“Theory of Computation”

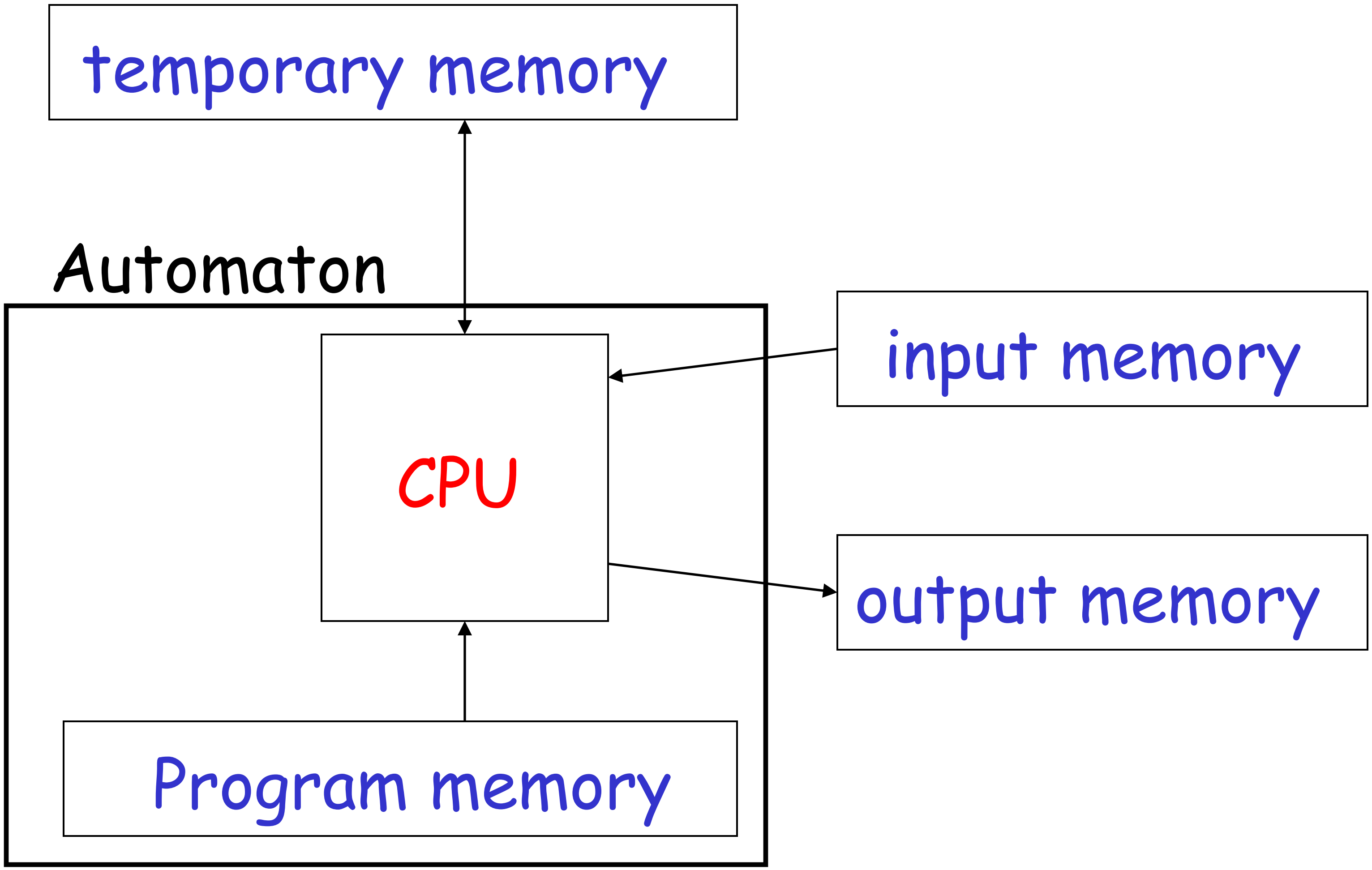
Part2

Automata and Language Hierarchy

Lecturer: รศ.ดร. เกียรติกูล เจียรนัยธนะกิจ

Email: kietikul@yahoo.com, kjkietik@kmitl.ac.th

Automaton



temporary memory
Automaton
CPU
input memory
output memory
Program memory

Different Kinds of Automata

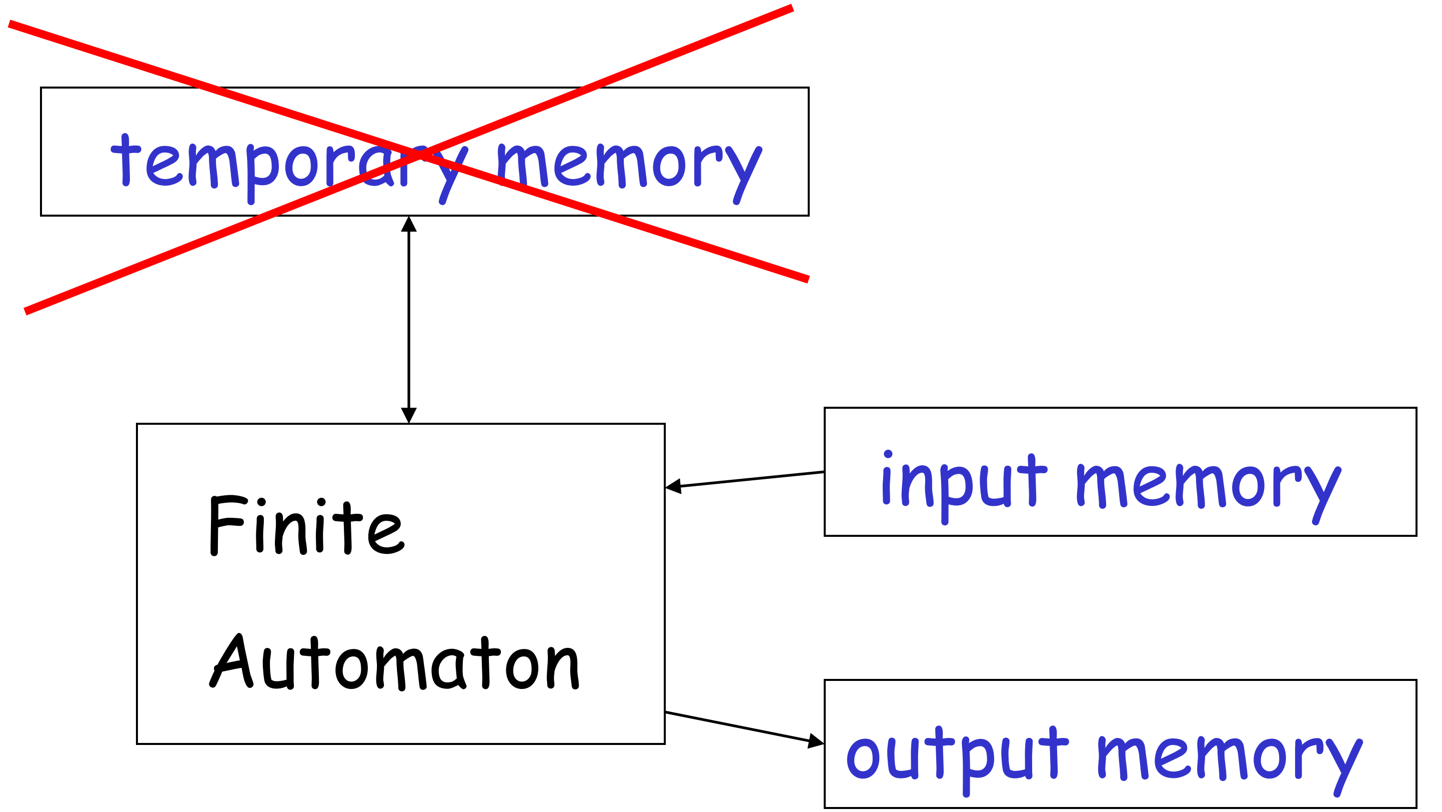
Automata are distinguished by the temporary memory

- **Finite Automata**: no temporary memory

- **Pushdown Automata**: stack

- **Turing Machines**: random access memory

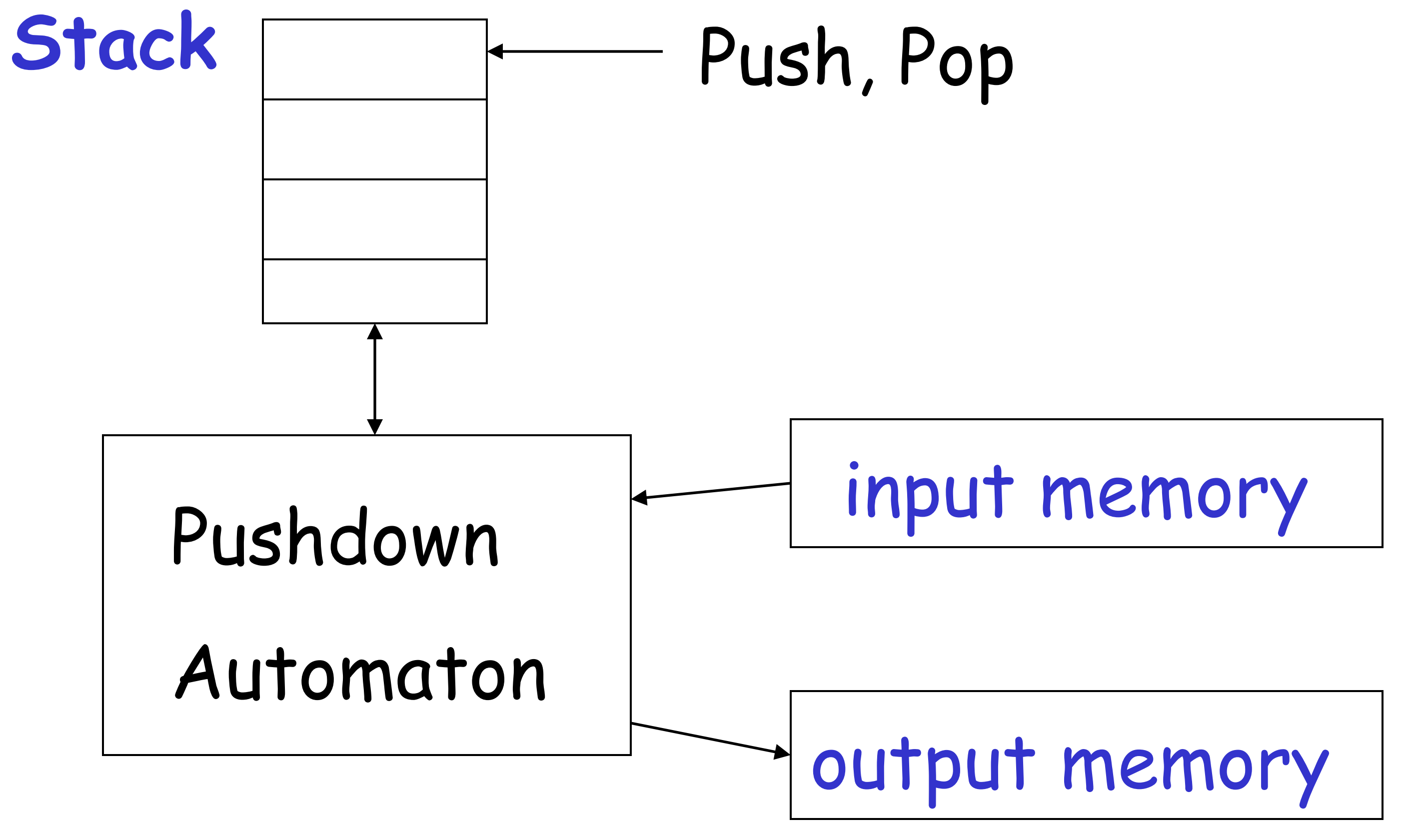
Finite Automaton



Example: Vending Machines

(small computing power)

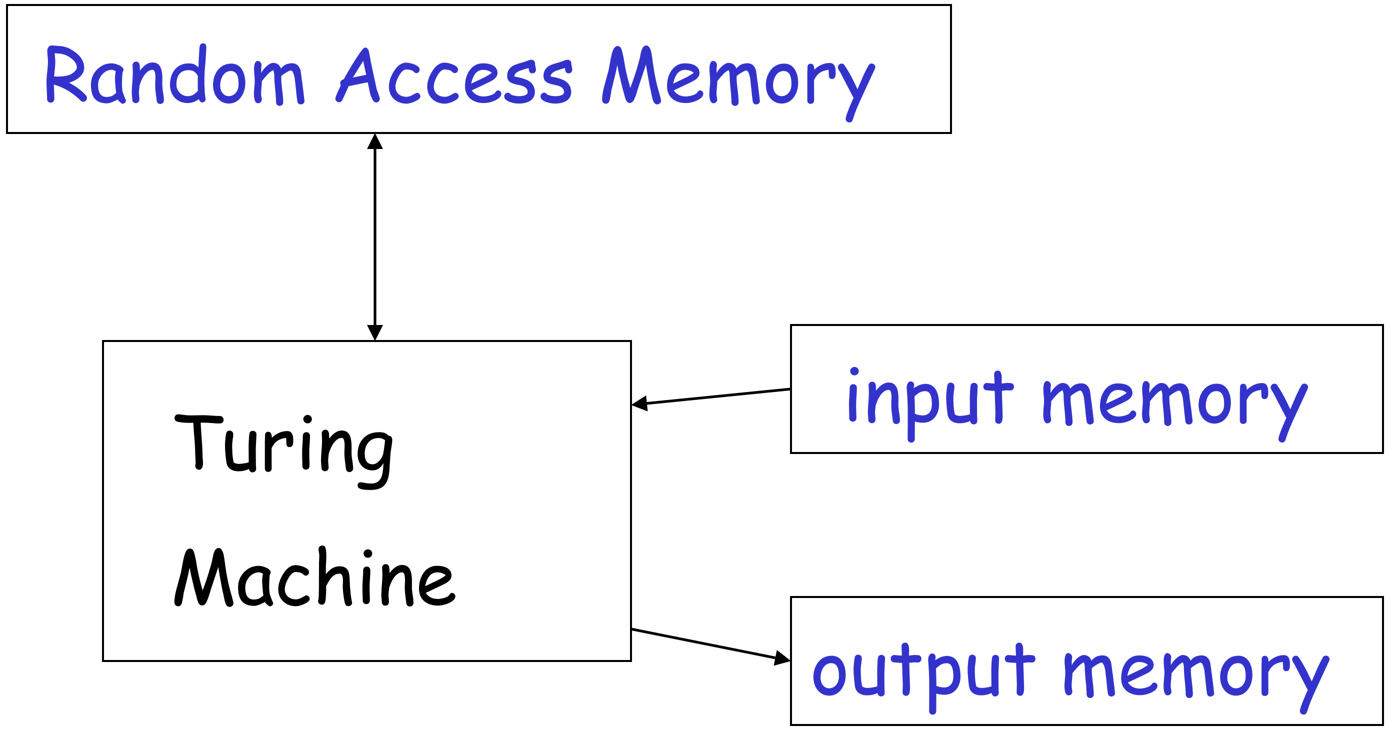
Pushdown Automaton



Example: Compilers for Programming Languages

(medium computing power)

Turing Machine



Examples: Any Algorithm

(highest computing power)

Power of Automata

Finite Automata $<$ Pushdown Automata $<$ Turing Machine

Less power $\longrightarrow$ More power

Solve more computational problems

Languages

A language is a set of strings

String: A sequence of letters

Examples: "cat", "dog", "house", ...

Defined over an alphabet:

$$\Sigma = \{a, b, c, \ldots, z\}$$

Alphabets and Strings

We will use small alphabets: $\Sigma = \{a, b\}$

Strings

a

ab $\qquad u = ab$

$abba$ $\qquad v = bbbaaa$

$baba$ $\qquad w = abba$

$aaabbbaabab$

String Operations

$w = a_1 a_2 \cdots a_n$ $\qquad$ $abba$

$v = b_1 b_2 \cdots b_m$ $\qquad$ $bbbaaa$

Concatenation

$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$ $\qquad$ $abbabbbaaa$

$$w = a_1 a_2 \cdots a_n$$

$$ababaaabbb$$

Reverse

$$w^R = a_n \cdots a_2 a_1$$

$$bbbaaababa$$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length: $|w| = n$

Examples: $|abba| = 4$

$$|aa| = 2$$

$$|a| = 1$$

Empty String

A string with no letters: λ

Observations: $|\lambda| = 0$

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = abba$$

Substring

Substring of string:

a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$abbab$

Prefixes	Suffixes
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$$w = \underbrace{u}_{\text{prefix}}\underbrace{v}_{\text{suffix}}$$

Another Operation

$$w^n = \underbrace{ww \cdots w}_{n}$$

Example: $$(abba)^2 = abbaabba$$

Definition: $$w^0 = \lambda$$

$$(abba)^0 = \lambda$$

The * Operation

Σ^*: the set of all possible strings from alphabet Σ

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

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

The + Operation

Σ^+ : the set of all possible strings from alphabet Σ except λ

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

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^+ = \Sigma^* - \lambda$$

$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Languages

A language is any subset of Σ^*

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

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Languages: $\{\lambda\}$

$$\{a, aa, aab\}$$

$$\{\lambda, abba, baba, aa, ab, aaaaaa\}$$

Note that:

Sets $\emptyset = \{\ \} \neq \{\lambda\}$

Set size $|\{\ \}| = |\emptyset| = 0$

Set size $|\{\lambda\}| = 1$

String length $|\lambda| = 0$

Another Example

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

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad abb \notin L$$

Operations on Languages

The usual set operations

$$\{a, ab, aaaa\} \bigcup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \bigcap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

Complement: $\overline{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

Reverse

Definition: $$L^R = \{w^R : w \in L\}$$

Examples: $$\{ab, aab, baba\}^R = \{ba, baa, abab\}$$

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

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

Concatenation

Definition: $$L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$$

Example: $$\{a, ab, ba\}\{b, aa\}$$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

Another Operation

Definition: $$L^n = \underbrace{LL\cdots L}_{n}$$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$
$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

Special case: $L^0 = \{\lambda\}$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$

More Examples

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

$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$

$$aabbaaabbb \in L^2$$

Star-Closure (Kleene *)

Definition: $L^* = L^0 \cup L^1 \cup L^2 \cdots$

Example:

$$\{a, bb\}^* = \left\{ \begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

