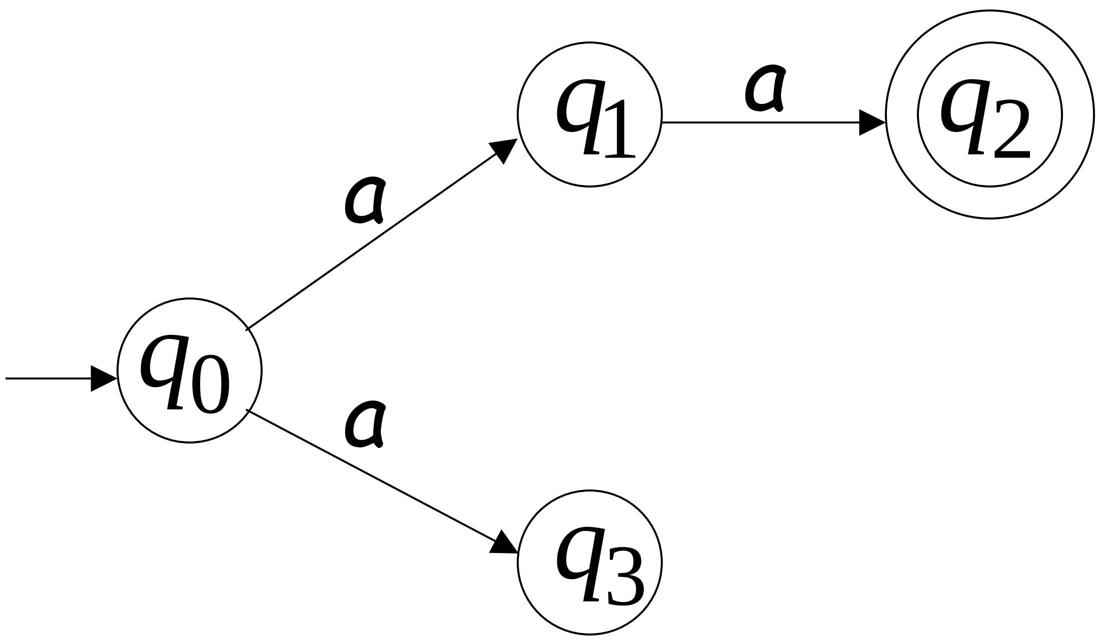
Formal Languages
Non-Deterministic Automata

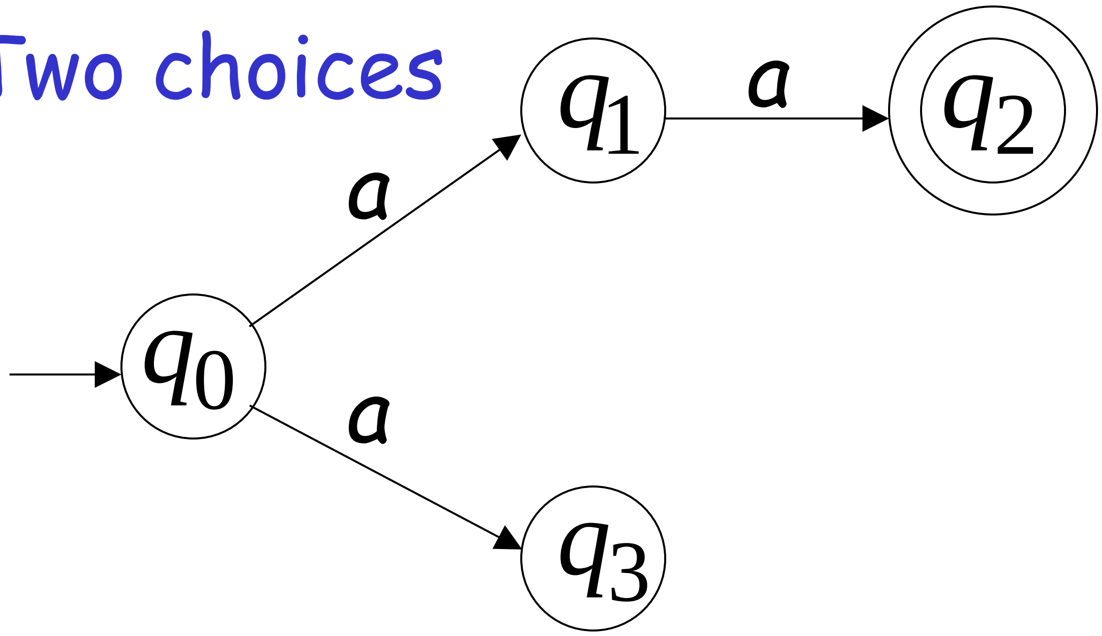
Nondeterministic Finite Automaton (NFA)

