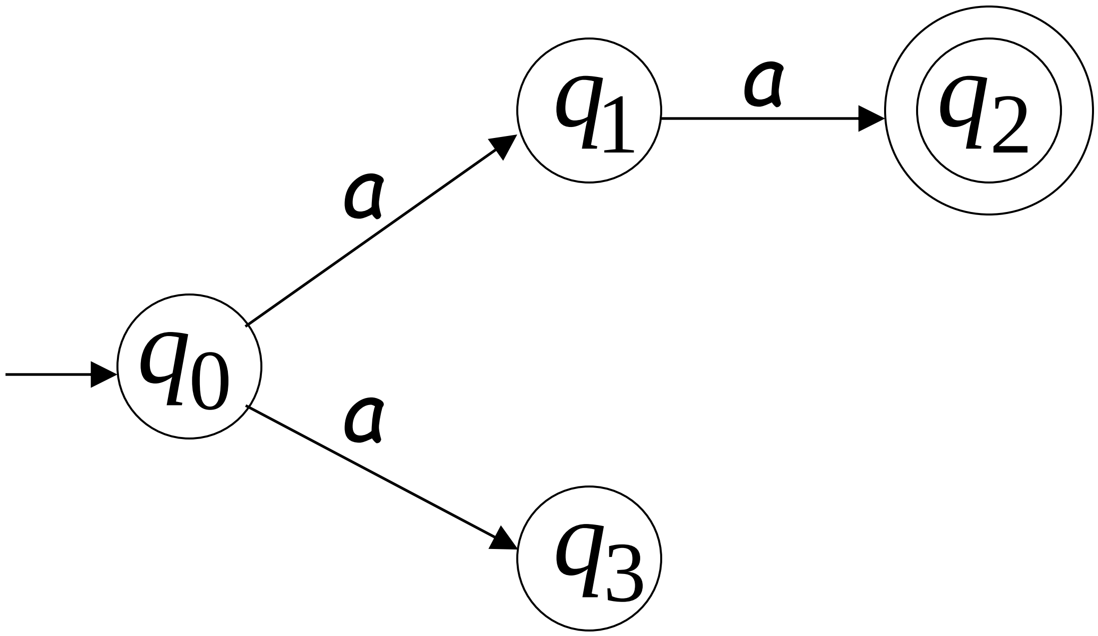
Nondeterministic Finite Automata

Nondeterministic Finite Automata (NFA)

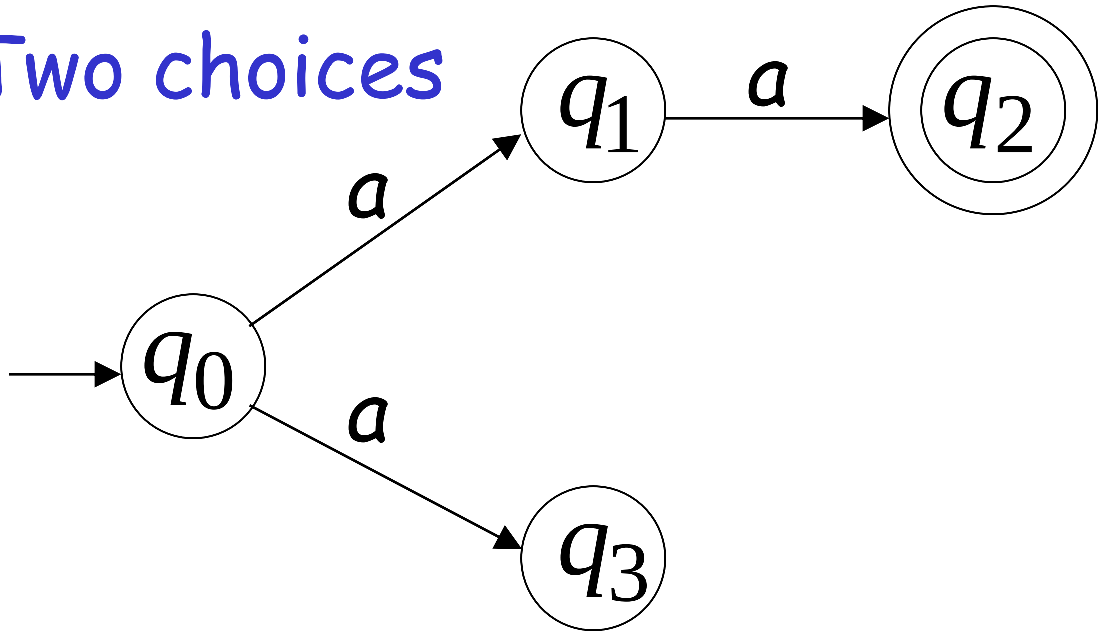
Alphabet = $\{a\}$



Nondeterministic Finite Automata (NFA)

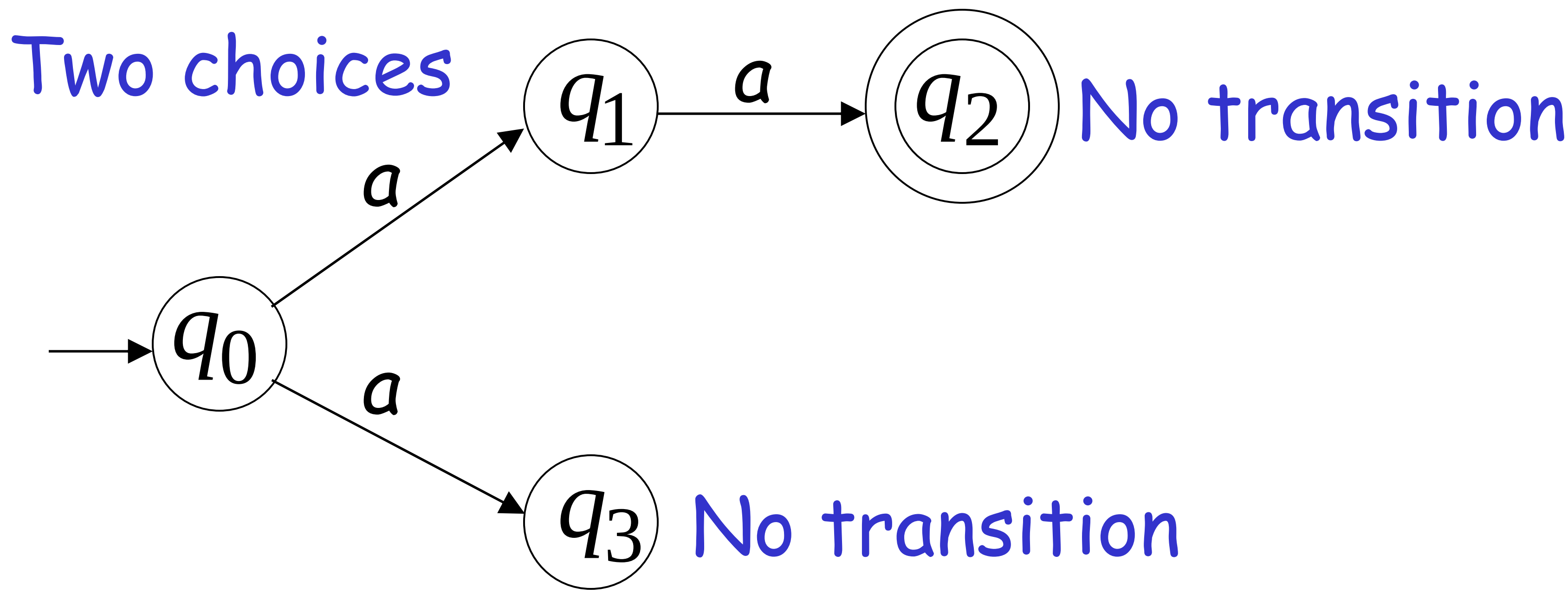
Alphabet = $\{a\}$

Two choices

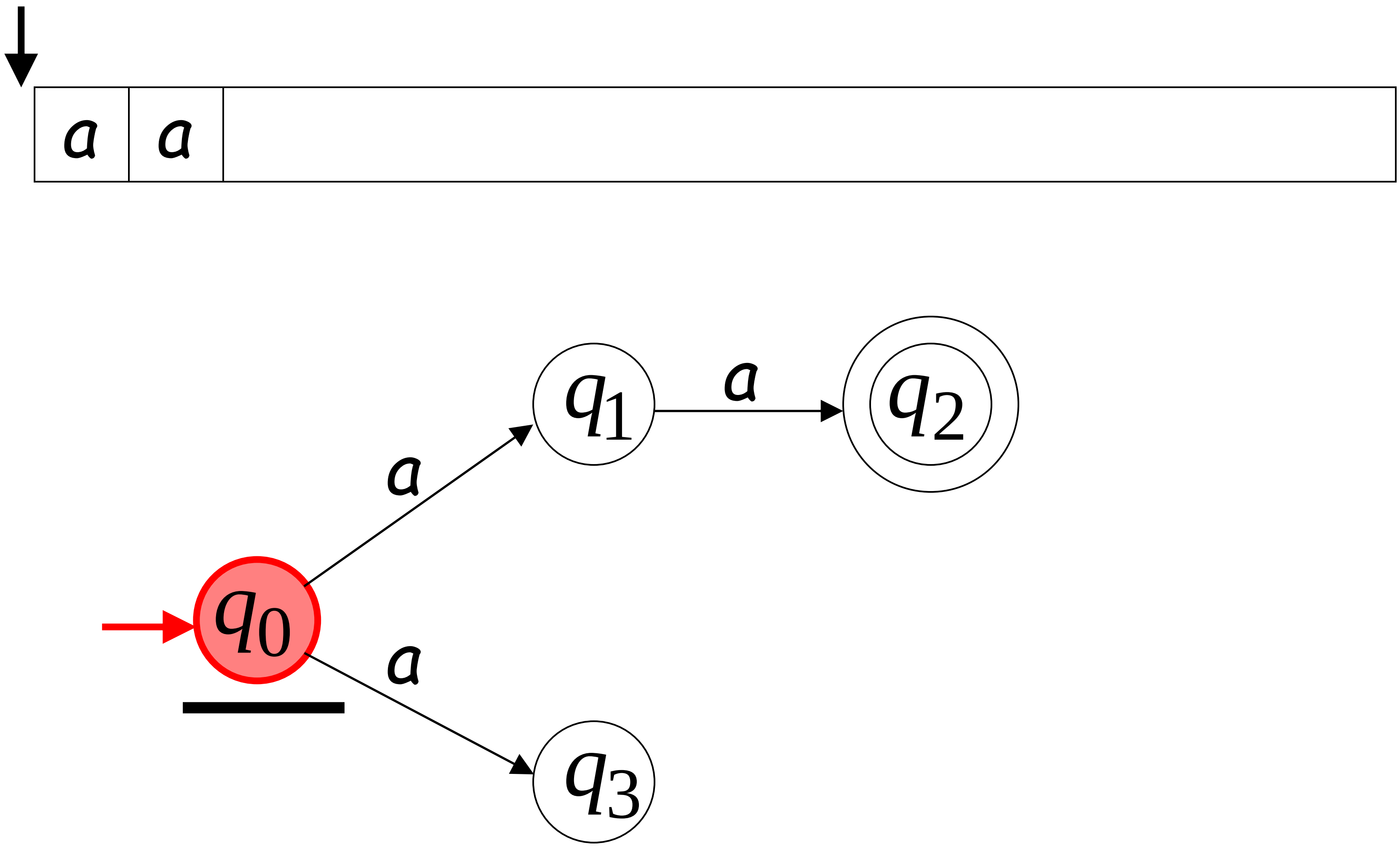


Nondeterministic Finite Automata (NFA)

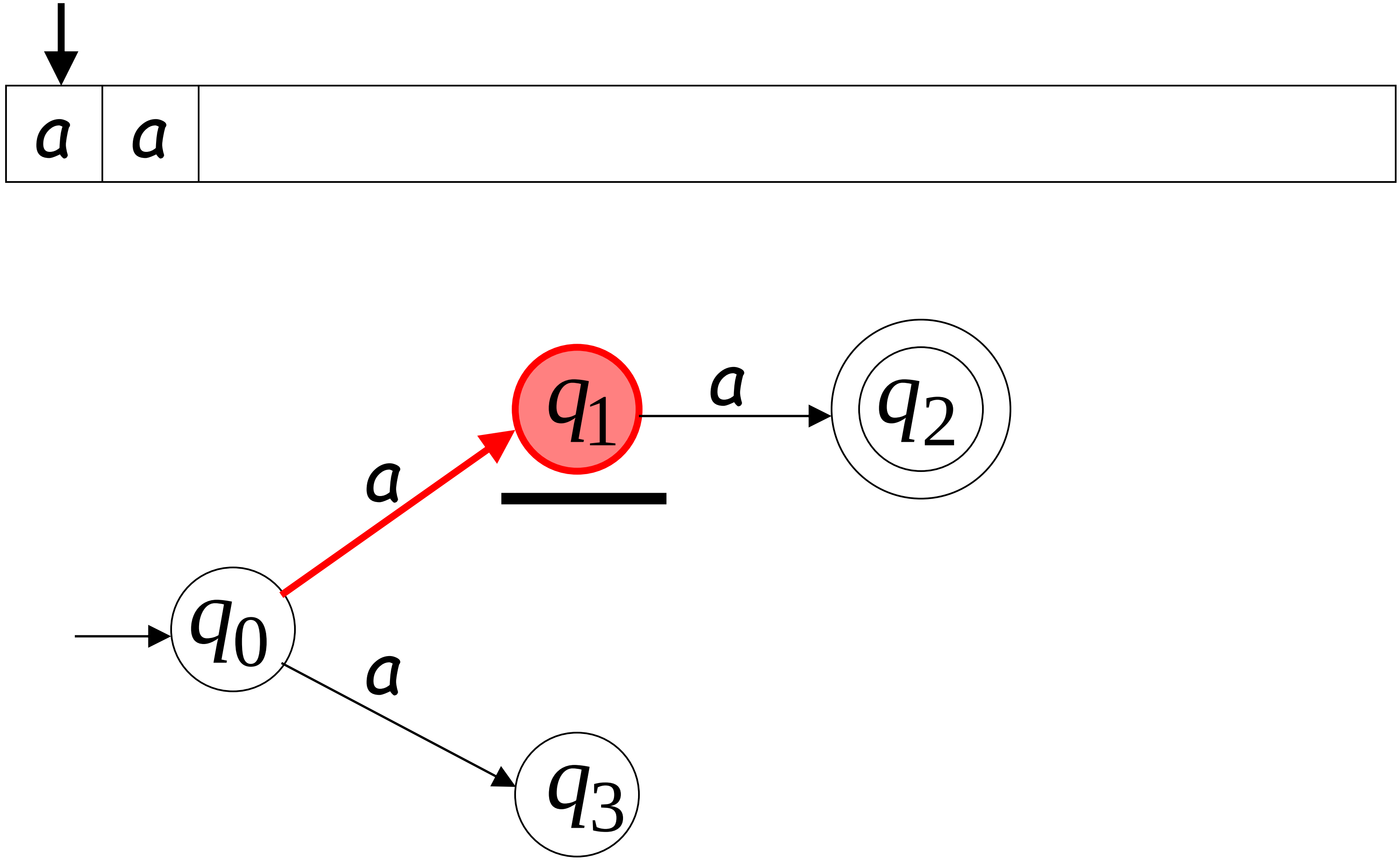
Alphabet = $\{a\}$



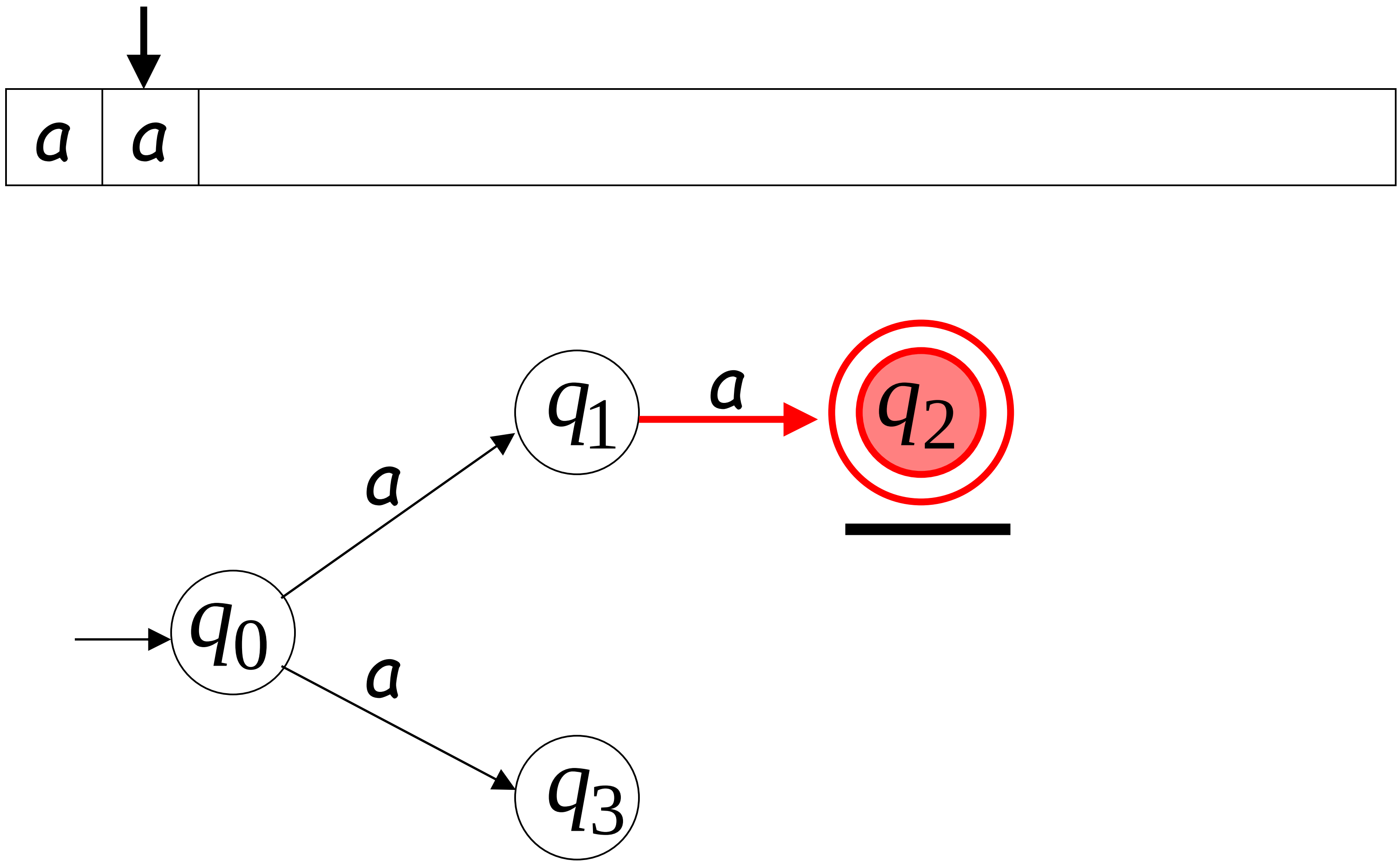
First Choice



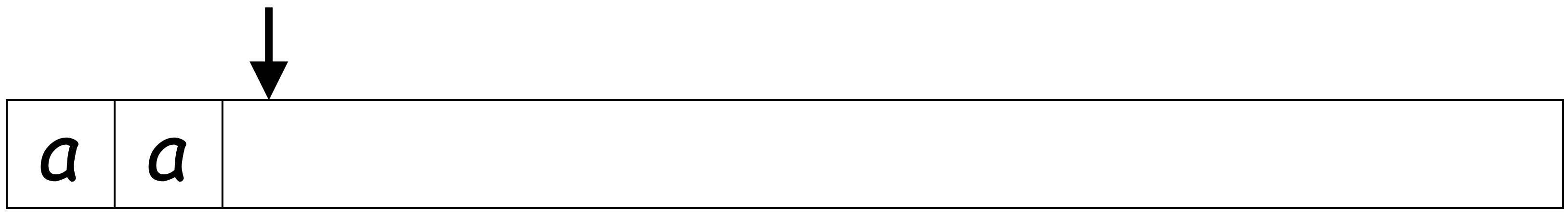
First Choice



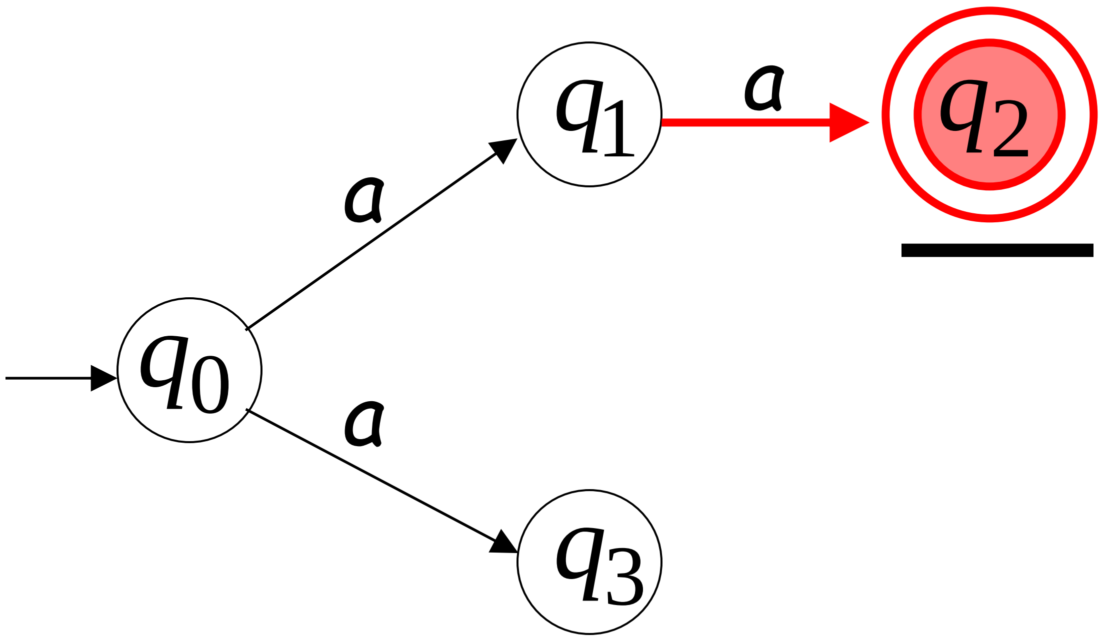
First Choice



First Choice

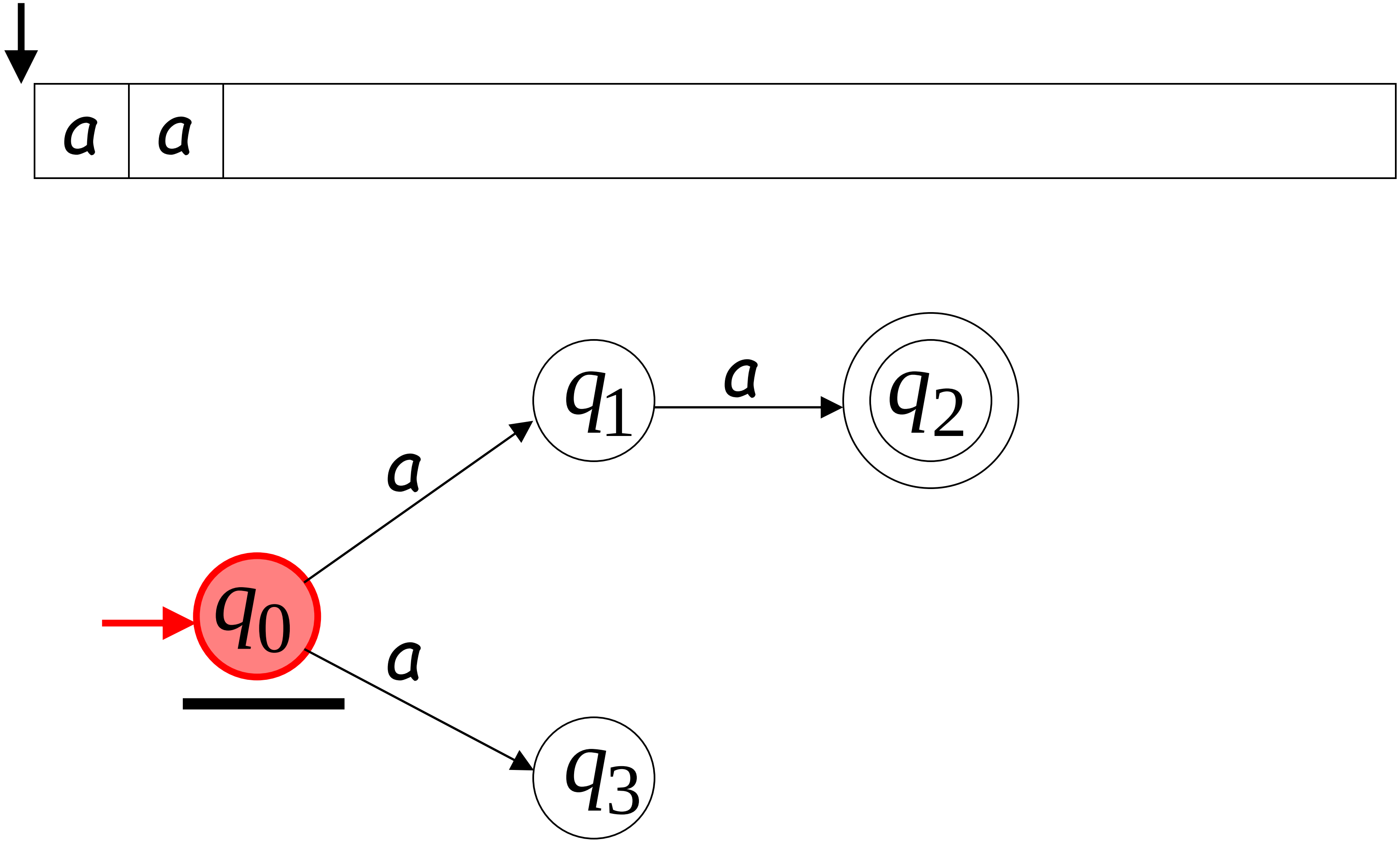


All input is consumed

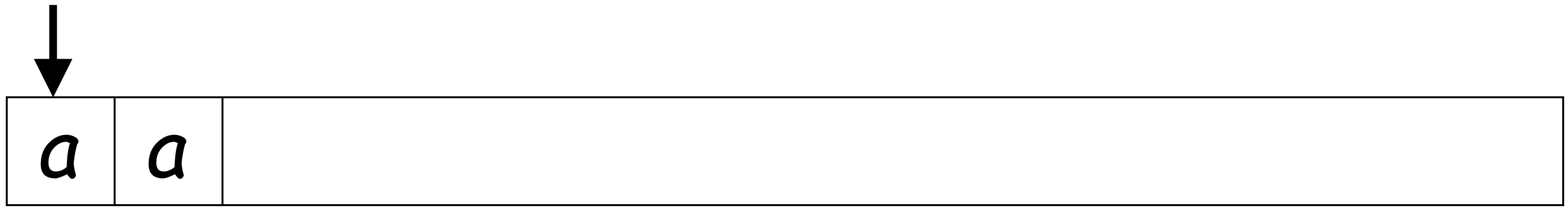


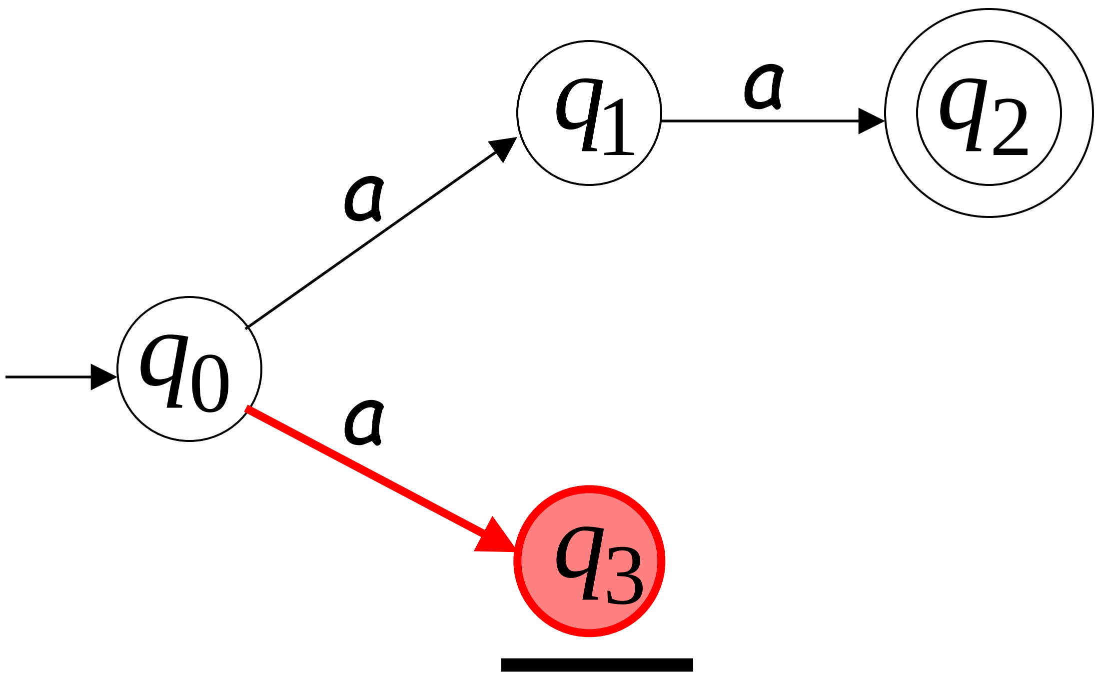
"accept"

