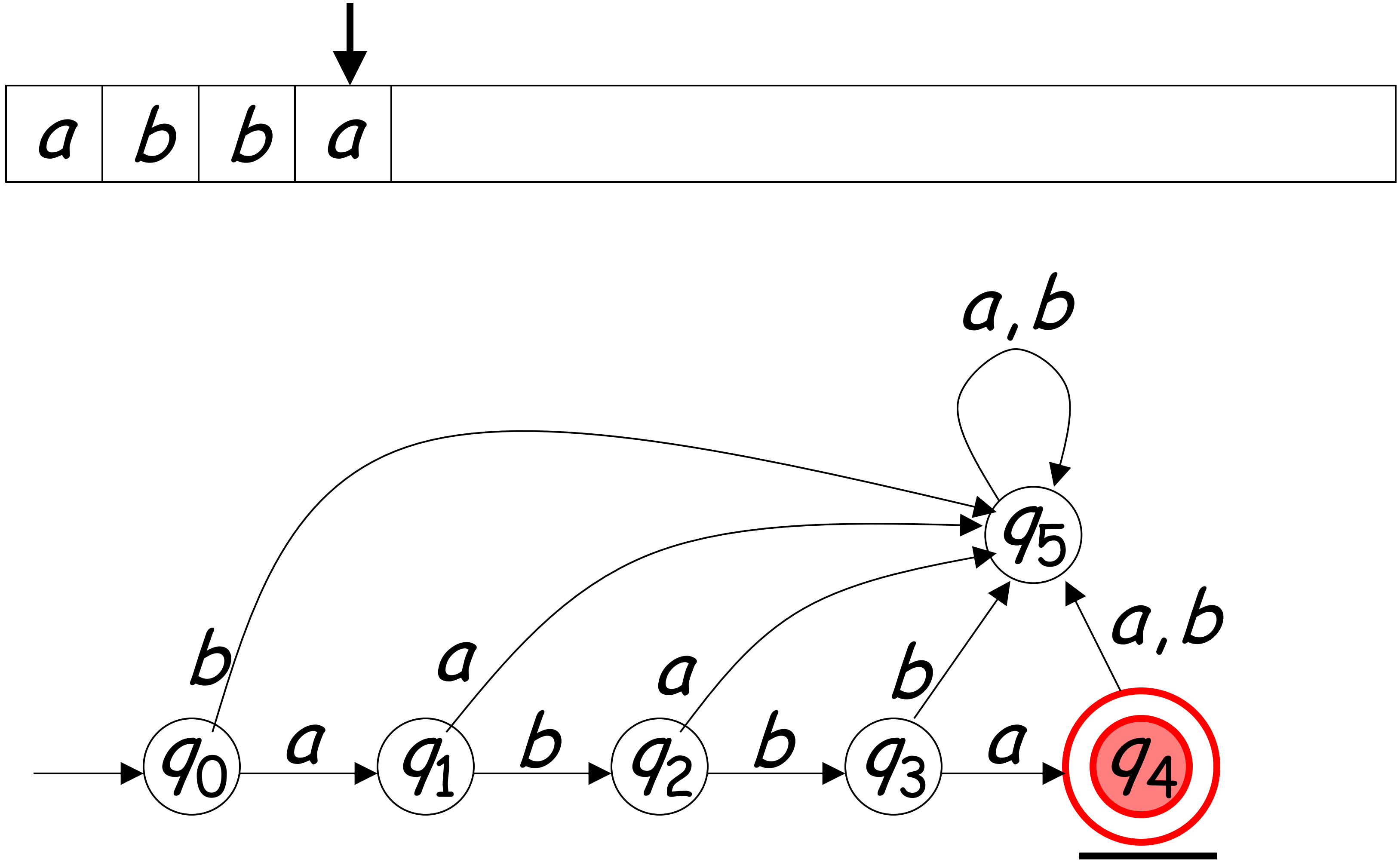

a
b
b
a


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


a
b
b
a

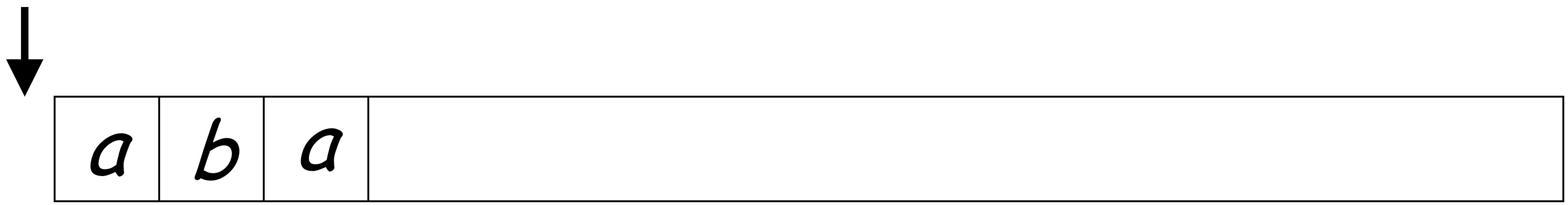

a,b
q_5
b
a
a
b
a,b
q_0
a
q_1
b
q_2
b
q_3
a
q_4

Input finished

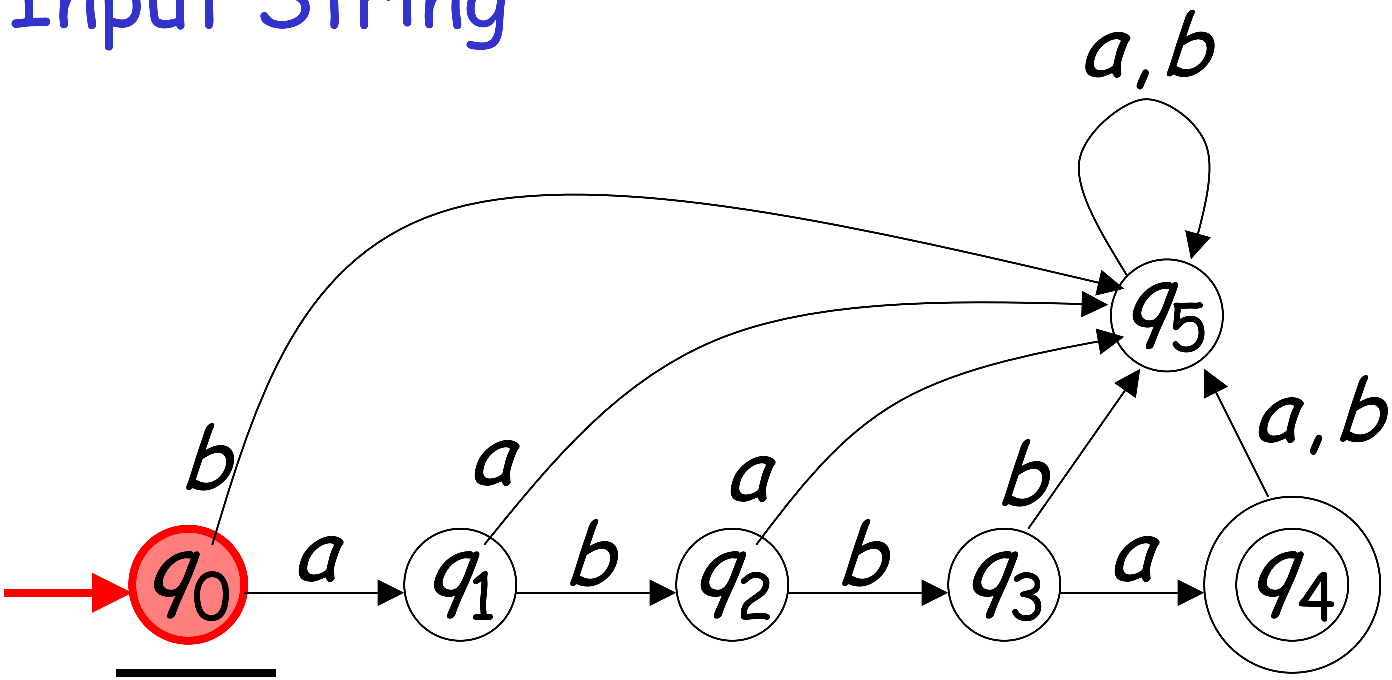


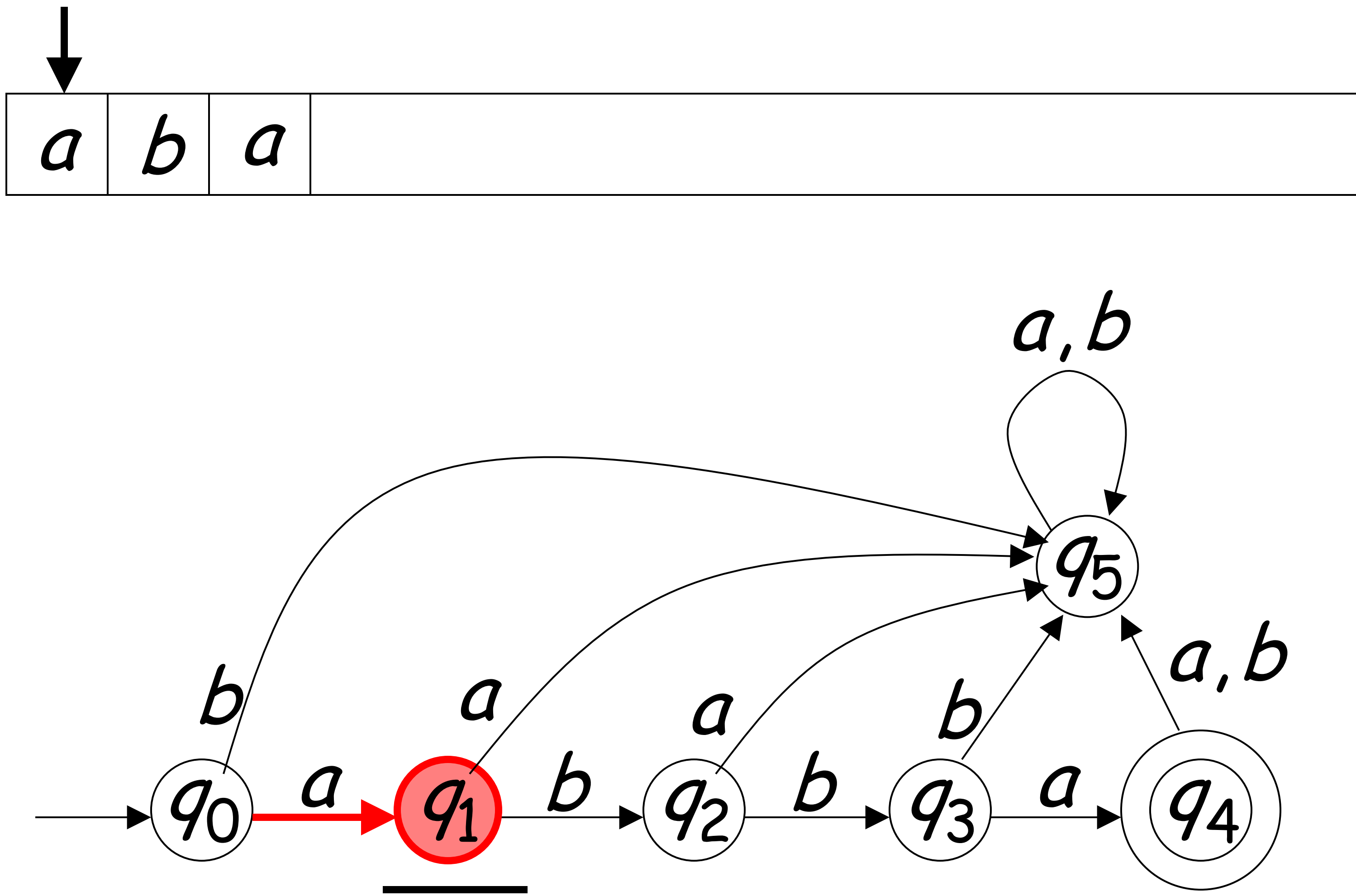
accept

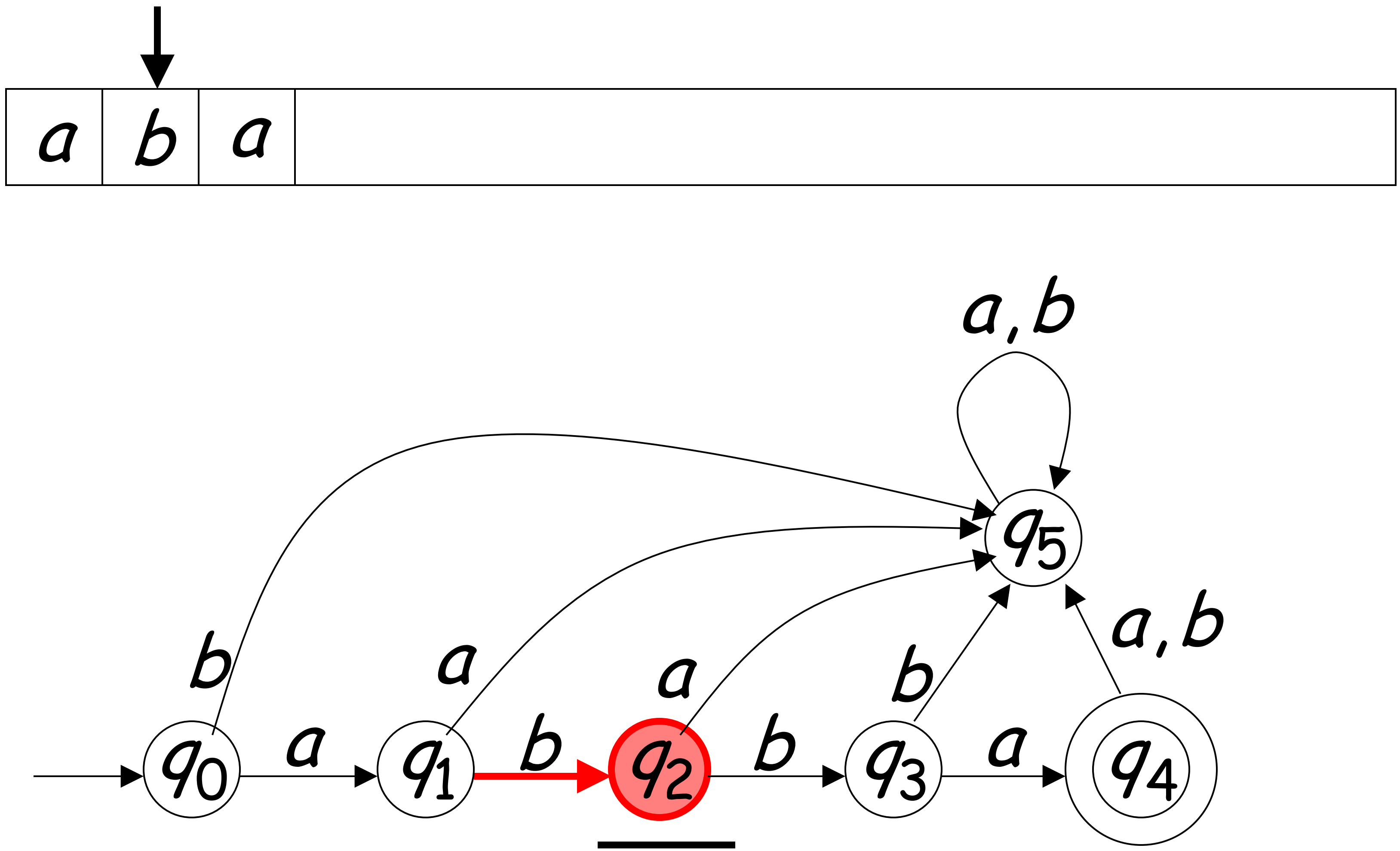
A Rejection Case



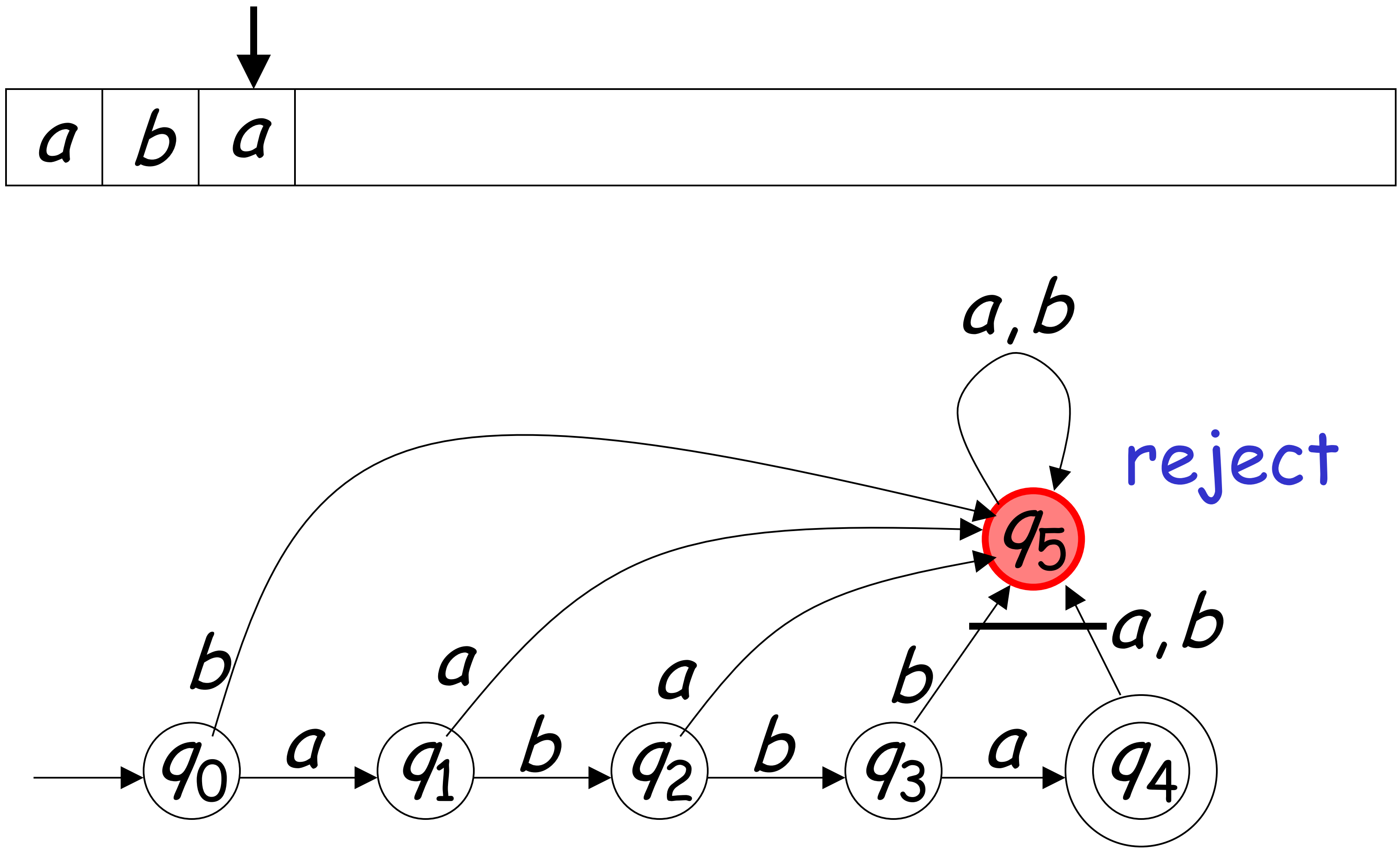
Input String