Alphabet = $\{a\}$

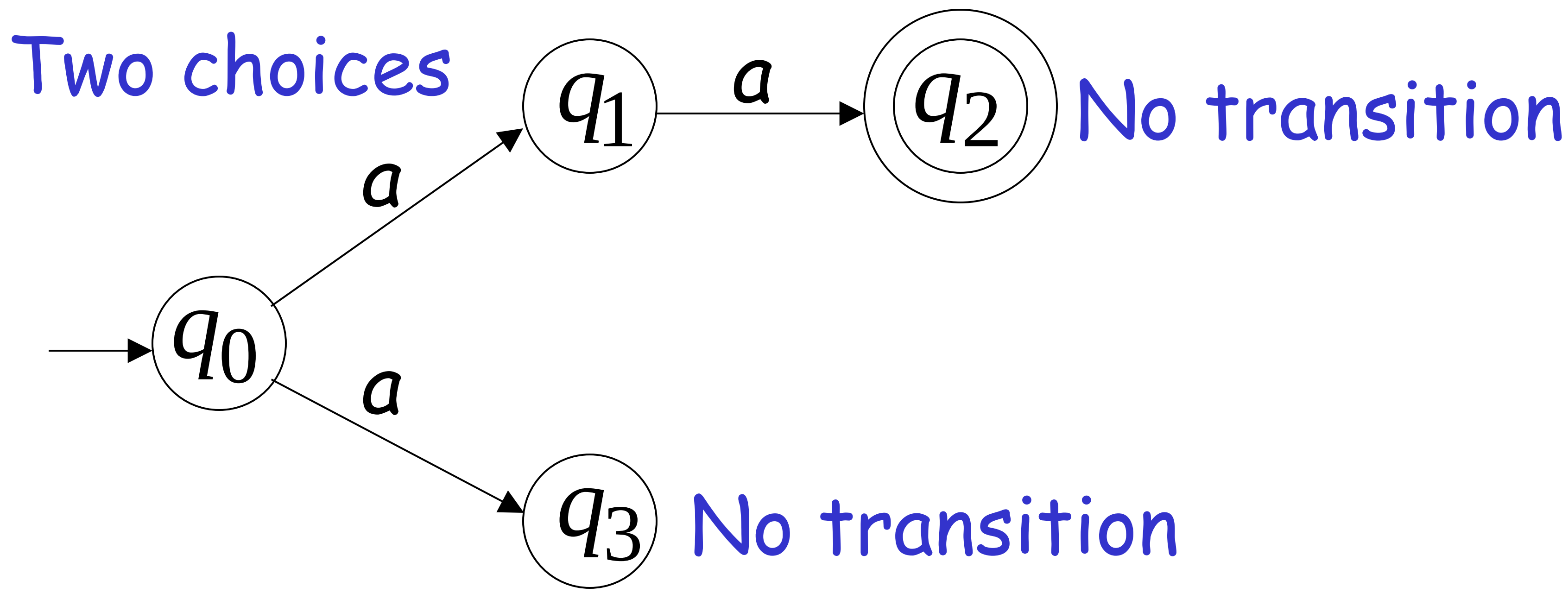


Alphabet = $\{a\}$

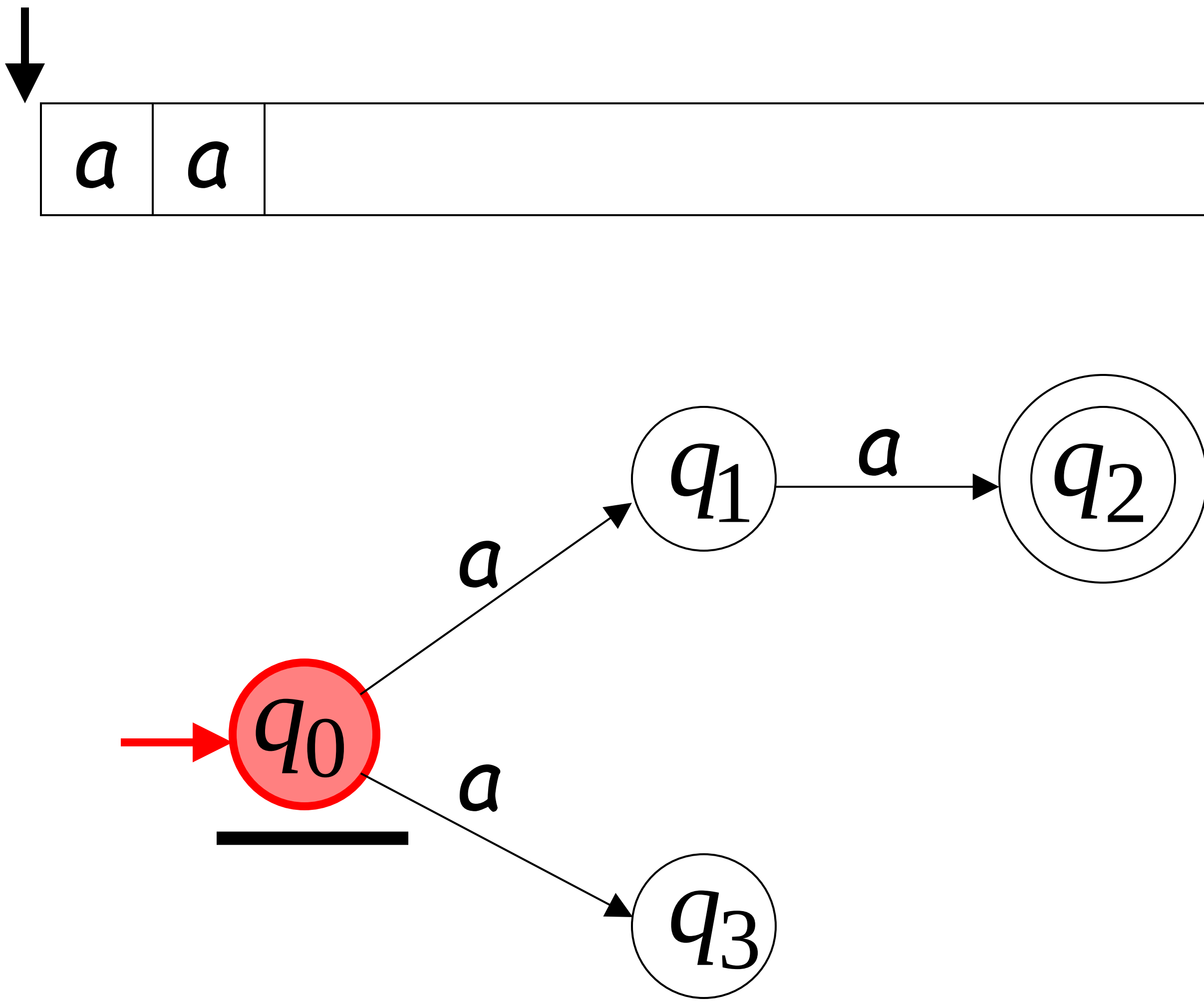
Two choices



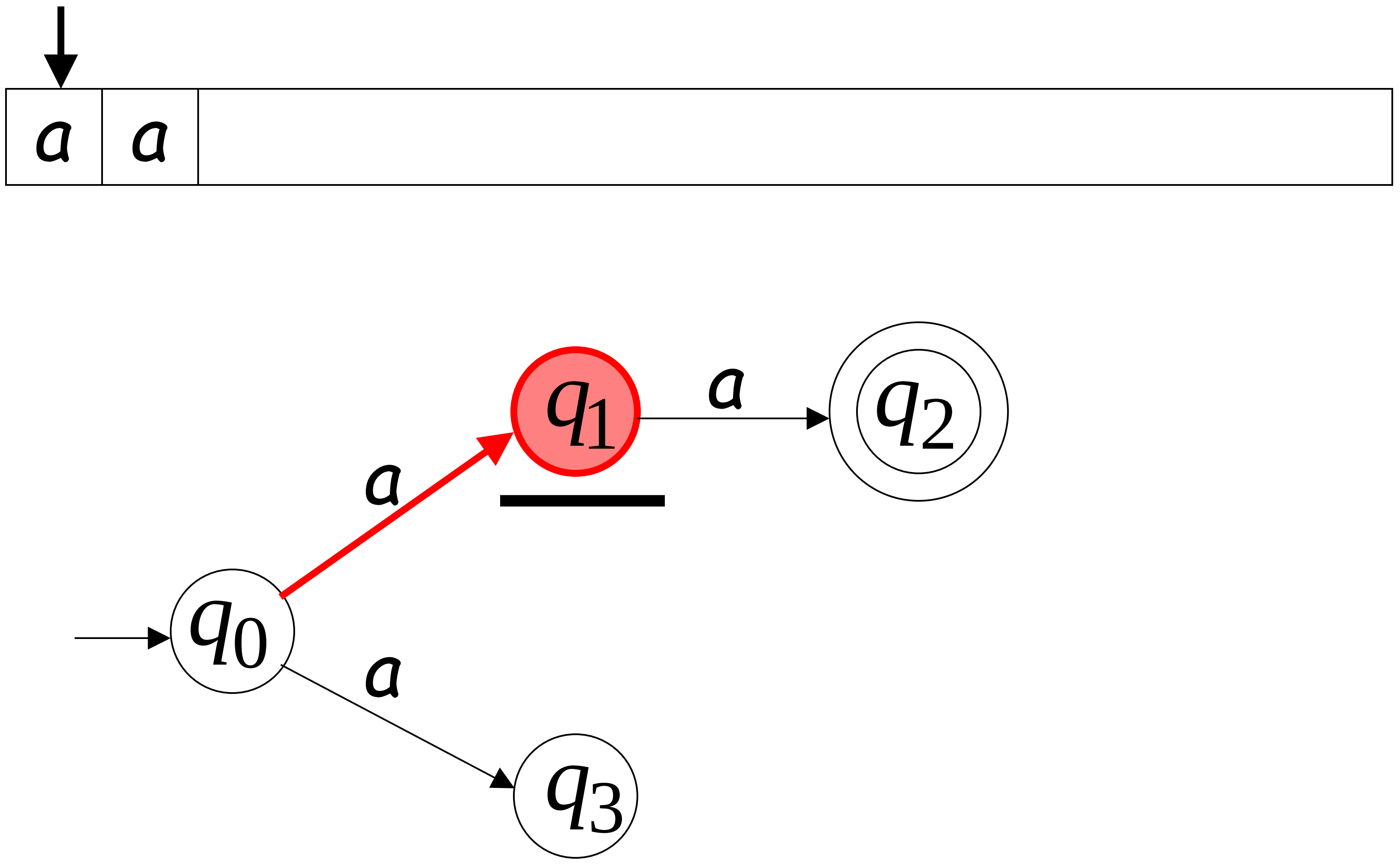
Alphabet = $\{a\}$



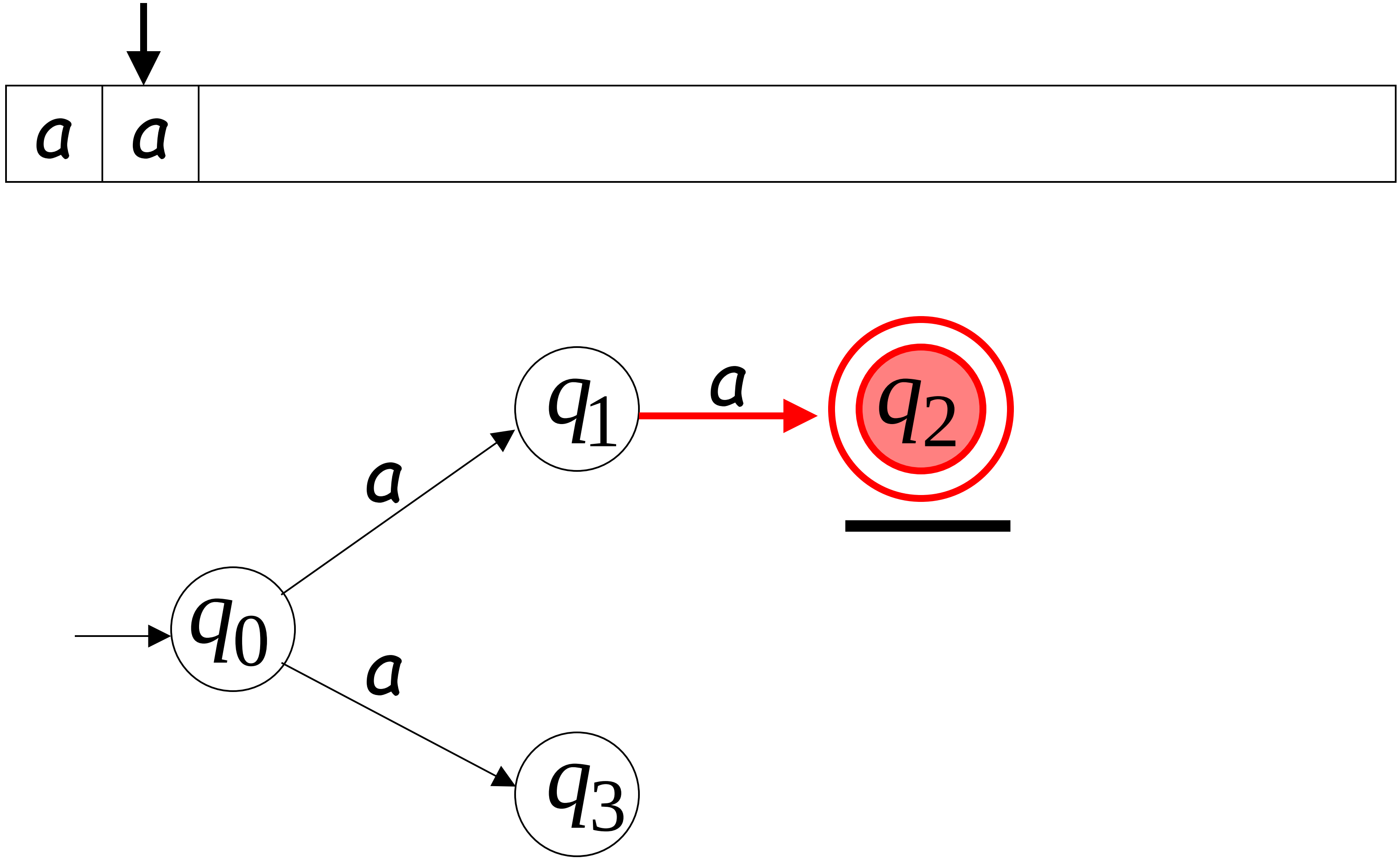
First Choice



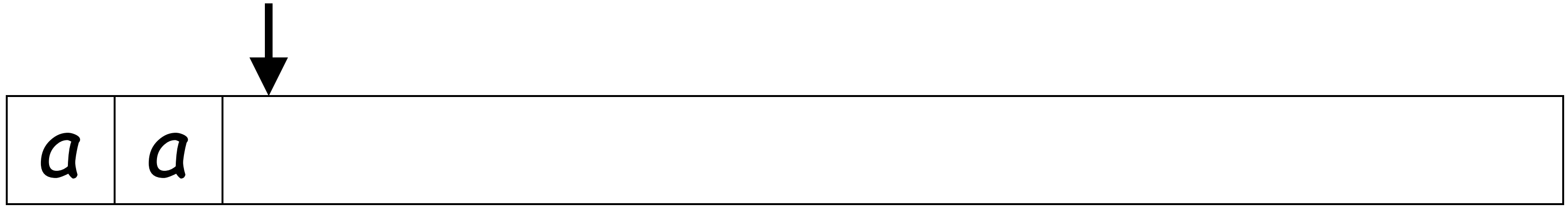
First Choice



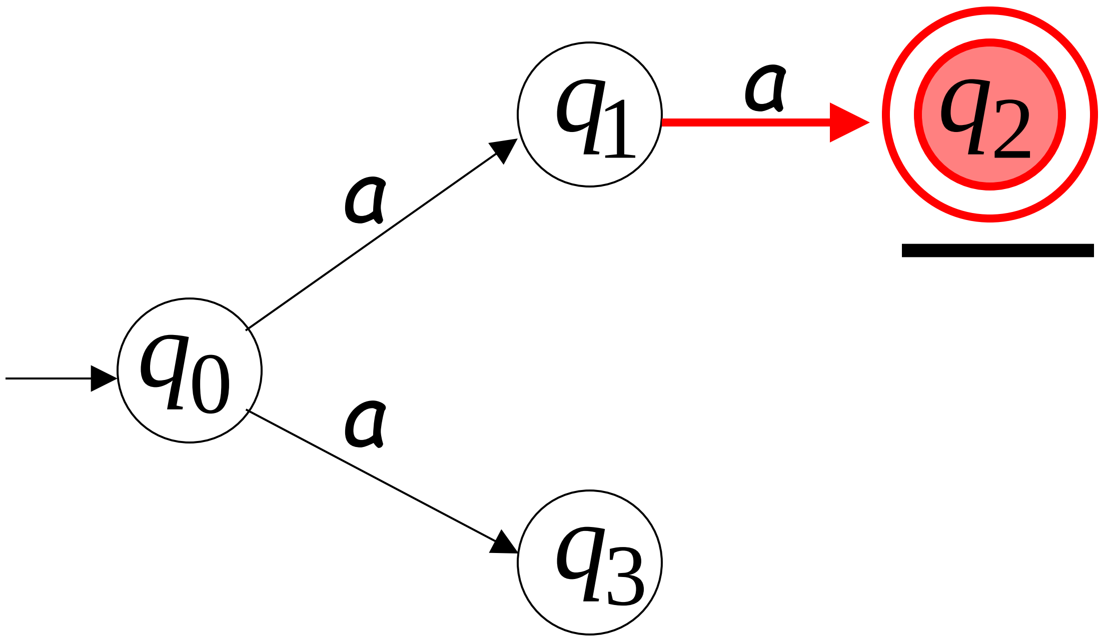
First Choice



First Choice

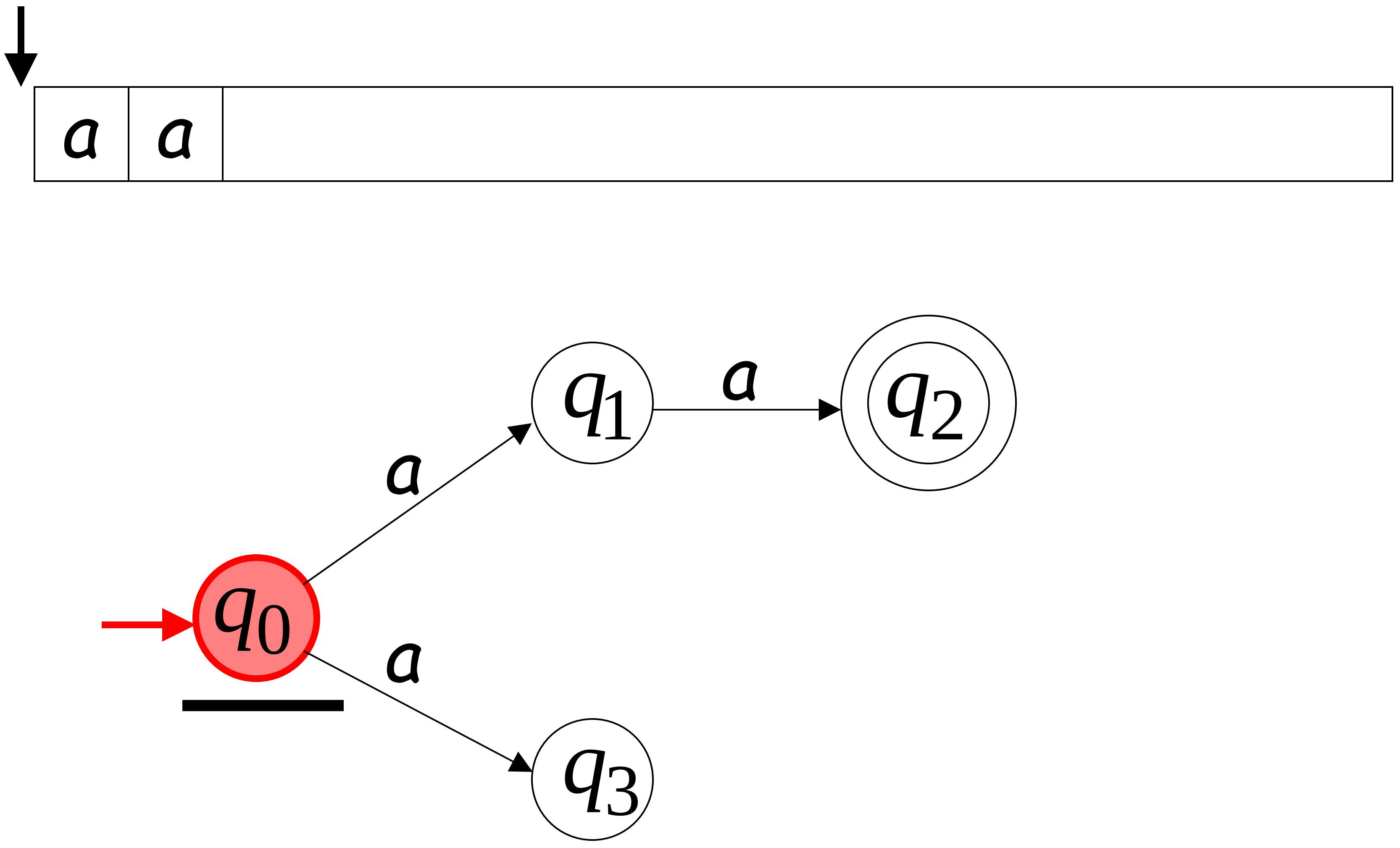


All input is consumed

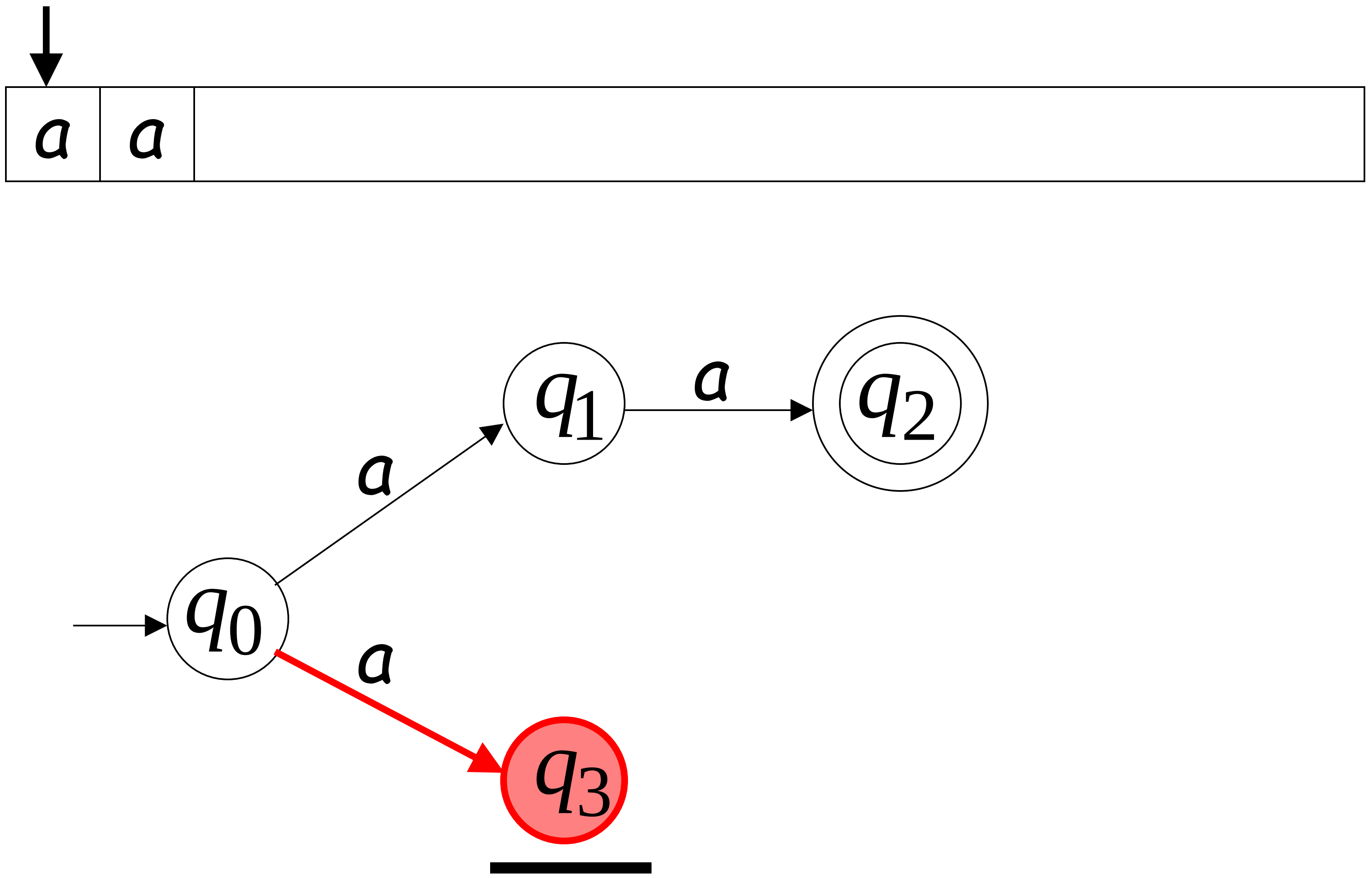


"accept"

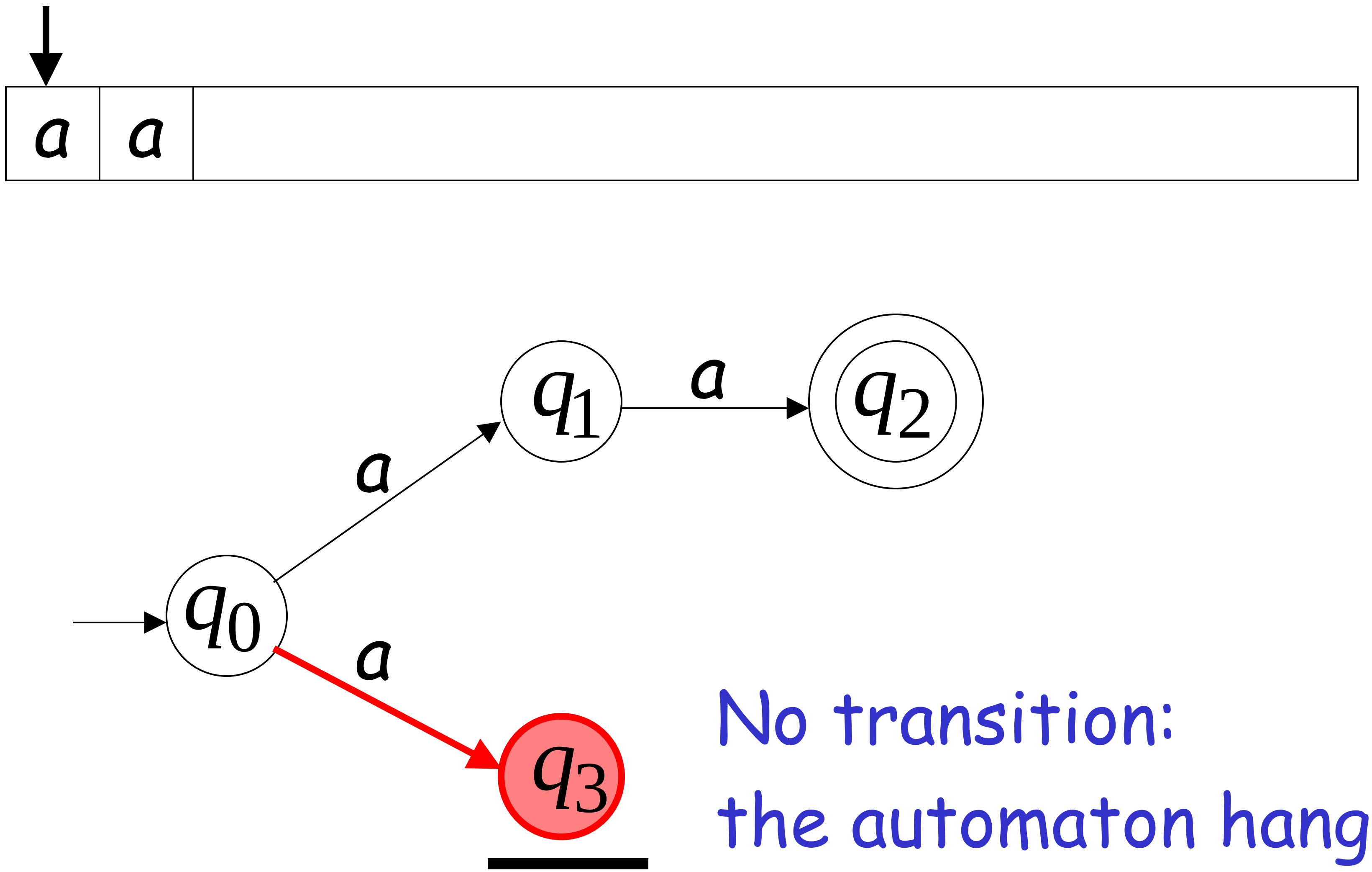
Second Choice



Second Choice

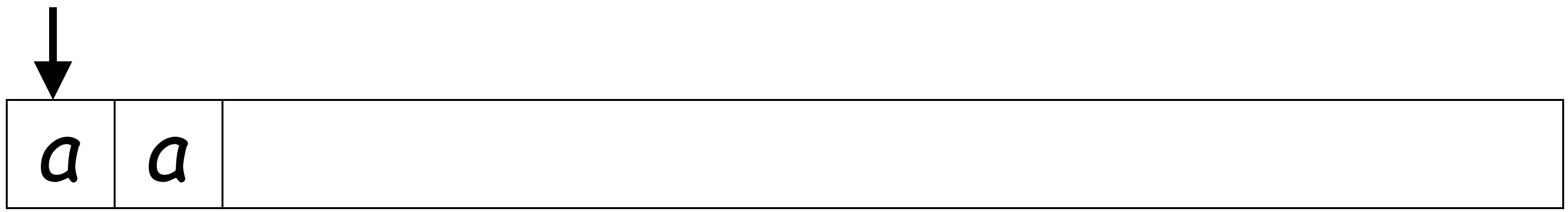


Second Choice

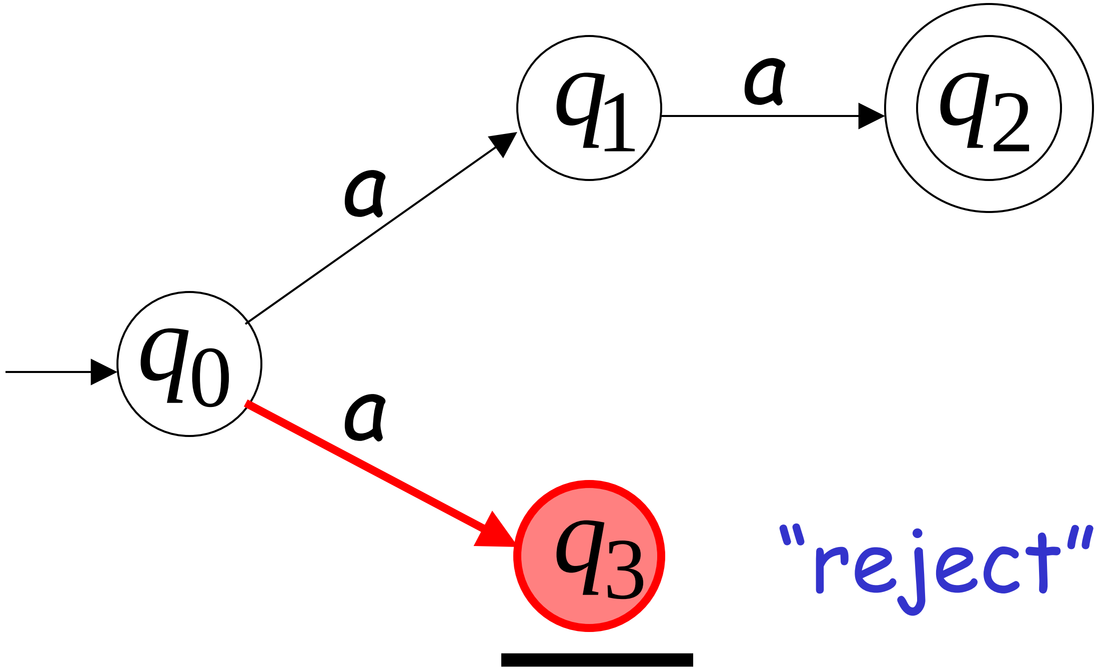


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



"reject"

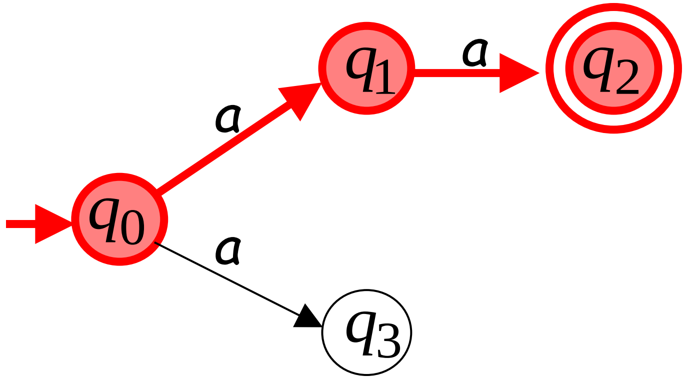
An NFA accepts a string:
when there is a computation of the NFA that accepts the string

There is a computation:
all the input is consumed and the automaton is in an accepting state

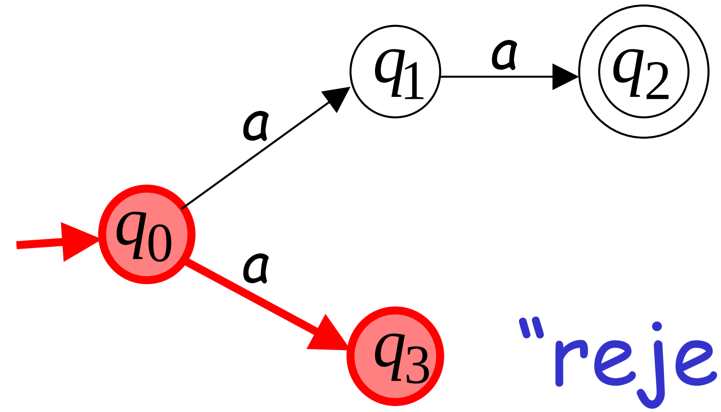
Example

aa is accepted by the NFA:

"accept"

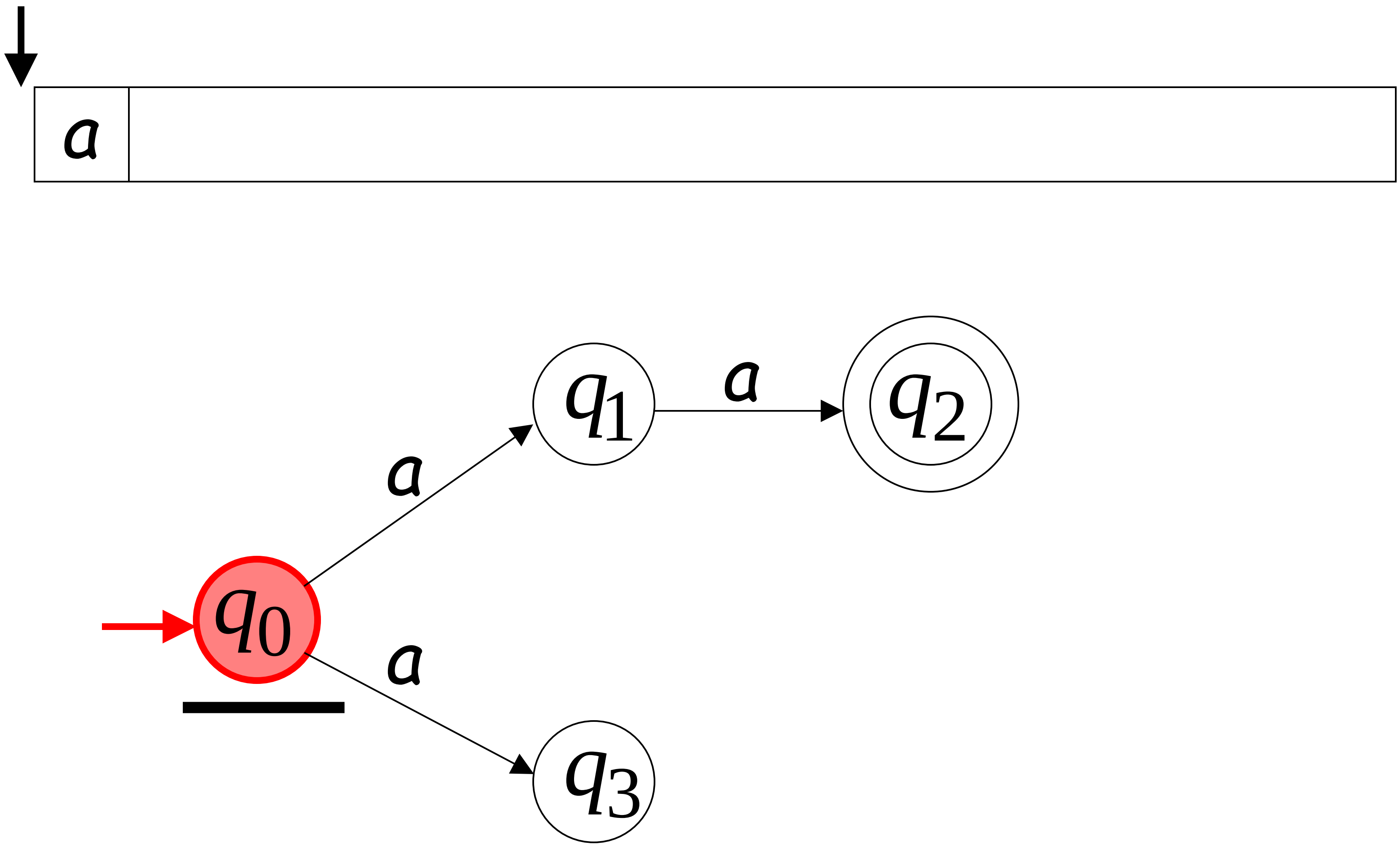


because this computation accepts aa

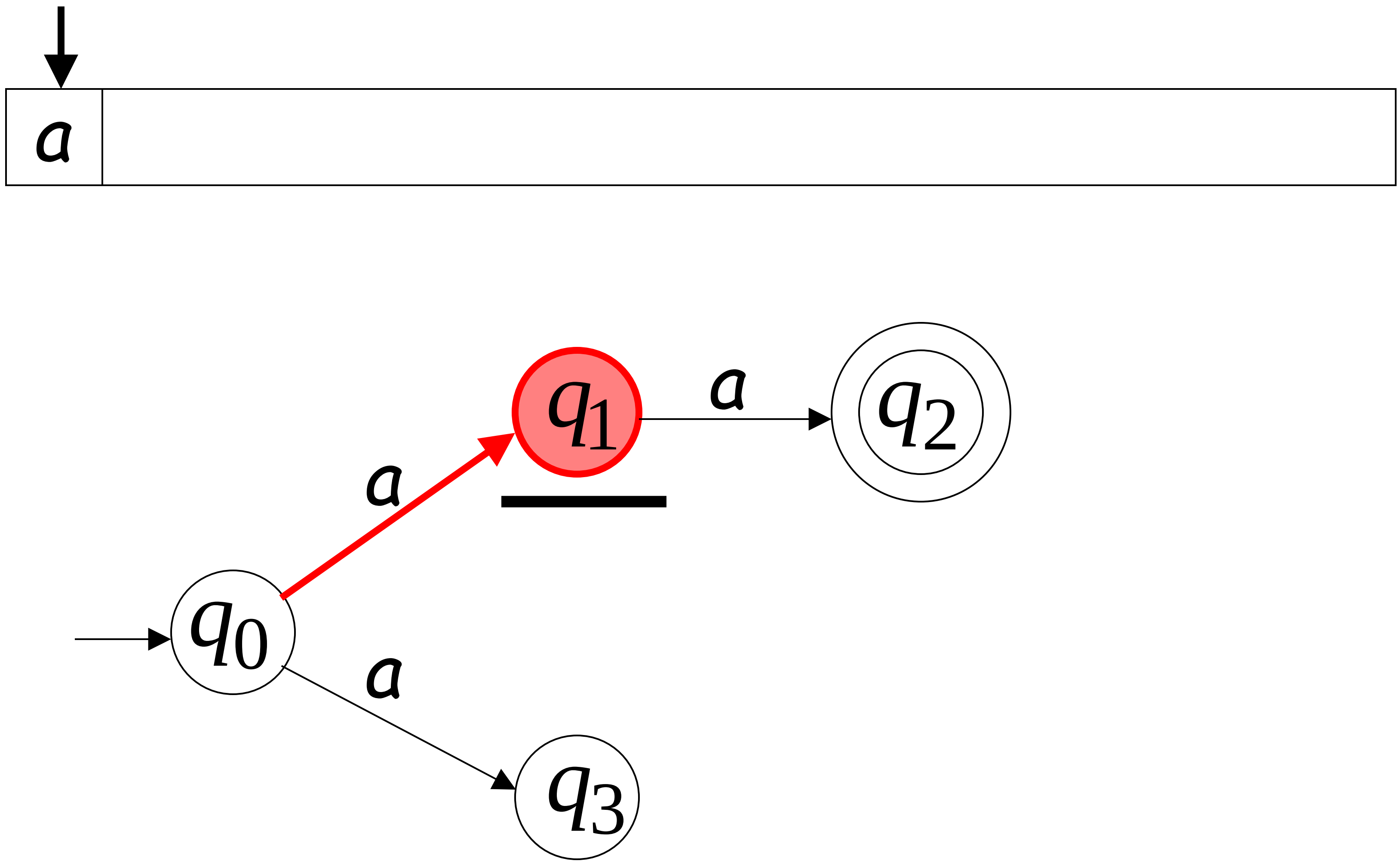


"reject"

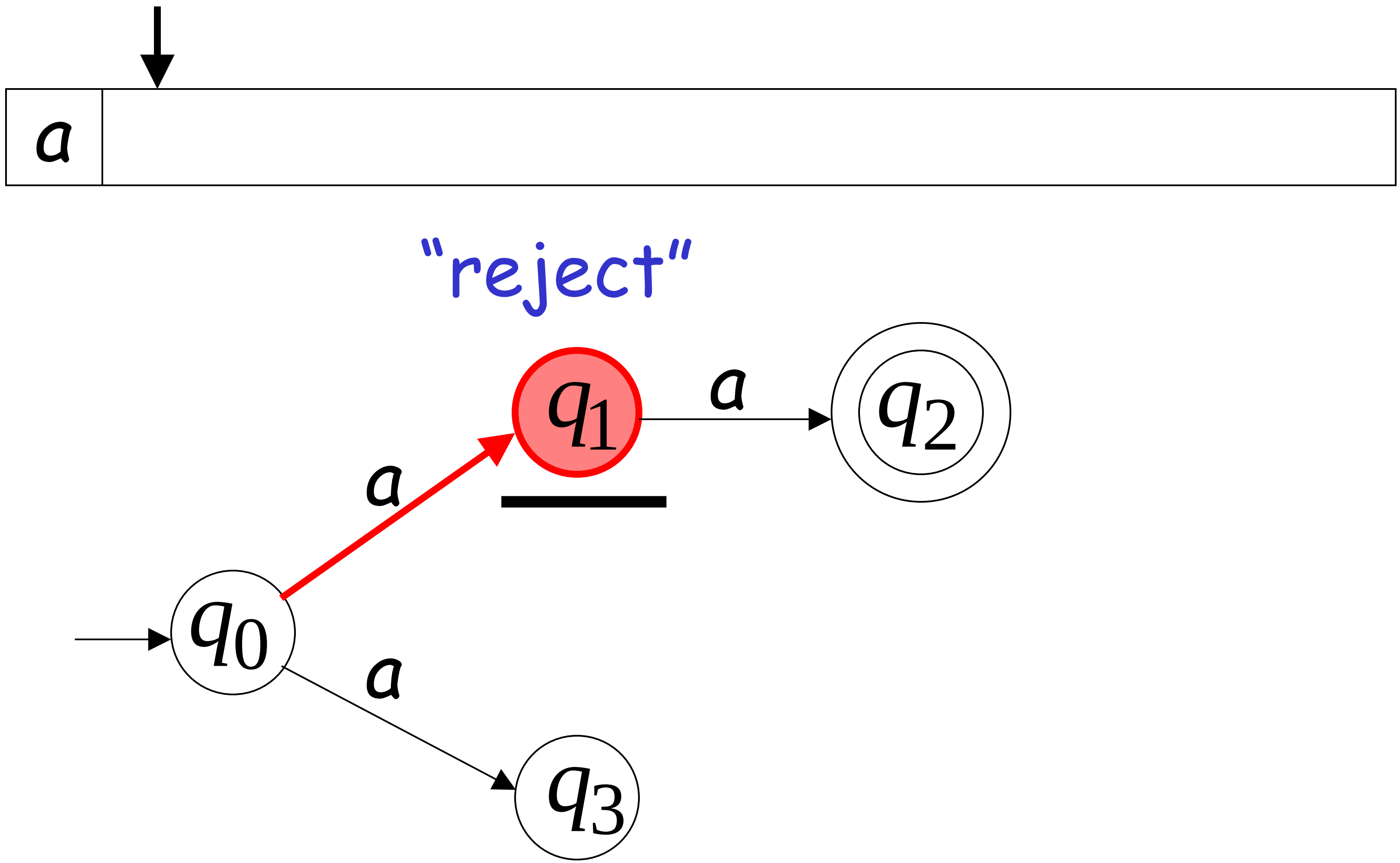
Rejection example



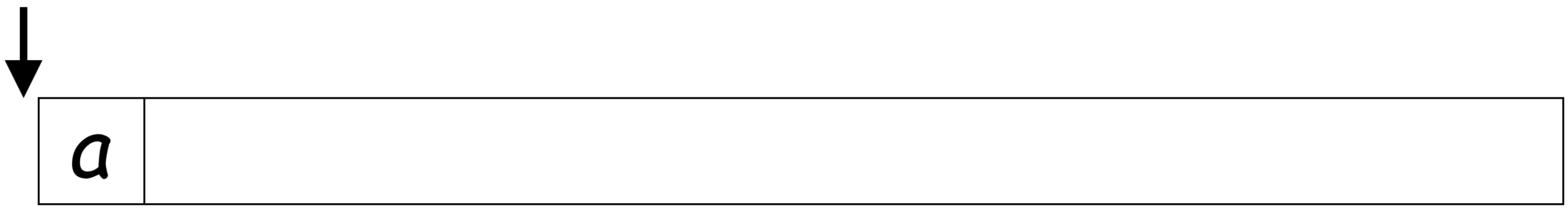
First Choice

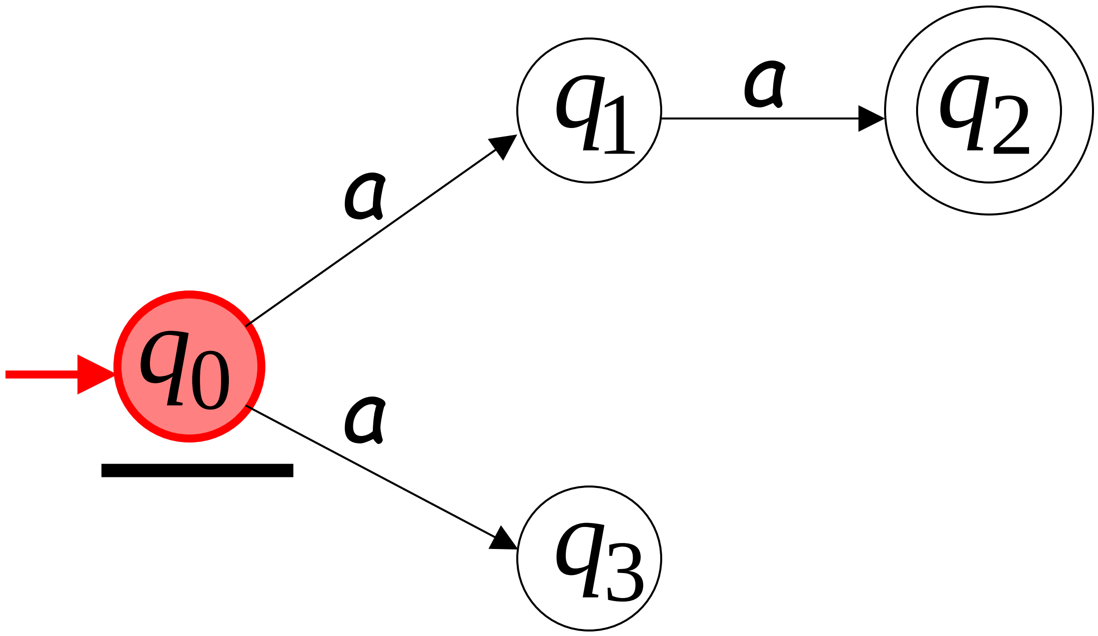


First Choice

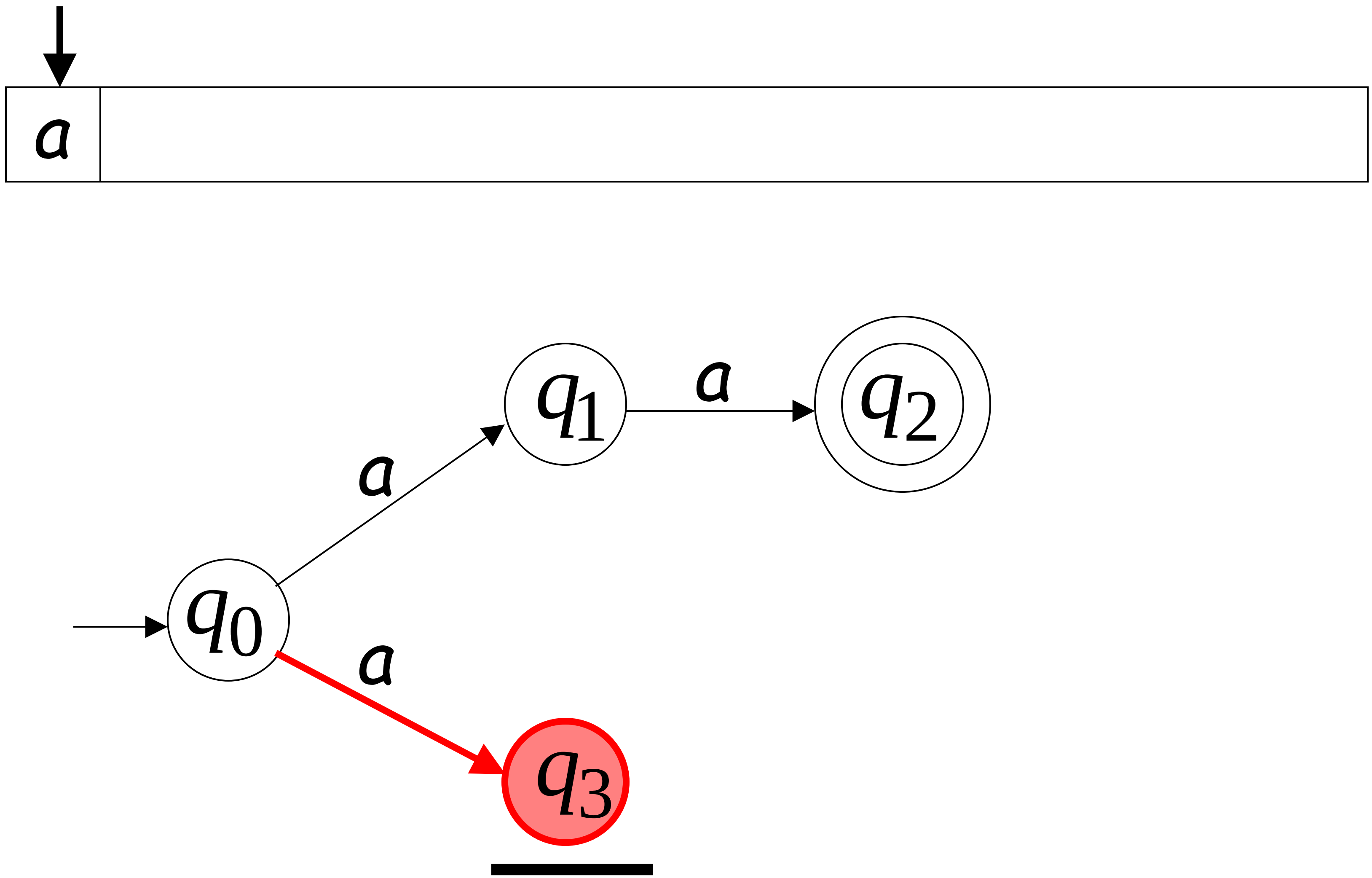


Second Choice

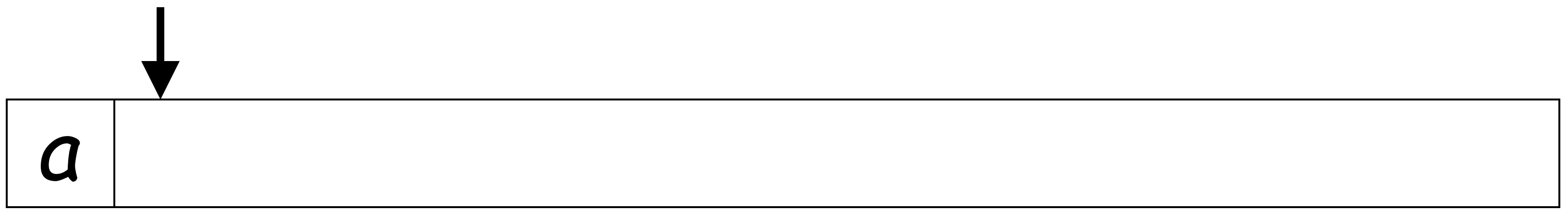


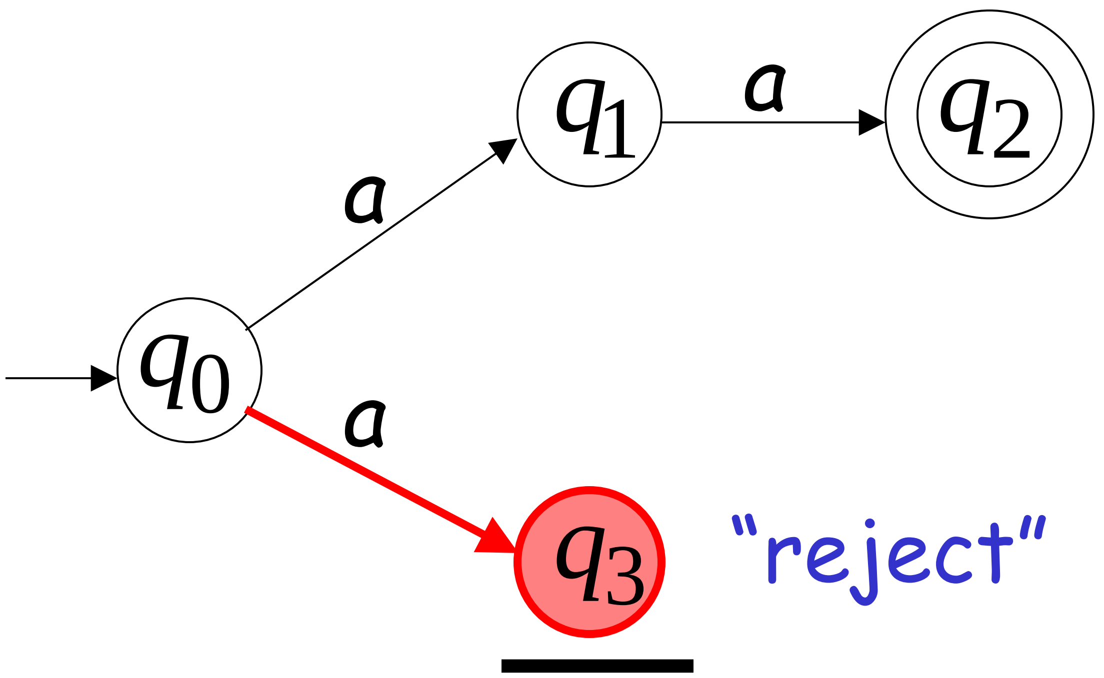


Second Choice



Second Choice





An NFA rejects a string:

when there is no computation of the NFA that accepts the string.

