Deterministic

Finite Automata

And Regular Languages

Deterministic

Finite Automata

Simple Automaton

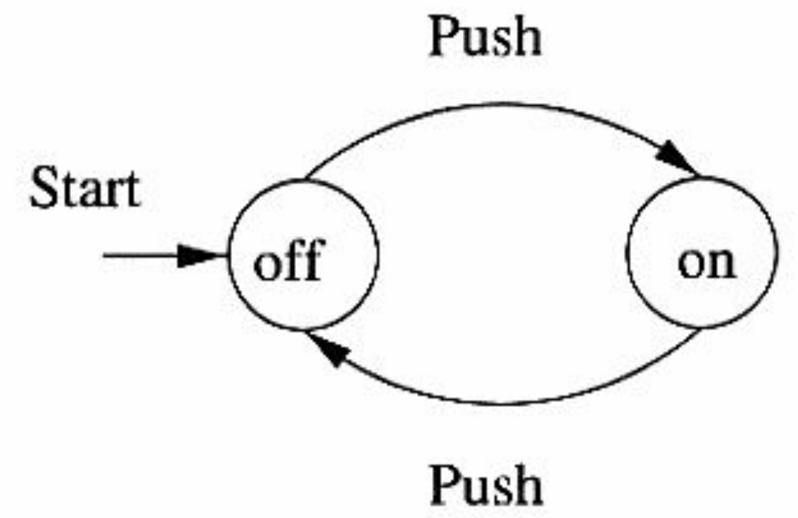


Figure 1.1: A finite automaton modeling an on/off switch

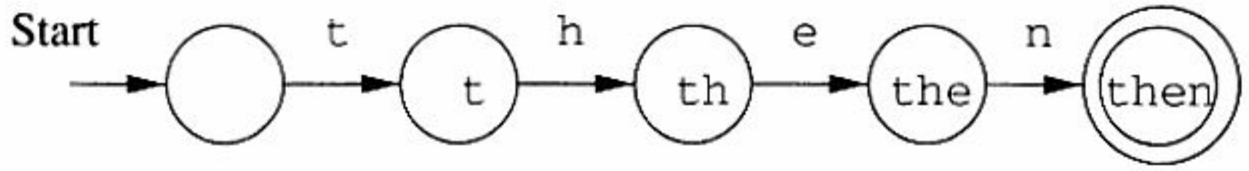
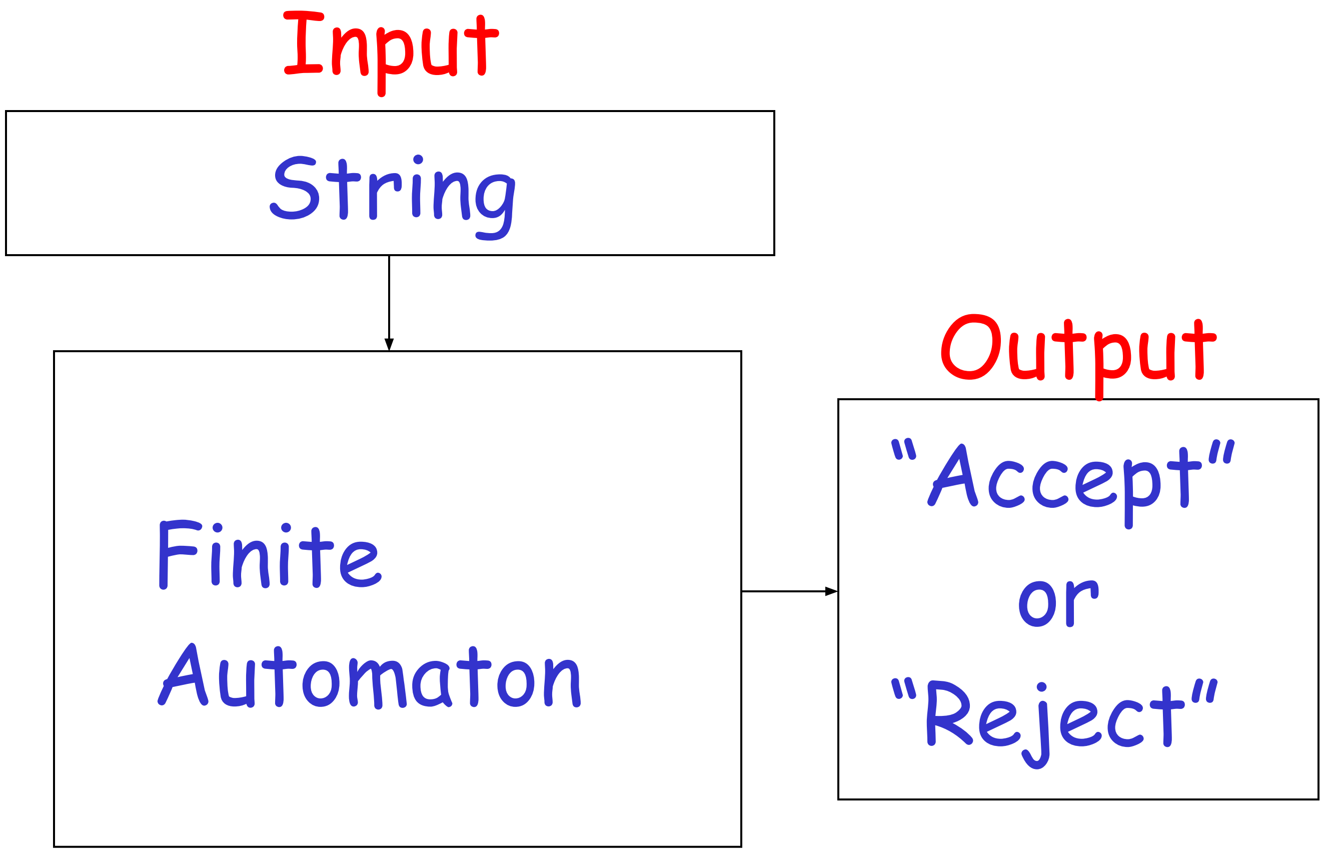


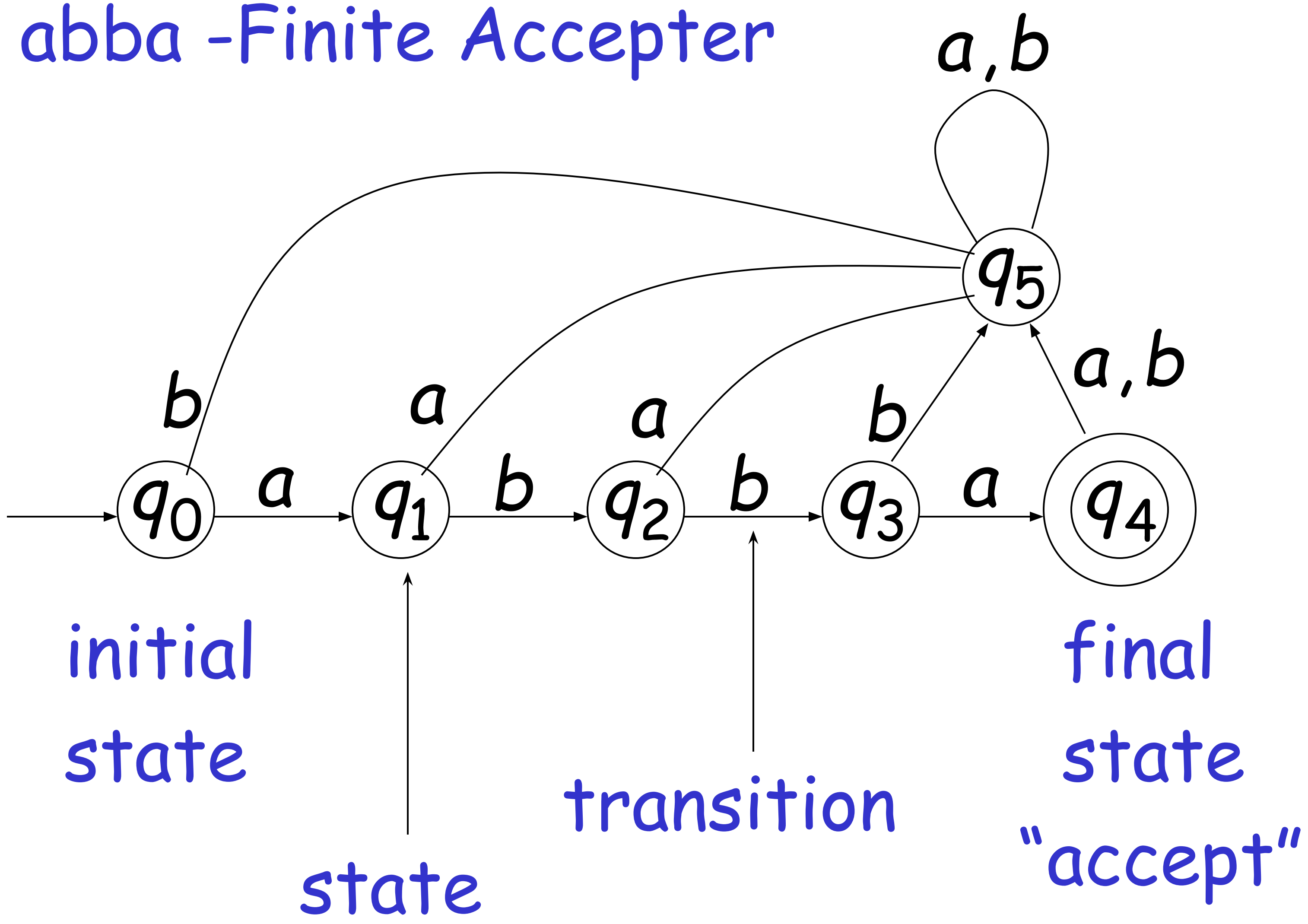
Figure 1.2: A finite automaton modeling recognition of **then**

Finite Accepter

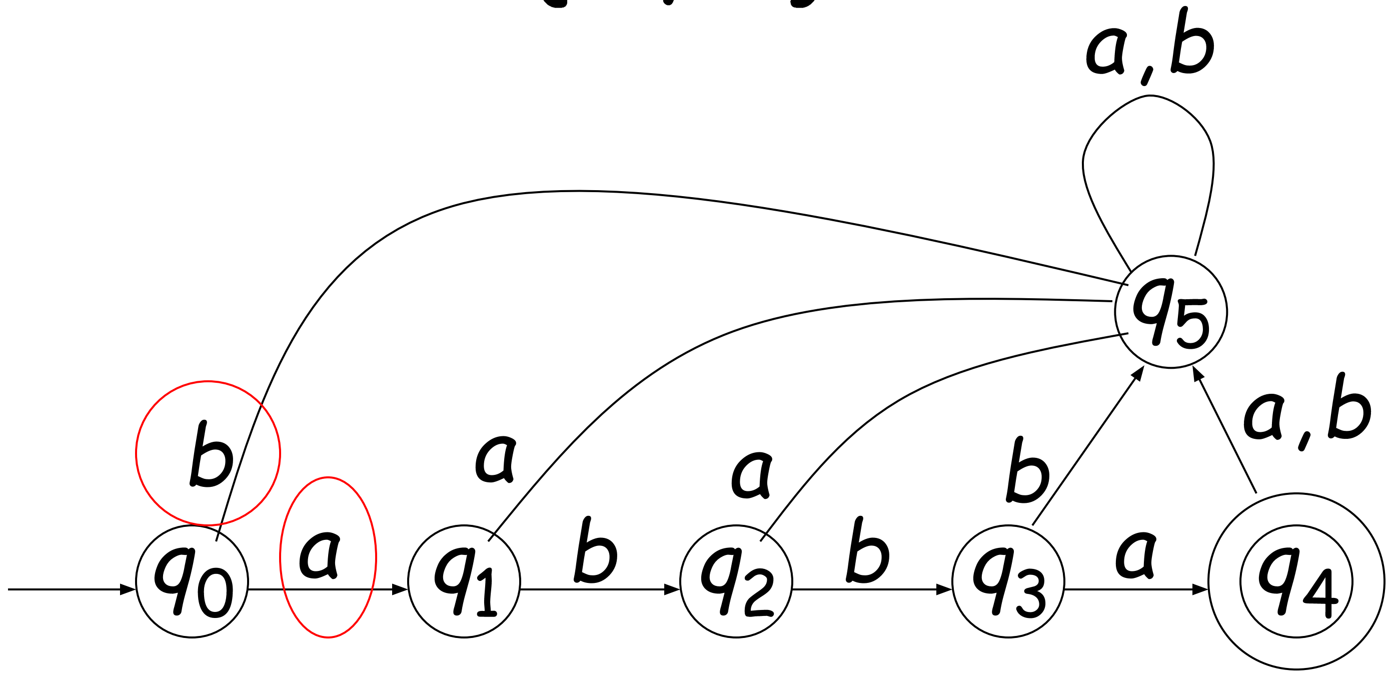


Transition Graph

abba -Finite Accepter

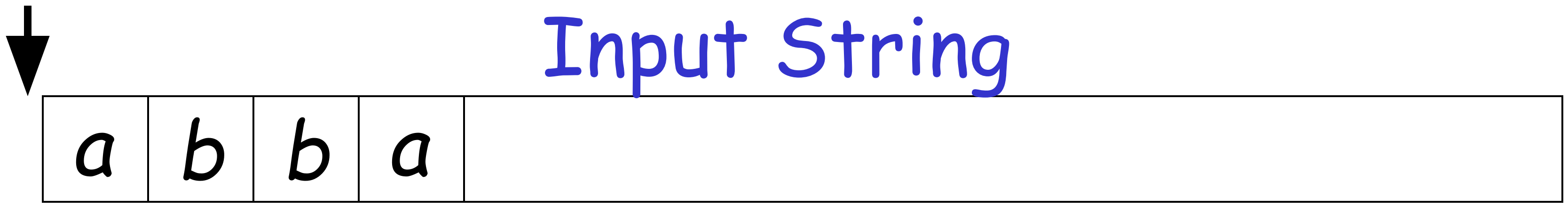


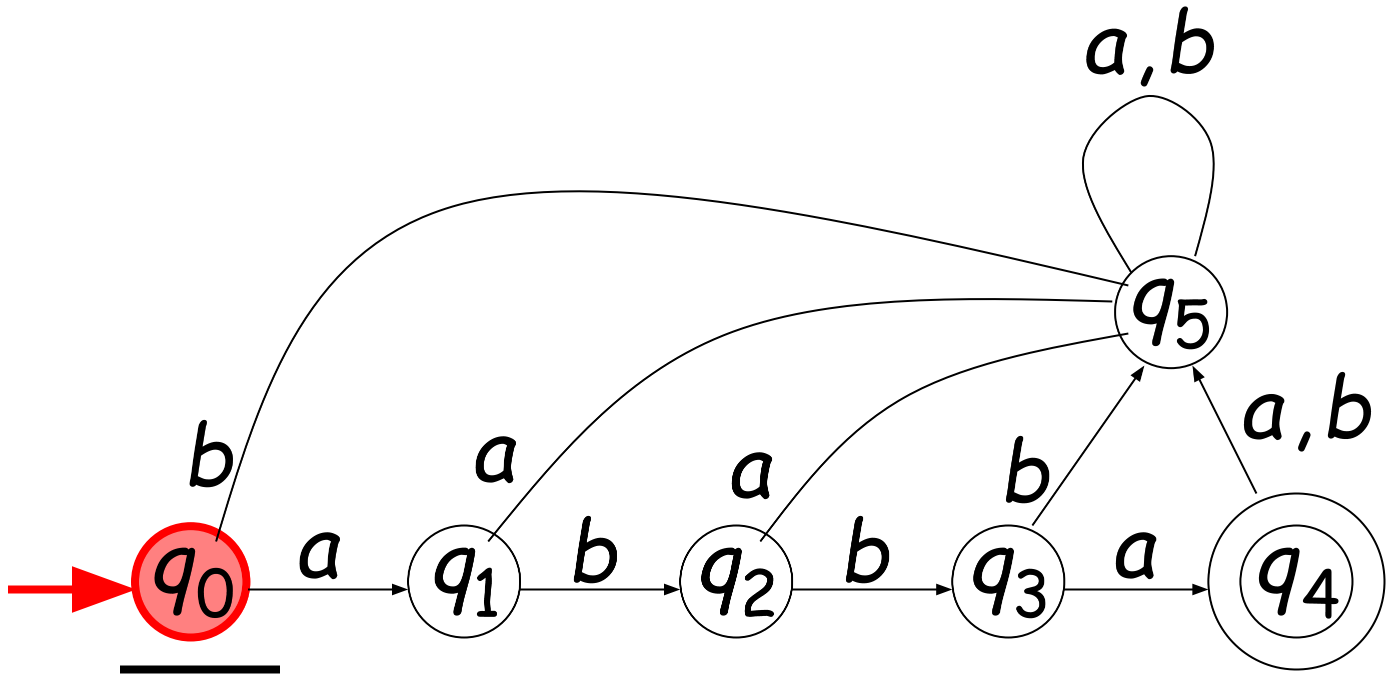
Alphabet $\Sigma = \{a, b\}$



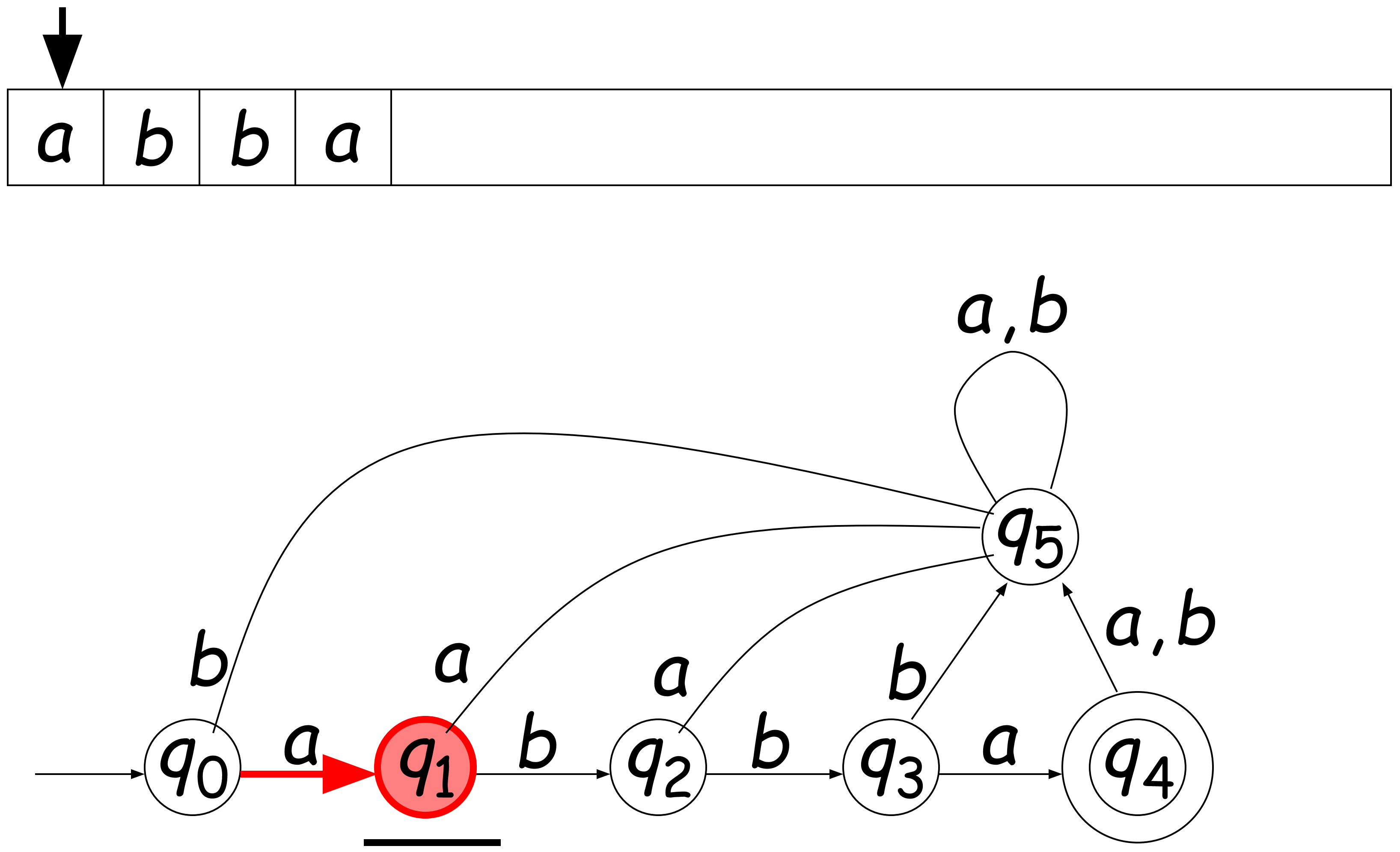
For <u>every</u> state, there is a transition for <u>every</u> symbol in the alphabet