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


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

Input finished

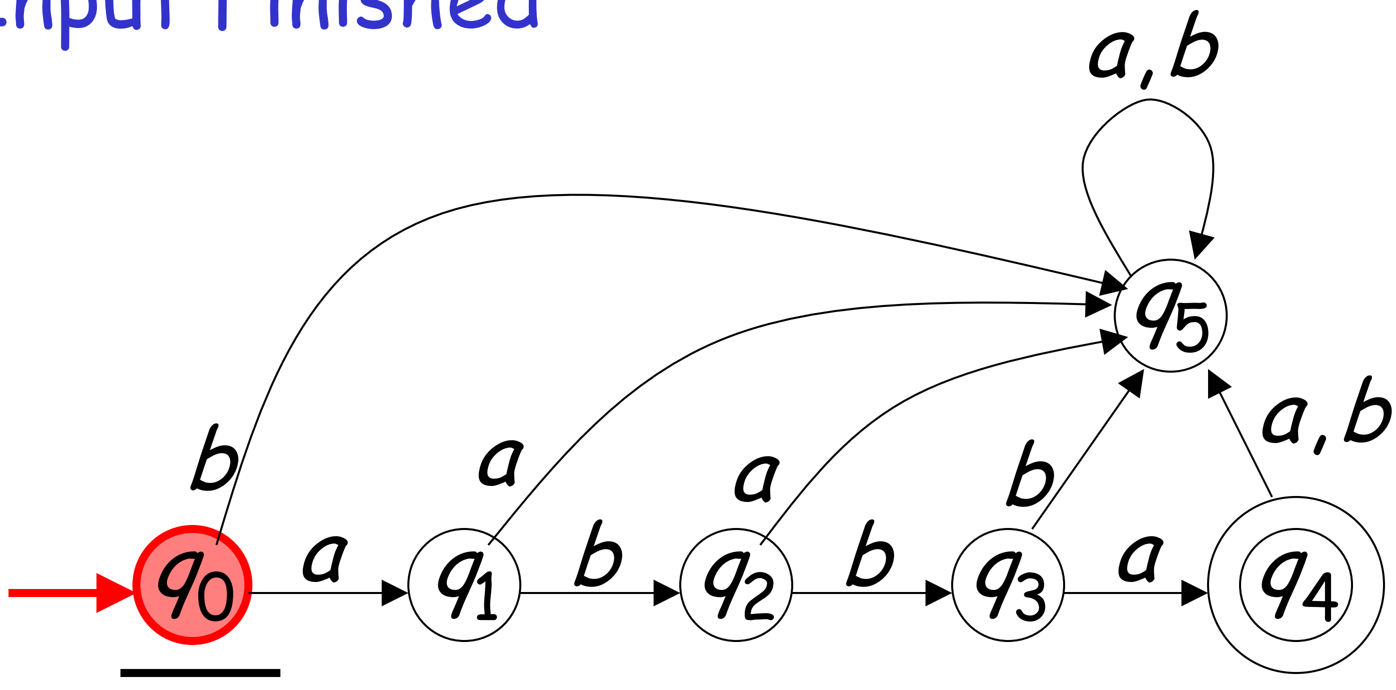


Another Rejection Case

Tape is empty

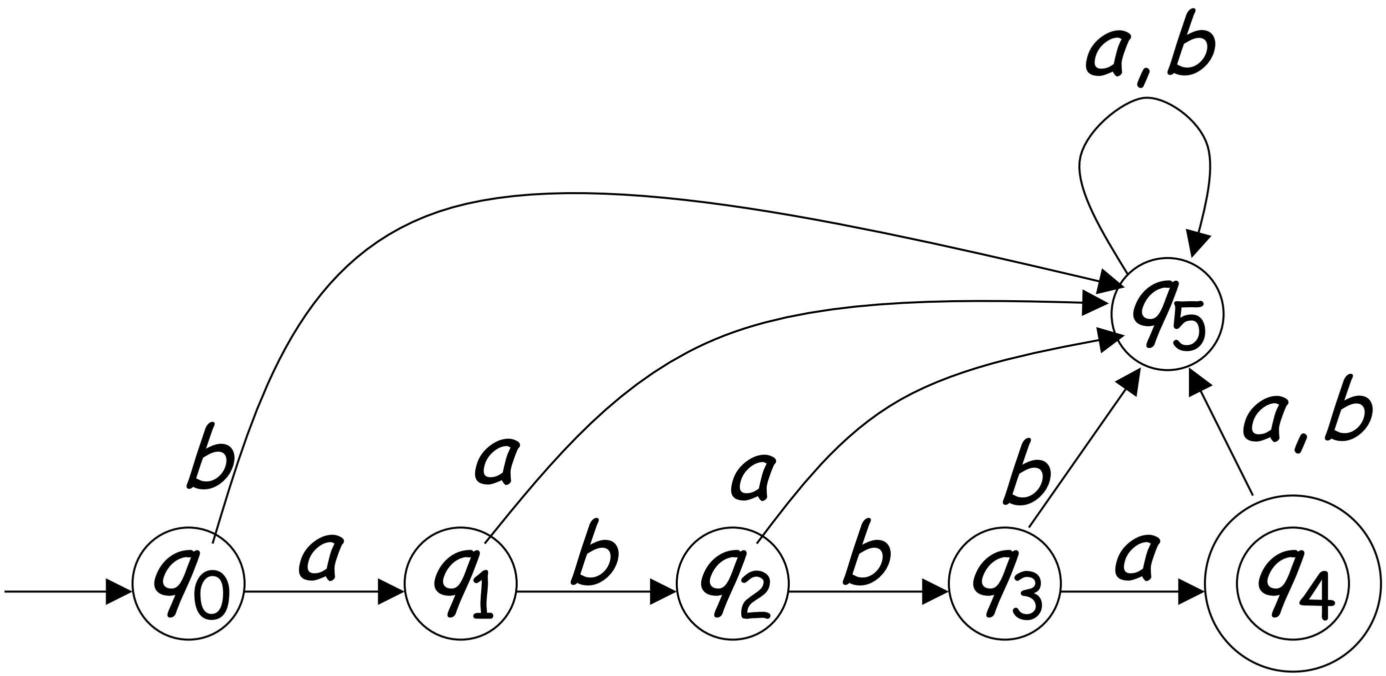
(ε)

Input Finished



reject

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



To accept a string:

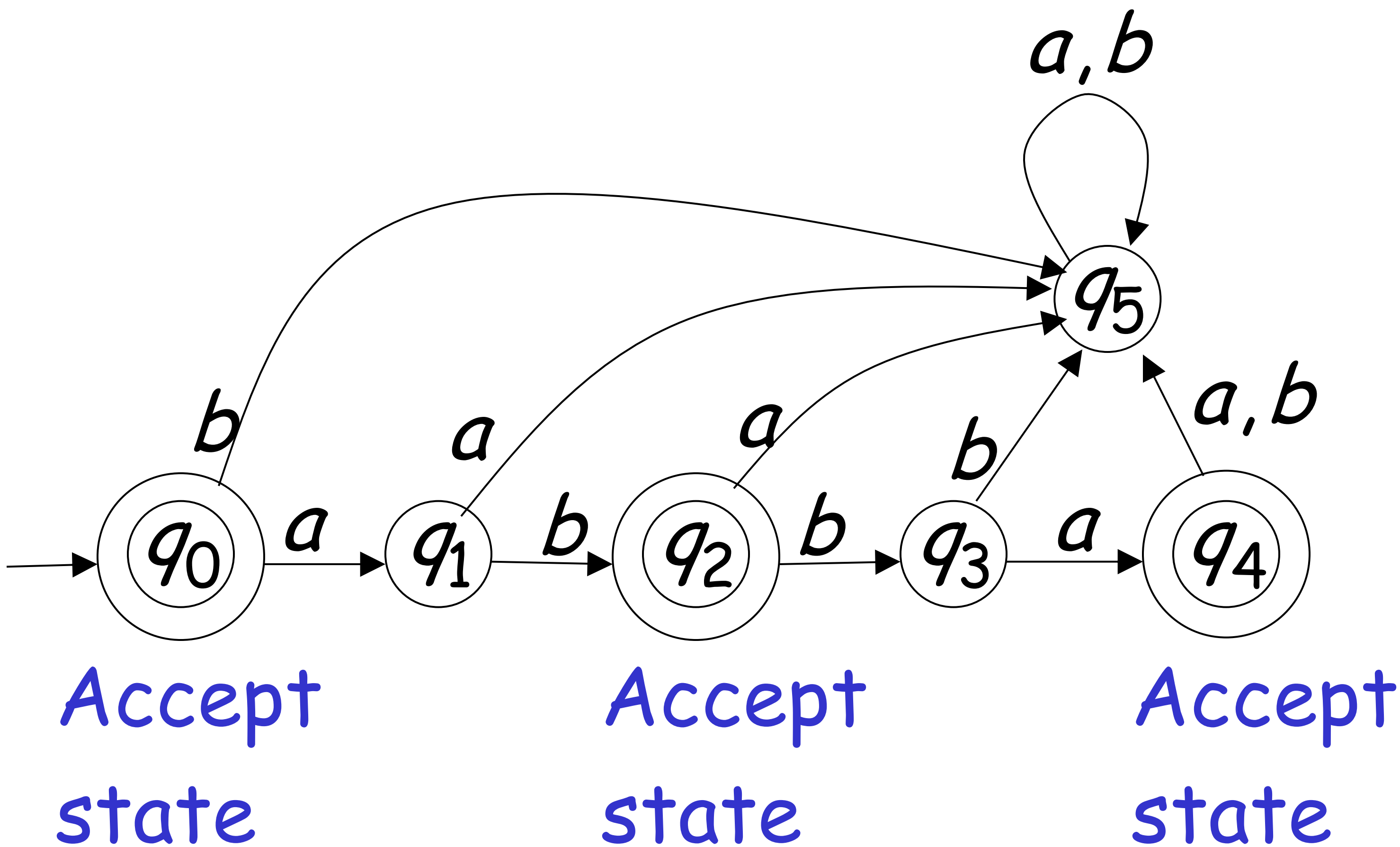
all the input string is scanned
and the last state is an accepting
state