For each computation:

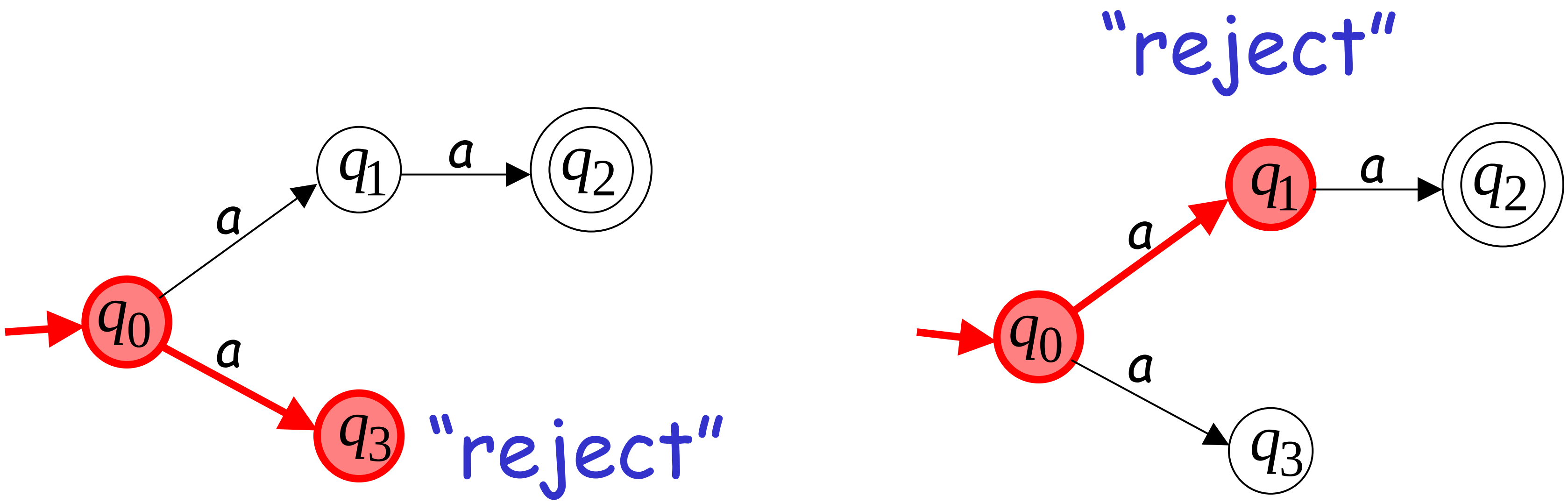
- All the input is consumed and the automaton is in a non final state

OR

- The input cannot be consumed

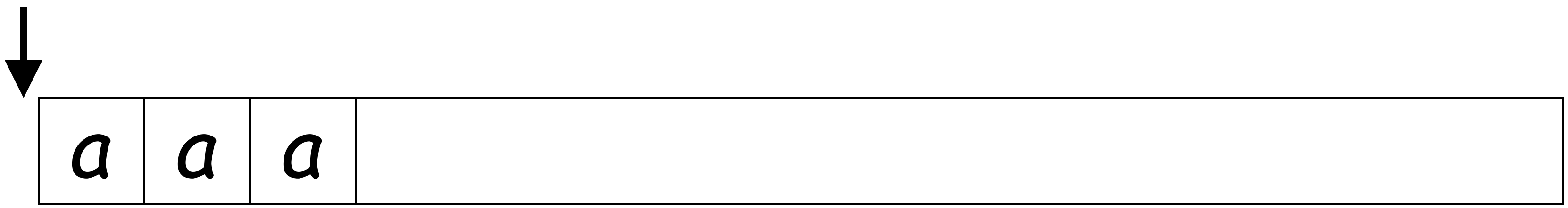
Example

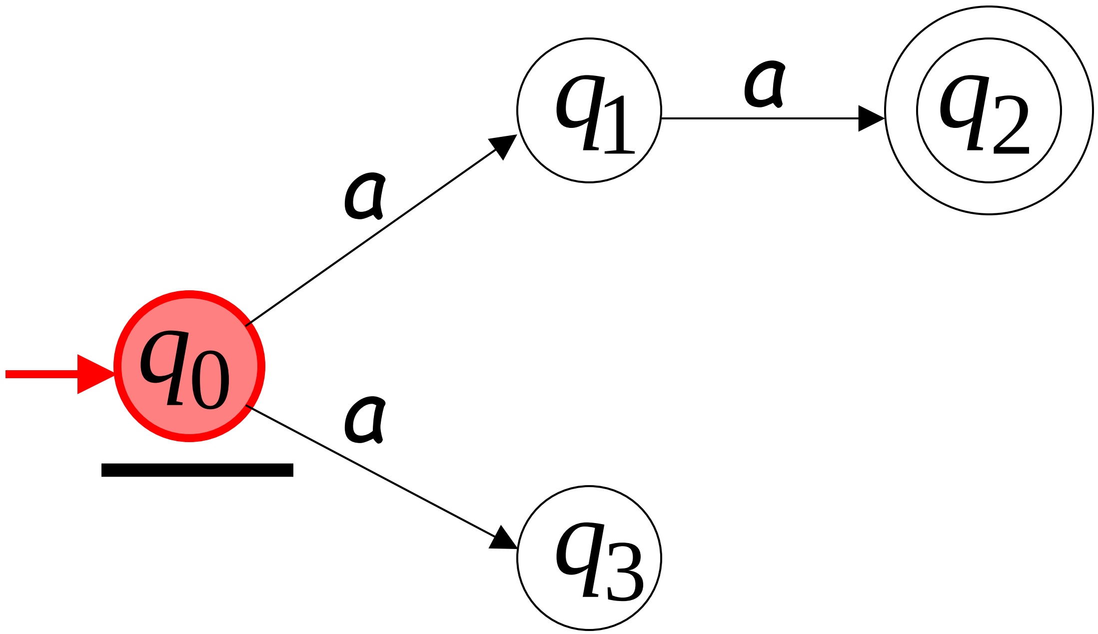
a is rejected by the NFA:



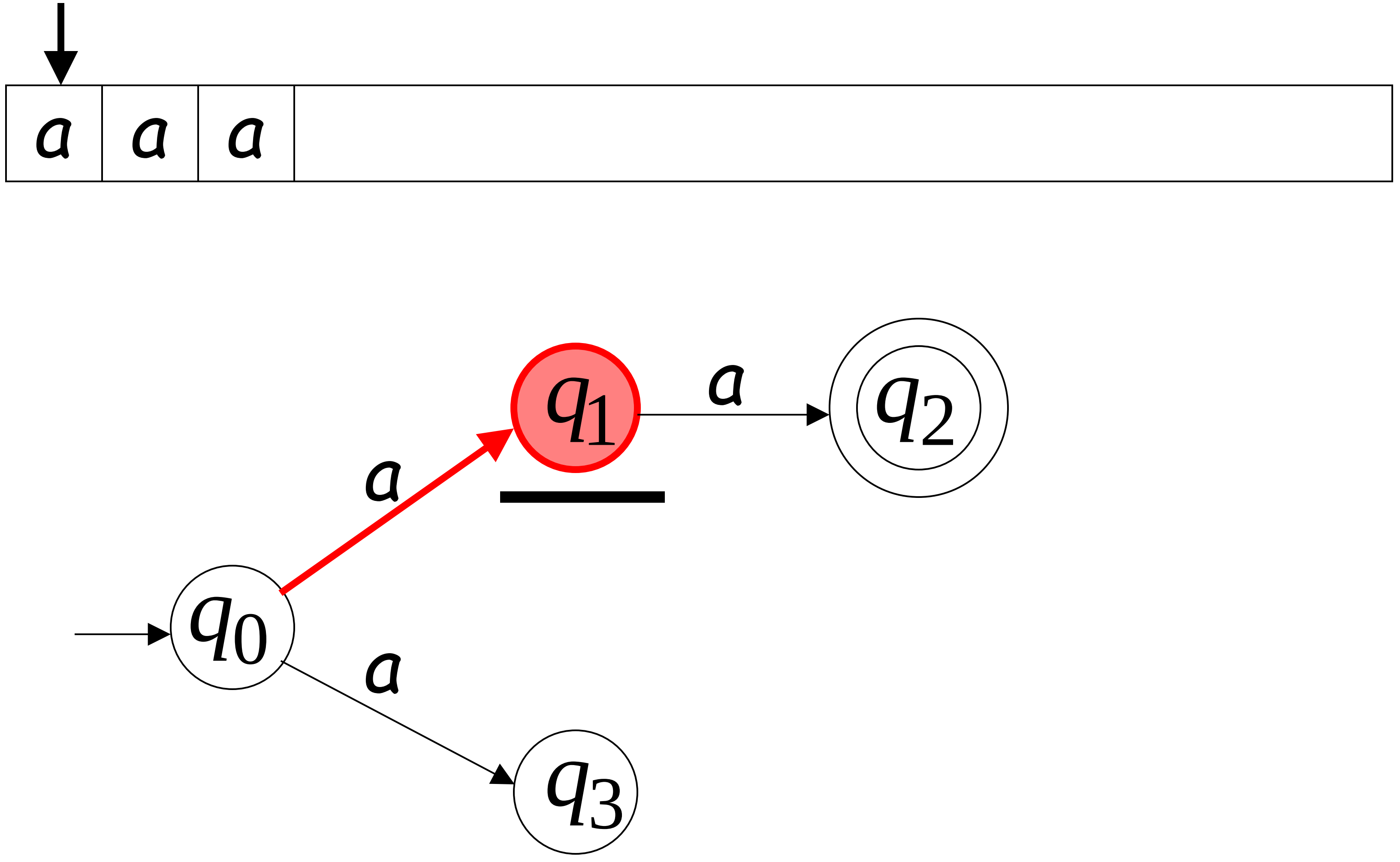
All possible computations lead to rejection