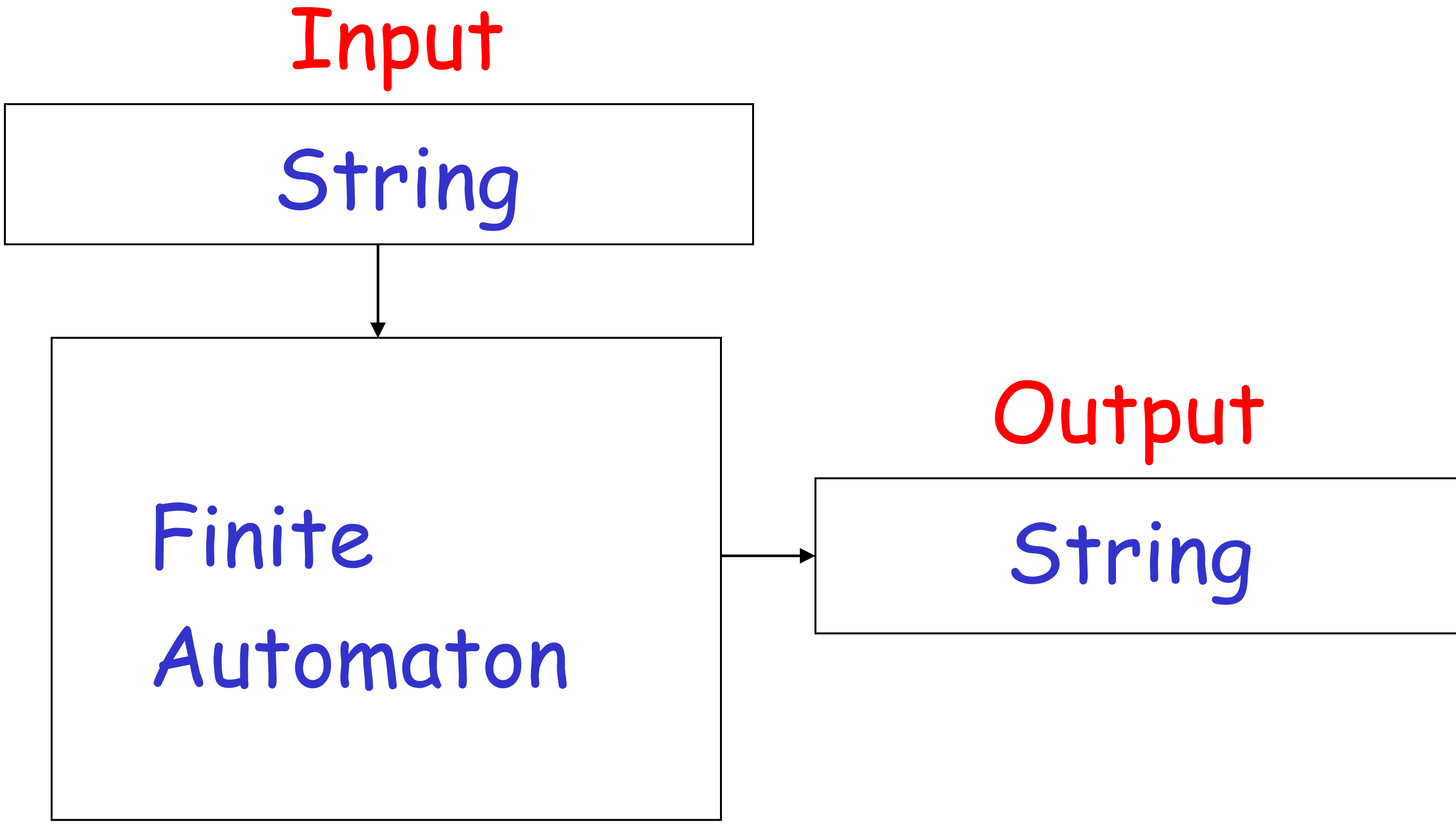
Positive Closure

Definition:
$$L^{+} = L^{1} \cup L^{2} \cup \cdots$$
$$= L^{*} - \{\lambda\}$$

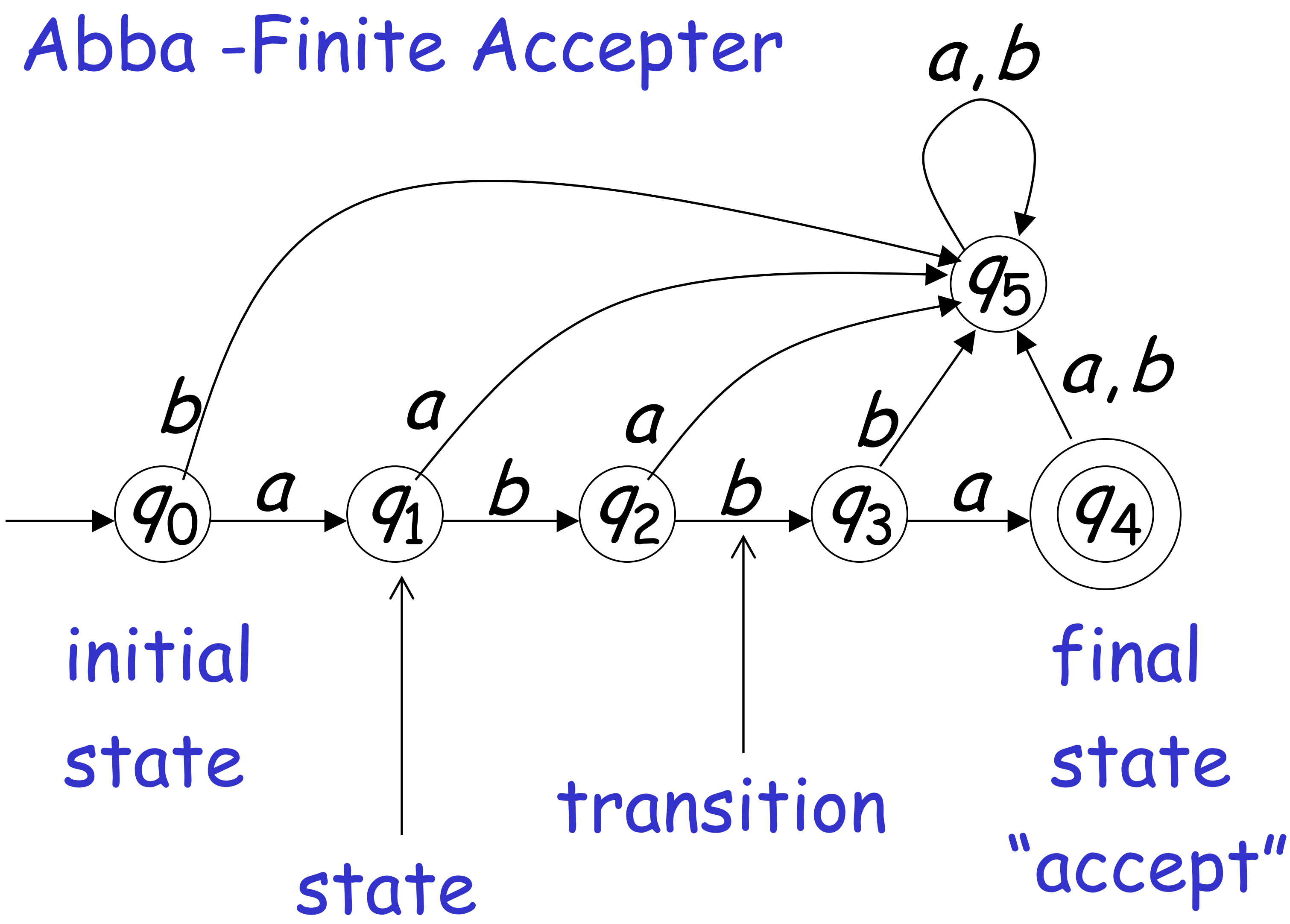
$$\{a, bb\}^{+} = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