To reject a string:

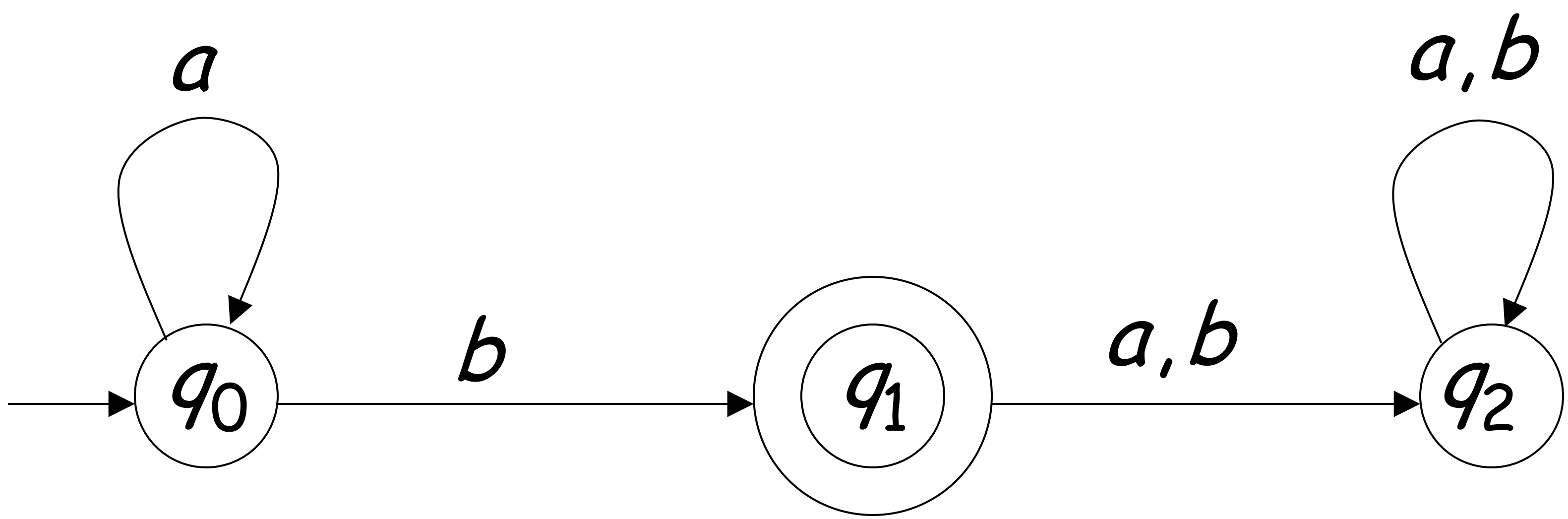
all the input string is scanned
and the last state is not an accepting
state