Rejection example

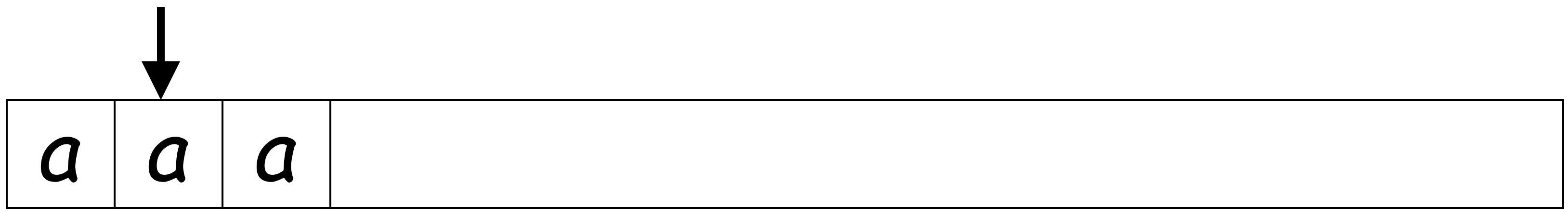


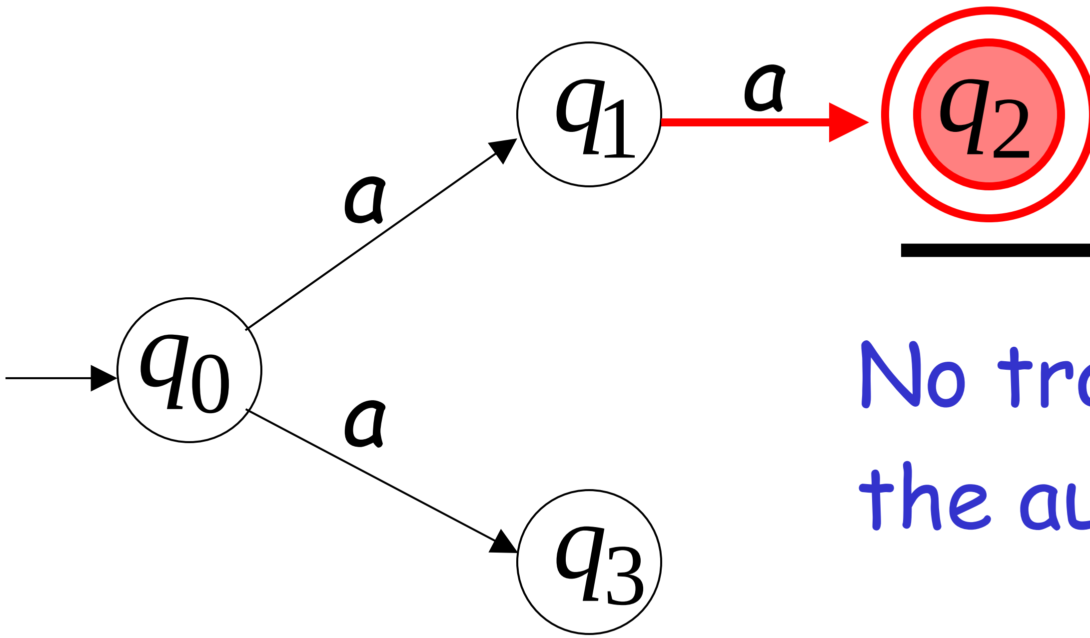


First Choice



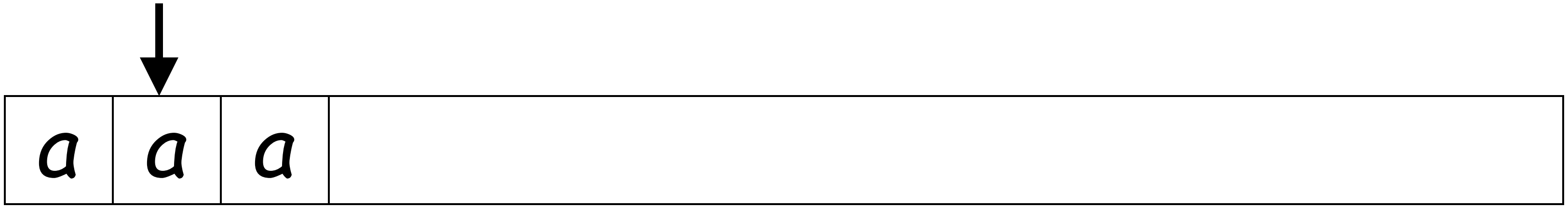
First Choice



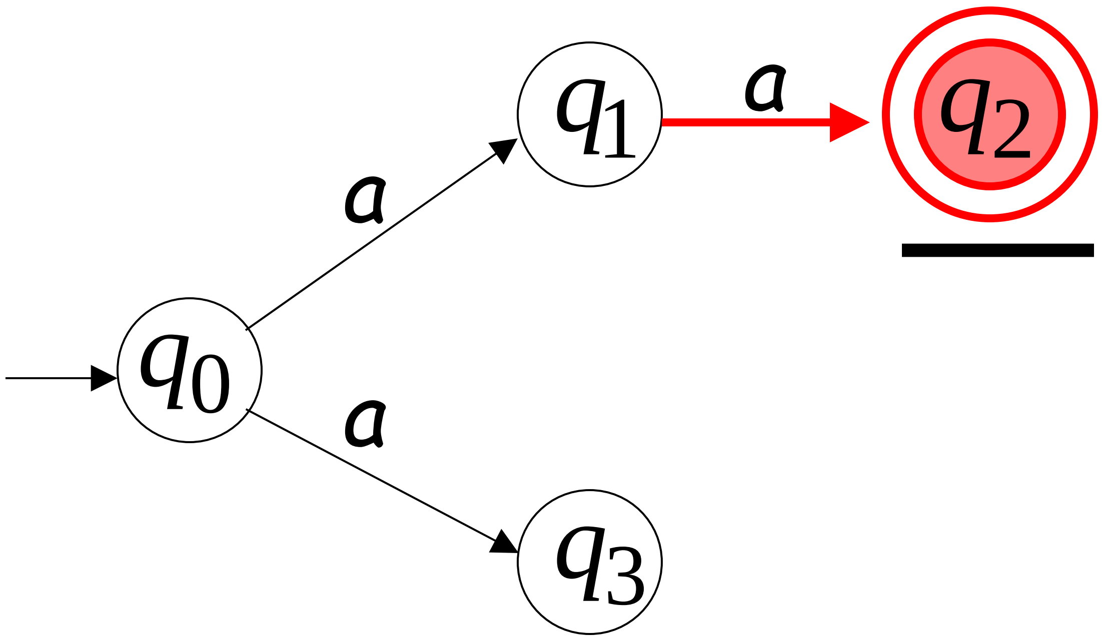


No transition:
the automaton hangs

First Choice

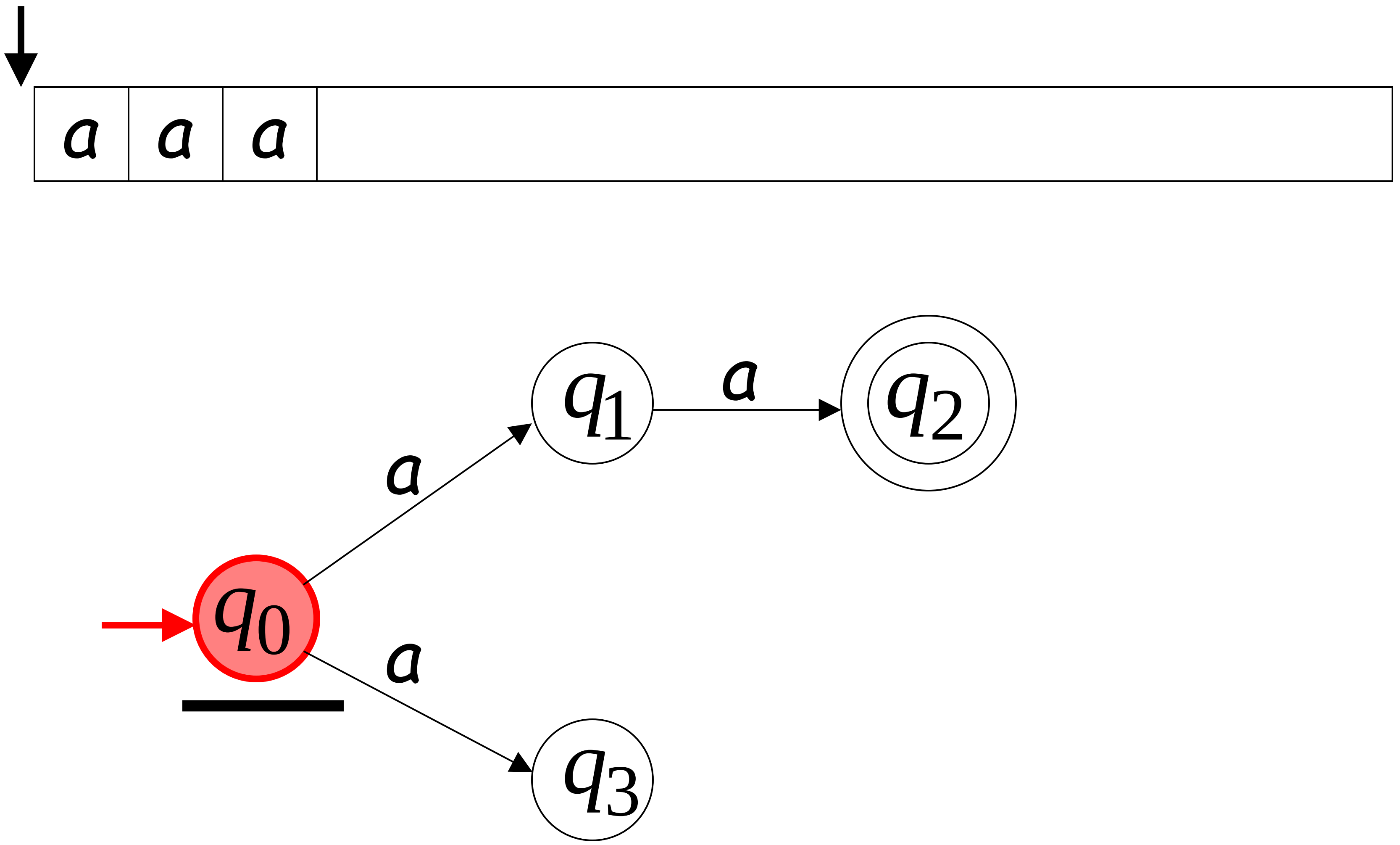


Input cannot be consumed

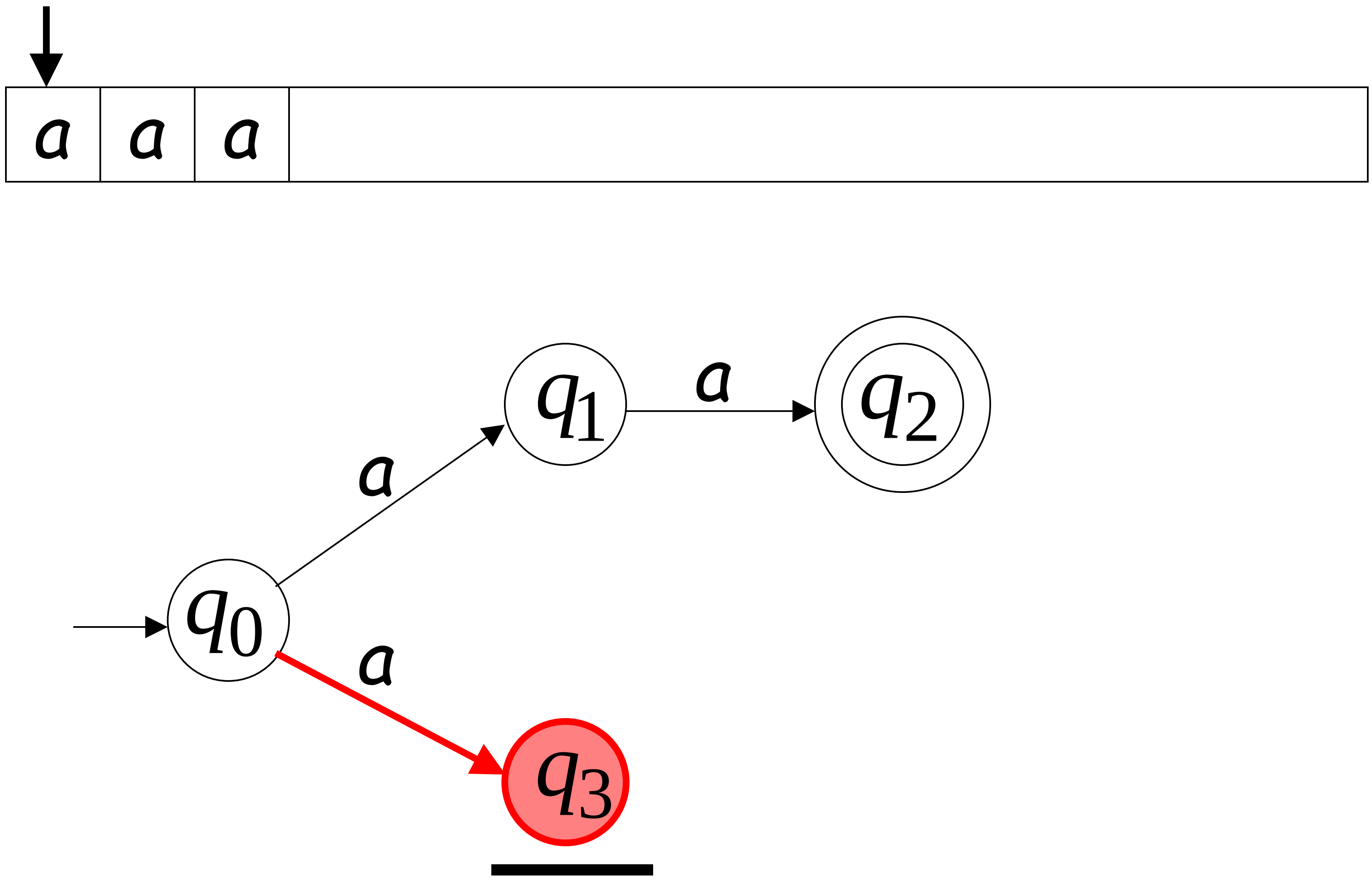


"reject"

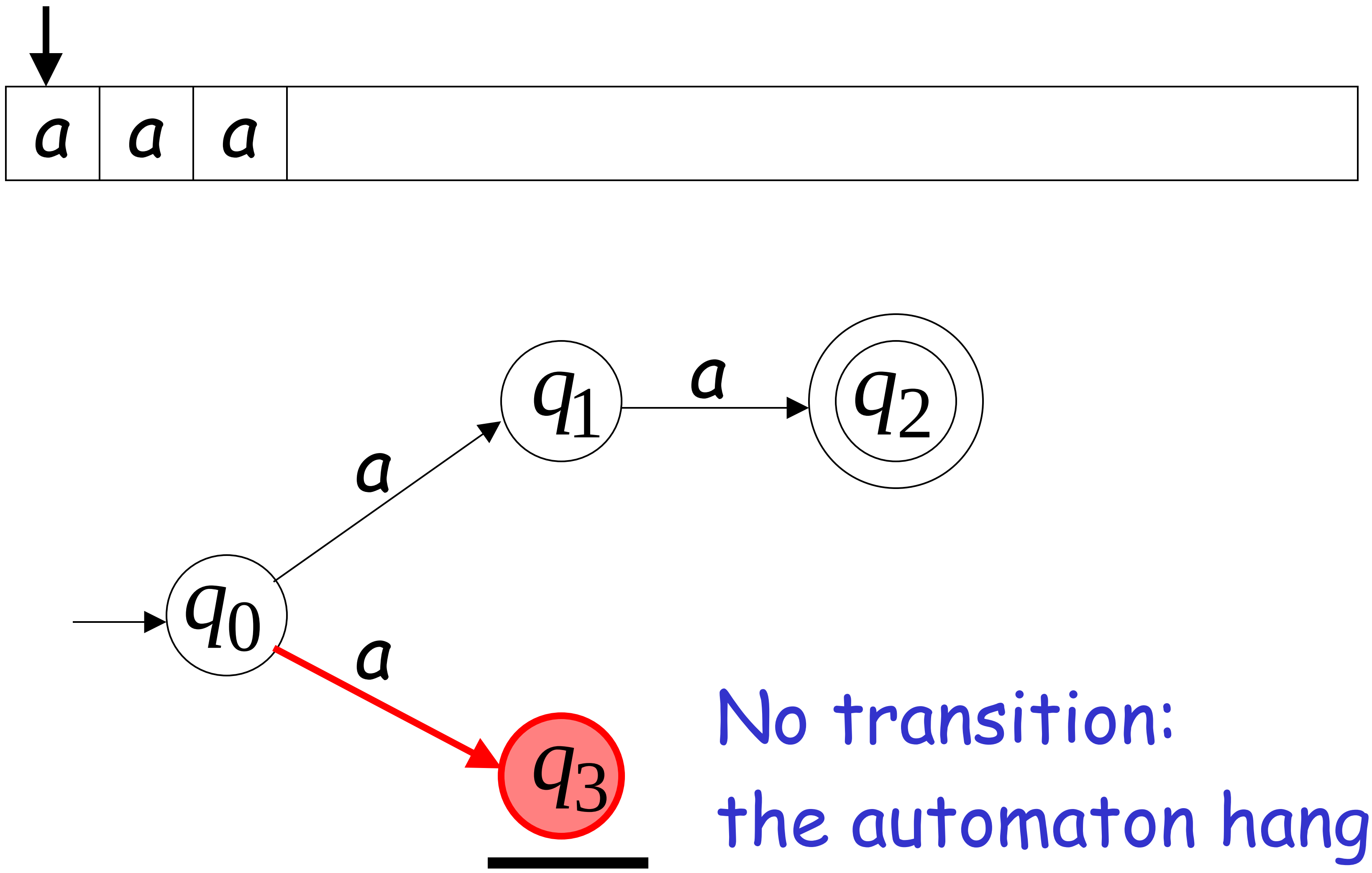
Second Choice



Second Choice

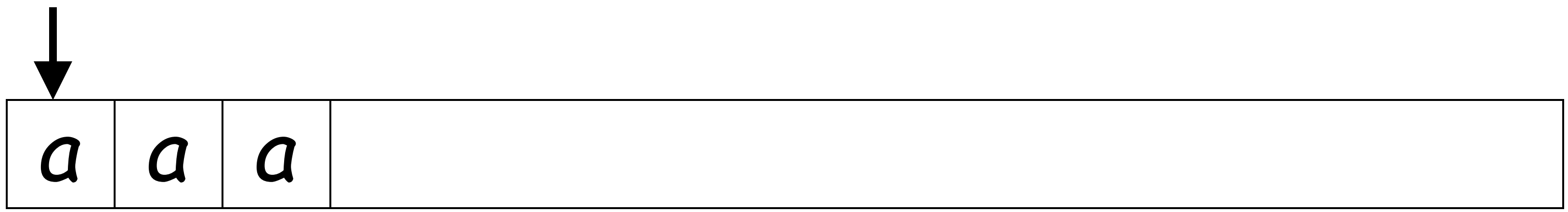


Second Choice

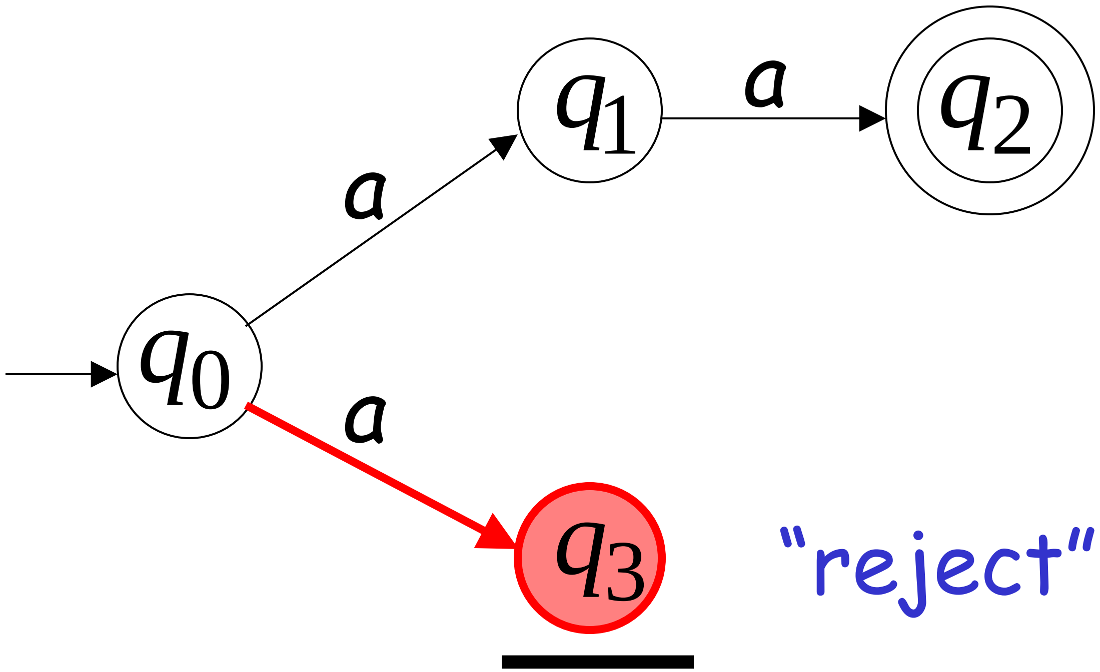


No transition:
the automaton hangs

Second Choice

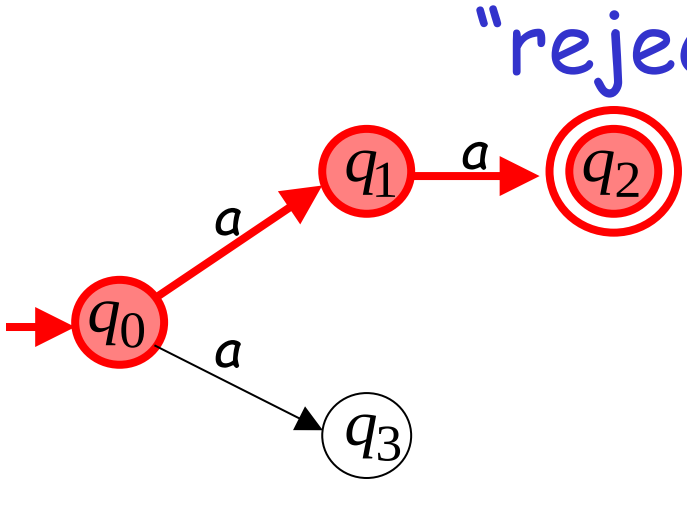


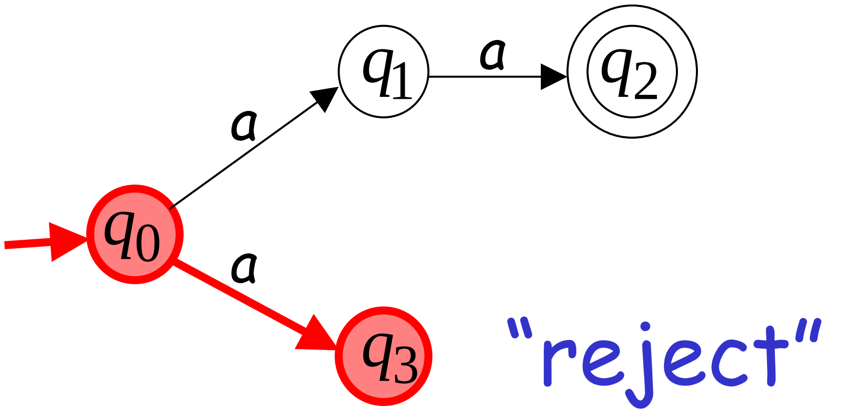
Input cannot be consumed



"reject"

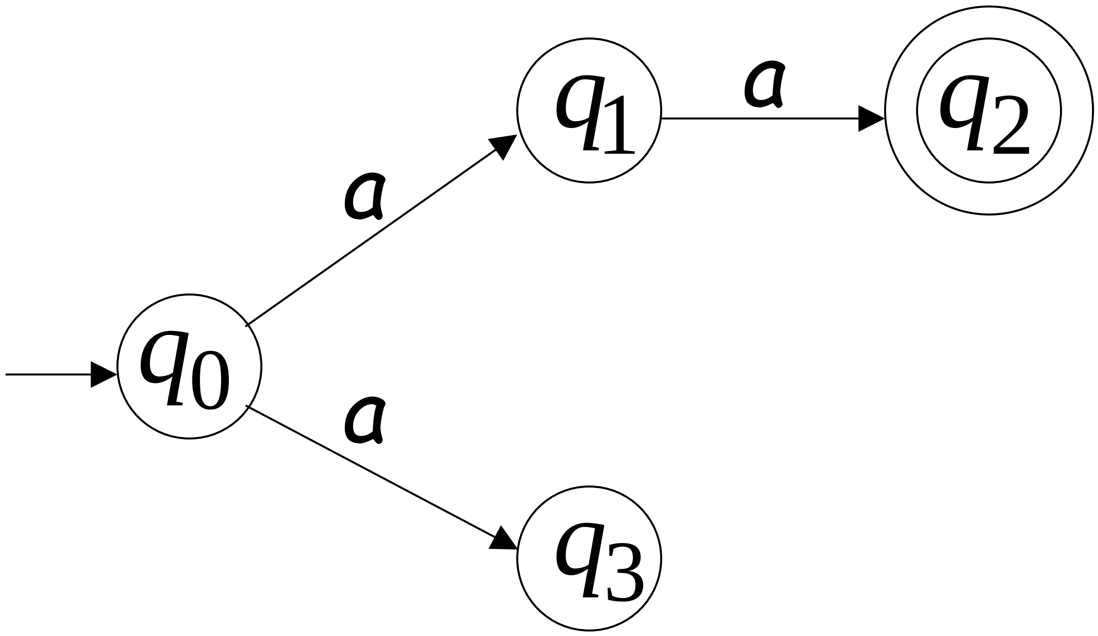
aaa is rejected by the NFA:



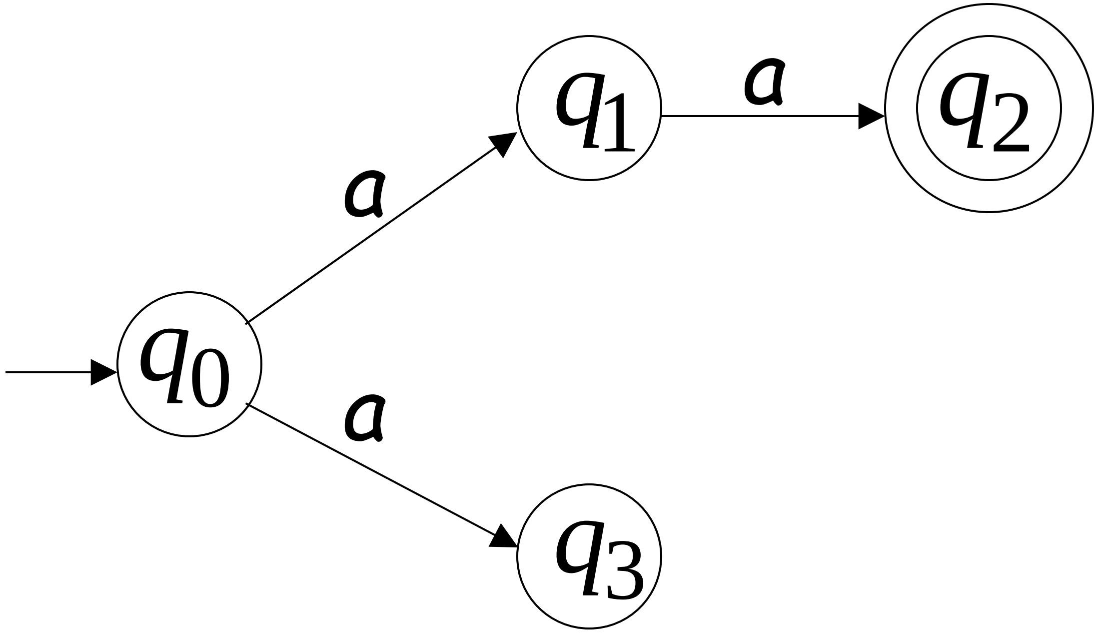


All possible computations lead to rejection

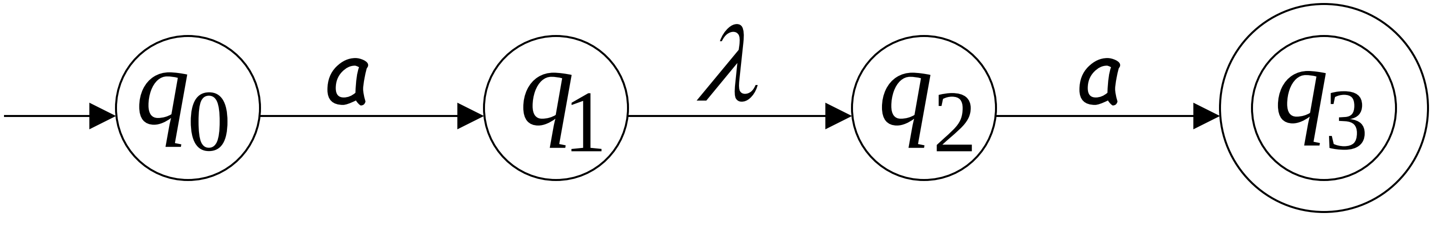
L(M)?