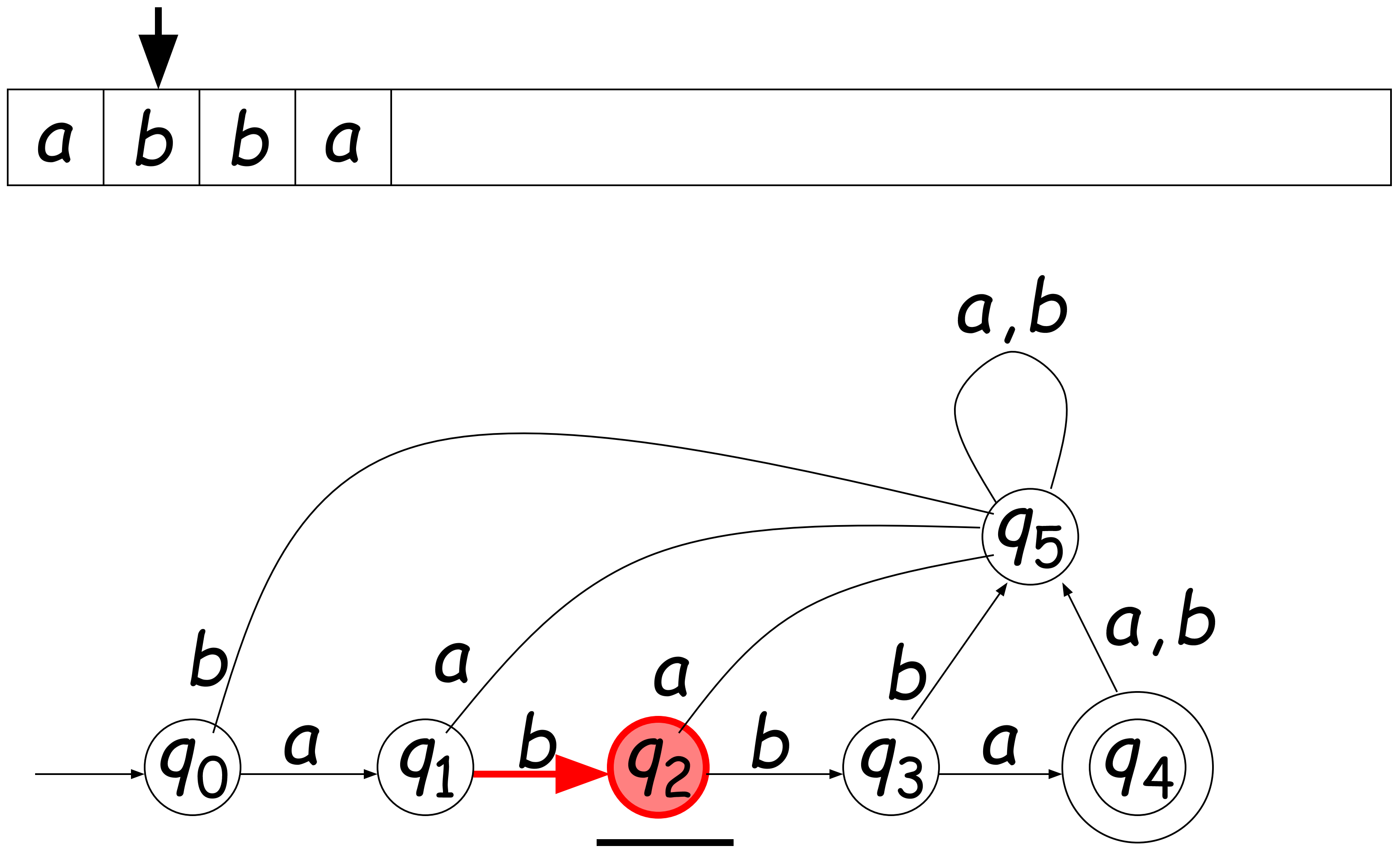
Initial Configuration



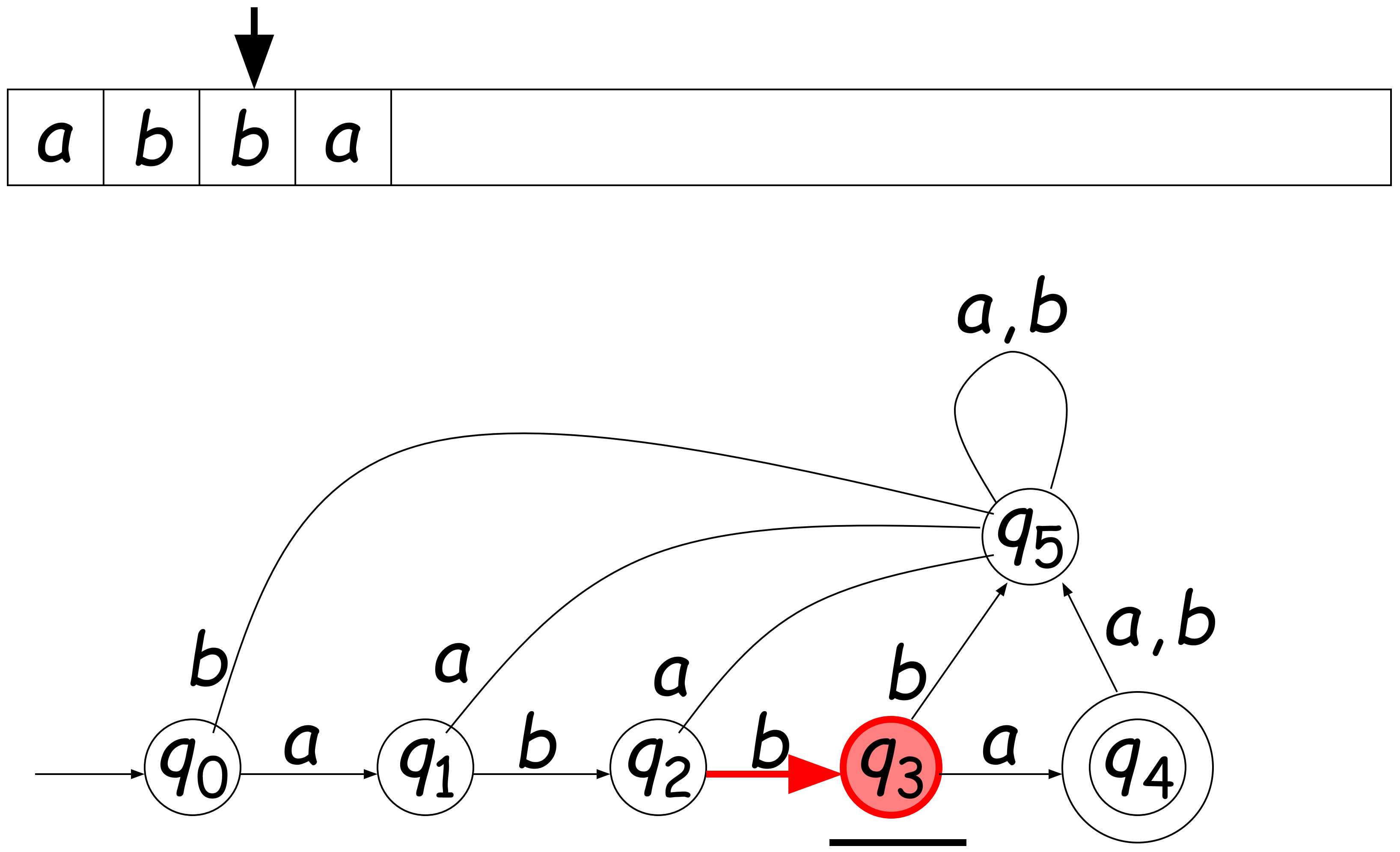


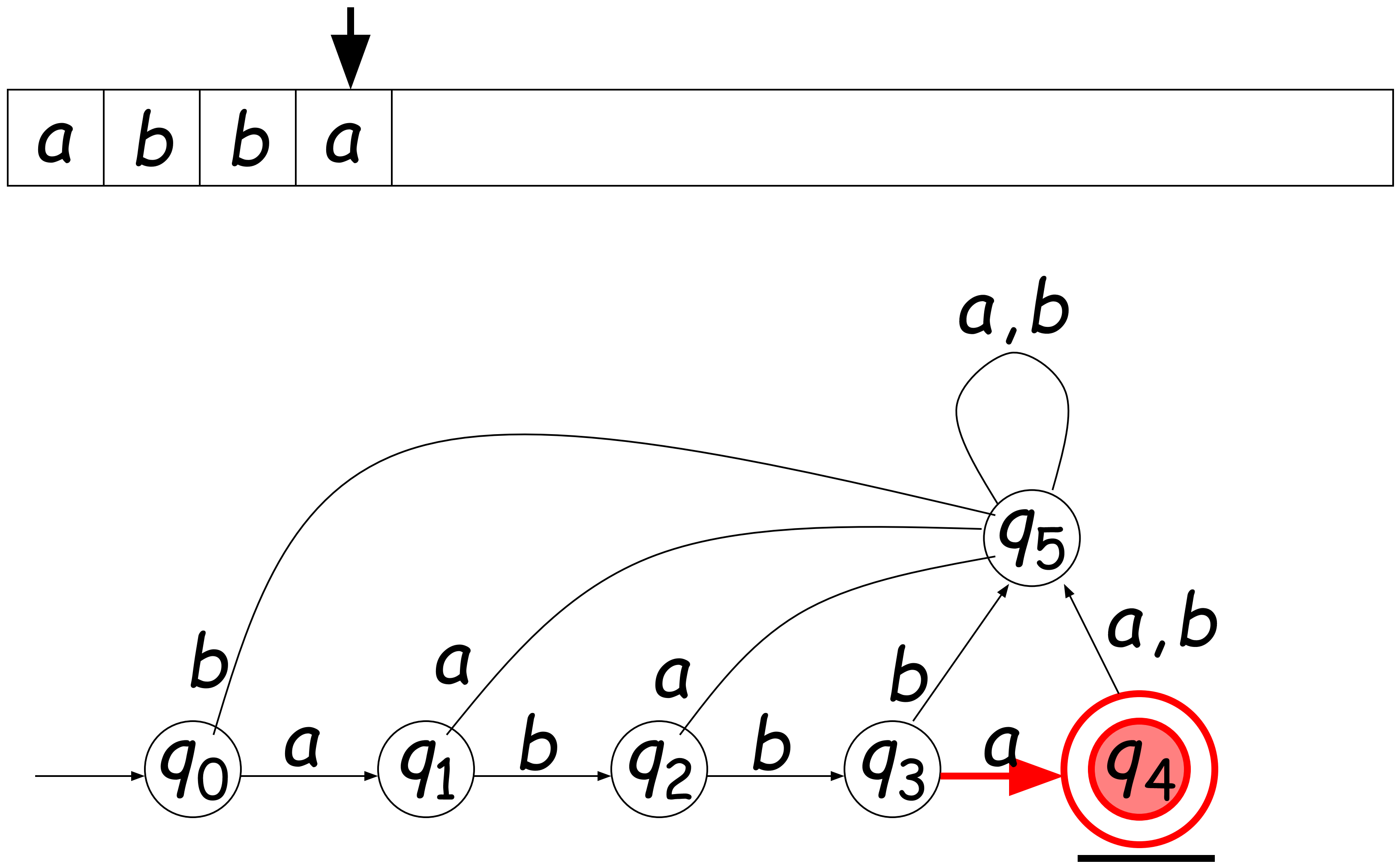
Reading the Input



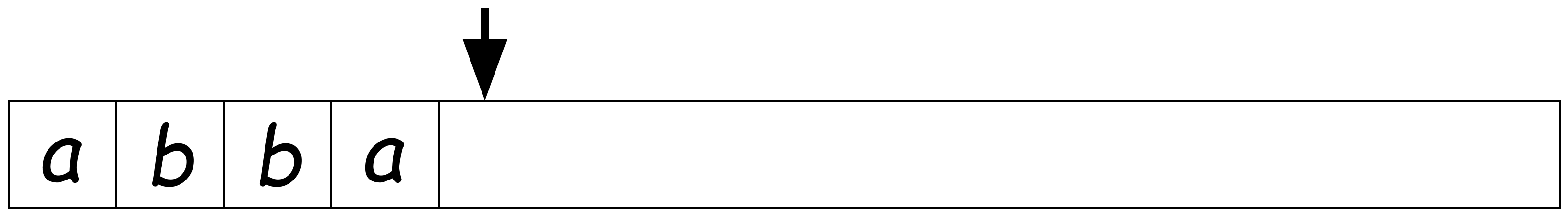


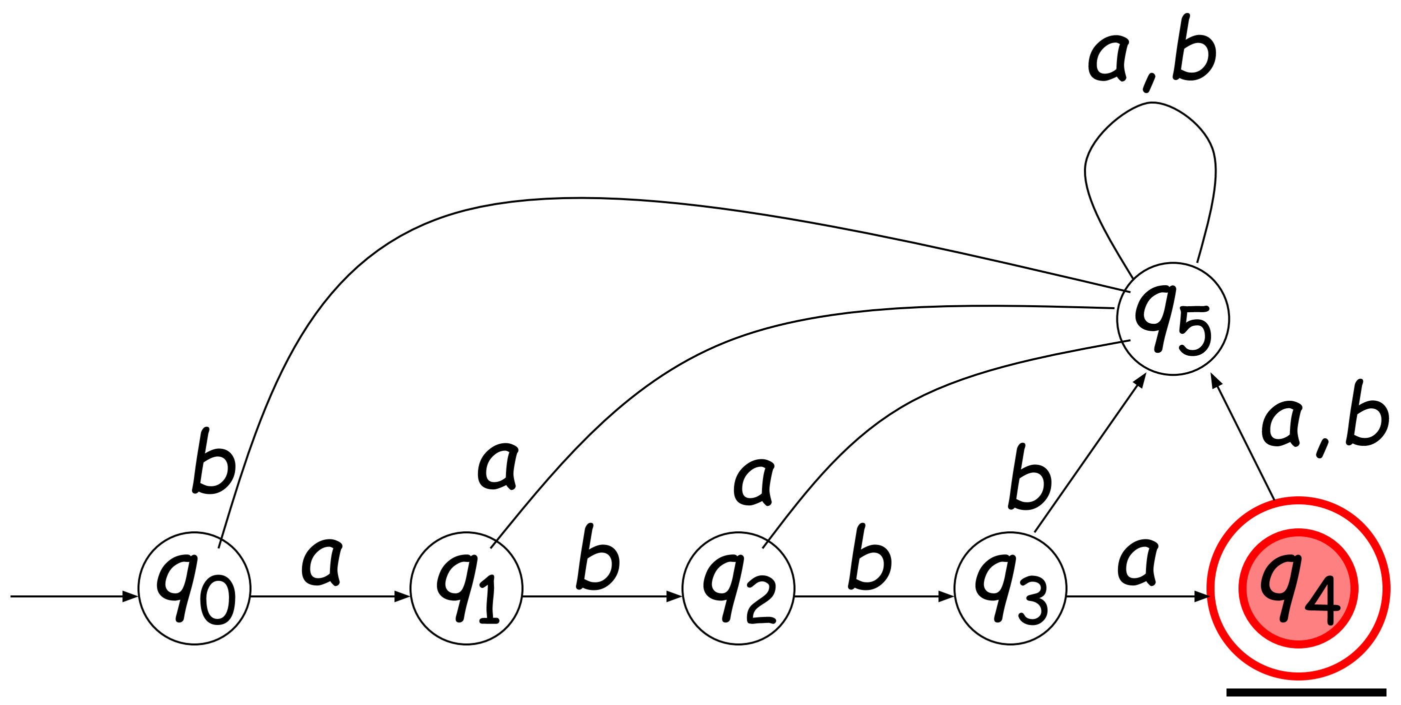
a
b
b
a
a,b
b
a
a
b
a,b
b
a
b
b
a


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


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

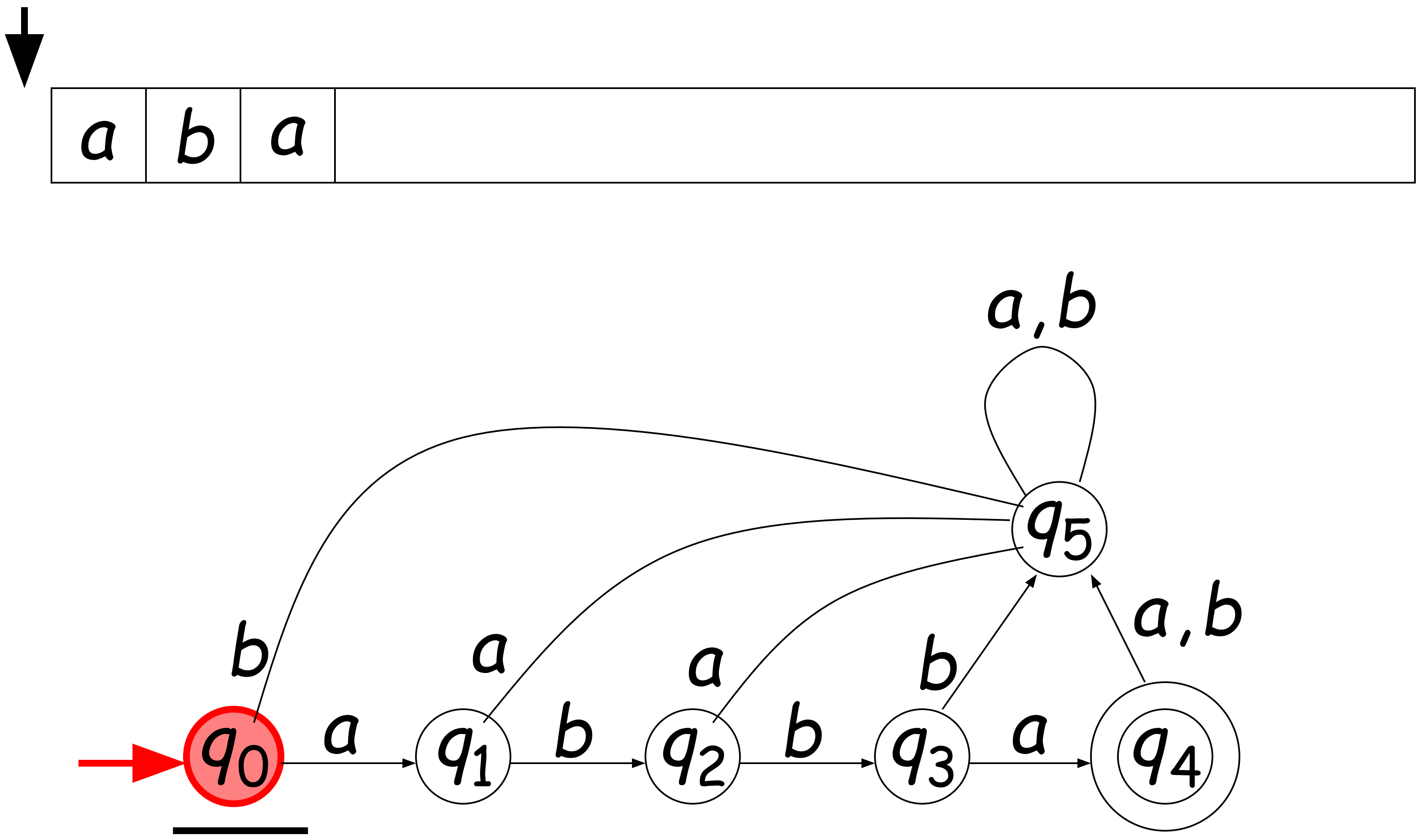
Input finished

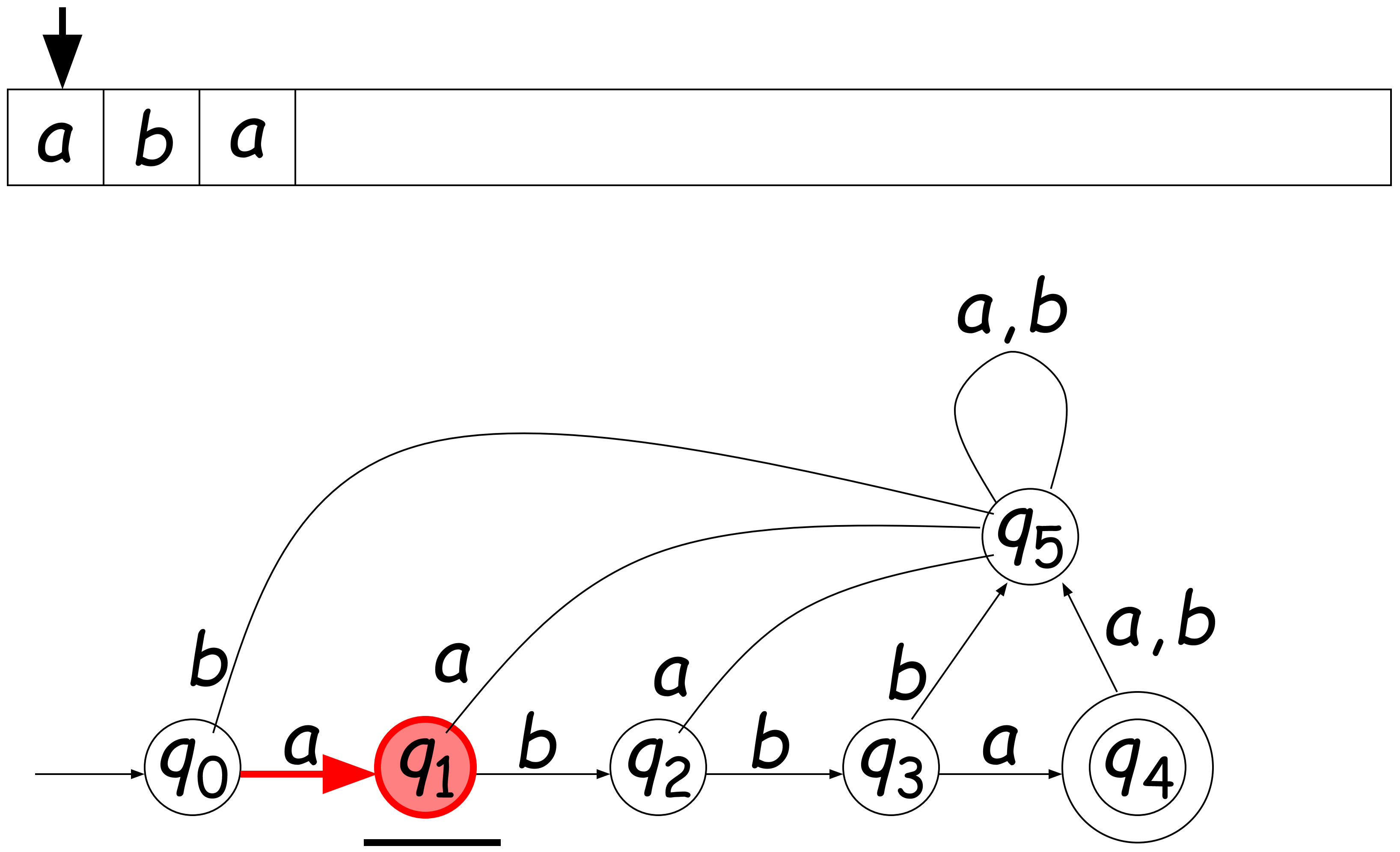


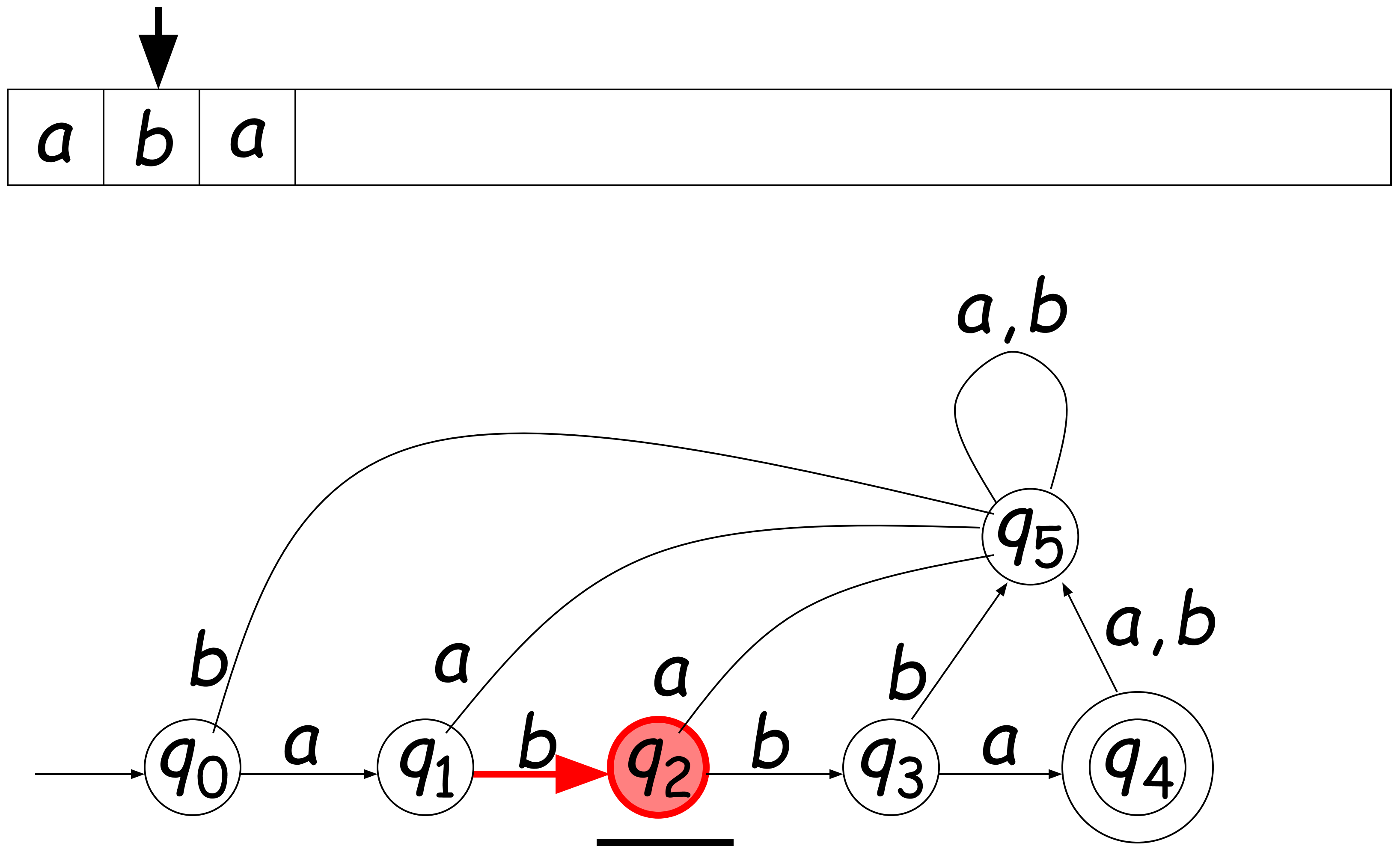


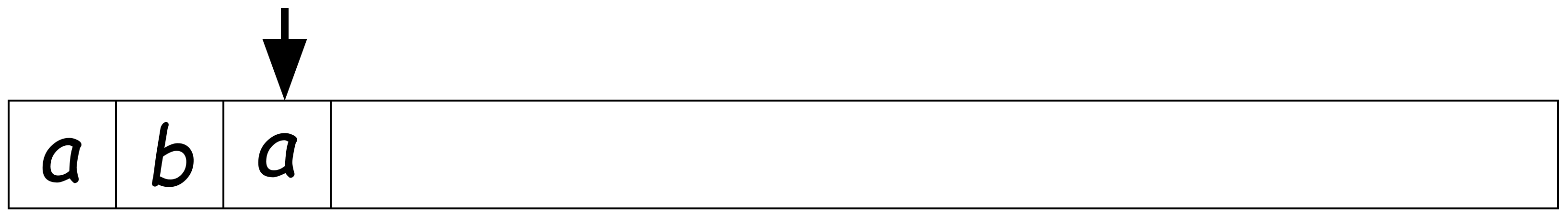
Output: "accept"