Another Example

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



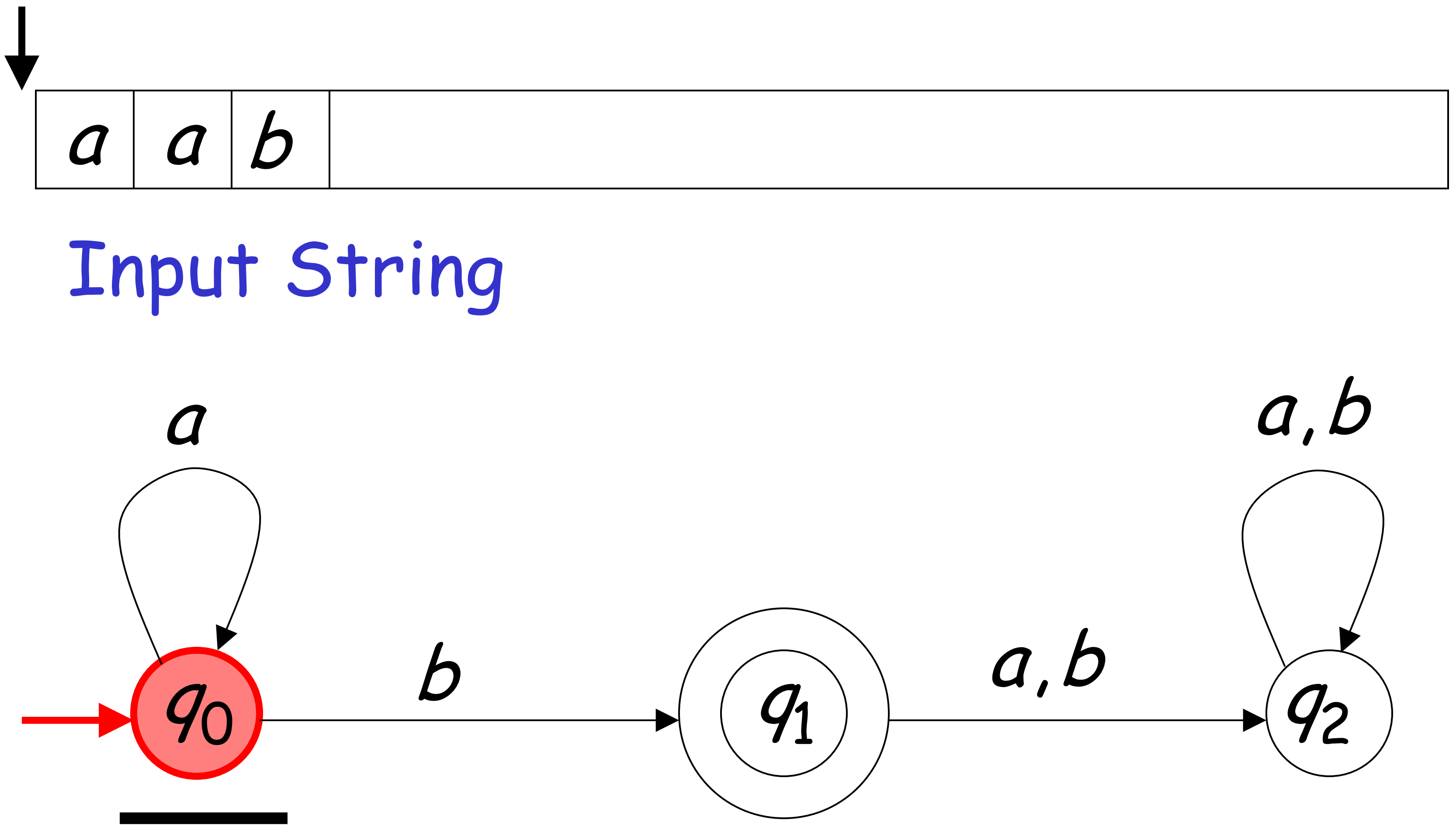
Another Example

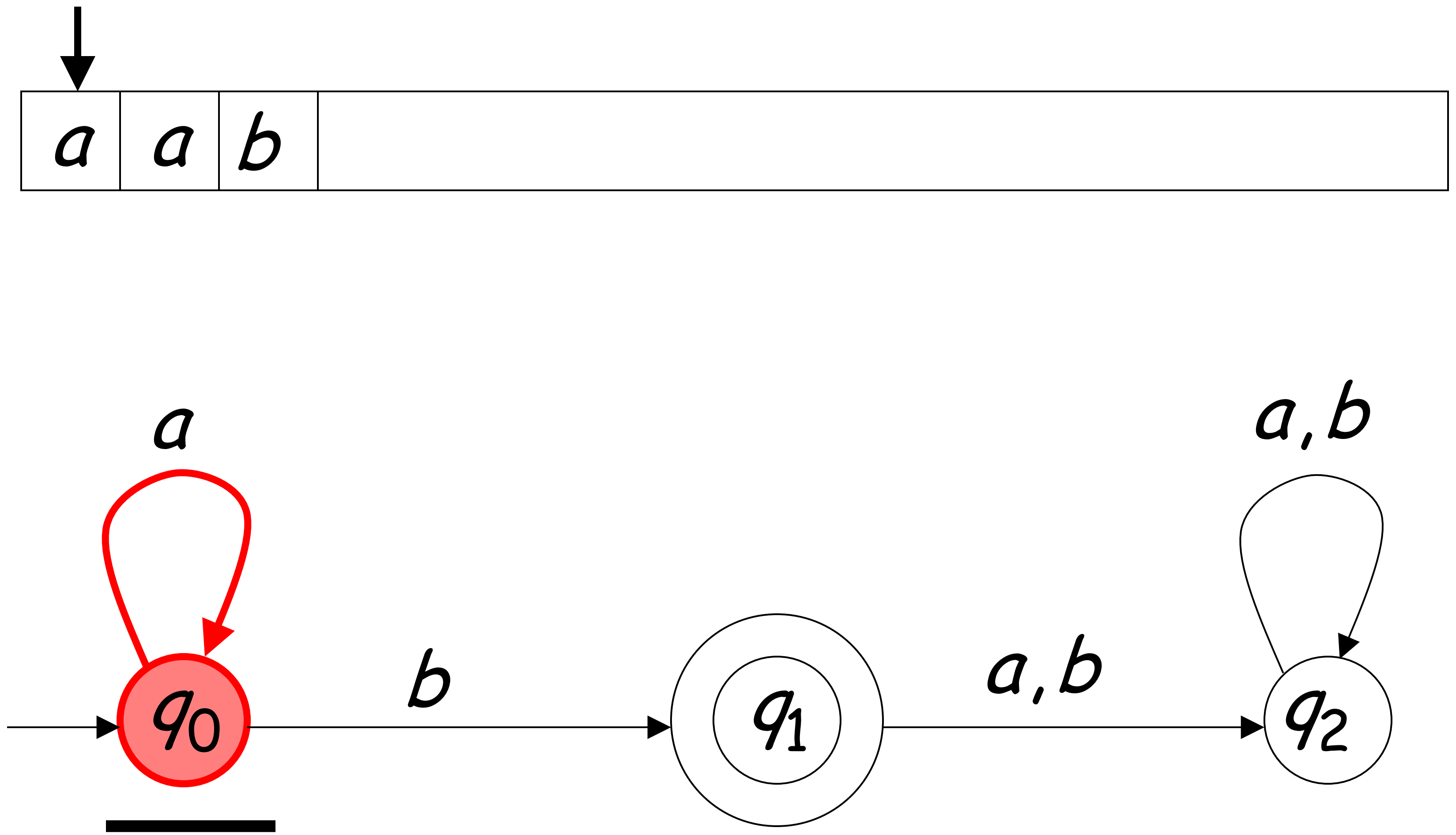


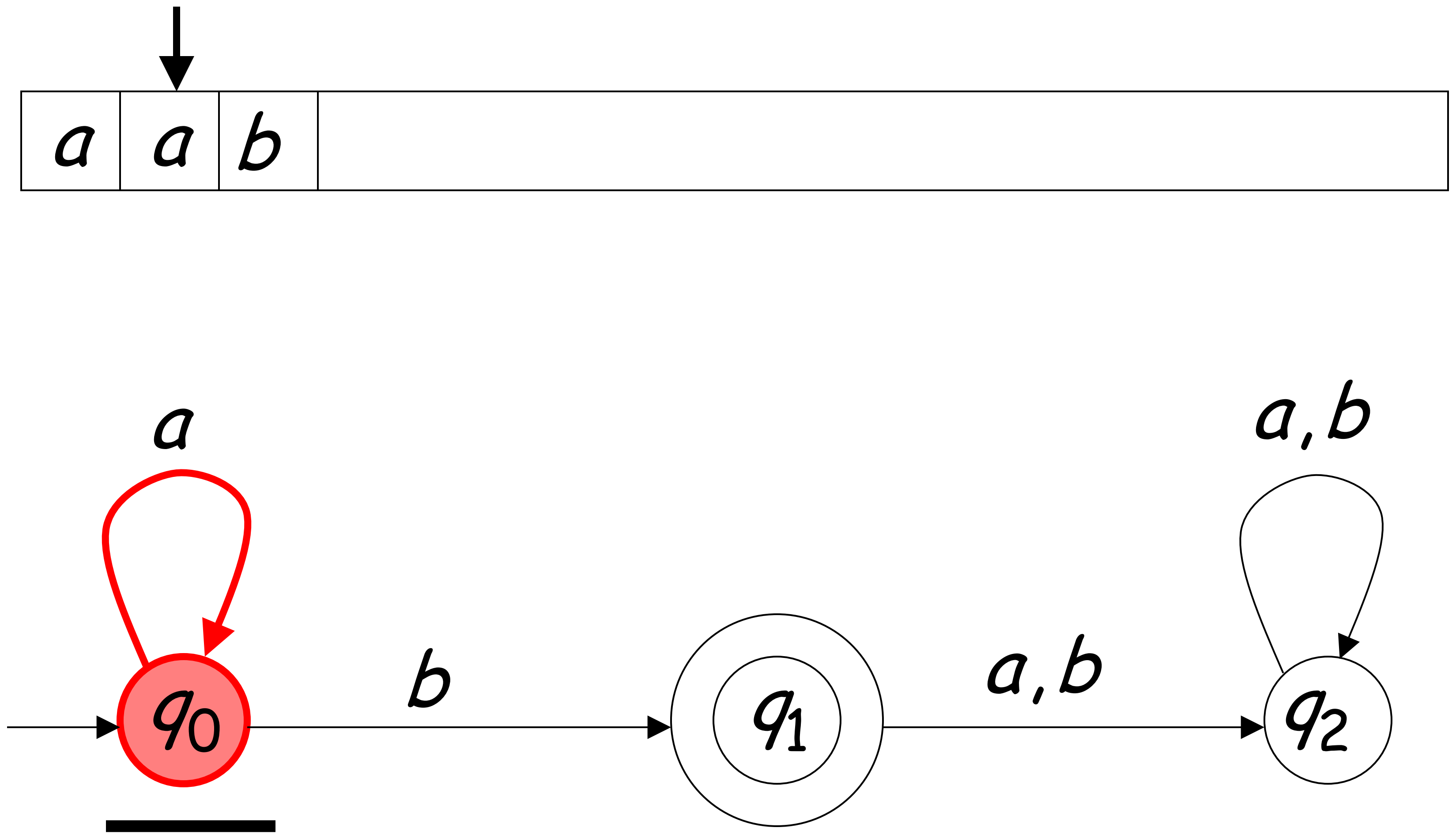
Accept state

trap state

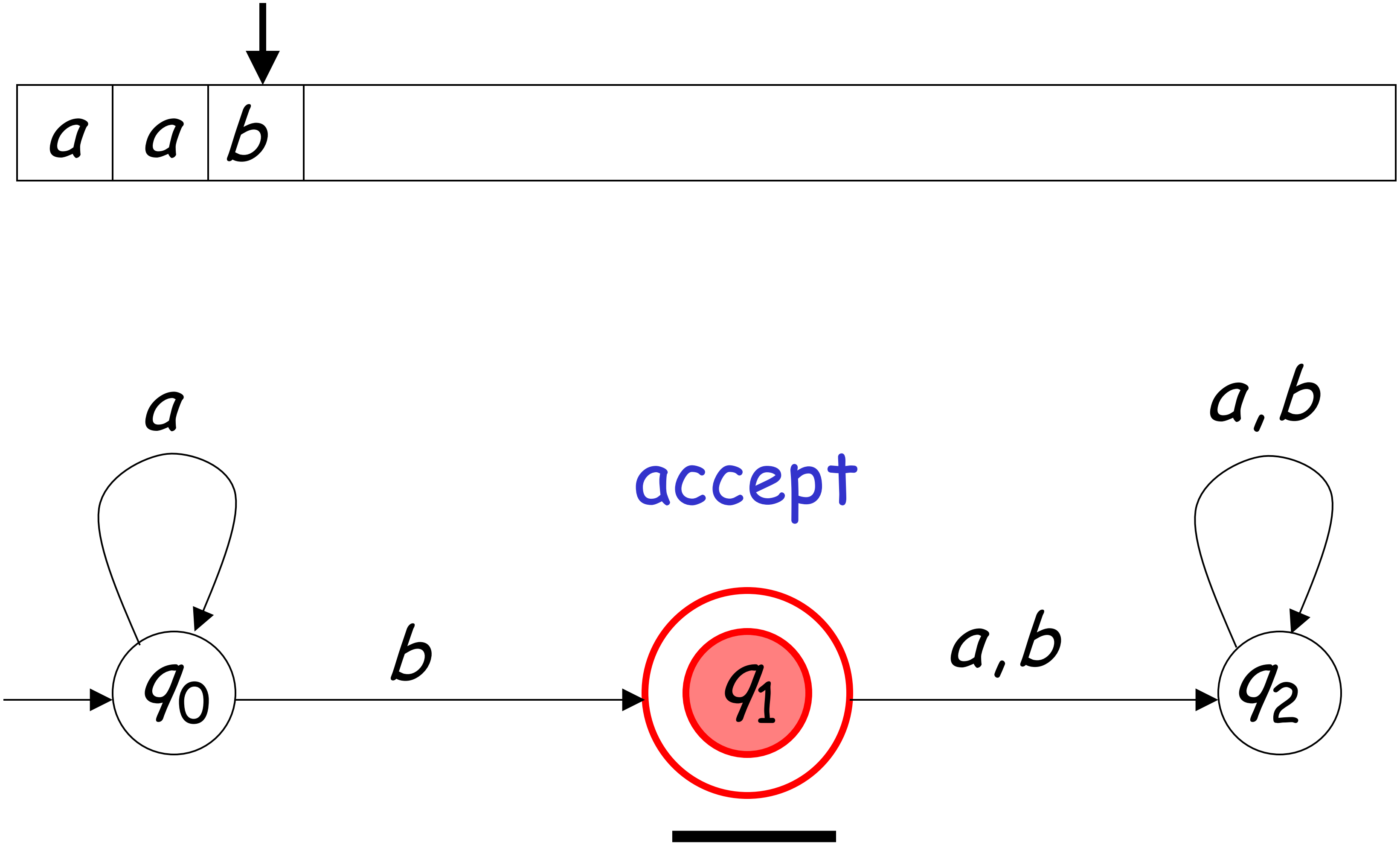
If a transition goes to a state from which it can never escape.