Finite Automata

Finite Automaton



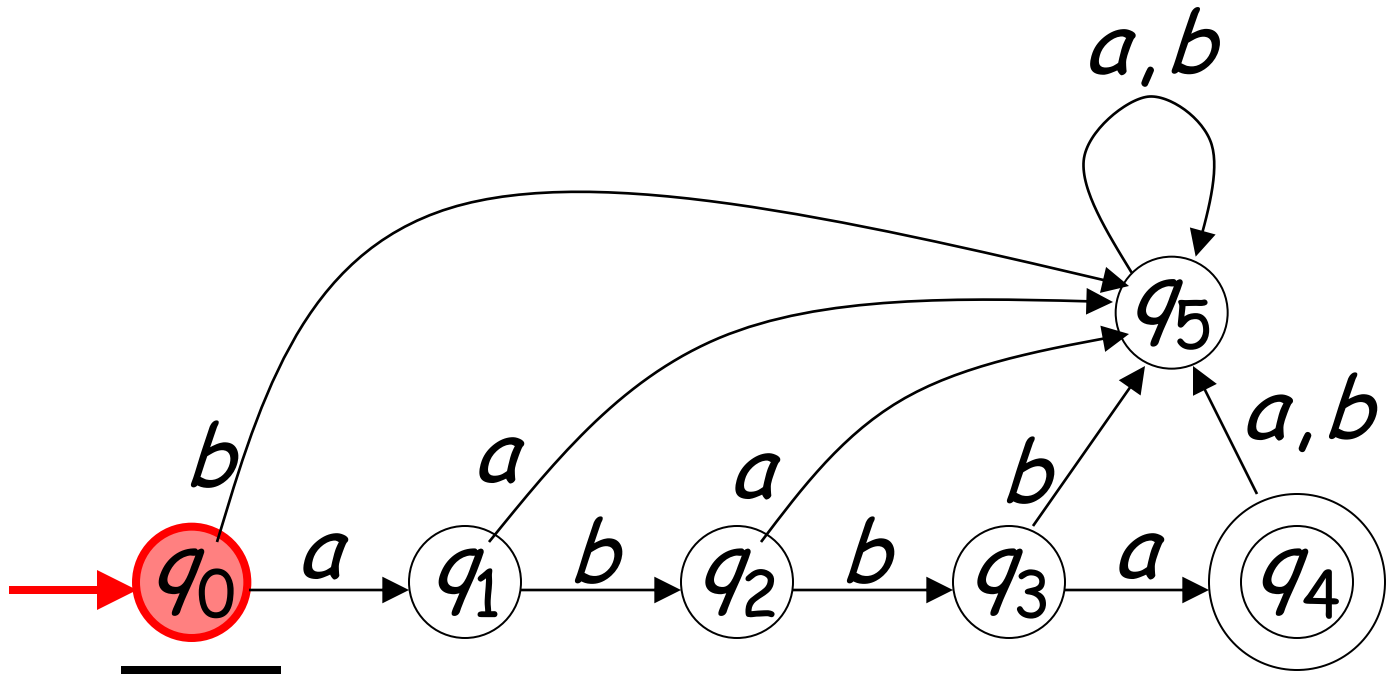
Transition Graph



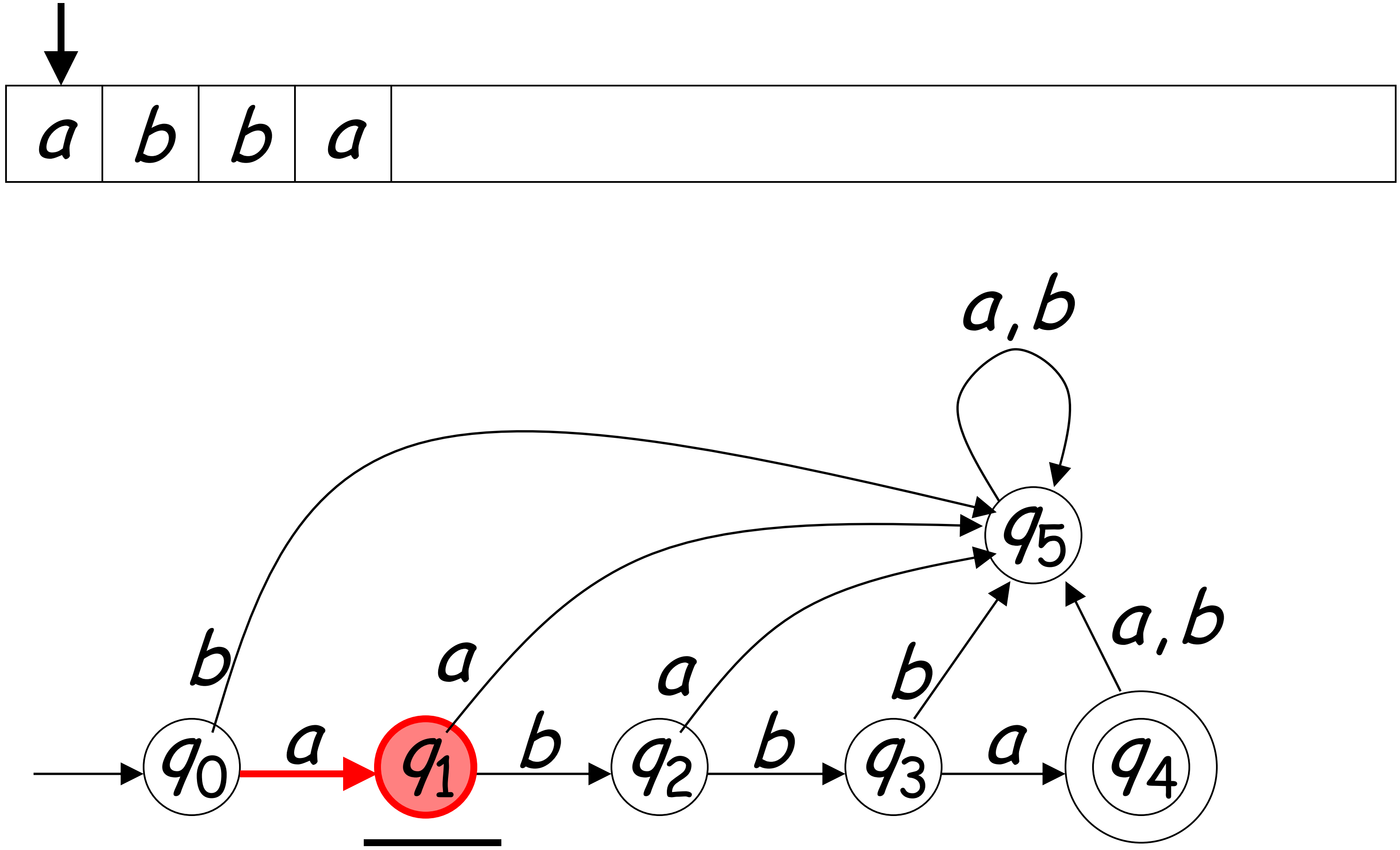
Initial Configuration

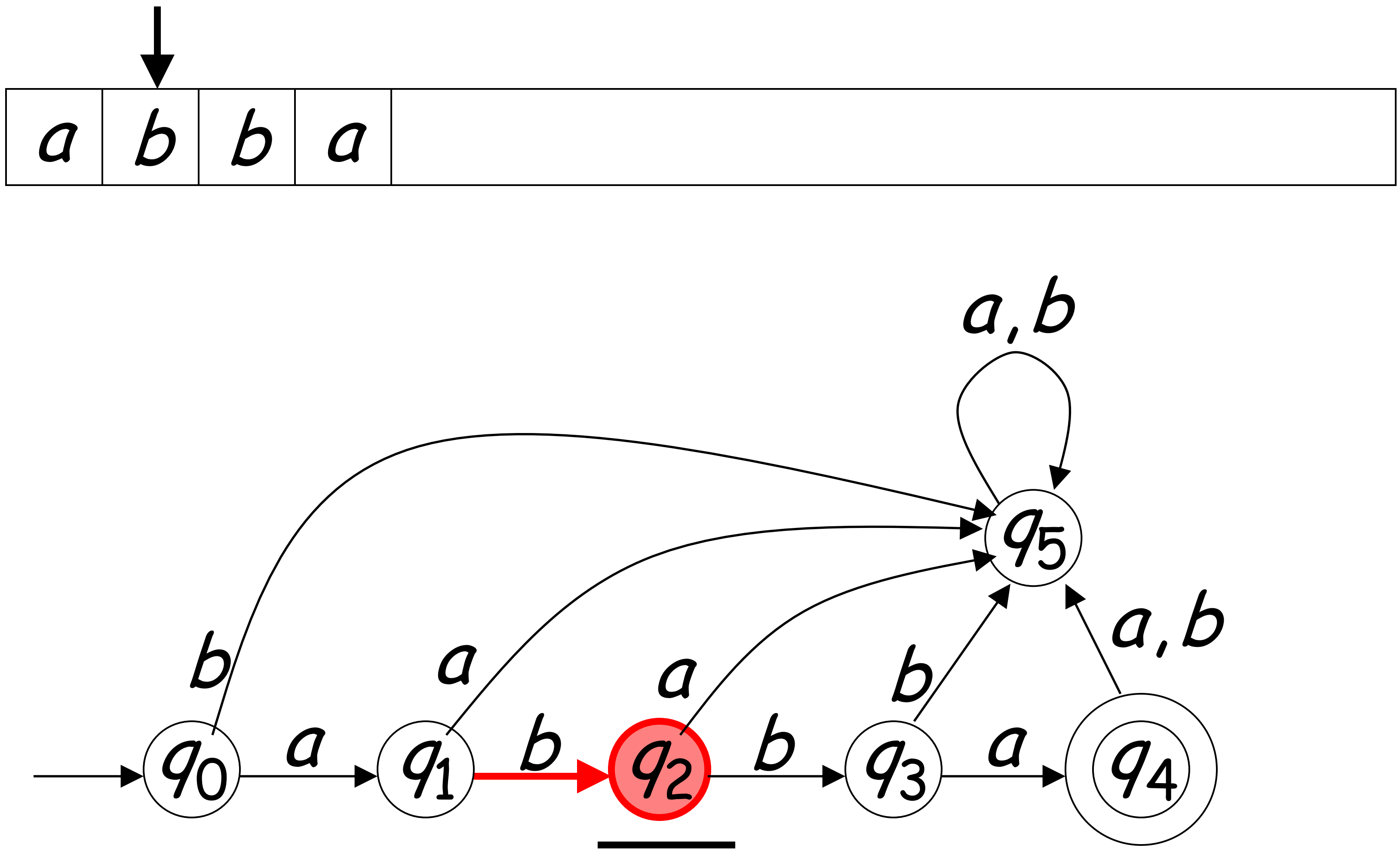
Input String

a	b	b	a	

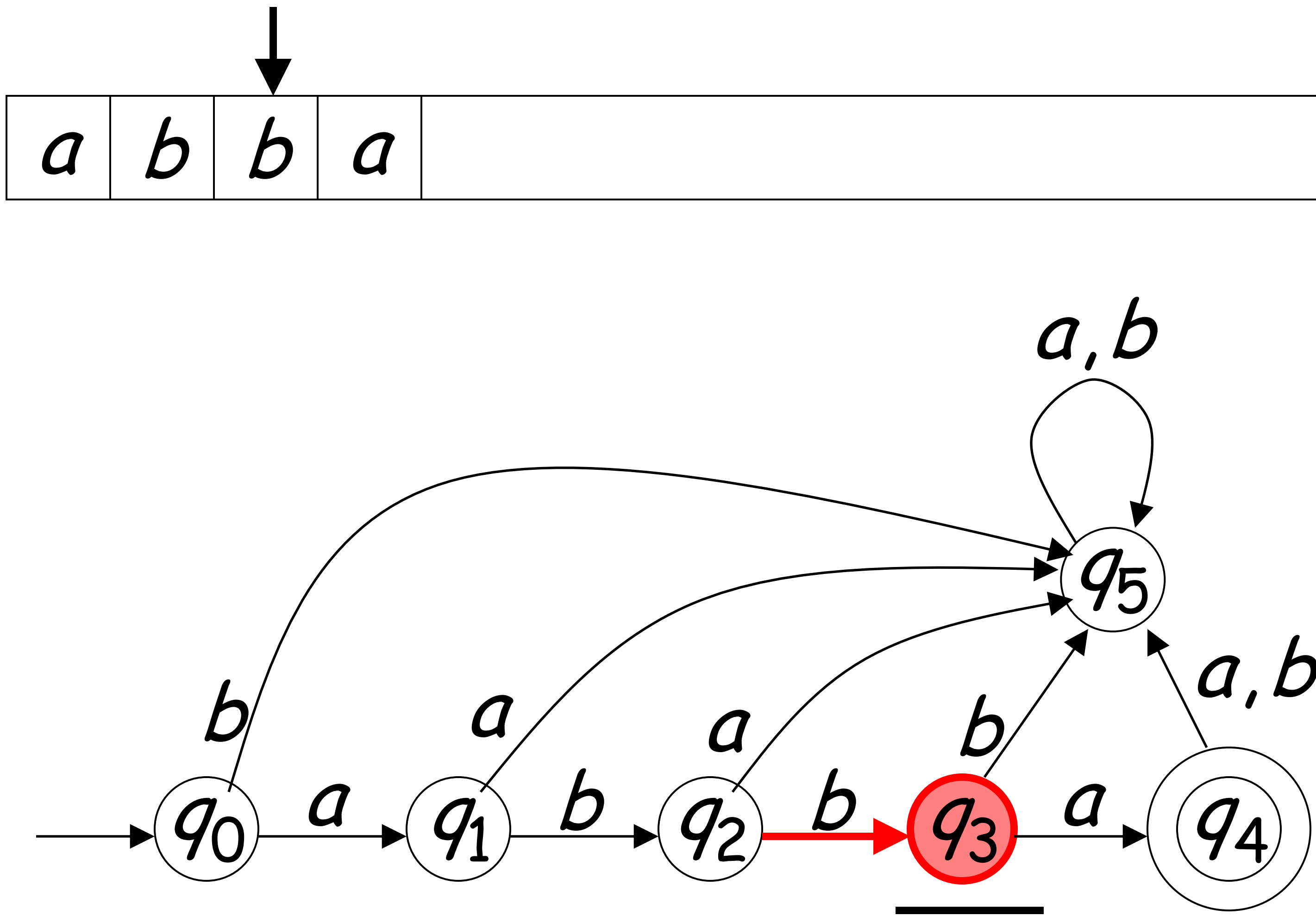


Reading the Input

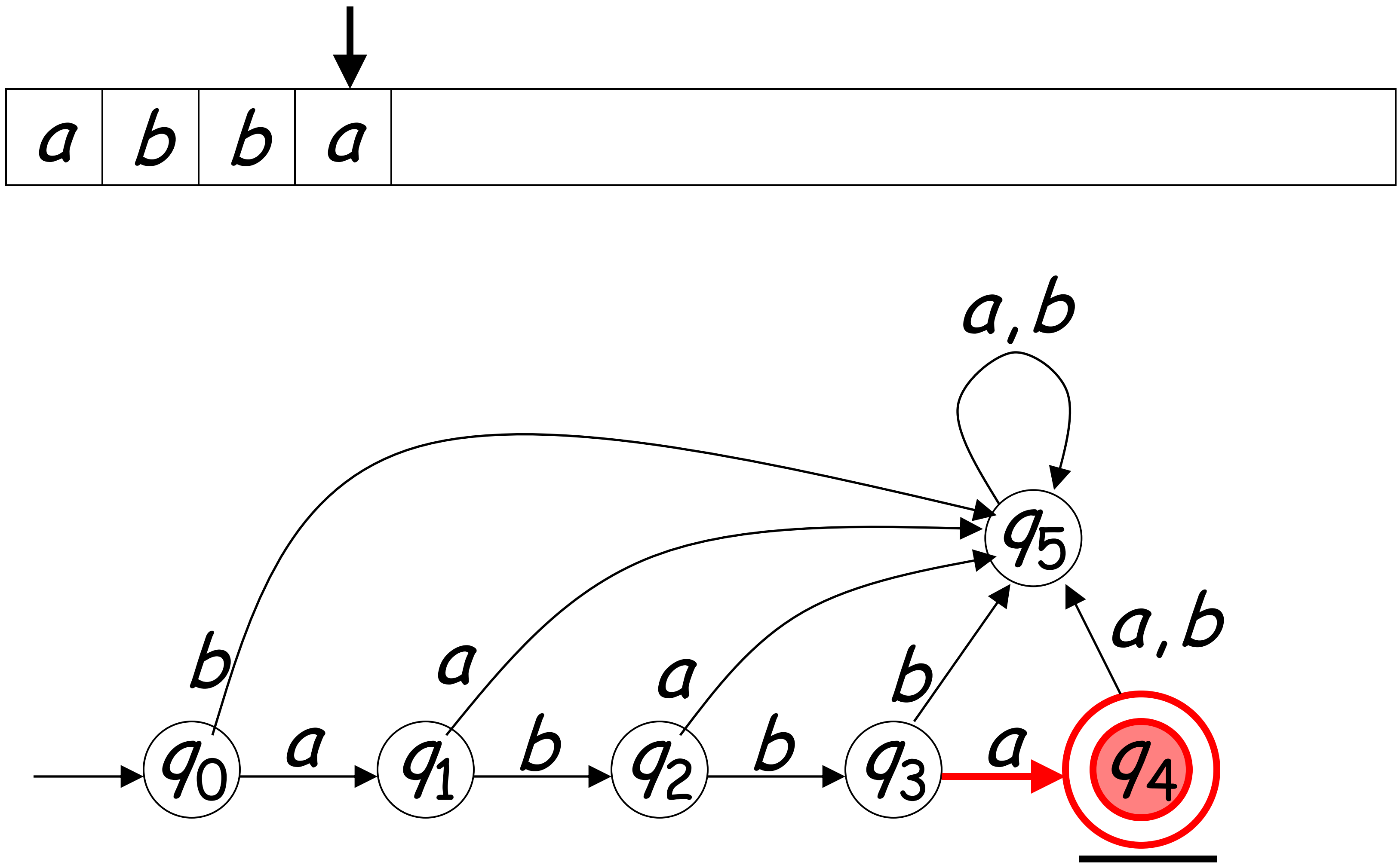




a
b
b
a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

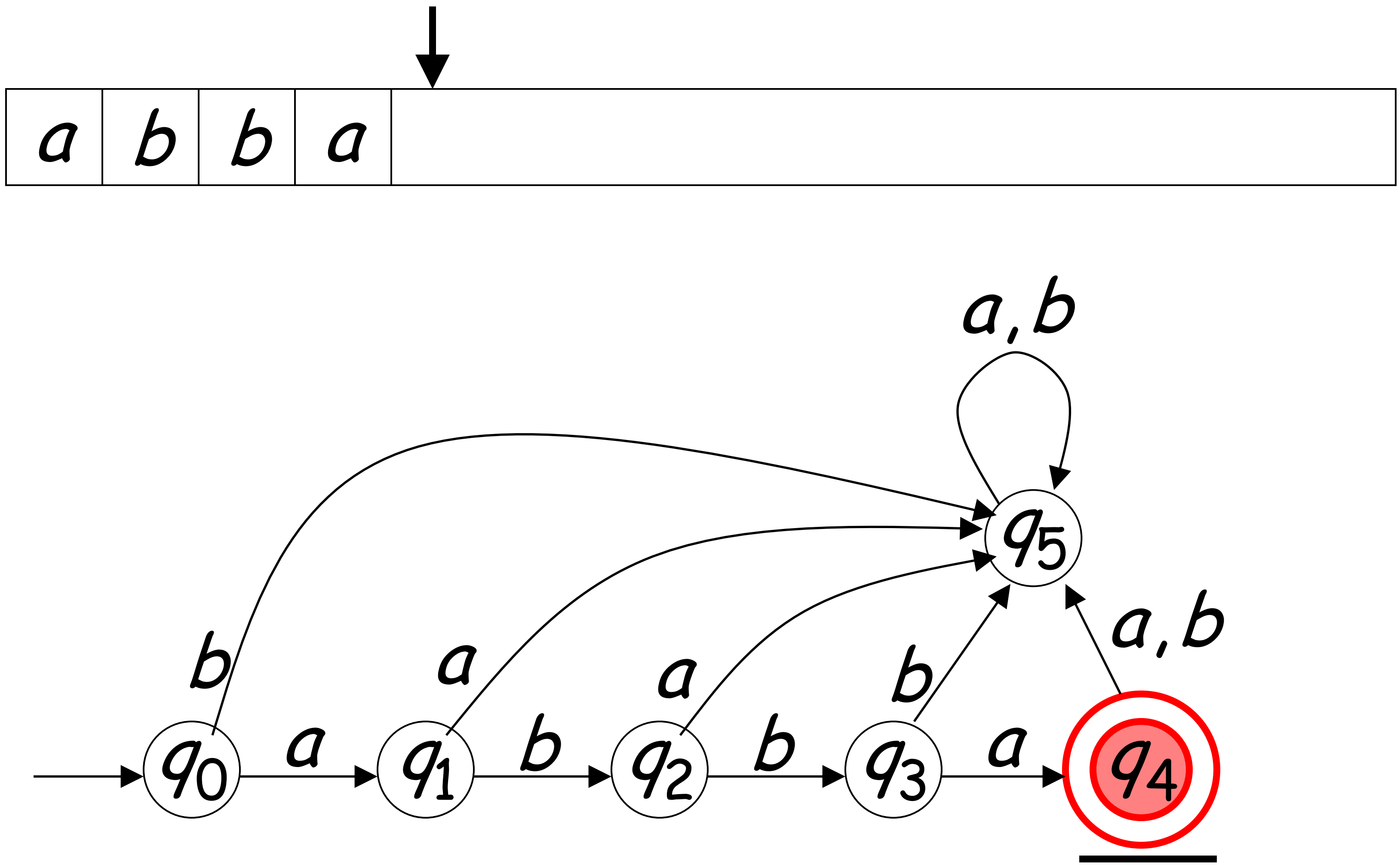


a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4



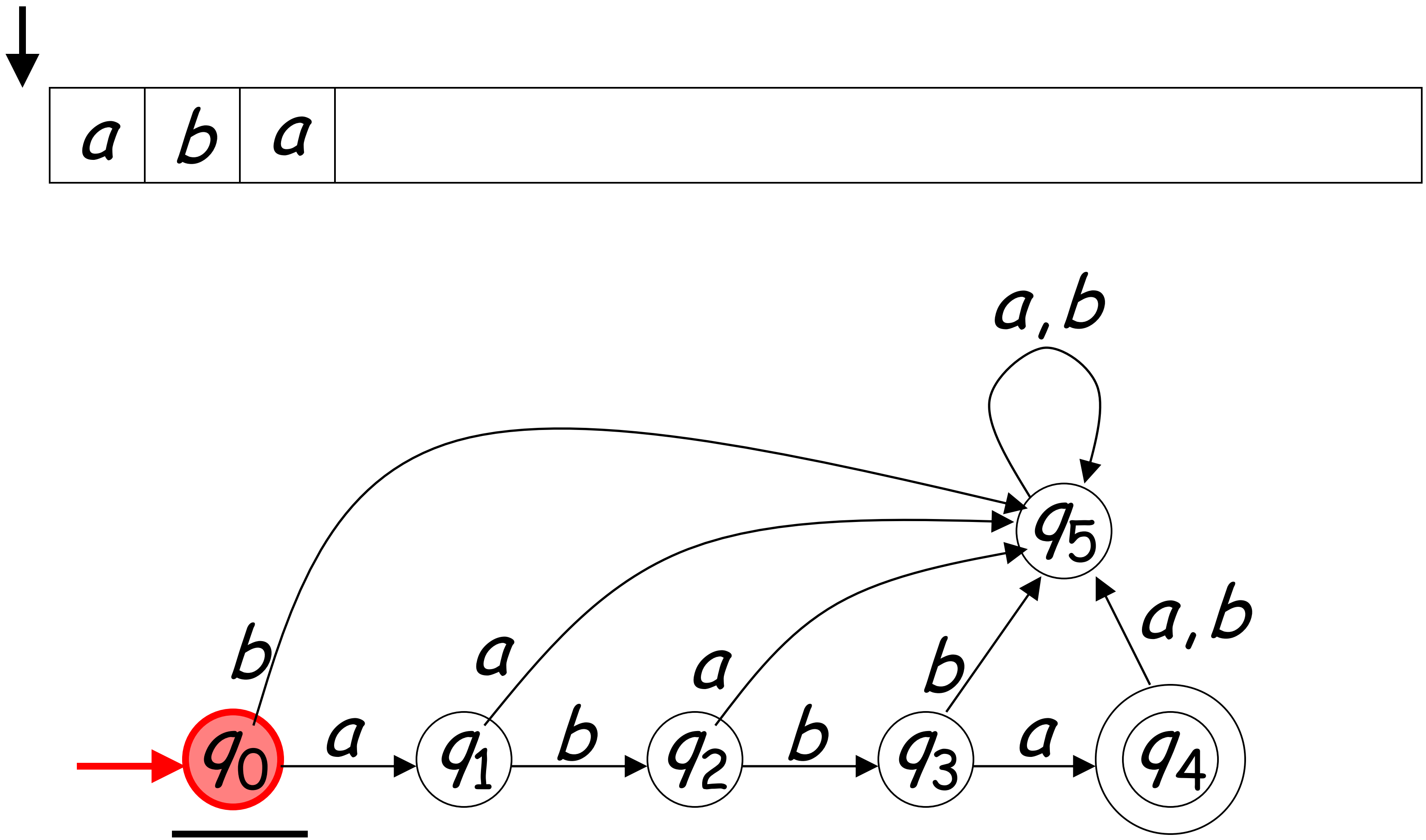
a
b
b
a
a, b
b
a
a
b
a
a, b
q_0
q_1
q_2
q_3
q_4
q_5

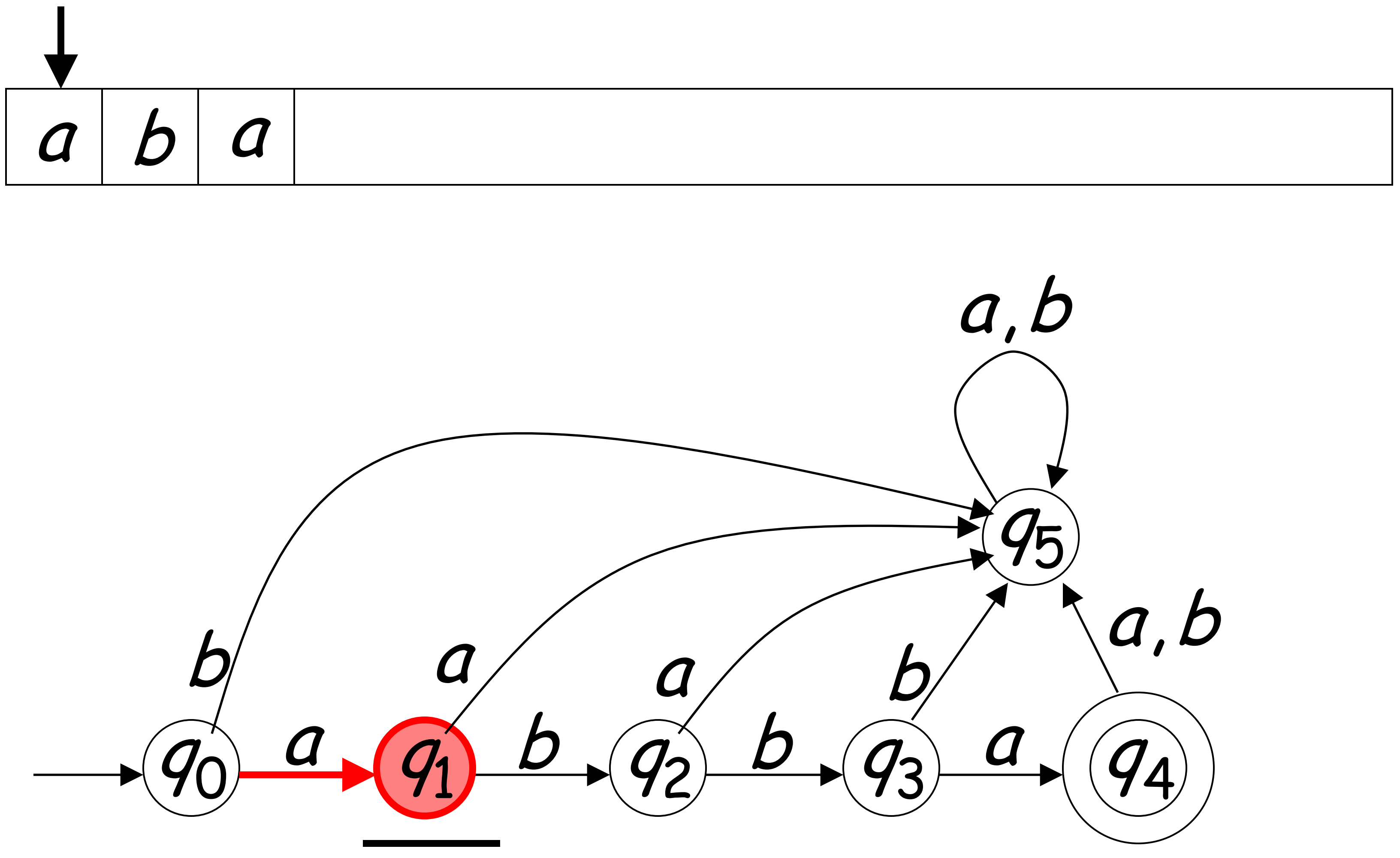
Input finished

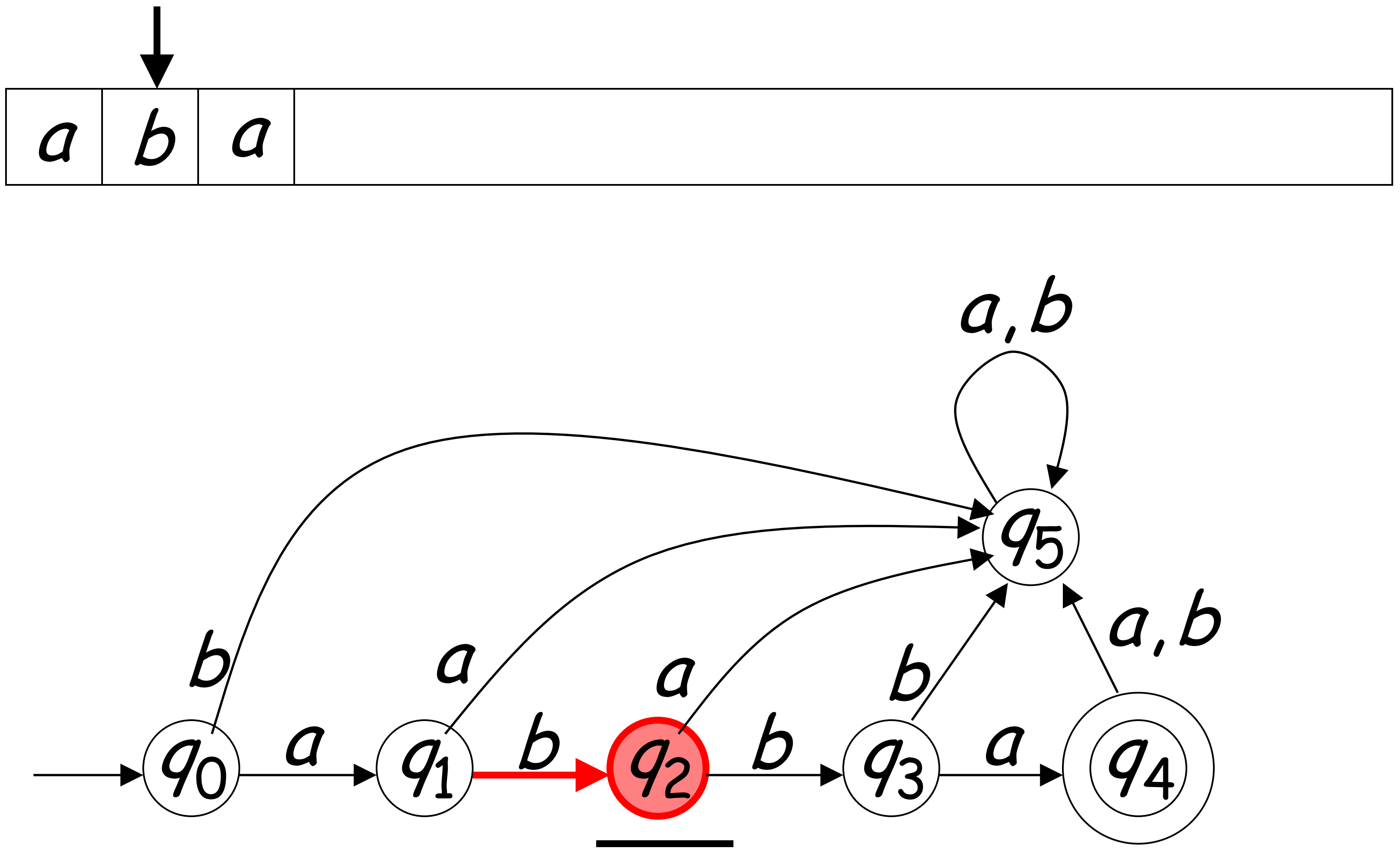


Output: "accept"