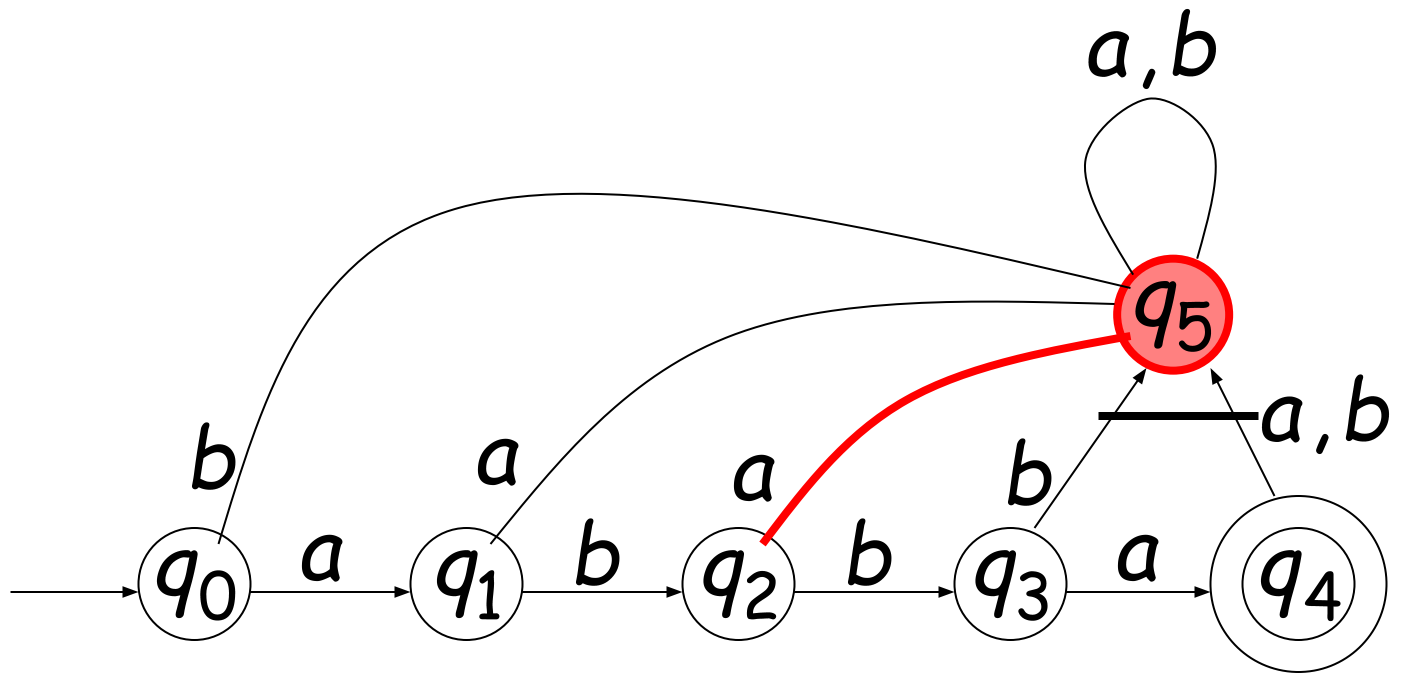
Rejection



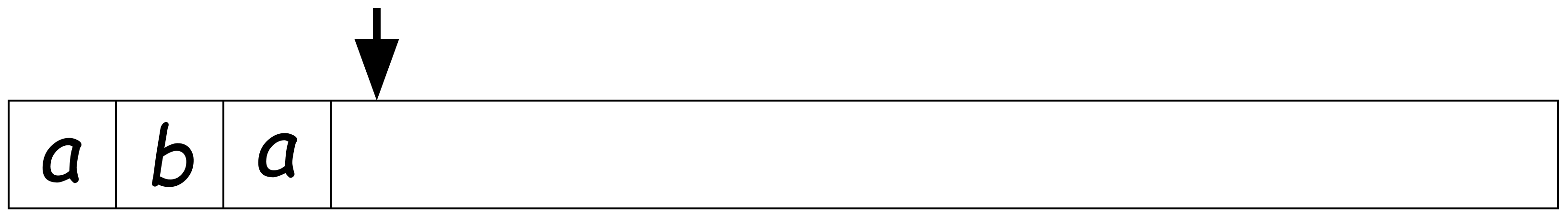

a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

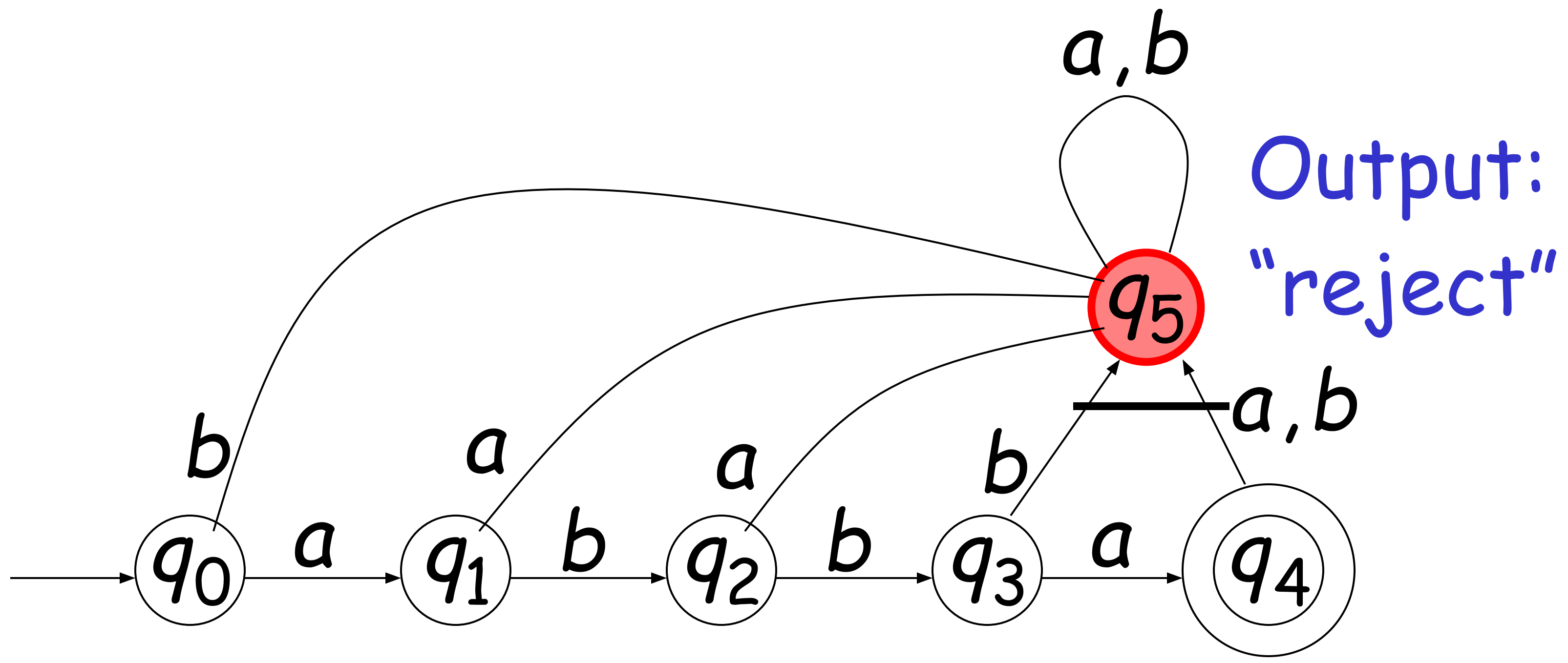

a
b
a
a, b
b
a
a
b
b
a
a, b
q0
q1
q2
q3
q4
q5


a
b
a

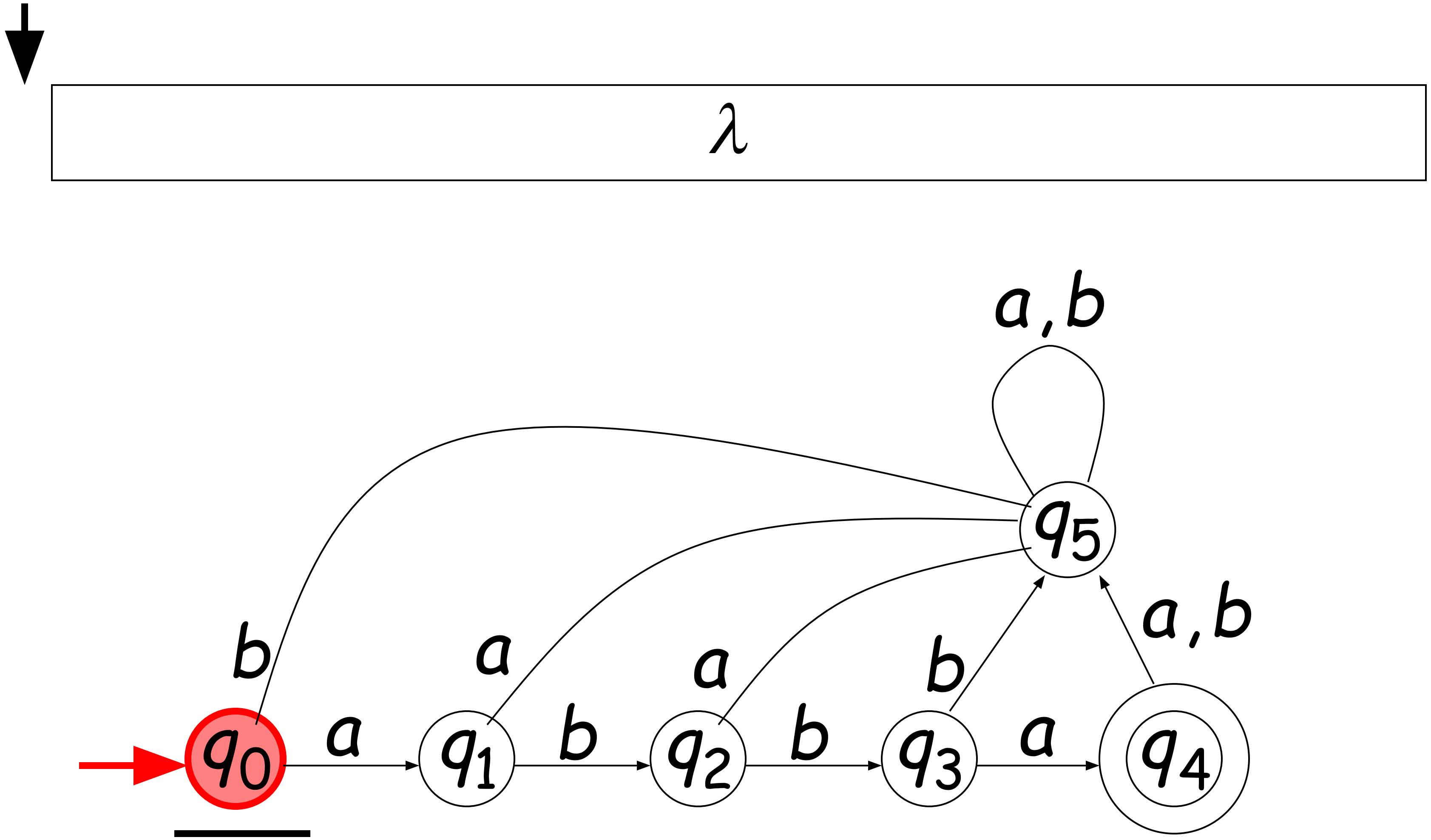

a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

Input finished

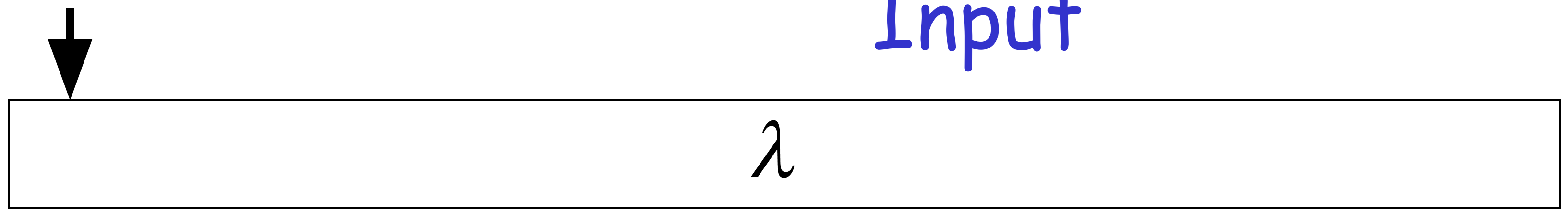




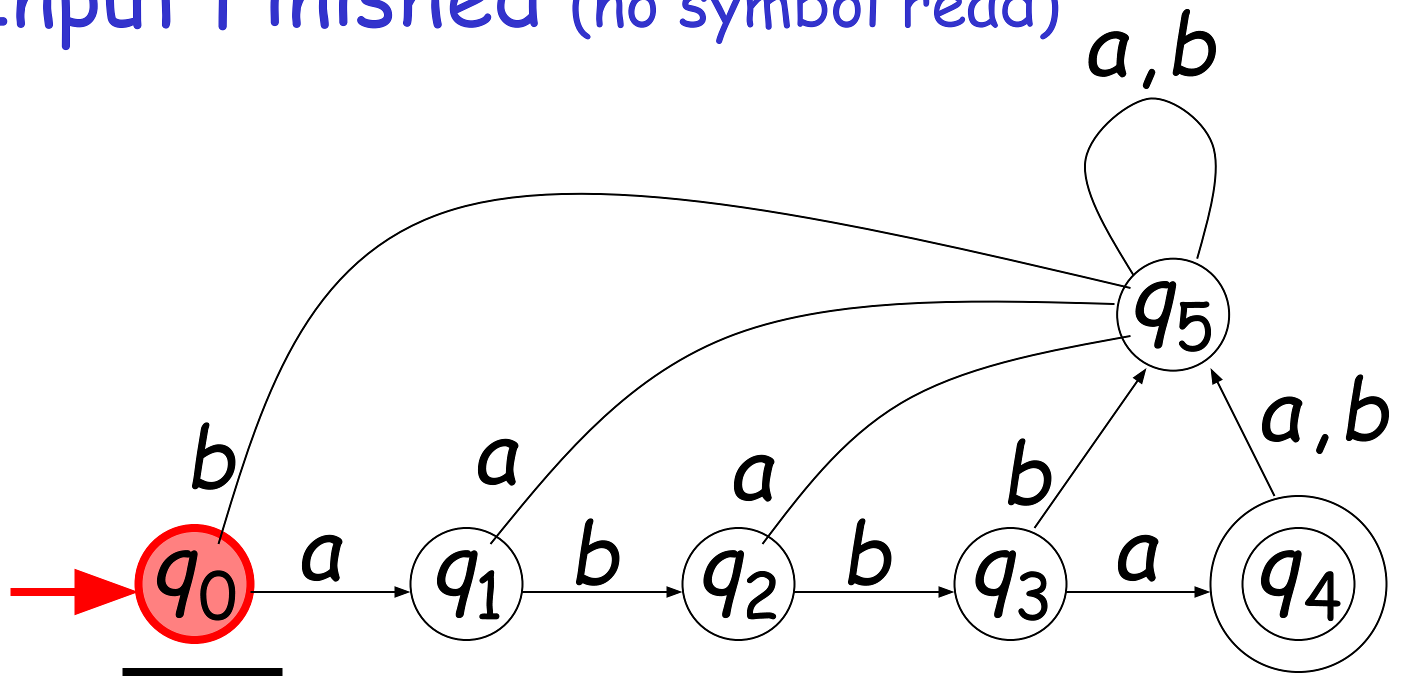
Another Rejection



Input

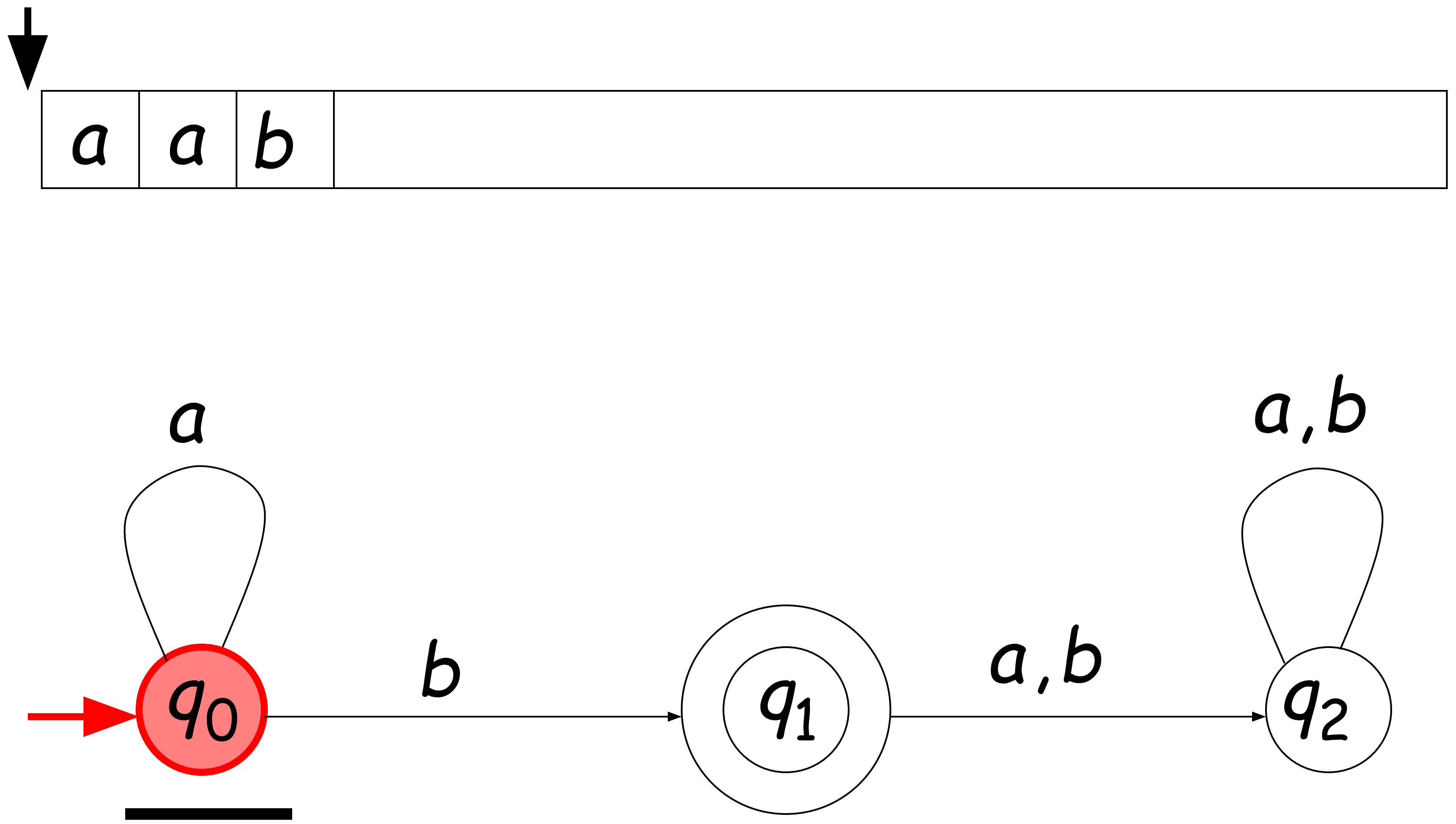


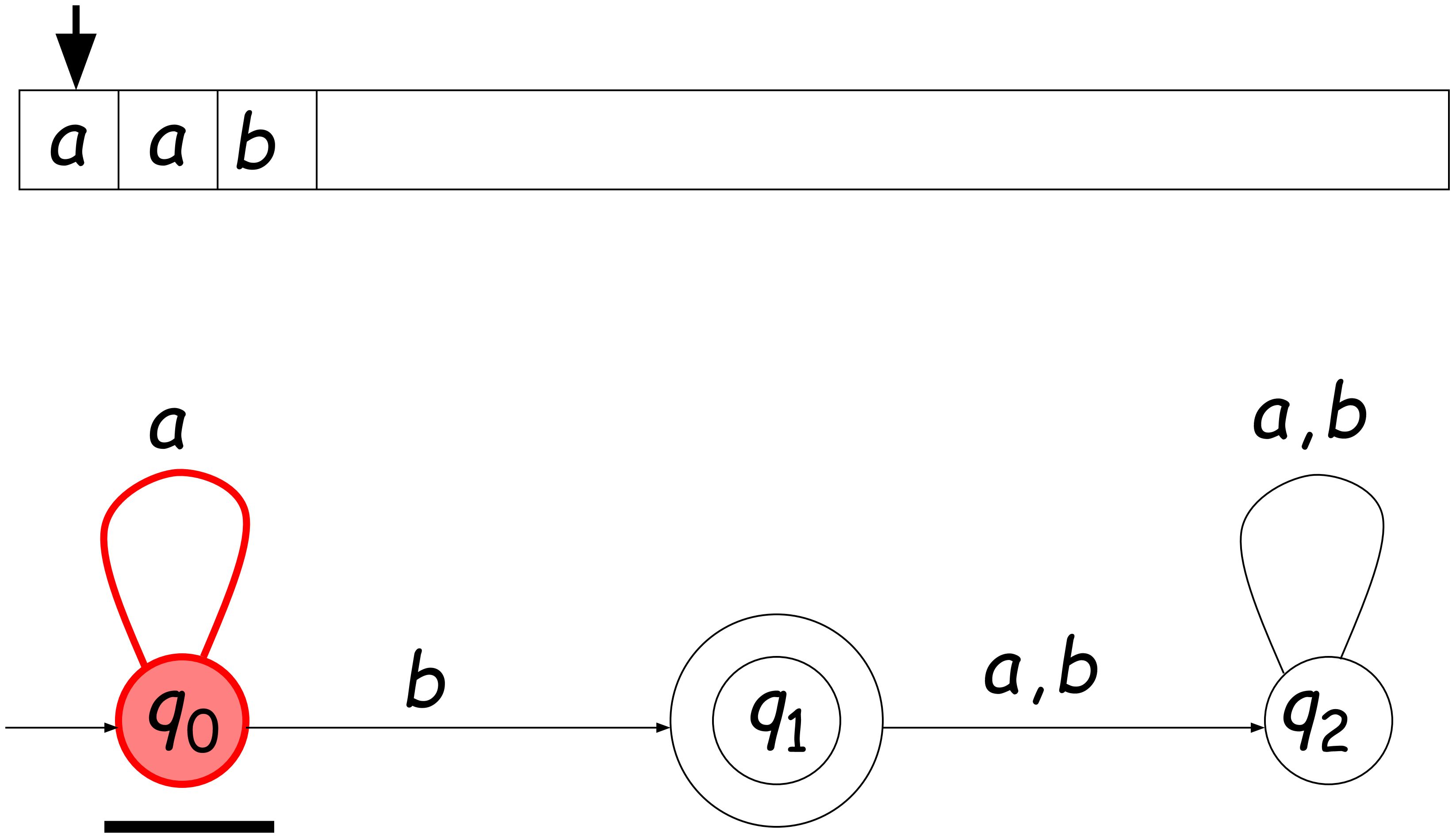
Input Finished (no symbol read)

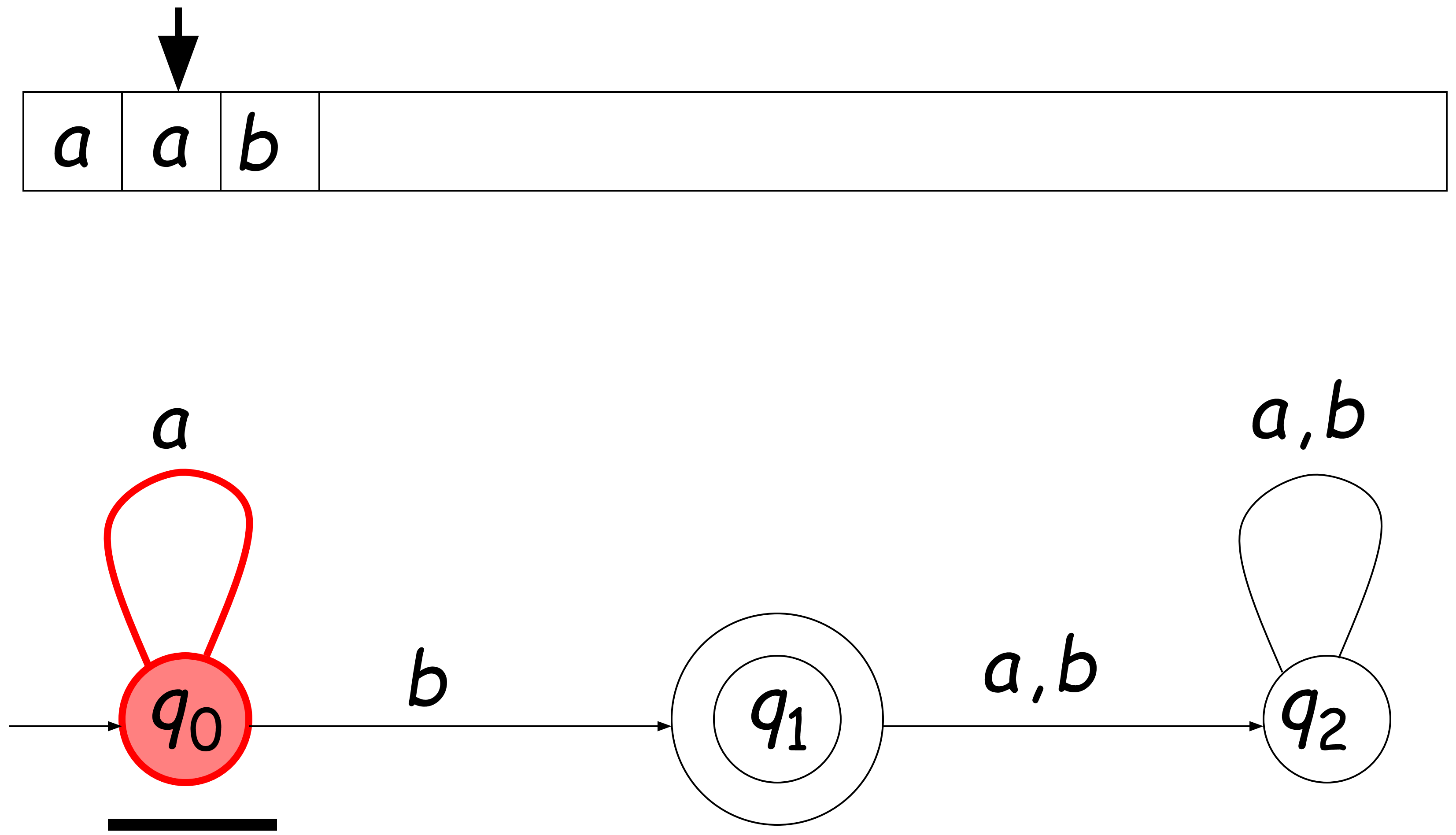


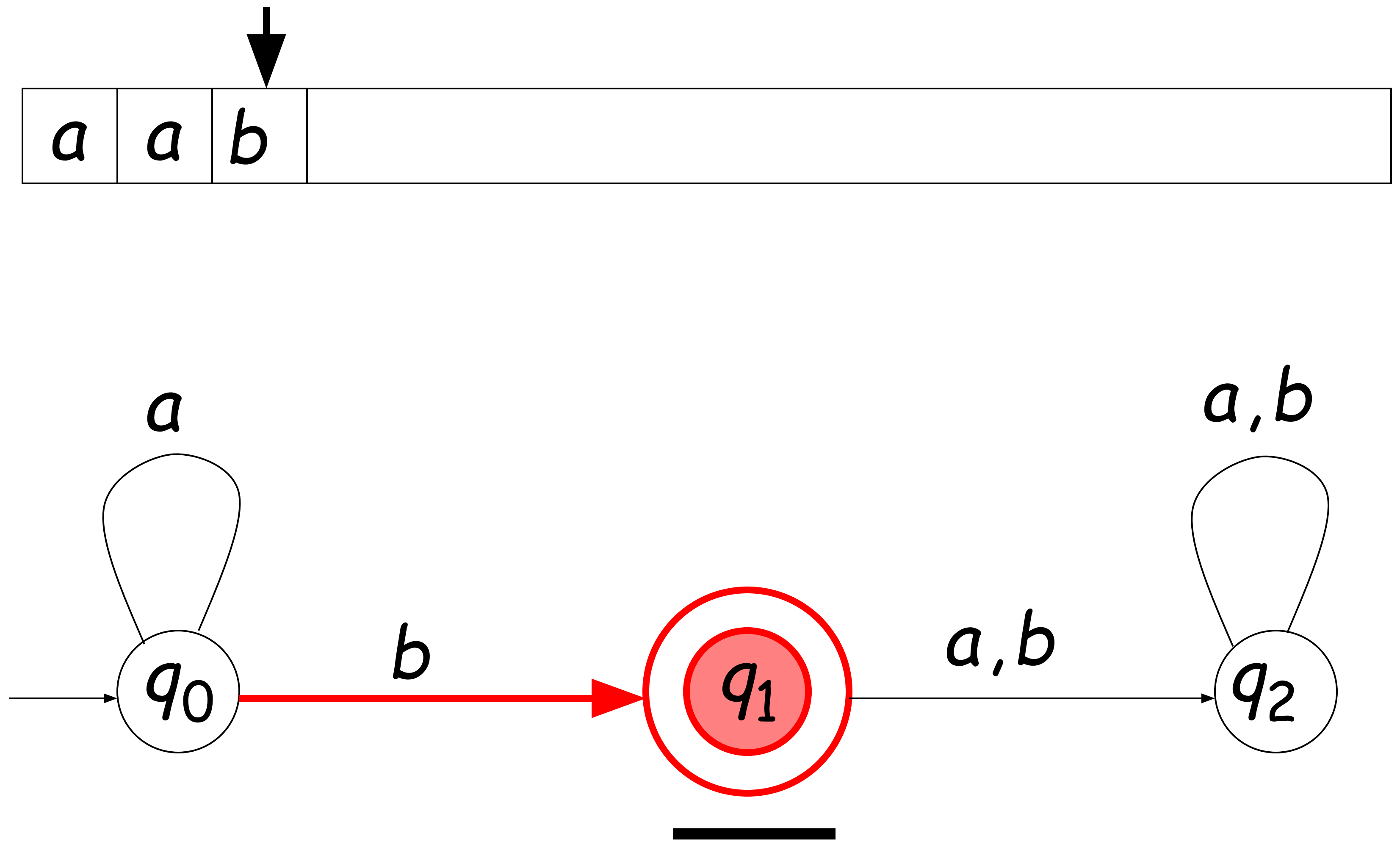
Output:
"reject"