Second Choice

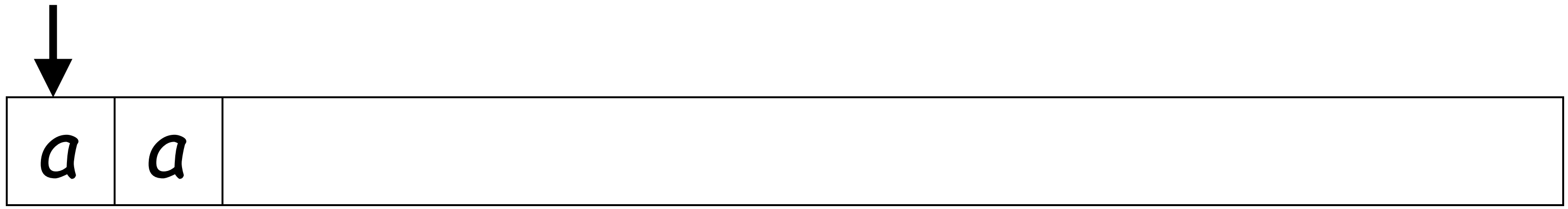


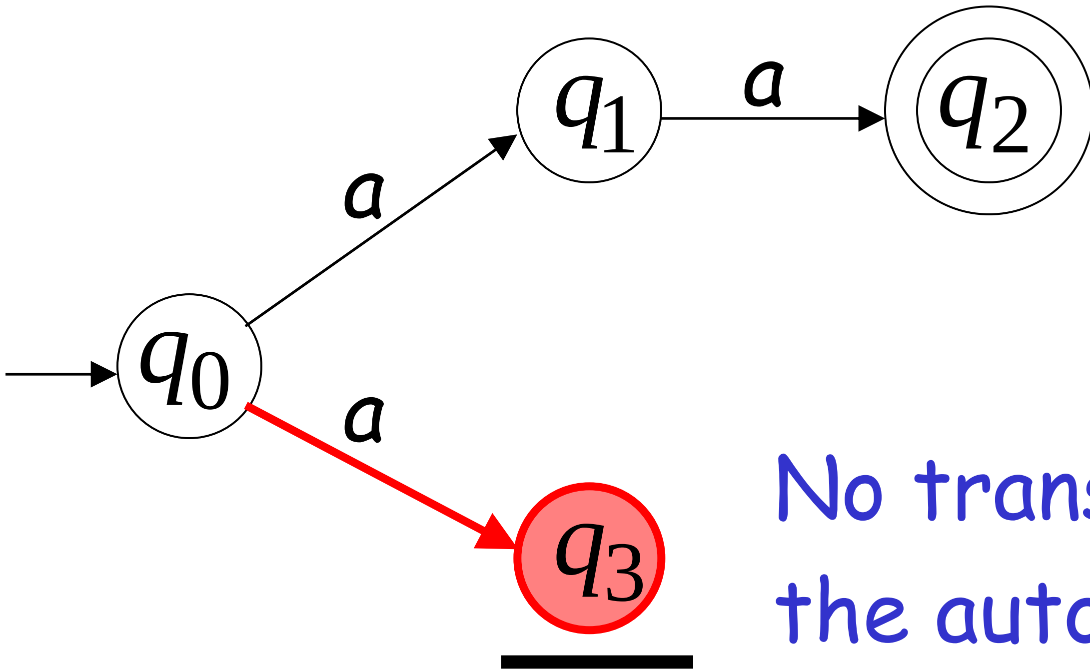
Second Choice





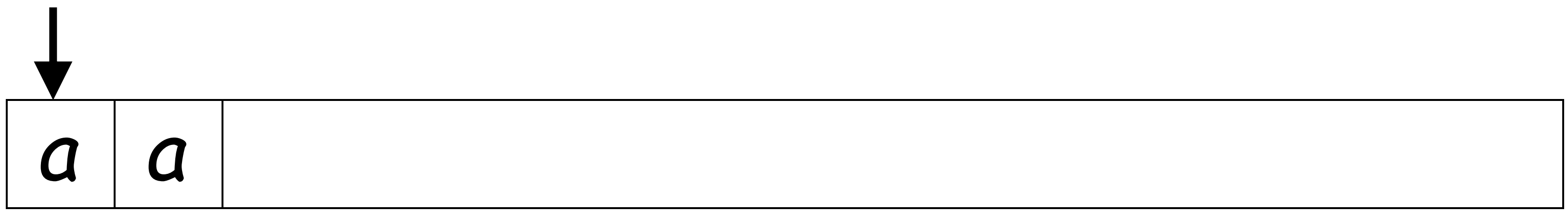
Second Choice



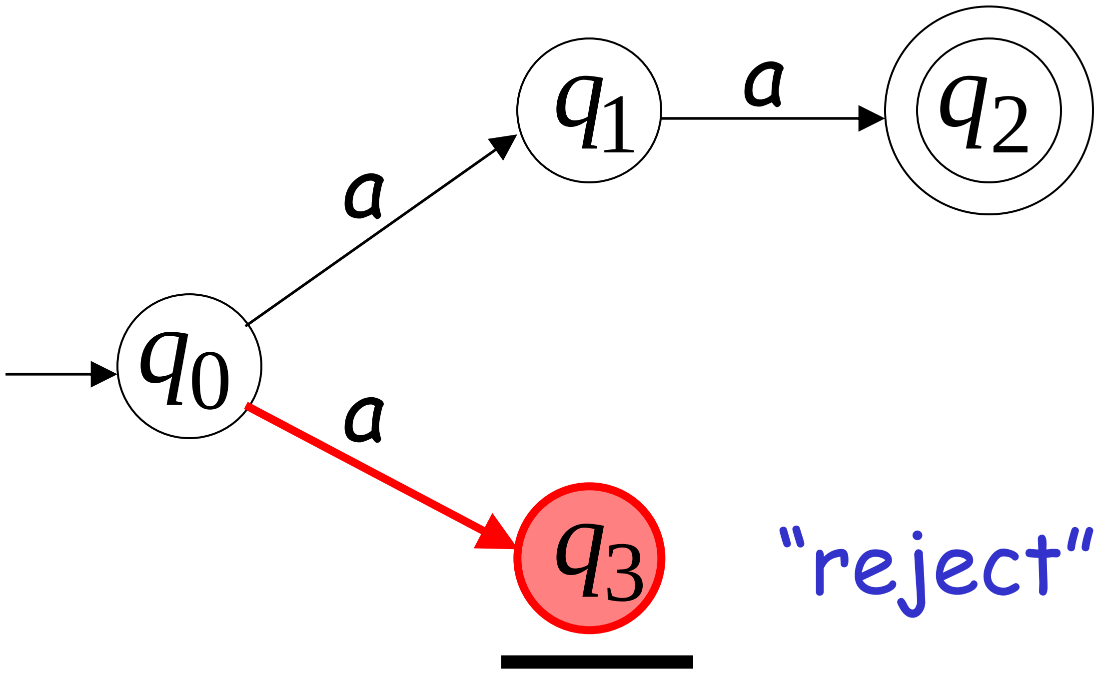


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



An NFA accepts a string:

If there is a computation such that:

All the input is consumed

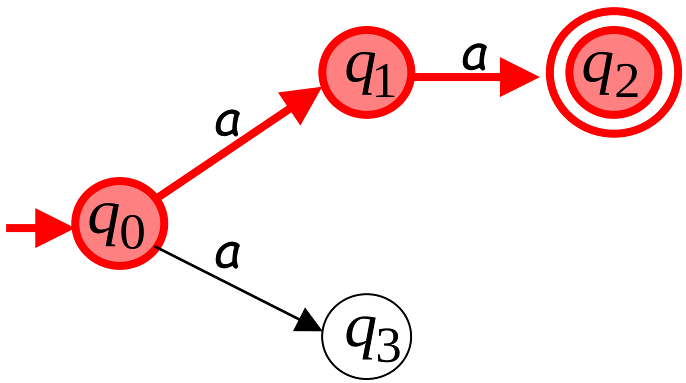
AND

The automata is in a final state

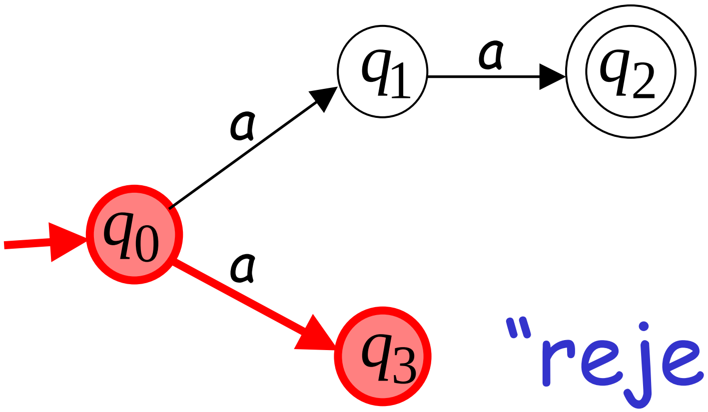
Example

aa is accepted by the NFA:

"accept"

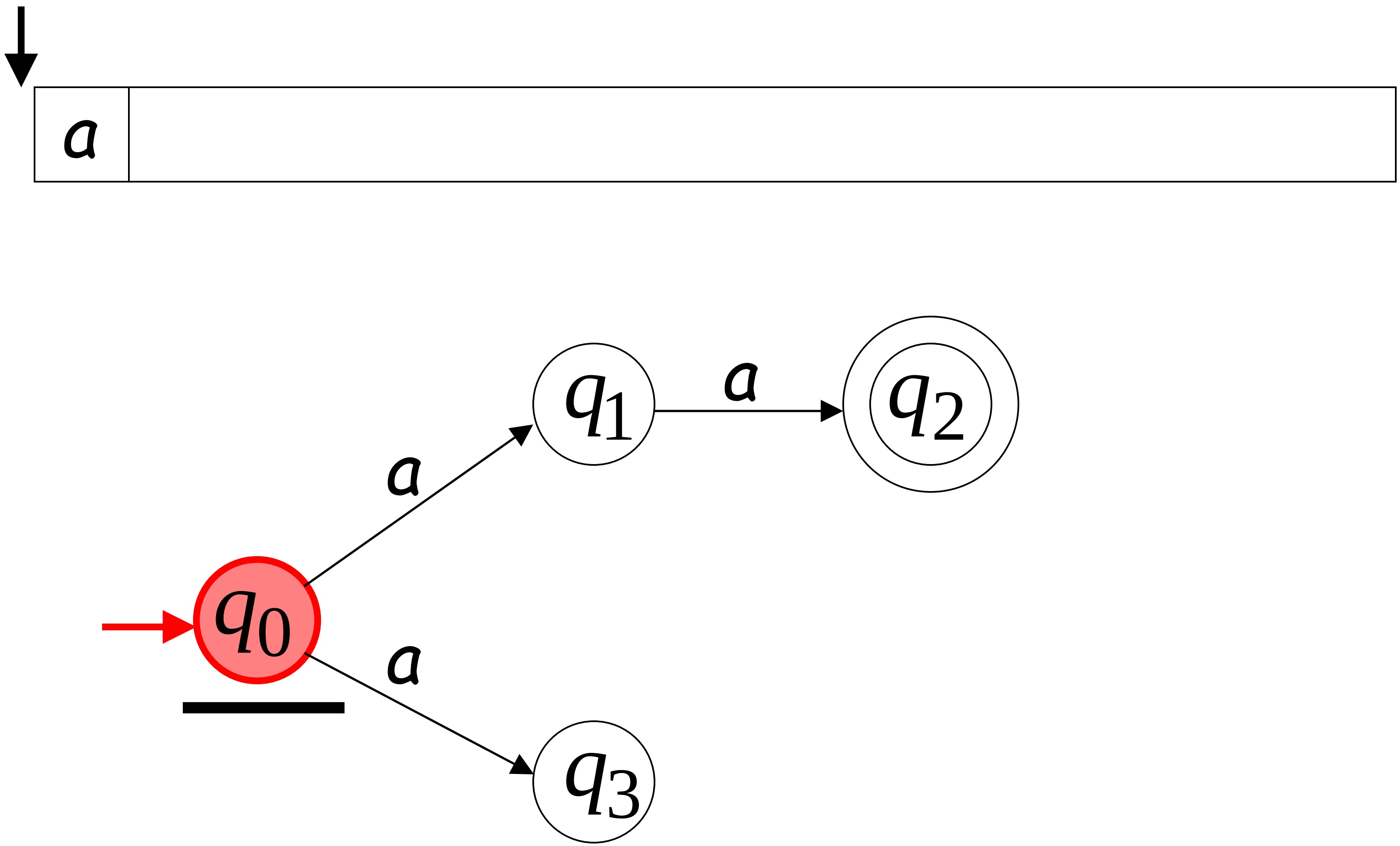


because this computation accepts aa

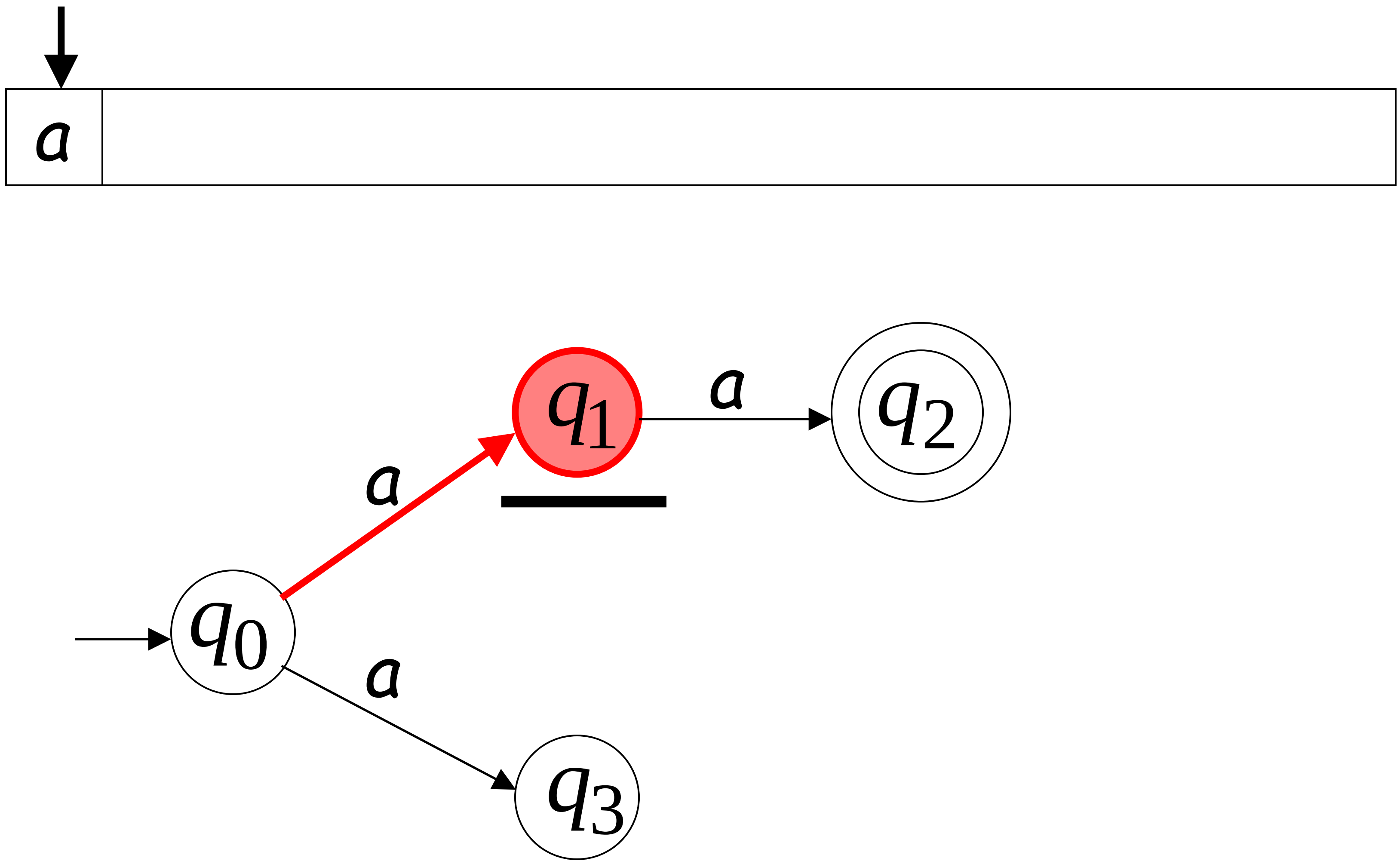


"reject"

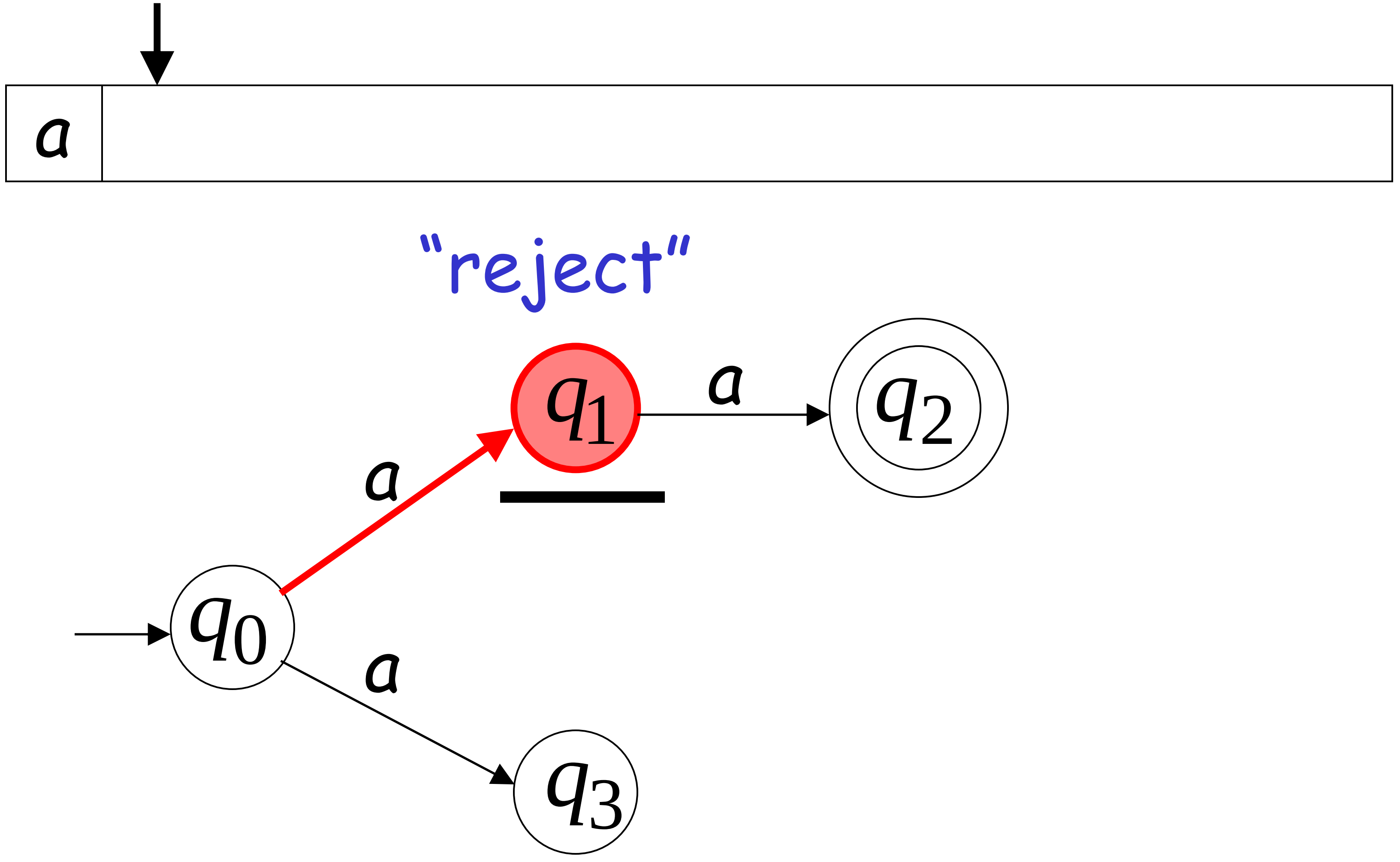
Rejection example



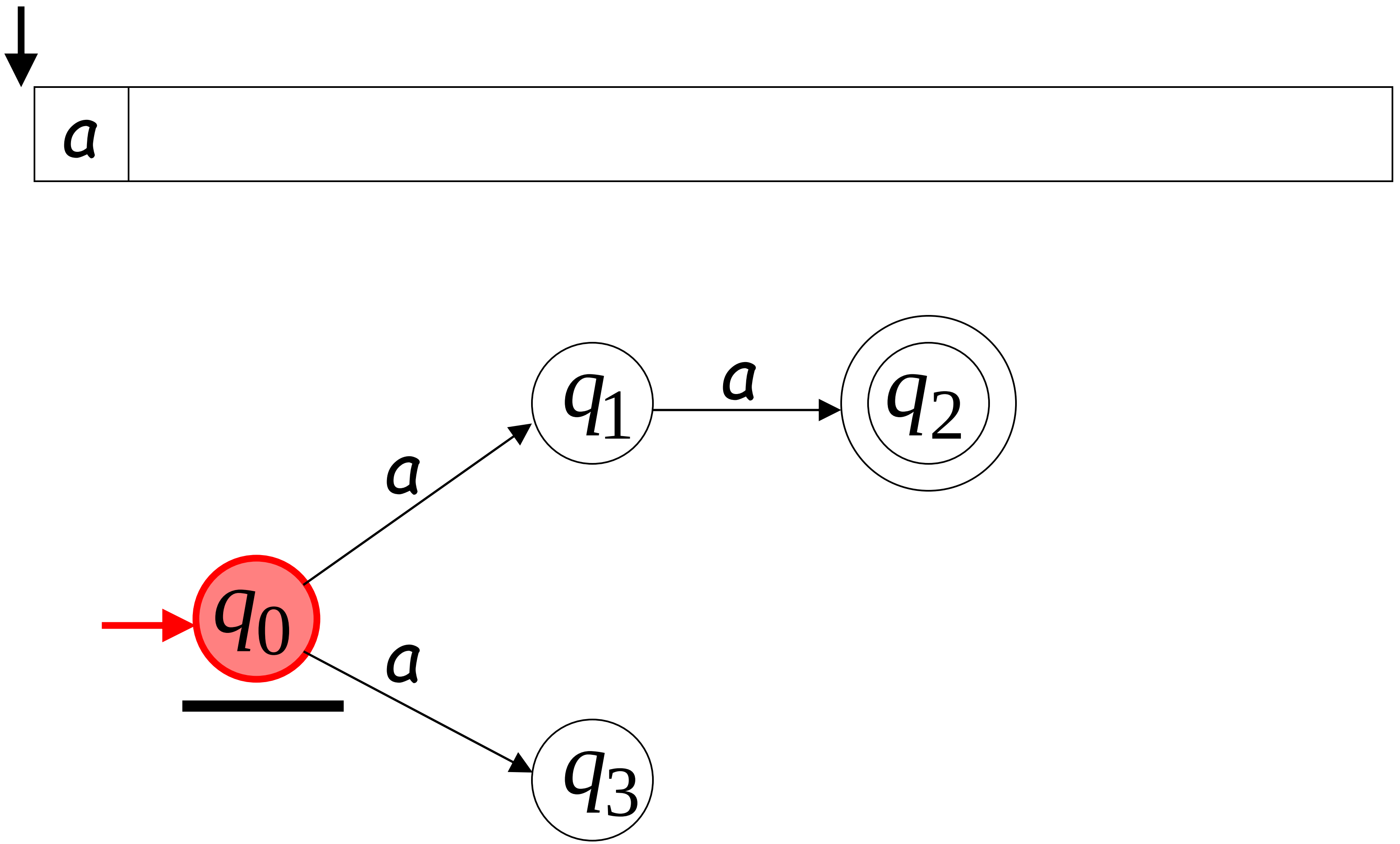
First Choice



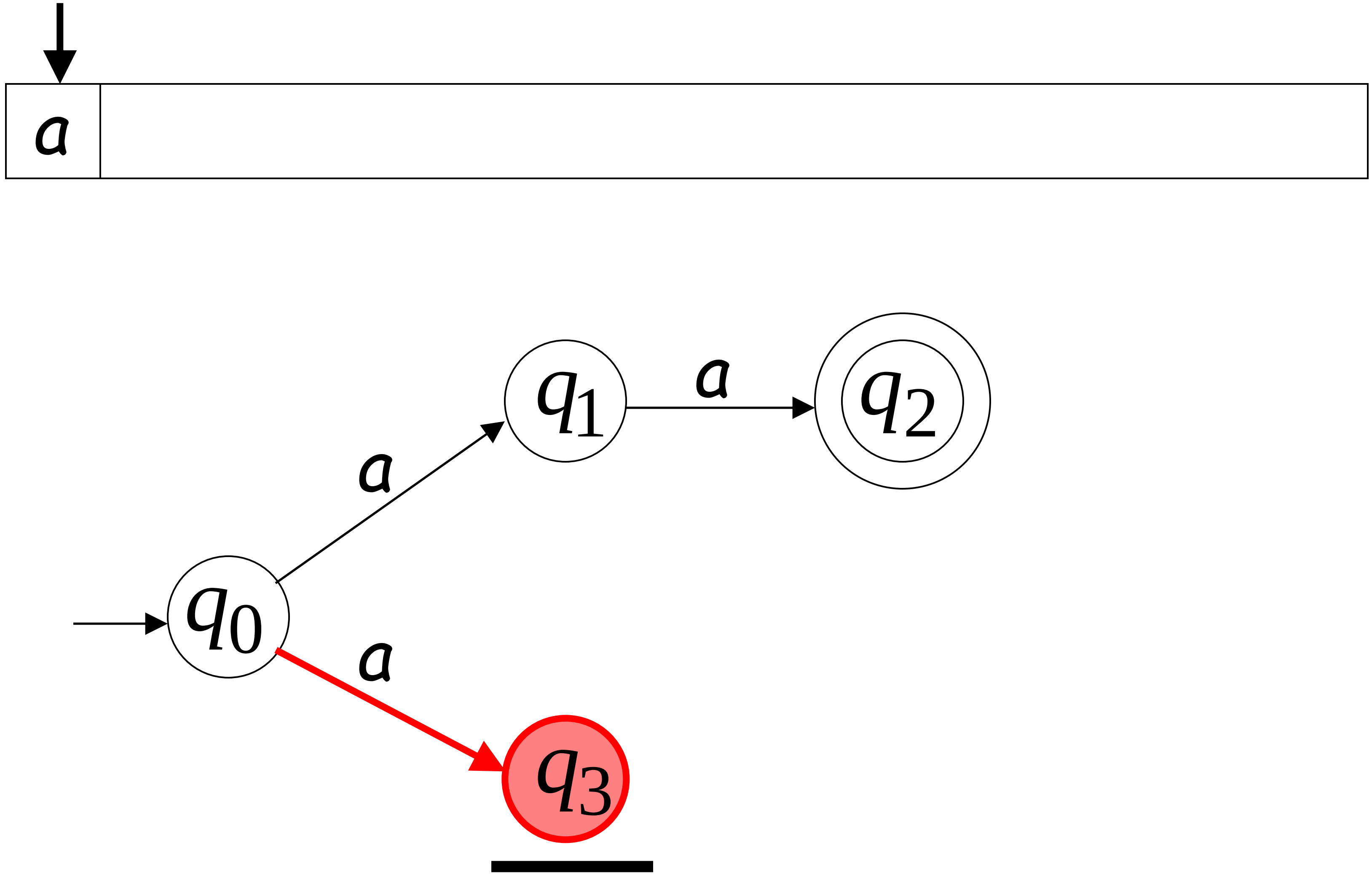
First Choice



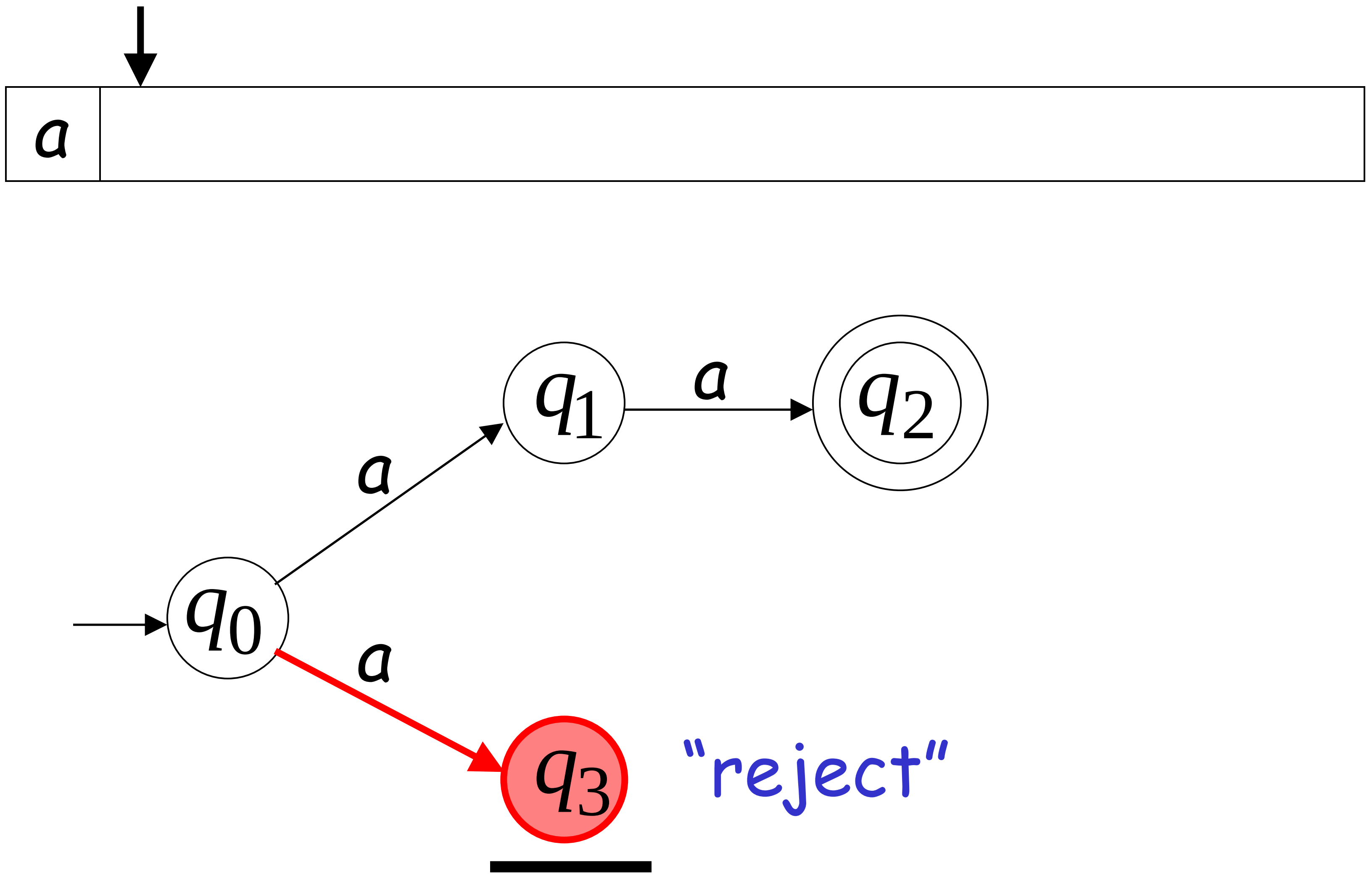
Second Choice



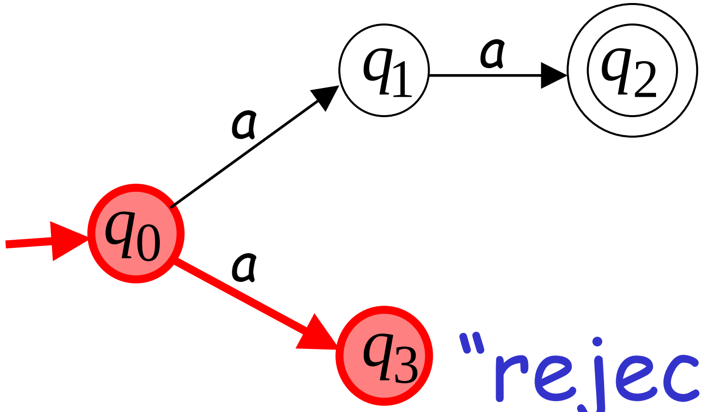
Second Choice



Second Choice

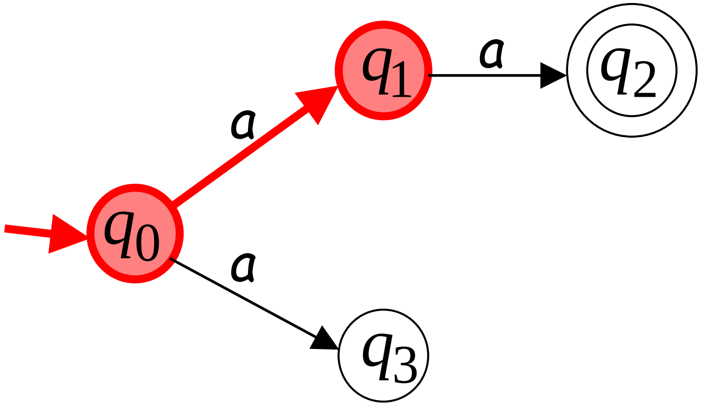


a is rejected by the NFA:



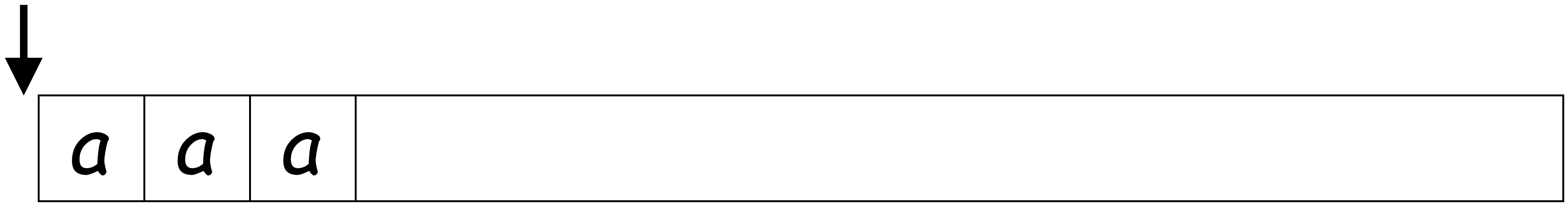
"reject"

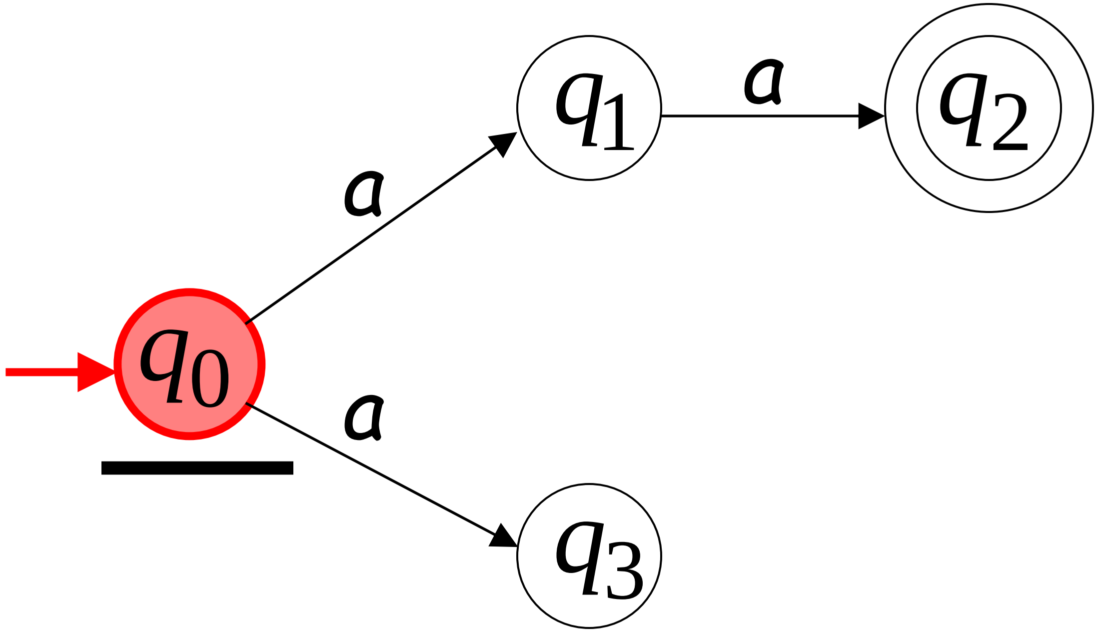
"reject"



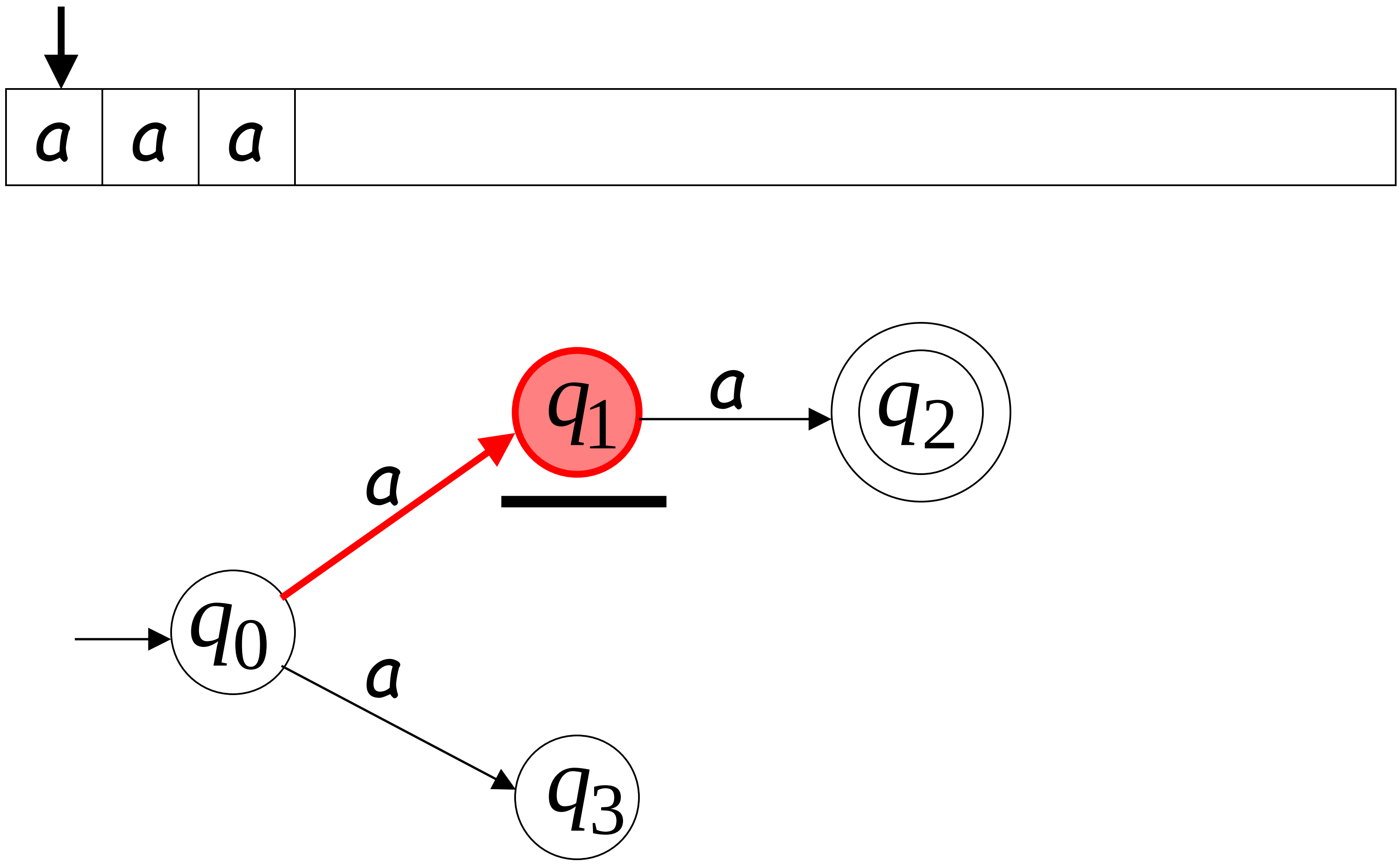
All possible computations lead to rejection

Rejection example

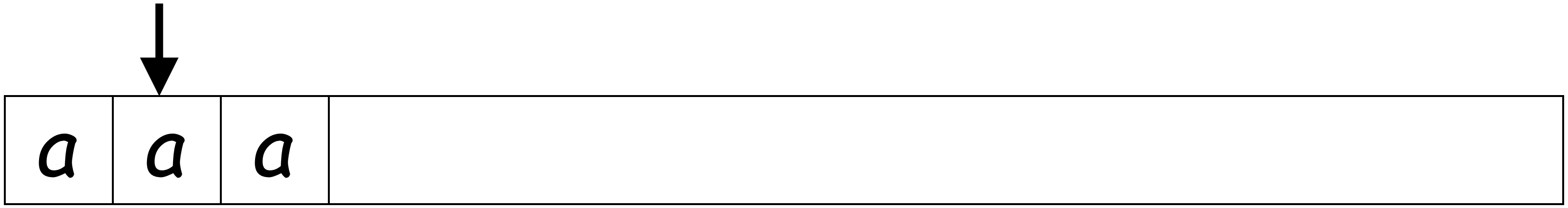


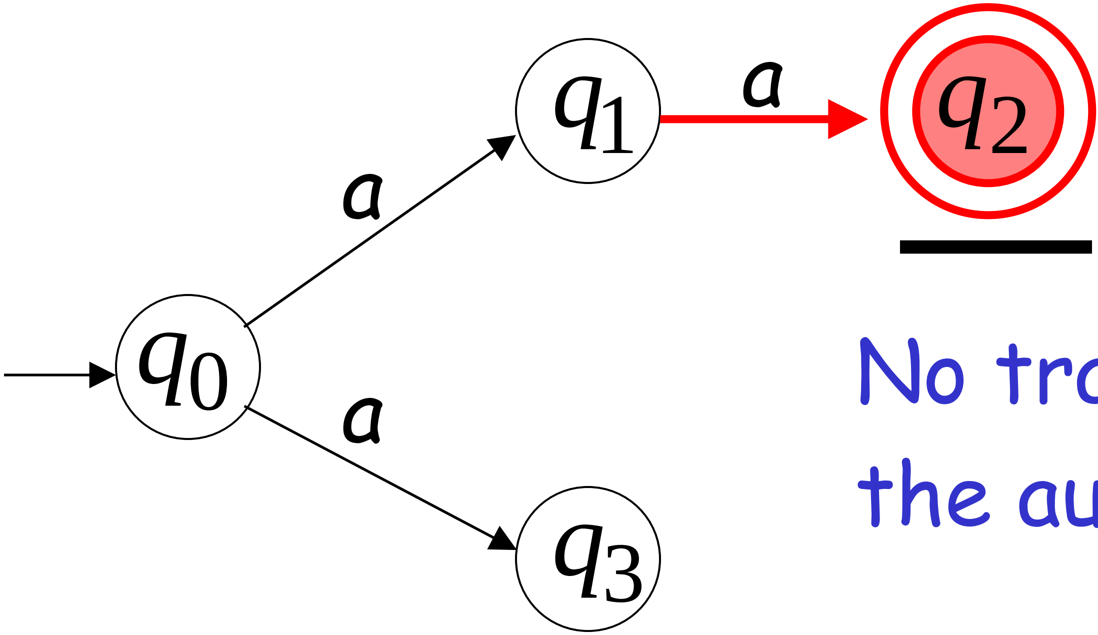


First Choice



First Choice

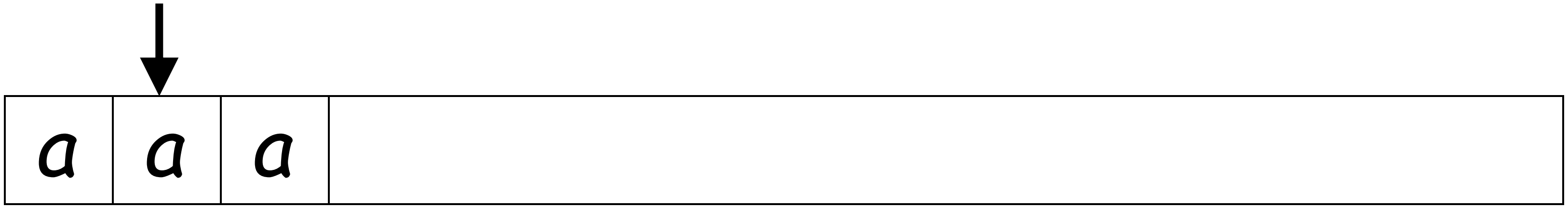




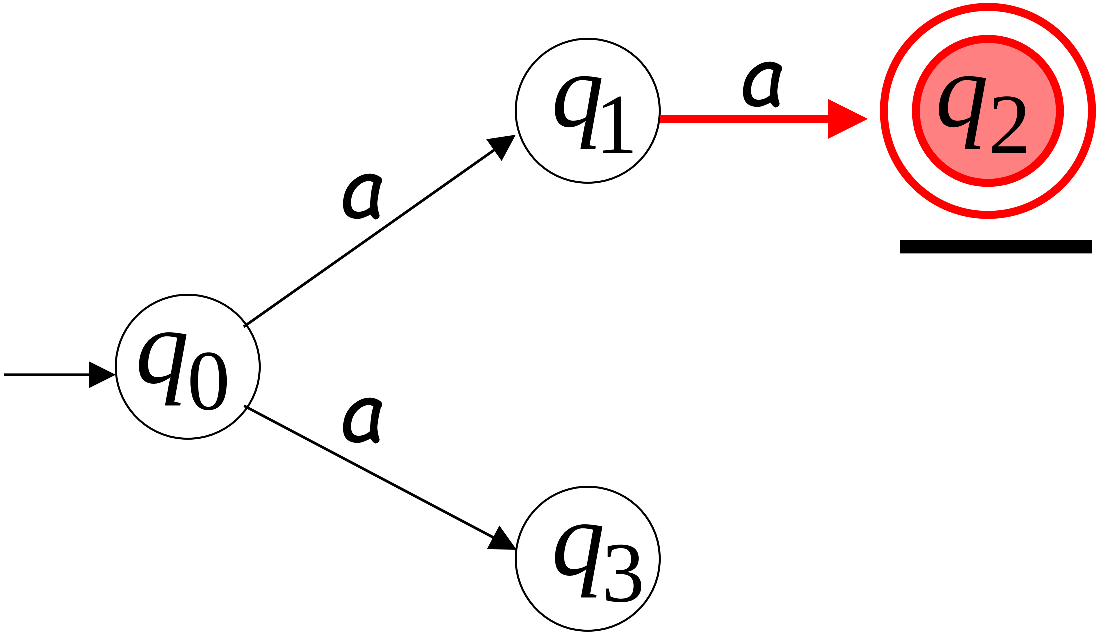
No transition:

the automaton hangs

First Choice

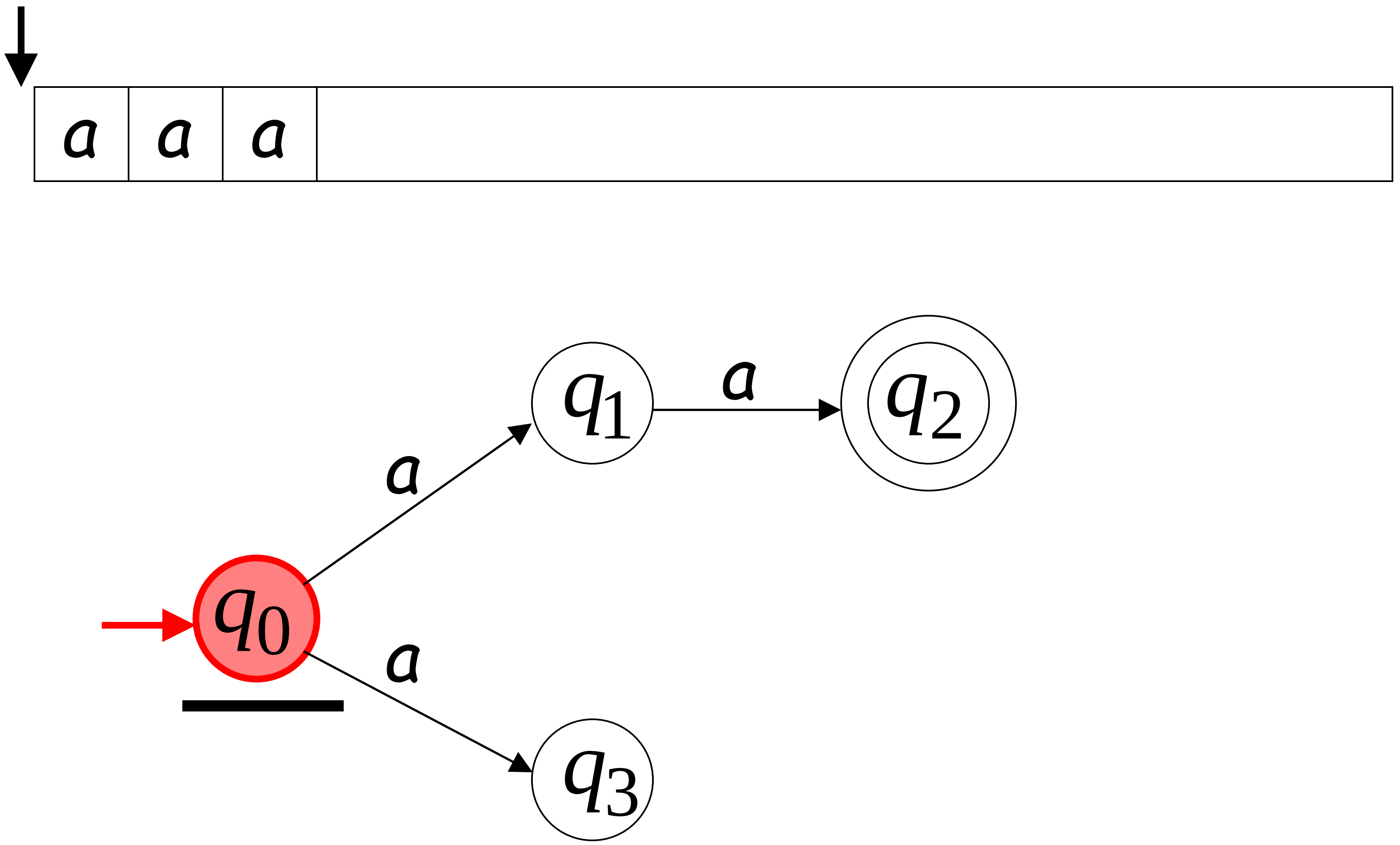


Input cannot be consumed

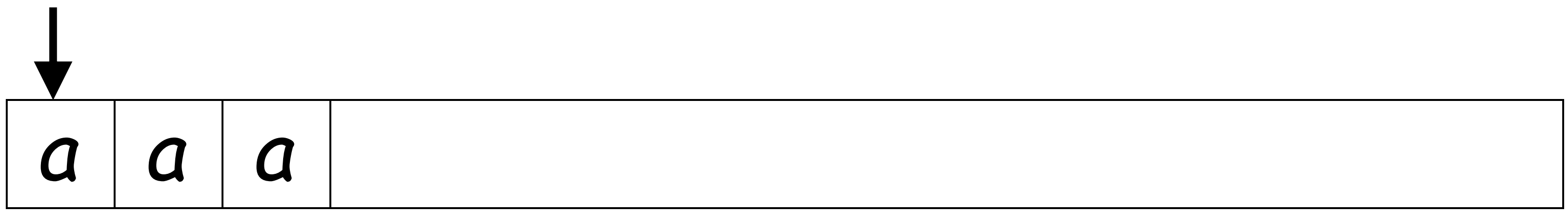


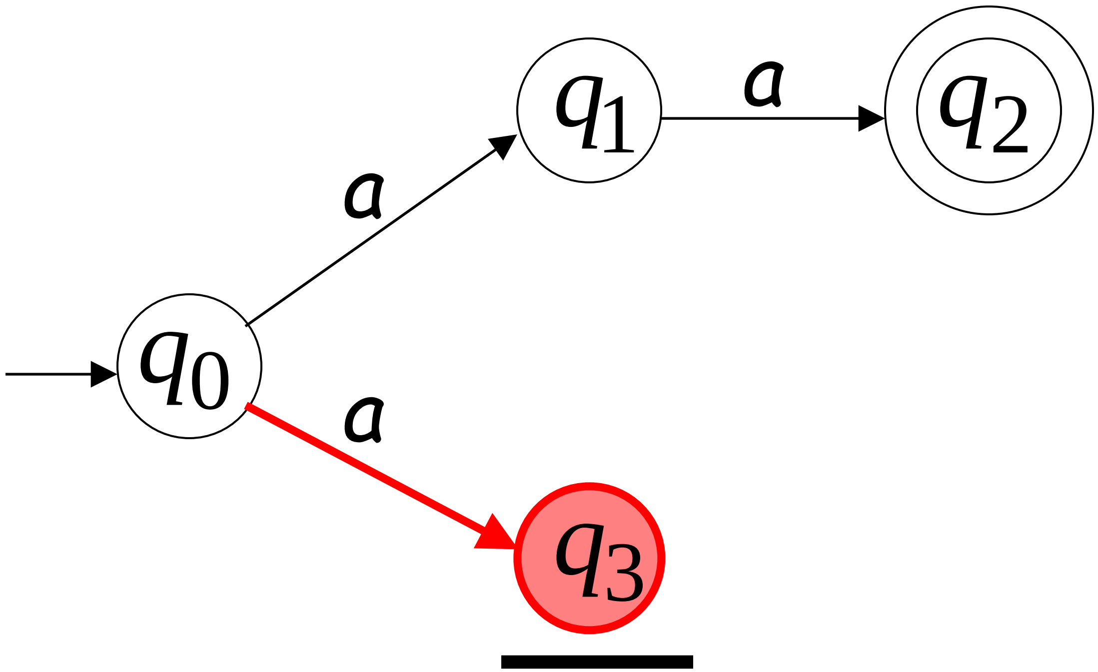
"reject"

Second Choice

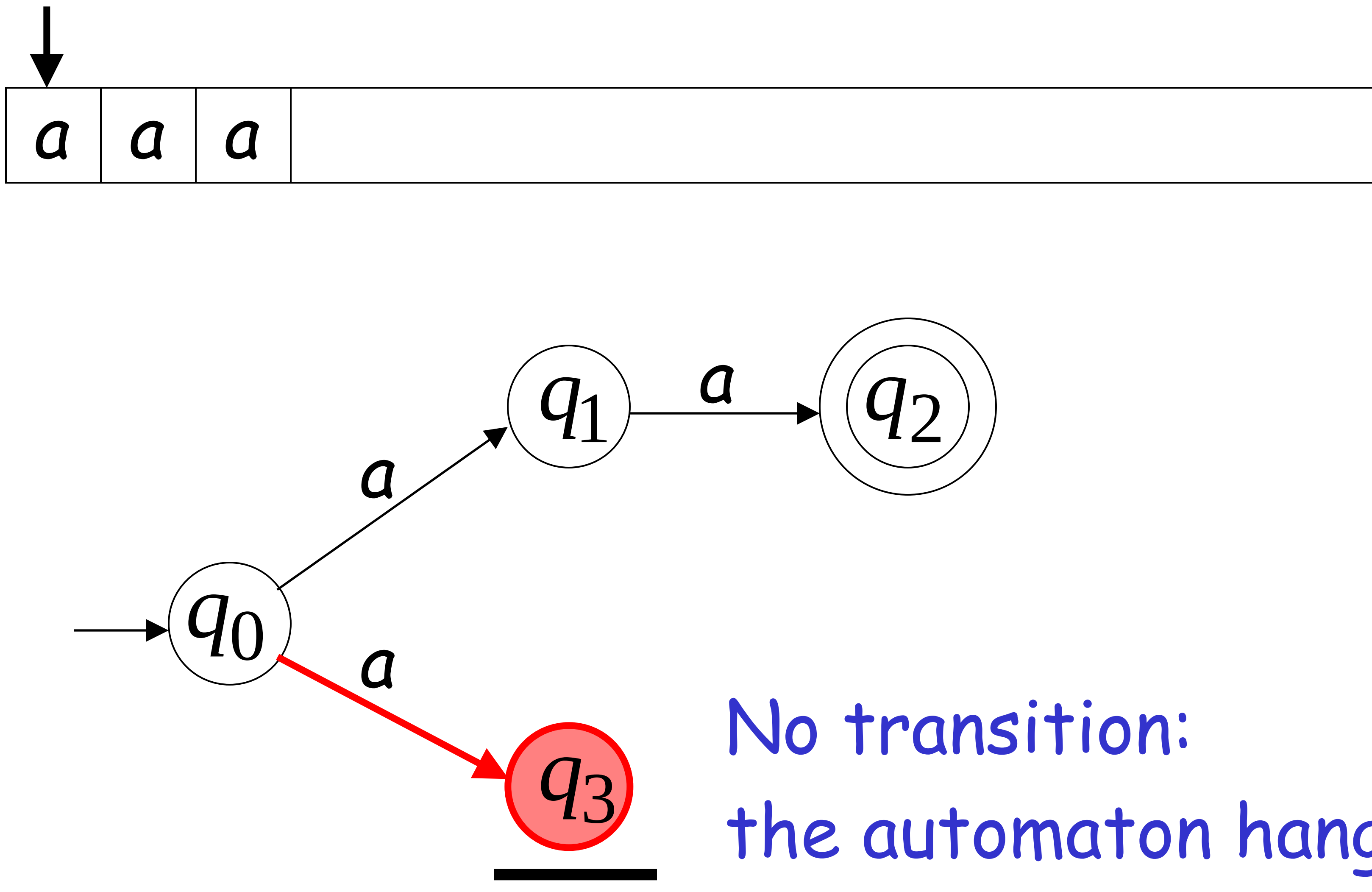


Second Choice



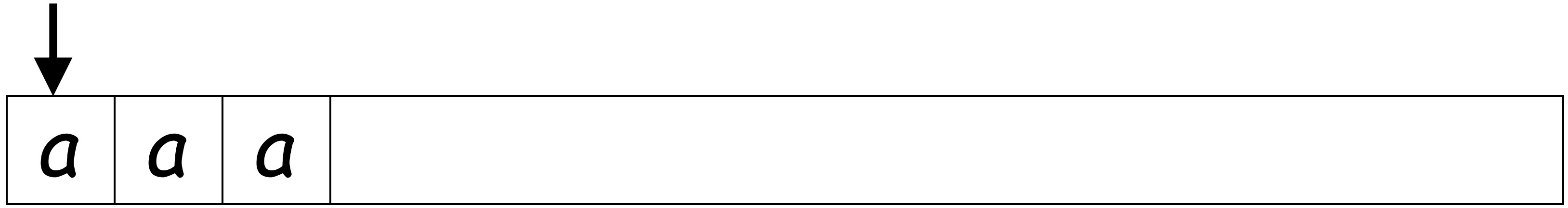


Second Choice

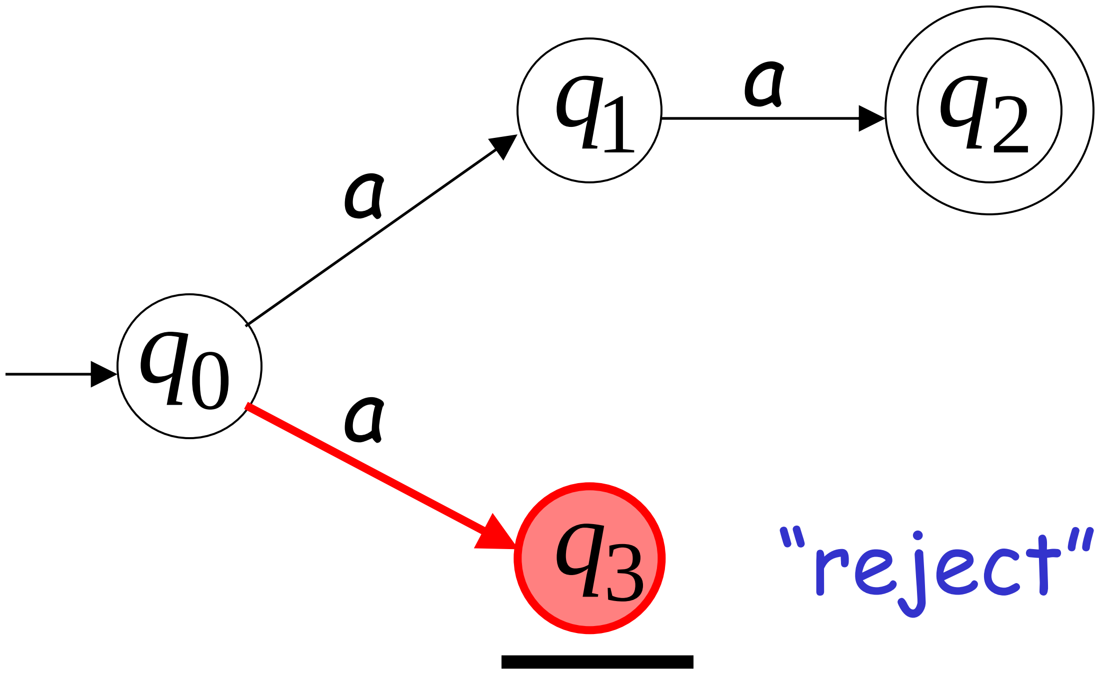


No transition:
the automaton hangs

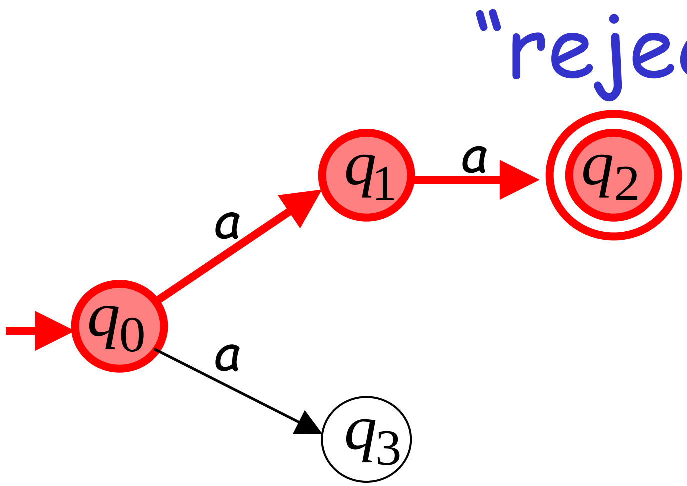
Second Choice

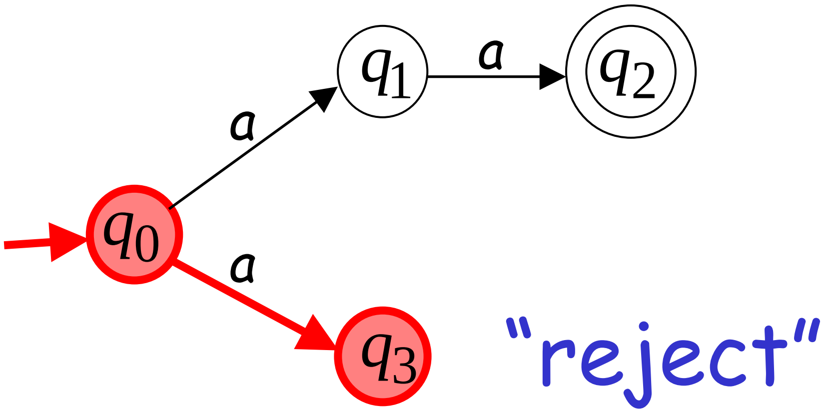


Input cannot be consumed



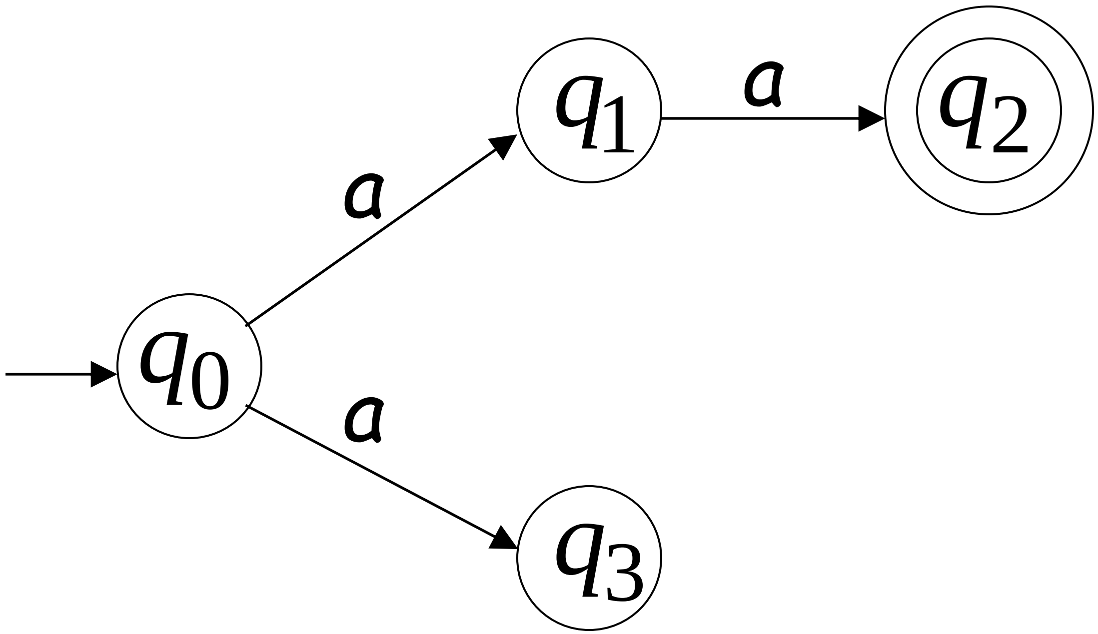
aaa is rejected by the NFA:



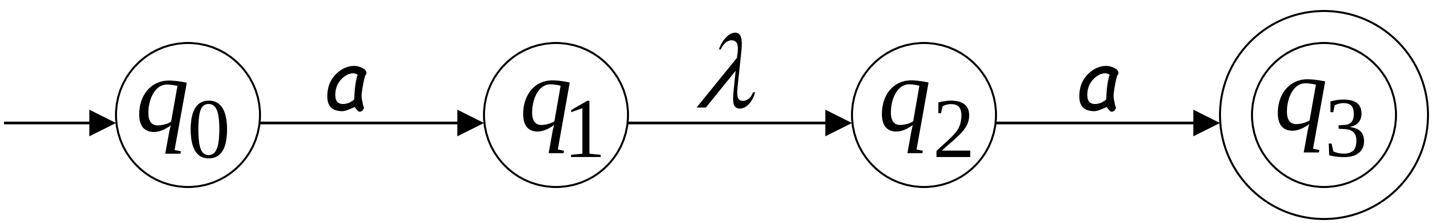


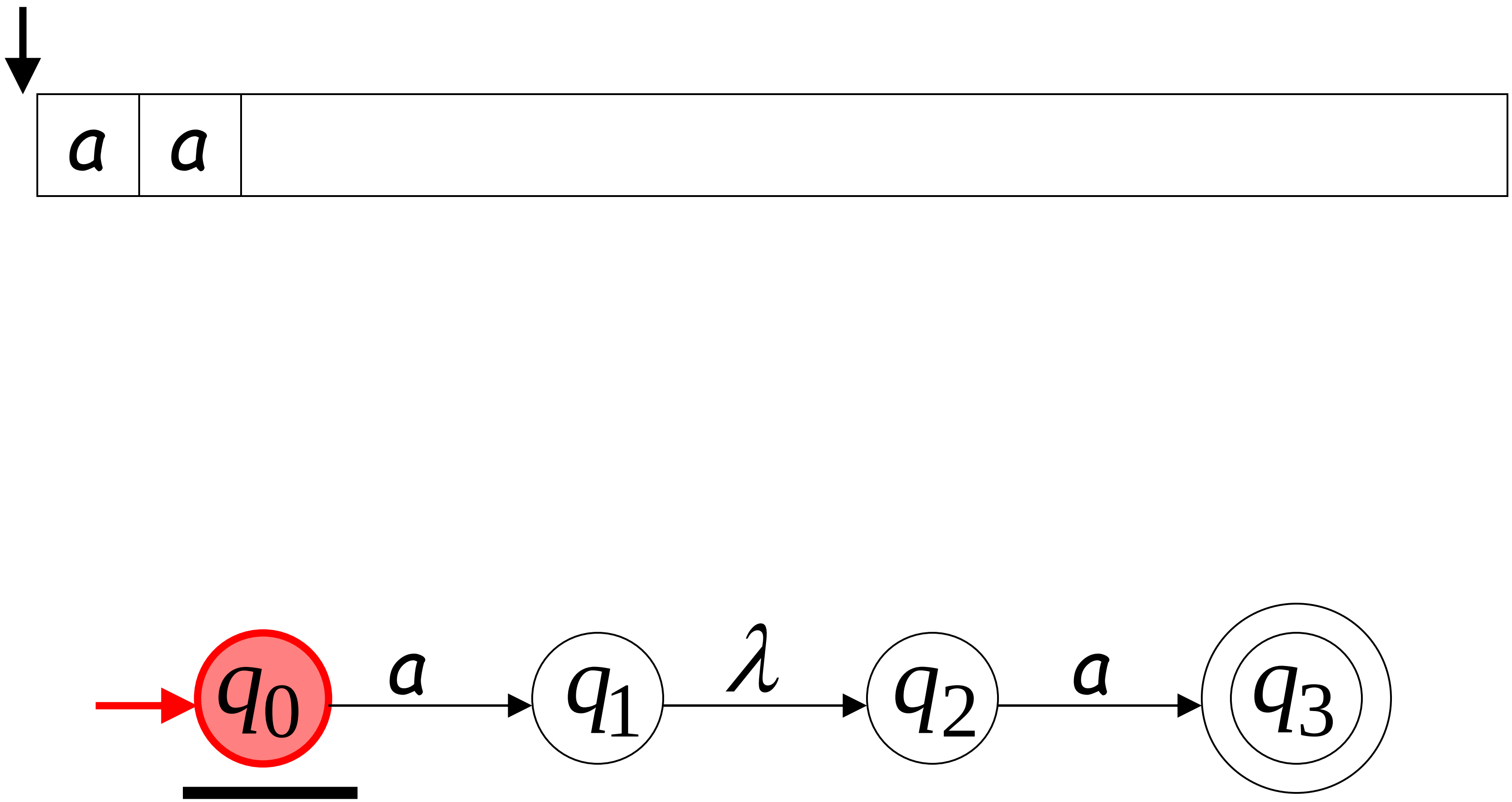
All possible computations lead to rejection

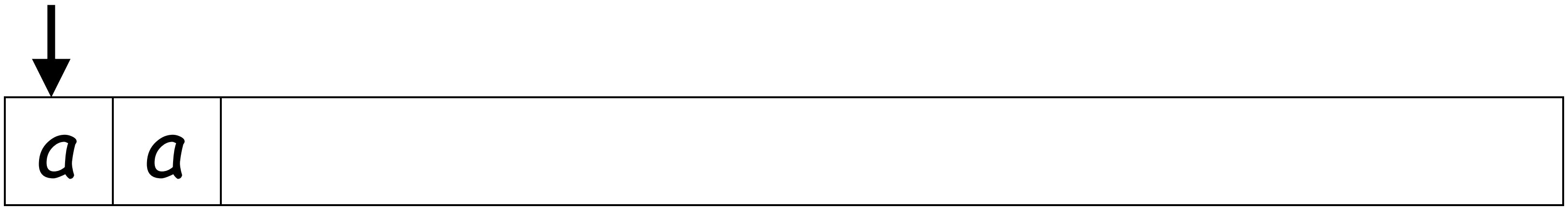
Language accepted: $L = \{aa\}$

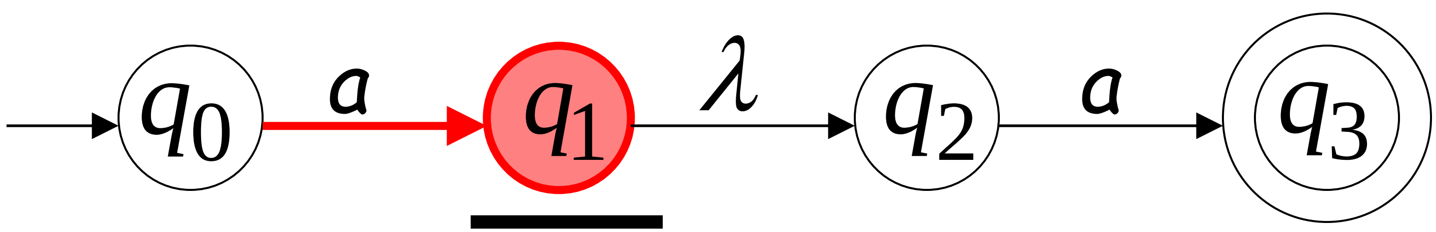


Lambda Transitions

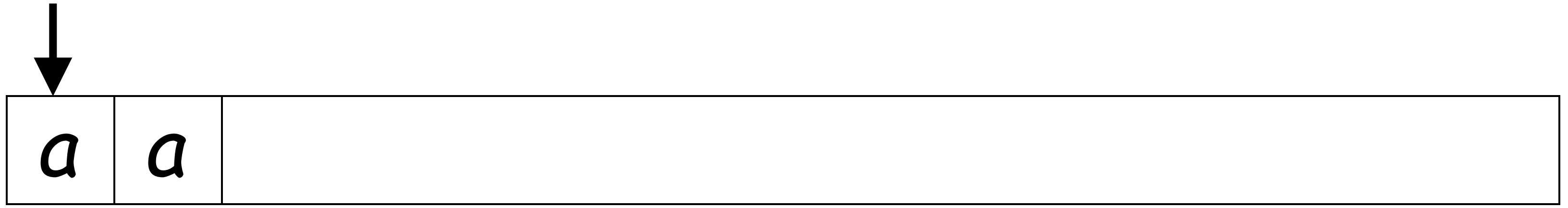


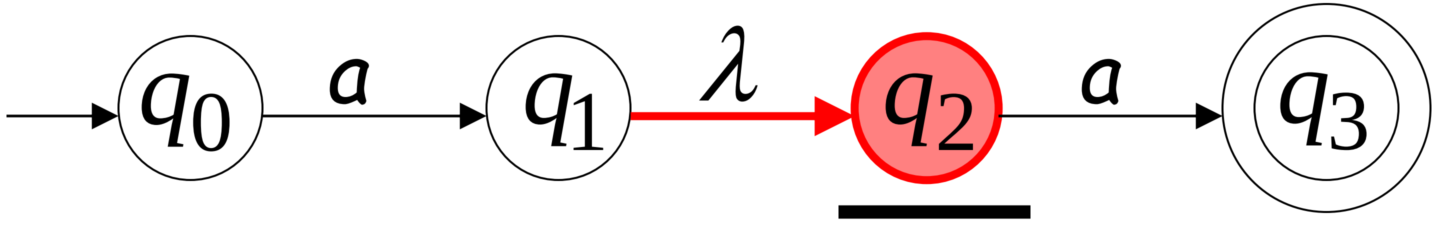

a
a
q0
a
q1
λ
q2
a
q3

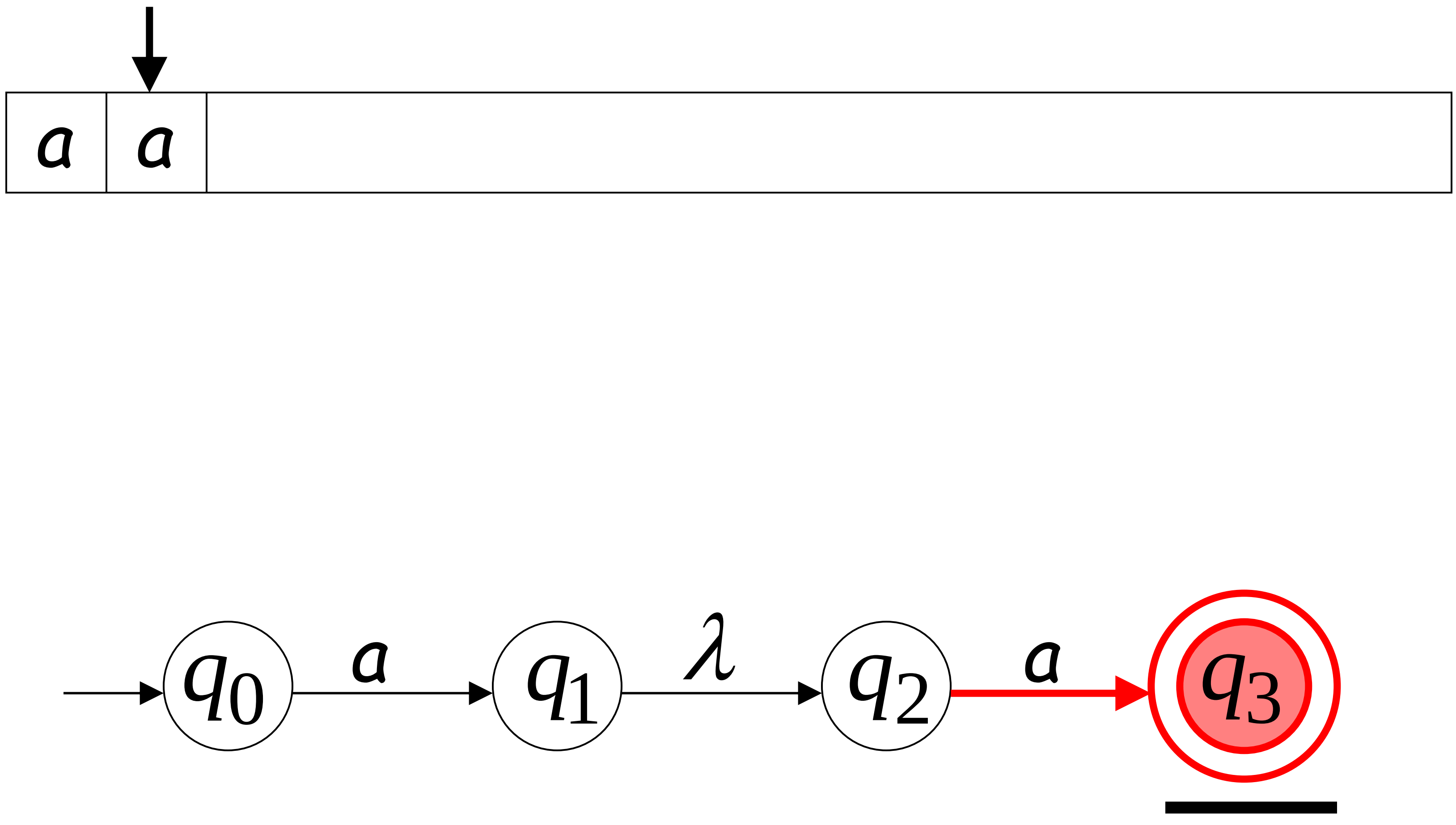

a
a


q0
a
q1
λ
q2
a
q3

(read head does not move)

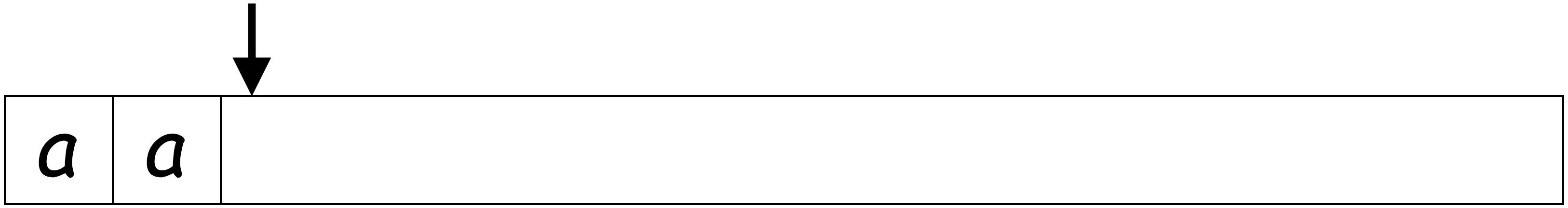


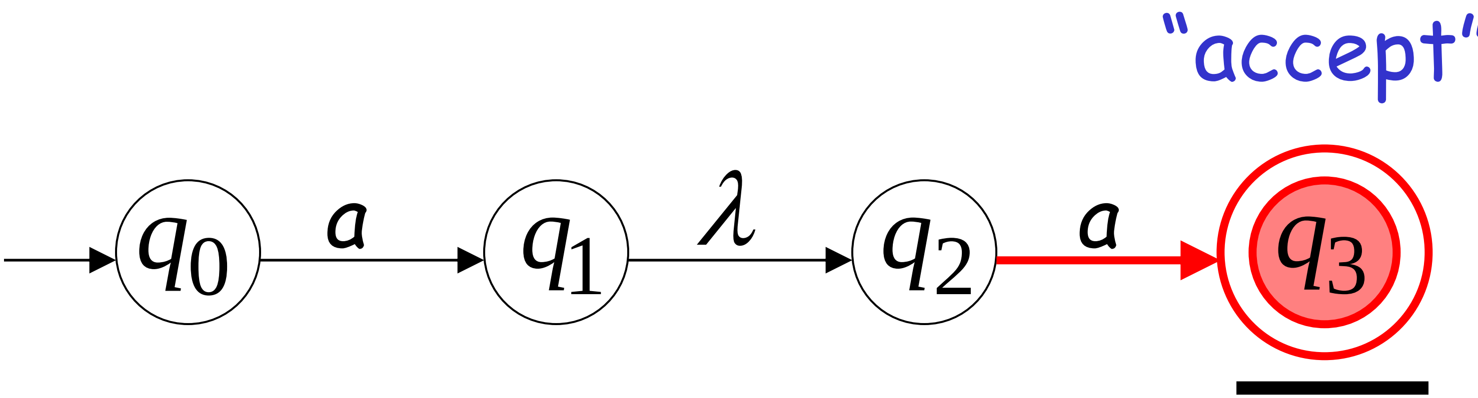




a
a
q_0
a
q_1
λ
q_2
a
q_3

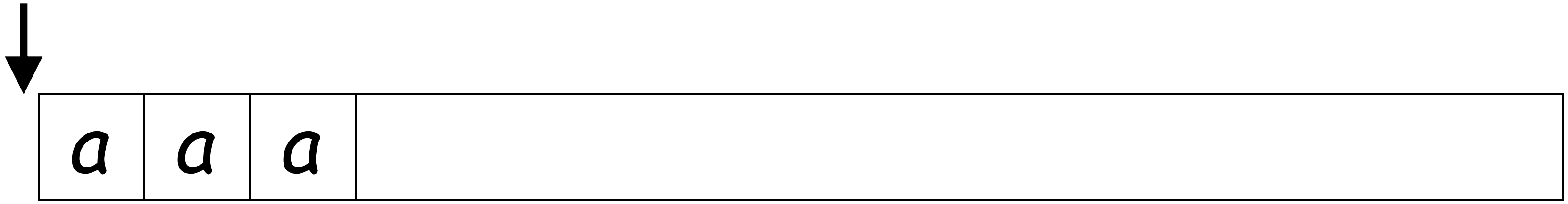
all input is consumed

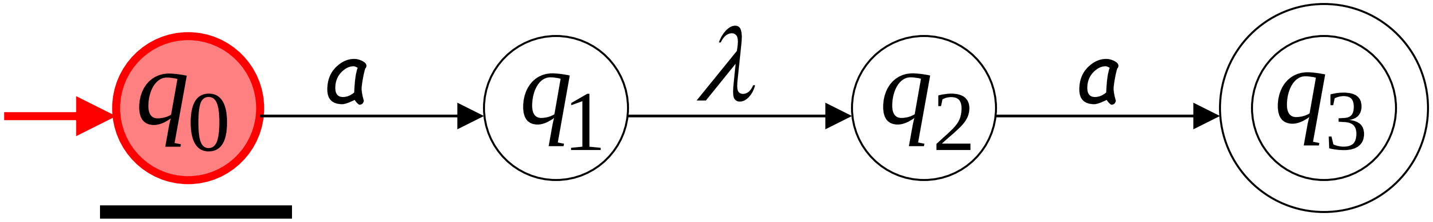


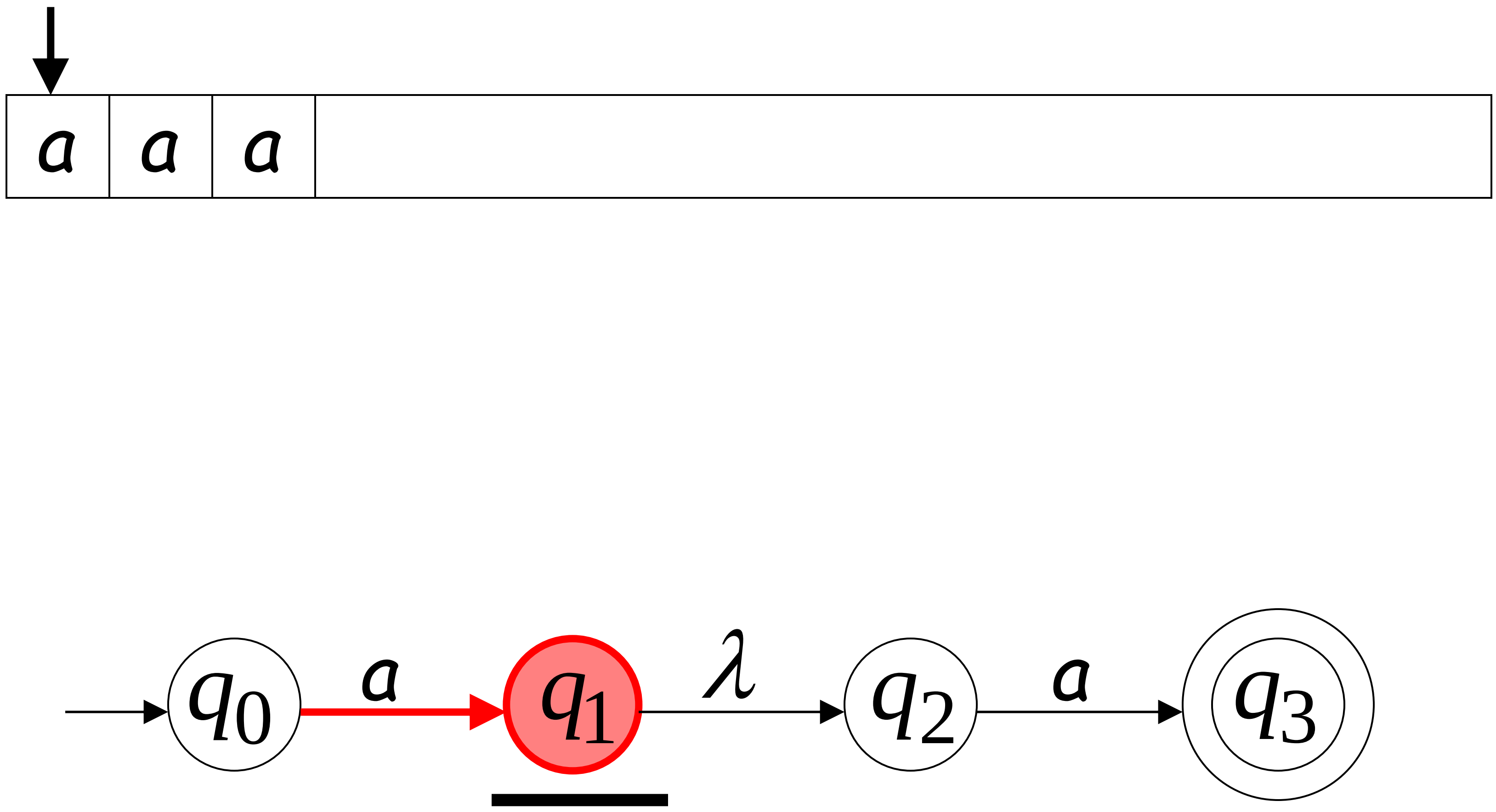


String aa is accepted

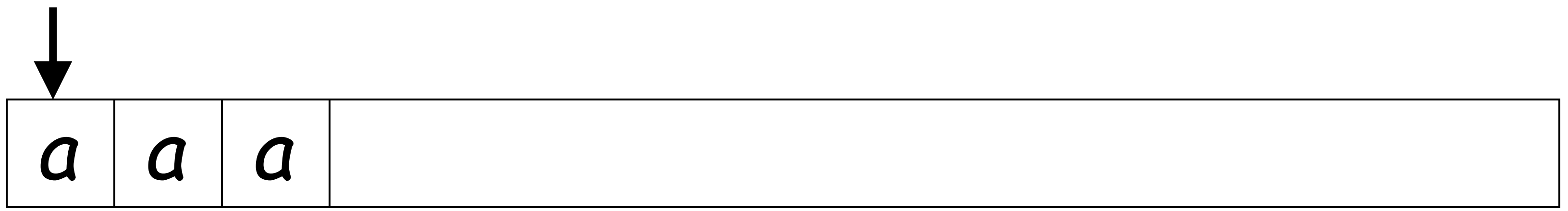
Rejection Example

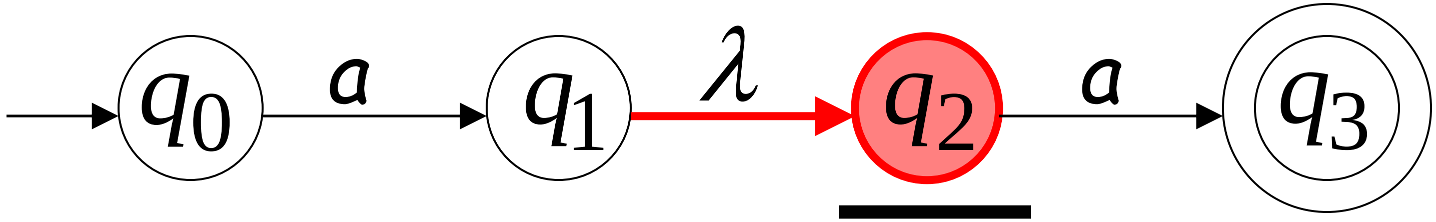


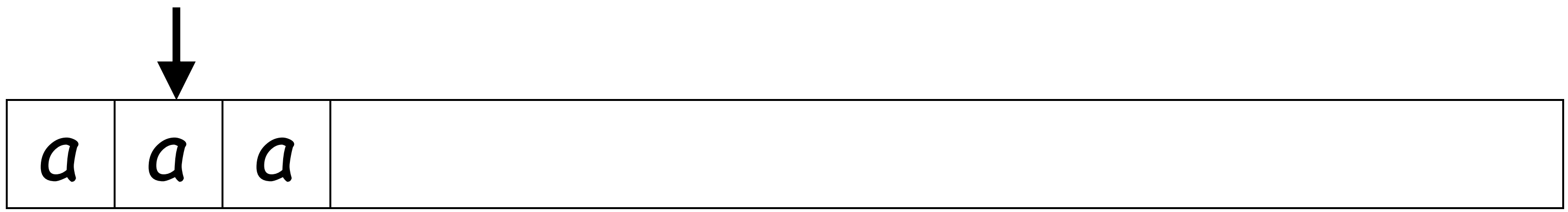


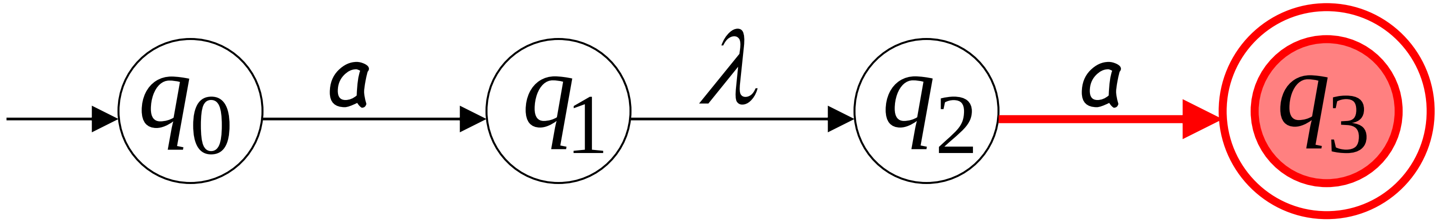

a a a
q0 a q1 λ q2 a q3

(read head doesn't move)