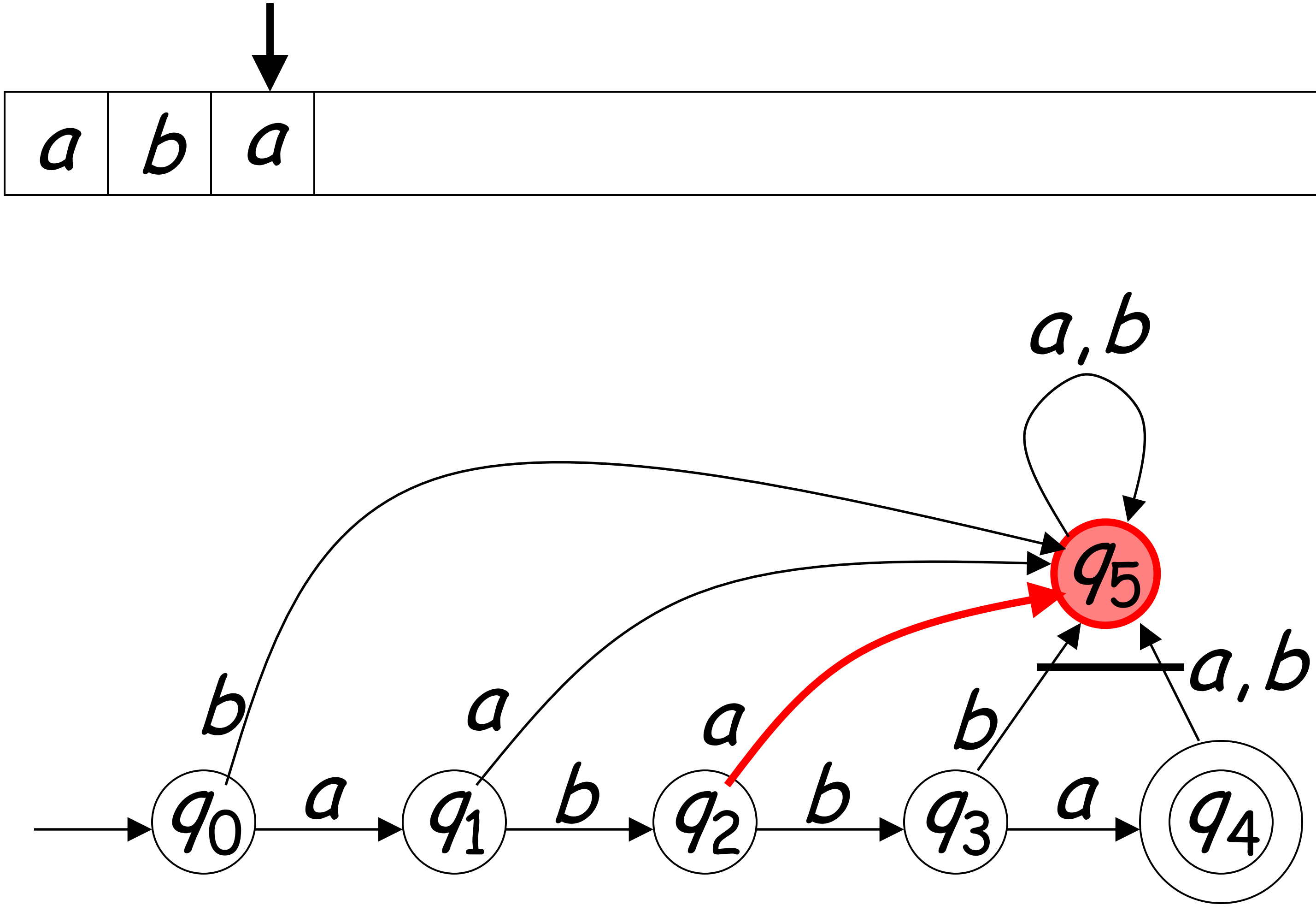
Rejection



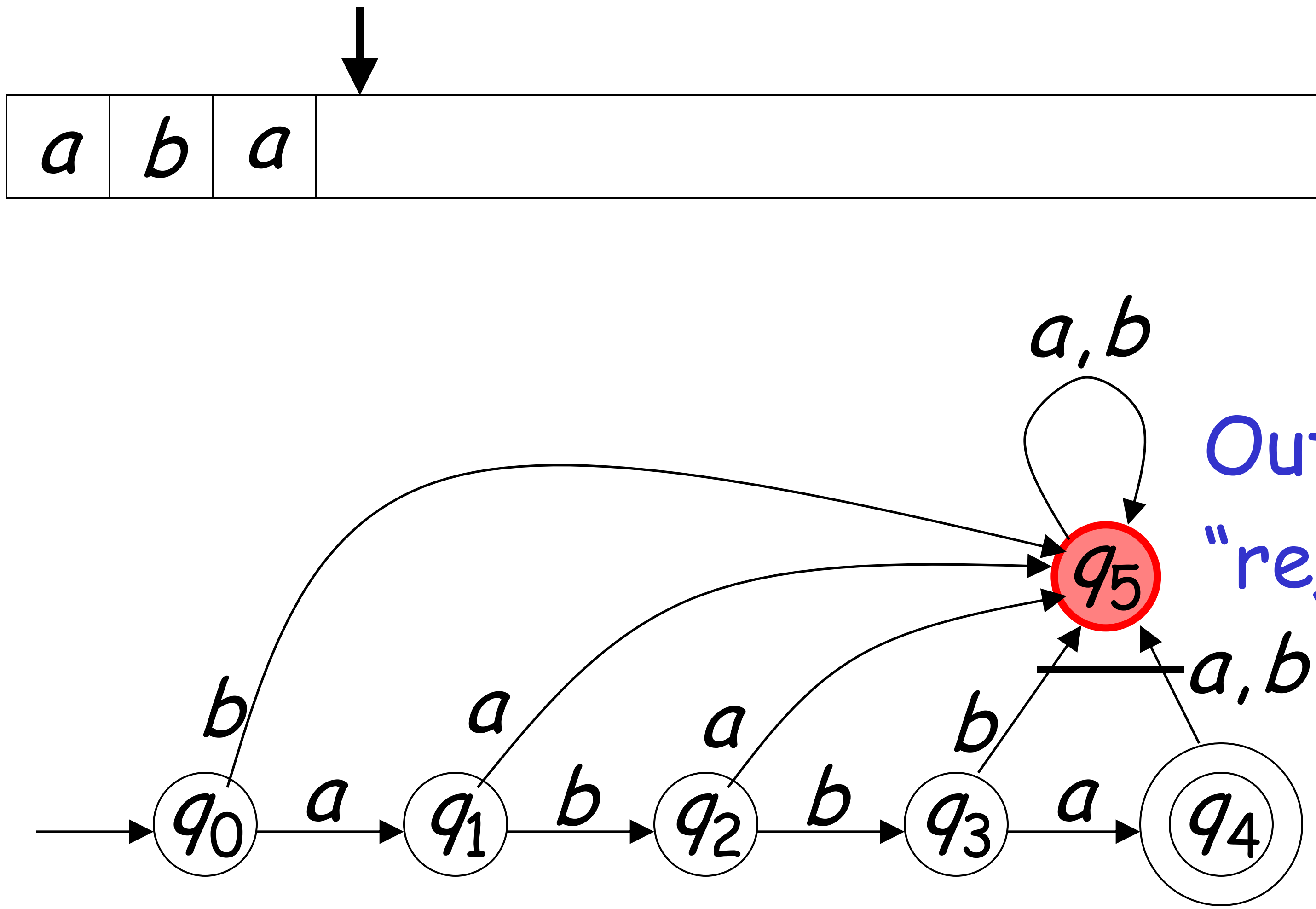

a
b
a
a,b
b
a
a
b
a
b
b
a
a,b
q_0
q_1
q_2
q_3
q_4
q_5



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

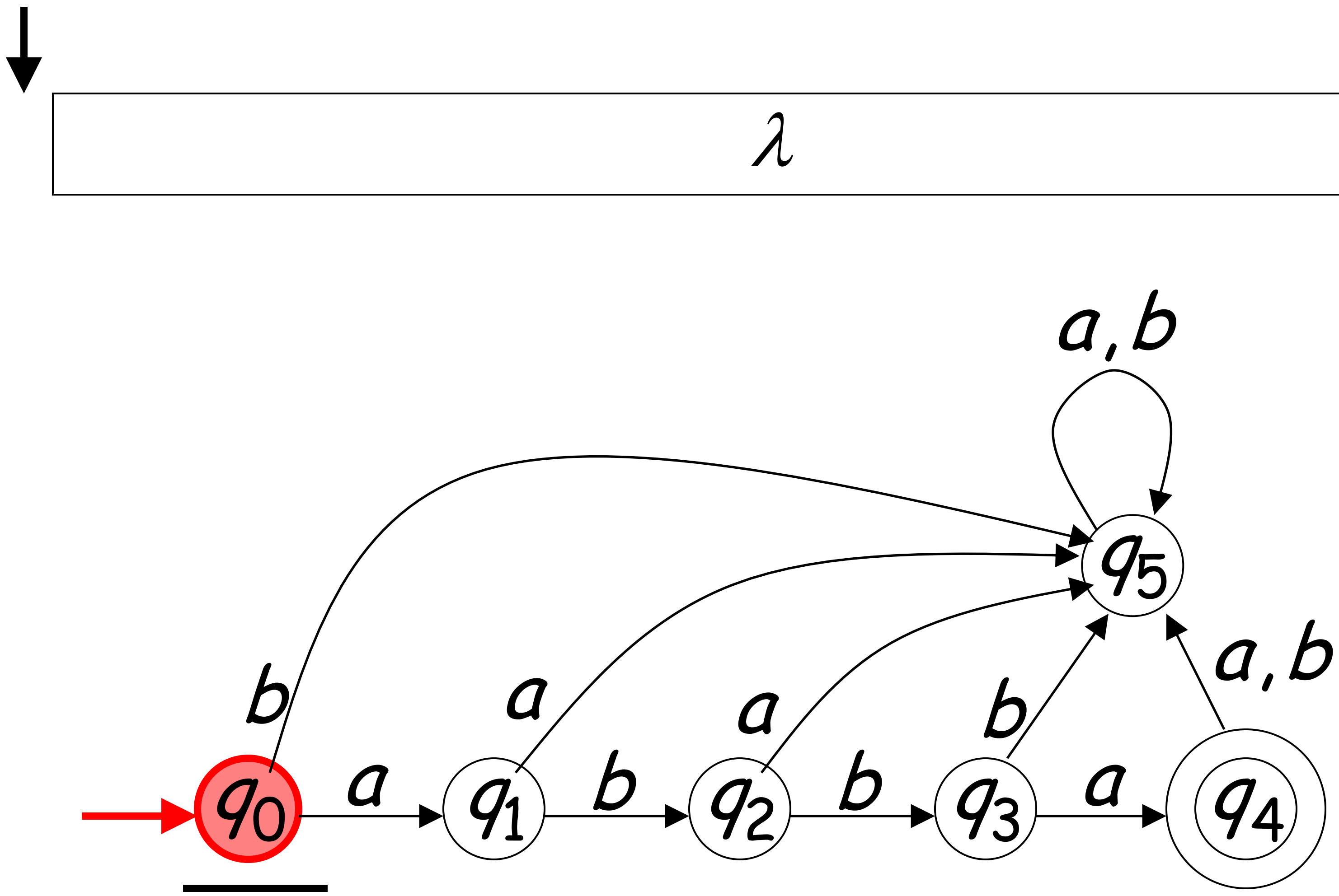

a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

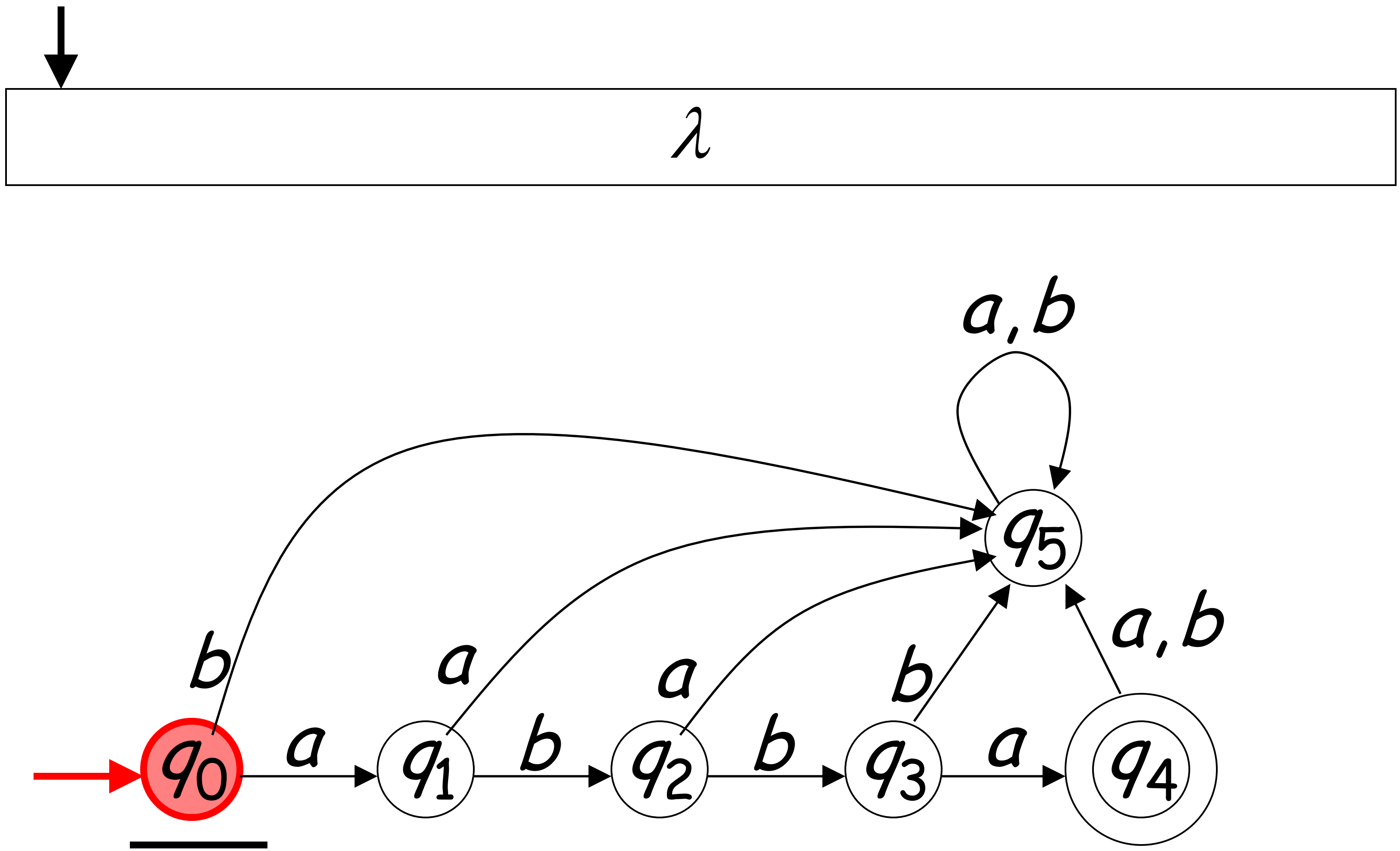
Input finished



Output: "reject"

Another Rejection





Output:
"reject"

Formalities

Deterministic Finite Accepter (DFA)