Another Example

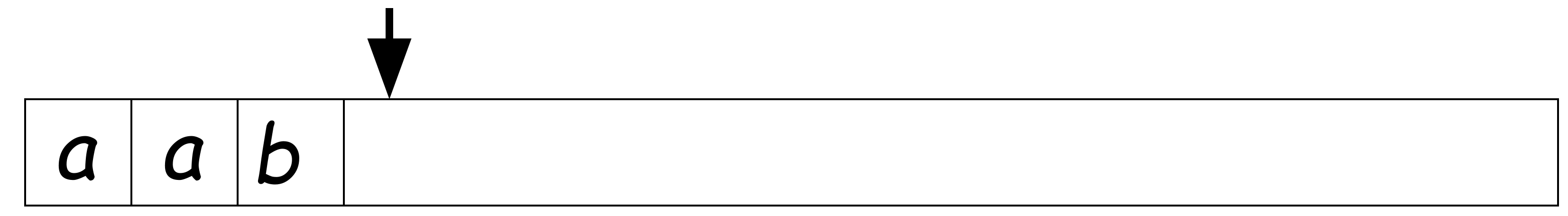


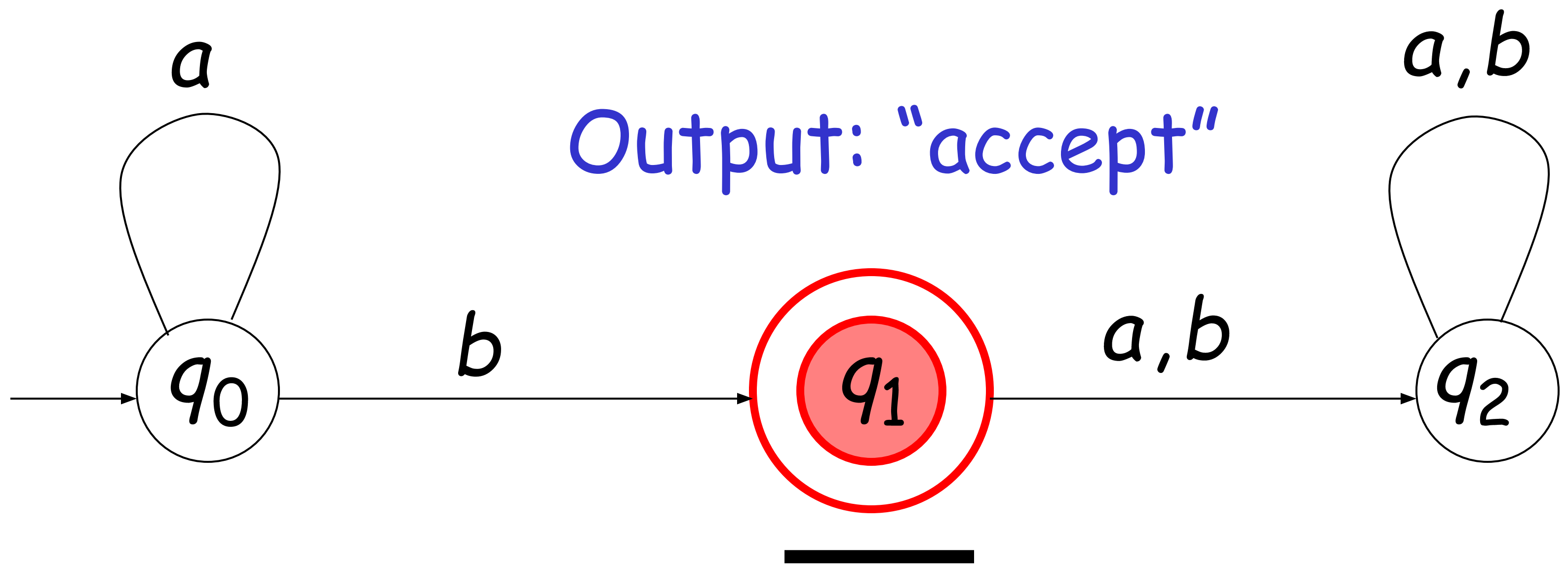

a
a
b
a
b
a,b
a,b
q_0
q_1
q_2


a
a
b
a
a,b
b
a,b
q0
q1
q2


a
a
b
a
a,b
b
a,b
q0
q1
q2

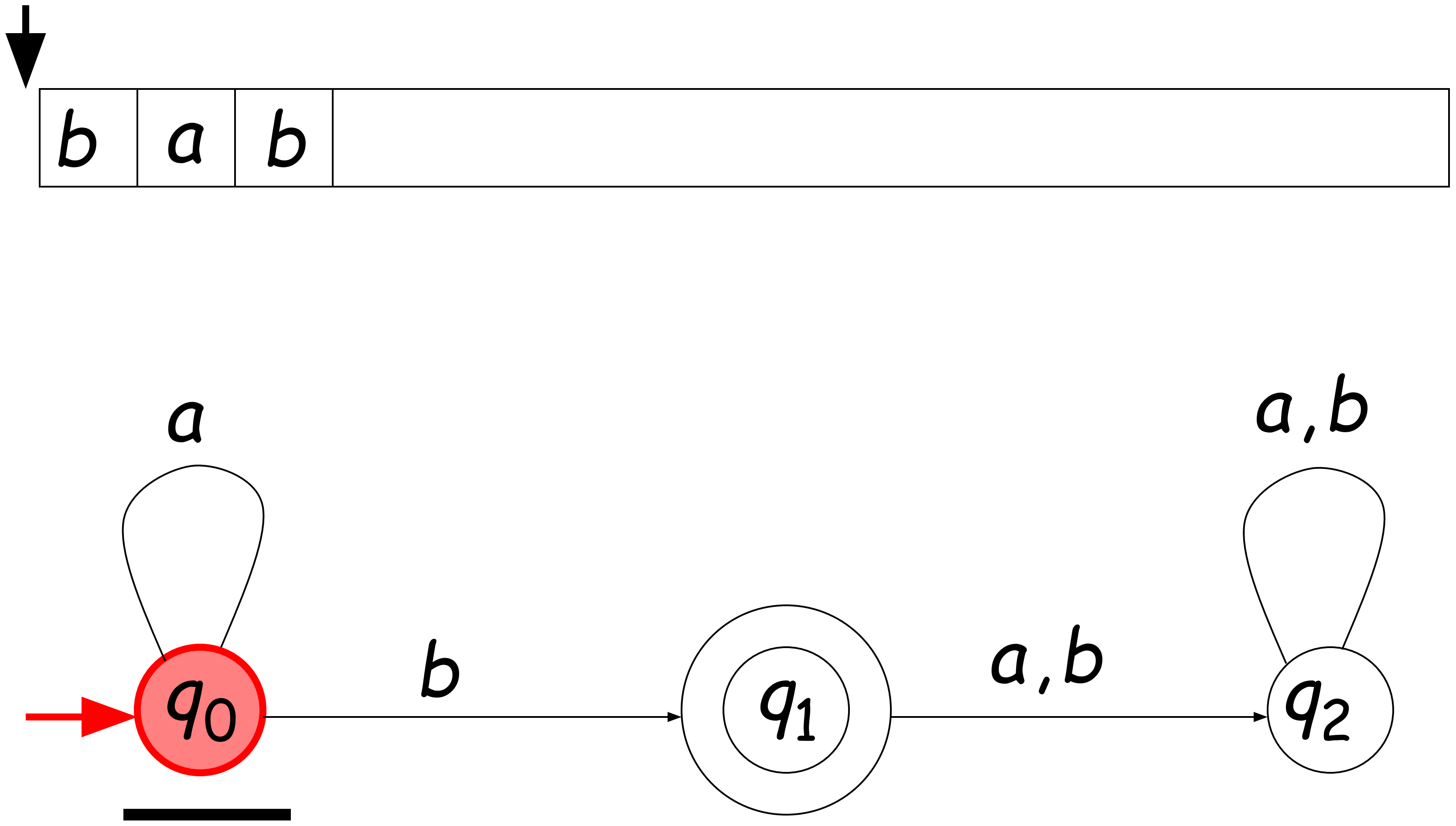
Input finished

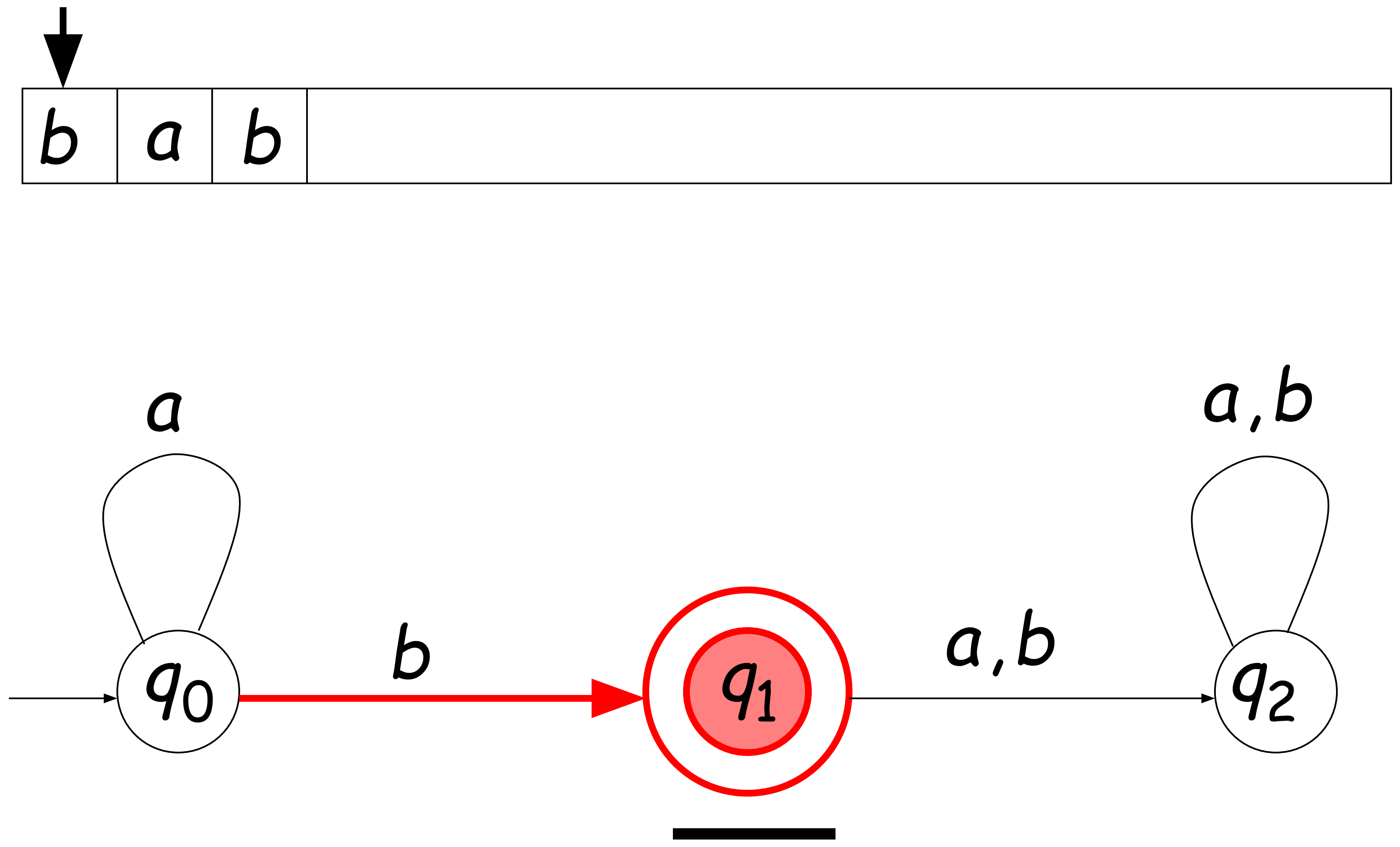


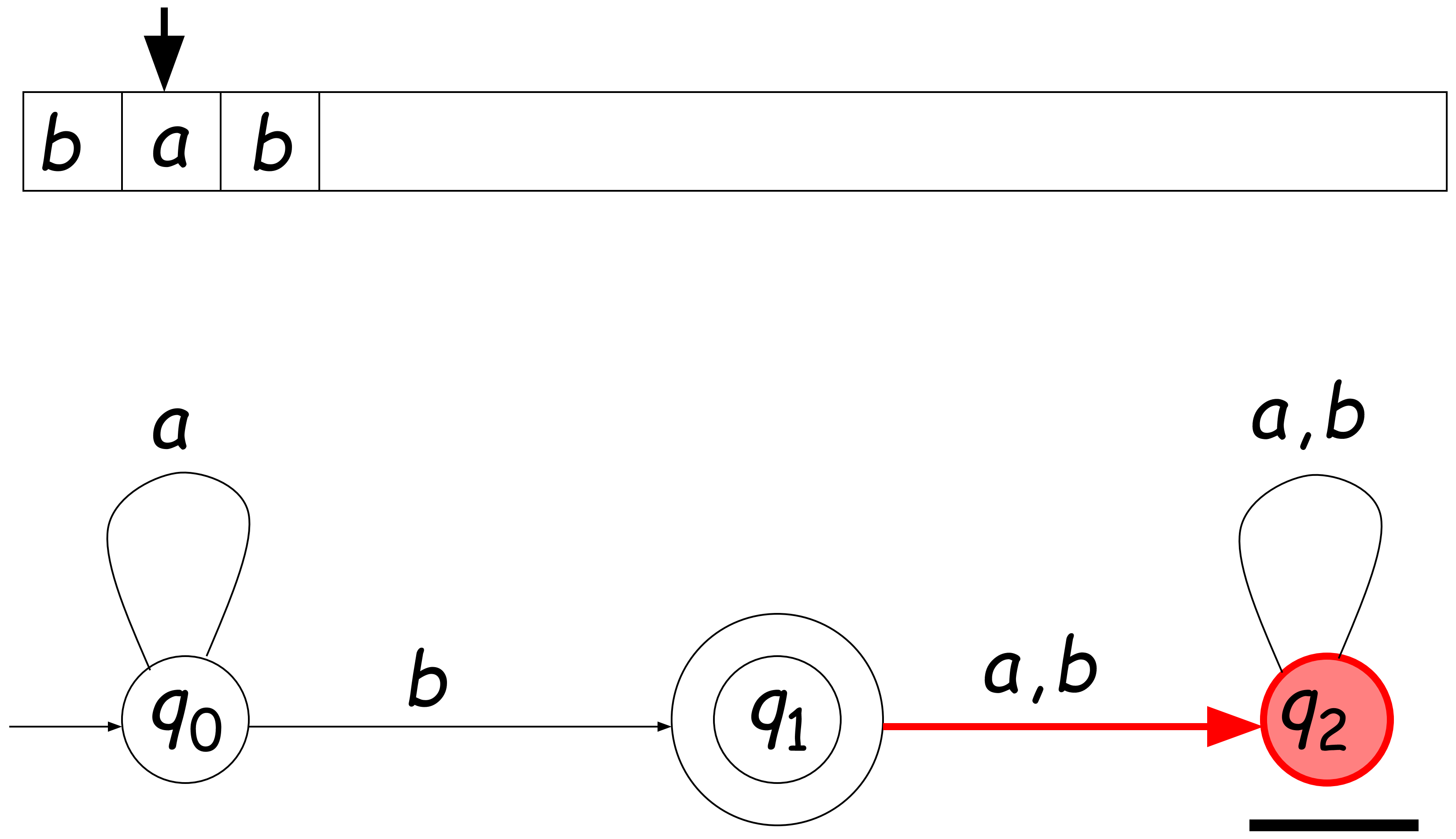


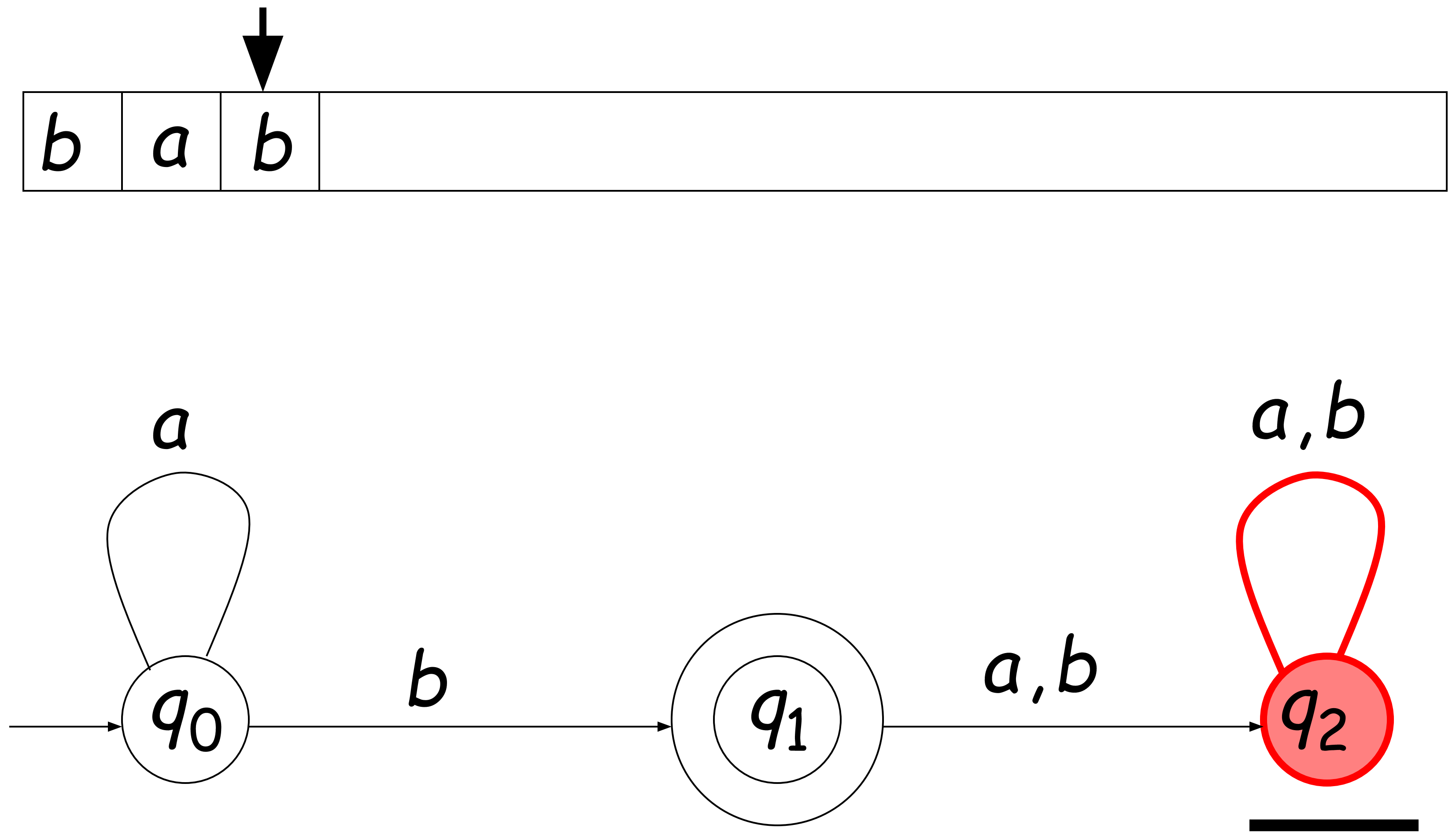
Output: "accept"

Rejection

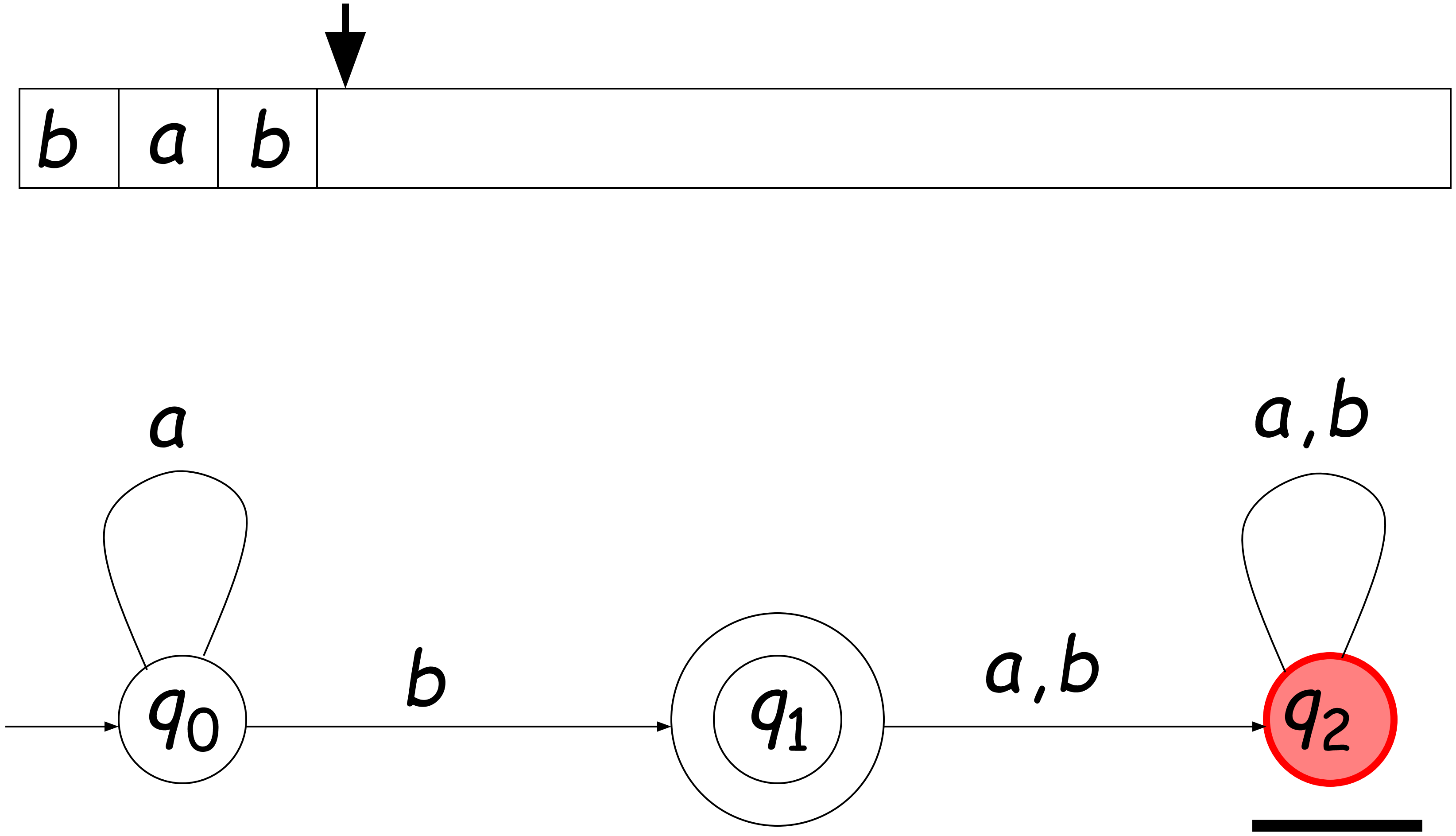



b
a
b
a
a,b
b
a,b
q_0
q_1
q_2


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


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

Input finished



Output: "reject"

Formal Definition

Deterministic Finite Automaton (DFA)

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

Q : finite set of states

Σ : finite set of input alphabet

δ : transition function

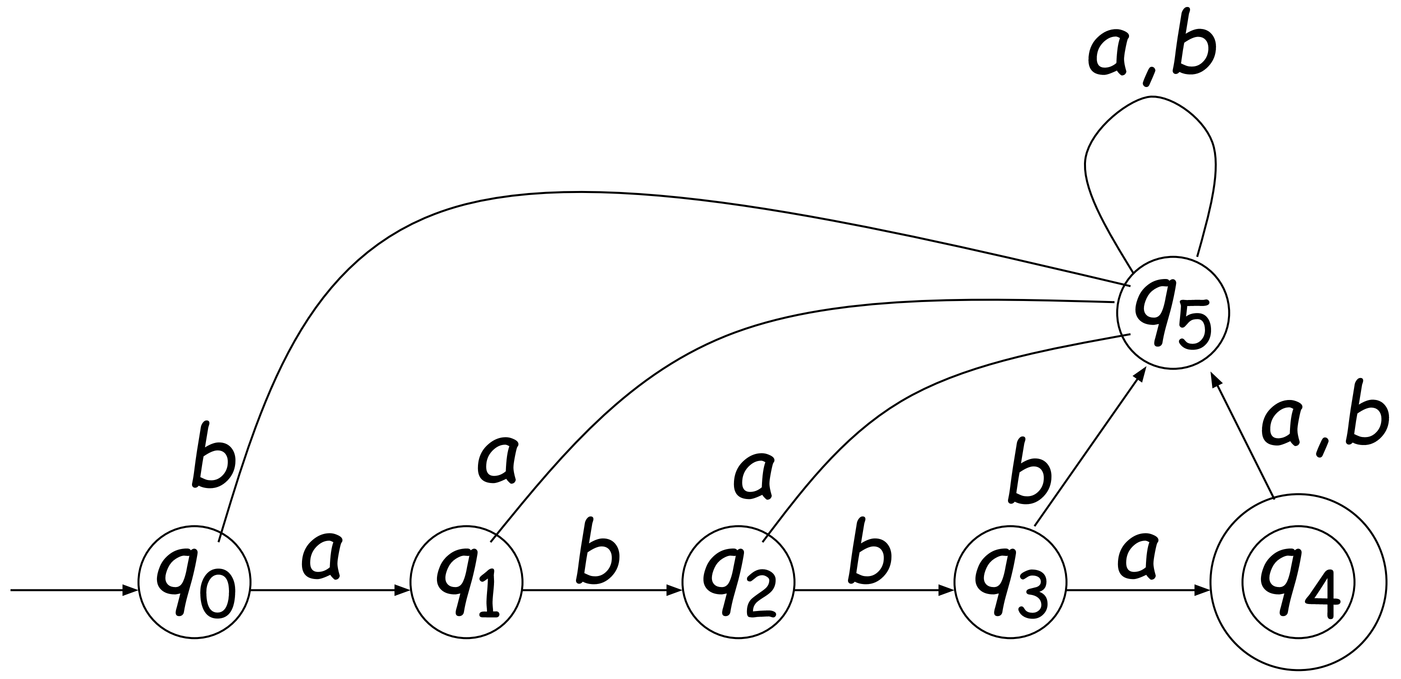
q_0 : initial state $q_0 \in Q$

F : set of final states $F \subseteq Q$

Input Alphabet Σ

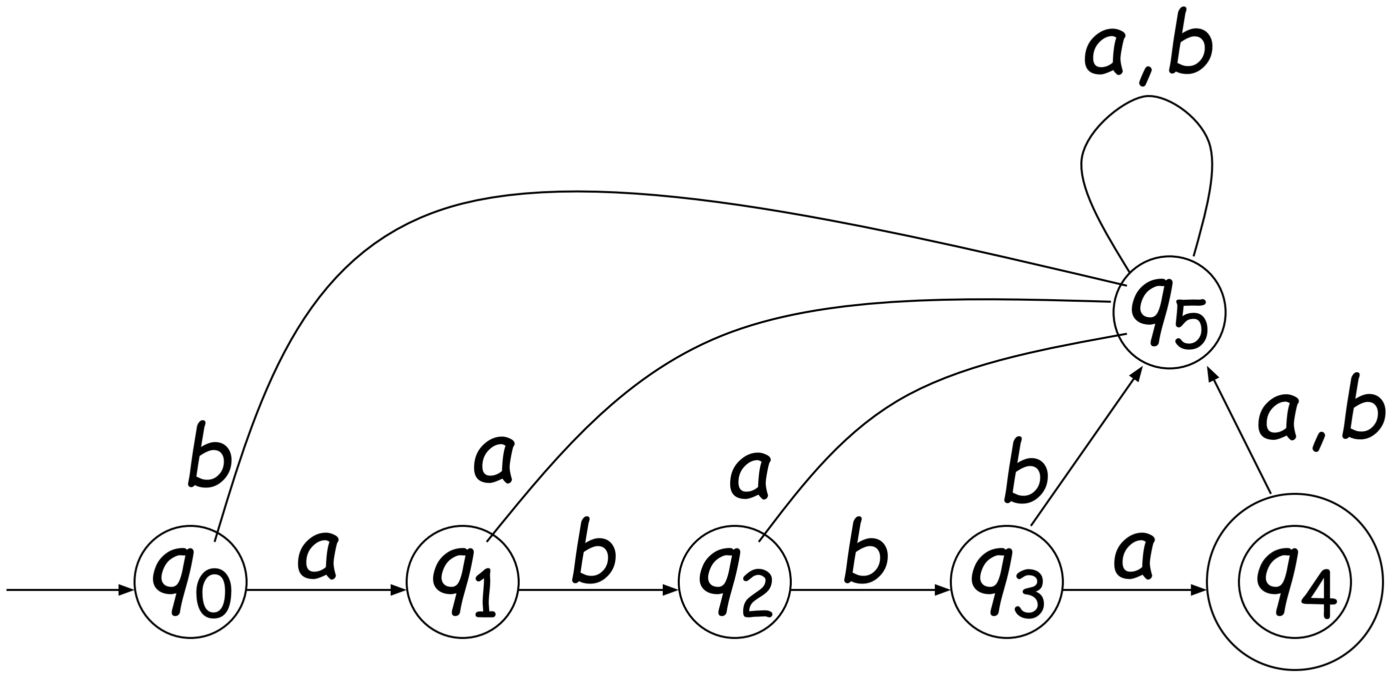
$\lambda \notin \Sigma$:the input alphabet never contains λ

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

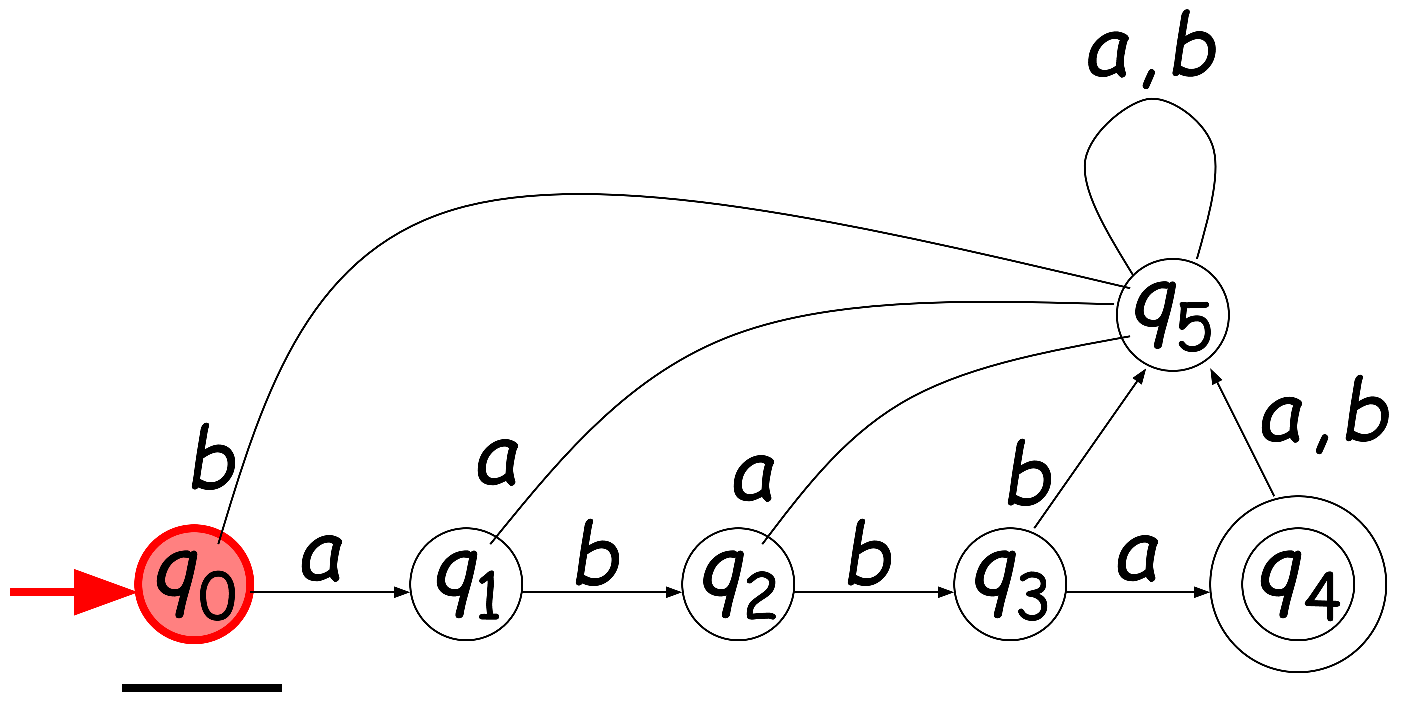


Set of States Q

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



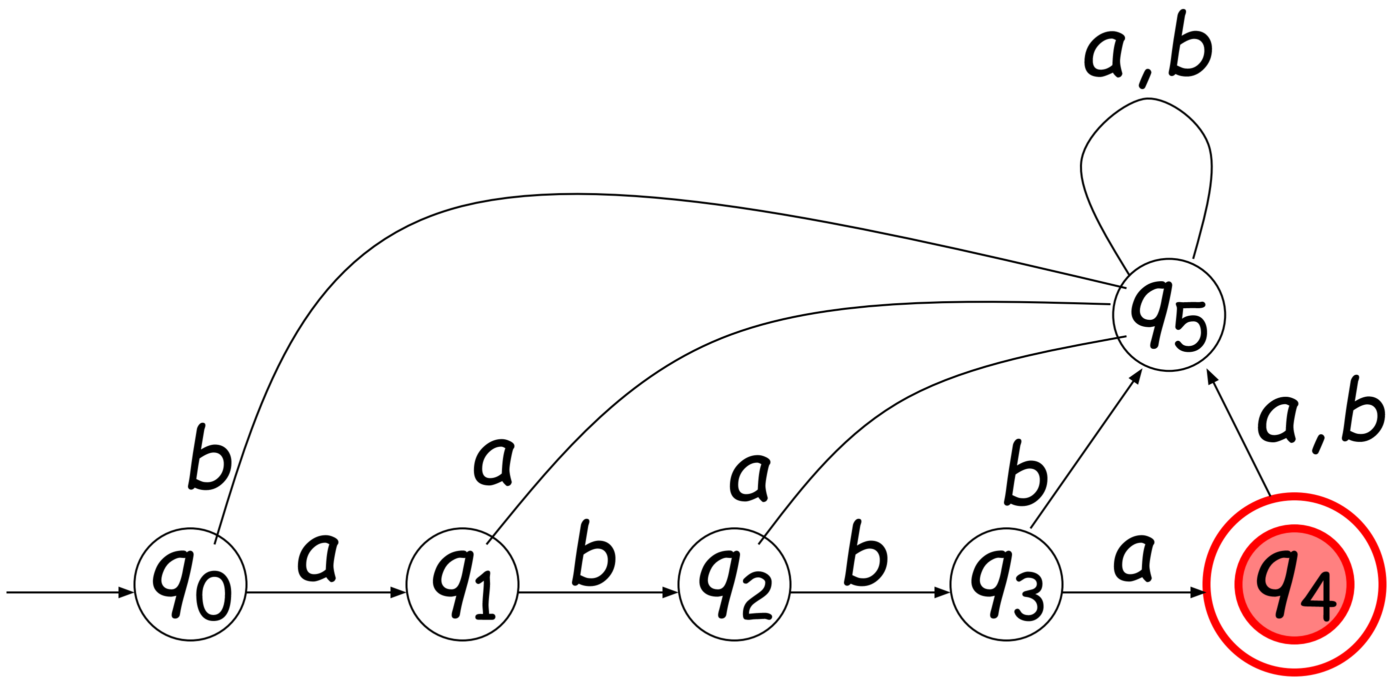
Initial State q_0



Set of Final States F

$F = \{q_4\}$

$F \subseteq Q$



To accept a string:

all the input string is scanned
and the last state (q_{final}) is accepting

$$q_{final} \in F$$

To reject a string:

all the input string is scanned
and the last state (q_{last}) is non-accepting

$q_{last} \notin F$ $\qquad$ $q_{last} \in (Q - F)$

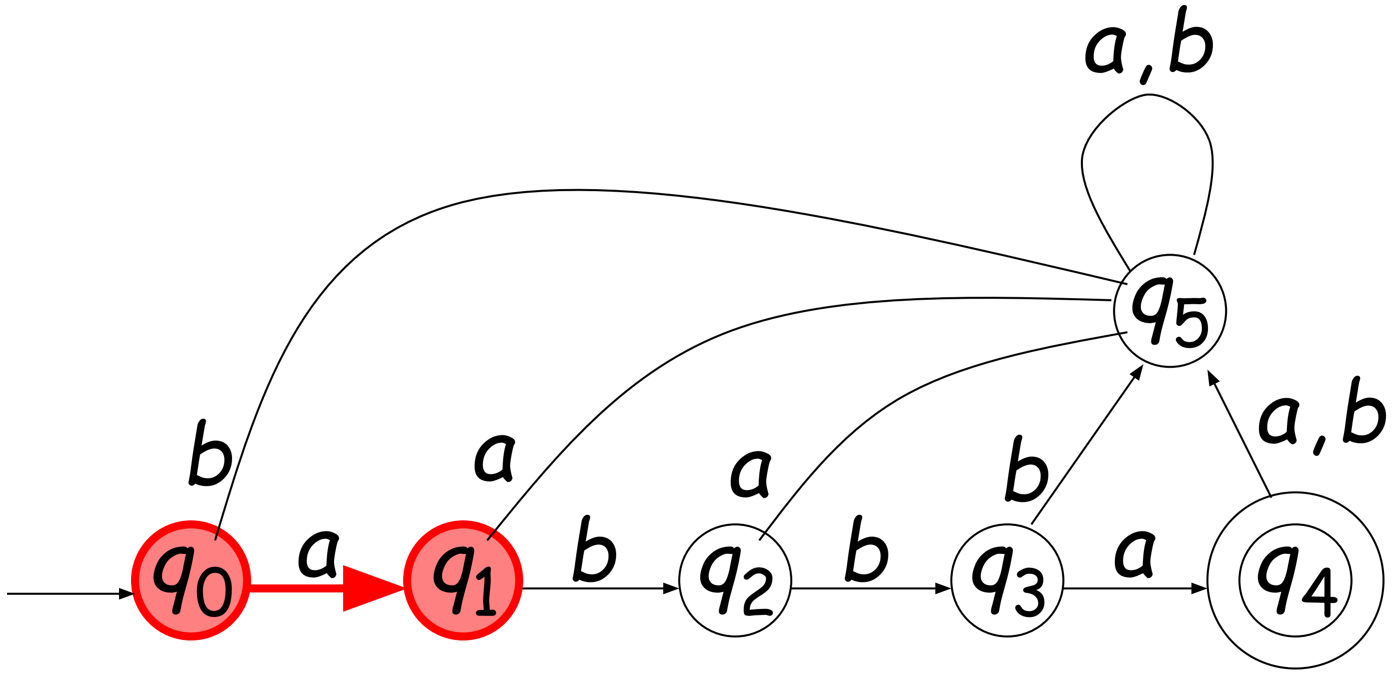
Transition Function

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

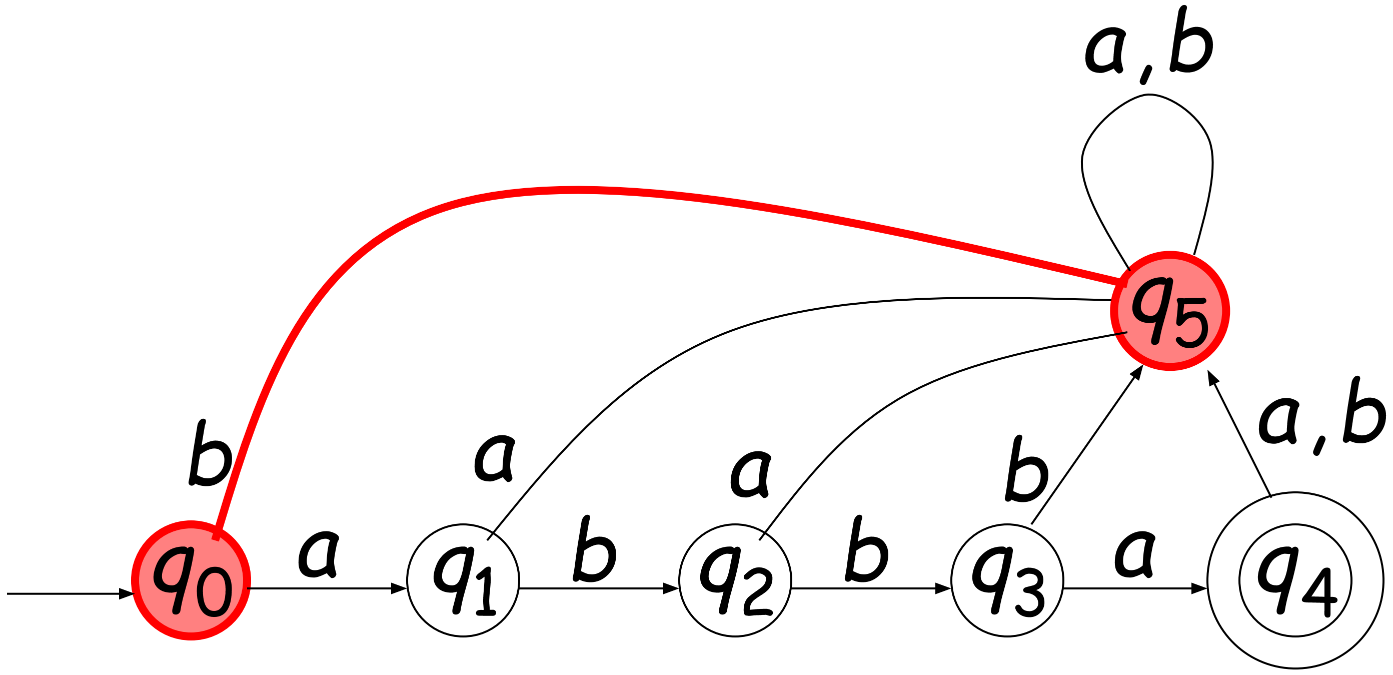
$$\begin{tikzcd} \circled{q} \arrow[r, "x"] & \circled{q'} \end{tikzcd}$$