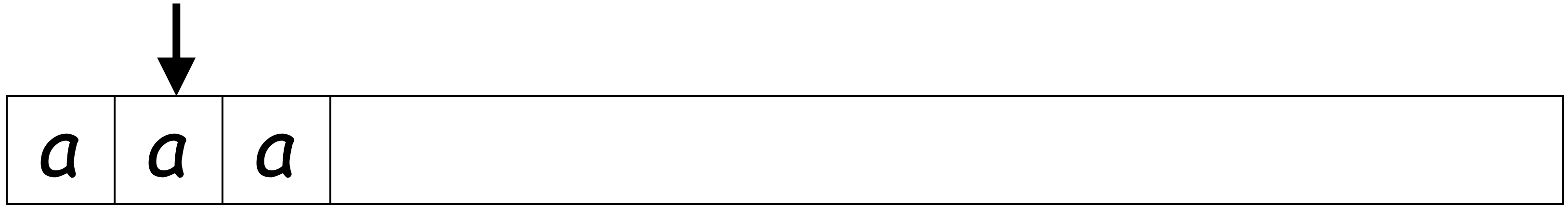


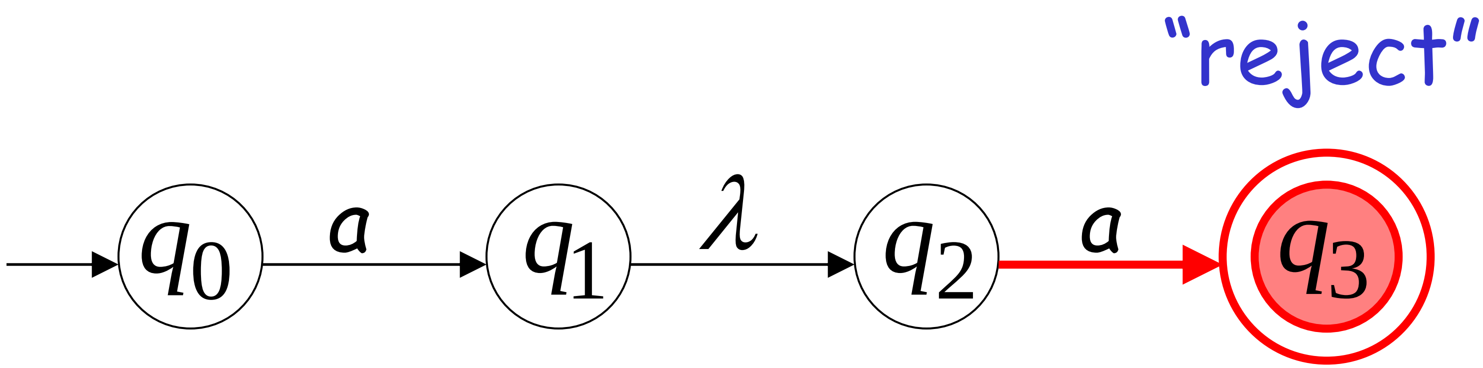


No transition:

the automaton hangs

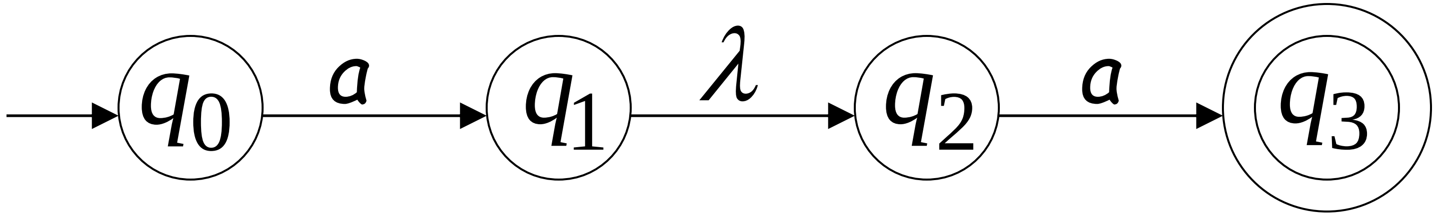
Input cannot be consumed



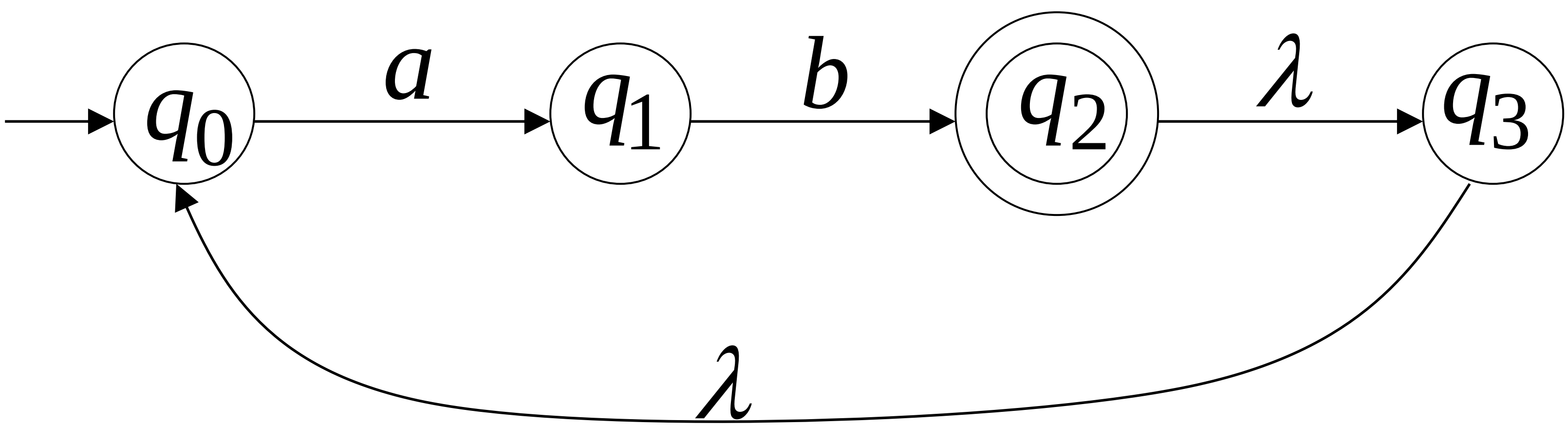


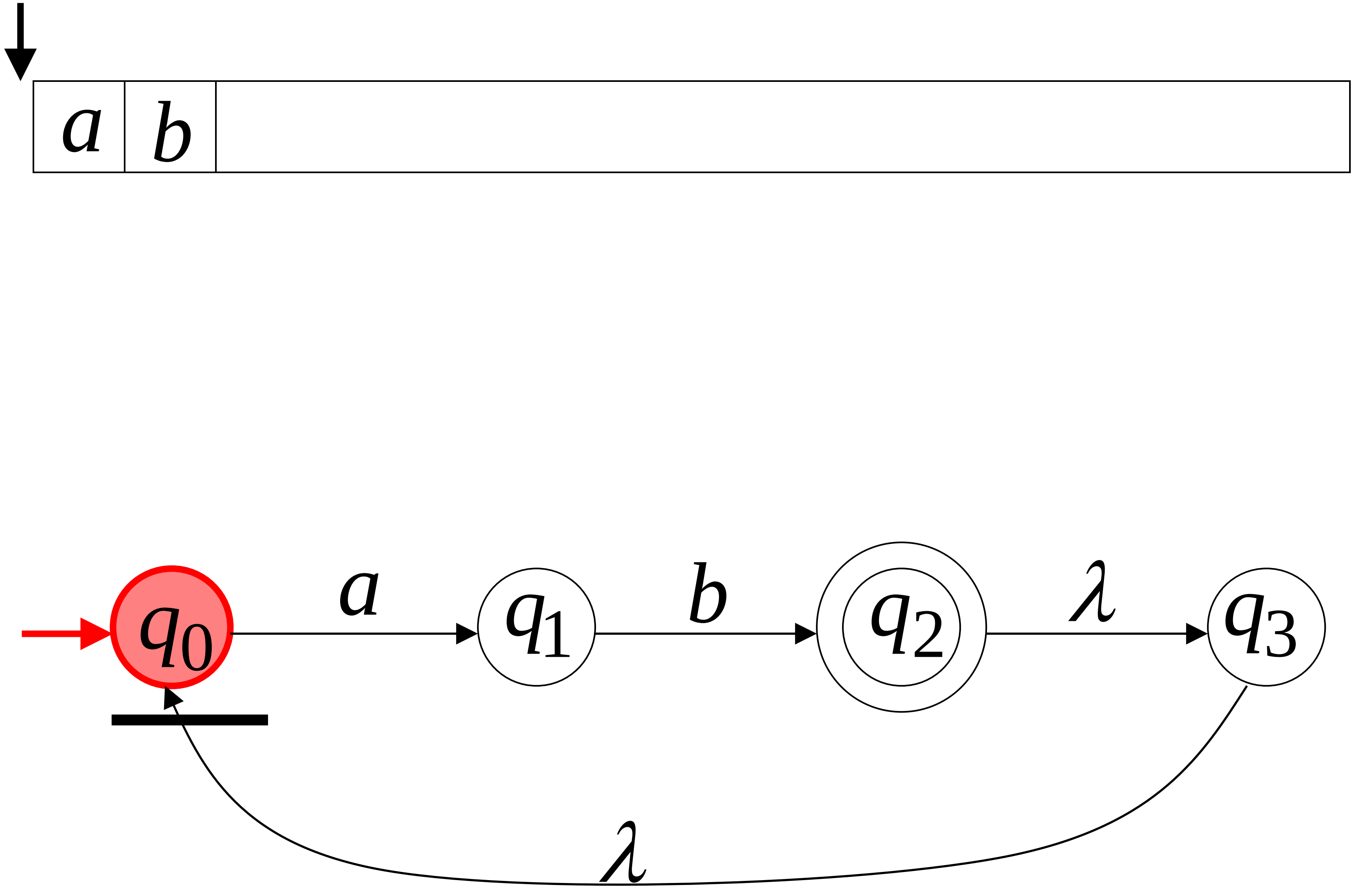
String aaa is rejected

Language accepted: $L = \{aa\}$

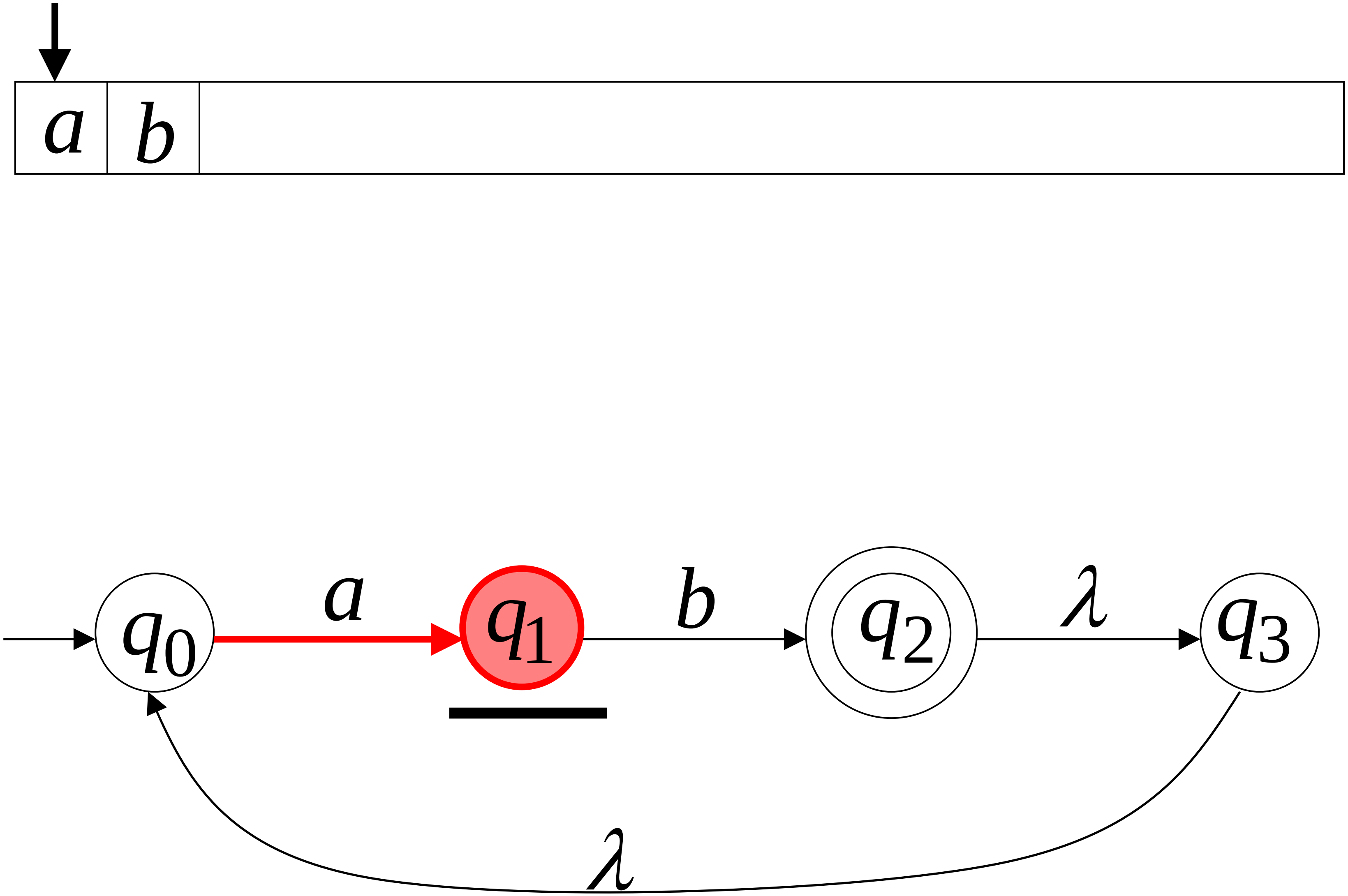


Another NFA Example

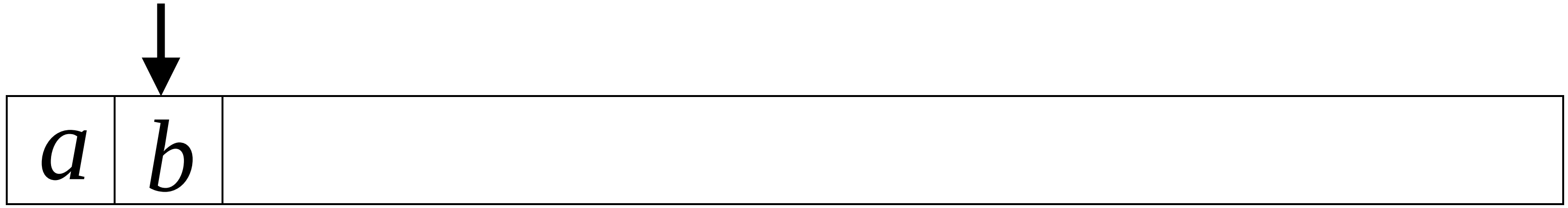


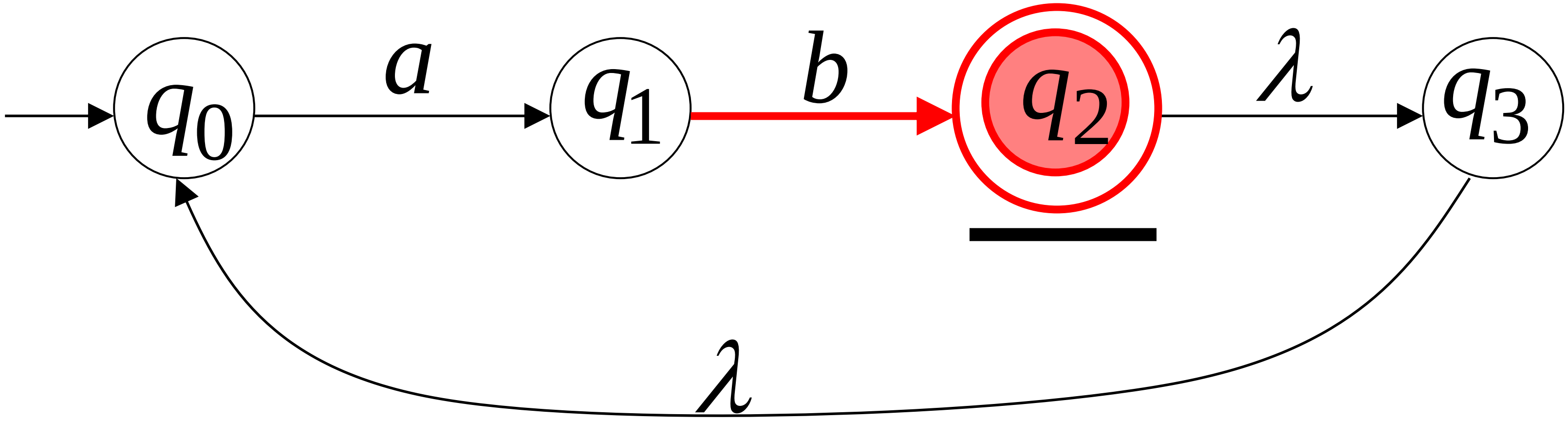


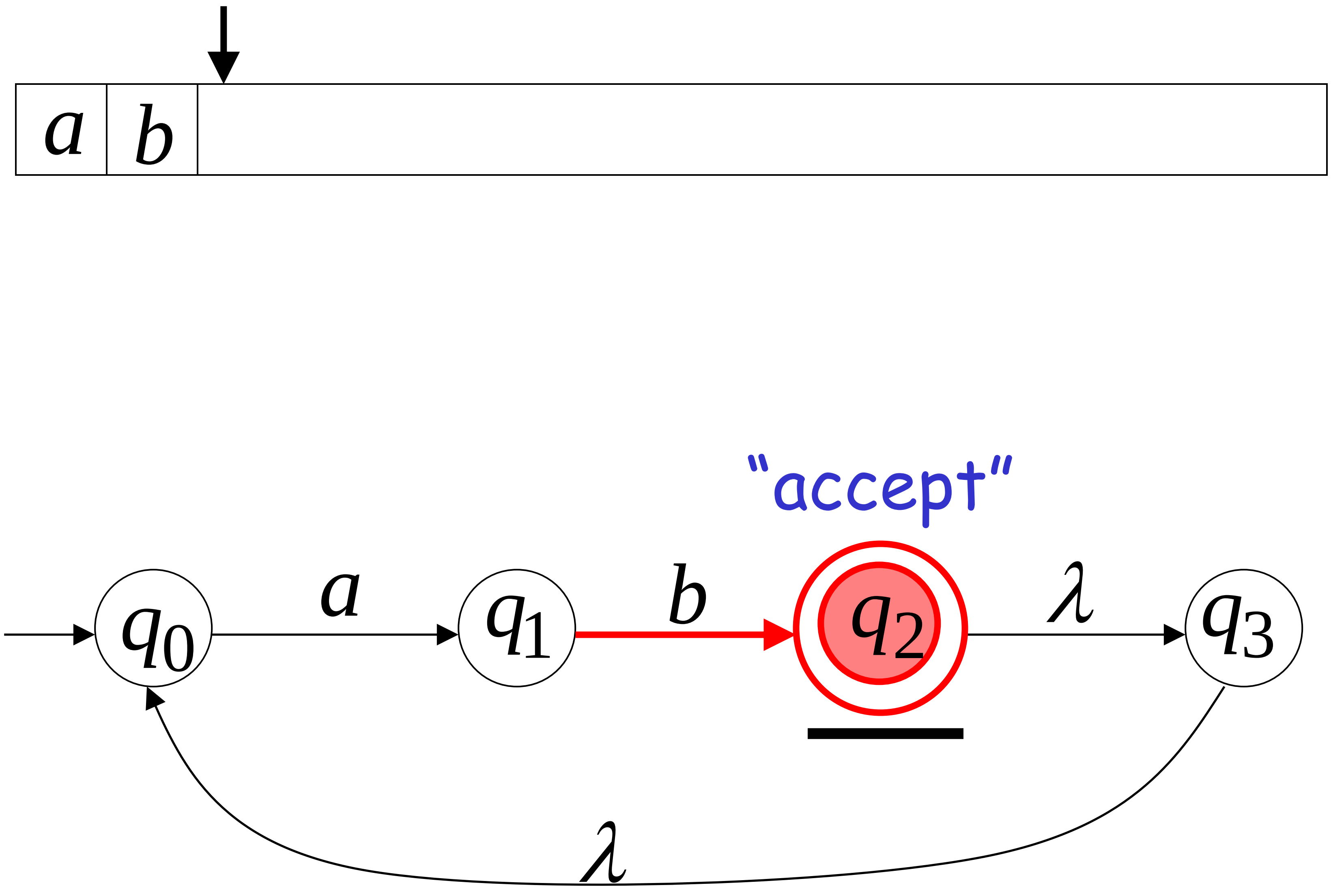
a
b
a
q0
q1
b
q2
λ
q3
λ



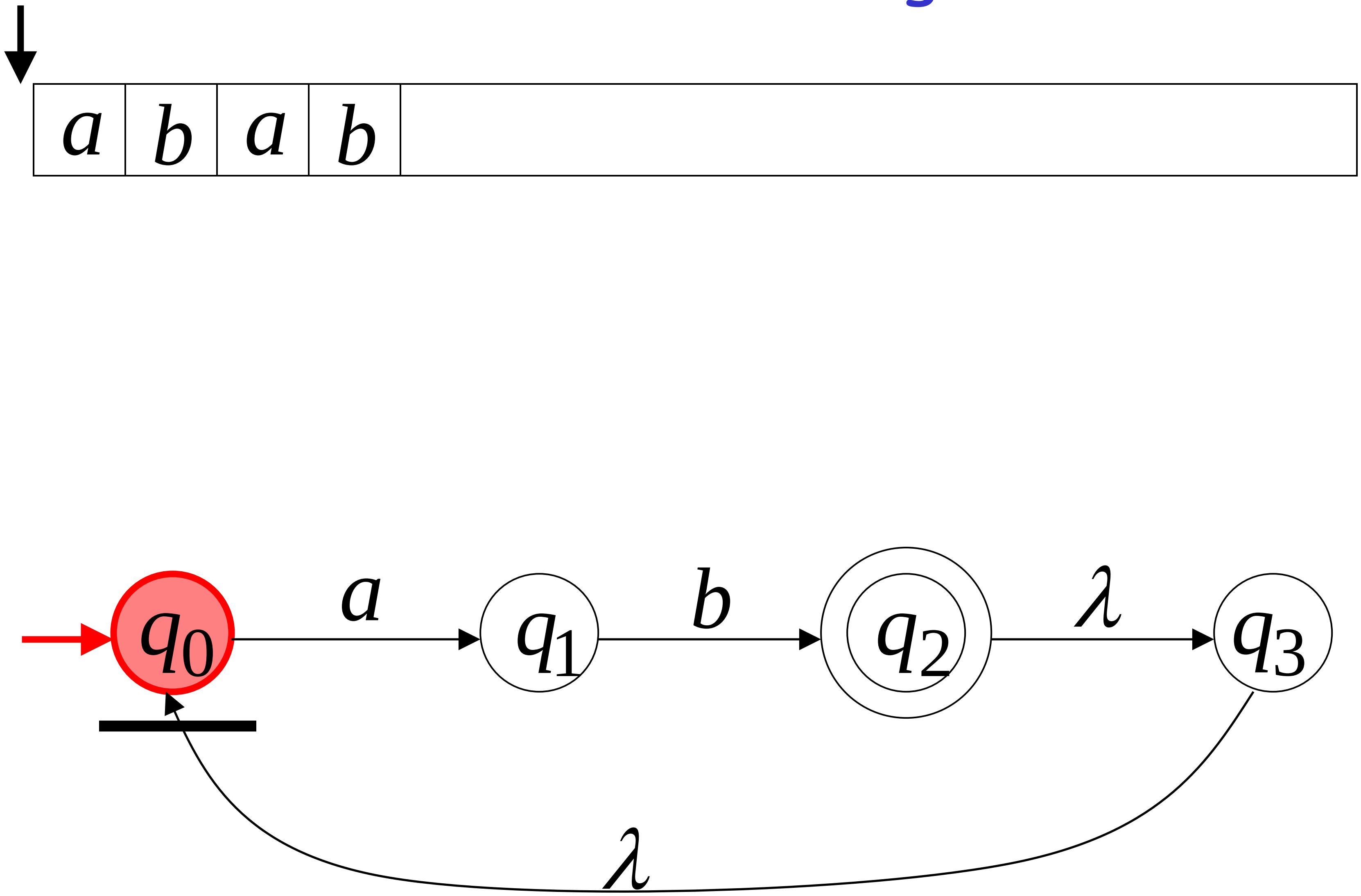
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