a
a
b
Input String
a
a,b
b
a,b
q_0
q_1
q_2

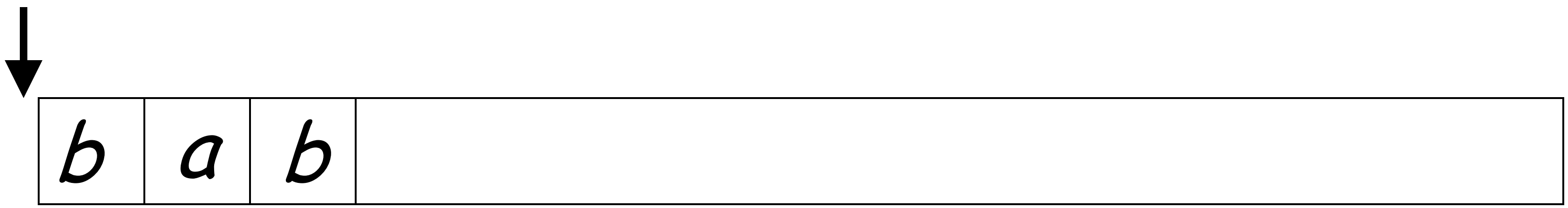

a
a
b
a
a,b
b
a,b
q_0
q_1
q_2


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

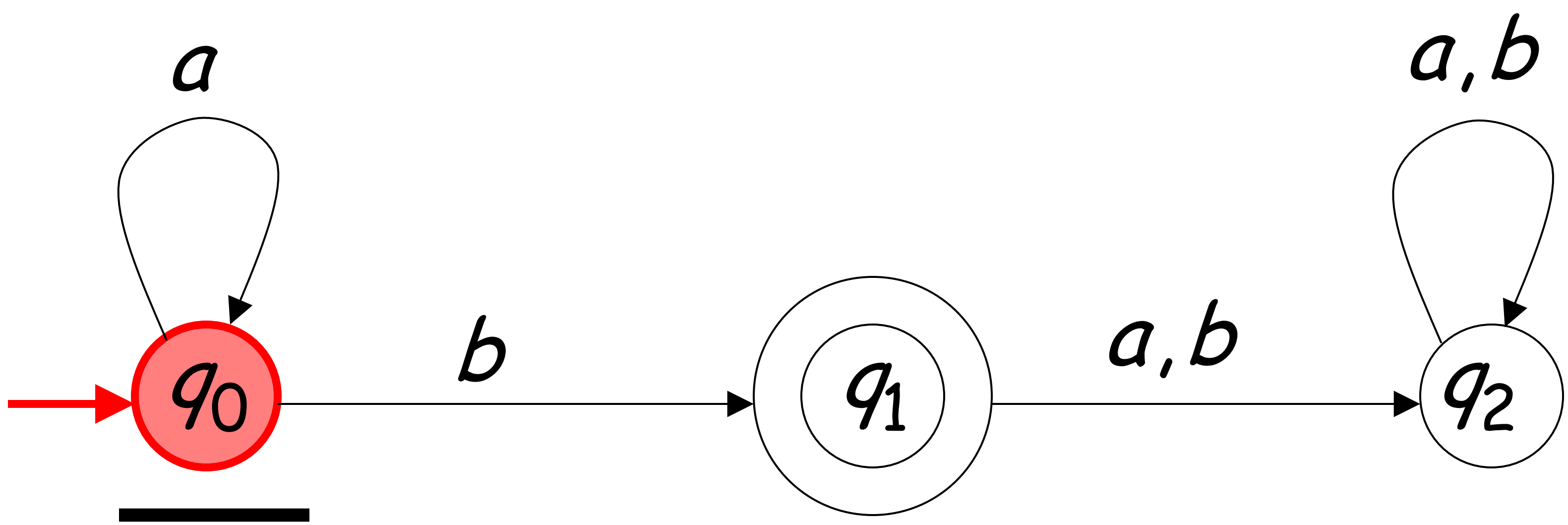
Input finished

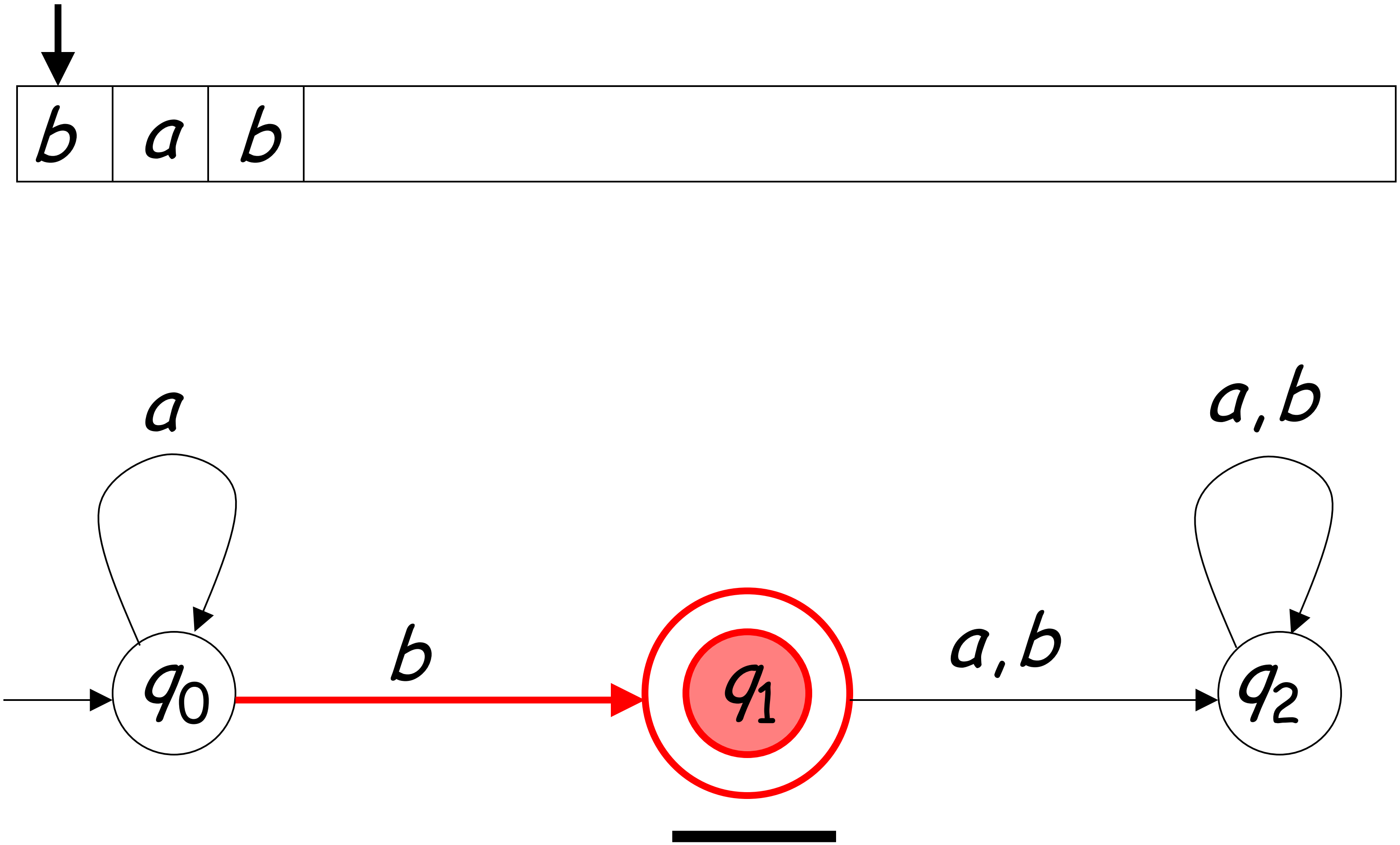


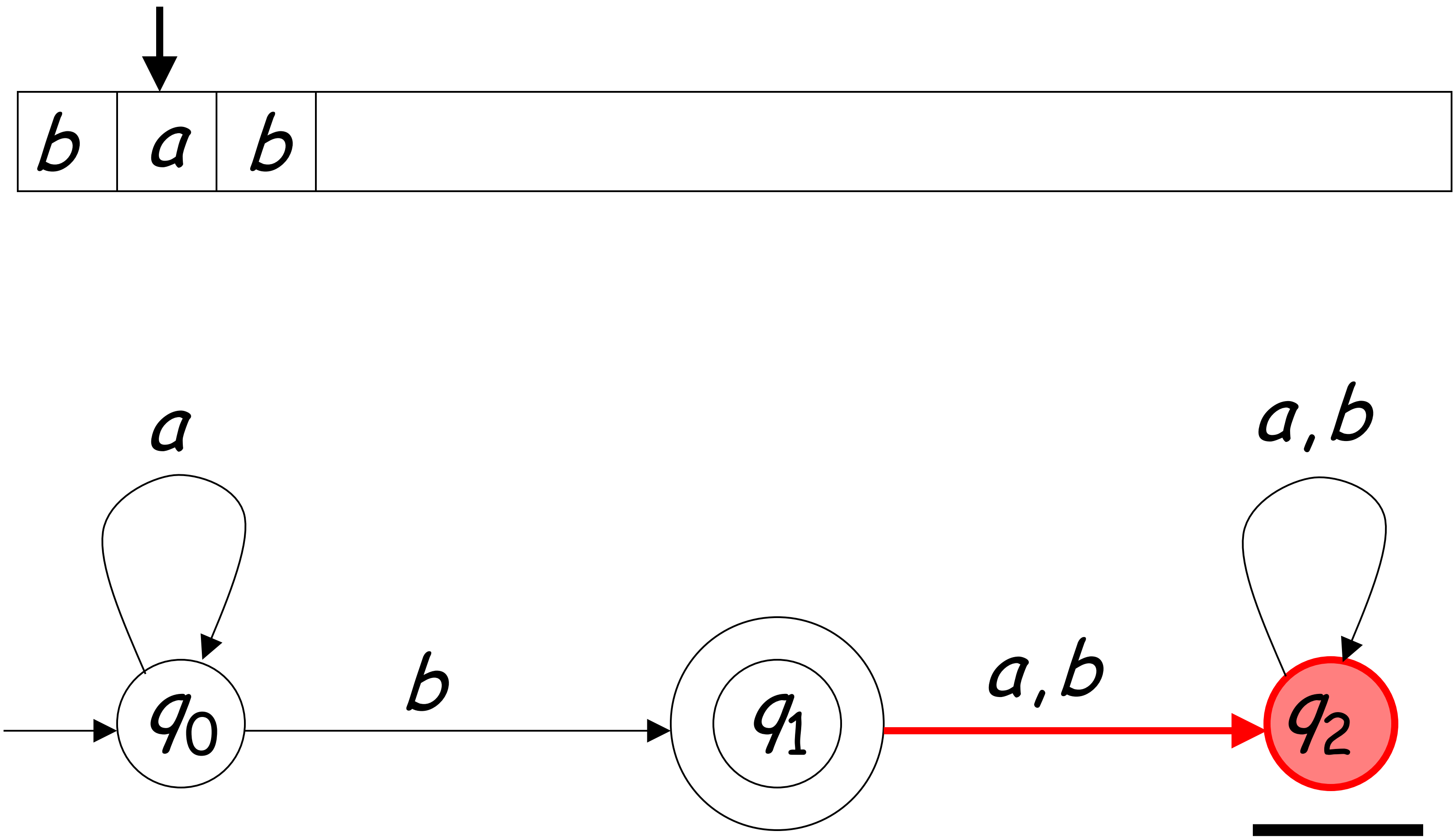
A rejection case



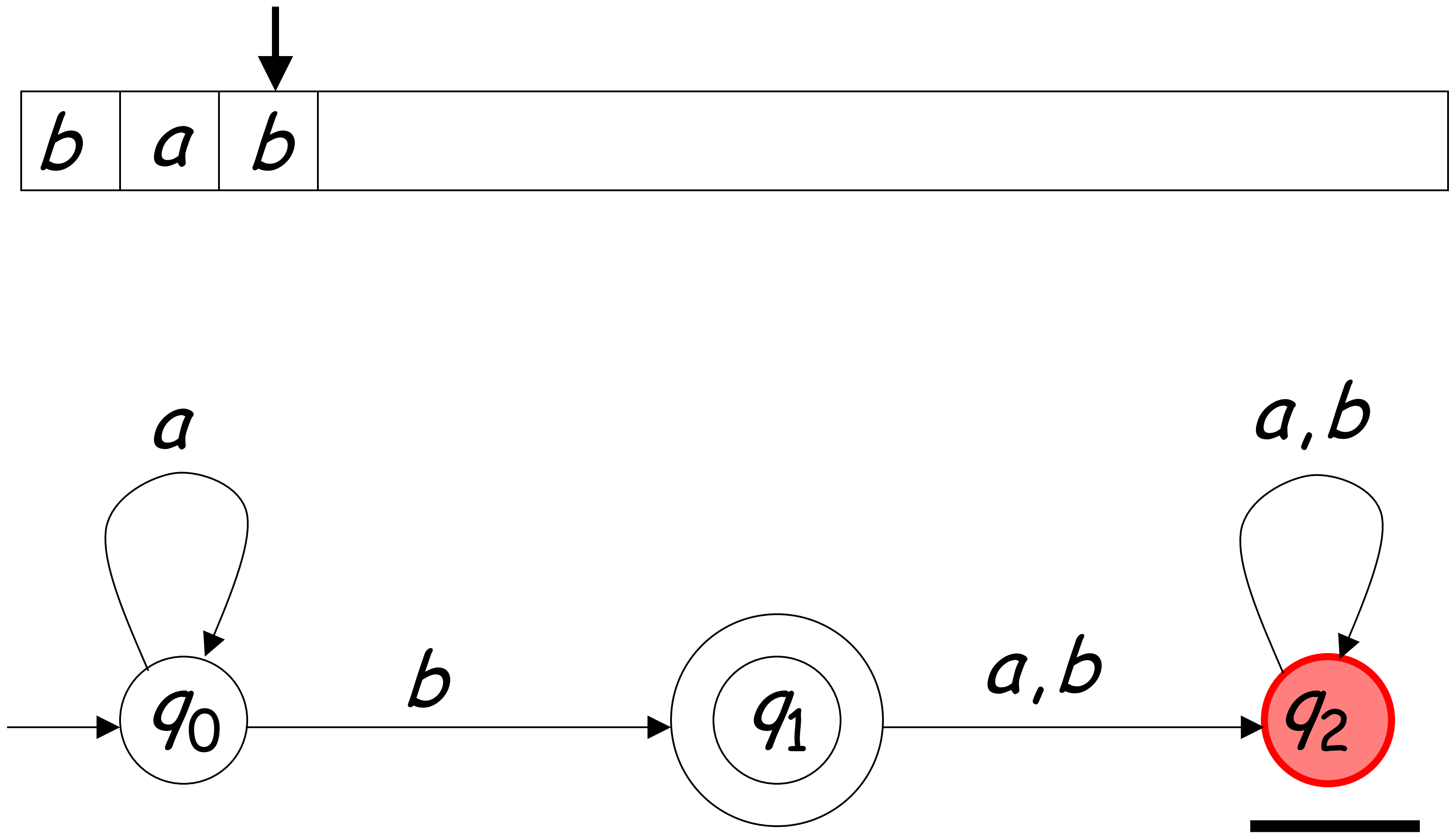
Input String




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

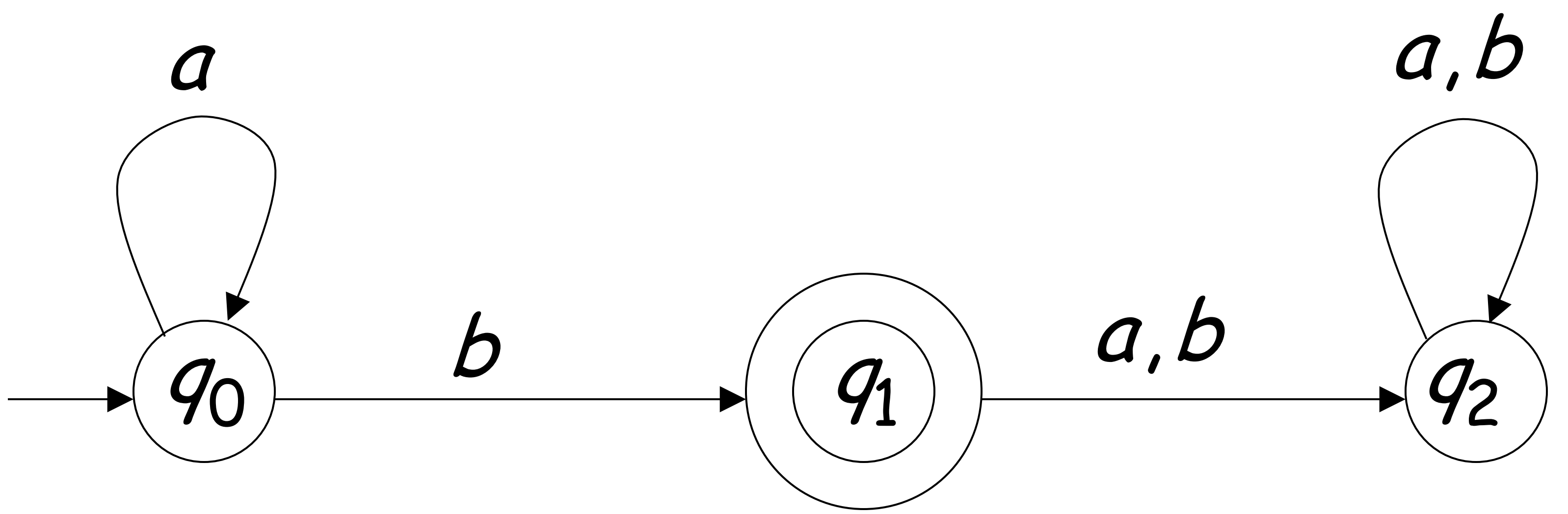

b
a
b
a
a,b
b
a,b
q_0
q_1
q_2

Input finished



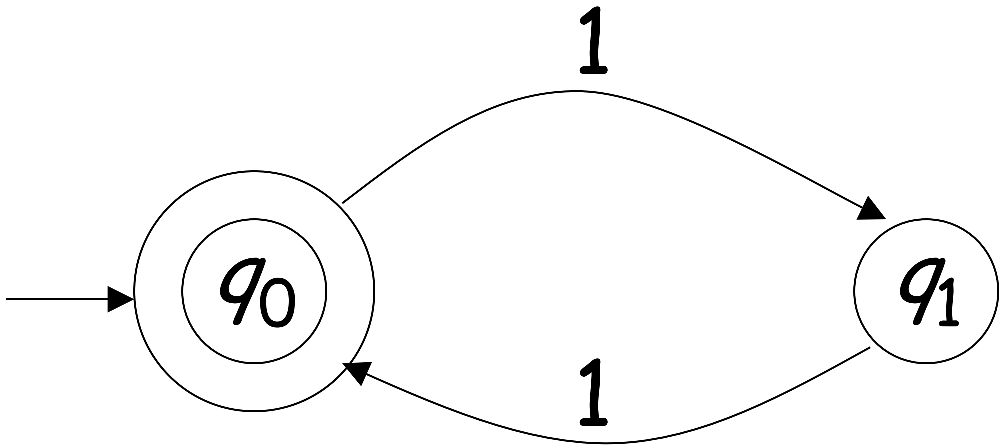
reject

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



Another Example

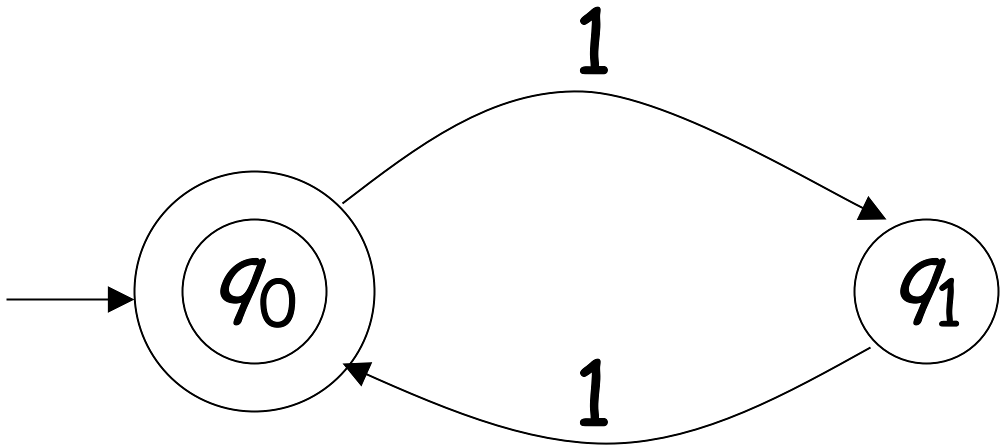
Alphabet: $\Sigma = \{1\}$



Language Accepted: ????????

Another Example

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



Language Accepted:

$$EVEN = \{x : x \in \Sigma^* \text{ and } |x| \text{ is even}\}$$

$$= \{\varepsilon, 11, 1111, 111111, \ldots\}$$

Formal Definition

Deterministic Finite Automaton (DFA)