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

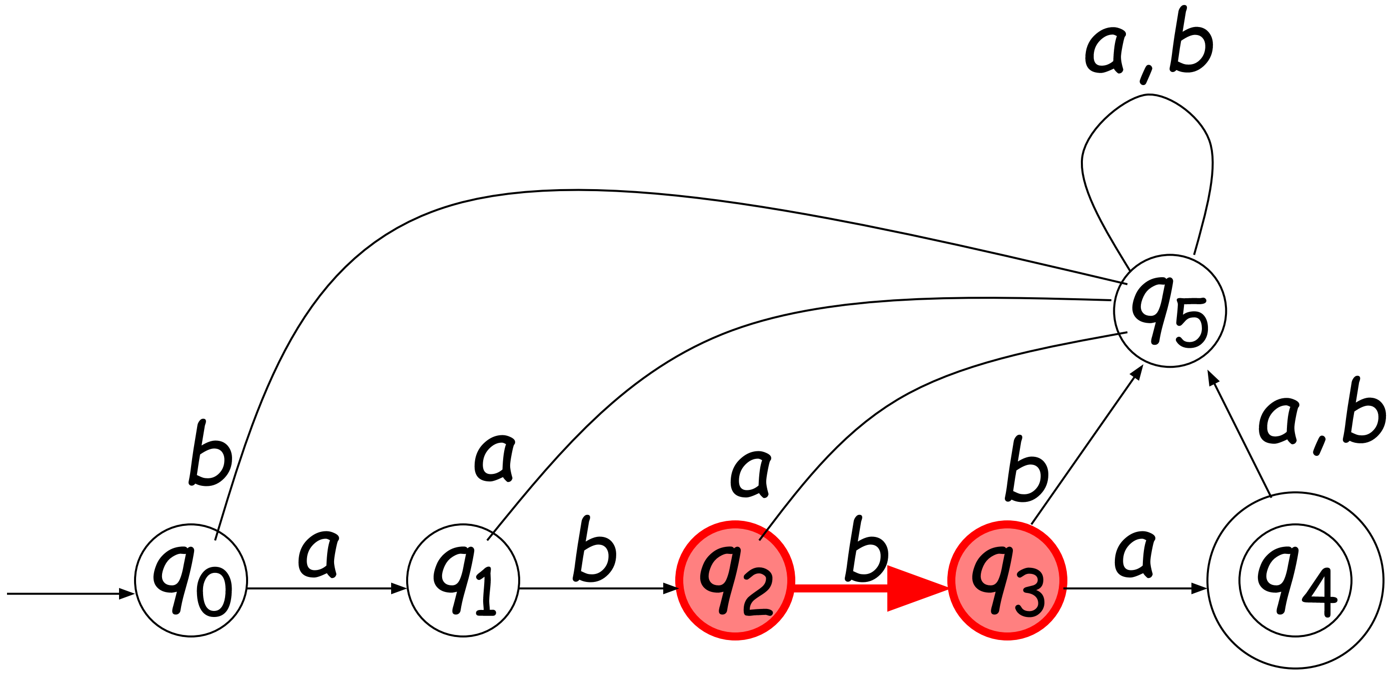
$$\delta(q_0, a) = q_1$$



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

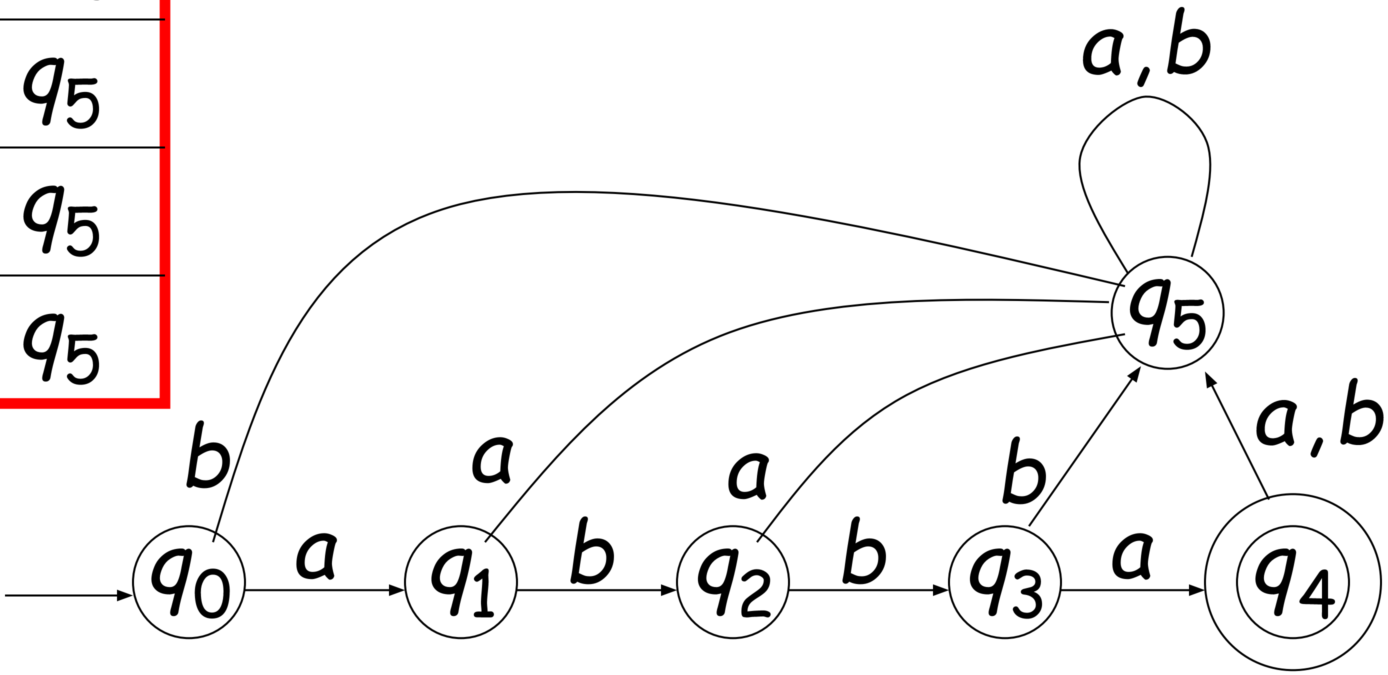


$$\delta(q_2, b) = q_3$$



Transition Function δ

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

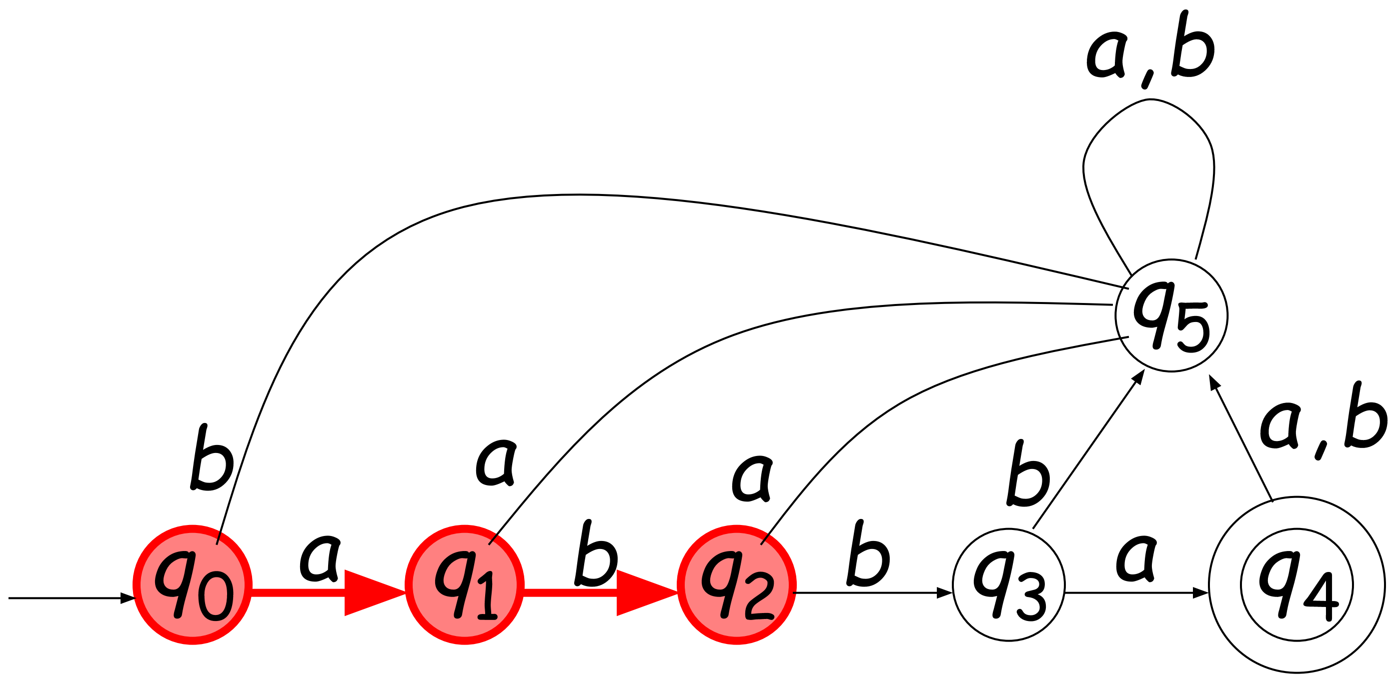


Extended Transition Function

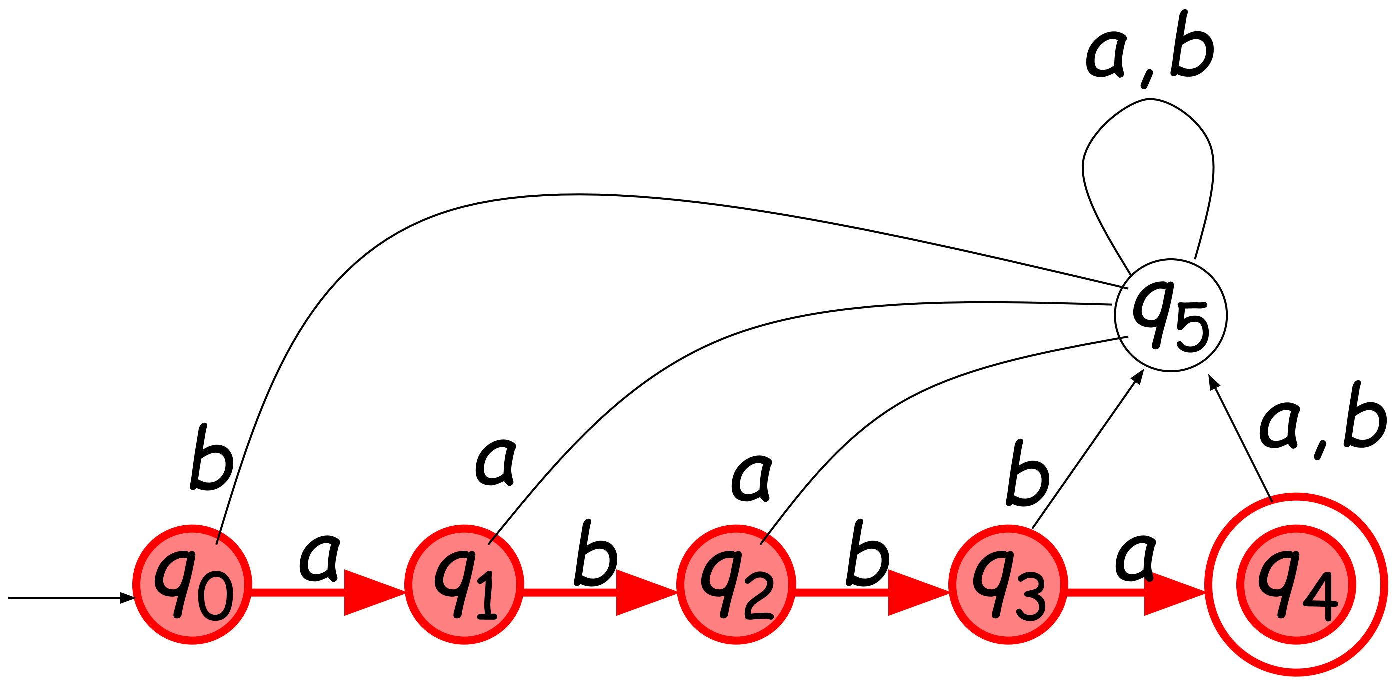
$$\delta^*(q, w) = q'$$

Describes the resulting state after scanning string w from state q

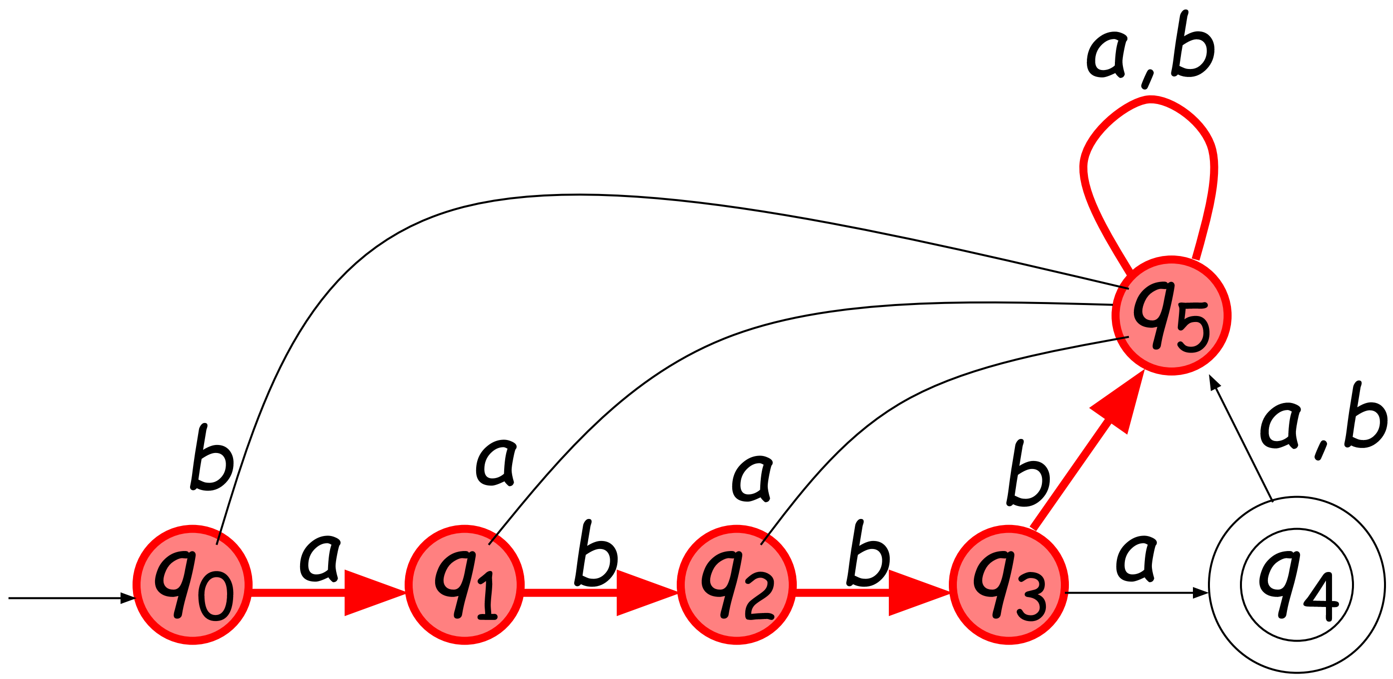
$$\delta^*(q_0, ab) = q_2$$



$$\delta^*(q_0, abba) = q_4$$



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



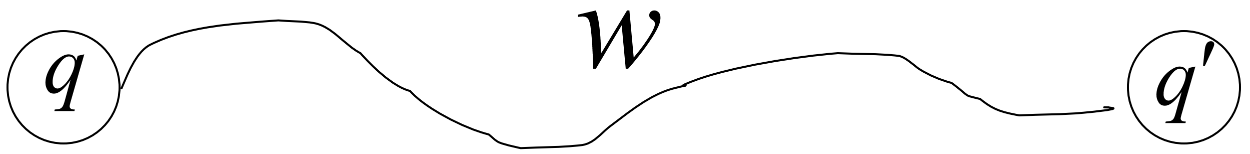
Special case:

for any state q

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

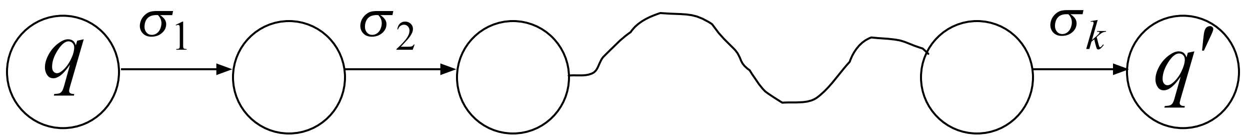
Observation: There is a walk from q to q' with label w

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



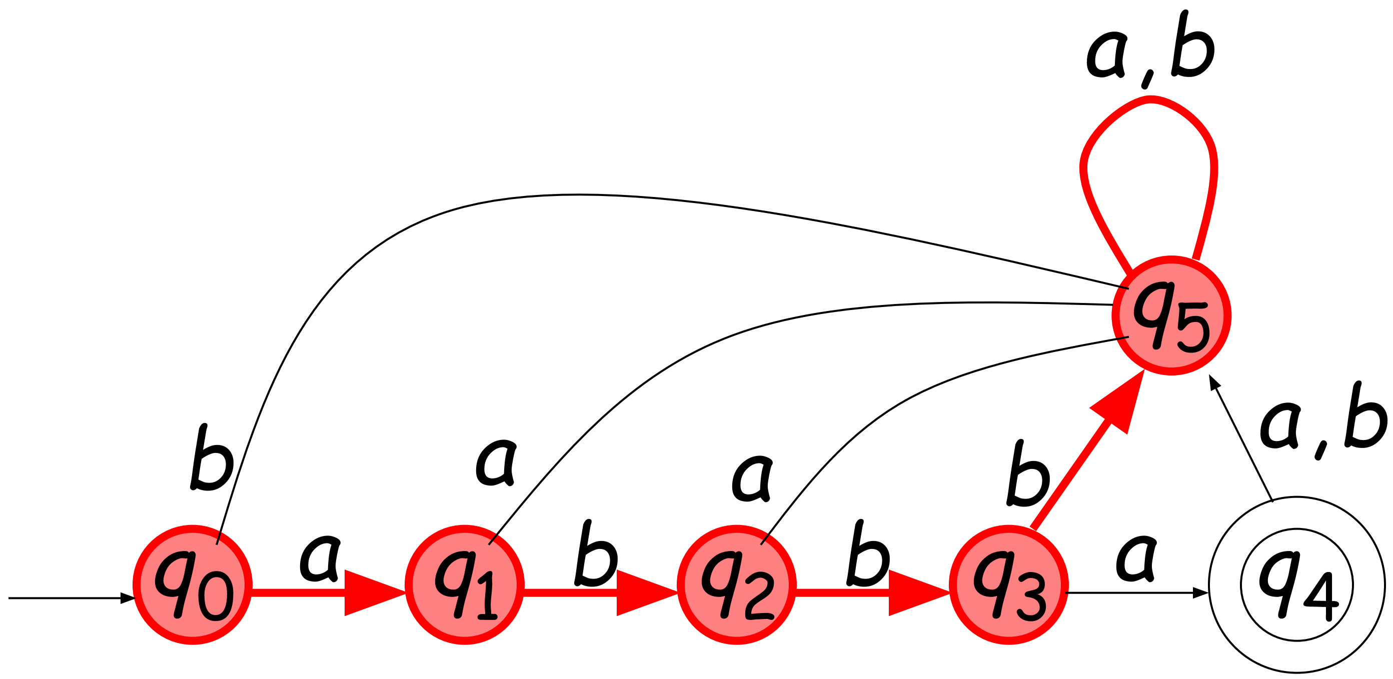
states may be repeated

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



Example: There is a walk from q_0 to q_5 with label $abbbaa$

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



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

$$\delta^*(q_0, ab) =$$

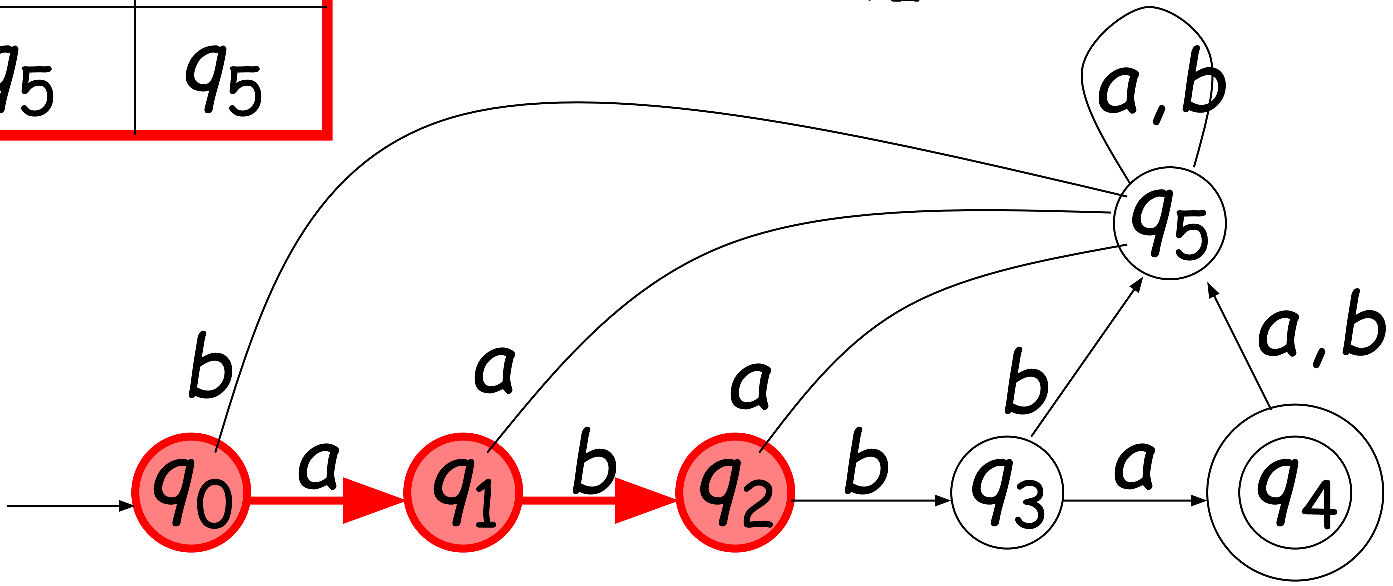
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

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

$$\delta(q_1, b) =$$

$$q_2$$



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

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

$$\delta^*(q_0, ab) =$$

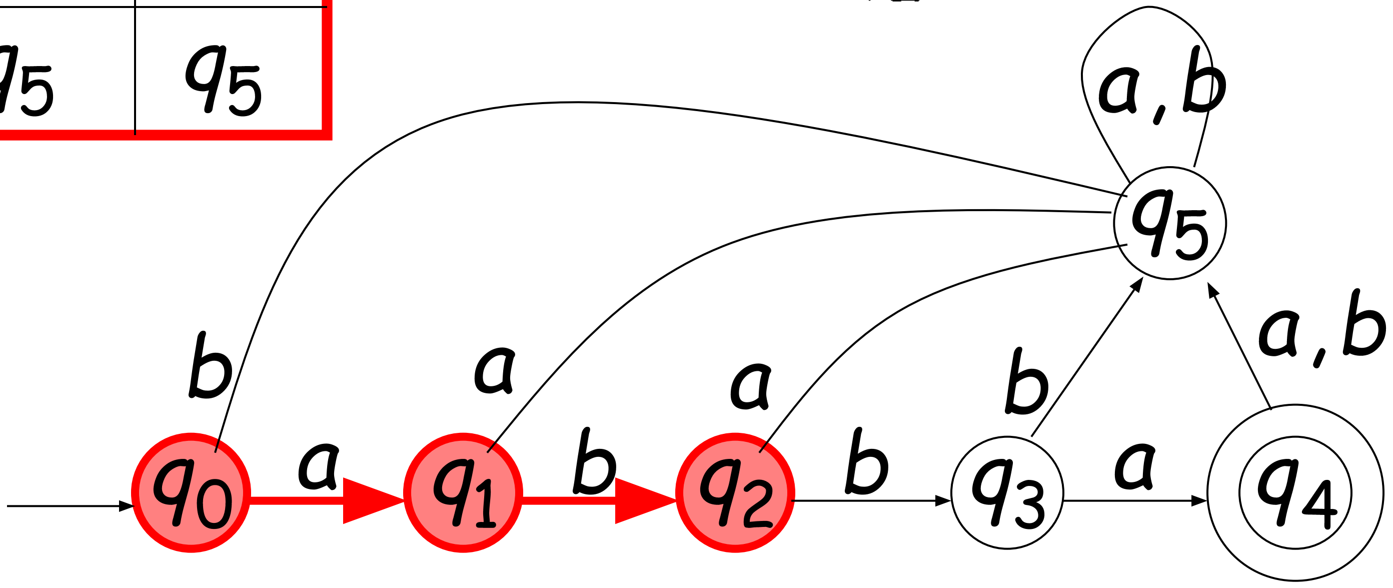
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

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

$$\delta(q_1, b) =$$

$$q_2$$



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

$$\delta^*(q_0, ab) =$$

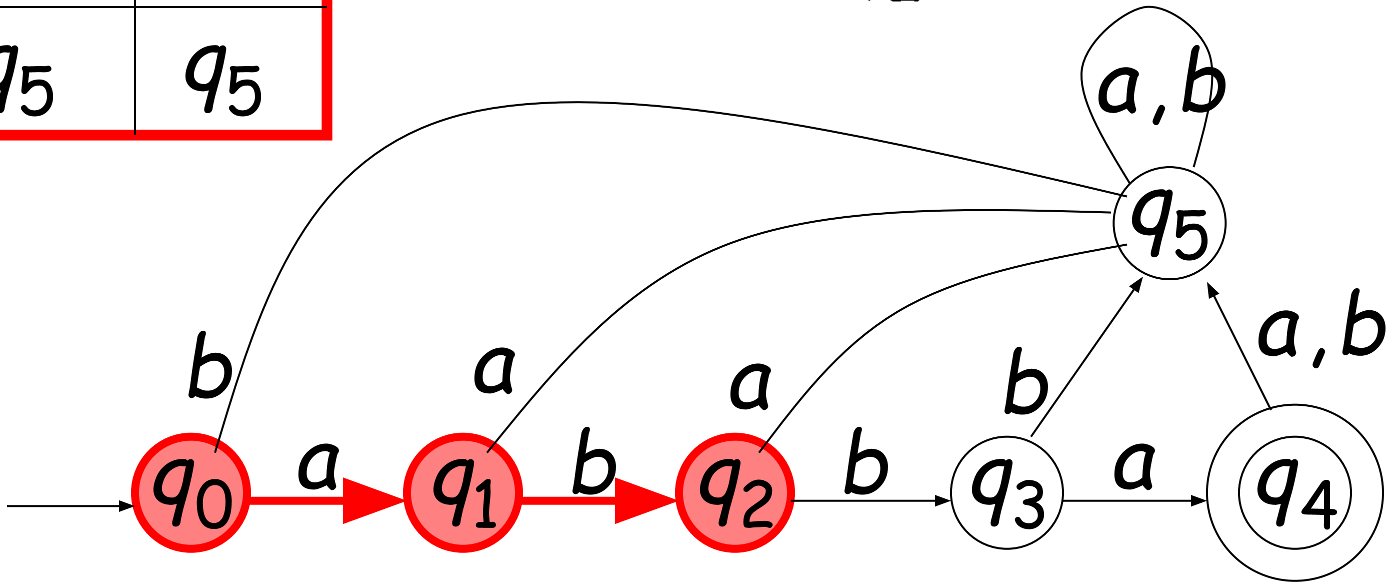
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