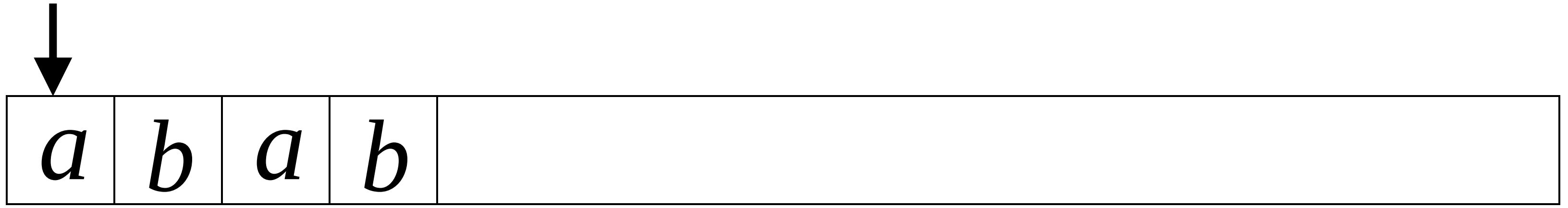

a
b

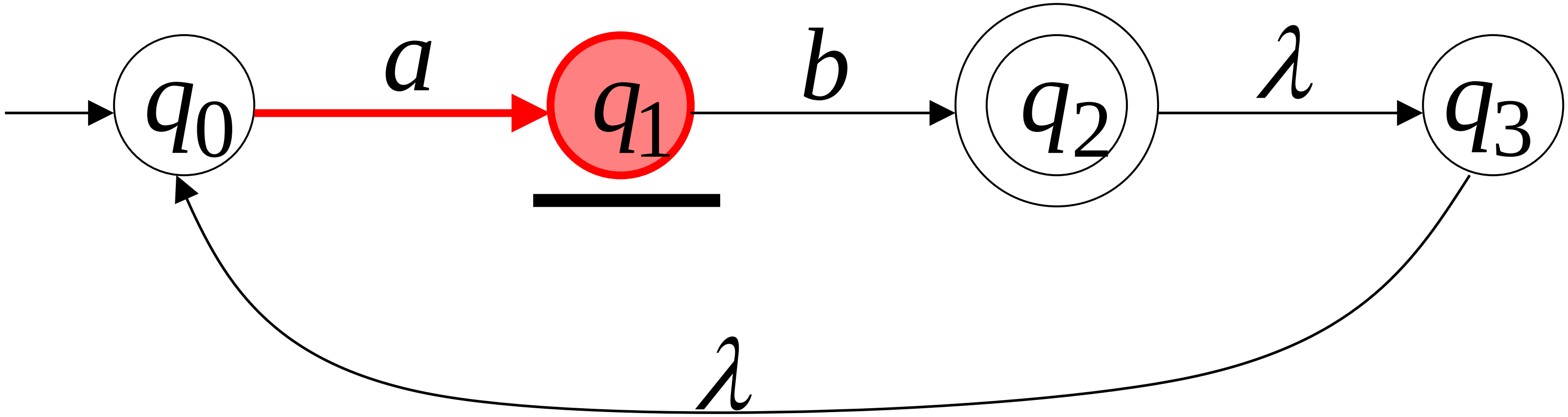

q_0
a
q_1
b
q_2
λ
q_3
λ

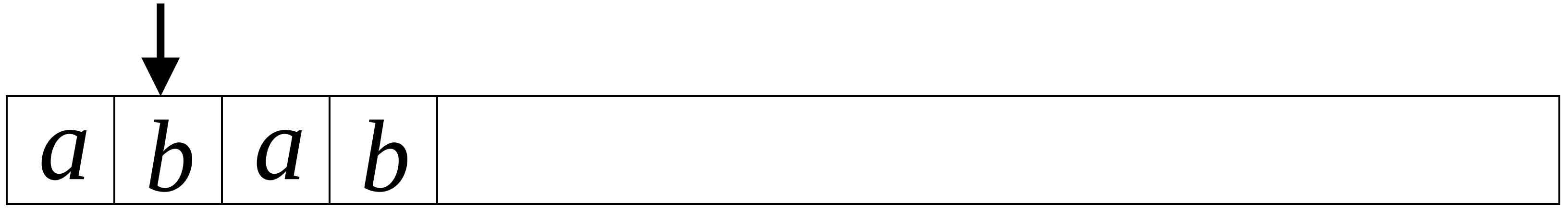

a
b
"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

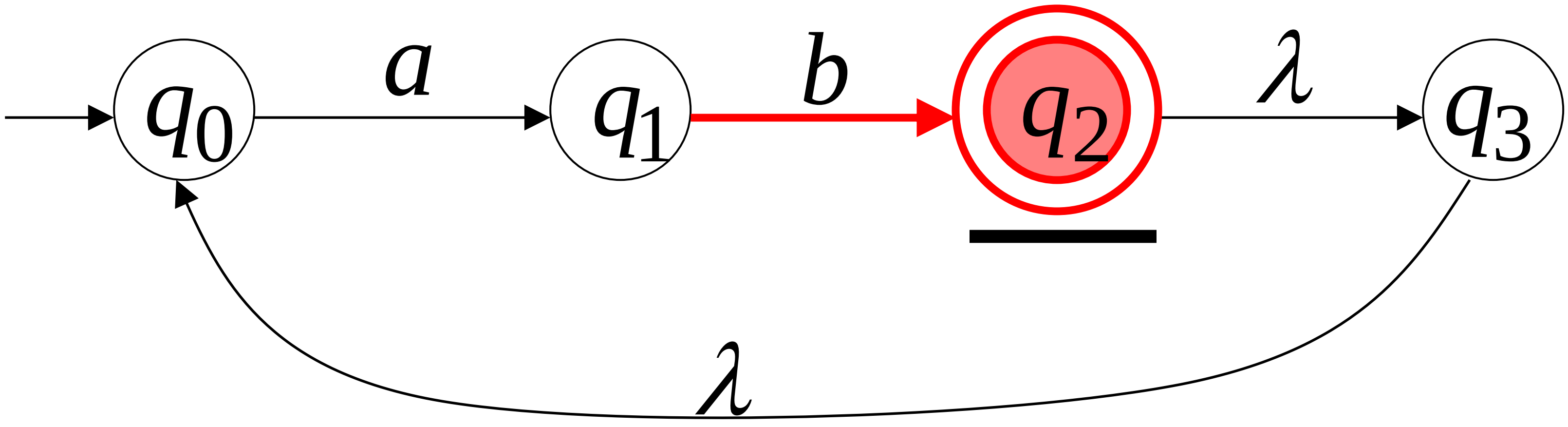
Another String

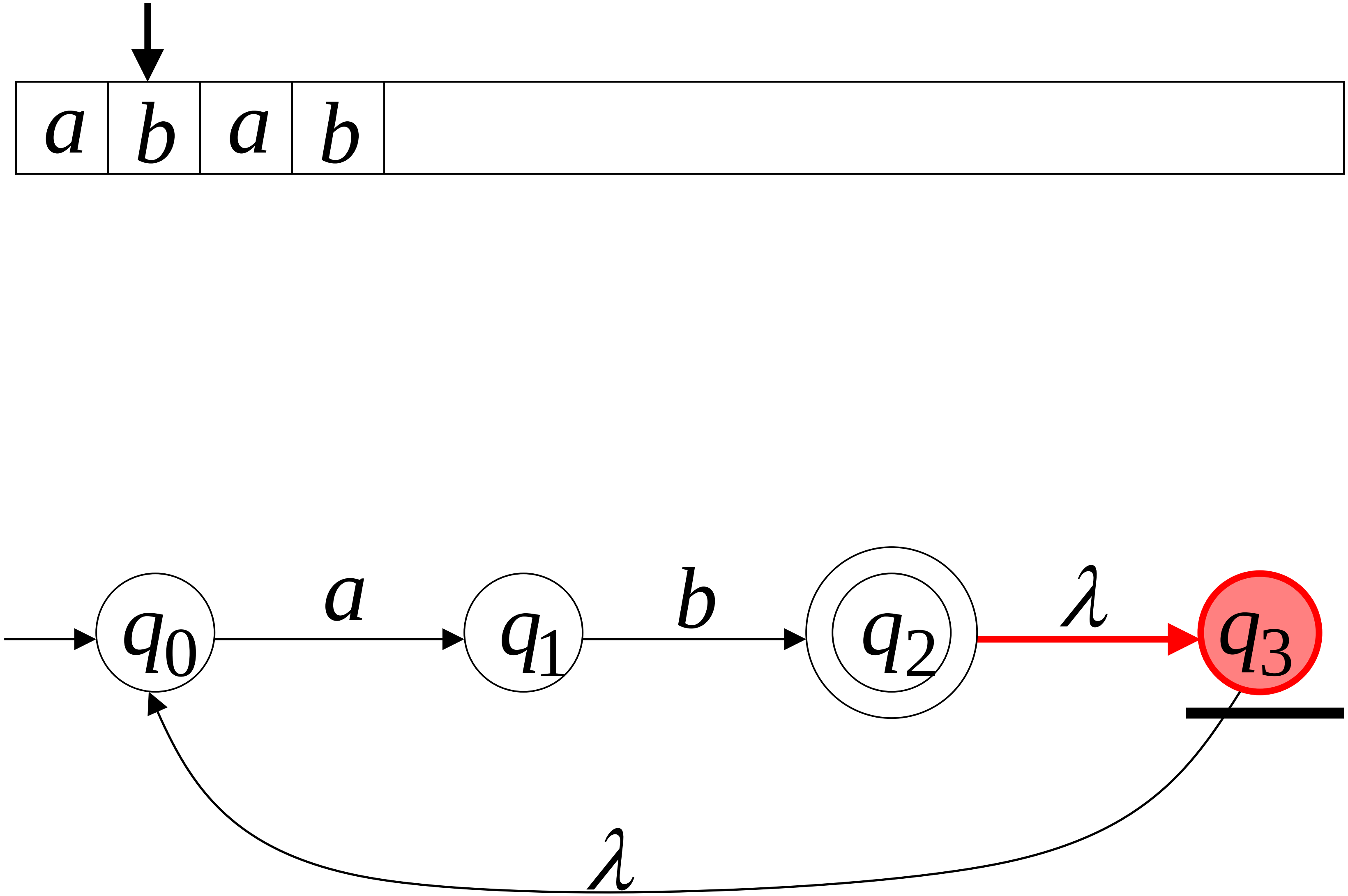


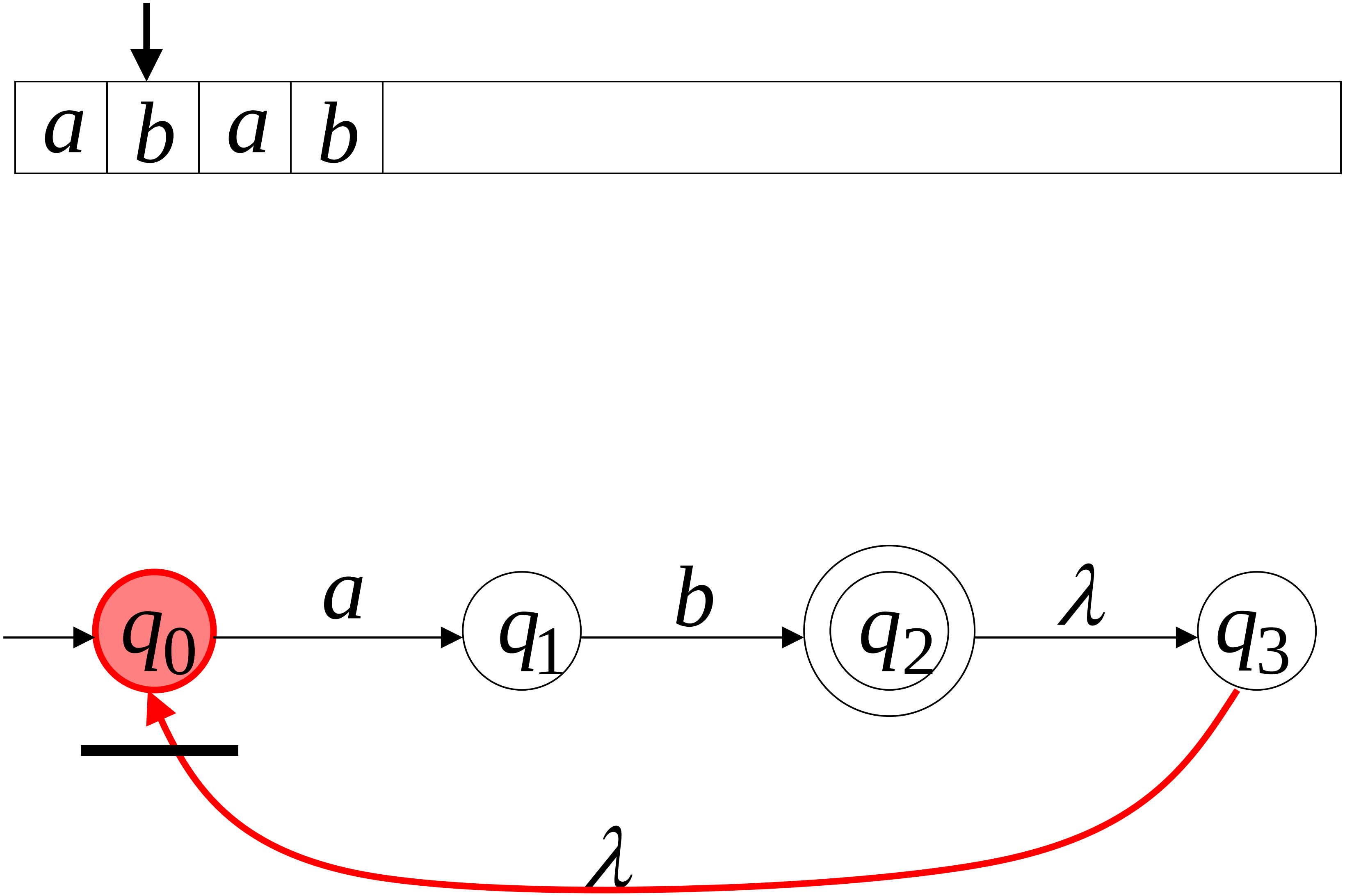

a
b
a
b

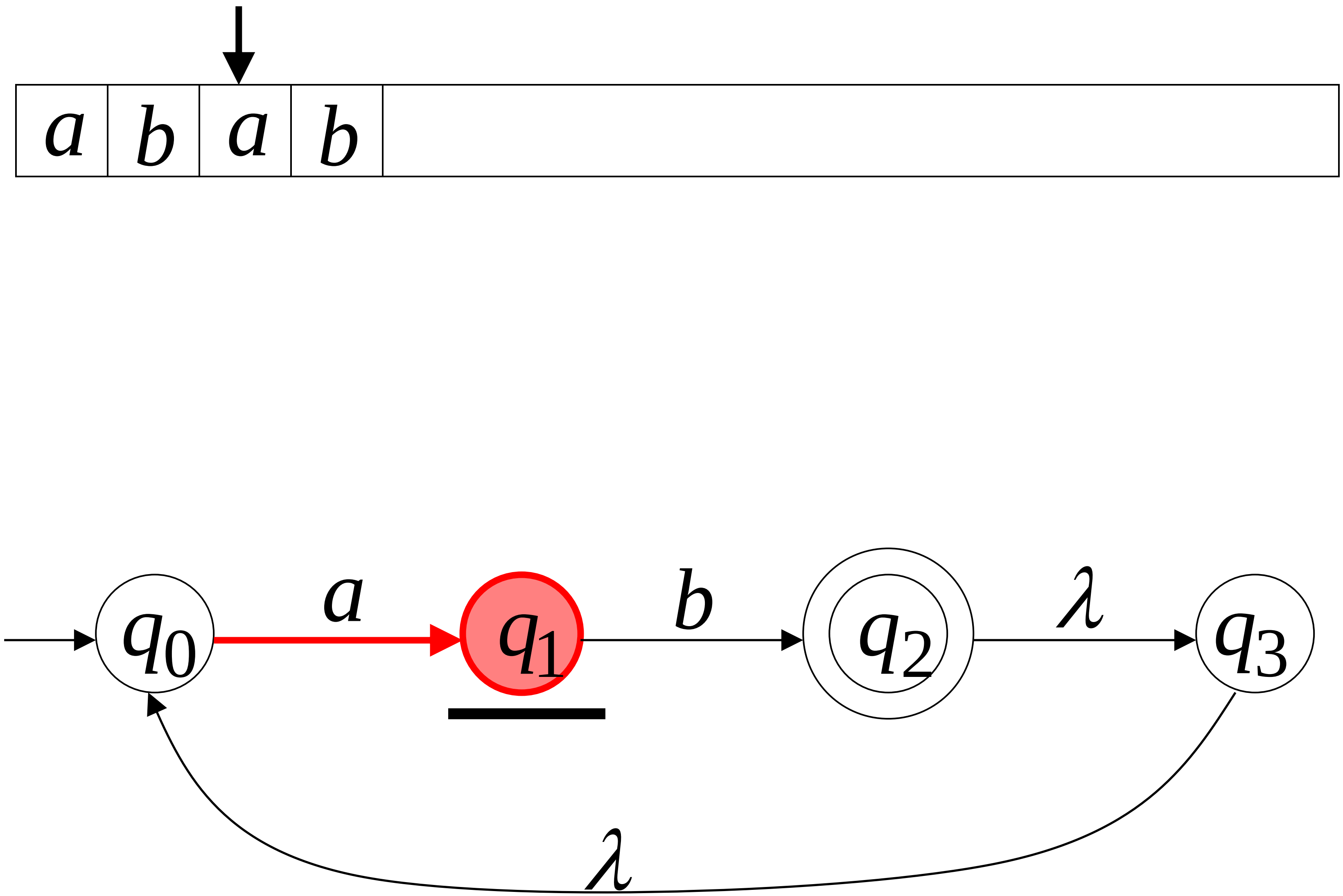

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b

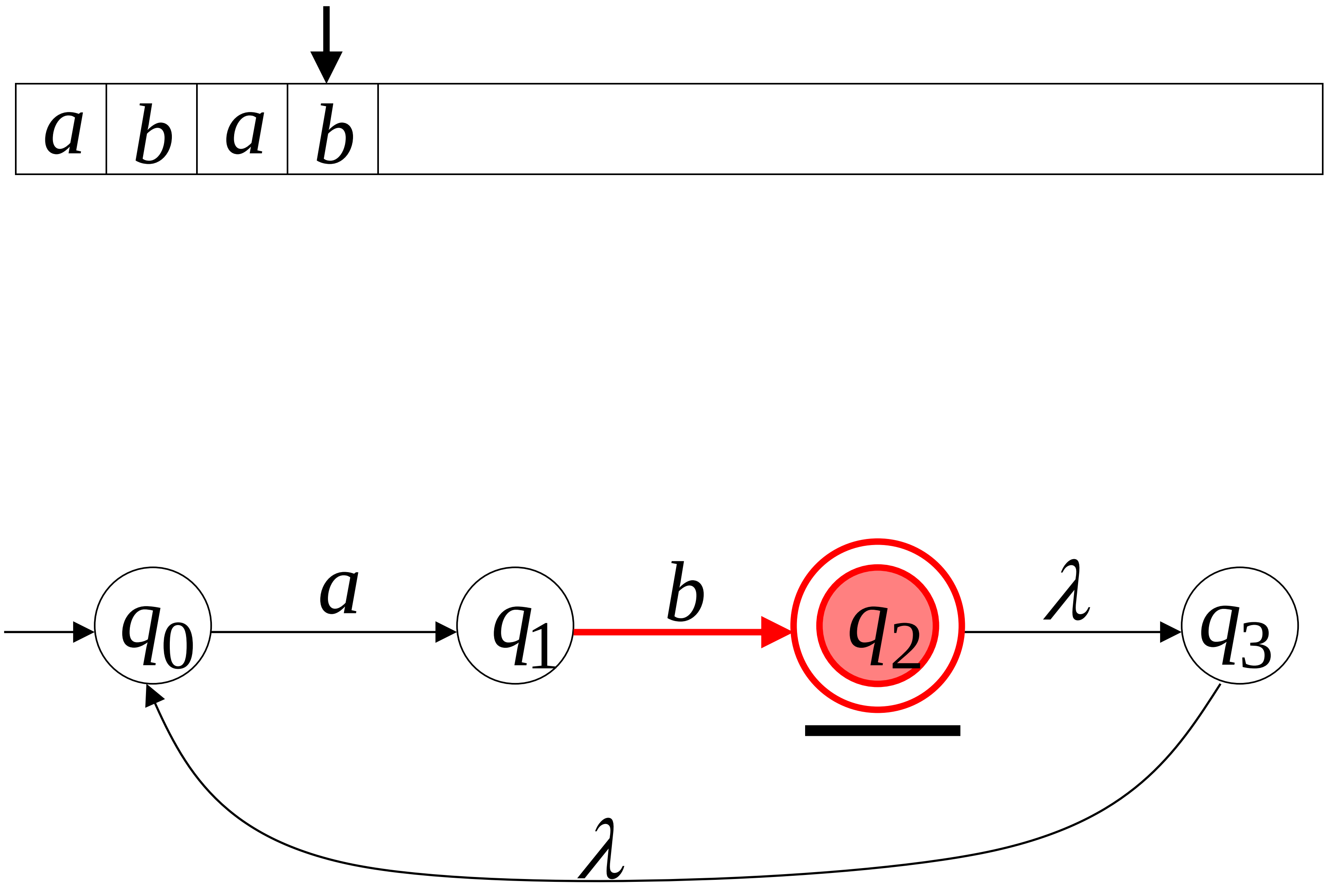

q0
a
q1
b
q2
λ
q3
λ

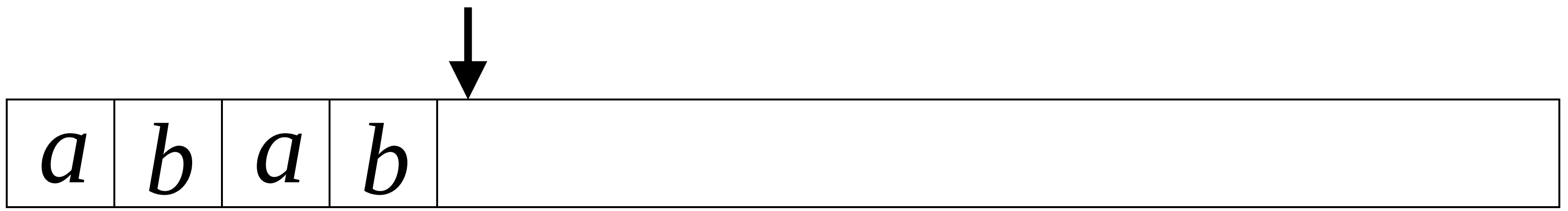

a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3


a
b
a
b
a
b
λ
q0
q1
q2
q3
λ

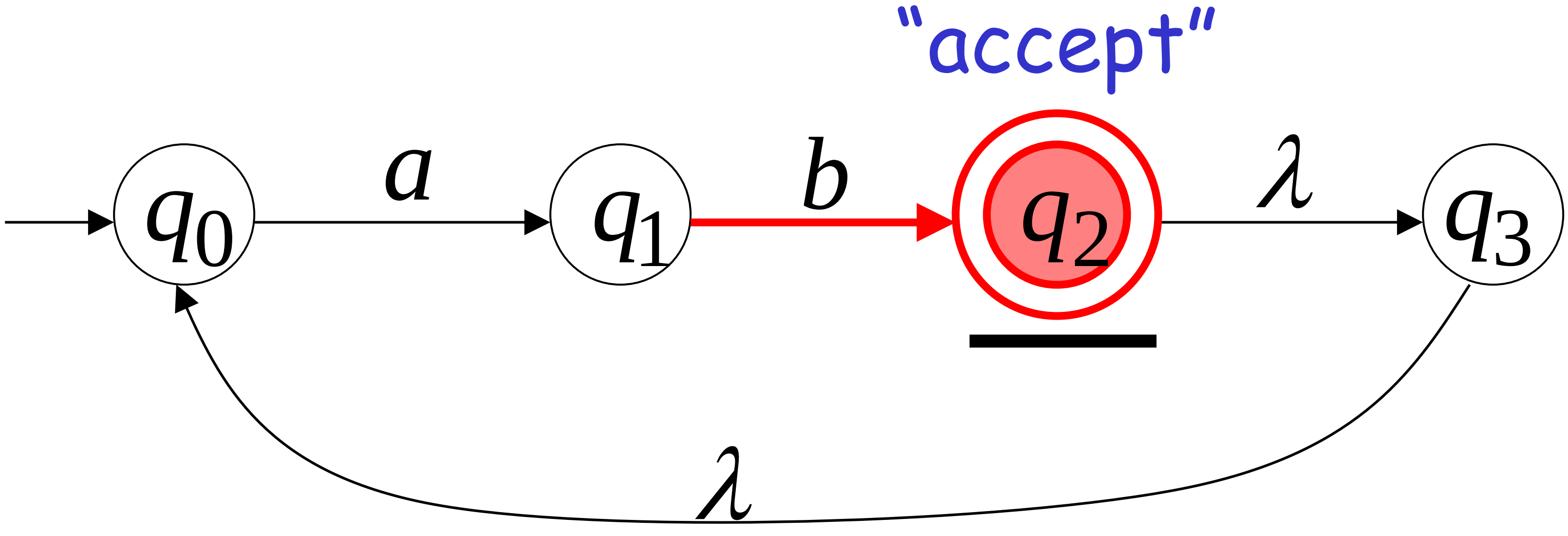


a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3


a
b
a
b
a
b
λ
λ
q0
q1
q2
q3



a
b
a
b

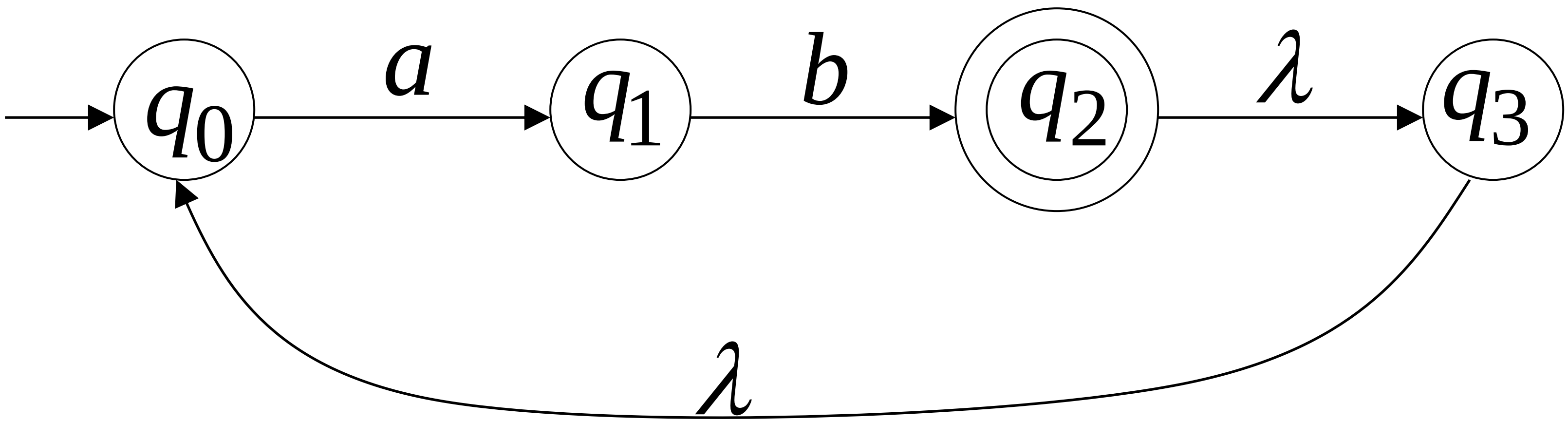


"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

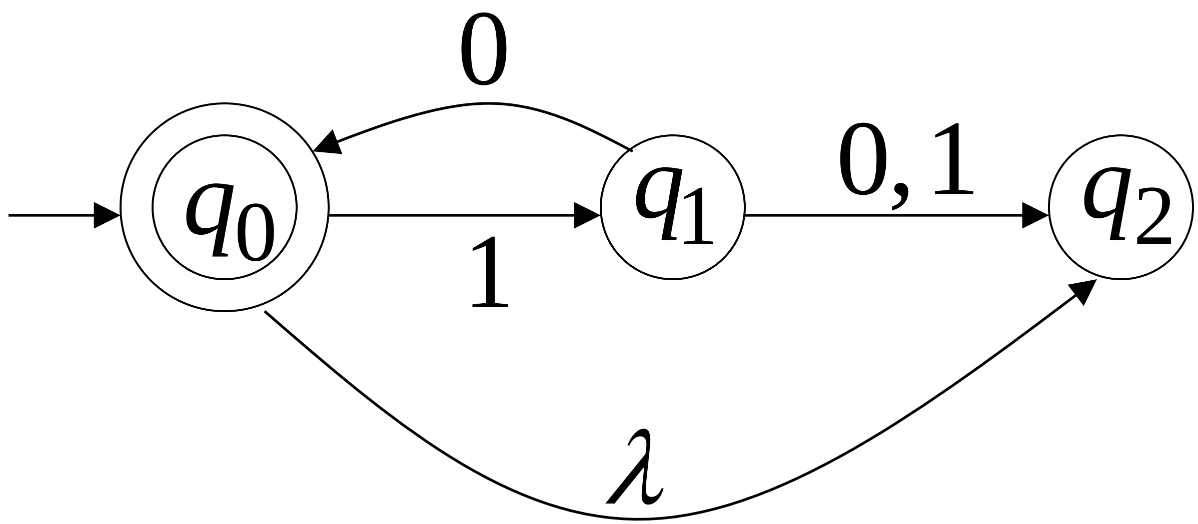
Language accepted

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

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



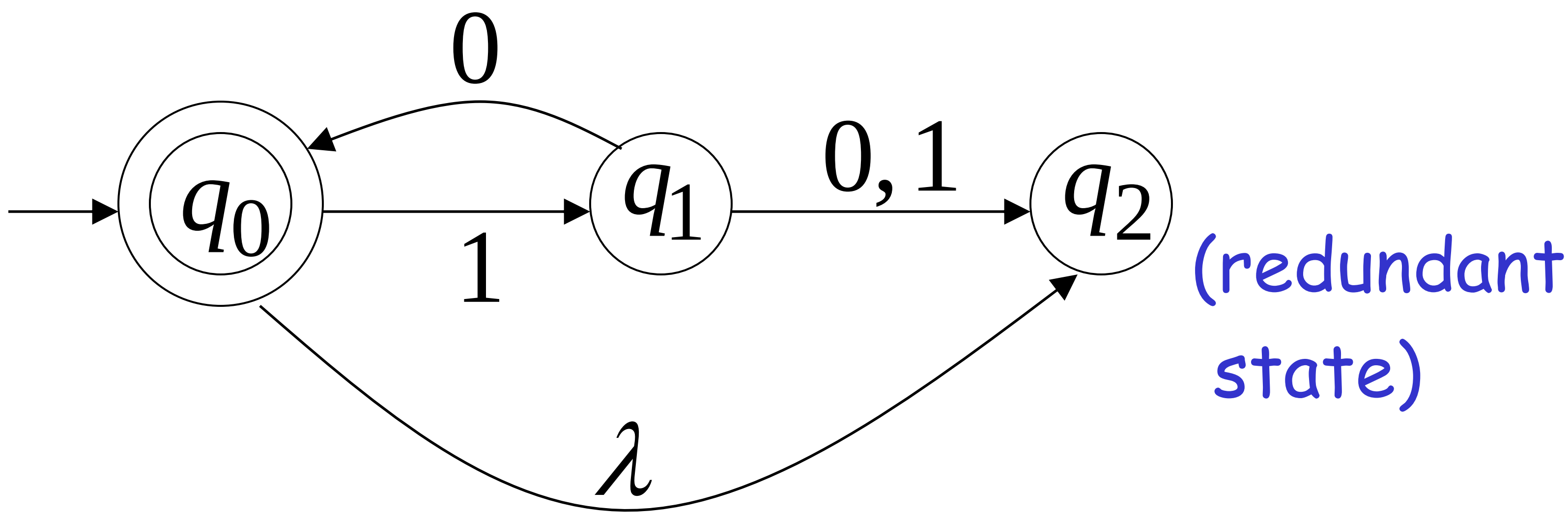
Another NFA Example



Language accepted

$$L(M) = \{\lambda,\ 10,\ 1010,\ 101010,\ \ldots\}$$

$$= \{10\}^*$$

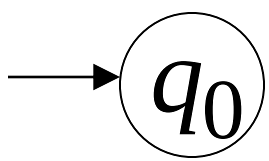


Remarks:

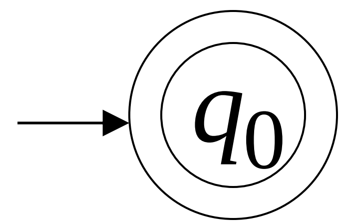
- The λ symbol never appears on the input tape

- Simple automata:

M_1



M_2

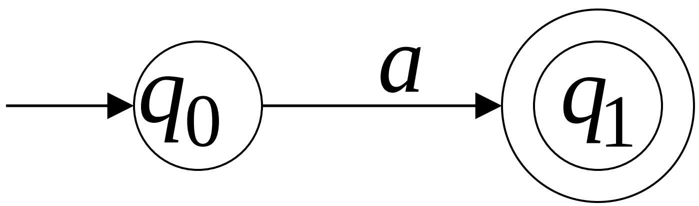


$L(M_1) = \{\}$

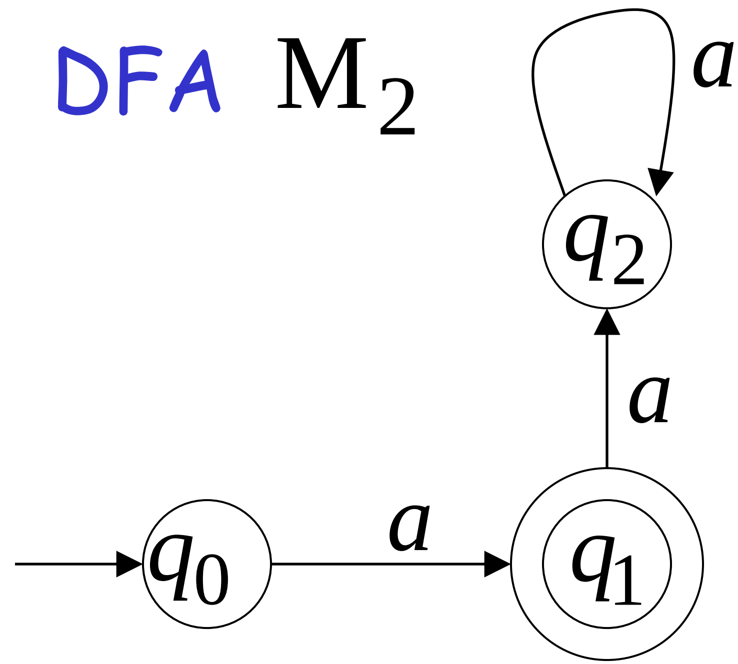
$L(M_2) = \{\lambda\}$

- 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. $\{a, b\}$

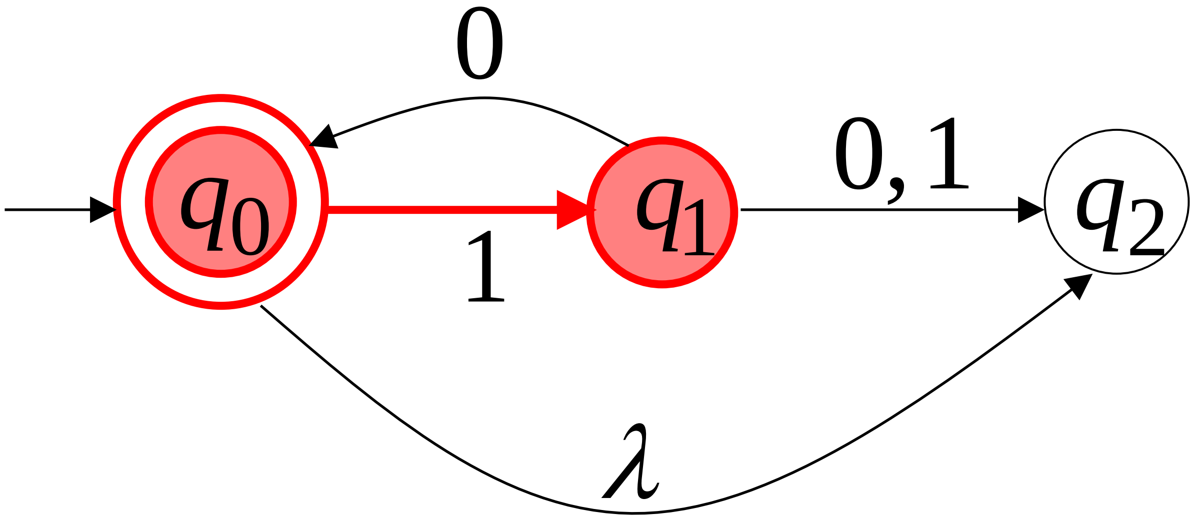
$\delta:$ Transition function

$q_0:$ Initial state

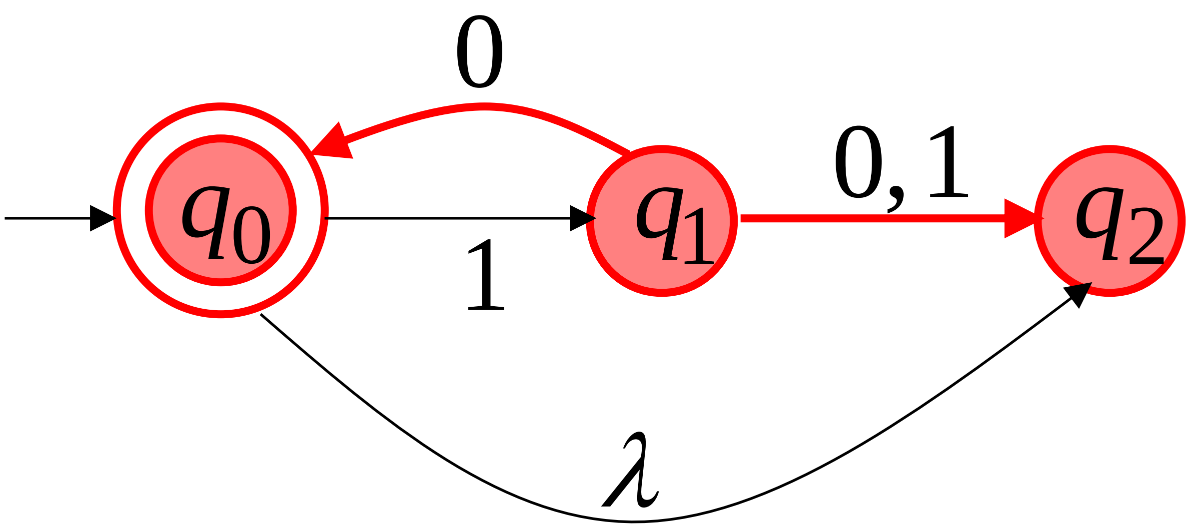
$F:$ Final states

Transition Function δ

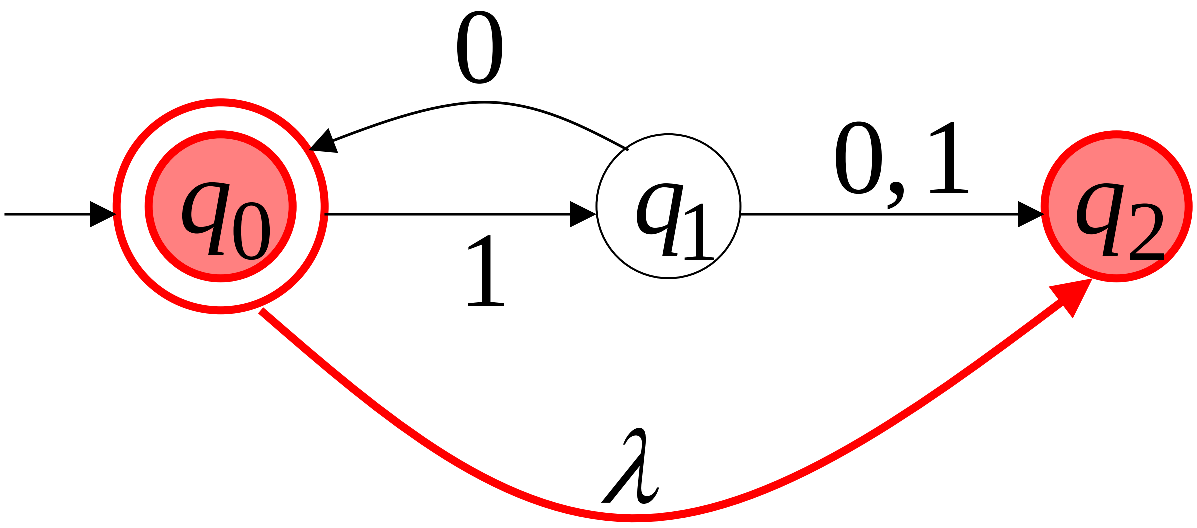
$$\delta(q_0, 1) = \{q_1\}$$



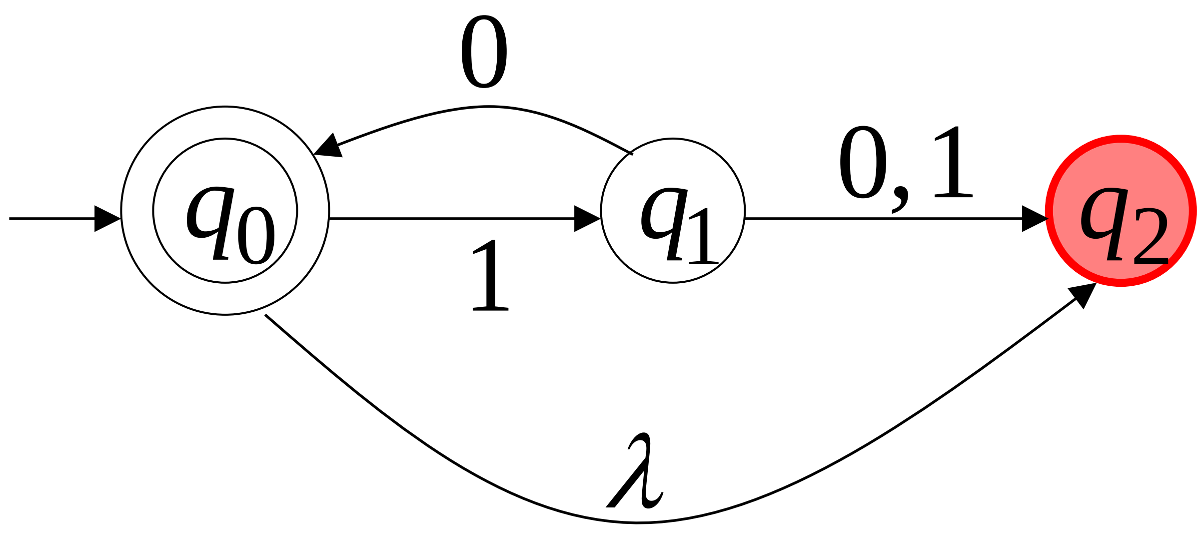
$$\delta(q_1, 0) = \{q_0, q_2\}$$



$$\delta(q_0, \lambda) = \{q_0, q_2\}$$

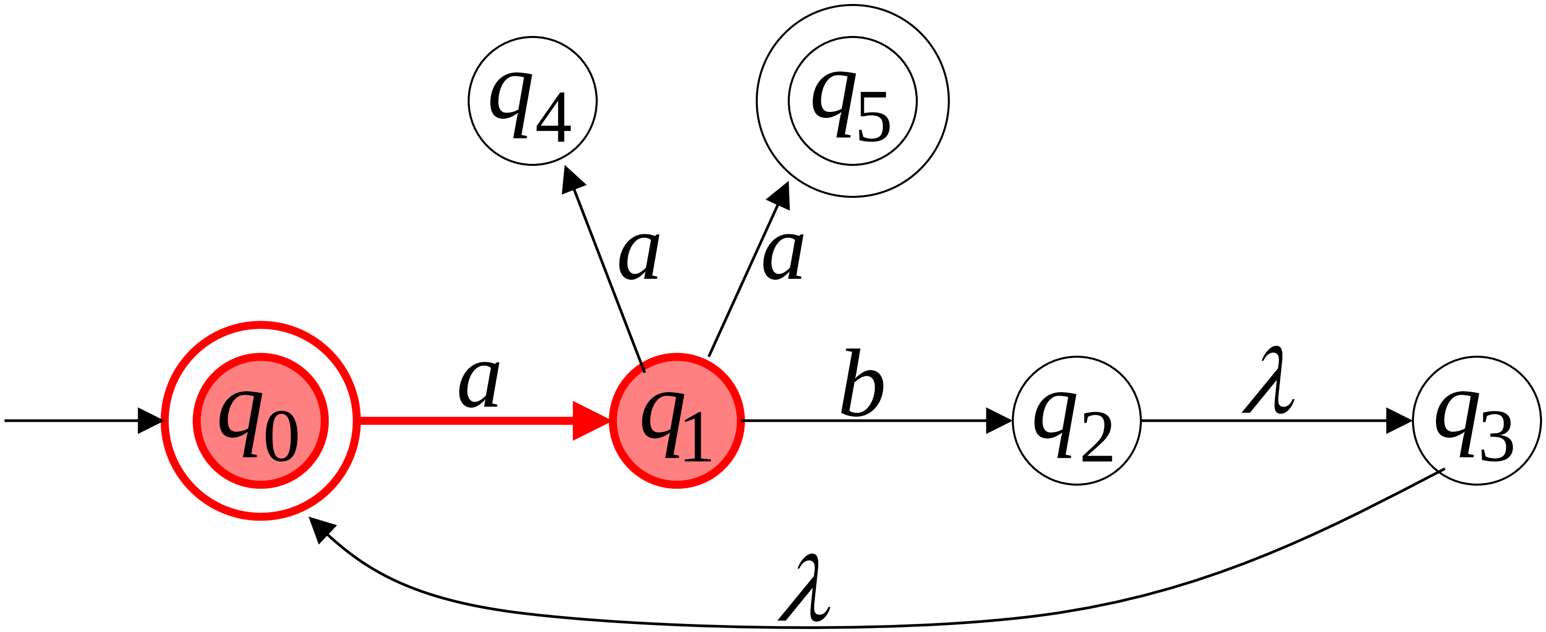


$$\delta(q_2, 1) = \varnothing$$

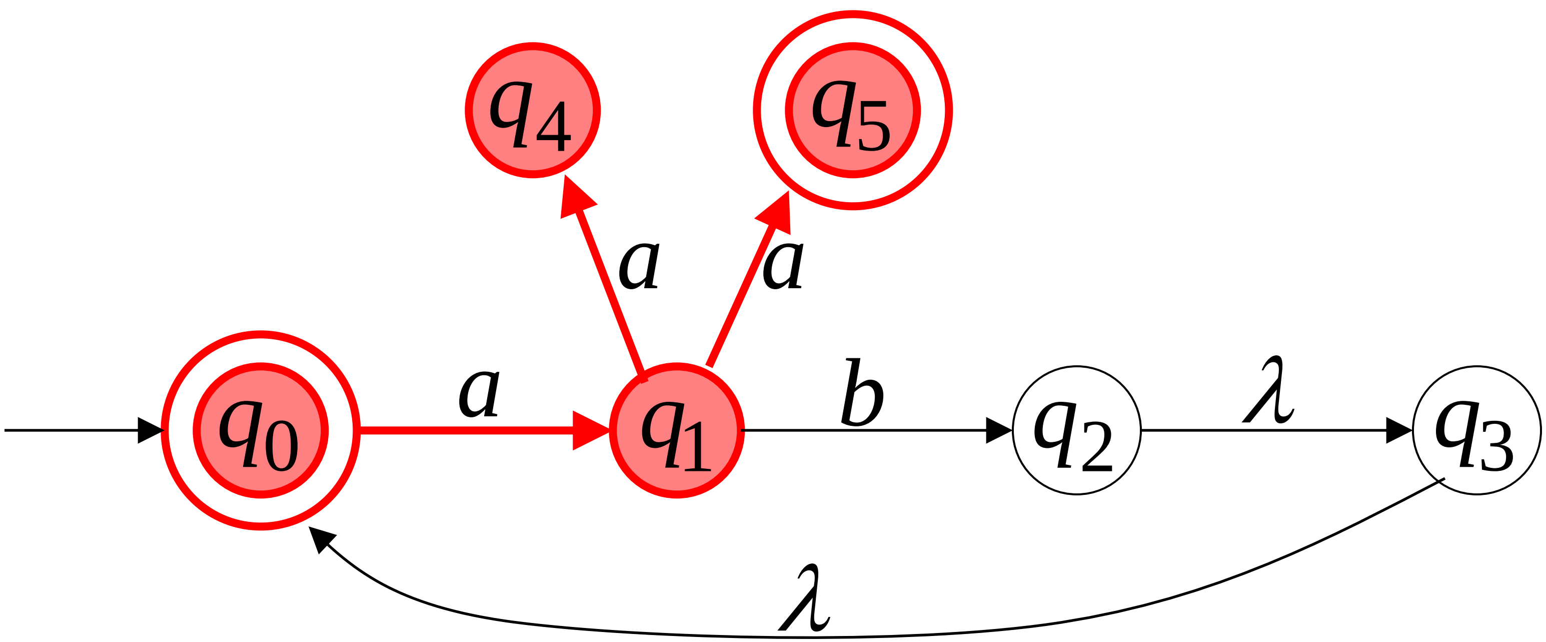


Extended Transition Function δ^*

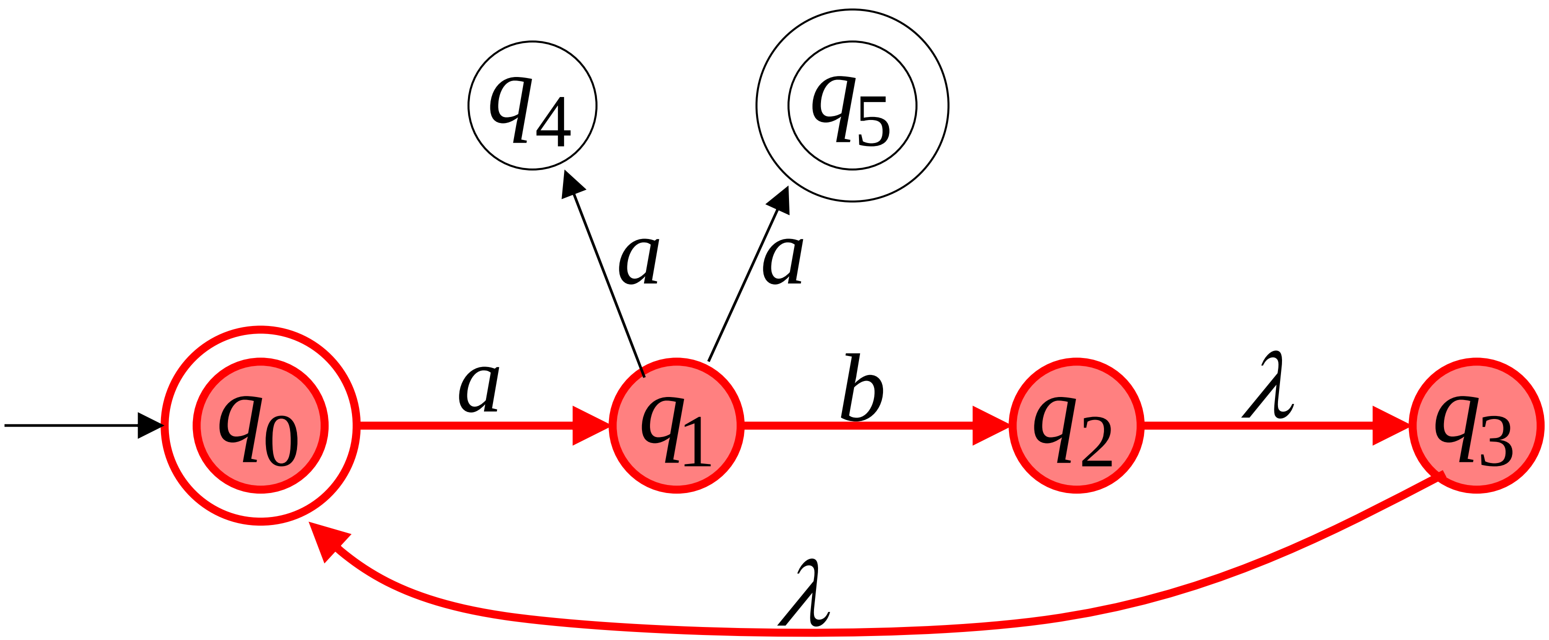
$$\delta^*(q_0, a) = \{q_1\}$$



$$\delta^*(q_0, aa) = \{q_4, q_5\}$$

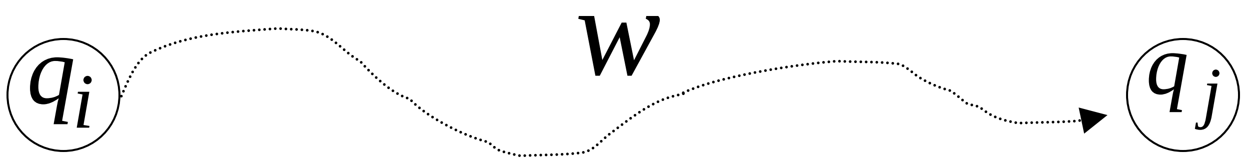


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

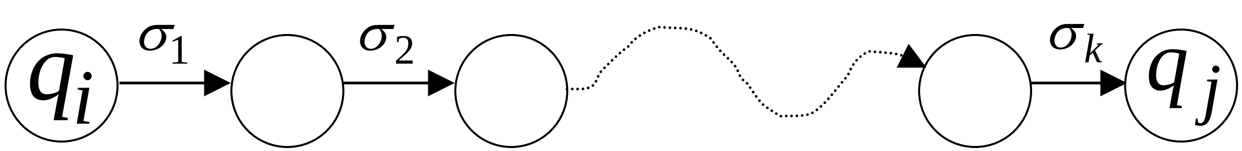


Formally

$q_j \in \delta^*(q_i, w)$: there is a walk from q_i to q_j with label w

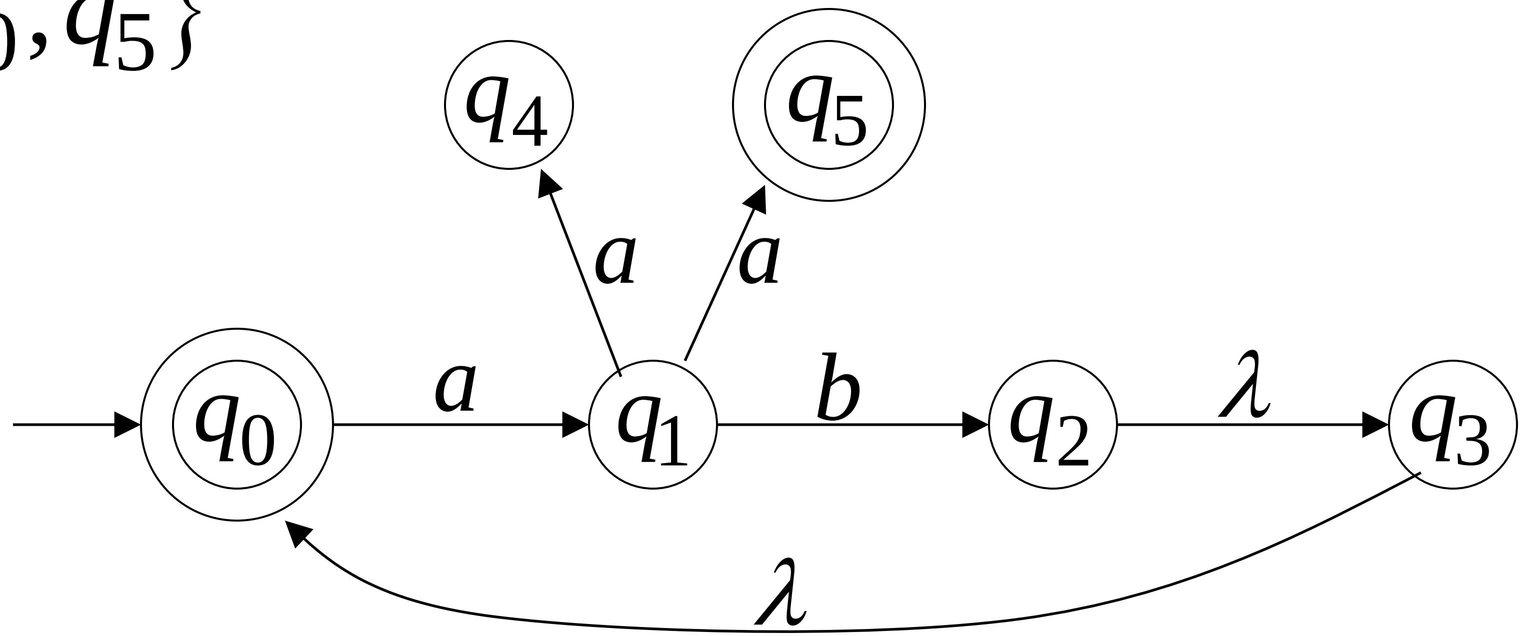


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



The Language of an NFA M

$F = \{q_0, q_5\}$

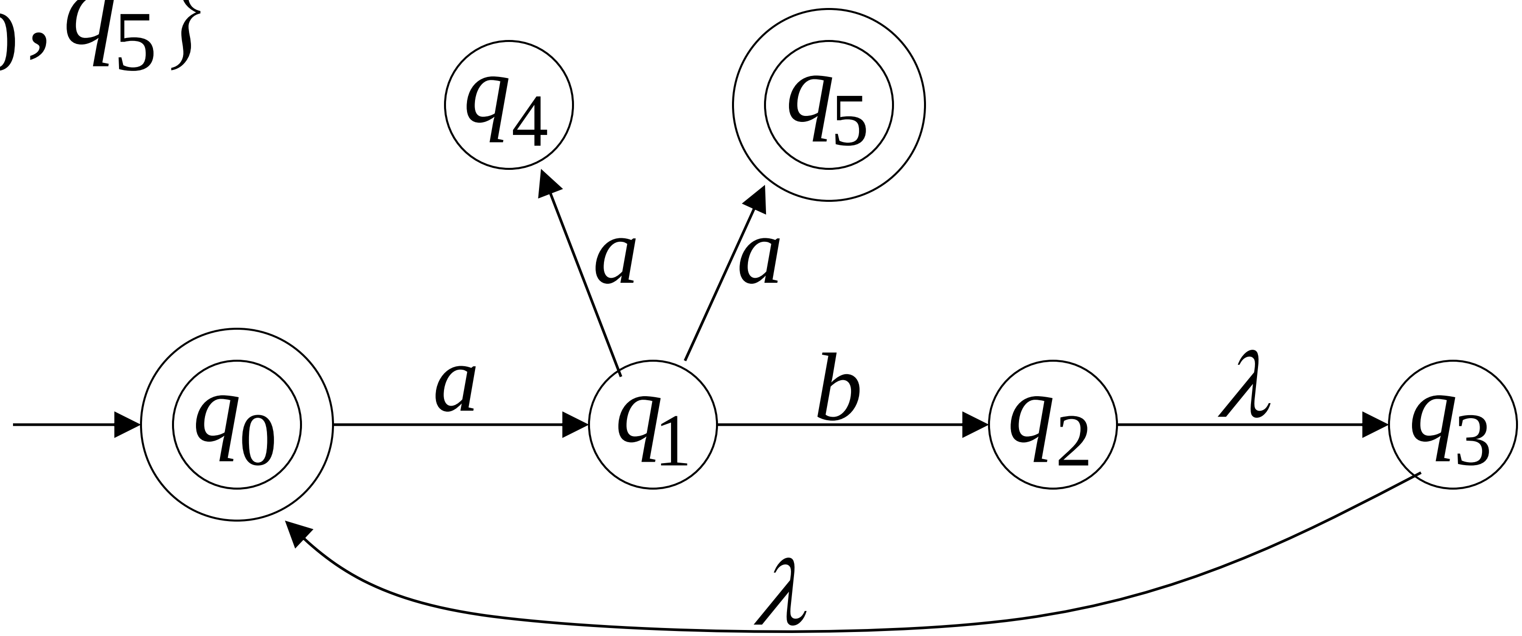


$$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\}$$

$$\in F$$

$$aa \in L(M)$$

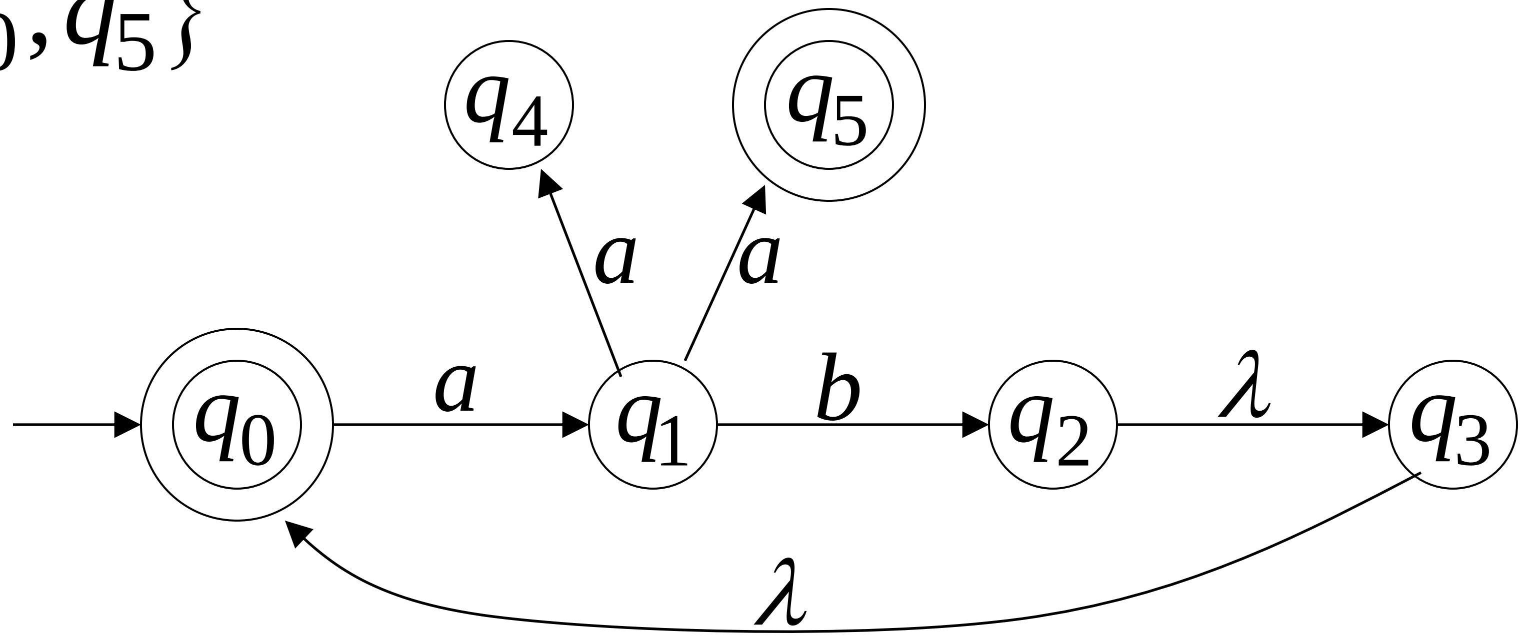
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\} \qquad ab \in L(M)$$

$$\in F$$

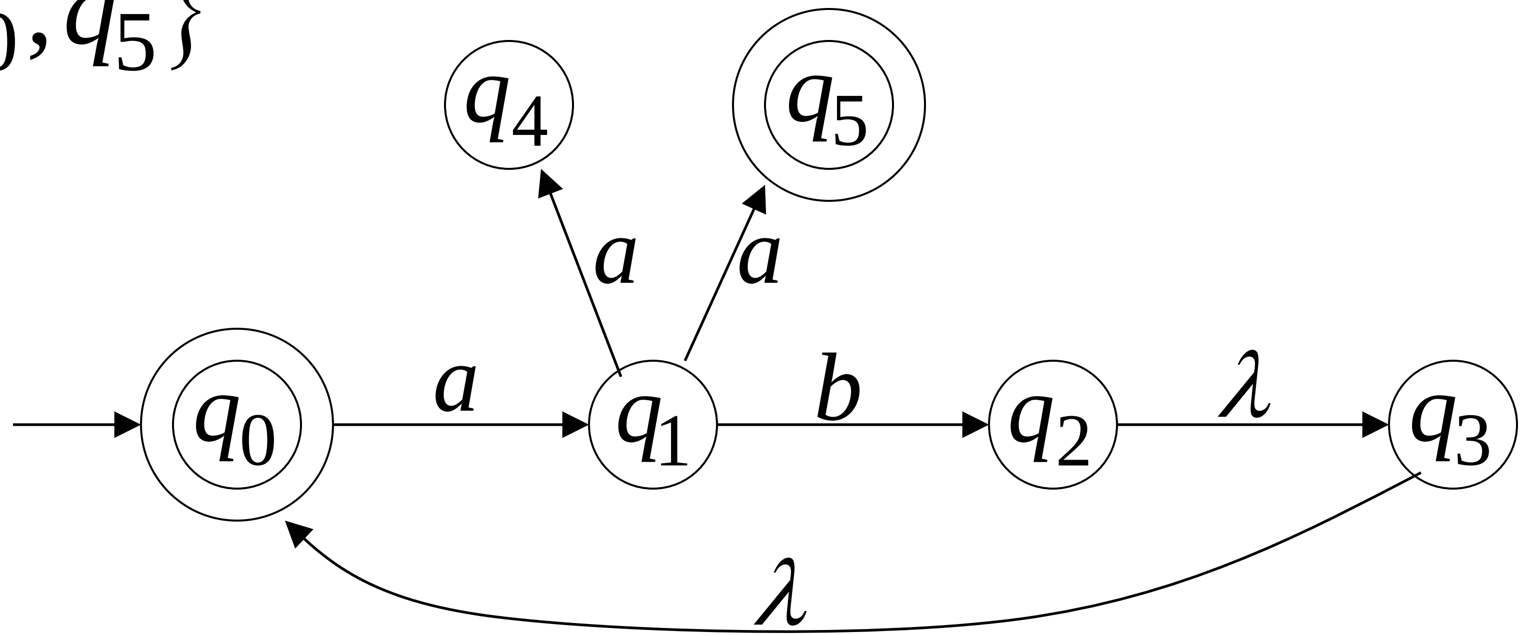
$F = \{q_0, q_5\}$



$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\}$ $\quad aaba \in L(M)$

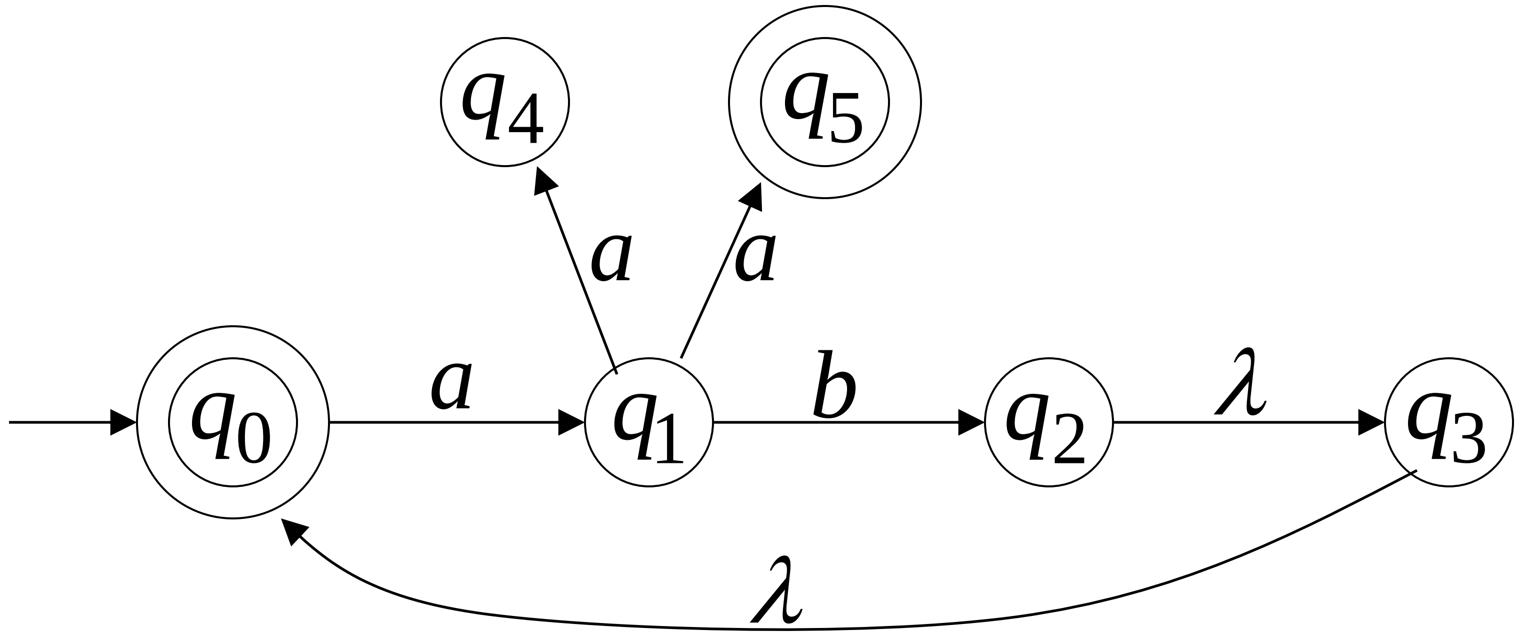
$\searrow \in F$

$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, aba) = \{q_1\} \qquad aba \notin L(M)$$

$$\notin F$$



$$L(M) = \{\lambda\} \cup \{ab\}^*.\{\lambda, aa\}$$
$$= \{ab\}^*.\{\lambda, aa\}$$