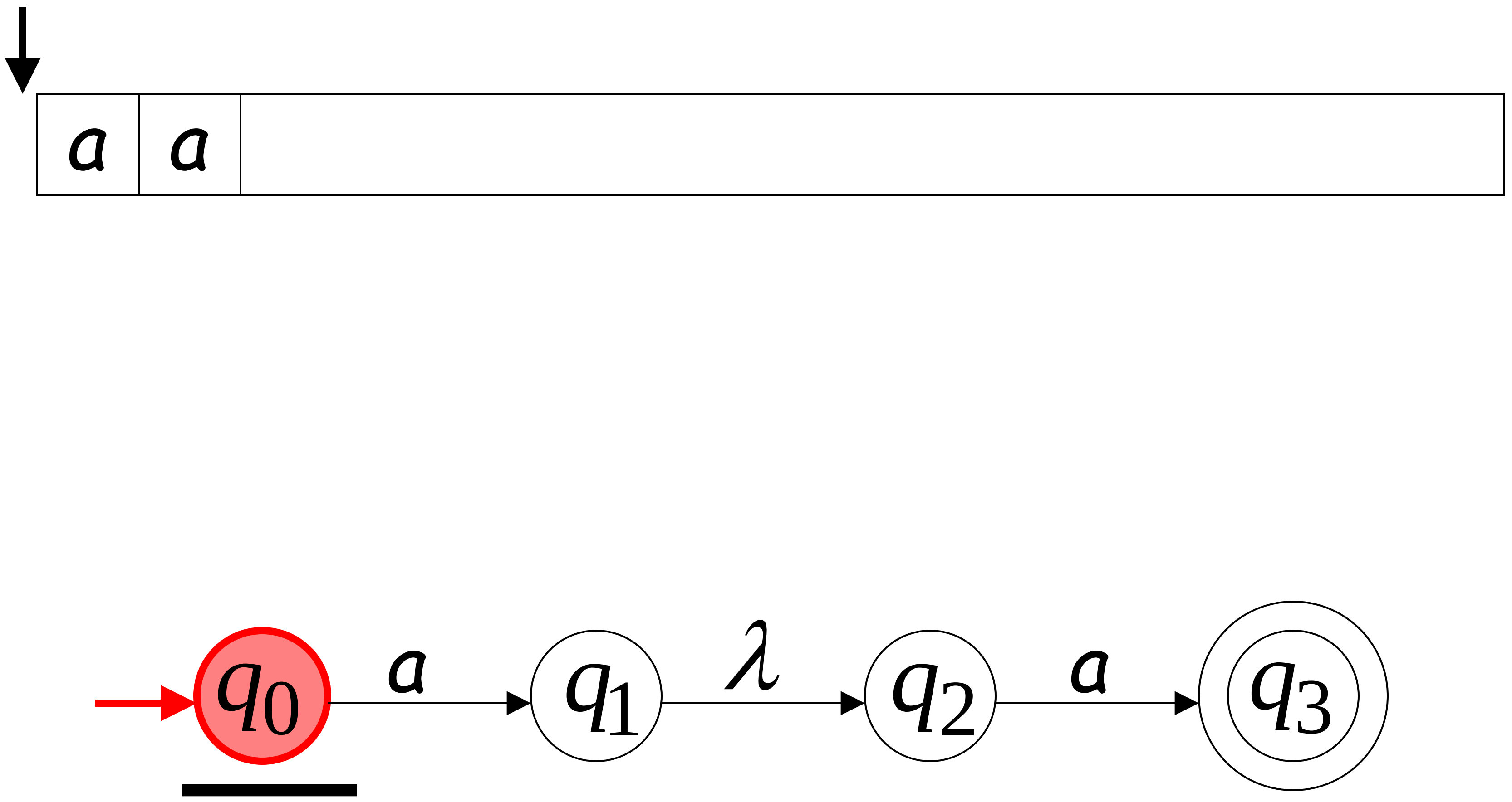


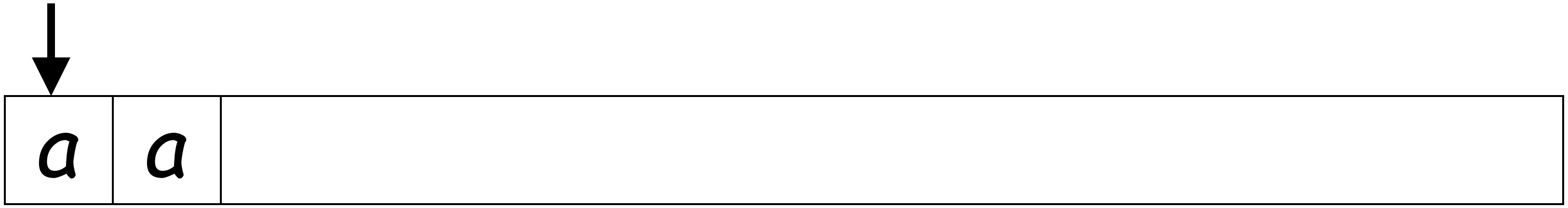
Language accepted: $L = \{aa\}$

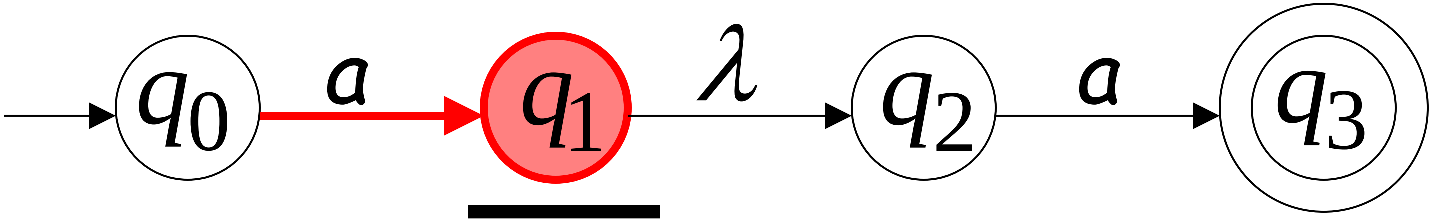


Lambda Transitions

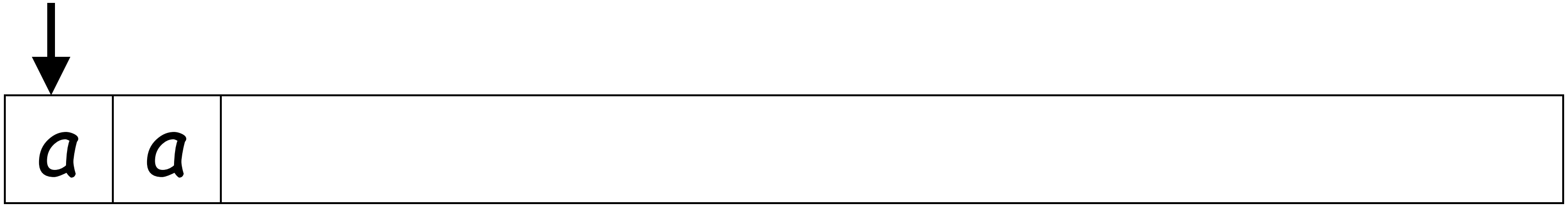


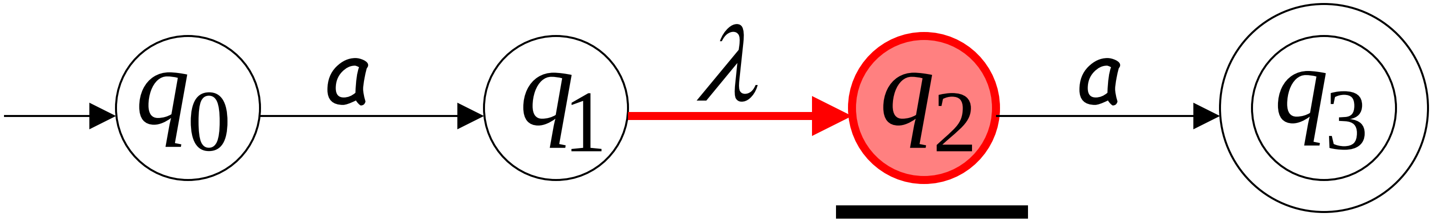

a
a
q0
a
q1
λ
q2
a
q3

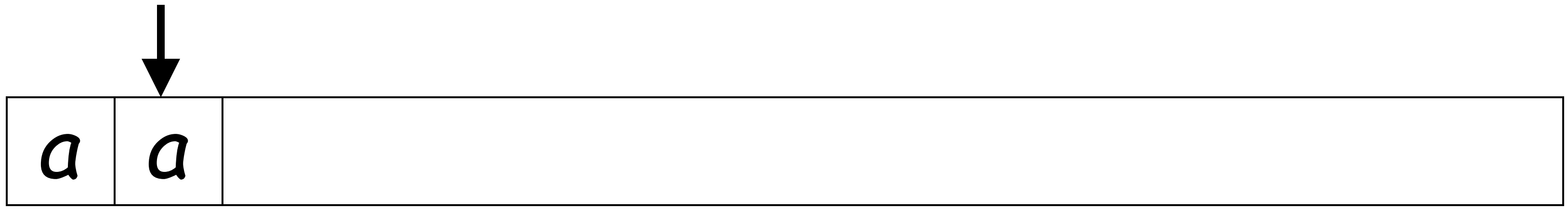

a
a

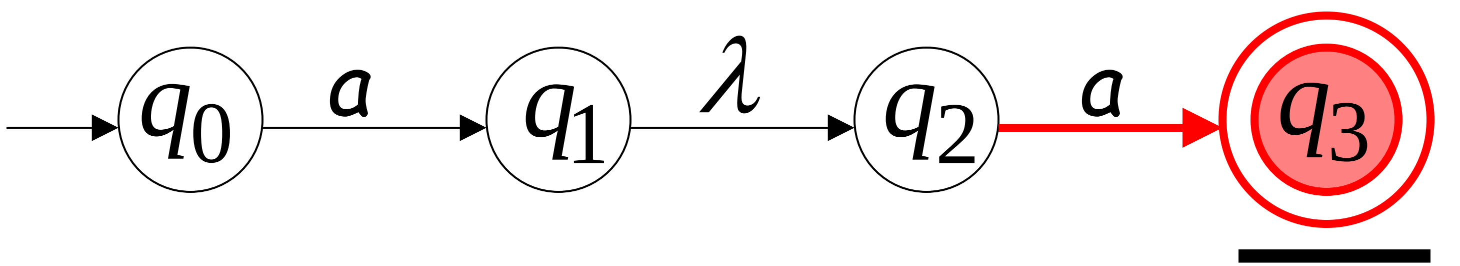

q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

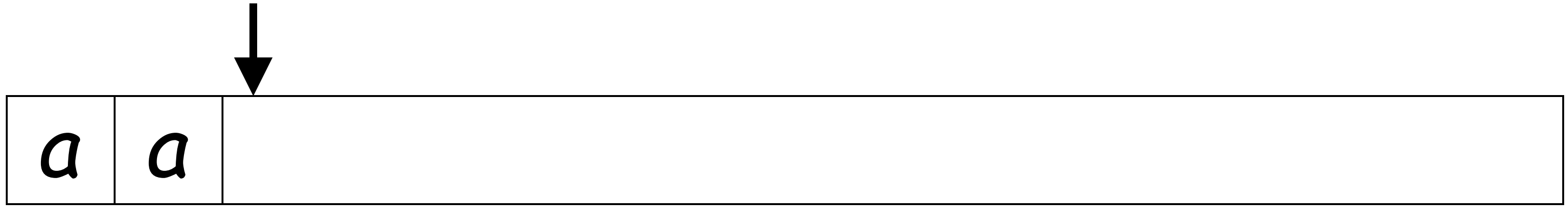




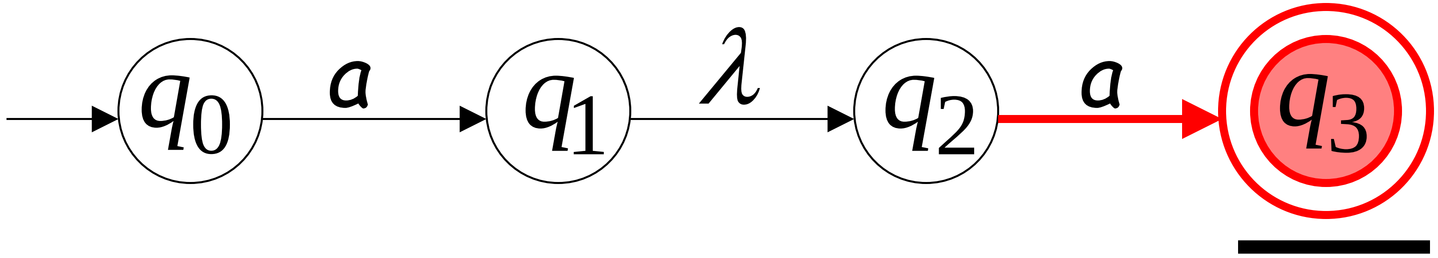

a
a


q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

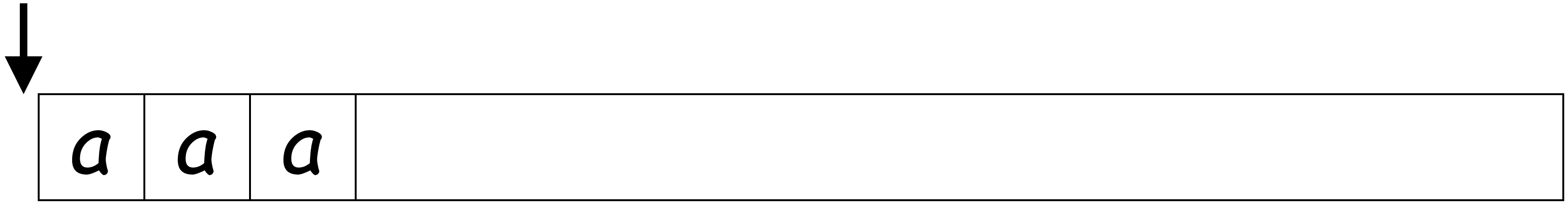


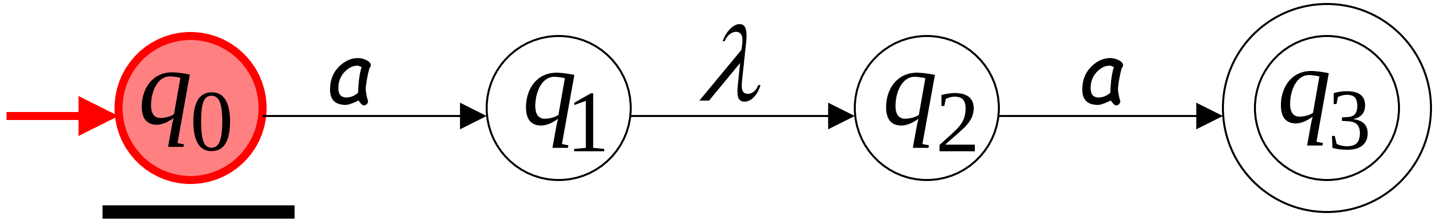
"accept"