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

$$\delta(q_1, b) =$$

$$q_2$$



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

$$\delta^*(q_0, ab) =$$

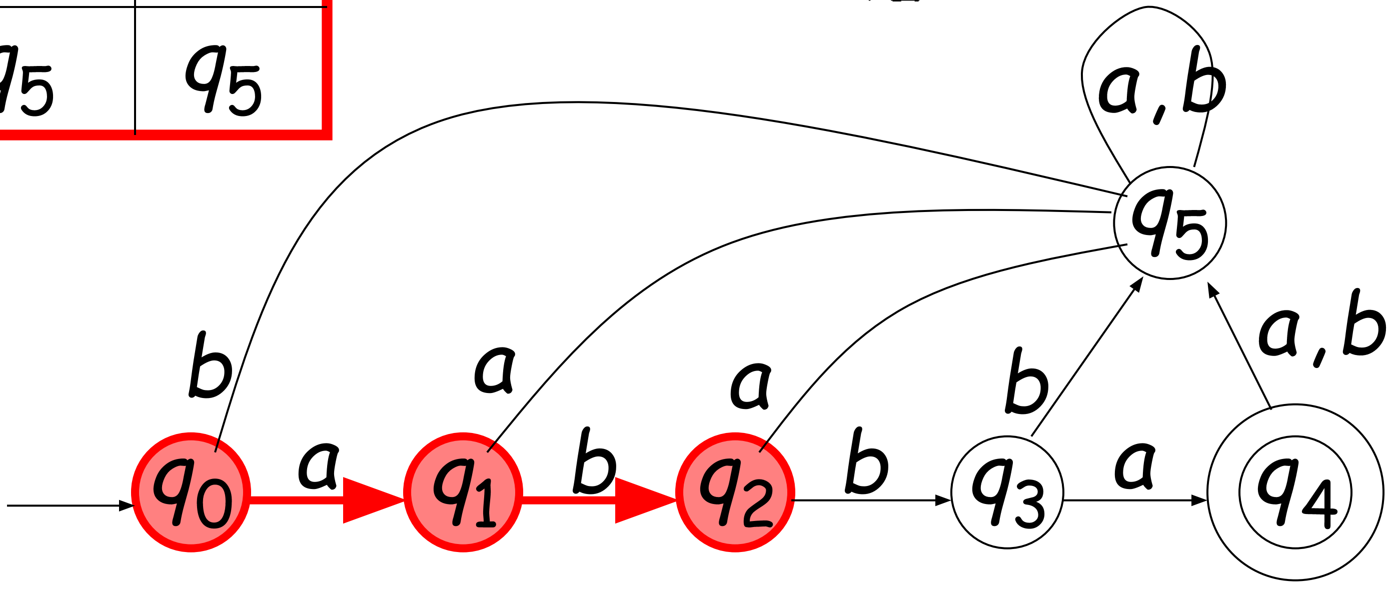
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

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

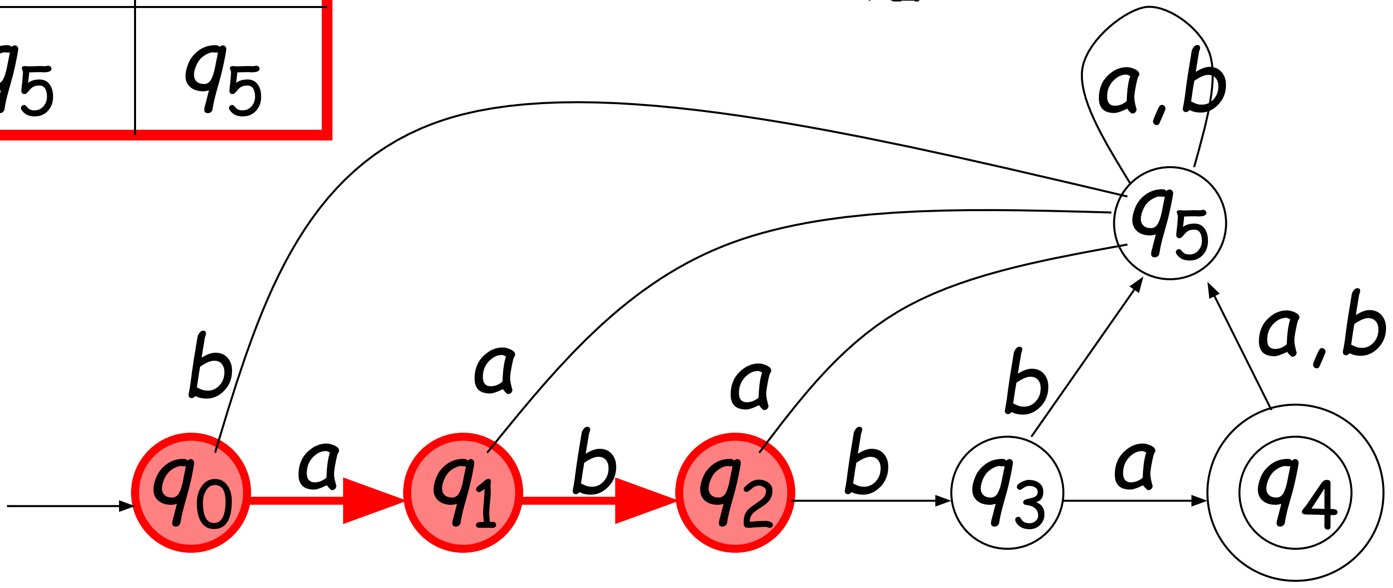
$$\delta(q_1, b) =$$

$$q_2$$



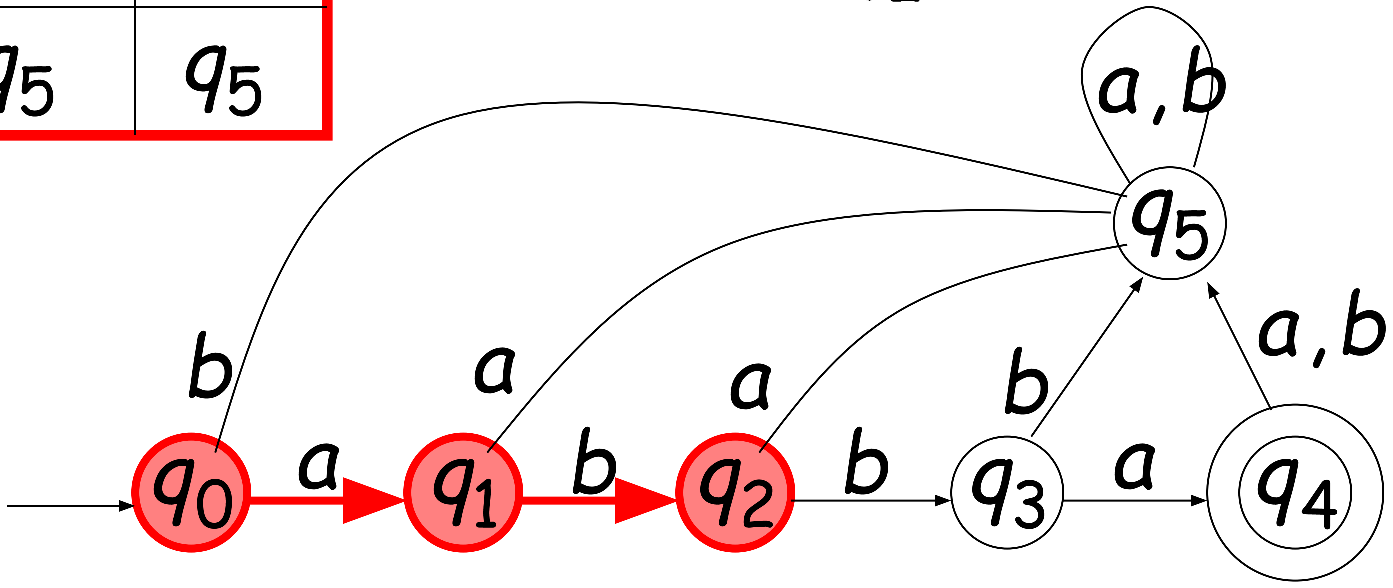
δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5

$$\delta^*(q_0, ab) =$$
$$\delta(\delta^*(q_0, a), b) =$$
$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$
$$\delta(\delta(q_0, a), b) =$$
$$\delta(q_1, b) =$$
$$q_2$$



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

$$\delta^*(q_0, ab) =$$
$$\delta(\delta^*(q_0, a), b) =$$
$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$
$$\delta(\delta(q_0, a), b) =$$
$$\delta(q_1, b) =$$
$$q_2$$



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

$$\delta^*(q_0, ab) =$$

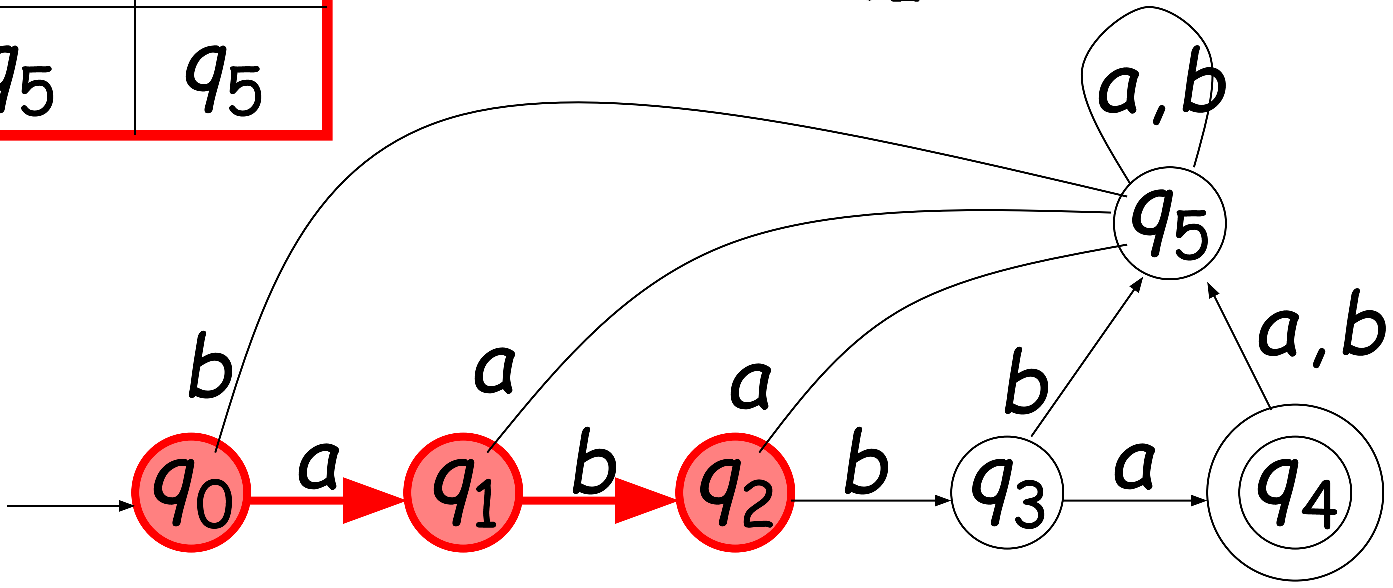
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

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

$$\delta(q_1, b) =$$

$$q_2$$



Languages Accepted by DFAs

Take DFA M

Definition:

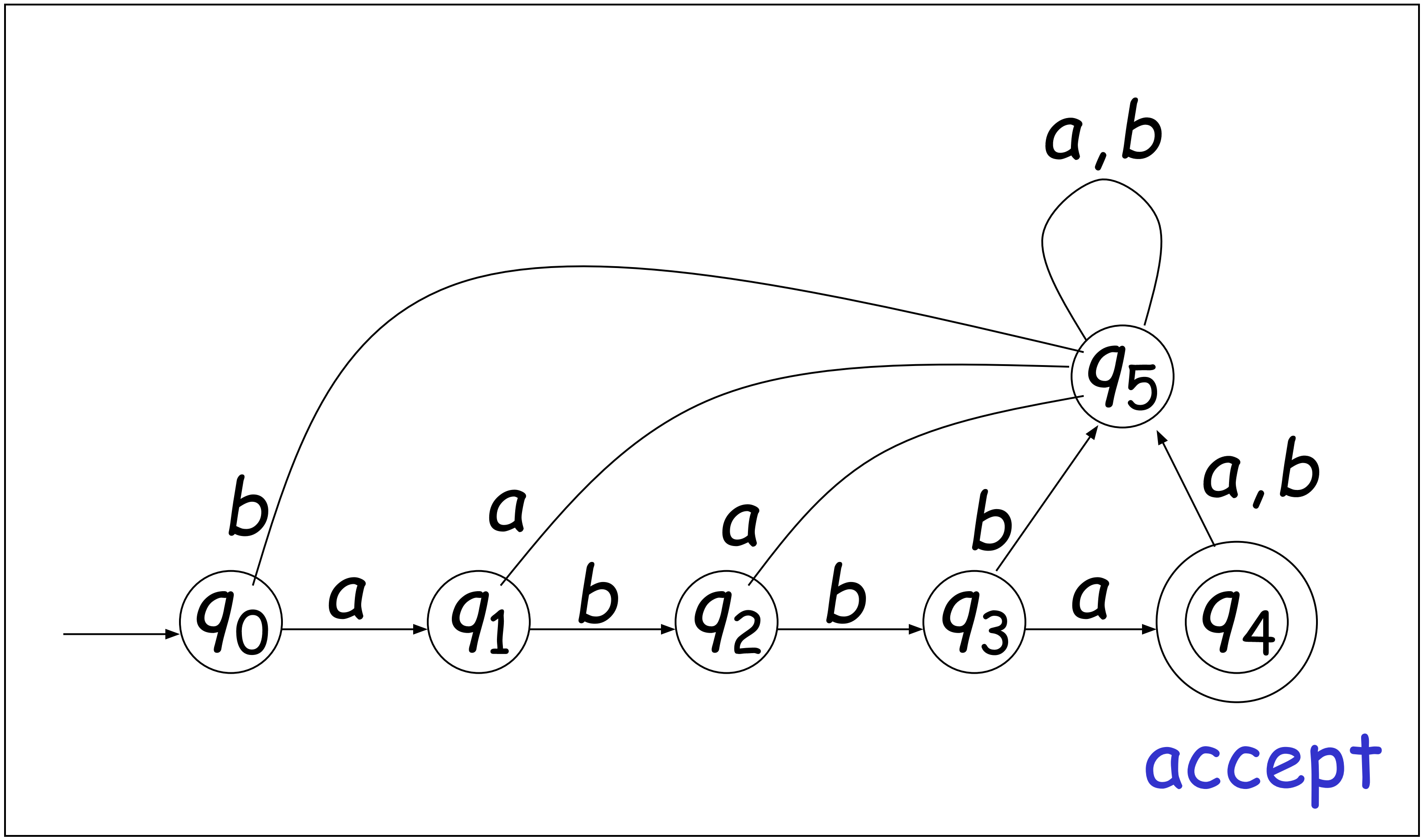
The language $L(M)$ contains all input strings accepted by M

$L(M)$ = { strings that drive M to a final state}

Example # 1

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

M

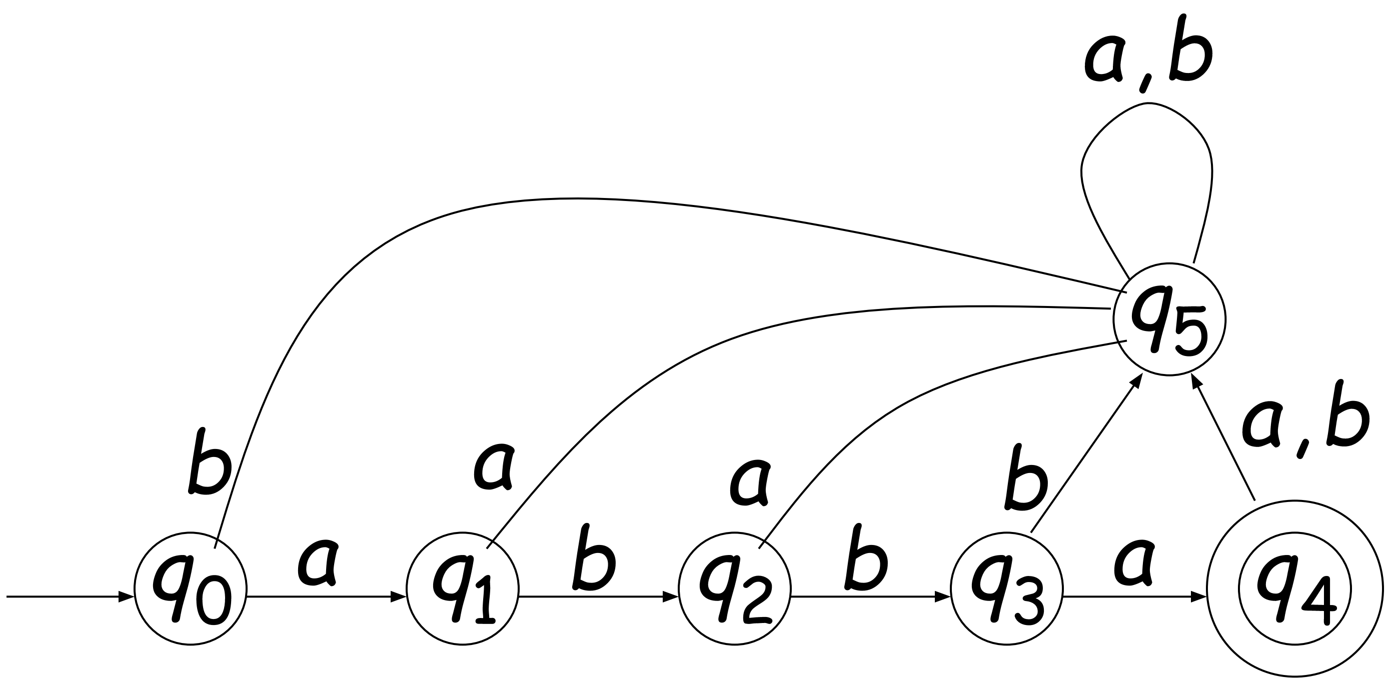

$$\begin{tikzcd}
 & & & & q_5 \arrow[loop above, "a,b"] \\
\arrow[r] & q_0 \arrow[r, "a"] \arrow[urrr, bend left, "b"] & q_1 \arrow[r, "b"] \arrow[urr, bend left, "a"] & q_2 \arrow[r, "b"] \arrow[ur, bend left, "a"] & q_3 \arrow[r, "a"] \arrow[u, "b"] & q_4 \arrow[ul, "a,b"']
\end{tikzcd}$$

Activity Time

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

This automaton accepts only one string

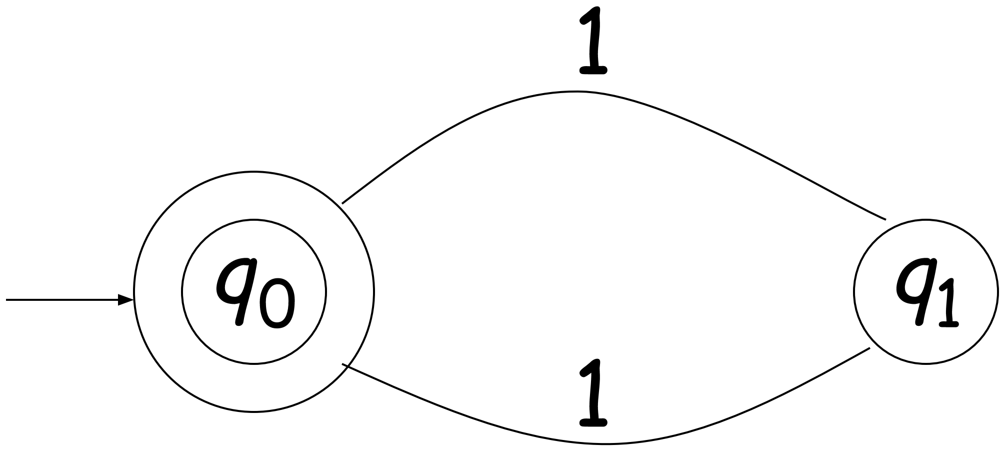
Language Accepted: $L = \{abba\}$


$$\begin{tikzcd}
 & & & & q_5 \arrow[loop above, "{a,b}"] & \\
\arrow[r] & q_0 \arrow[r, "a"] \arrow[urrr, bend left, "b"] & q_1 \arrow[r, "b"] \arrow[urr, bend left, "a"] & q_2 \arrow[r, "b"] \arrow[ur, bend left, "a"] & q_3 \arrow[r, "a"] \arrow[u, "b"] & q_4 \arrow[ul, "{a,b}"']
\end{tikzcd}$$

Make it to accept two strings $L = \{\lambda, abba\}$

Example # 2

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



Language Accepted:

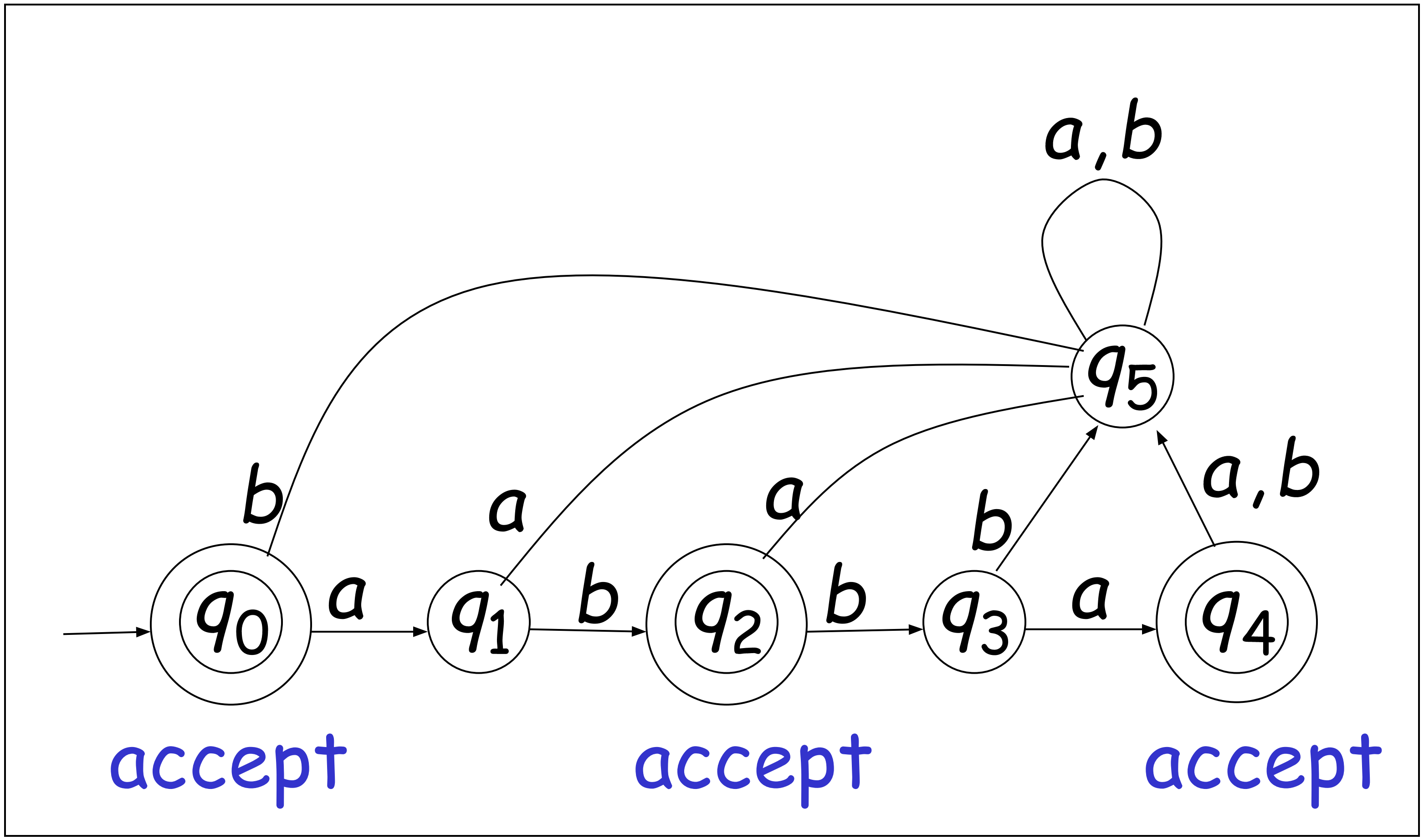
$$EVEN = \{x : x \in \Sigma^{*} \text{ and } |x| \text{ is even}\}$$

$$= \{\lambda, 11, 1111, 111111, \square\ \}$$

Example # 3

$L(M) = \{\lambda, ab, abba\}$

M



Languages Accepted by DFAs

Take DFA M

Definition:

The language $L(M)$ contains all input strings accepted by M

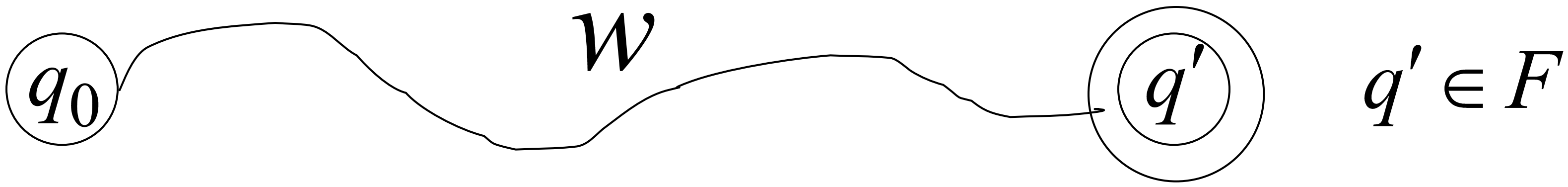
$L(M)$ = { strings that drive M to a final state}

Formally

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

Language accepted by M :

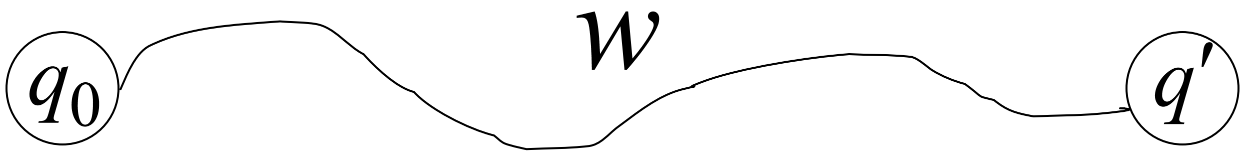
$$L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$$



Observation

Language rejected by M :

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



$q' \notin F$

Regular Languages

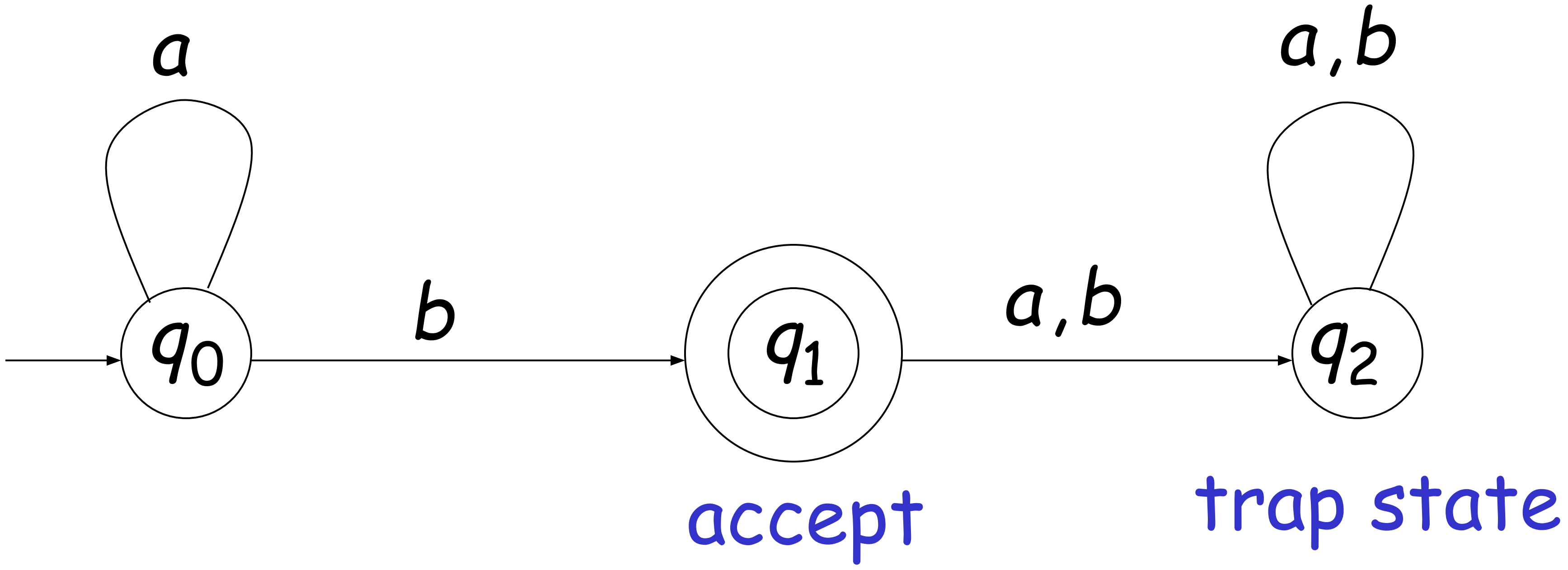
Definition:

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

Activity # 1

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

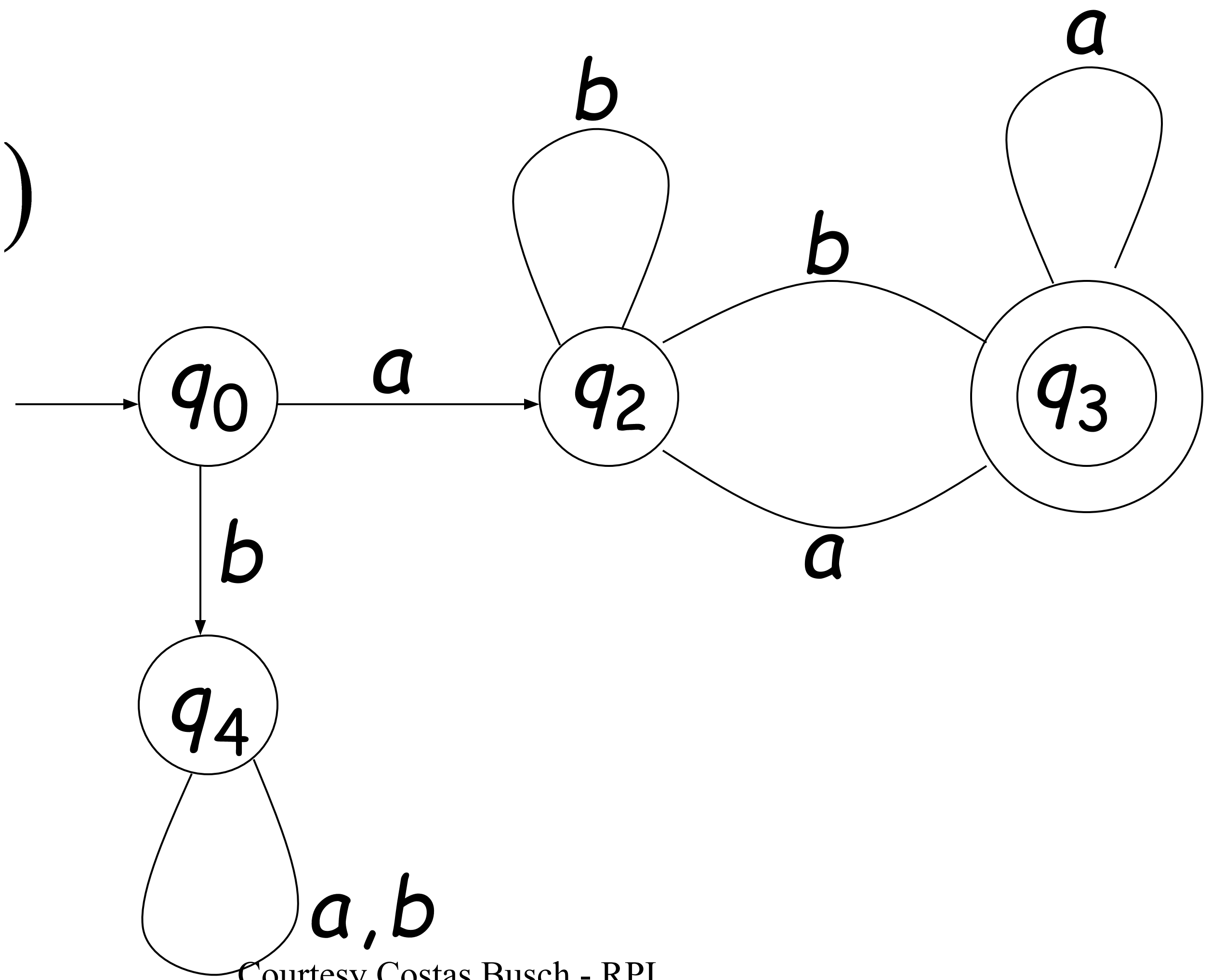


Activity # 2

The language is regular:

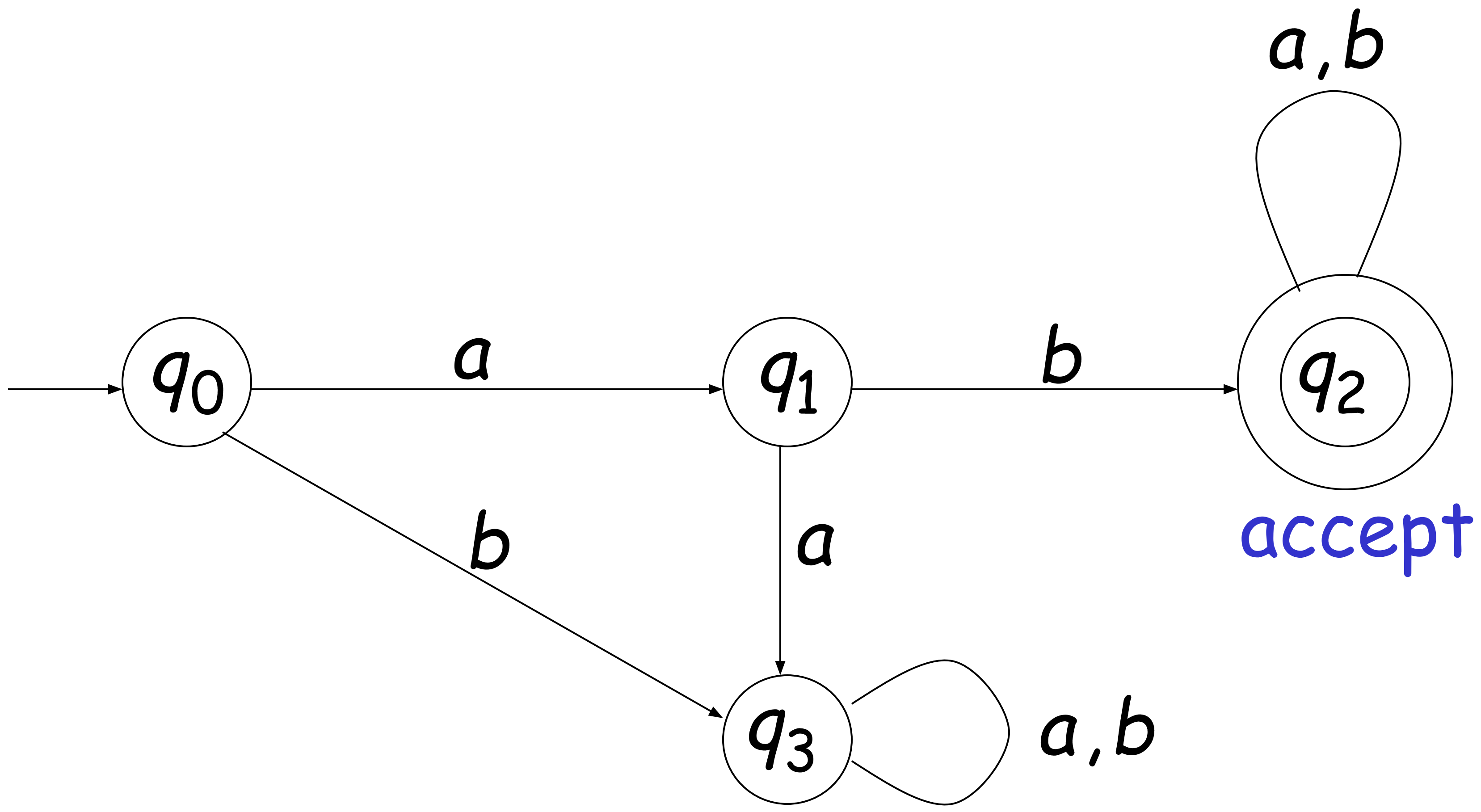
$$L = \{awa : w \in \{a,b\}^*\}$$

$$L = L(M)$$



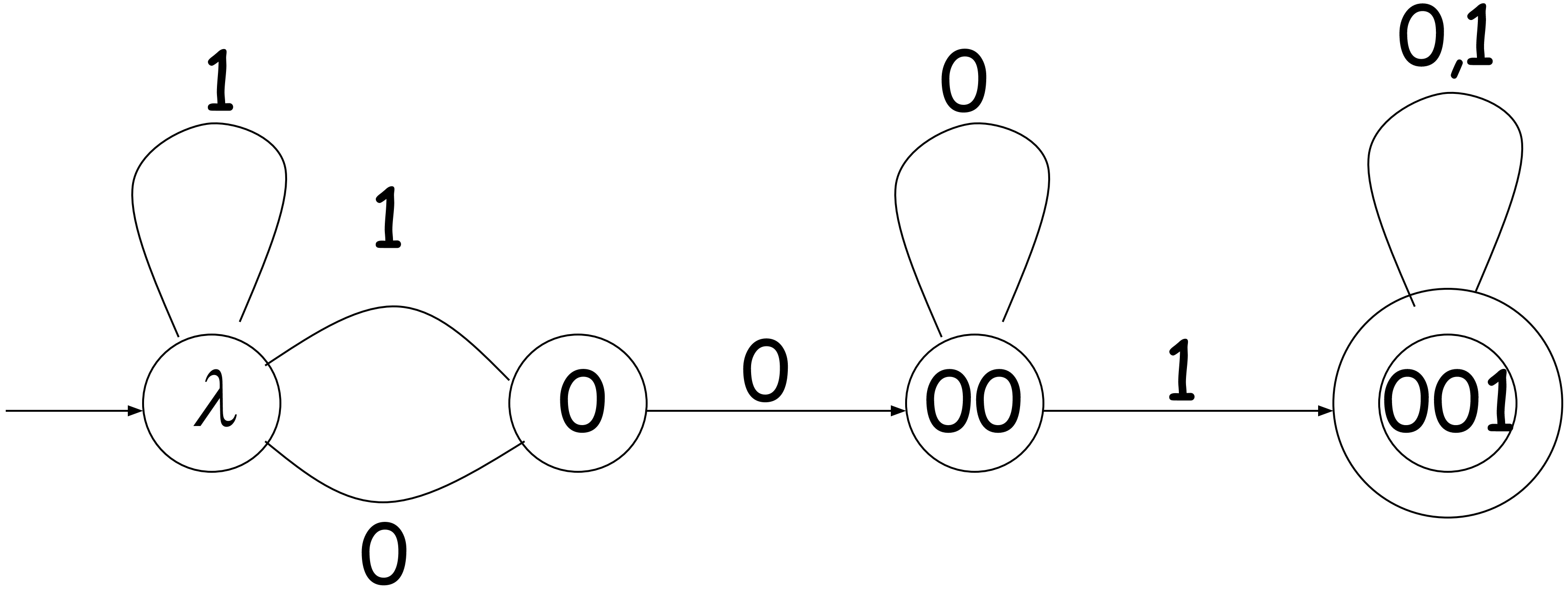
Activity # 3

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



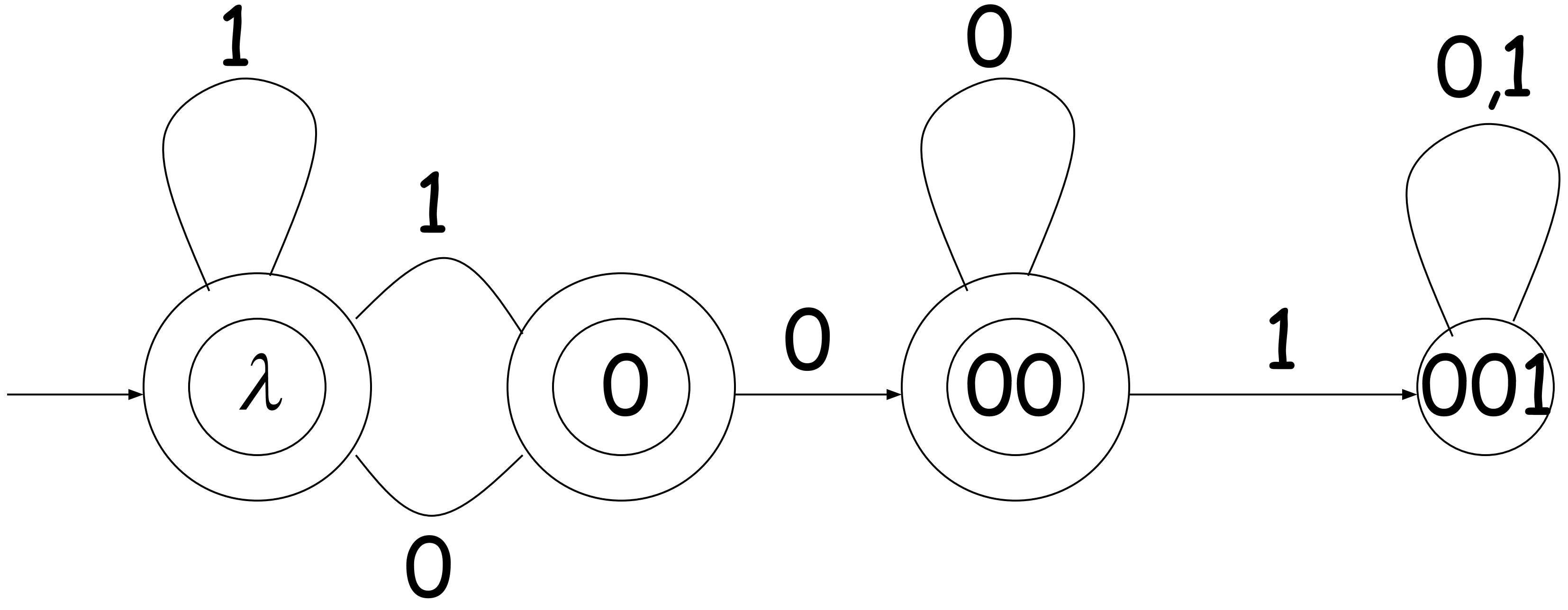
Activity # 4

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



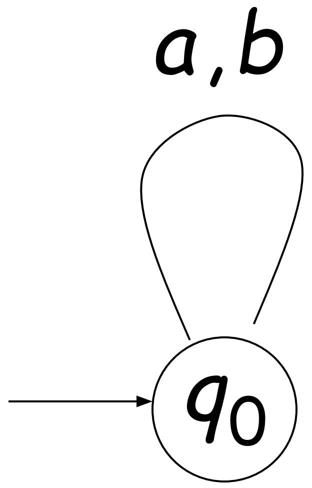
Activity # 5

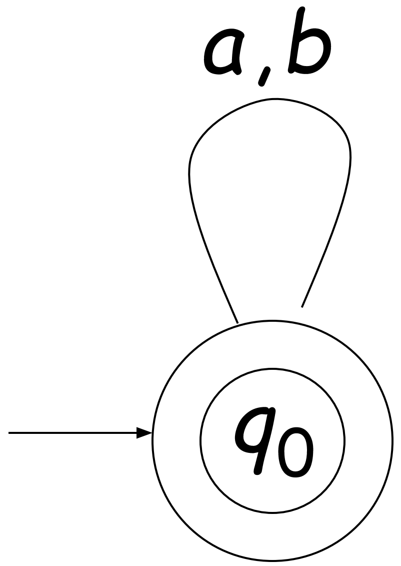
$L(M)$ = { all binary strings without substring 001 }



Activity # 6 & 7

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





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

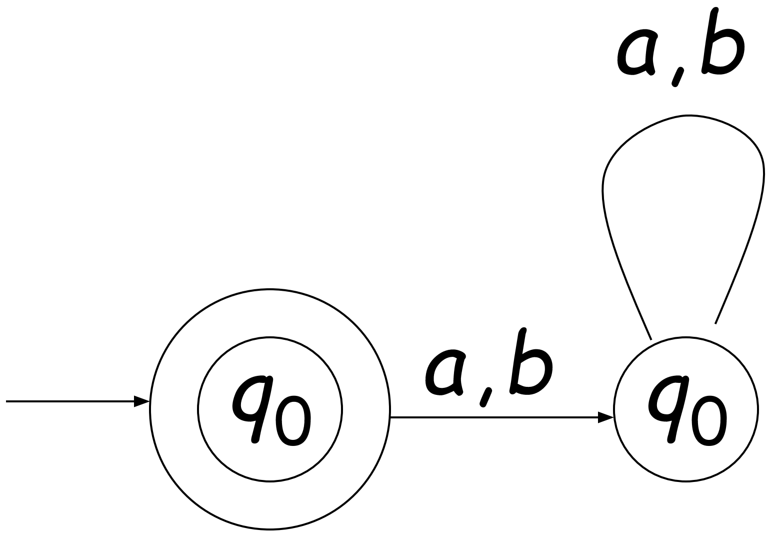
Empty language

$L(M) = \Sigma^*$

All strings

Activity # 8

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



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

Language of the empty string

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

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

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

There are no DFAs that accept these languages

(we will prove this in a later class)

Task

- Build an FA accepting the Language L of Strings, defined over $\Sigma = \{a, b\}$, **beginning with and ending in same letters.**

  Solution: The language L may be expressed by the following regular expression

  $$(a+b)+a(a + b)^{*}a + b(a + b)^{*}b$$

  This language L may be accepted by the following FA

beginning with and ending in same letters.

$$\begin{tikzcd}
 & 2+ \arrow[loop above, "a"] \arrow[r, "b"] & 4 \arrow[loop above, "b"] \arrow[r, dash, bend left, "a"] \arrow[r, dash, bend right, "b"'] & 6+ \arrow[loop above, "a"] \\
1- \arrow[ur, "a"] \arrow[dr, "b"'] & & & \\
 & 3+ \arrow[loop above, "b"] \arrow[r, "a"] & 5 \arrow[loop above, "a"] \arrow[r, dash, bend left, "b"] \arrow[r, dash, bend right, "a"'] & 7+ \arrow[loop above, "b"]
\end{tikzcd}$$

Example

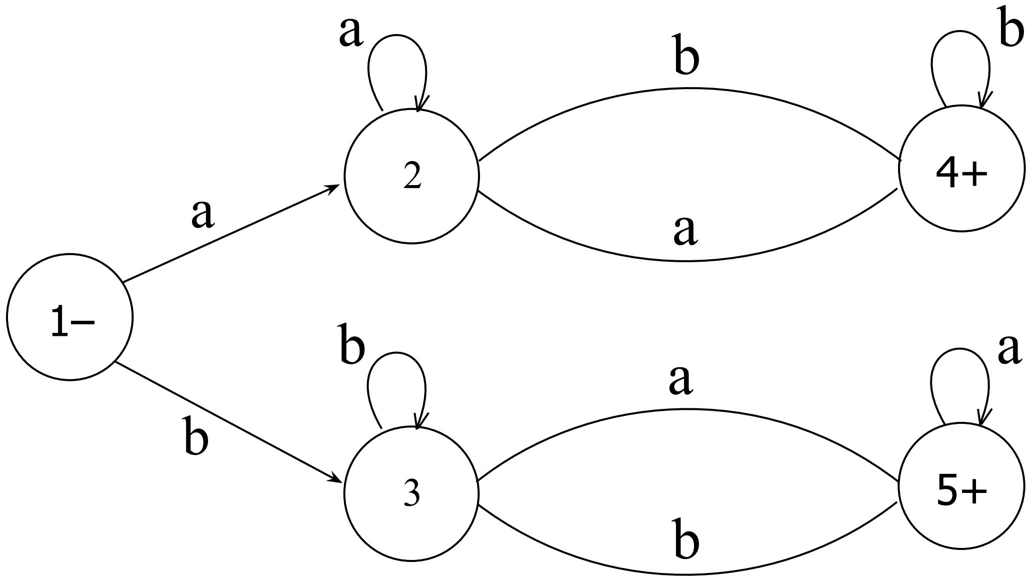
Consider the Language L of Strings , defined over Σ = {a, b}, **beginning with and ending in different letters.**

The language L may be expressed by the following regular expression

$$a\,(a + b)^{*}\,b + b\,(a + b)^{*}\,a$$

This language may be accepted by the following FA

beginning with and ending in different letters.



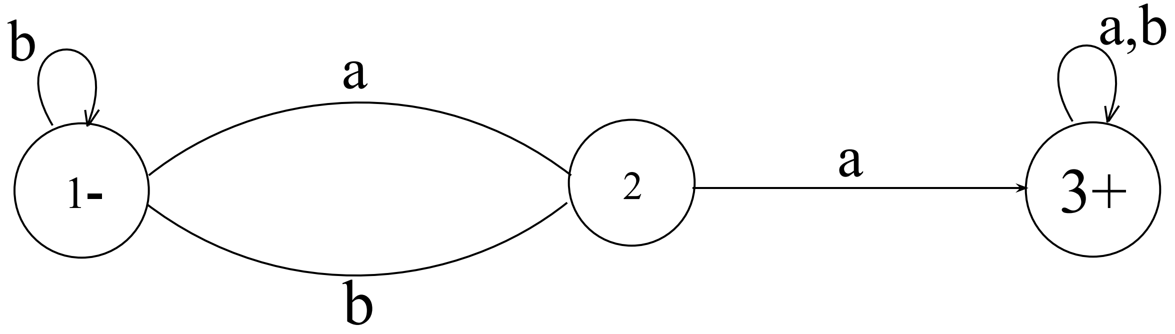
Example

Consider the Language L of strings , defined over Σ = {a, b}, **containing double a.**

The language L may be expressed by the following regular expression

$(a+b)^* (aa) (a+b)^*$. This language may be accepted by the following FA

containing double a.

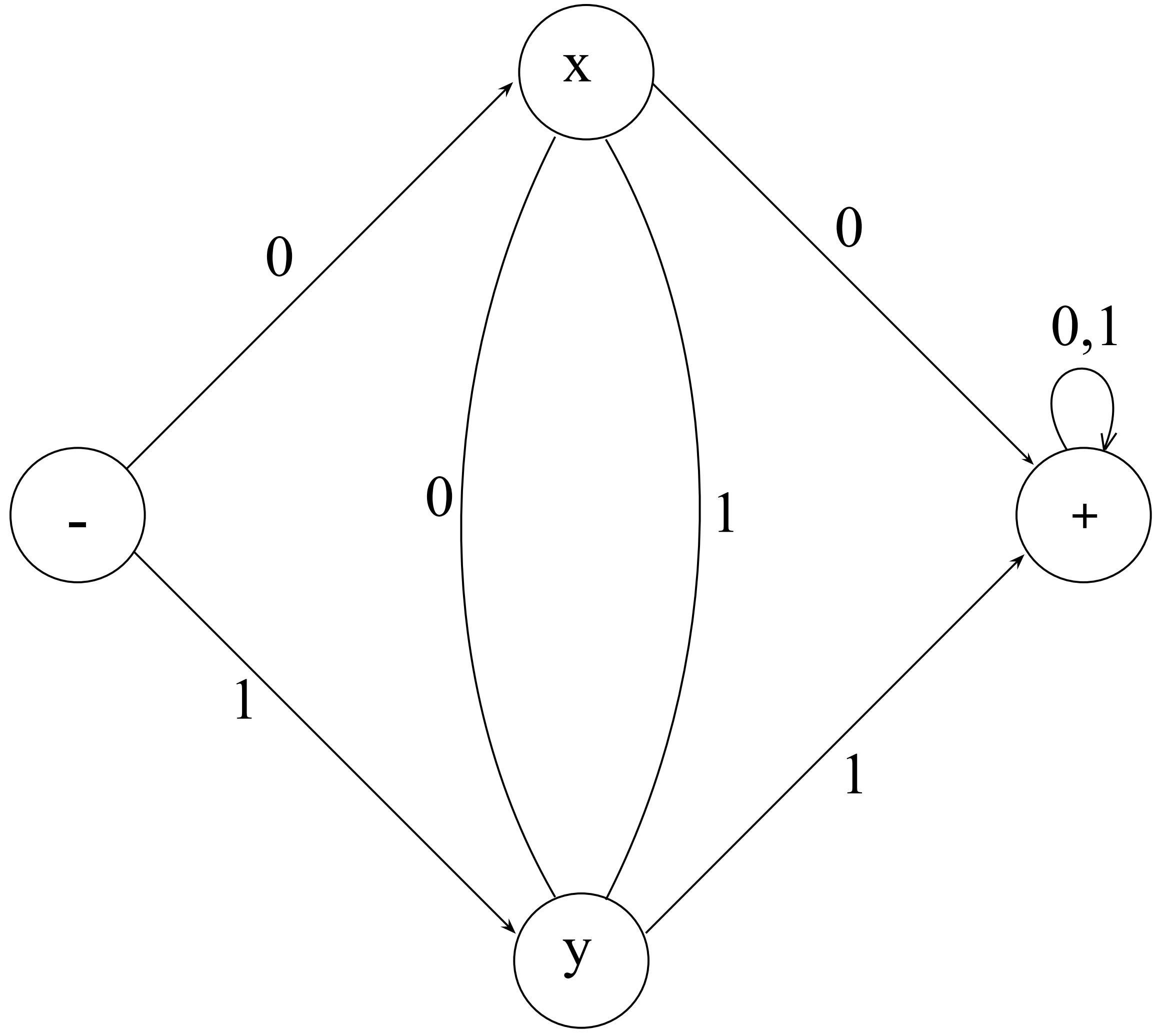


Example

Consider the language L of strings, defined over $\Sigma=\{0, 1\}$, **having double 0's or double 1's,** The language L may be expressed by the regular expression $(0+1)^*(00 + 11)(0+1)^*$

This language may be accepted by the following FA

having double 0's or double 1's,



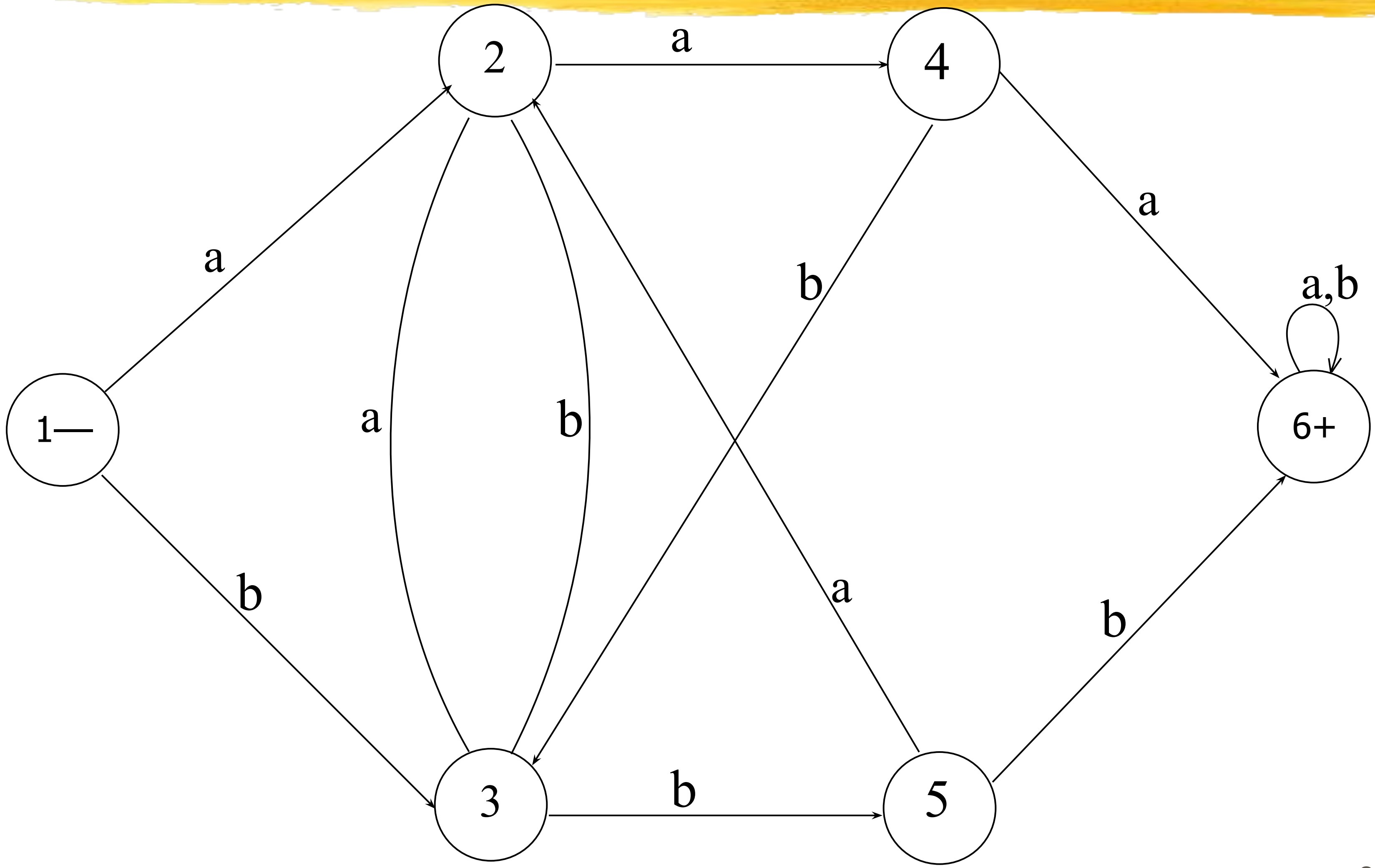
Example

Consider the language L of strings, defined over $\Sigma=\{a, b\}$, **having triple a's or triple b's.**
The language L may be expressed by RE

$$(a+b)^* (aaa + bbb) (a+b)^*$$

This language may be accepted by the following FA

having triple a’s or triple b’s.