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

Q : finite set of states

Σ : finite input alphabet $\varepsilon \notin \Sigma$

δ : transition function

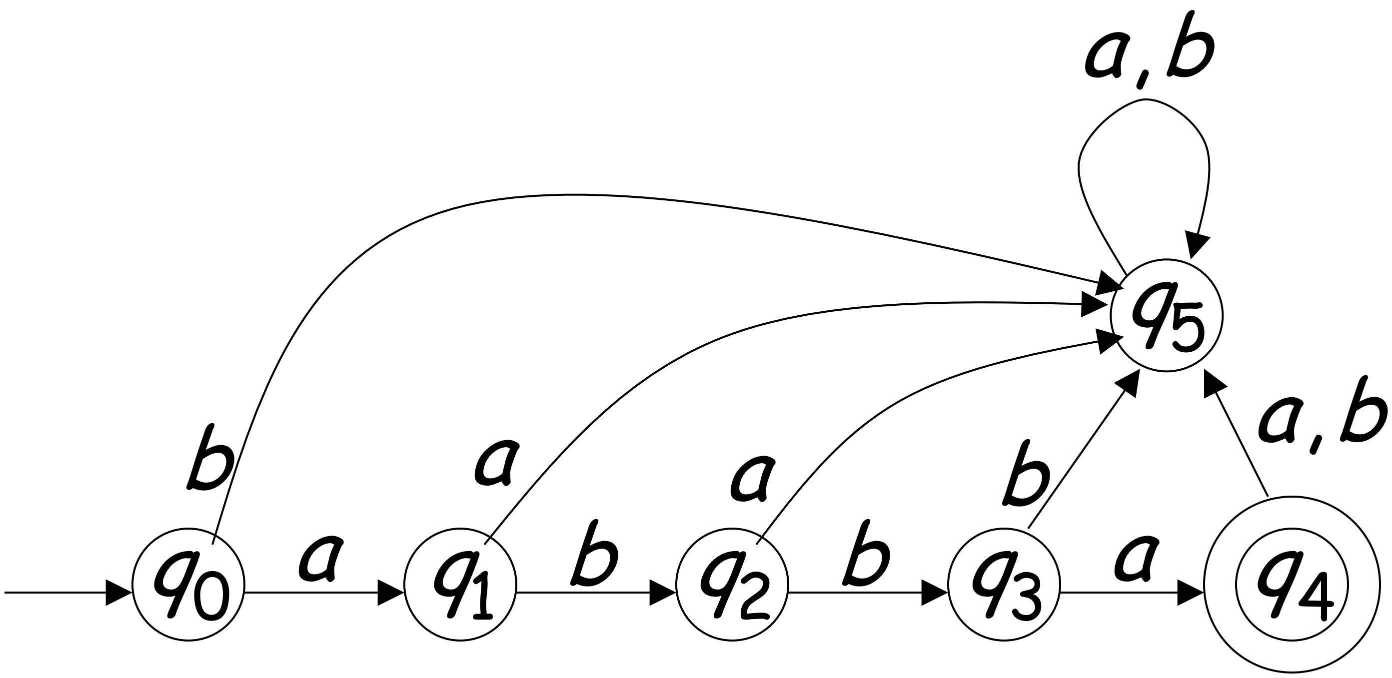
q_0 : initial state

F : set of accepting states

Set of States Q

Example

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

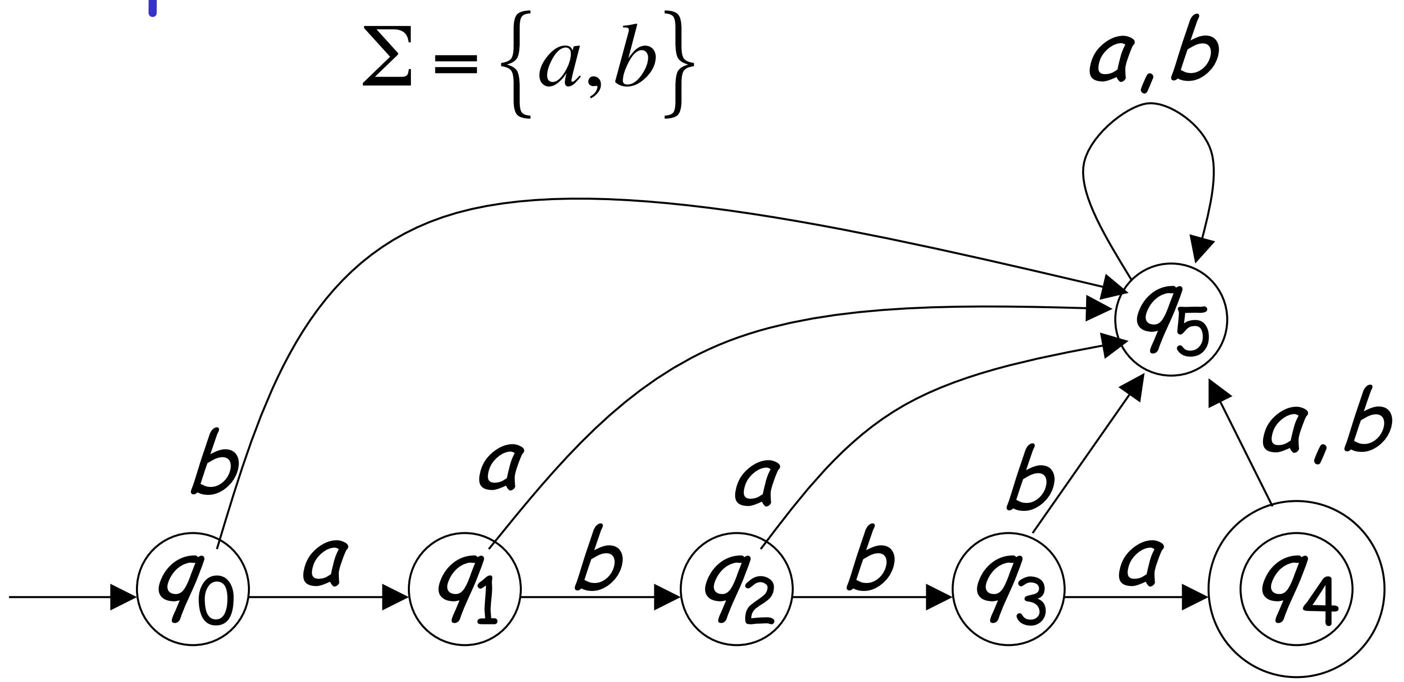


Input Alphabet Σ

$\varepsilon \notin \Sigma$:the input alphabet never contains ε

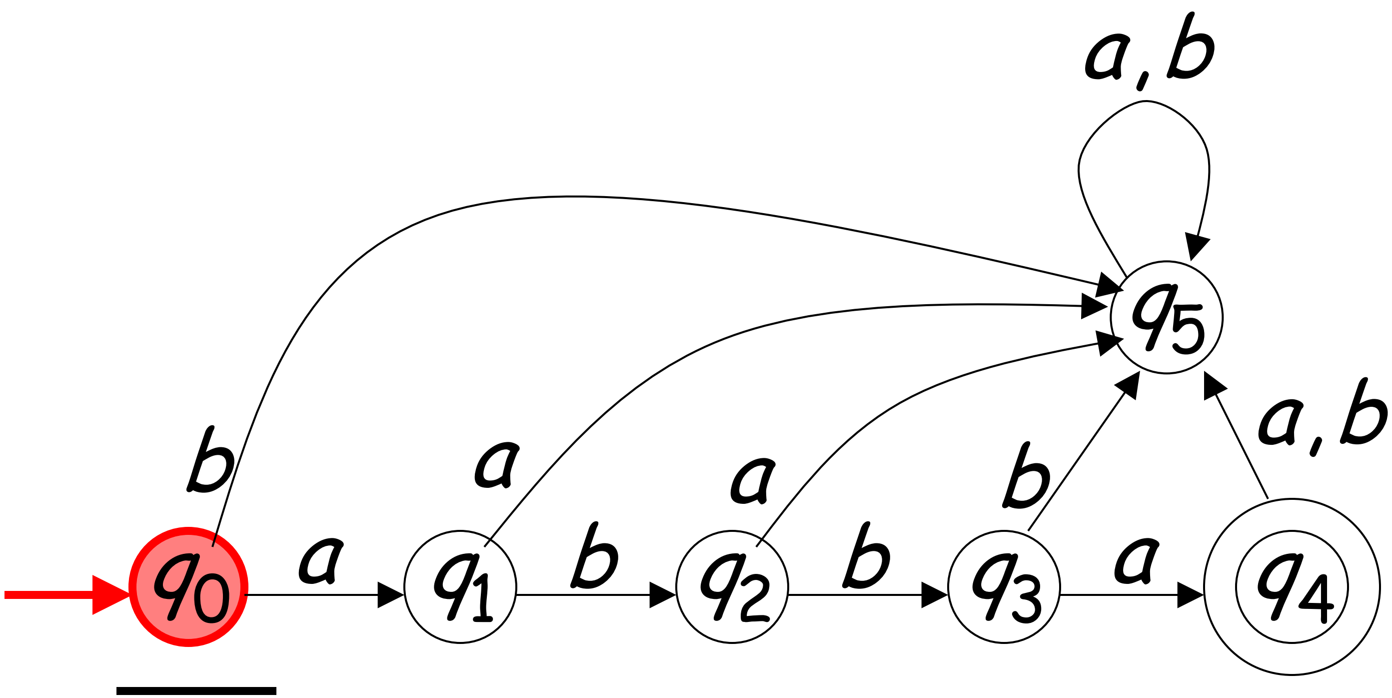
Example

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



Initial State q_0

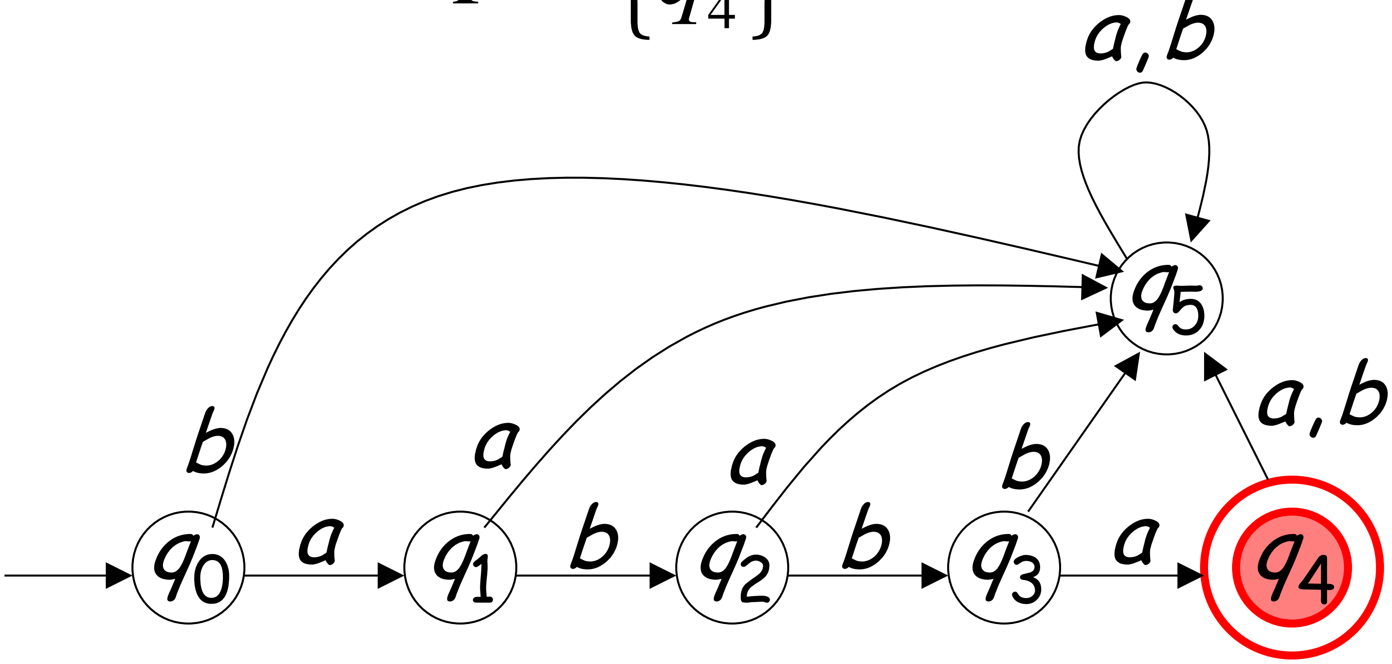
Example



Set of Accepting States $F \subseteq Q$

Example

$$F = \{q_4\}$$



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

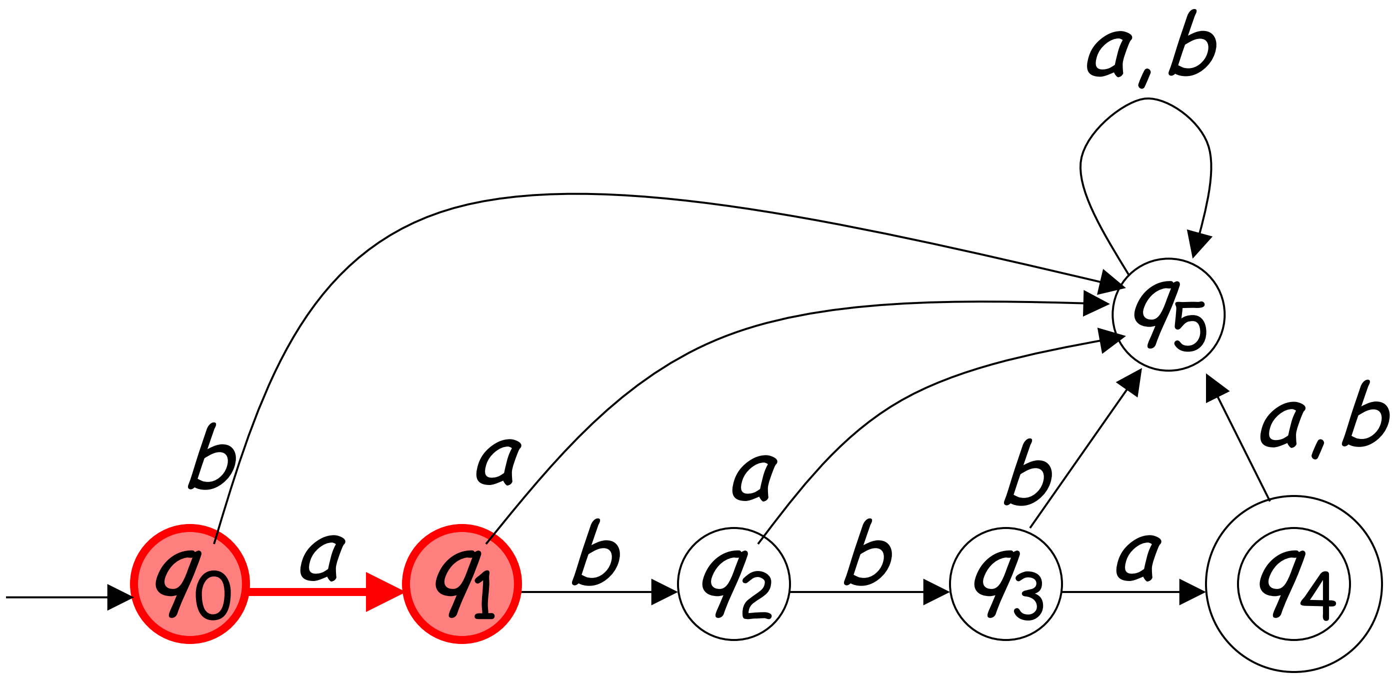
$$\delta(q, x) = q'$$

$$q \xrightarrow{x} q'$$

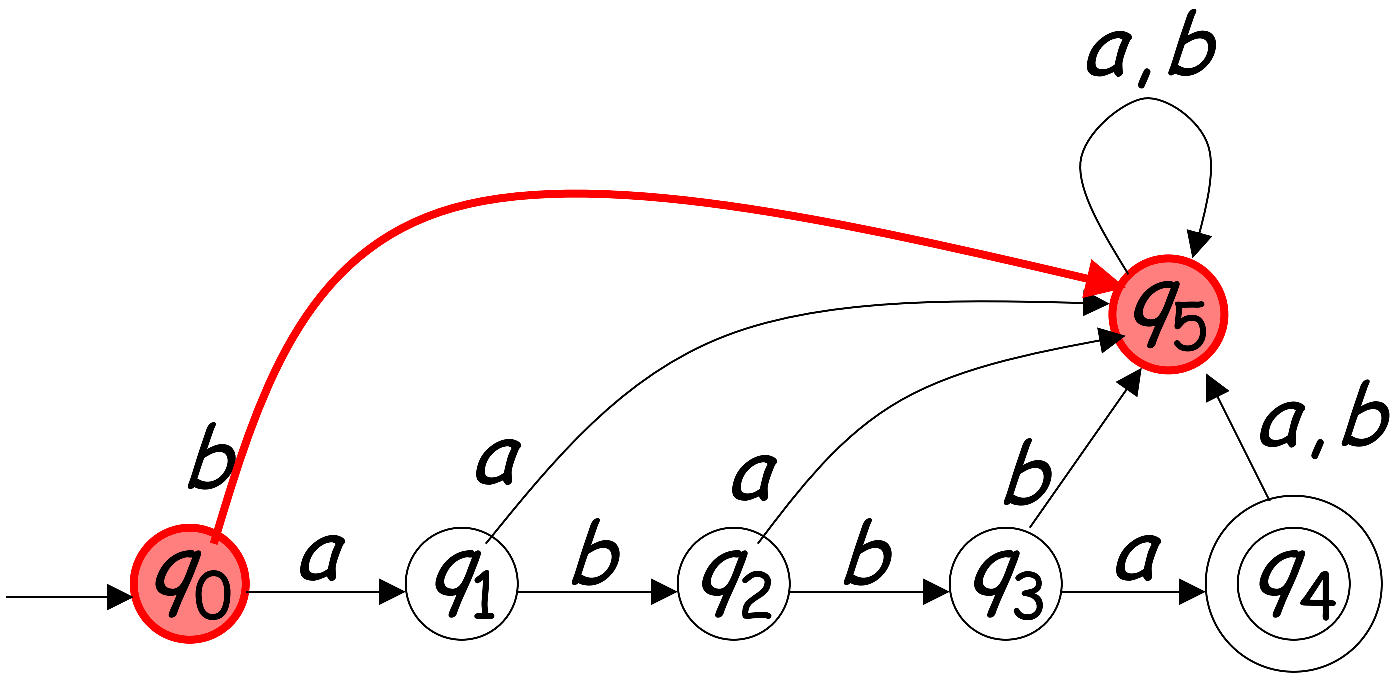
Describes the result of a transition from state q with symbol x

Example:

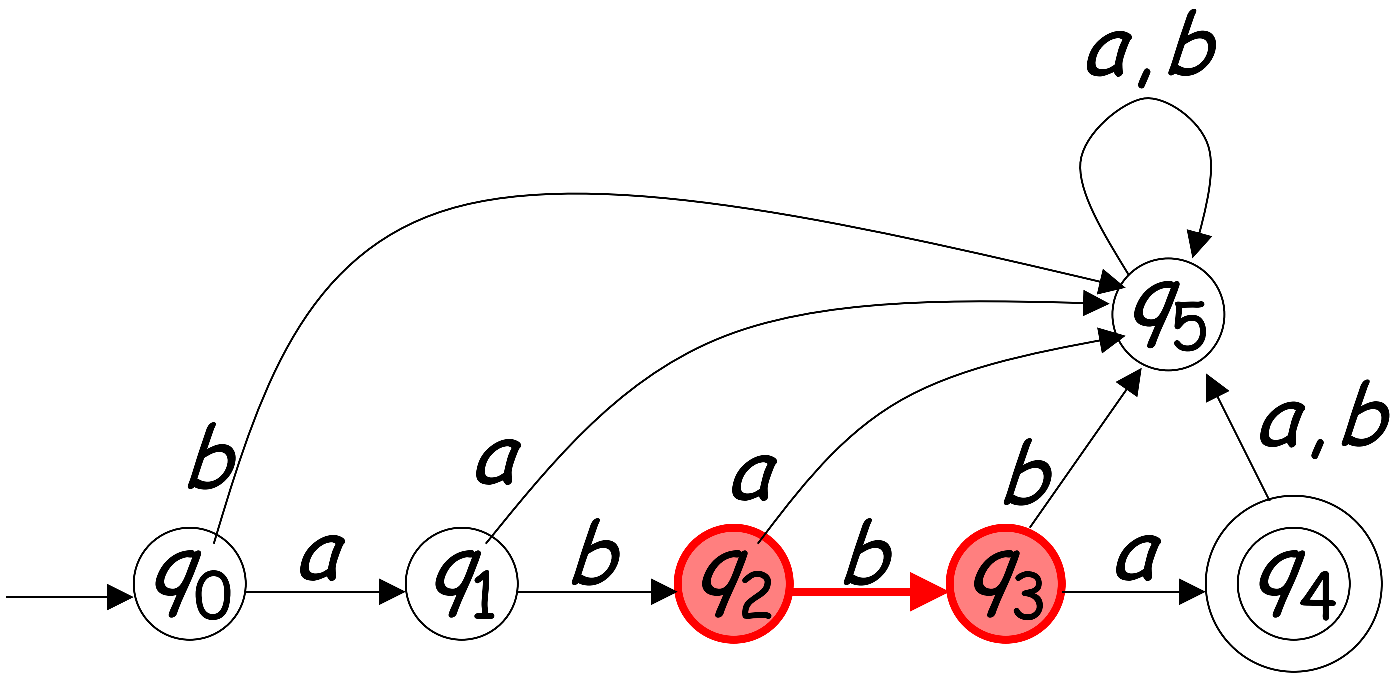
$$\delta(q_0, a) = q_1$$



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



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

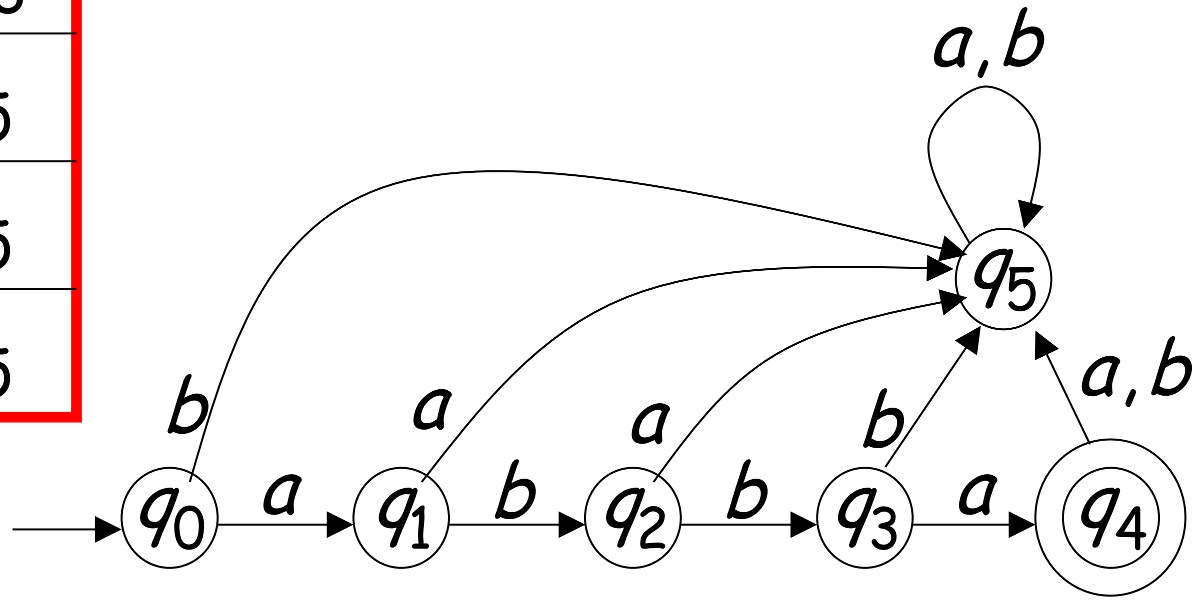


Transition Table for δ

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

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

$$\hat{\delta}(q, w) = q'$$

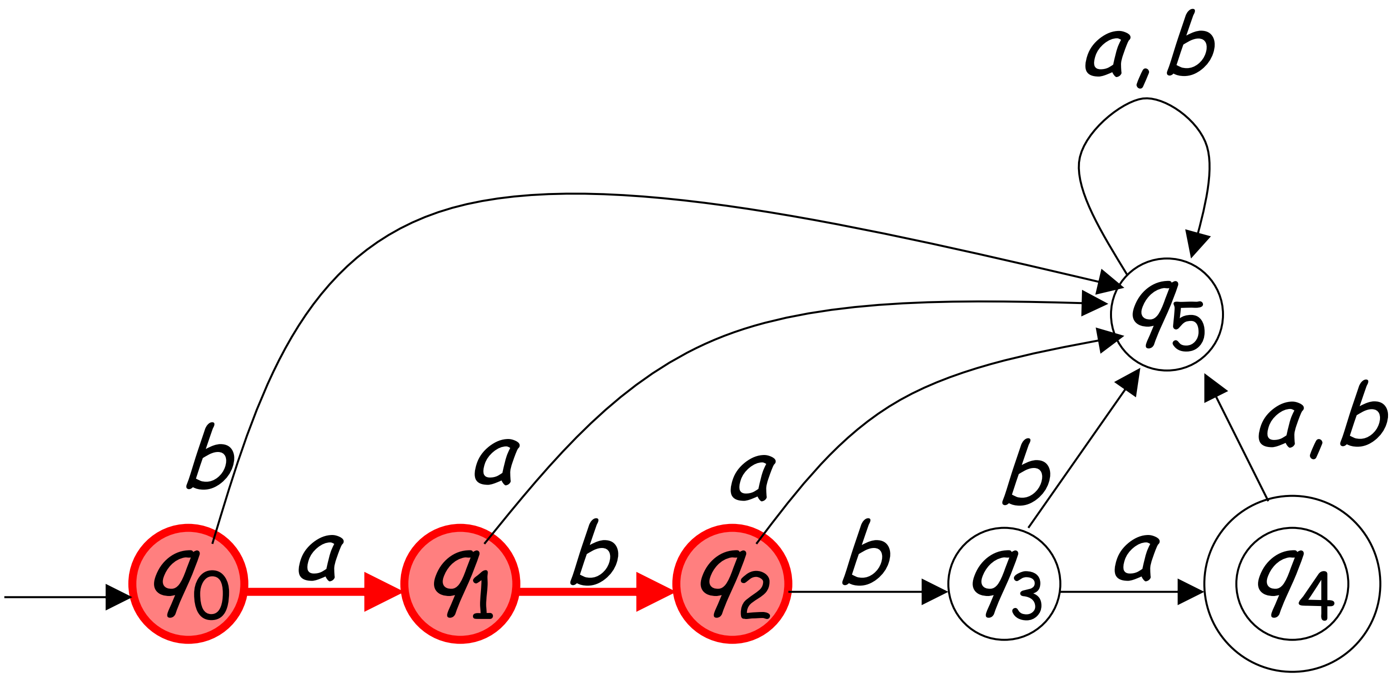
Describes the resulting state
after scanning string w from state q

Definition of $\hat{\delta}$:

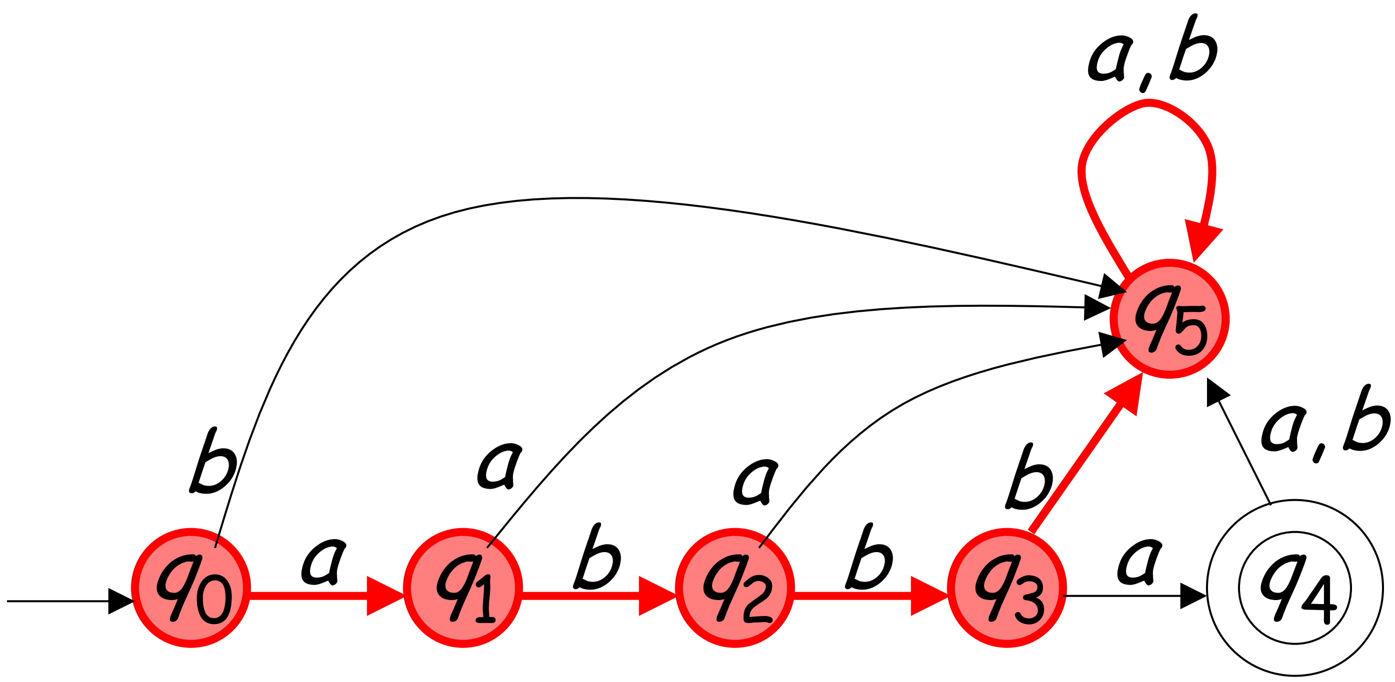
1 $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

2 $\hat{\delta}\left(q, \varepsilon\right) = q$

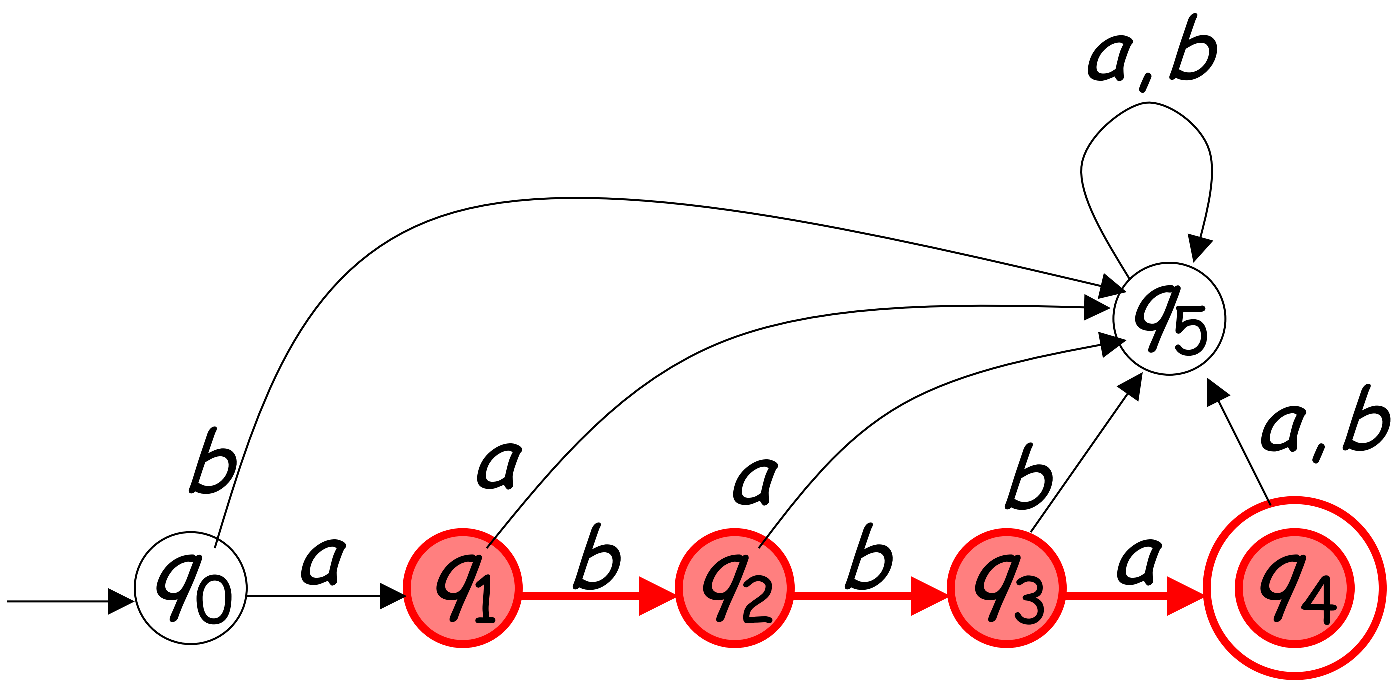
Example: $\hat{\delta}(q_0, ab) = q_2$



$$\hat{\delta}(q_0, abbbaa) = q_5$$



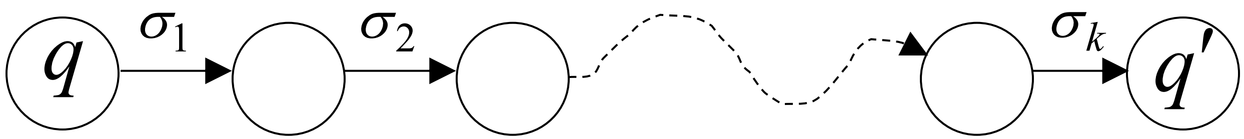
$$\hat{\delta}(q_1, bba) = q_4$$



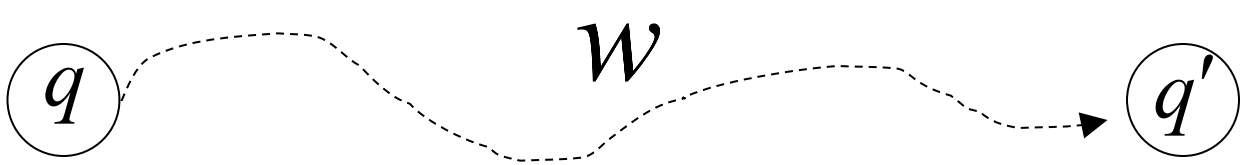
In general: $\hat{\delta}(q, w) = q'$

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 :