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

Q : set of states

Σ : input alphabet

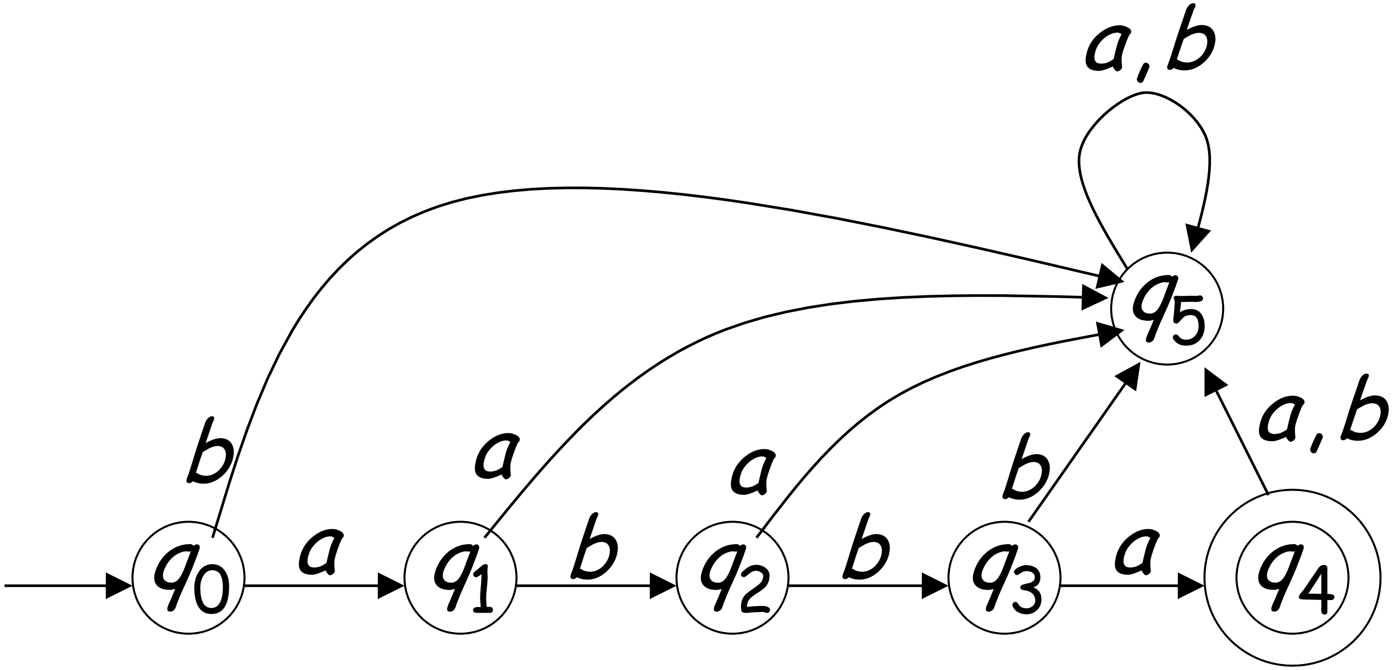
δ : transition function

q_0 : initial state

F : set of final states

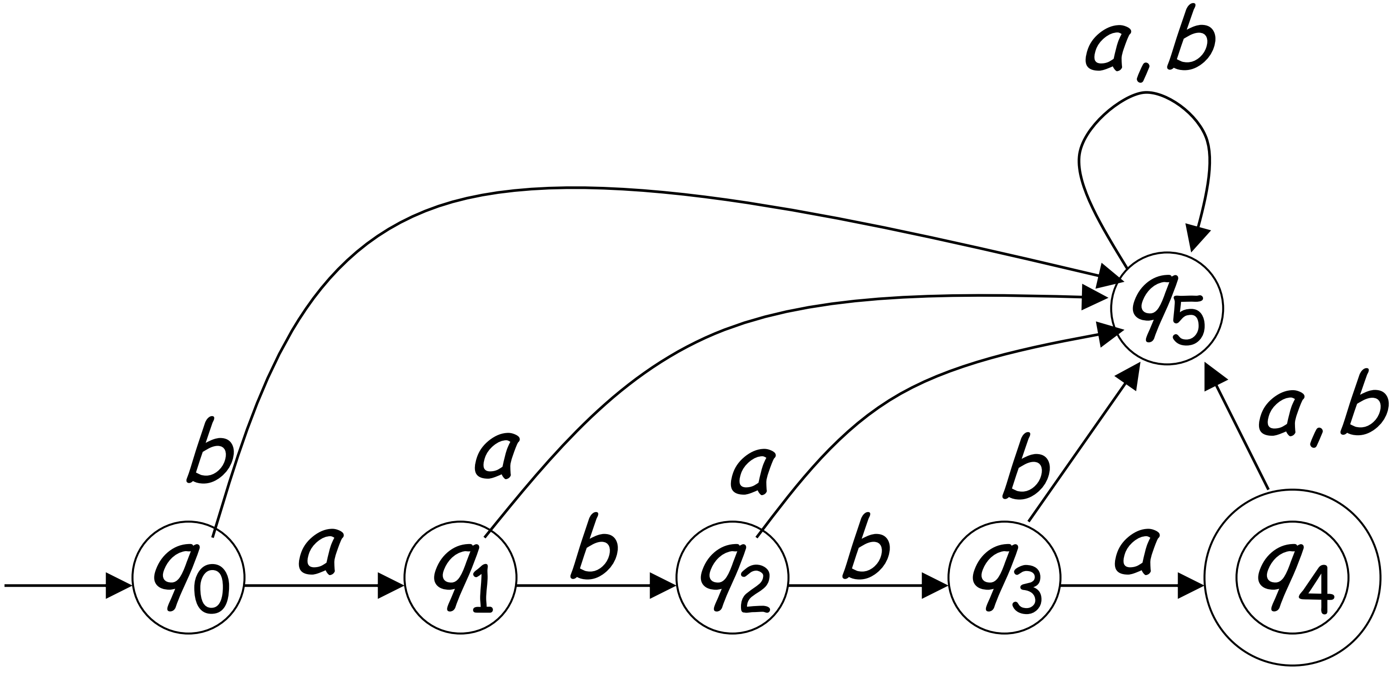
Input Alphabet Σ

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

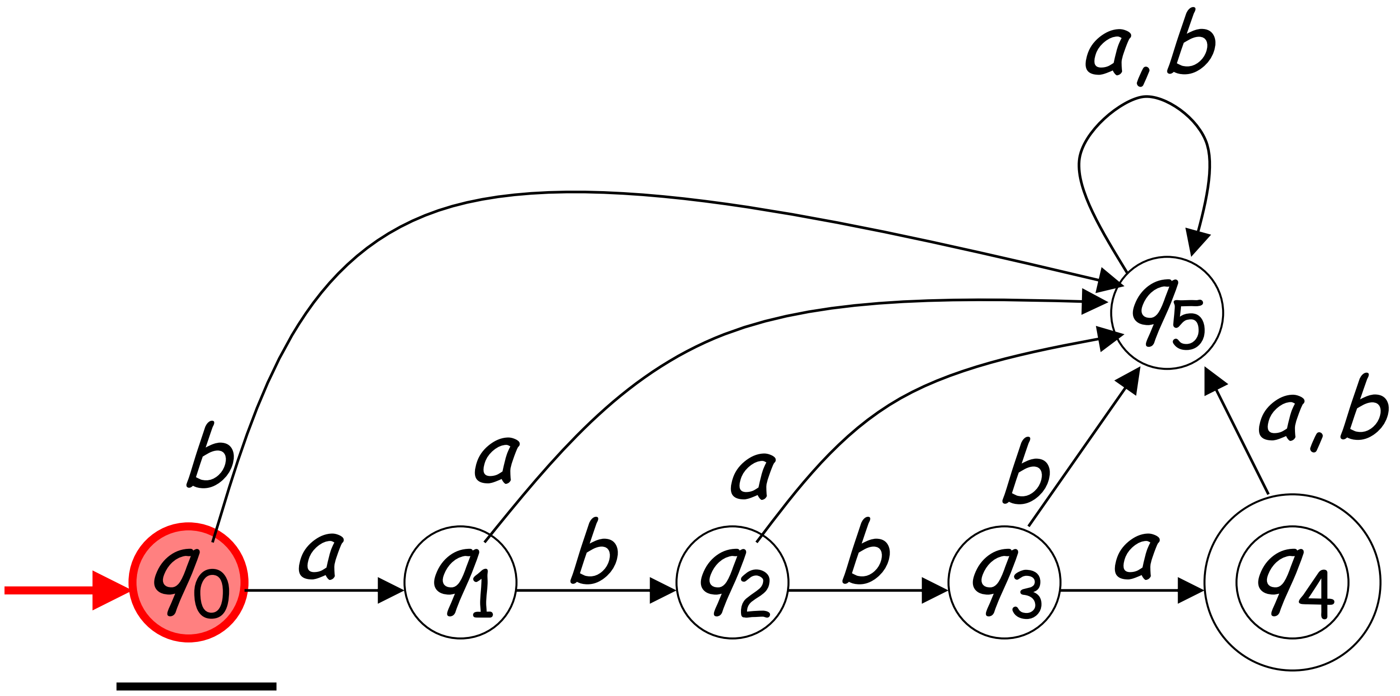


Set of States Q

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

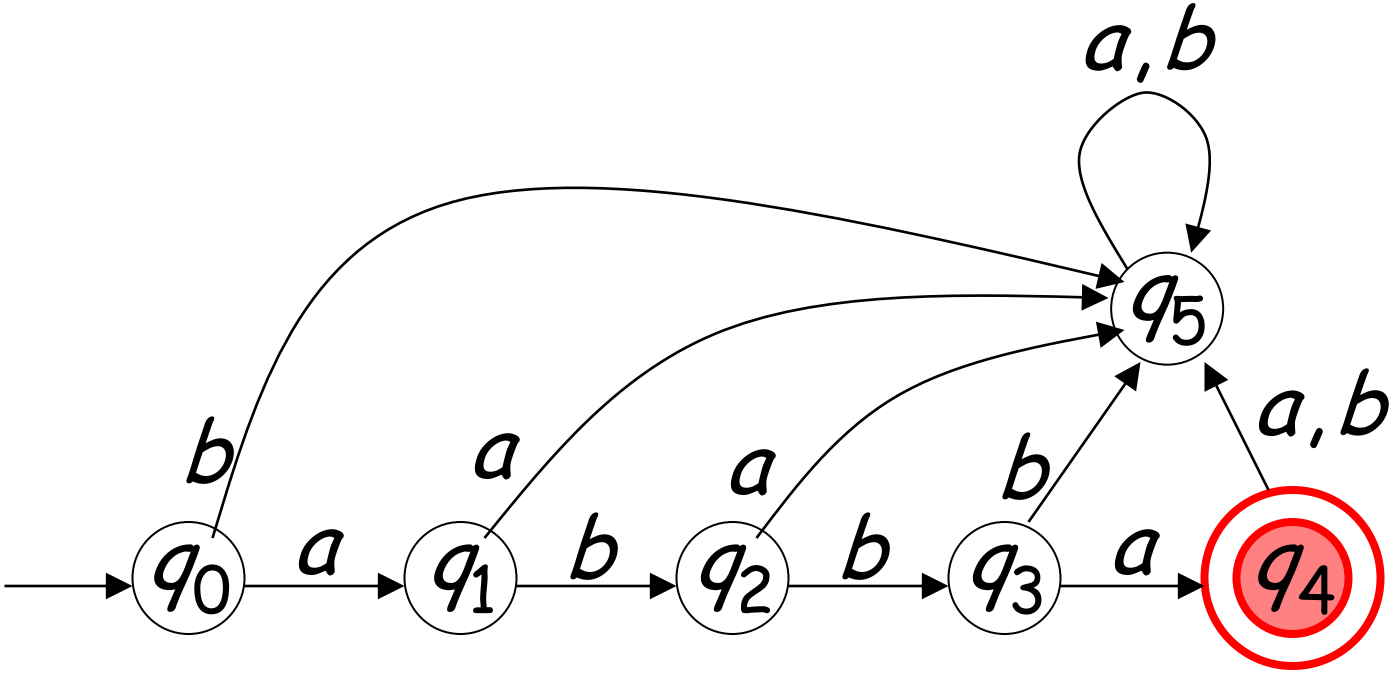


Initial State q_0



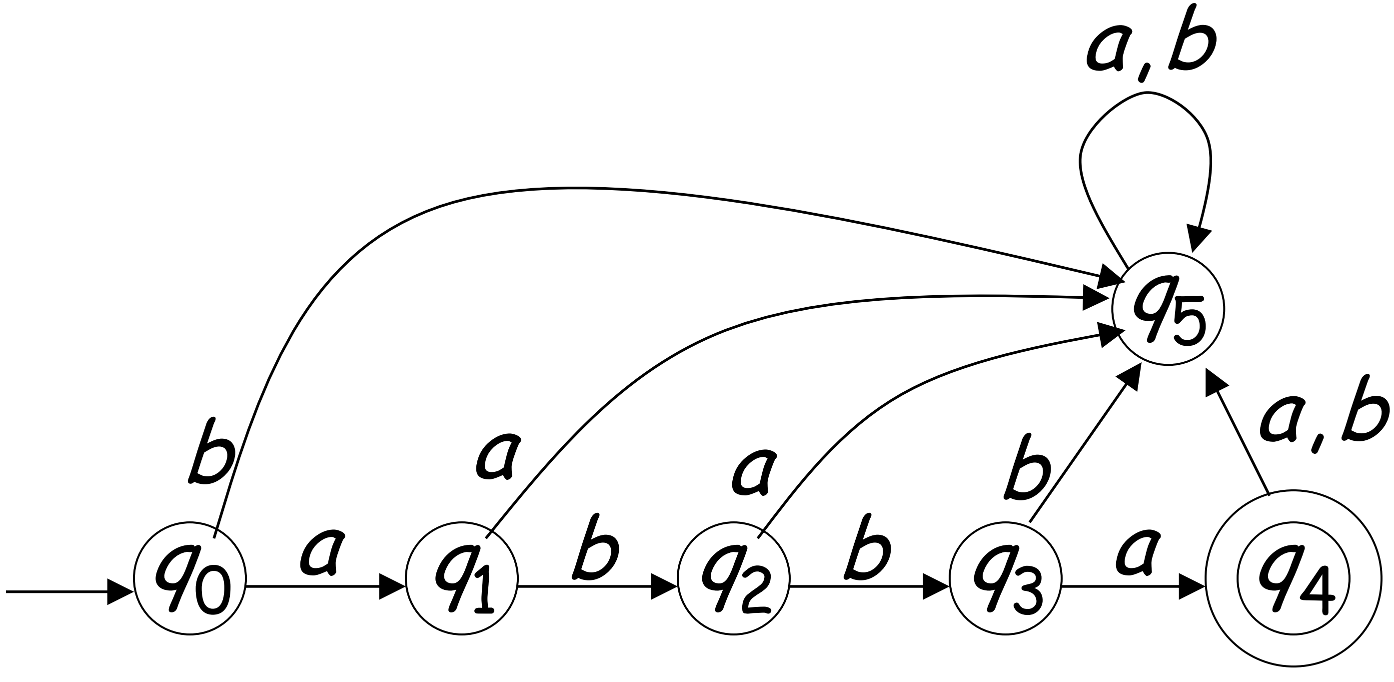
Set of Final States F

$$F = \{q_4\}$$

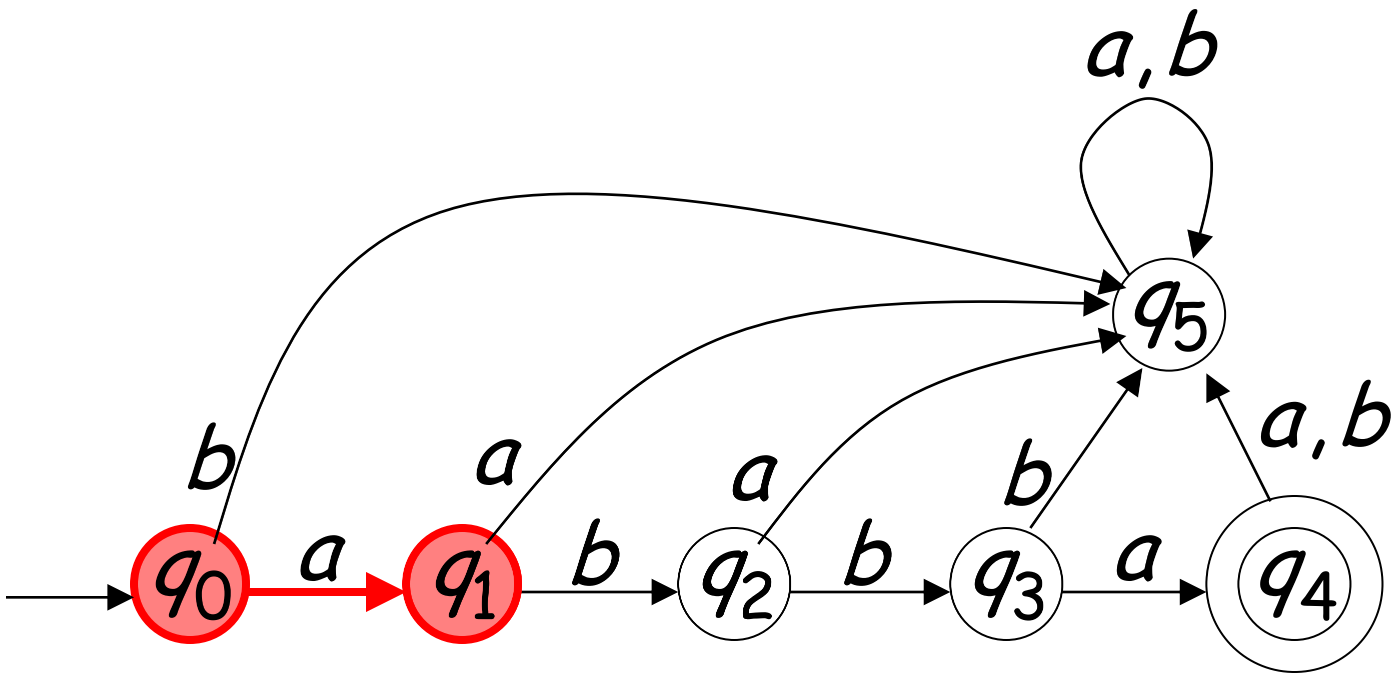


Transition Function δ

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

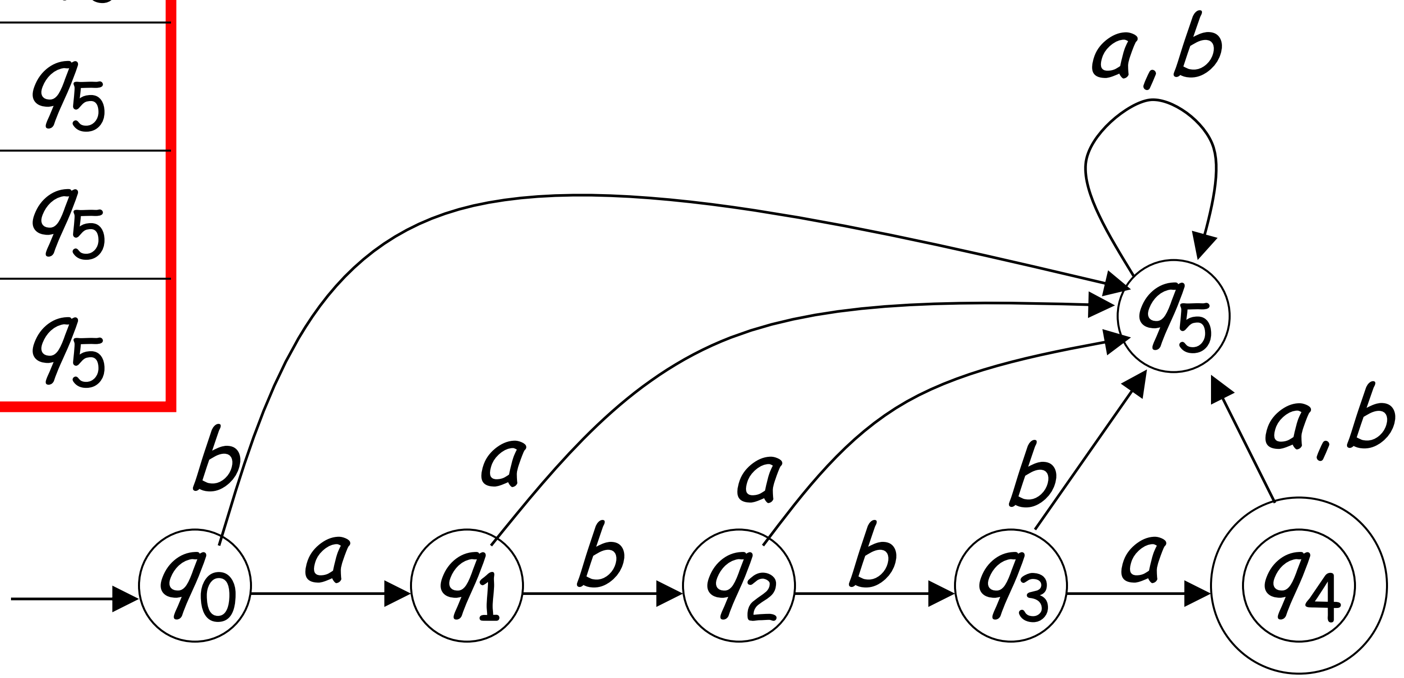


$$\delta(q_0, a) = q_1$$



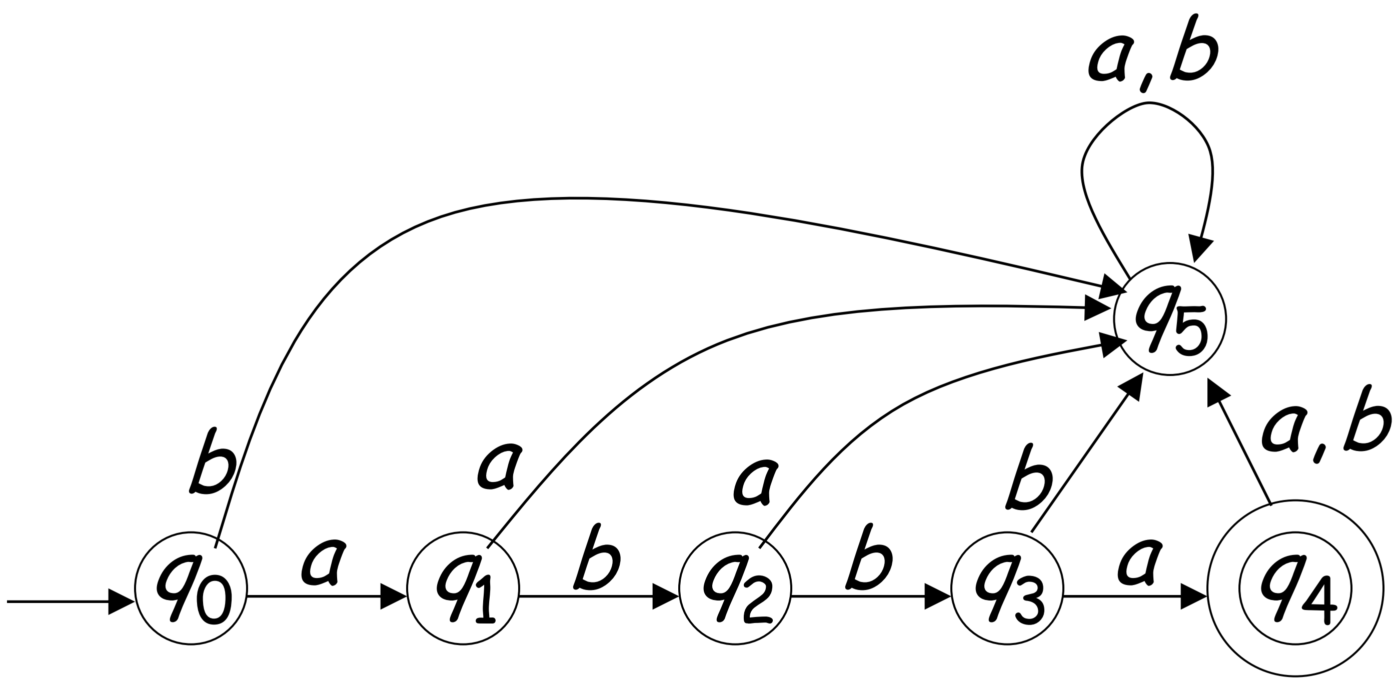
Transition Function δ

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

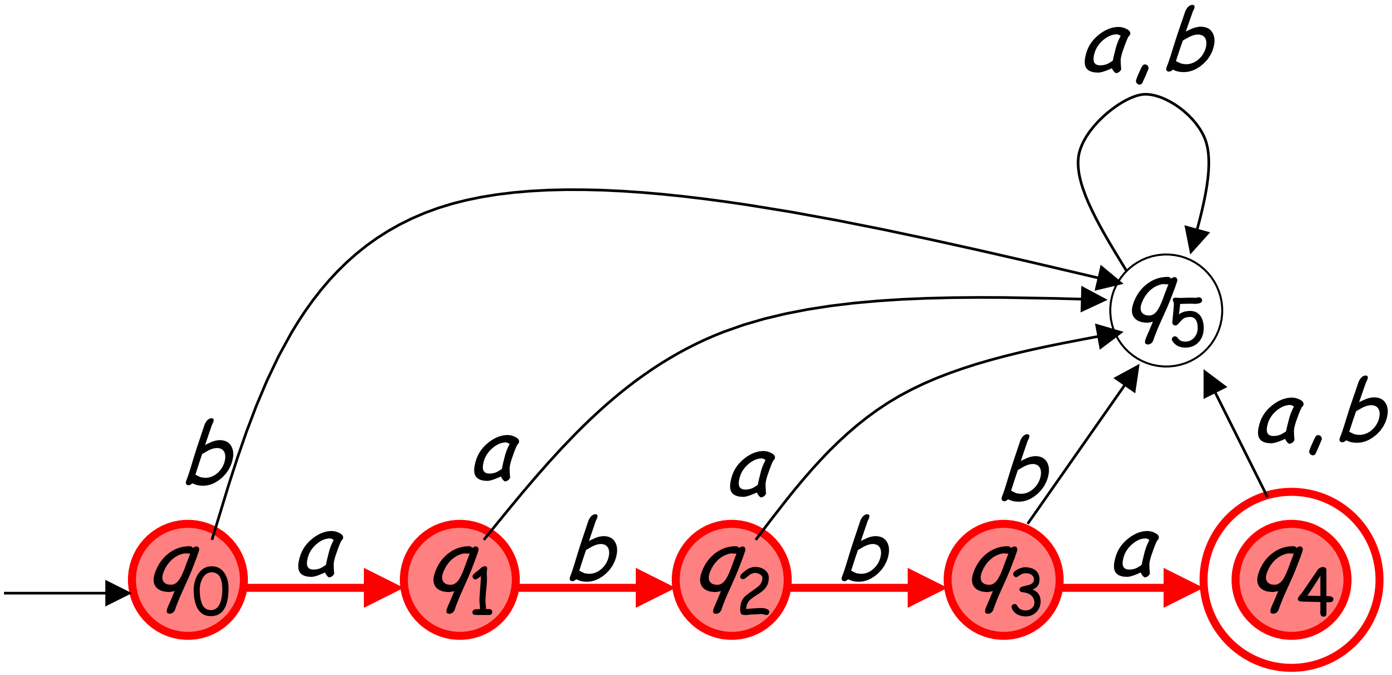


Extended Transition Function $\delta *$

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



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



Languages Accepted by DFAs

Take DFA M

Definition:

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

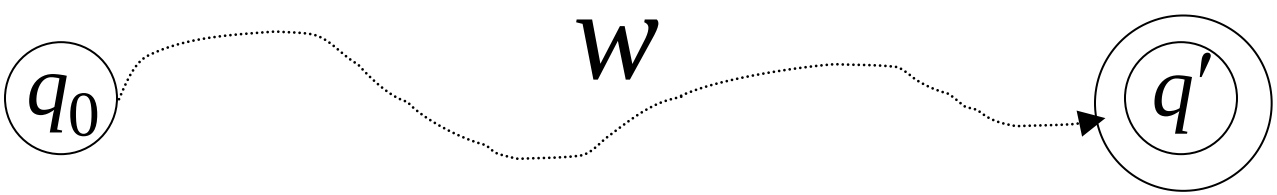
$L(M)$ = { strings that drive M to a final state}

Formally

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

Language accepted by M :

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

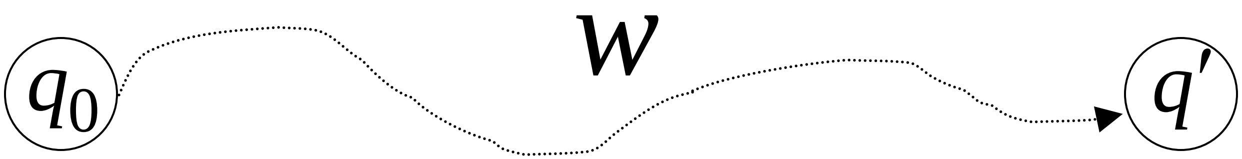


$q' \in F$

Observation

Language rejected by M :

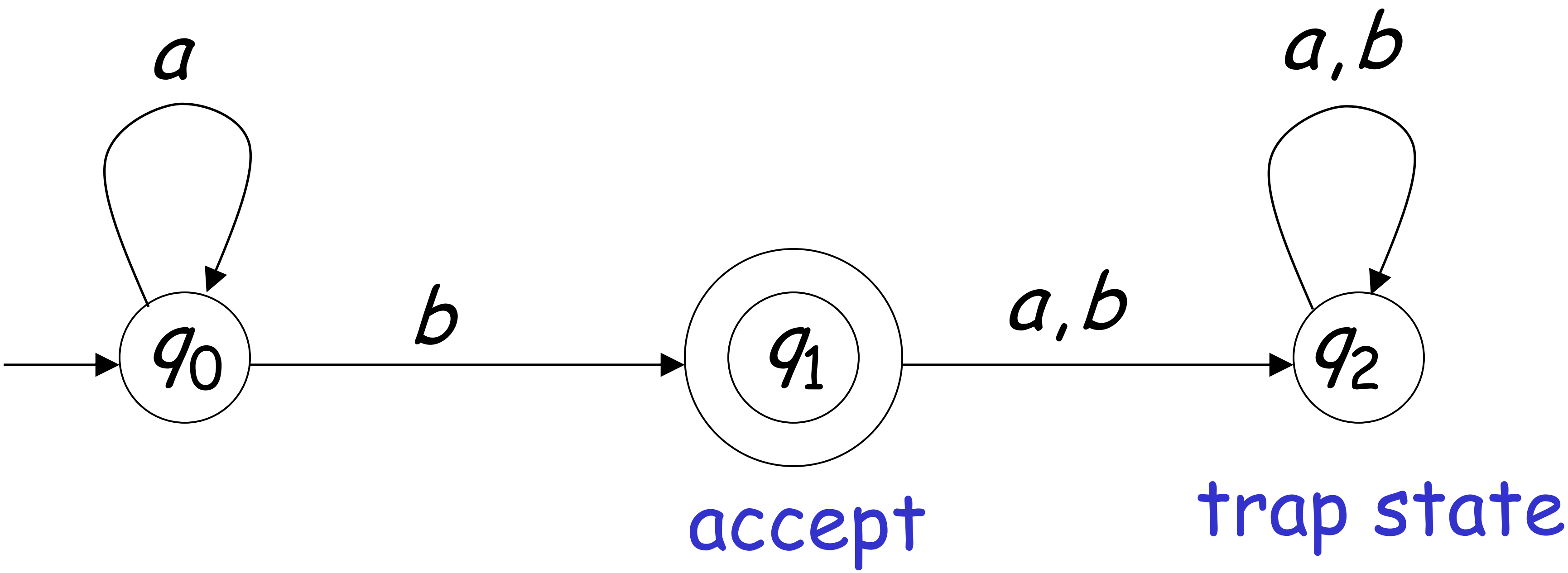
$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$