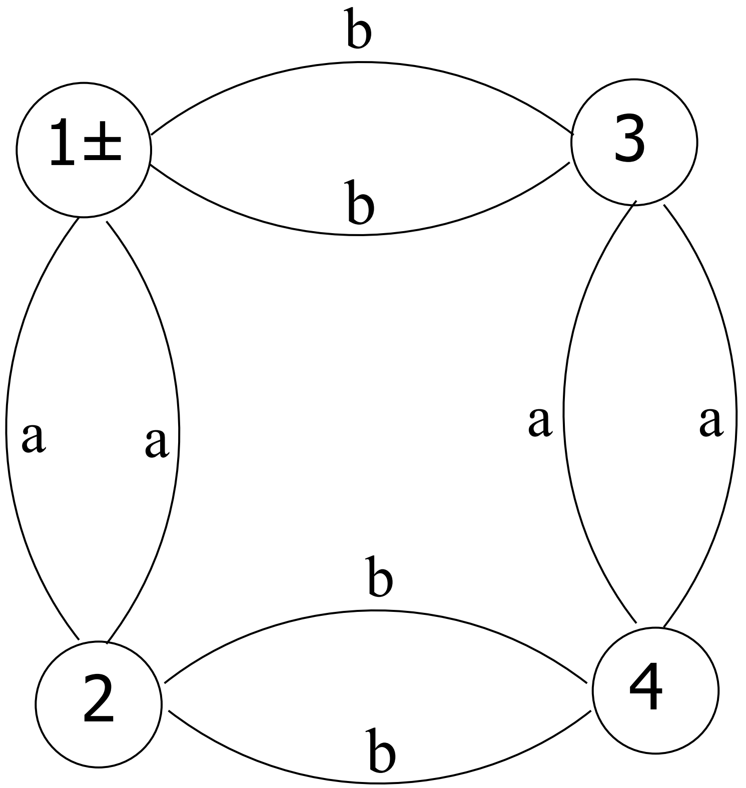


Example

- Consider the **EVEN-EVEN** language, defined over Σ={a, b}. As discussed earlier that **EVEN-EVEN** language can be expressed by the regular expression $(aa+bb+(ab+ba)(aa+bb)^*(ab+ba))^*$

  EVEN-EVEN language may be accepted by the following FA

EVEN-EVEN language

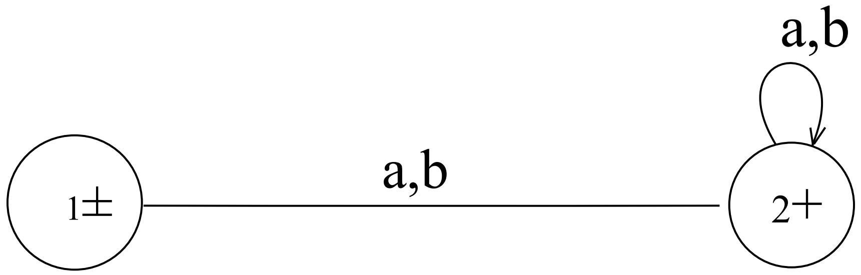


Summing Up

- **Language of strings beginning with and ending in different letters, Accepting all strings, accepting non-empty strings, accepting no string, containing double a’s, having double 0’s or double 1’s, containing triple a’s or triple b’s, EVEN-EVEN**

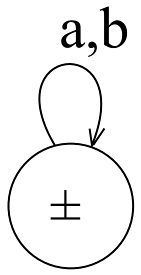
Example

- Consider the Language L , defined over Σ = {a, b} of **all strings including Λ**, The language L may be accepted by the following FA



- The language L may also be accepted by the following FA

Example Continued …

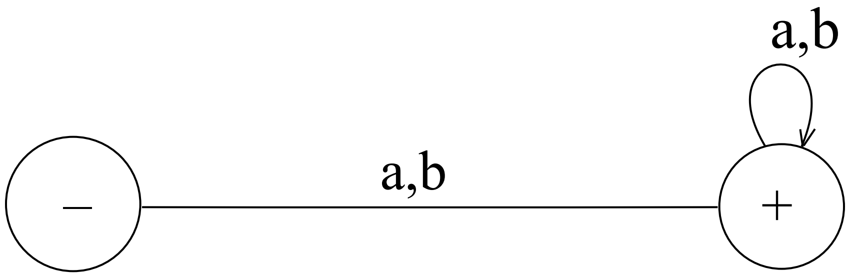


- The language L may be expressed by the following regular expression

$(a + b)^*$

Example

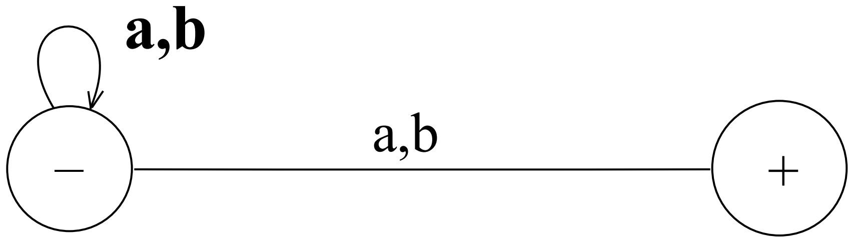
- Consider the Language L , defined over Σ = {a, b} of **all non empty strings**. The language L may be accepted by the following FA



The above language may be expressed by the following regular expression $(a + b)^+$

Example

- Consider the following FA, defined over Σ = {a, b}



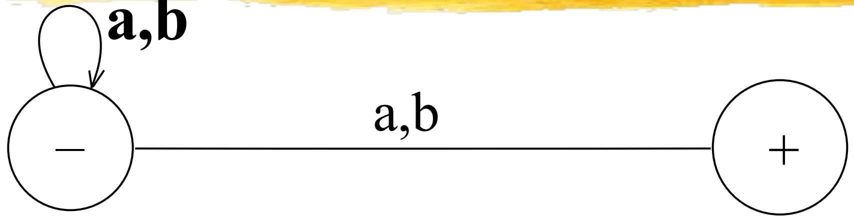
- It is to be noted that the above FA **does not accept any string**. Even it does not accept the null string. As there is no path starting from initial state and ending in final state.

Equivalent FAs

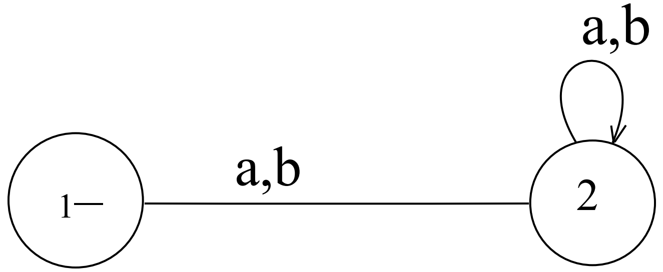
- It is to be noted that two FAs are said to be equivalent, if they accept the same language, as shown in the following FAs.

Equivalent FAs Continued …

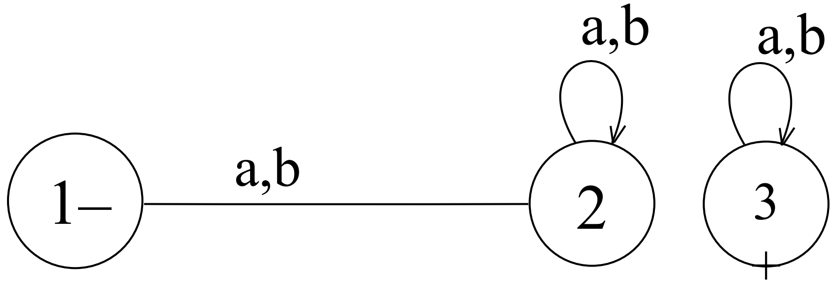
FA_1



FA_2



FA_3



FA corresponding to finite languages

- **Example**

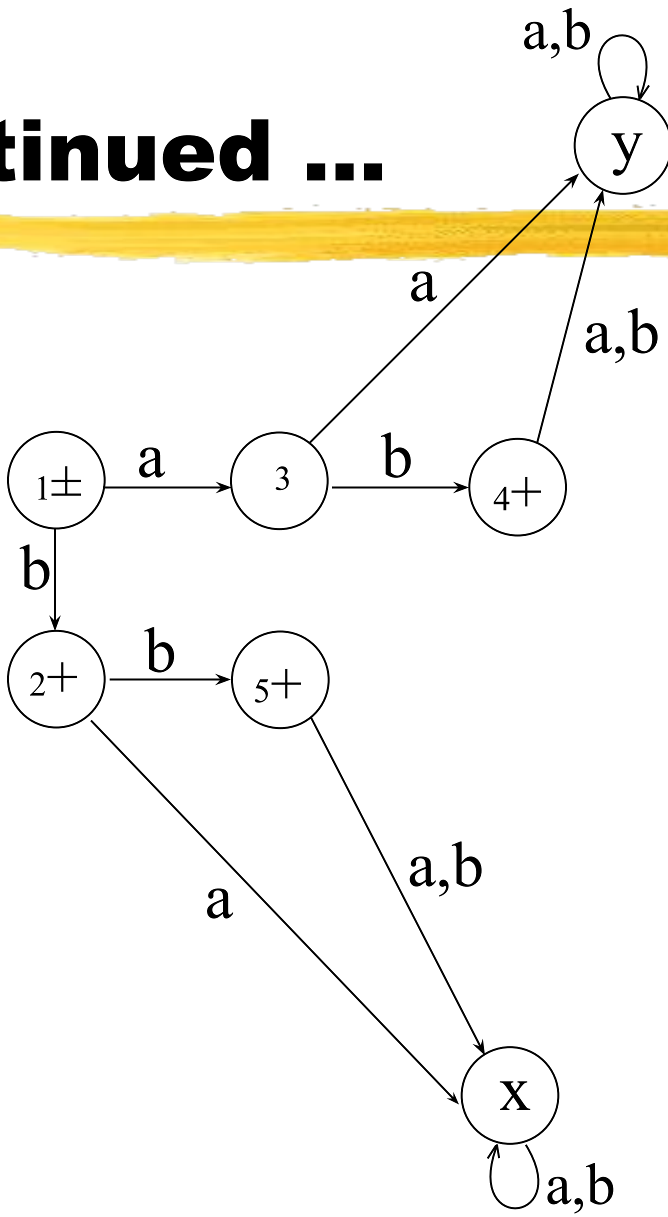
  Consider the language

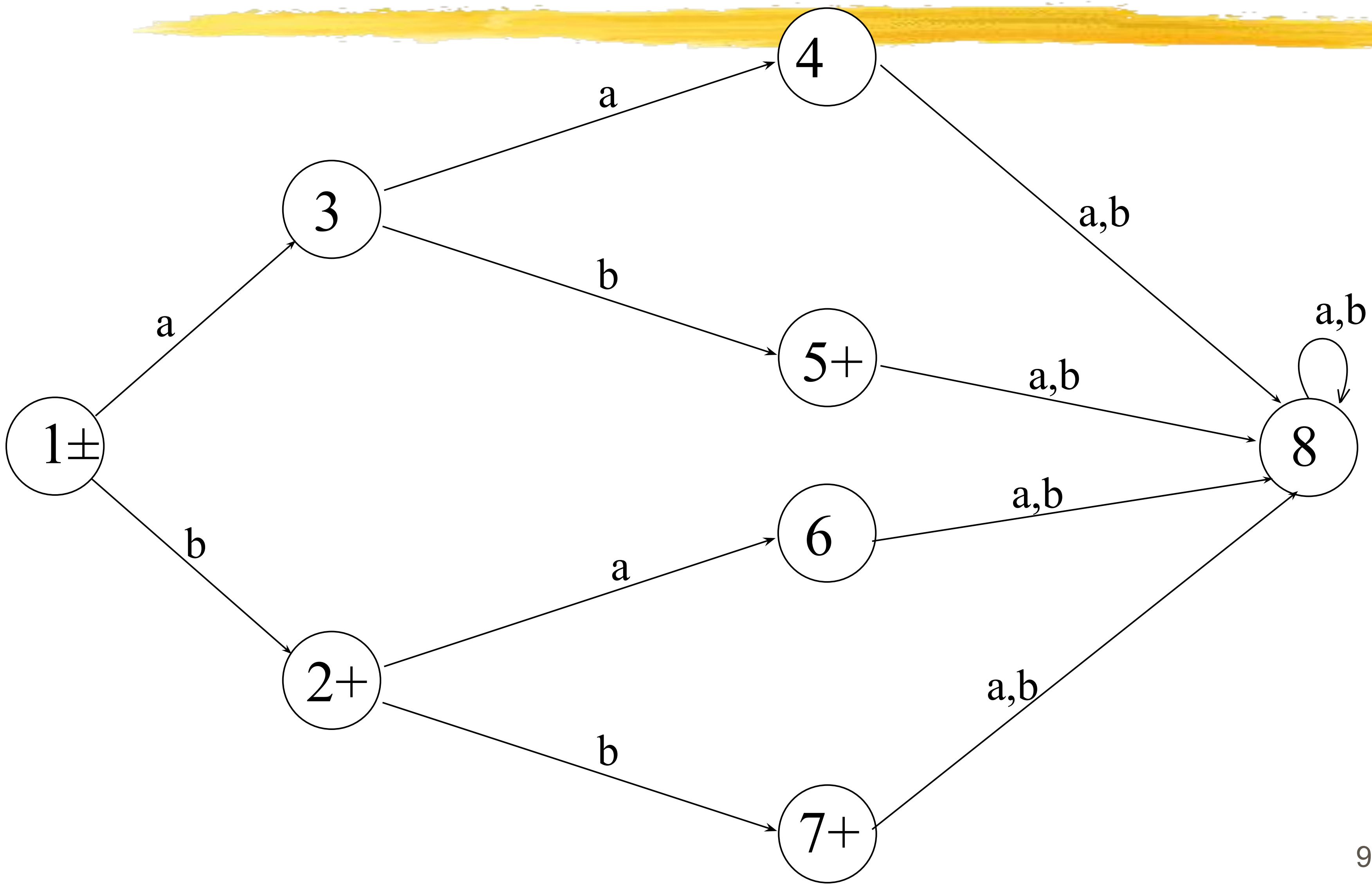
  L = {Λ, b, ab, bb}, defined over

  Σ ={a, b}, expressed by Λ + b + ab + bb OR Λ + b (Λ + a + b).

  The language L may be accepted by the following FA

Example continued ...




1±
2+
3
4
5+
6
7+
8
a
b
a
b
a
b
a,b
a,b
a,b
a,b
a,b

Example

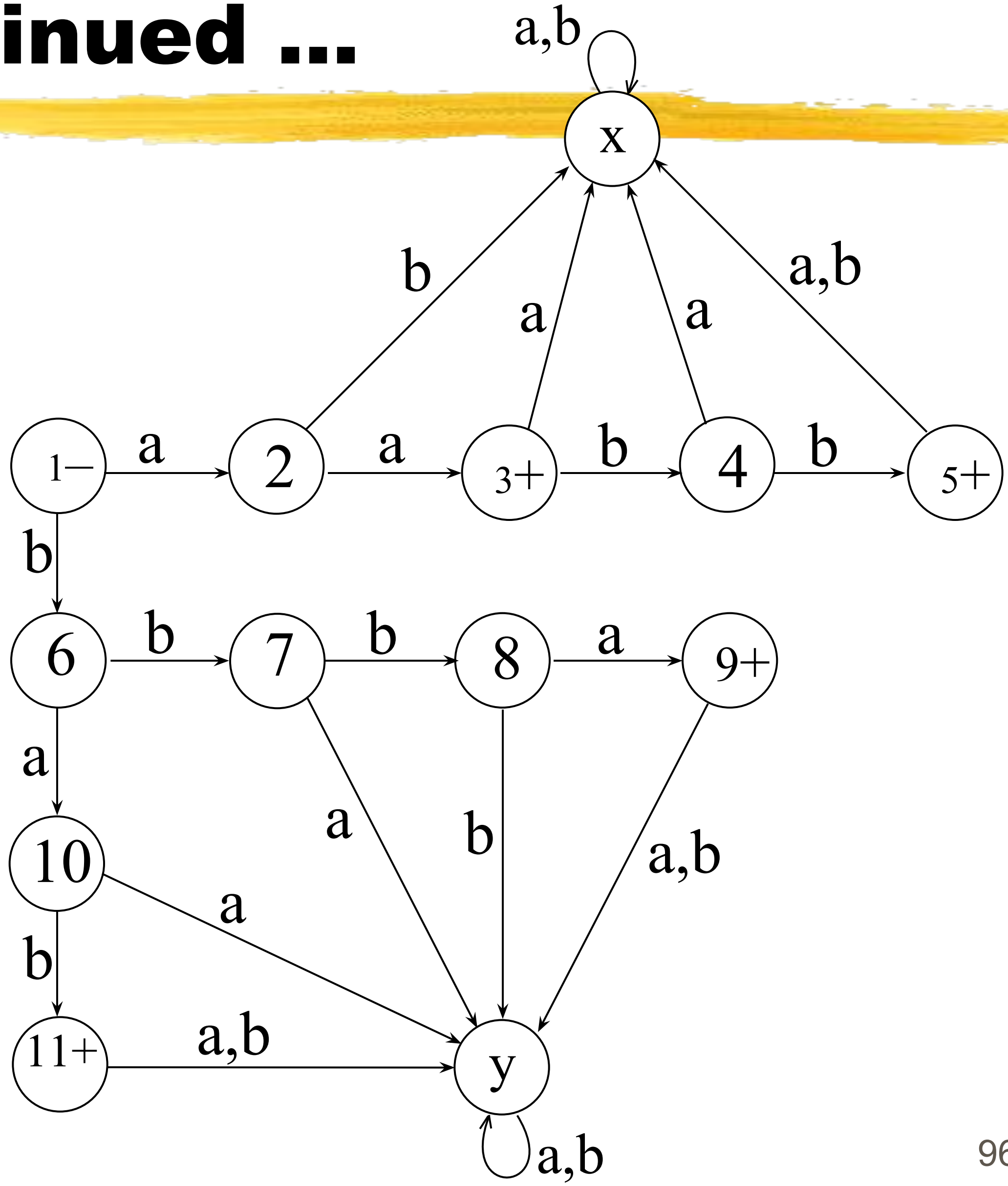
Consider the language

L = {aa, bab, aabb, bbba}, defined over

Σ ={a, b}, expressed by aa + bab + aabb + bbba OR aa (Λ + bb) + b (ab + bba)

The above language may be accepted by the following FA

Example Continued …



Example

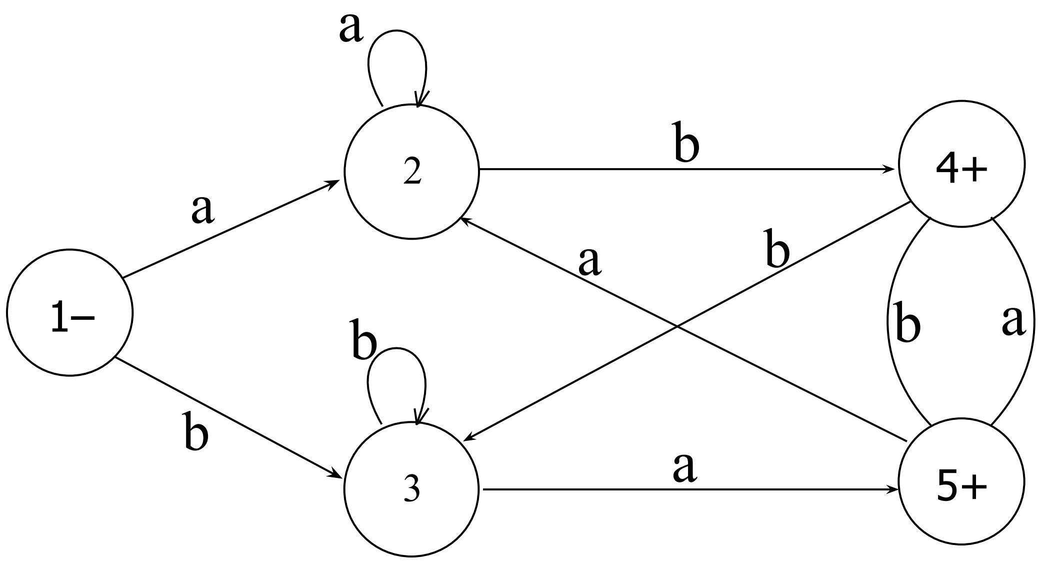
Consider the language L = {w belongs to $\{a,b\}^*$: length(w) >= 2 and w neither ends in **aa** nor **bb**}.

The language L may be expressed by the regular expression

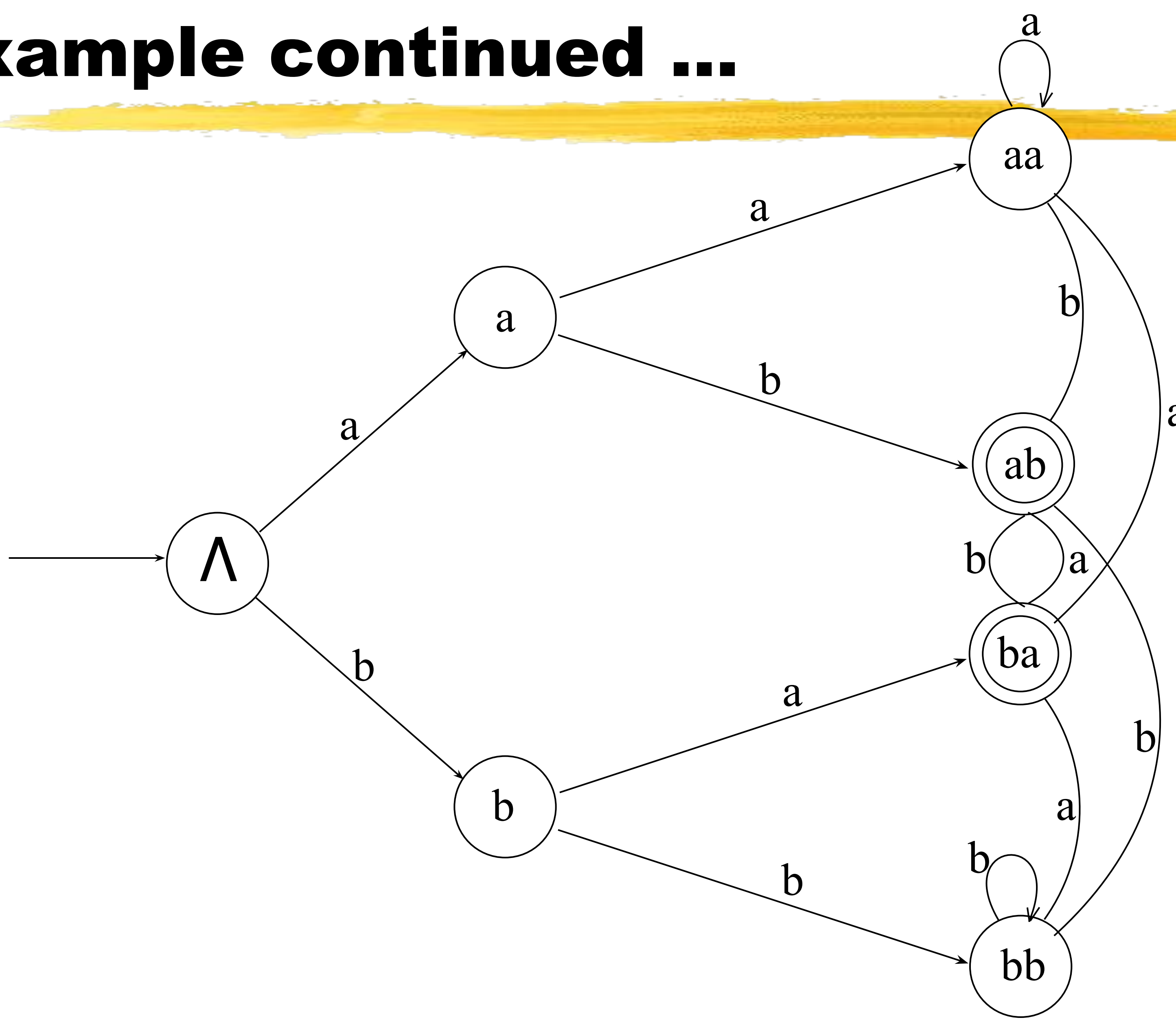
$(a+b)^*(ab+ba)$

This language may be accepted by the following FA

Example Continued …



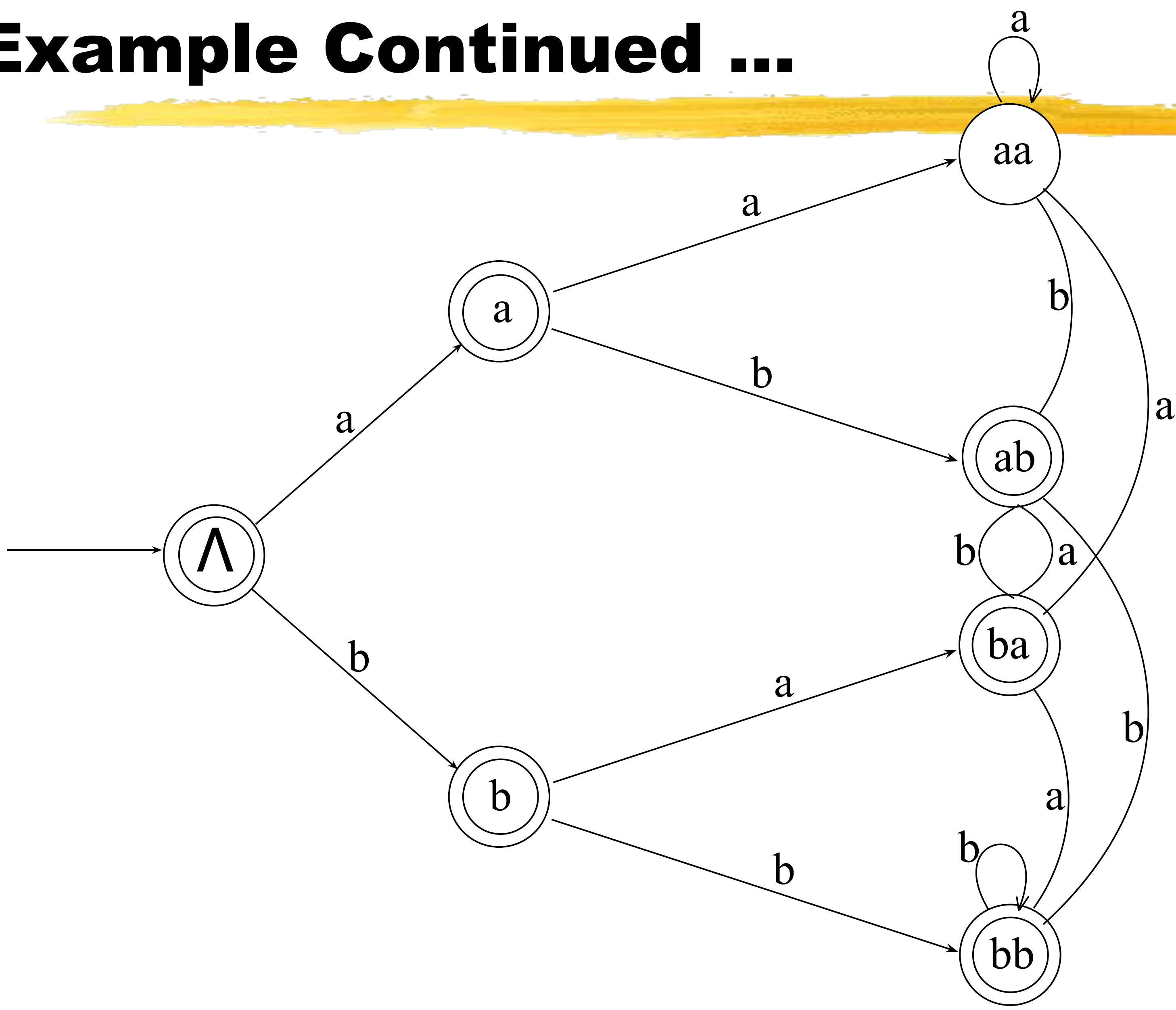
Example continued ...



Example

- Consider the language
  L = {w belongs to $\{a,b\}^*$: w does not end in **aa**}.
  The language L may be expressed by the regular expression
  $\Lambda + a + b + (a+b)^*(ab+ba+bb)$
  This language may be accepted by the following FA

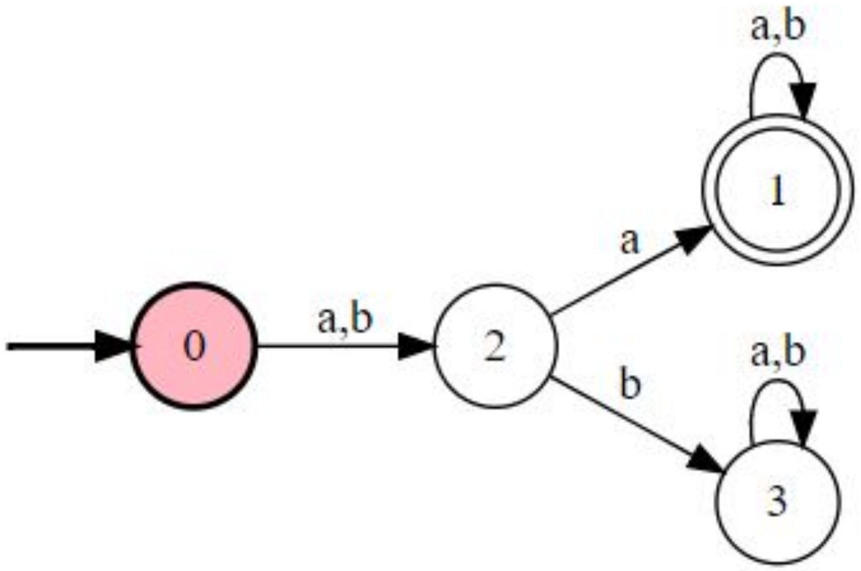
Example Continued …



Task

L = {w belongs to {a,b}*: length(w) >= 2 and second letter of w, from right is a}.

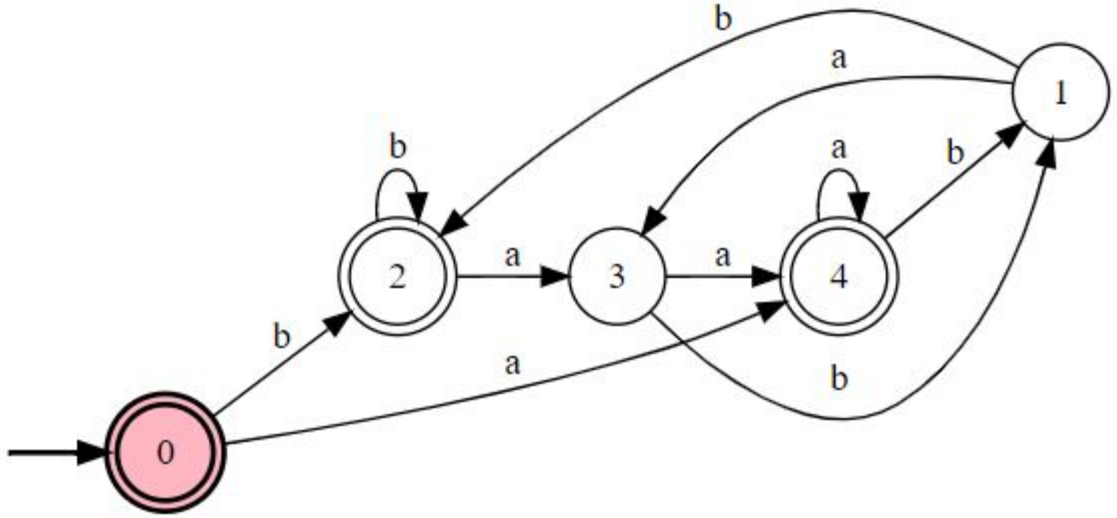
$$(\boldsymbol{a}+\boldsymbol{b})\boldsymbol{a}(\boldsymbol{a}+\boldsymbol{b})^*$$


a,b
a,b
a
b
a,b
0
2
1
3

Task

L = {w belongs to {a,b}: w neither ends in **ab** nor **ba**}.

$$(a + b)a(a + b)^*$$


b
a
1
b
a
b
2
a
3
a
4
b
a
b
0

Defining Languages (continued)…

- **Method 5 (Transition Graph)**

  Definition: A Transition graph (TG), is a collection of the followings

  1) Finite number of states, at least one of which is start state and some (maybe none) final states.
  2) Finite set of input letters (Σ) from which input strings are formed.
  3) Finite set of transitions that show how to go from one state to another based on reading specified substrings of input letters, possibly even the null string (Λ).