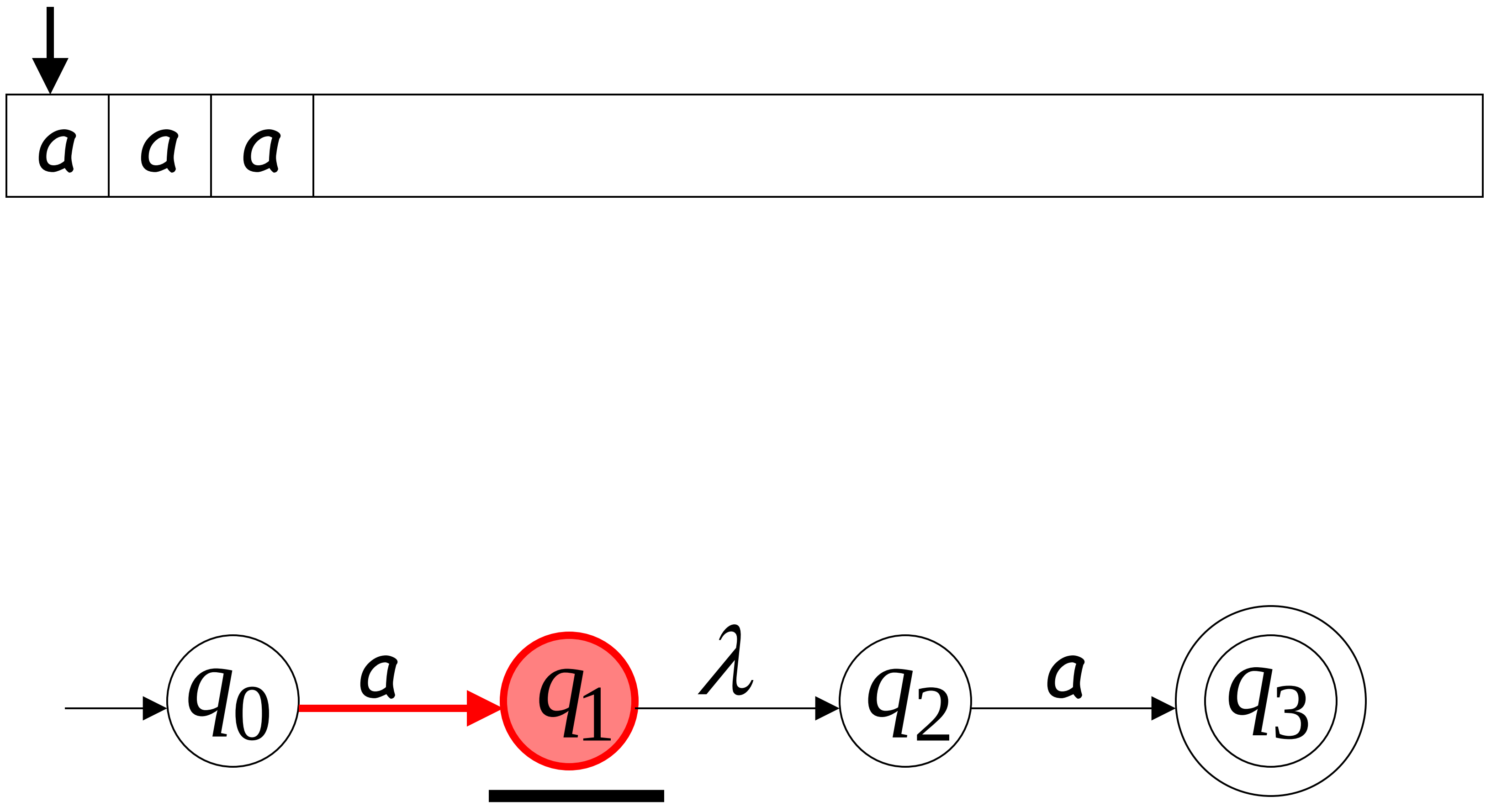


String aa is accepted

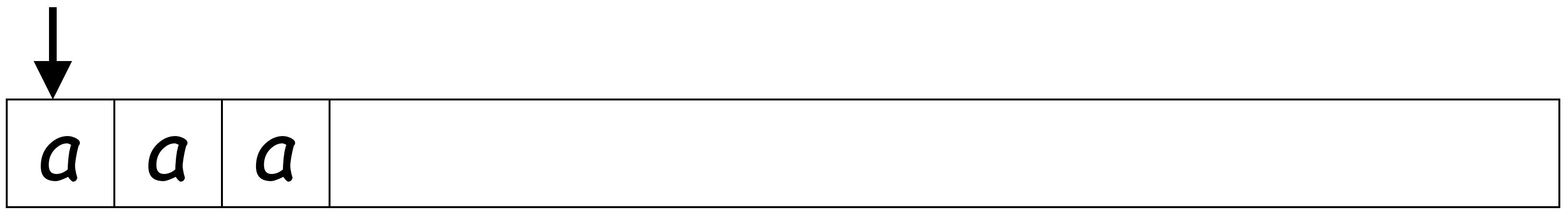
Rejection Example

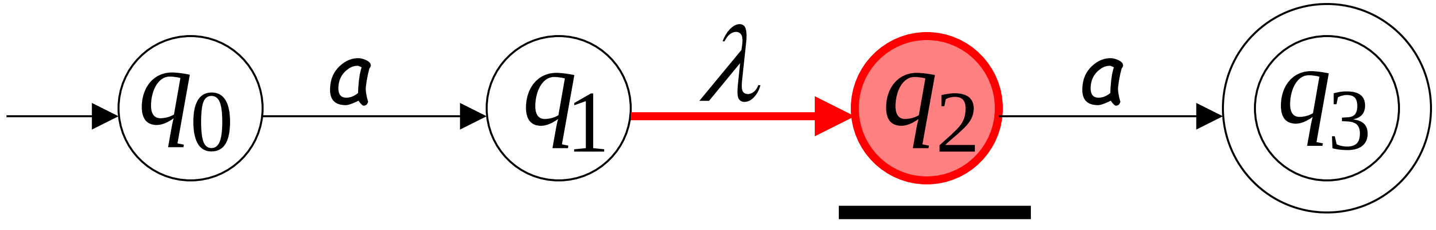


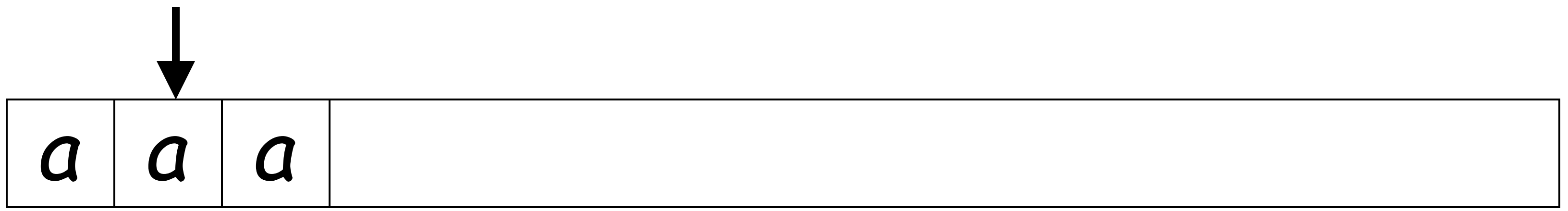


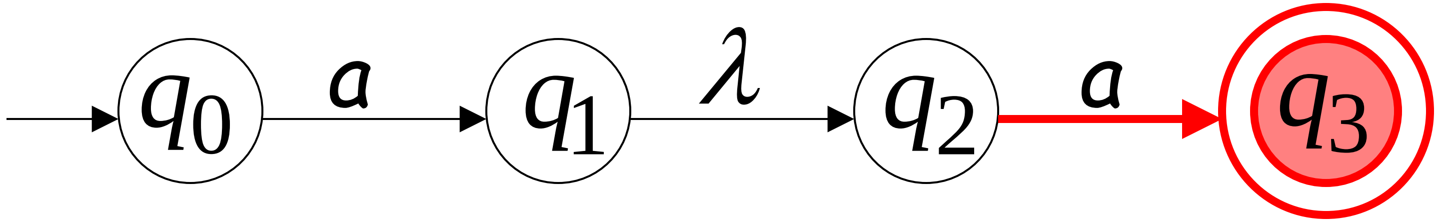

a
a
a
a
q_0
q_1
λ
q_2
a
q_3

(read head doesn't move)



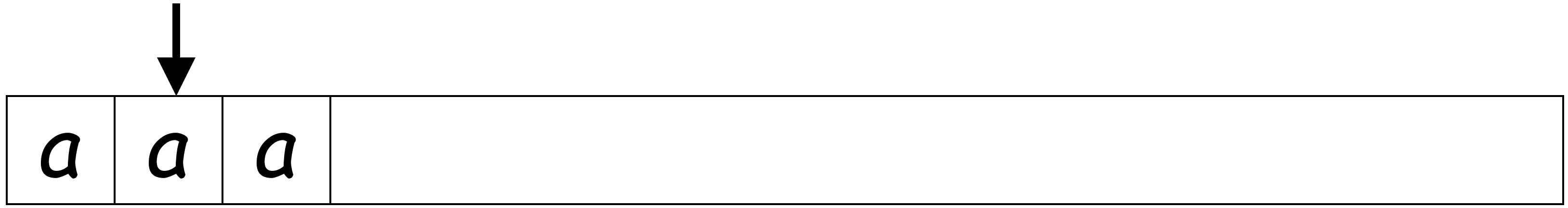


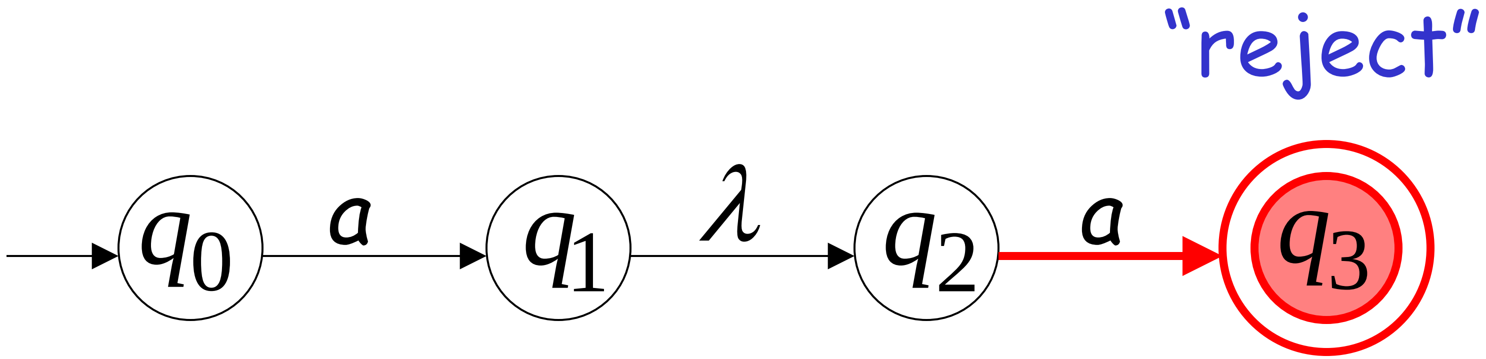




No transition:
the automaton hangs

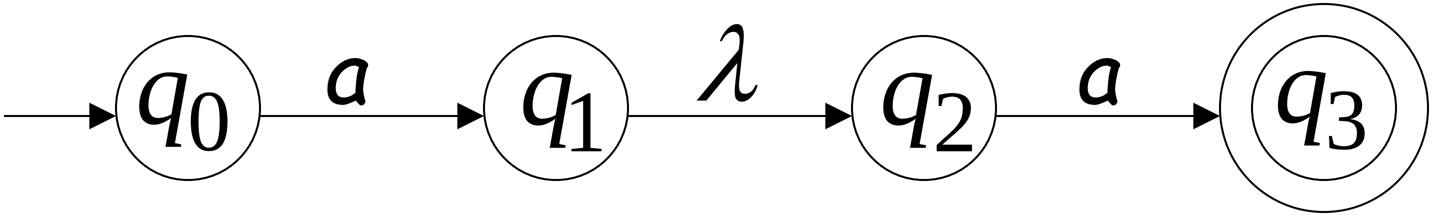
Input cannot be consumed



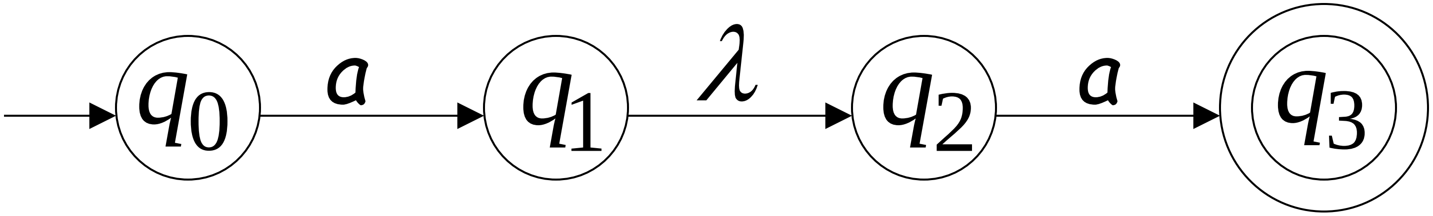


String aaa is rejected

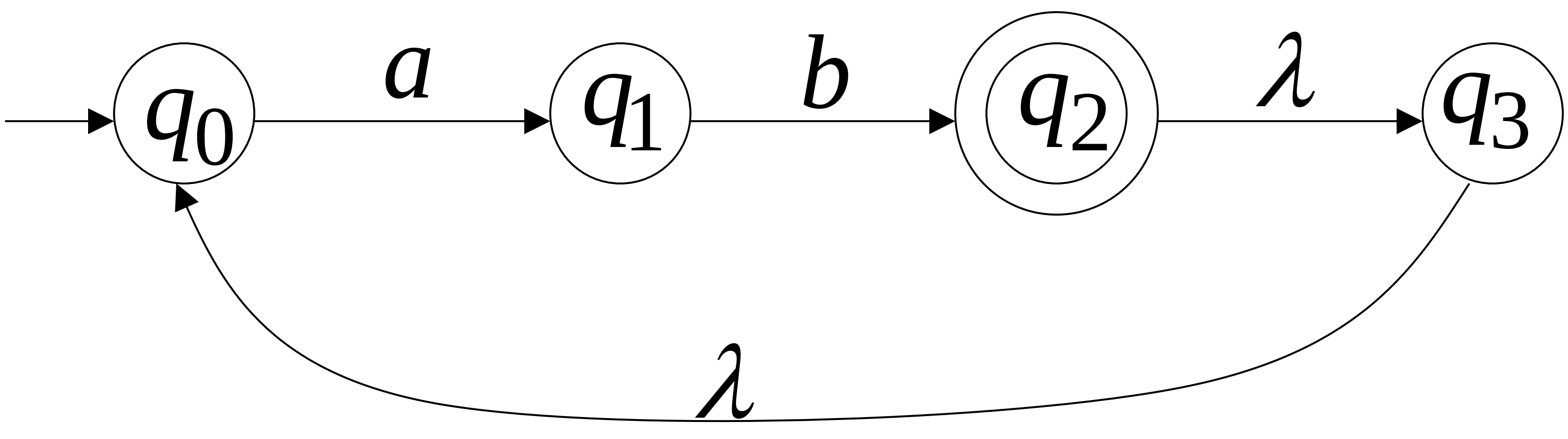
L(M)?

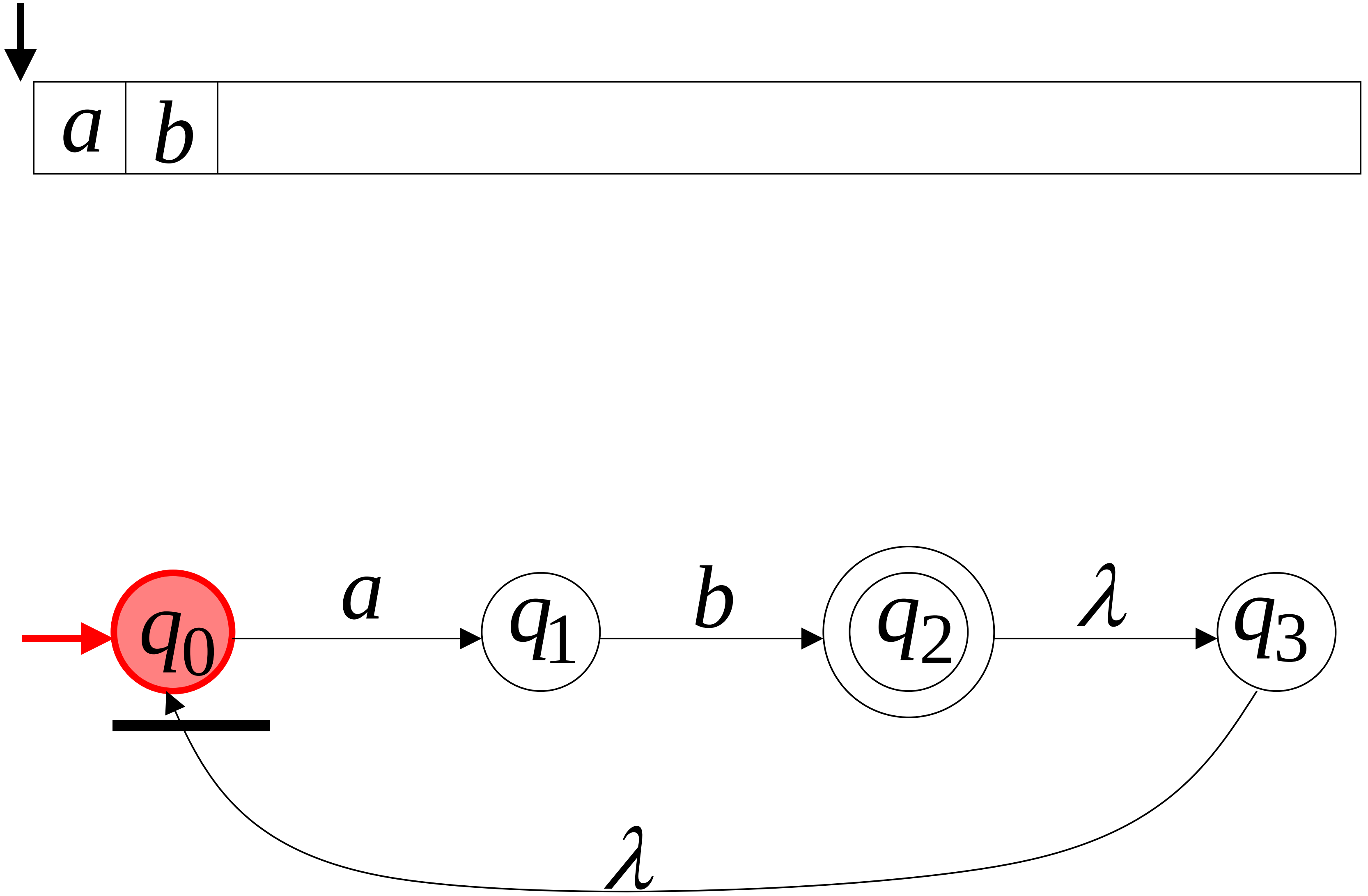


Language accepted: $L = \{aa\}$

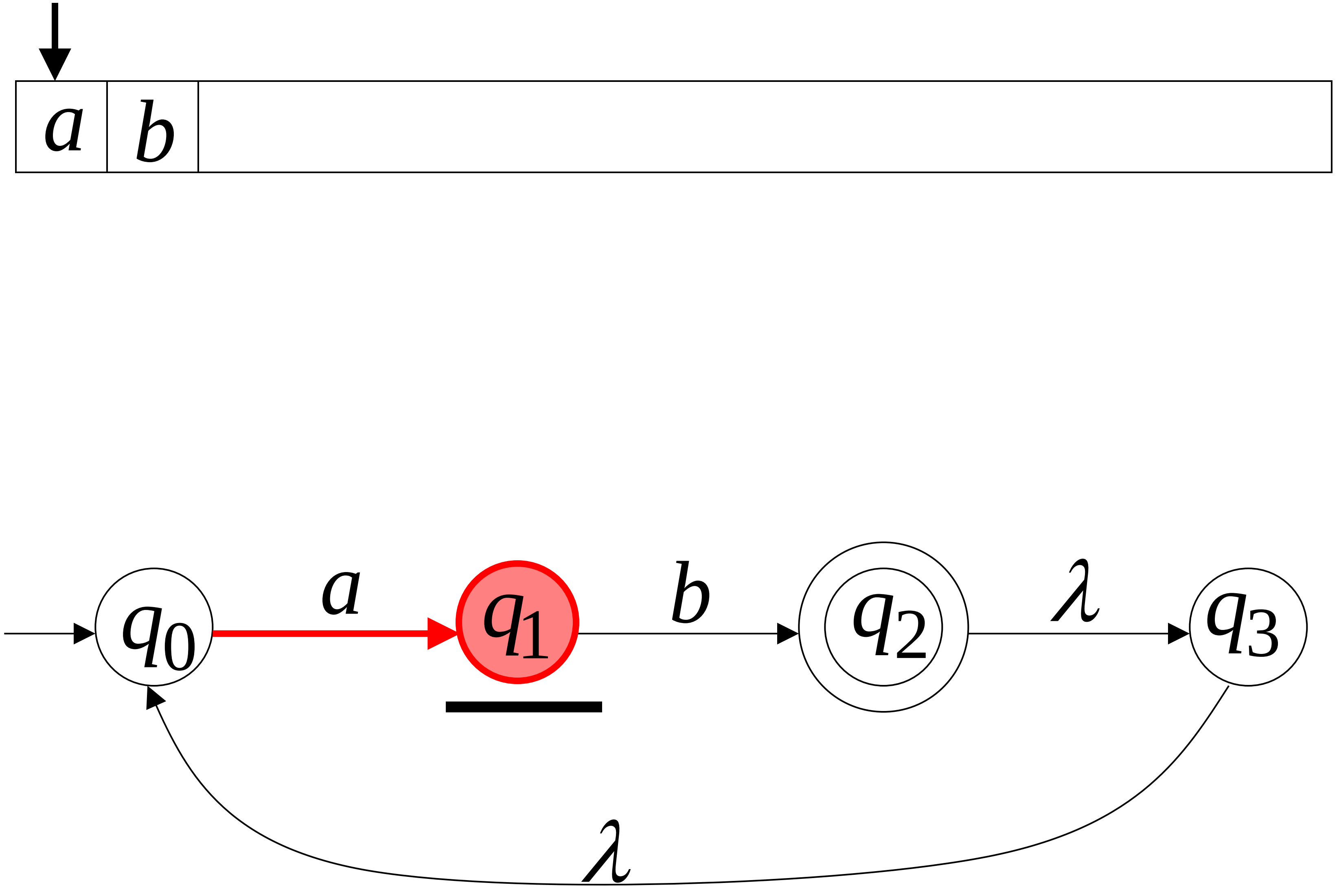


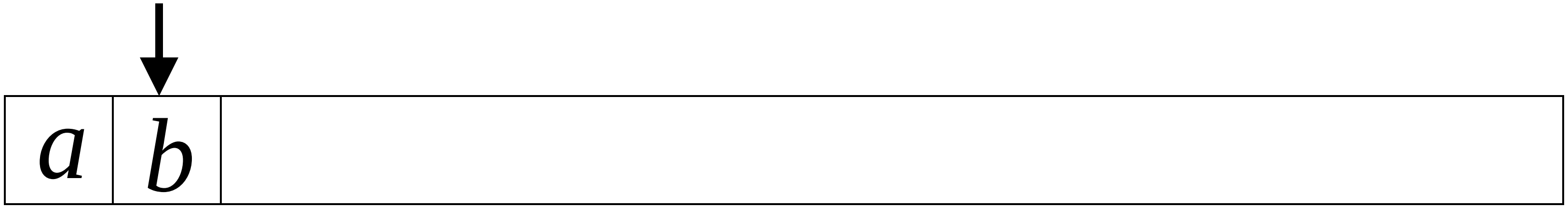
Another NFA Example: L(M)?