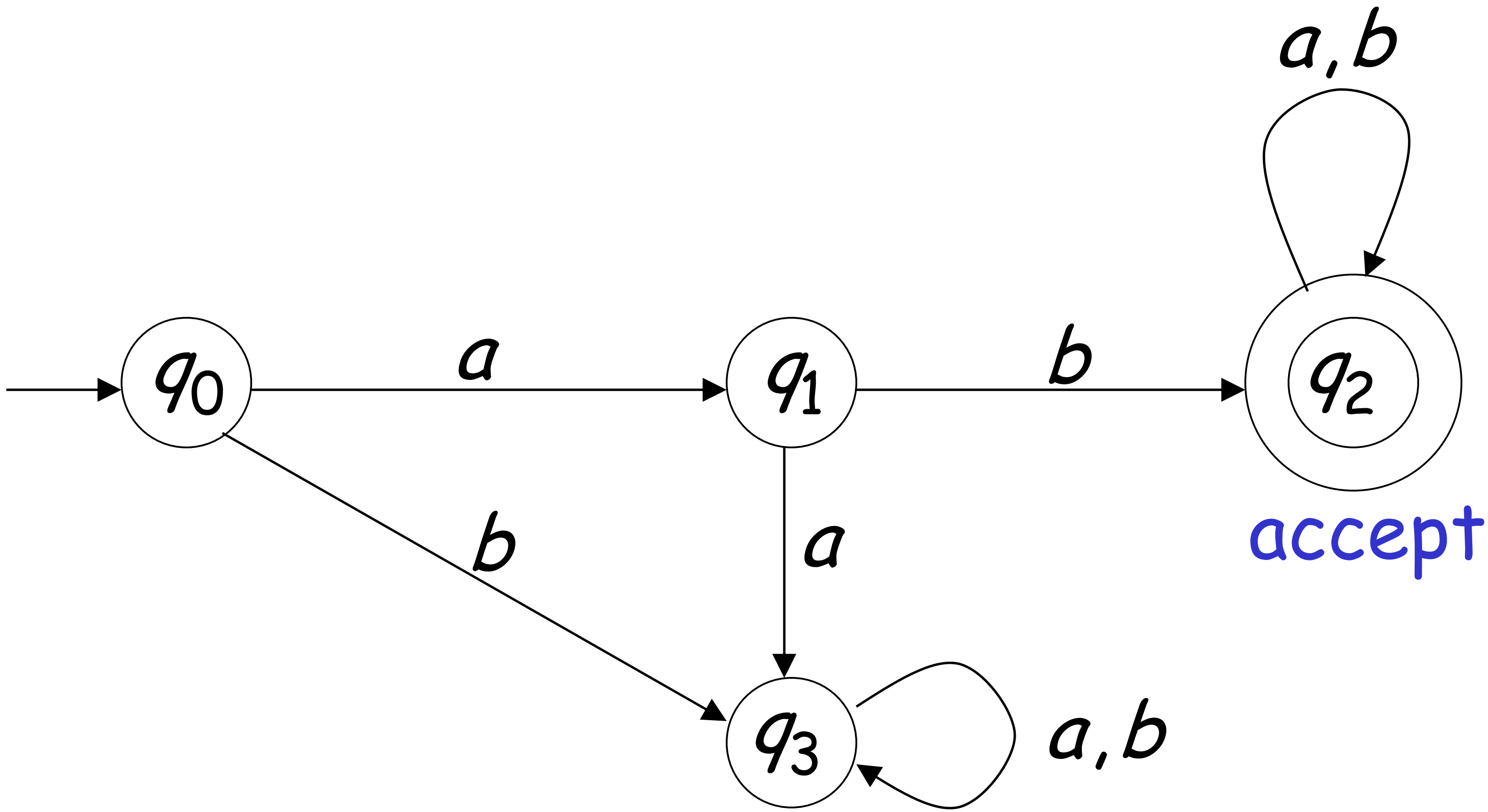
$q' \notin F$

More Examples

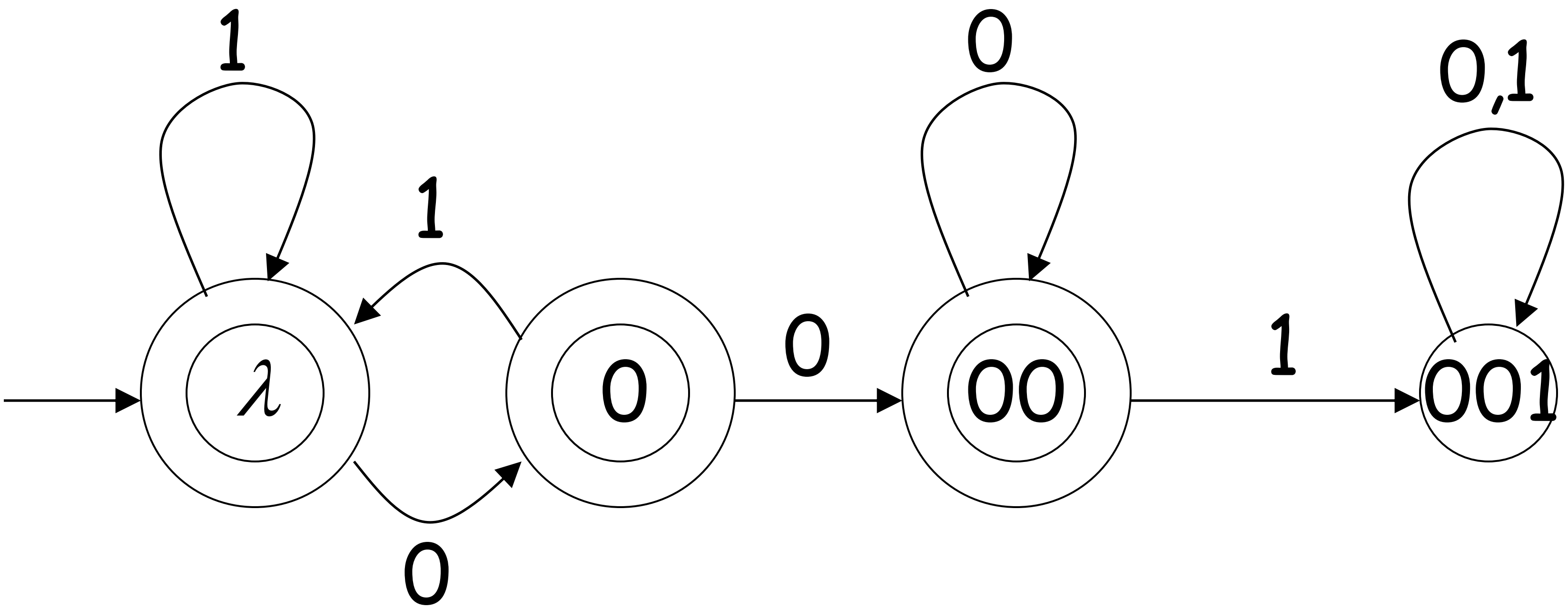
$$L(M) = \{a^n b : n \geq 0\}$$



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



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



Regular Languages

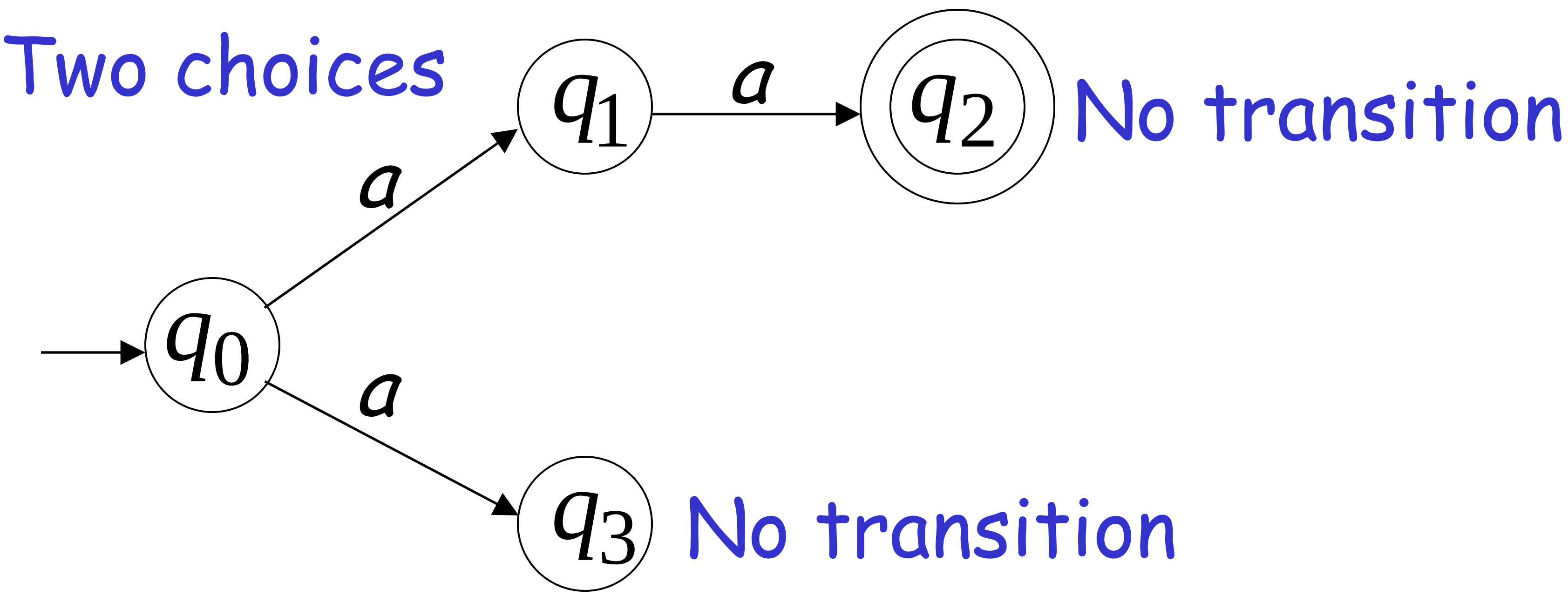
A language L is regular if there is a DFA M such that $L = L(M)$

All regular languages form a language family

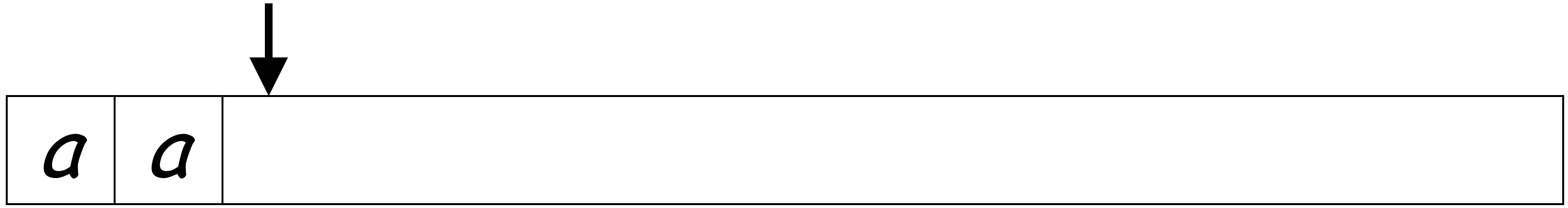
Nondeterministic Automata

Nondeterministic Finite Accepter (NFA)

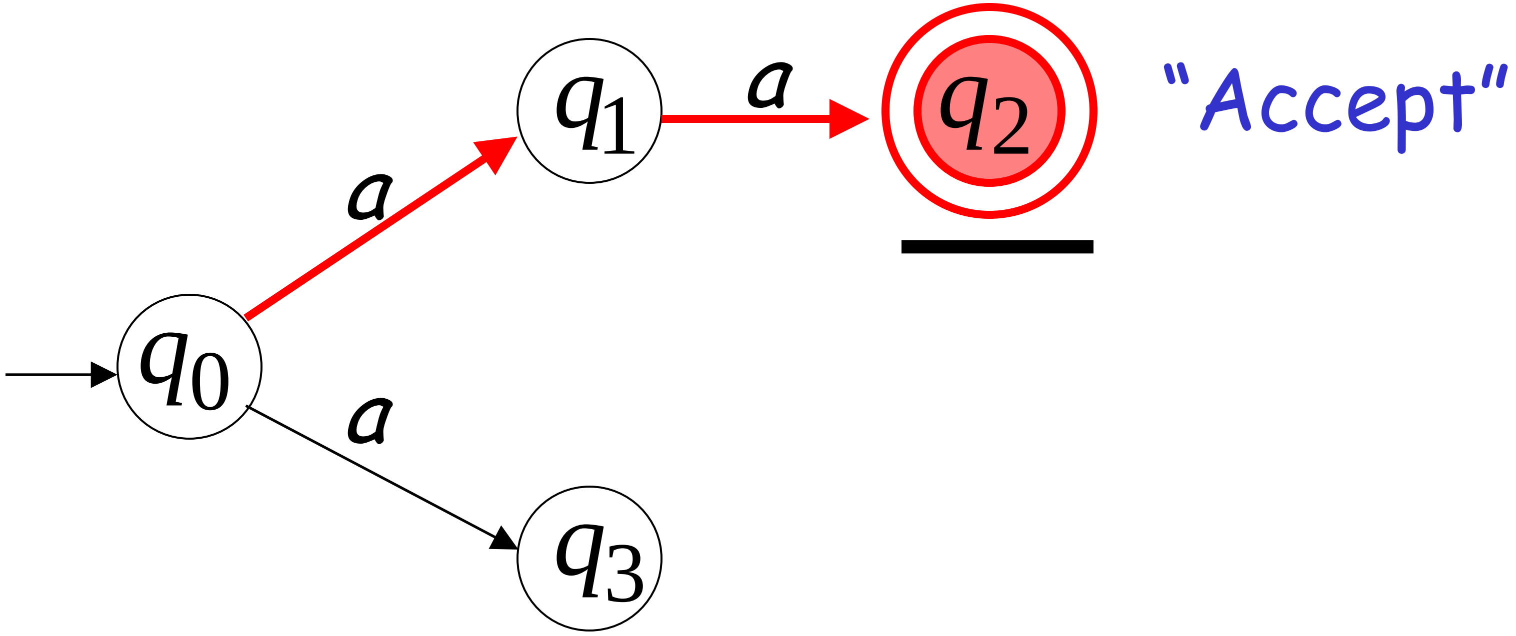
Alphabet = $\{a\}$



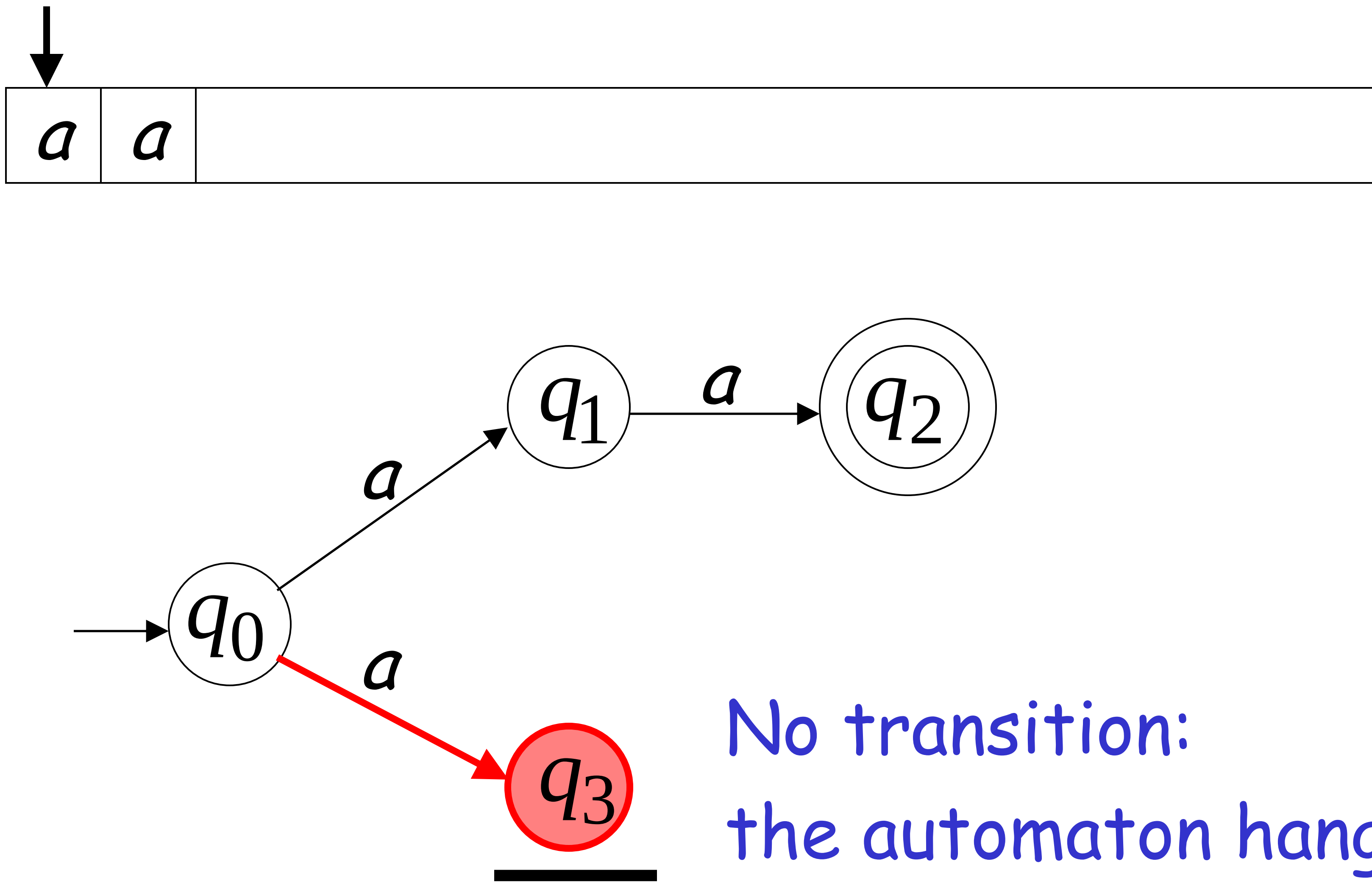
First Choice



All input is consumed



Second Choice



No transition:
the automaton hangs
"Reject"

An NFA accepts a string:

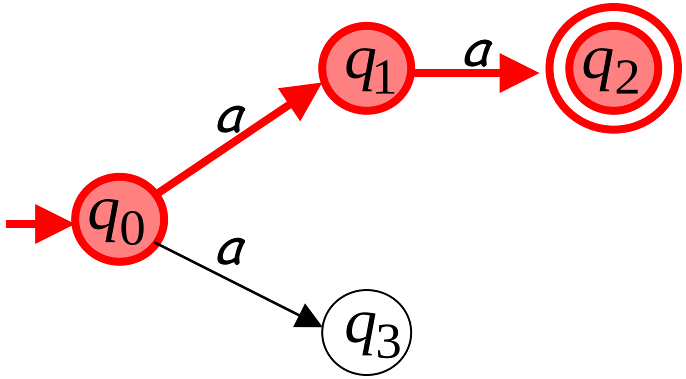
when there is a computation of the NFA that accepts the string

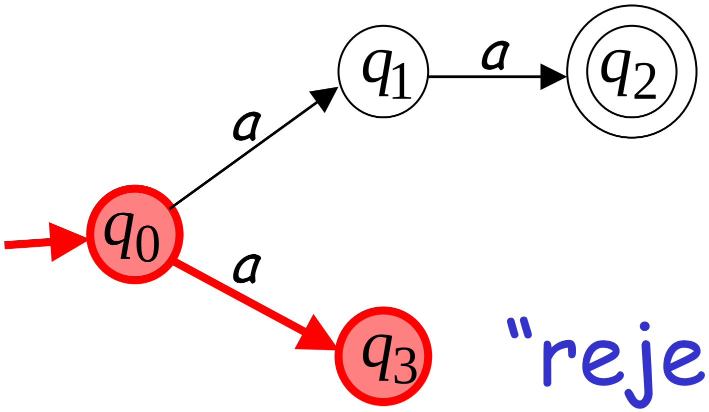
AND

all the input is consumed and the automaton is in a final state

Therefore, aa is accepted by the NFA:

"accept"

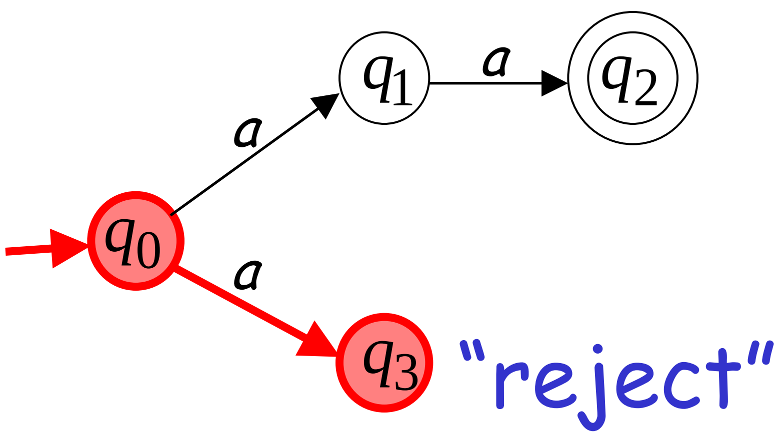


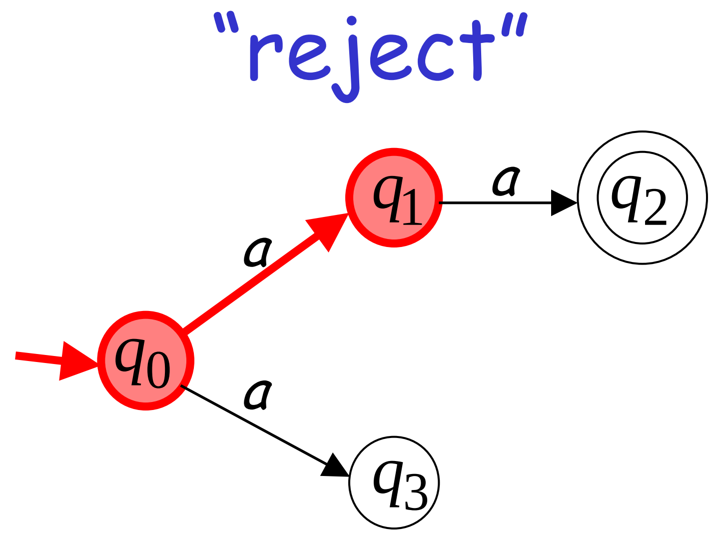


"reject"

because this computation accepts aa

On the contrary, a is rejected by the NFA:

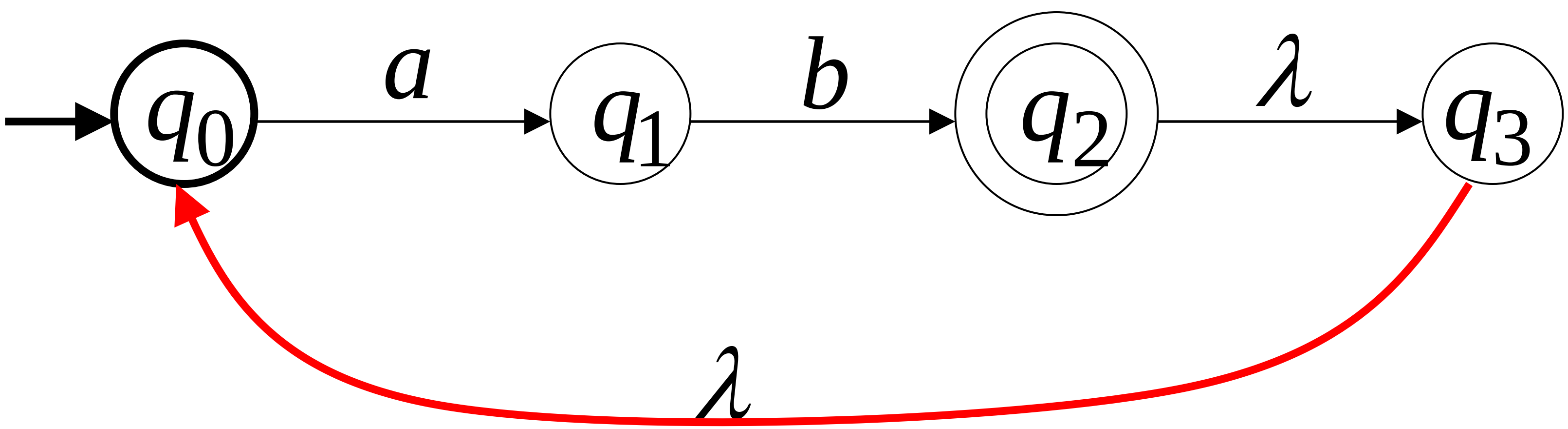




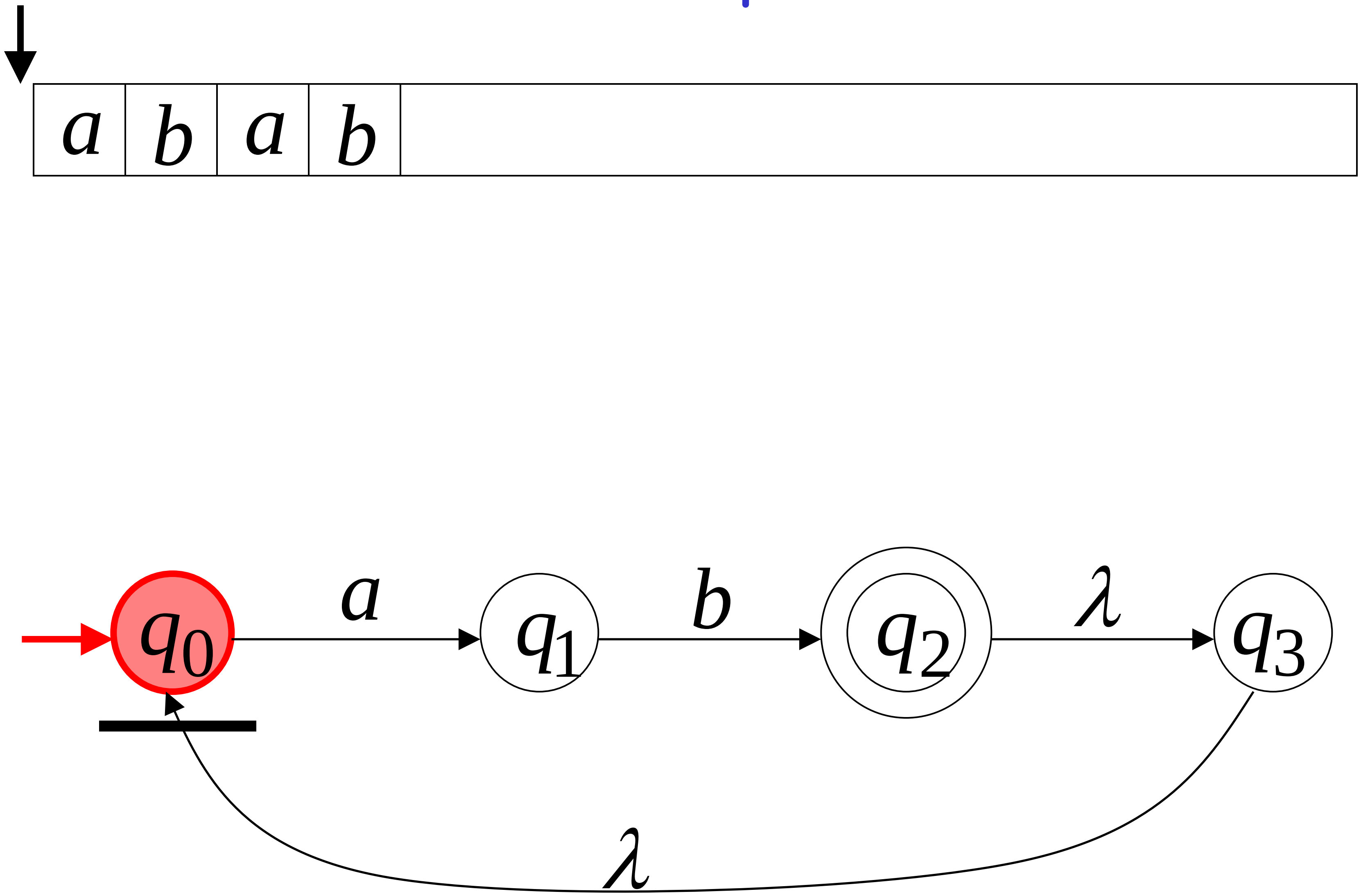
All possible computations lead to rejection

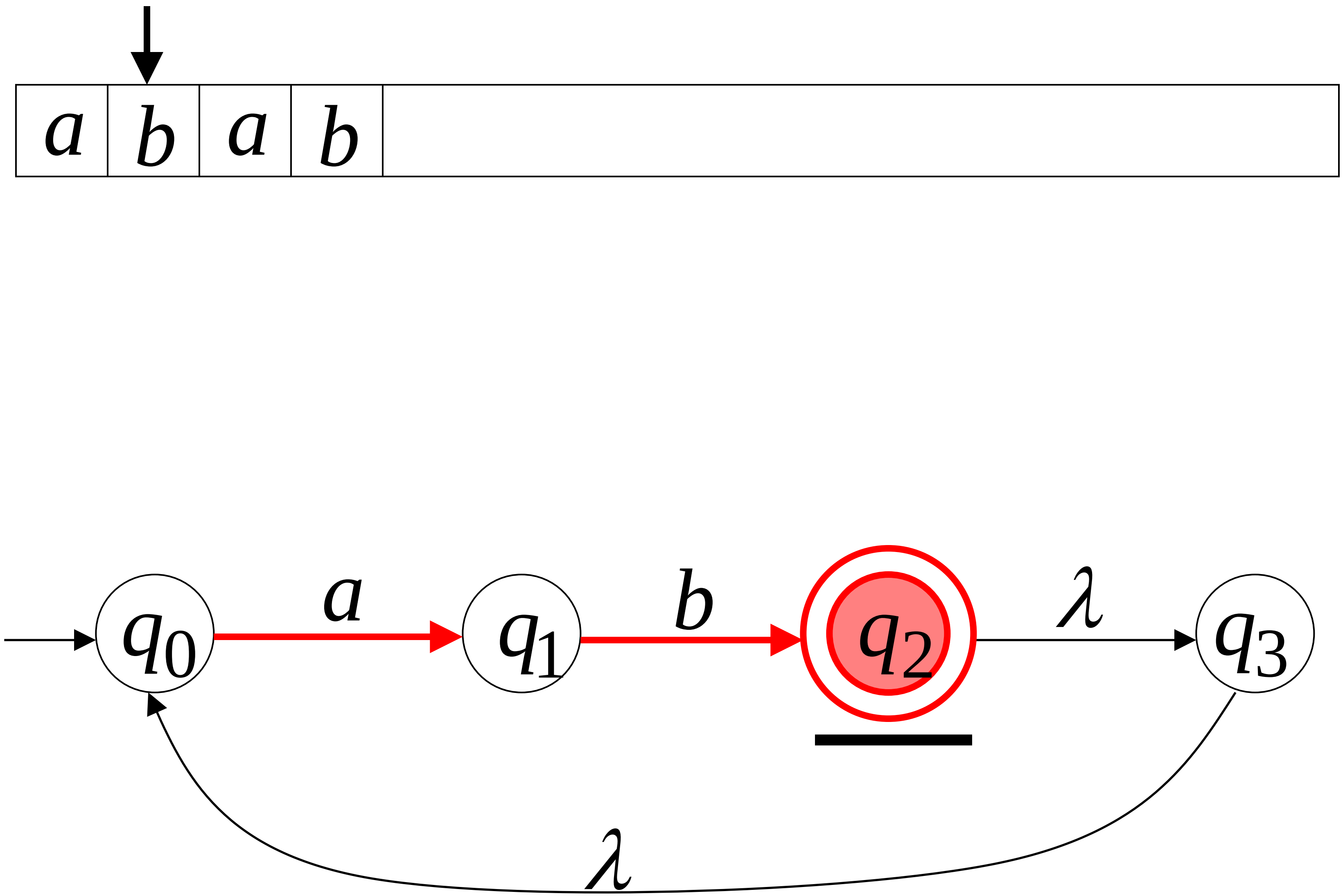
Lambda Transition

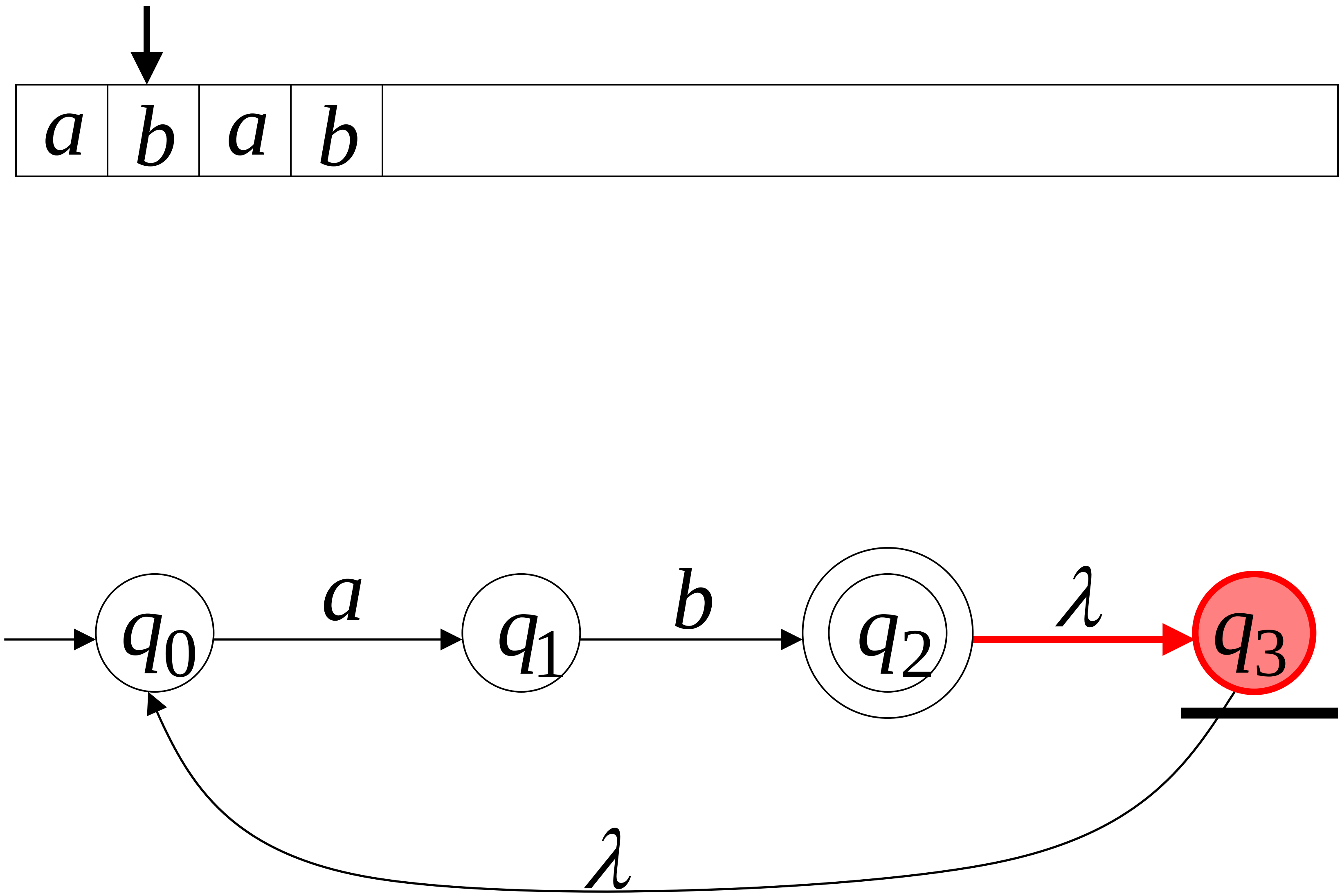
A transition that does not need to read an input alphabet.