Formally

The language accepted by NFA M is:

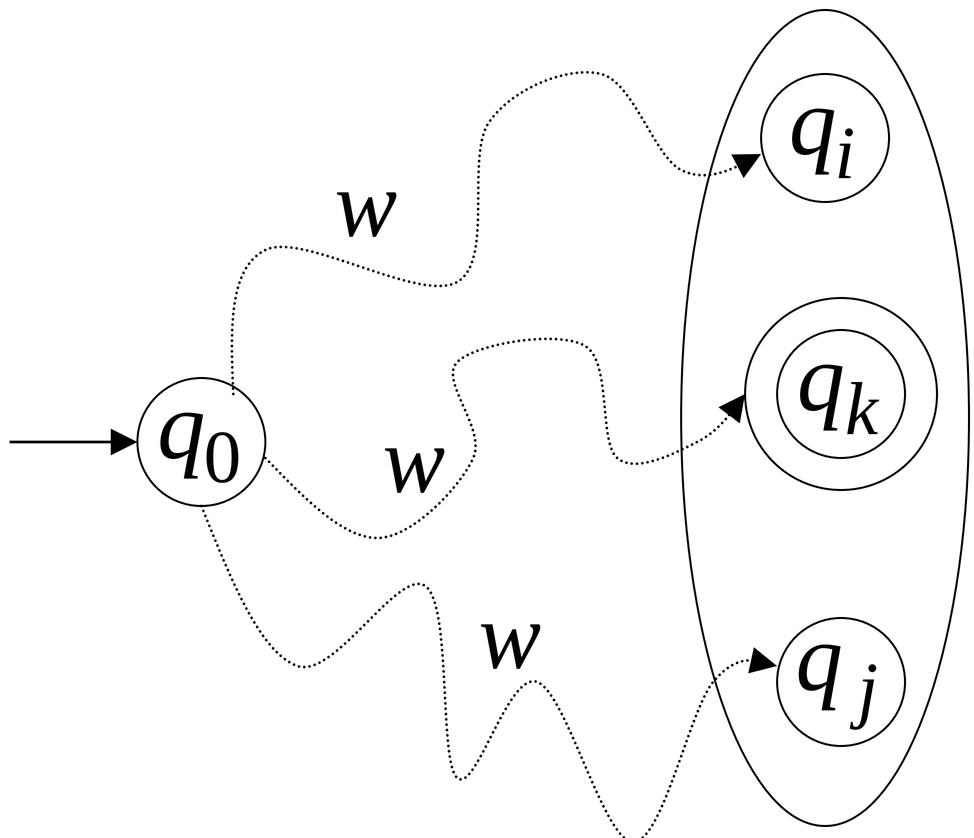
$$L(M) = \{w_1, w_2, w_3, \ldots\}$$

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

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:

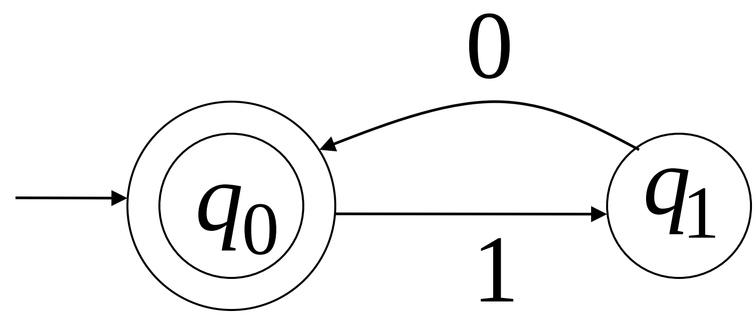
Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

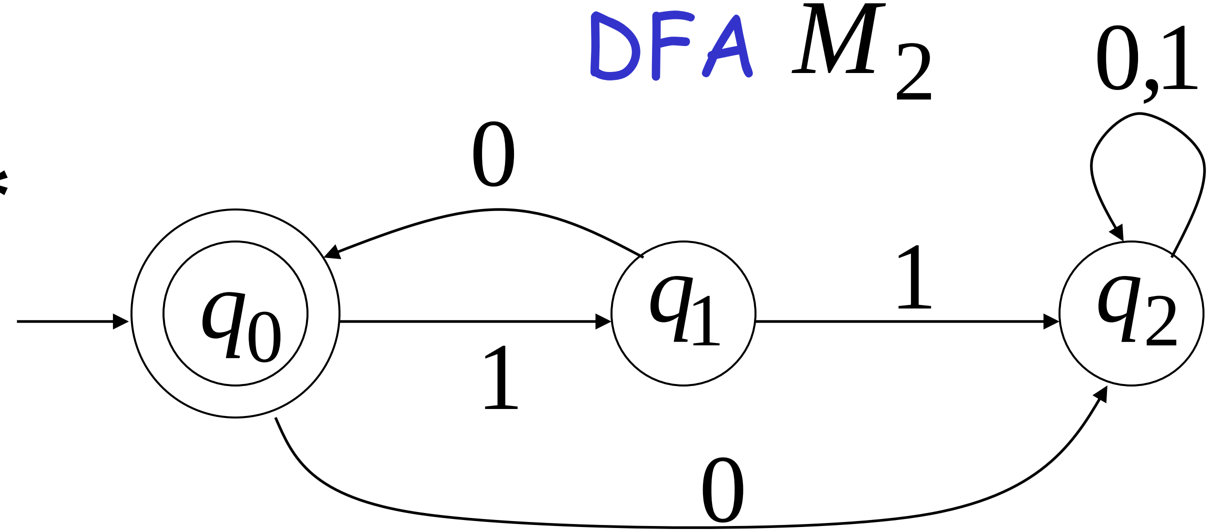
$$L(M_1) = \{10\}^*$$

NFA M_1



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

DFA M_2



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

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

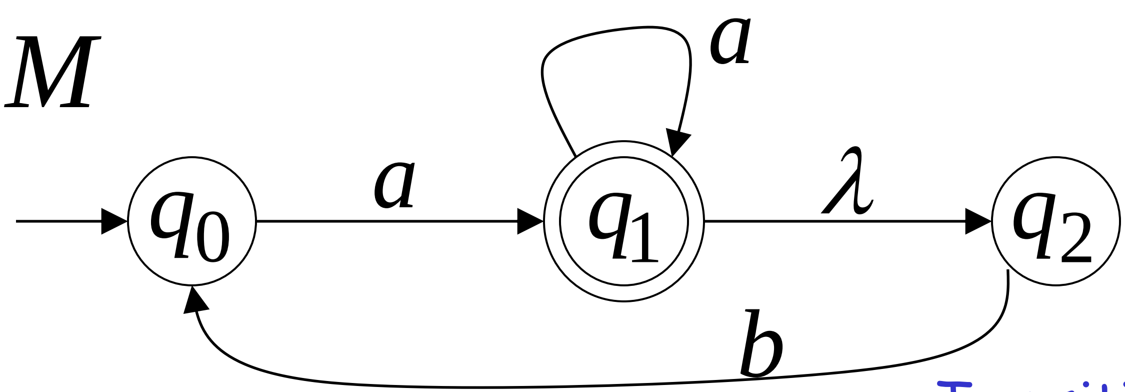
Proof: Any NFA can be converted to an equivalent DFA

$\Downarrow$

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

Convert NFA to DFA

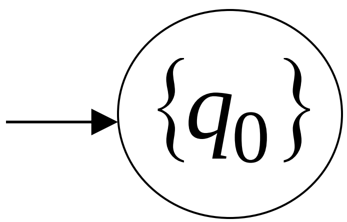
NFA M



Transition table for NFA M

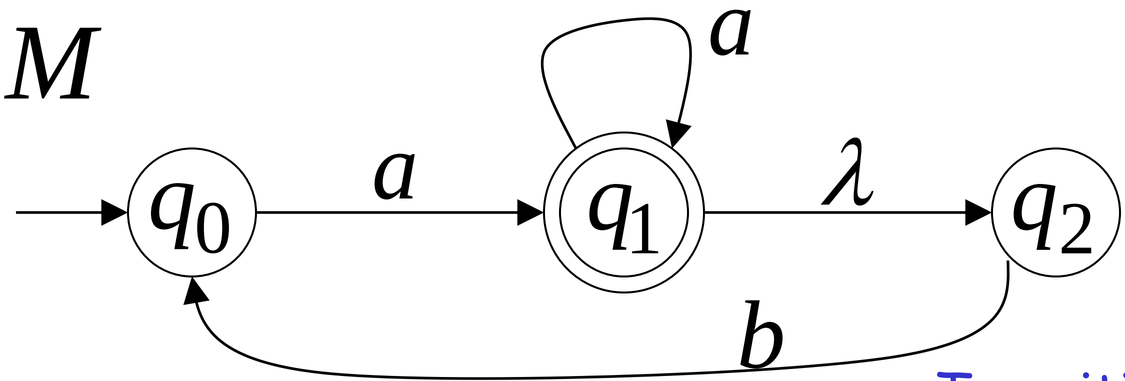
	a	b
q0	{q1, q2}	∅
q1	{q1, q2}	{q0}
q2	∅	{q0}

DFA M'



Convert NFA to DFA

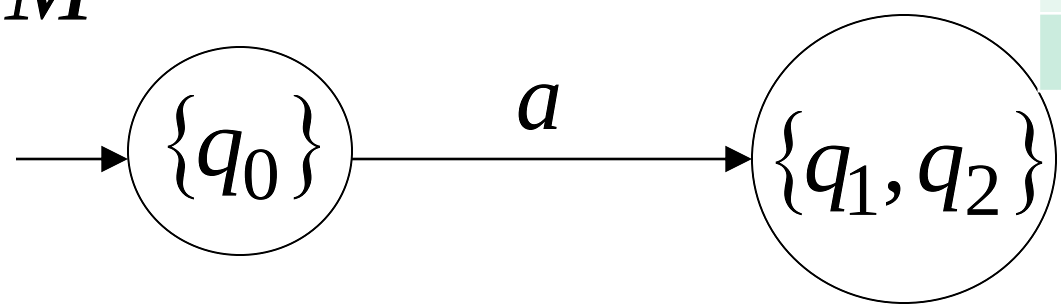
NFA M



Transition table for NFA M

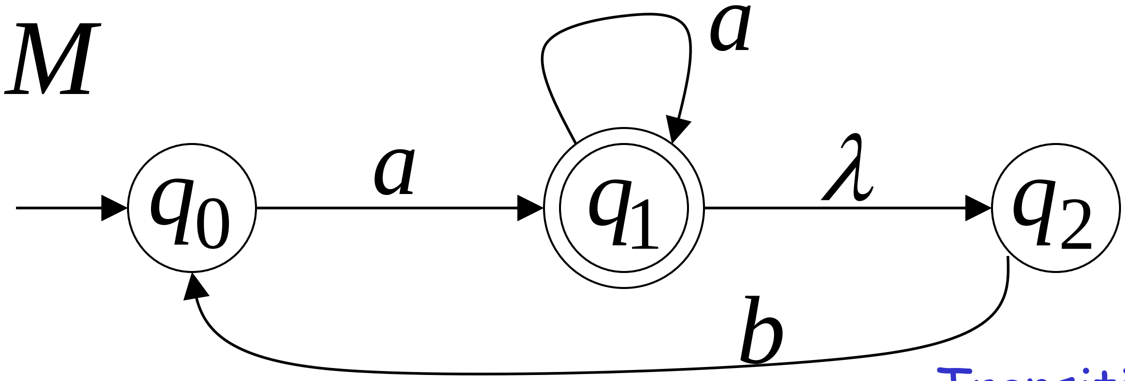
	a	b
q0	{q1, q2}	∅
q1	{q1, q2}	{q0}
q2	∅	{q0}

DFA M'



Convert NFA to DFA

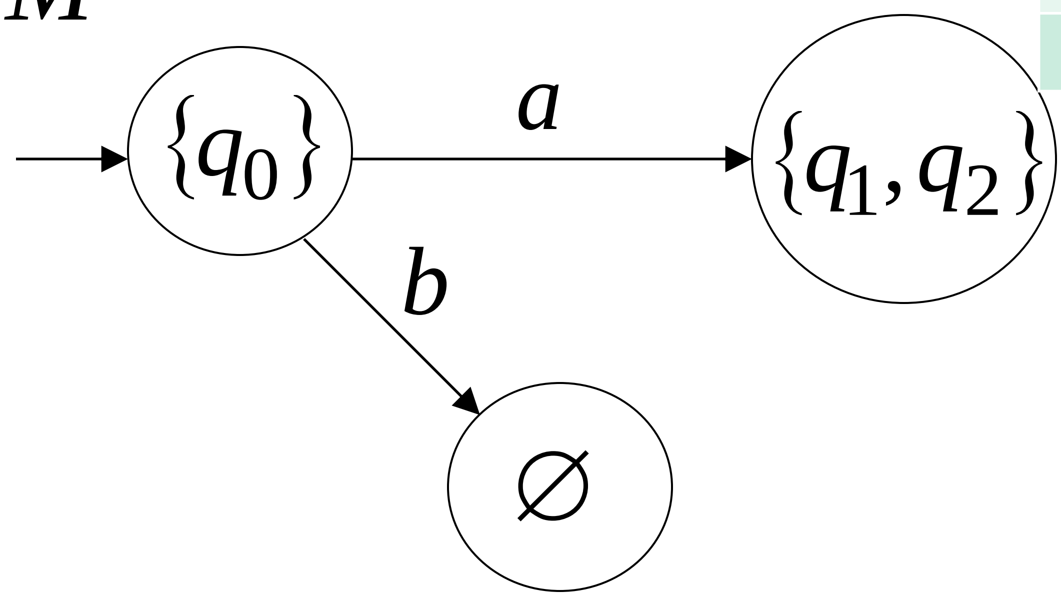
NFA M



Transition table for NFA M

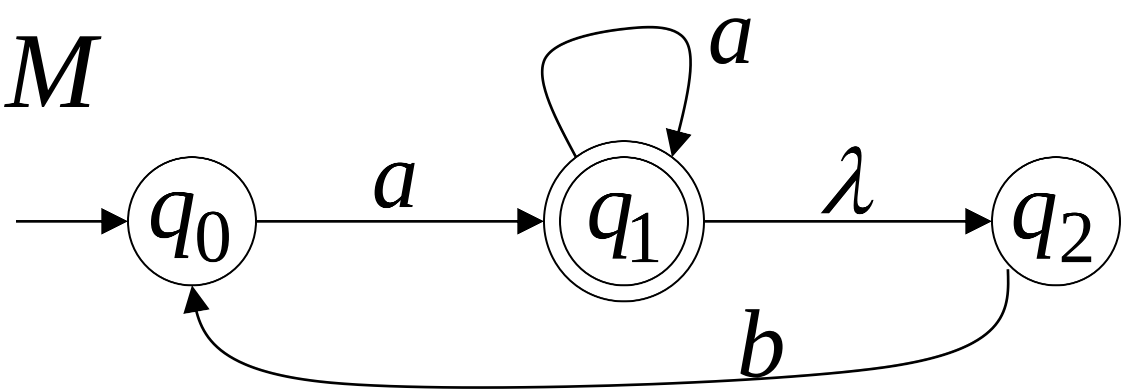
	a	b
q0	{q1, q2}	∅
q1	{q1, q2}	{q0}
q2	∅	{q0}

DFA M'

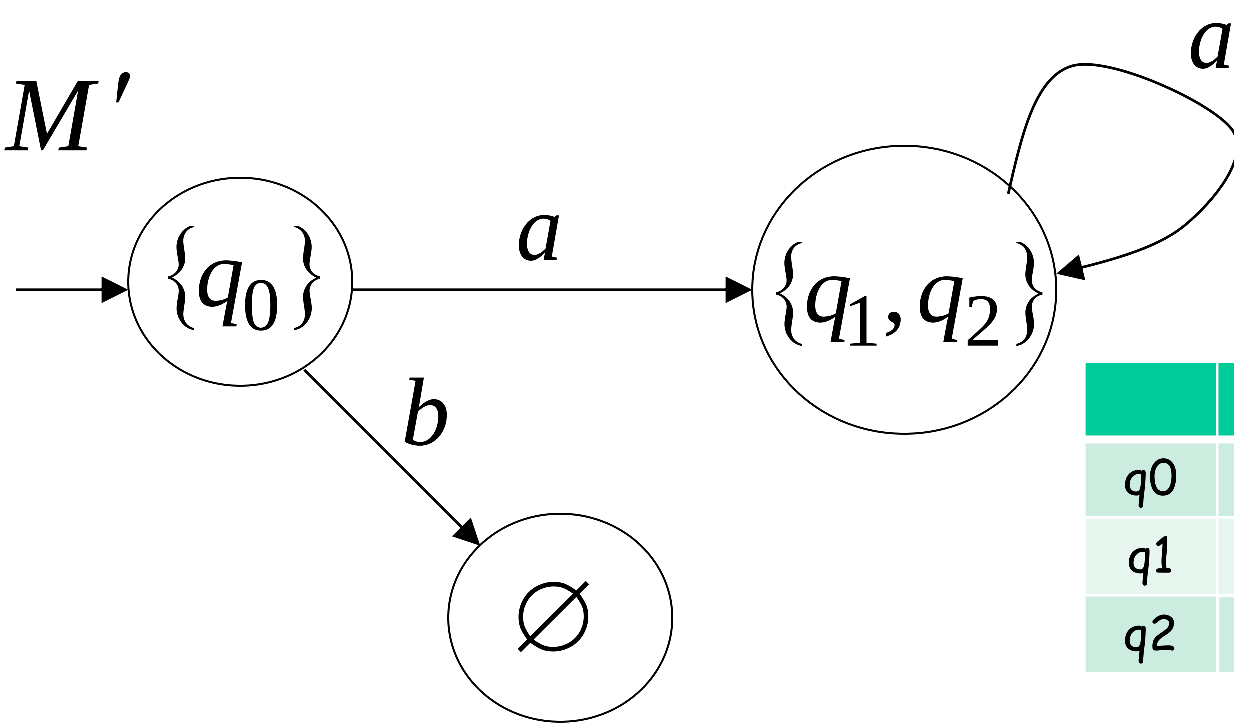


Convert NFA to DFA

NFA M



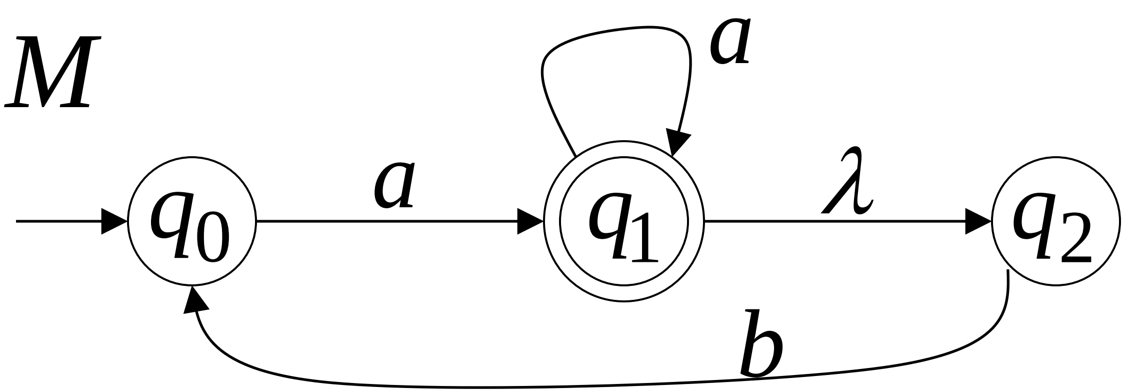
DFA M'



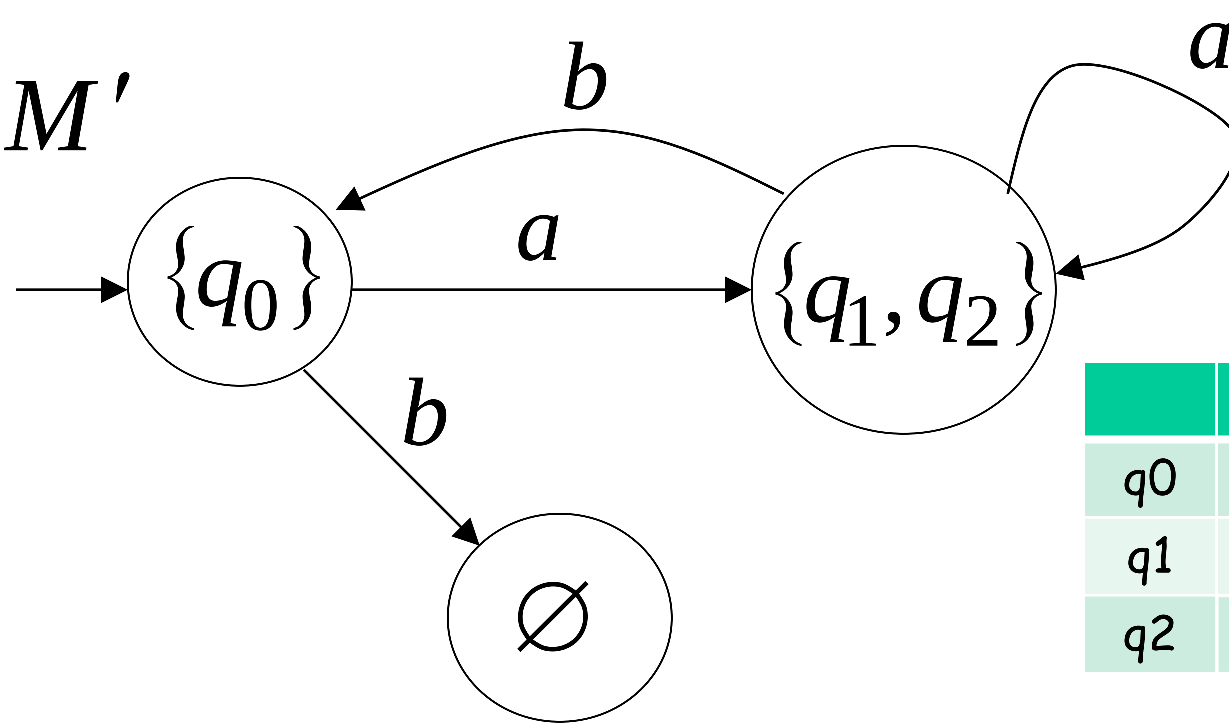
	a	b
q0	{q1, q2}	Ø
q1	{q1,q2}	{q0}
q2	Ø	{q0}

Convert NFA to DFA

NFA M



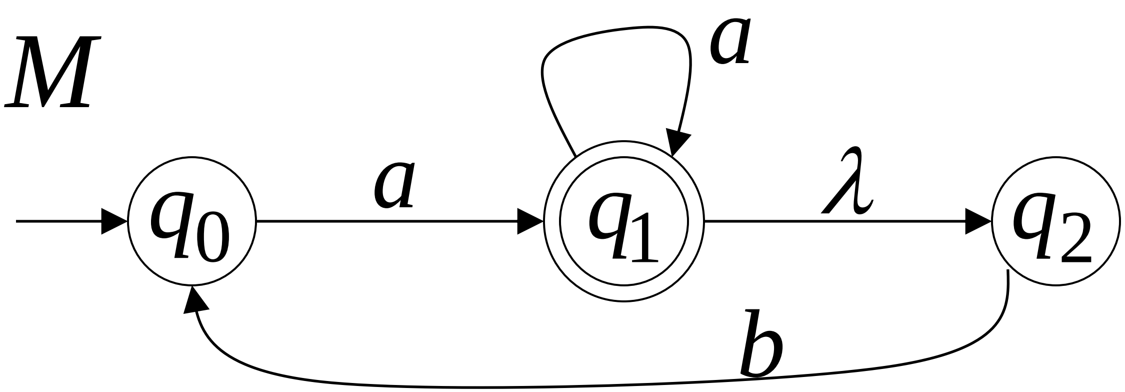
DFA M'



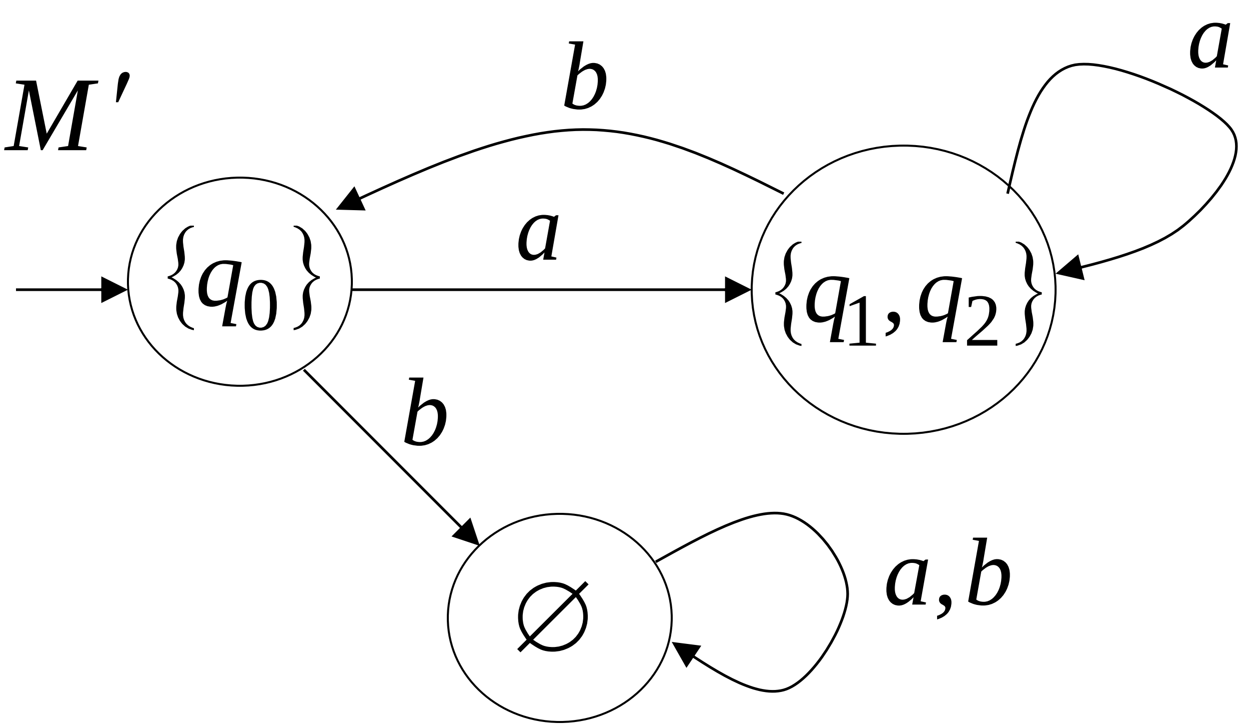
	a	b
q0	{q1, q2}	∅
q1	{q1, q2}	{q0}
q2	∅	{q0}

Convert NFA to DFA

NFA M

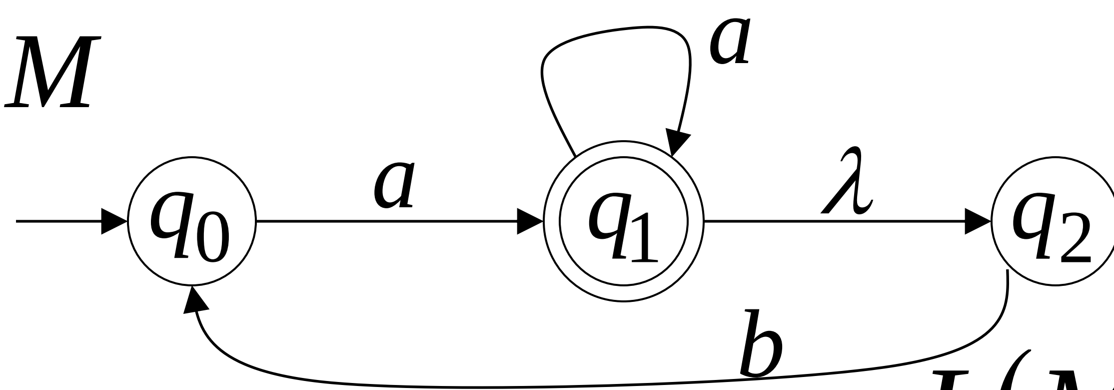


DFA M'



Convert NFA to DFA

NFA M



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

DFA M'