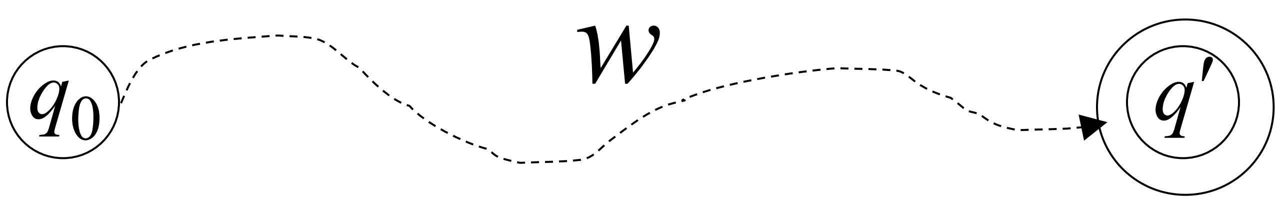
it is denoted as $L(M)$ and contains all the strings accepted by M

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 :

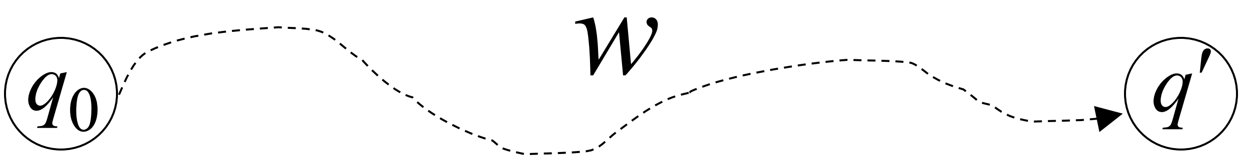
$$L(M) = \left\{ w \in \Sigma^* : \hat{\delta}(q_0, w) \in F \right\}$$



$q' \in F$

Language rejected by M:

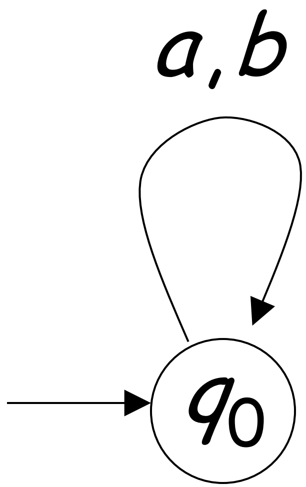
$$\overline{L(M)} = \left\{ w \in \Sigma^* : \hat{\delta}(q_0, w) \notin F \right\}$$

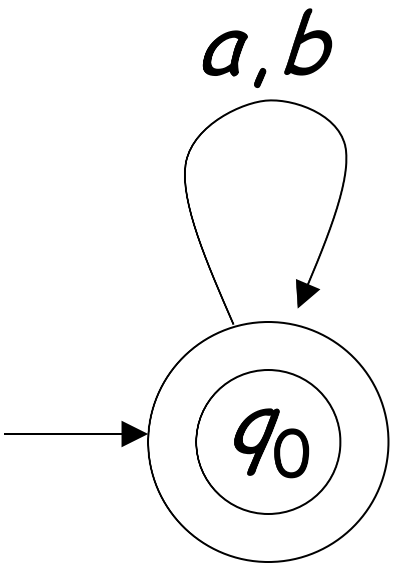


$$q' \notin F$$

More DFA Examples

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





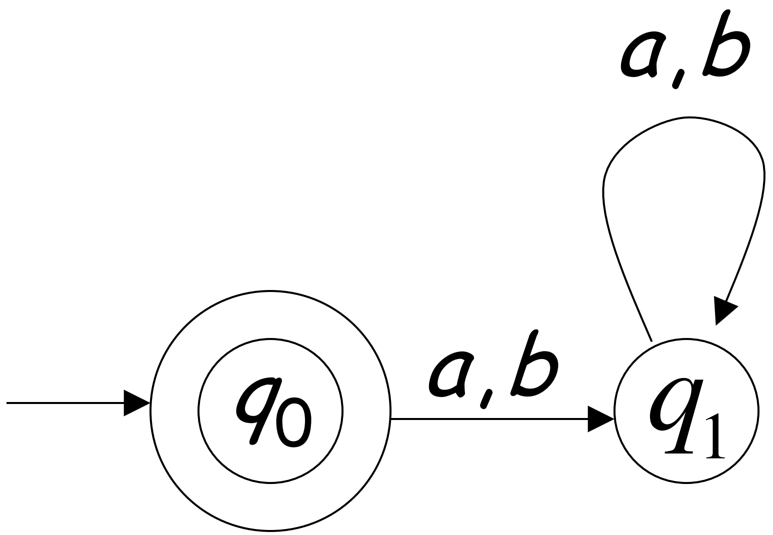
$$L(M) = \{\ \}$$

Empty language

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

All strings

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

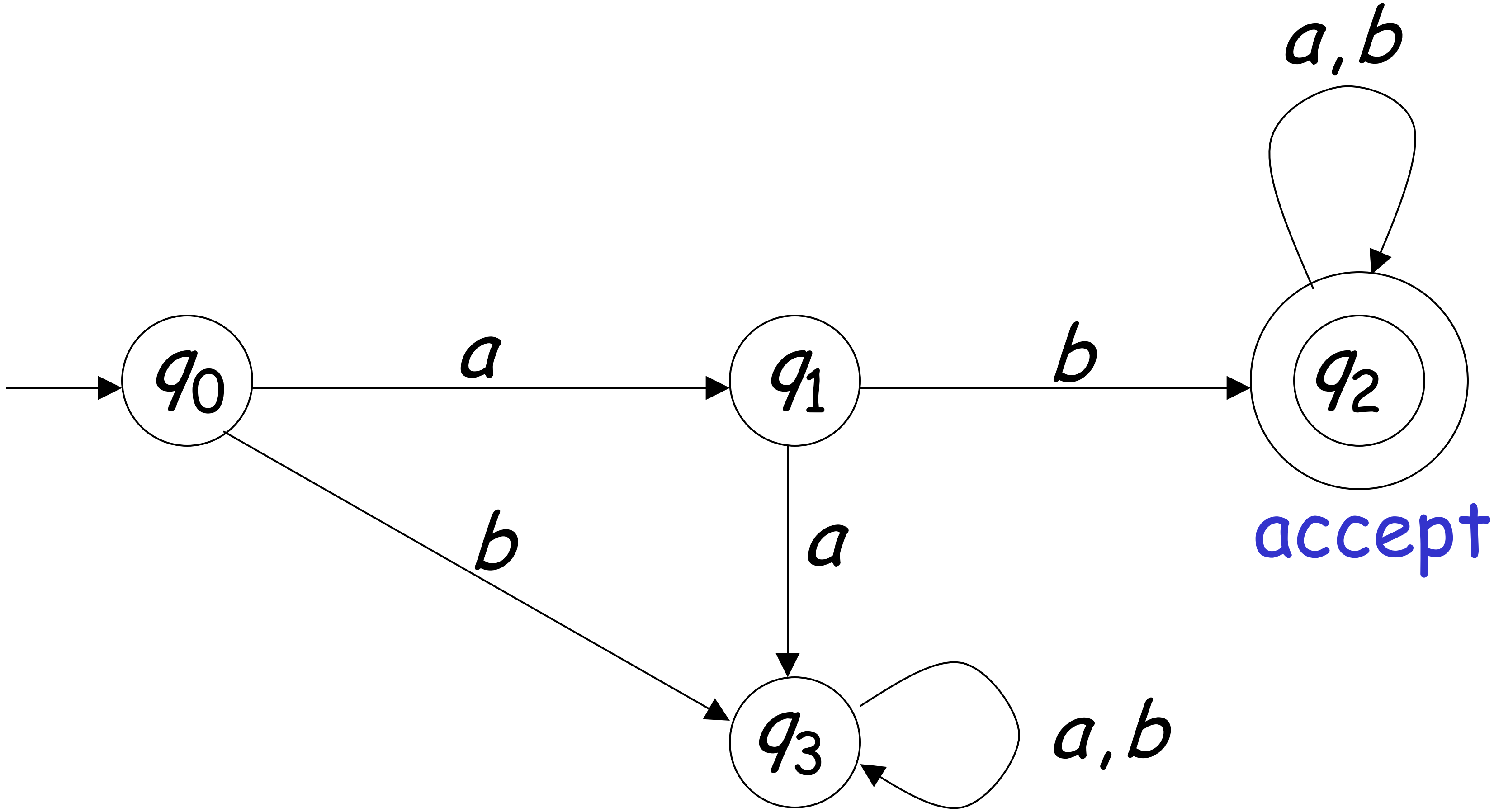


$$L(M) = \{\varepsilon\}$$

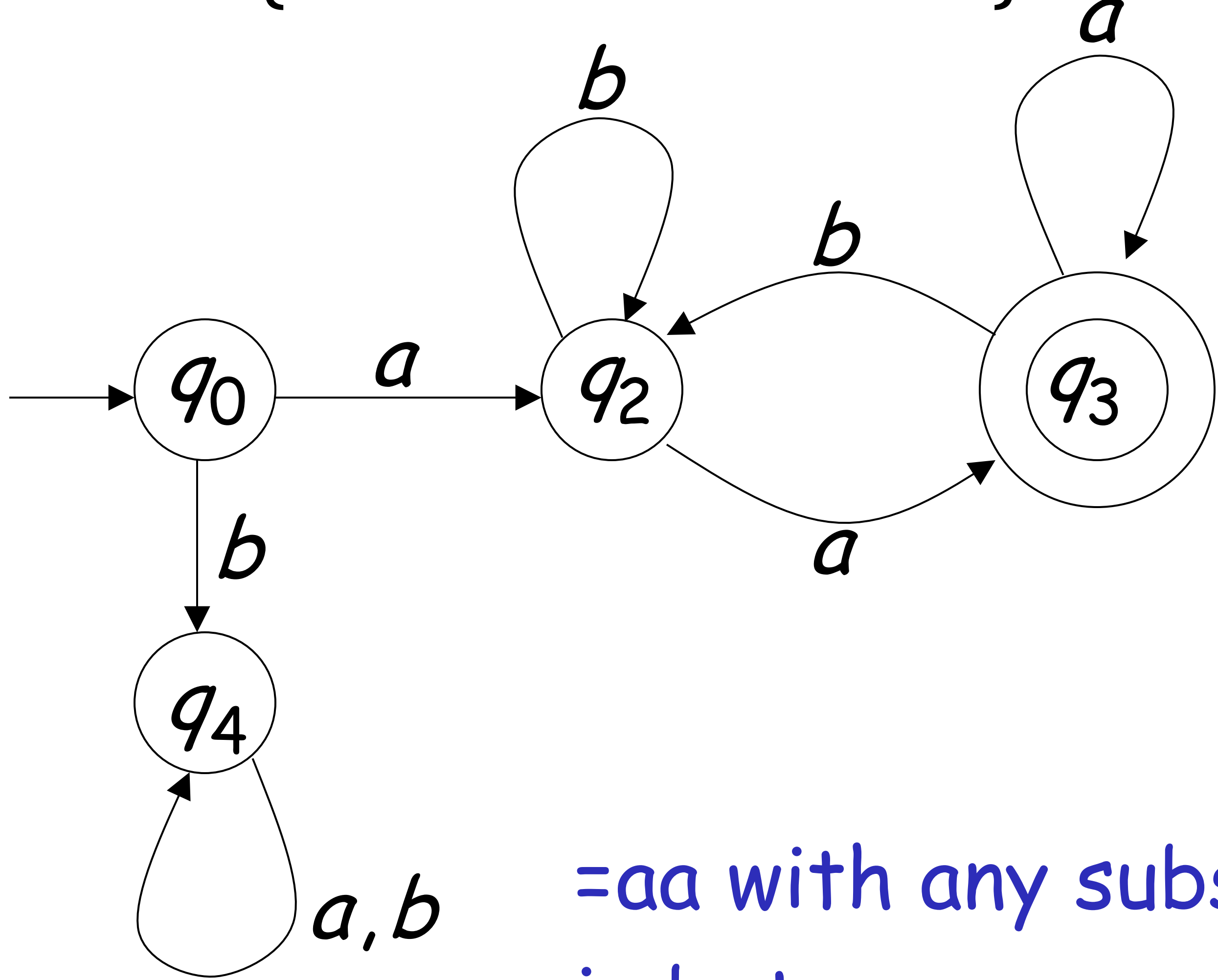
Language of the empty string

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

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



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



=aa with any substring in between

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:

$\{abba\}$ $\{\varepsilon, ab, abba\}$

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

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

{ all binary strings without substring 001}

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

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

There exists a DFA that accept each of these languages

There exist languages which are not Regular:

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

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

There is no DFA that accepts these languages

(we will prove this in a later class)

Why define a language?

- We might want to formally define a language for:
  - Natural language (to allow eg automatic analysis of text).
  - A programming language (we need to do this!)
- Finite Automata are not quite expressive enough - we will extend them later.

Why define a language?

Another reason:

- The letters in the alphabet might model events (inputs and outputs).
- A language could then model system behaviour.

- Remember – eg Statecharts and their use for embedded systems.

Why use finite automata?

There are more expressive formalisms but:

- FA are simpler to use/understand.
- FA are simpler to analyse (eg it is feasible to analyse large FA).

- We will see:
  - As formalisms become more expressive, analysis becomes more difficult.