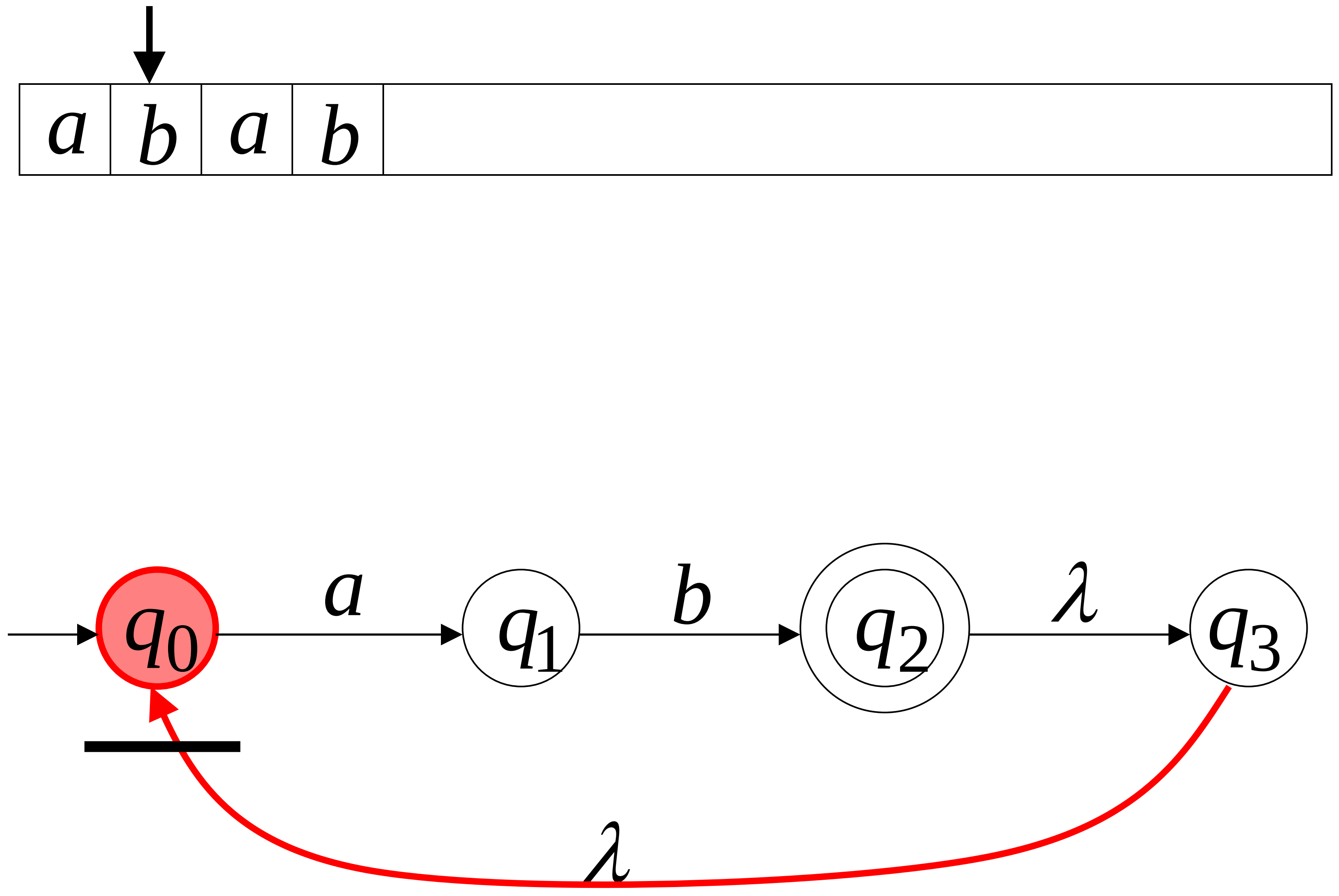


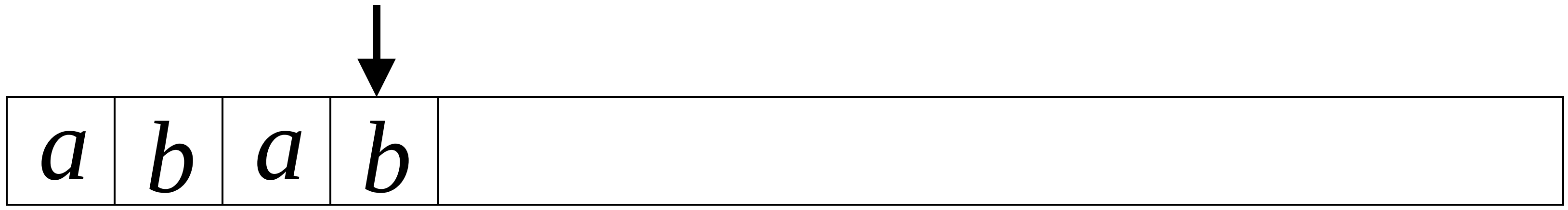
Example

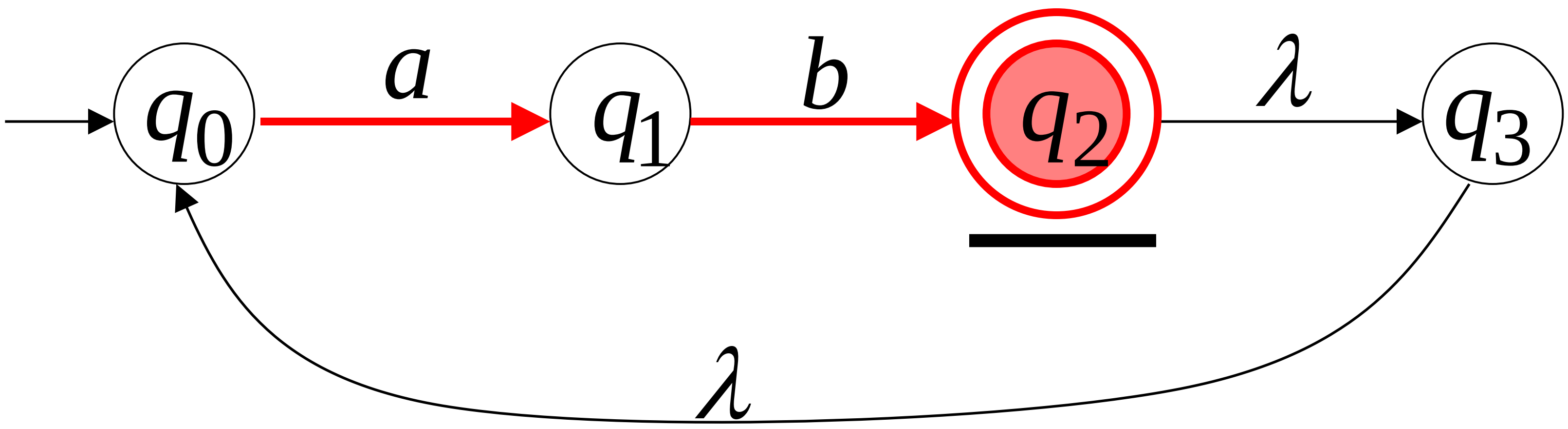


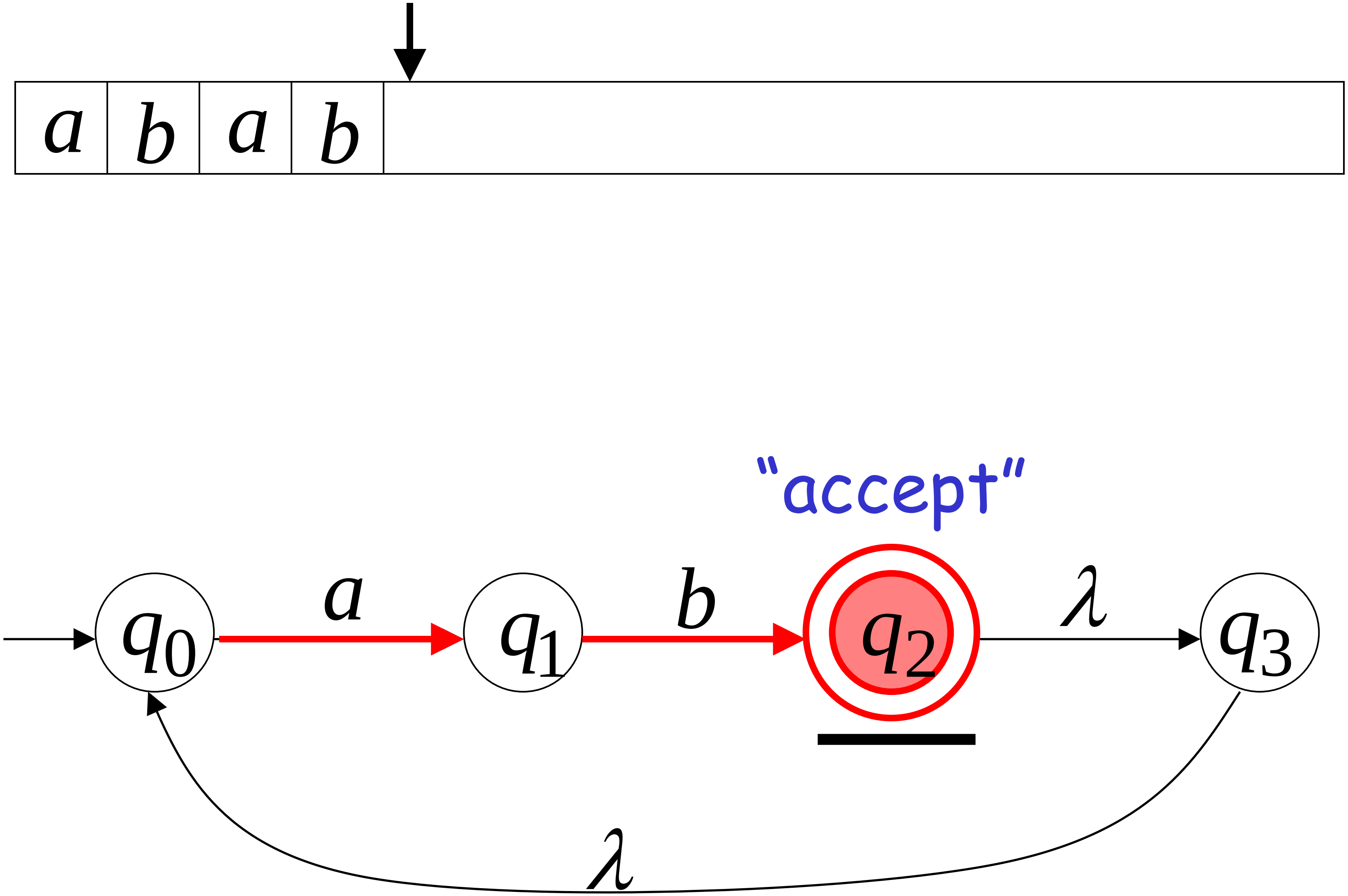

a
b
a
b
a
b
λ
q0
q1
q2
q3
λ


a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3


a
b
a
b
a
b
λ
λ
q0
q1
q2
q3


a
b
a
b

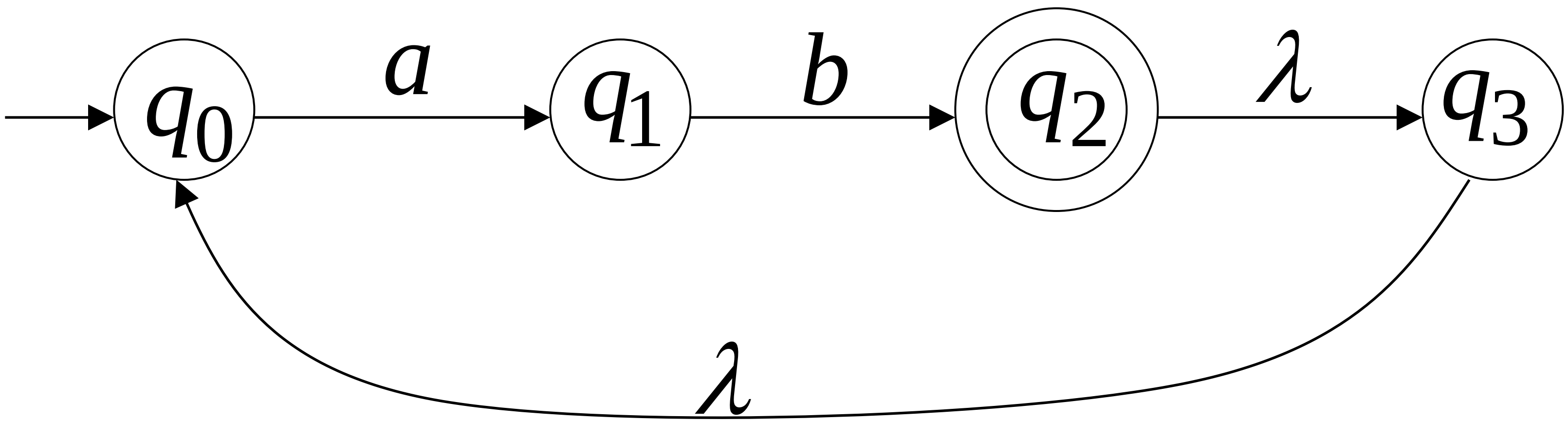

q0
a
q1
b
q2
λ
q3
λ


a
b
a
b
"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

Language accepted

$$L = \{ab,\ abab,\ ababab,\ \ldots\}$$

$$= \{ab\}^+$$

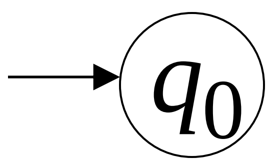


Remarks:

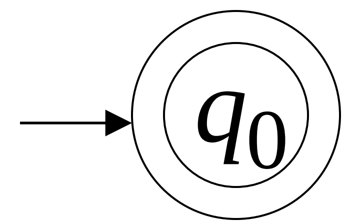
- The λ symbol never appears on the input tape

- Simple automata:

M_1



M_2

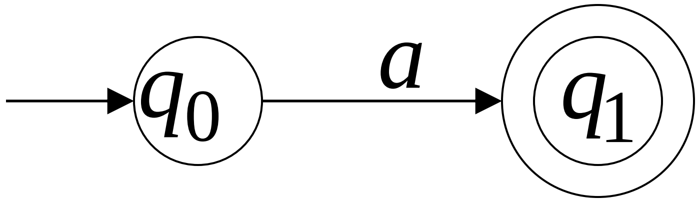


$L(M_1) = \{\}$

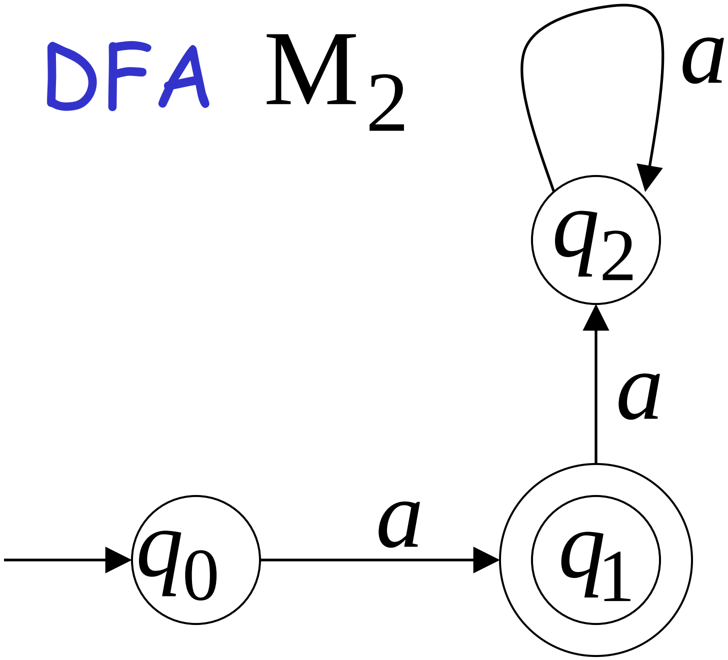
$L(M_2) = \{\lambda\}$

- NFAs are interesting because we can express languages easier than DFAs

NFA M_1



DFA M_2



$$L(M_1) = \{a\}$$

$$L(M_2) = \{a\}$$

Formal Definition of NFAs

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

$Q:$ Set of states, i.e. $\{q_0, q_1, q_2\}$

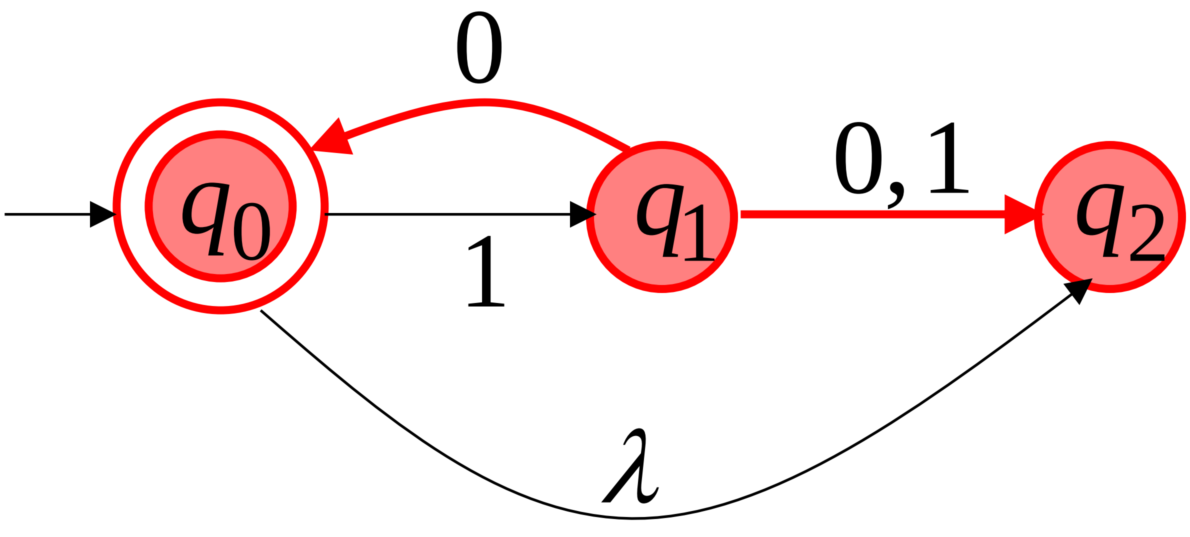
$\Sigma:$ Input aplhabet, i.e. $\{a, b\}$

$\delta:$ Transition function ($\delta: Q \times \Sigma \rightarrow 2^Q$)

$q_0:$ Initial state

$F:$ Final states

$$\delta(q_1, 0) = \{q_0, q_2\}$$



Formally

The language accepted by NFA M is:

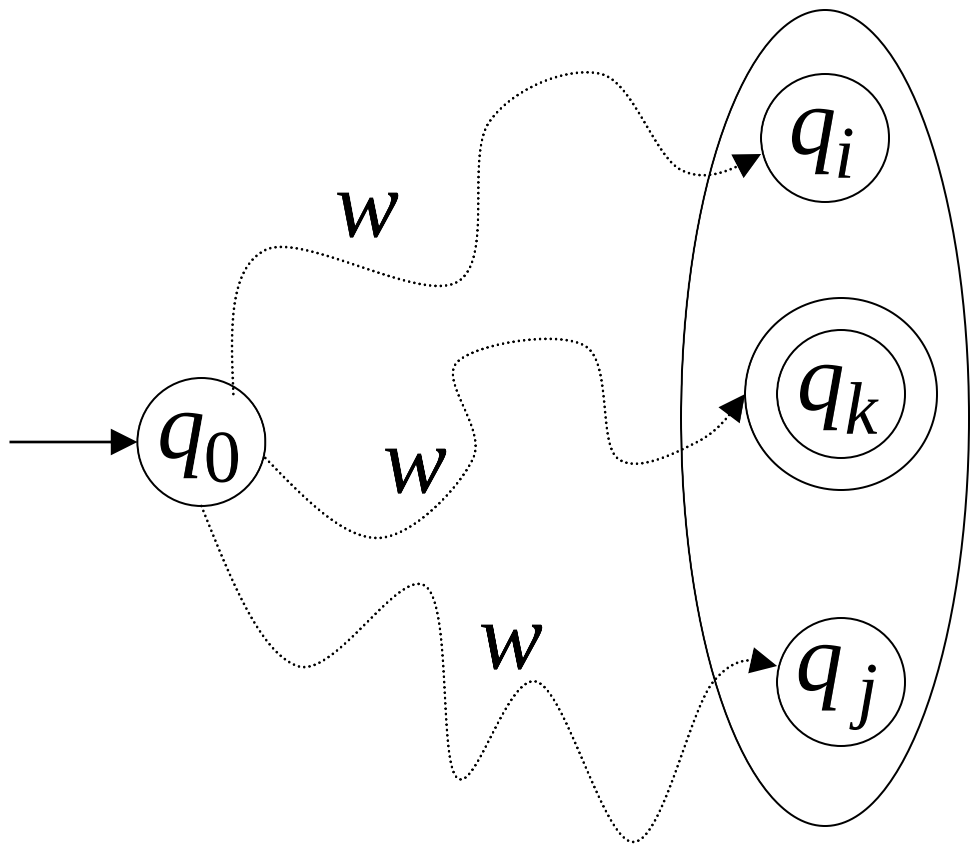
$$L(M) = \{w_1, w_2, w_3, ...\}$$

where $\delta^*(q_0, w_m) = \{q_i, q_j, ..., q_k, \ldots\}$

and there is some $q_k \in F$ (final state)

$w \in L(M)$

$\delta^*(q_0, w)$



$q_k \in F$

NFAs accept the Regular Languages

Equivalence of Machines

Definition for Automata:

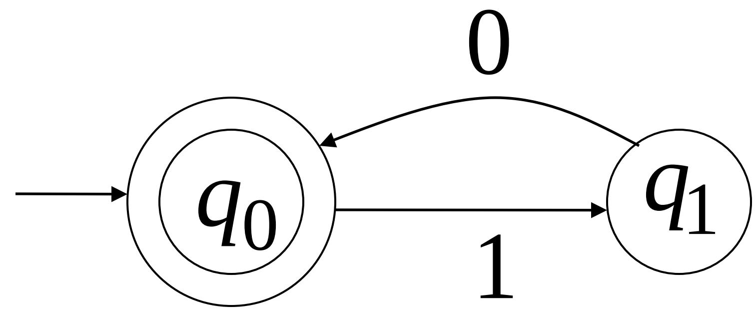
Machine M_1 is equivalent to machine M_2

if $L(M_1) = L(M_2)$

Example of equivalent machines

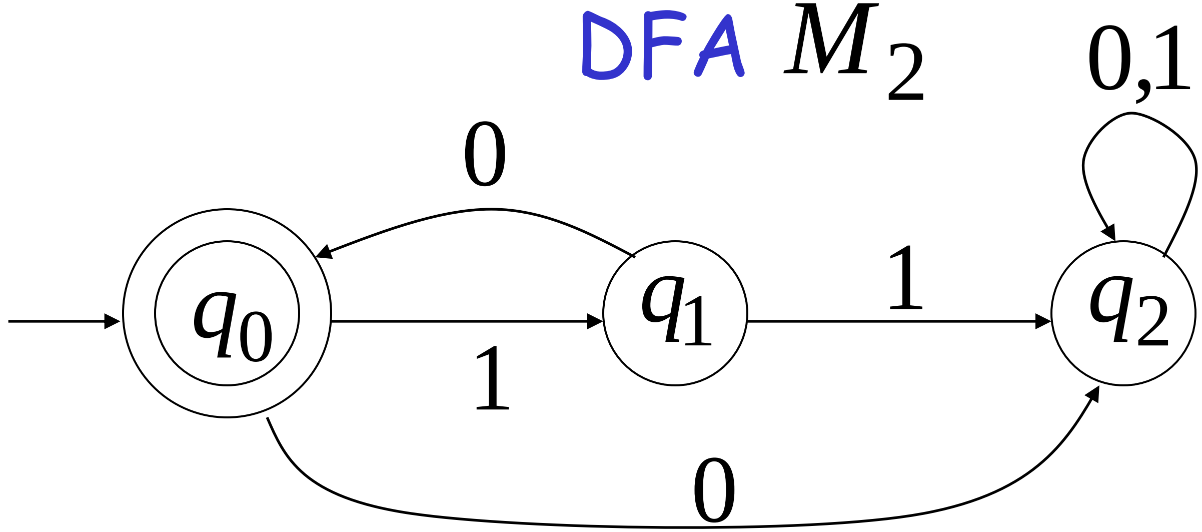
NFA M_1

$$L(M_1) = \{10\}^*$$



DFA M_2

$$L(M_2) = \{10\}^*$$



We will prove:

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Languages
accepted
by DFAs

NFAs and DFAs have the same computation power

Step 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Proof: Every DFA is trivially an NFA

$\Downarrow$

Any language L accepted by a DFA is also accepted by an NFA

Step 2

$$\left\{\begin{array}{l}\text{Languages}\\\text{accepted}\\\text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\}$$

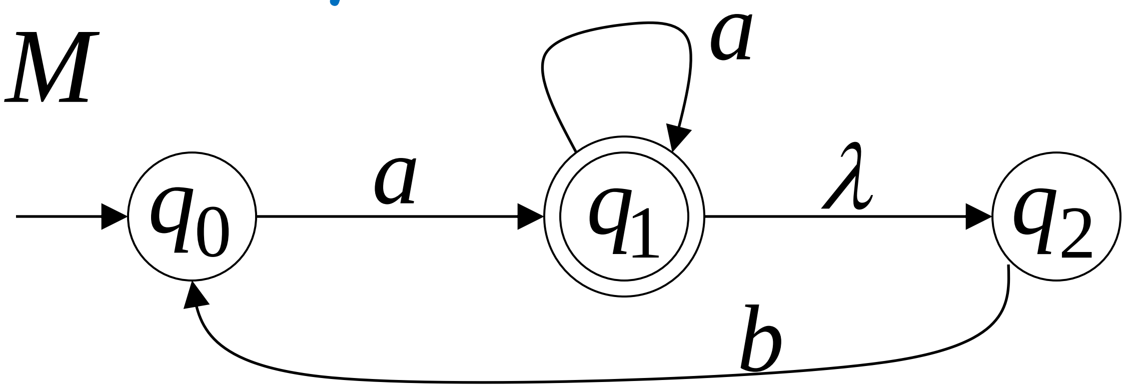
Proof: Any NFA can be converted to an equivalent DFA

$\Downarrow$

Any language L accepted by an NFA is also accepted by a DFA

Convert NFA to DFA (Systematic method)

NFA M

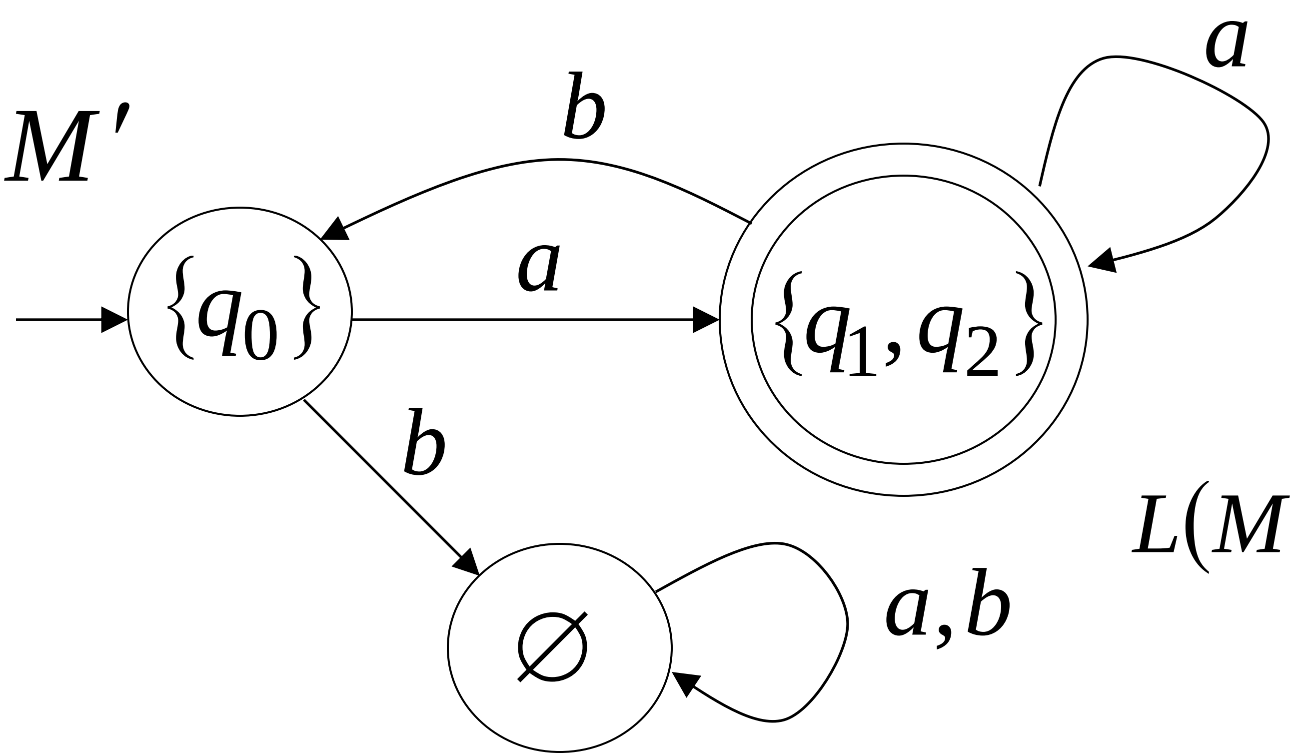


Create a transition table

δ	a	b
q_0	q_1, q_2	$\emptyset$
q_1	q_1, q_2	q_0
q_2	$\emptyset$	q_0

δ	a	b
q_0	q_1, q_2	$\varnothing$
q_1	q_1, q_2	q_0
q_2	$\varnothing$	q_0

DFA M'



$$L(M) = L(M')$$