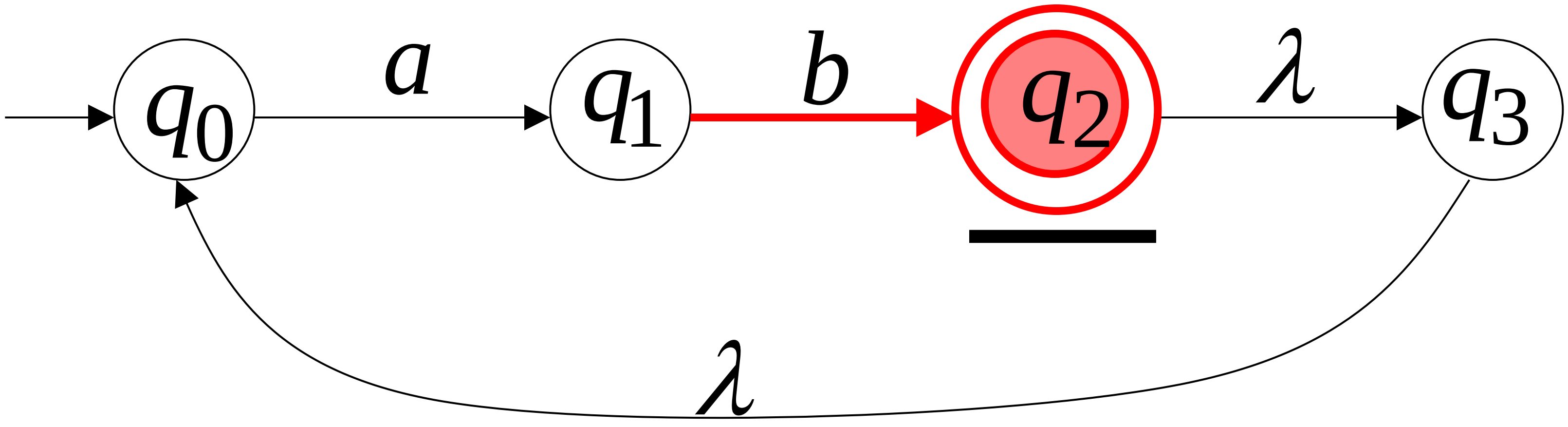


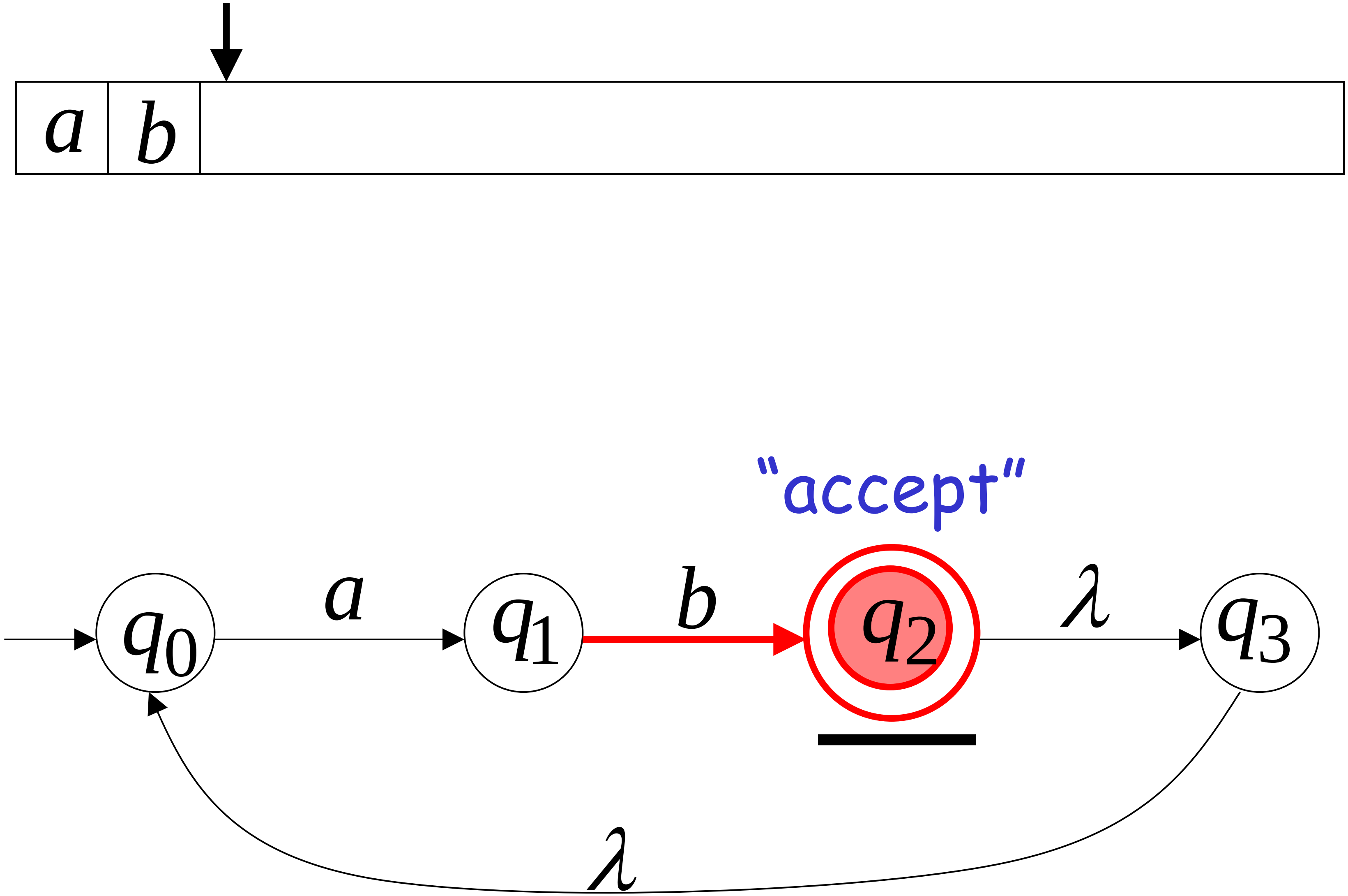


a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

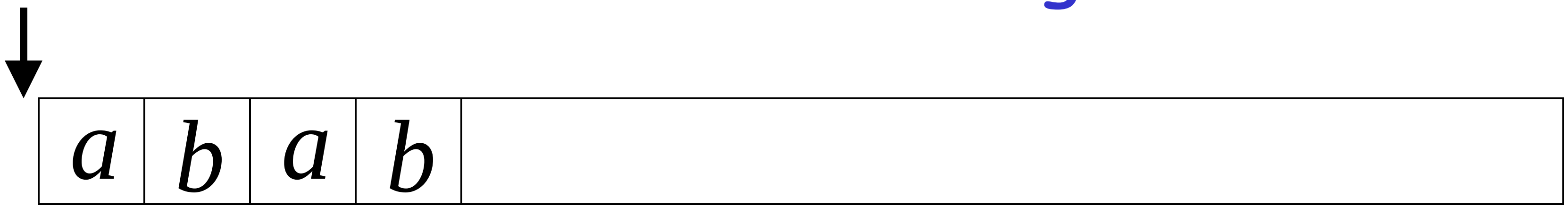

a
b
a
b
λ
λ
q_0
q_1
q_2
q_3

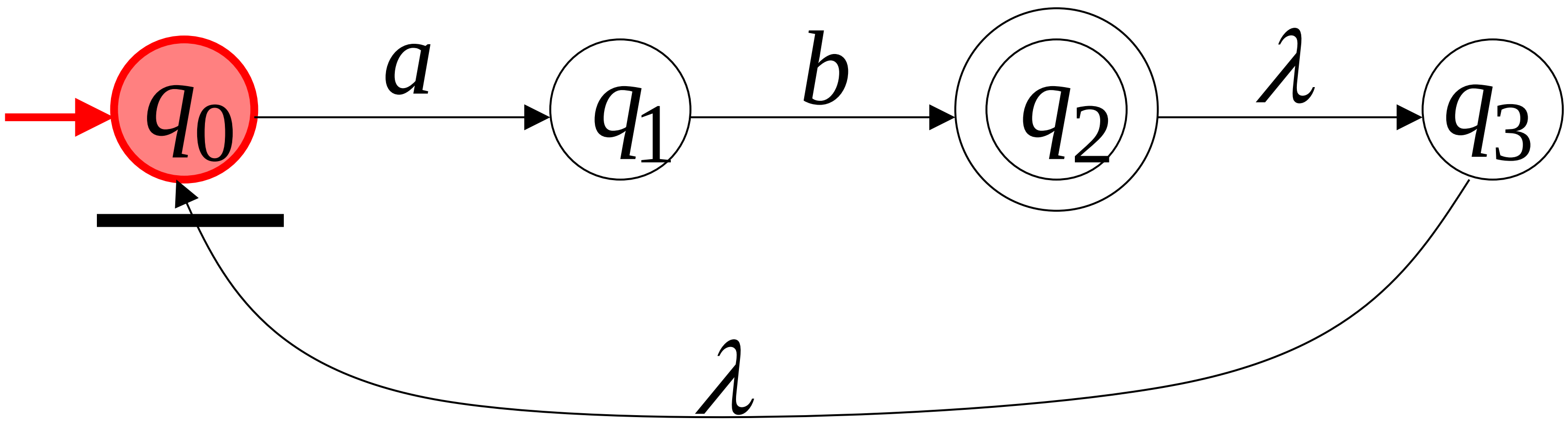

a
b

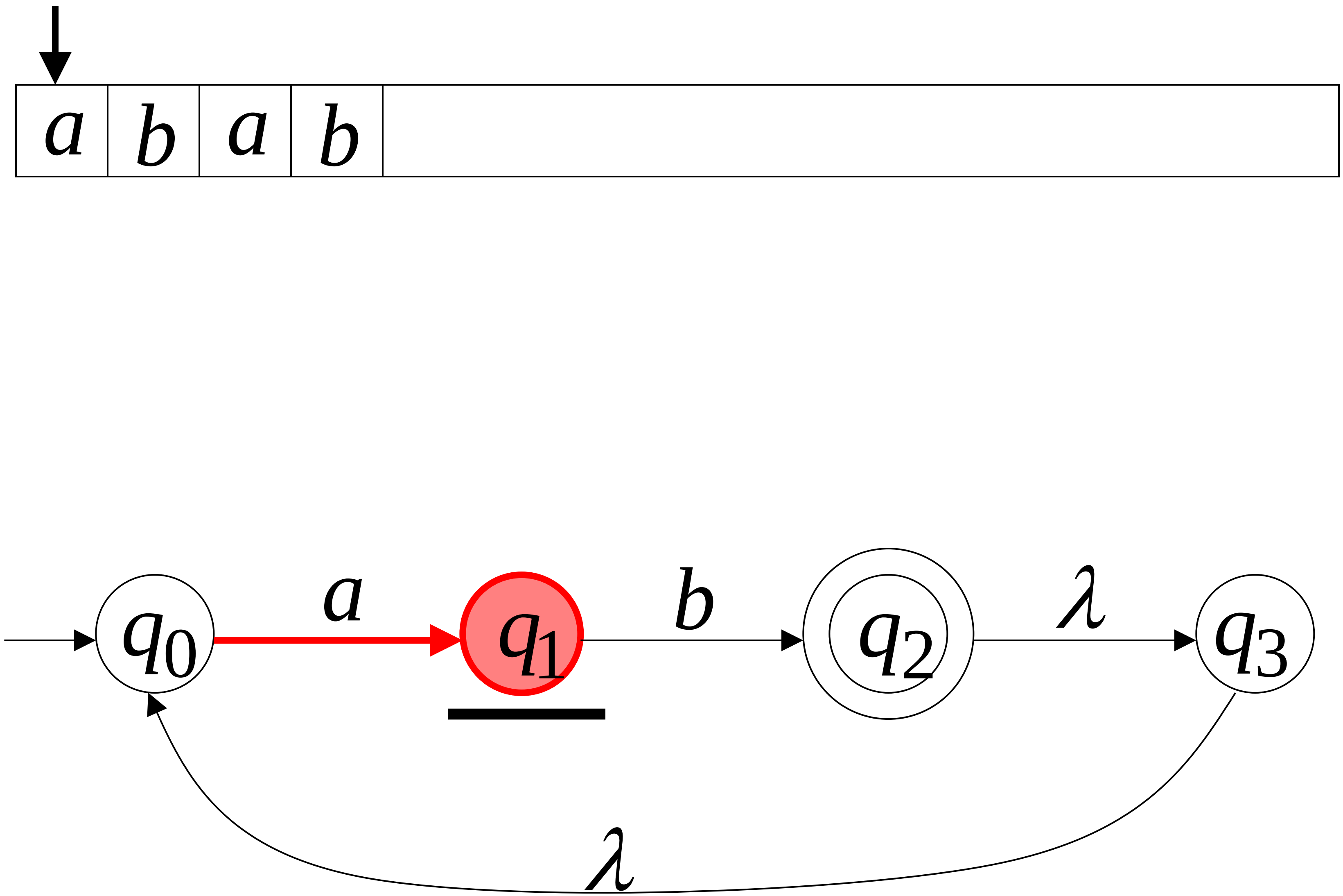

a
b
λ
λ
q0
q1
q2
q3

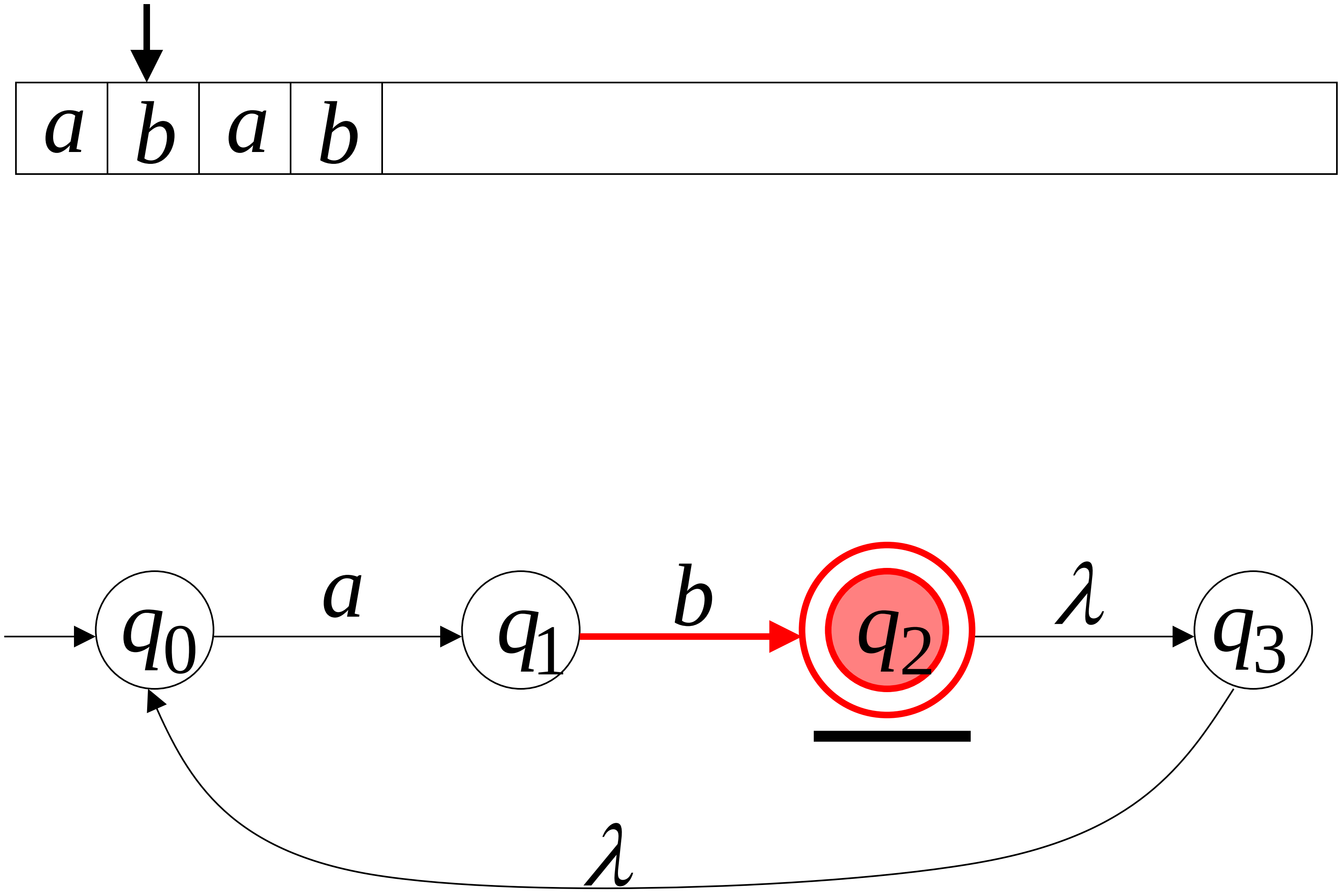

a
b
"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

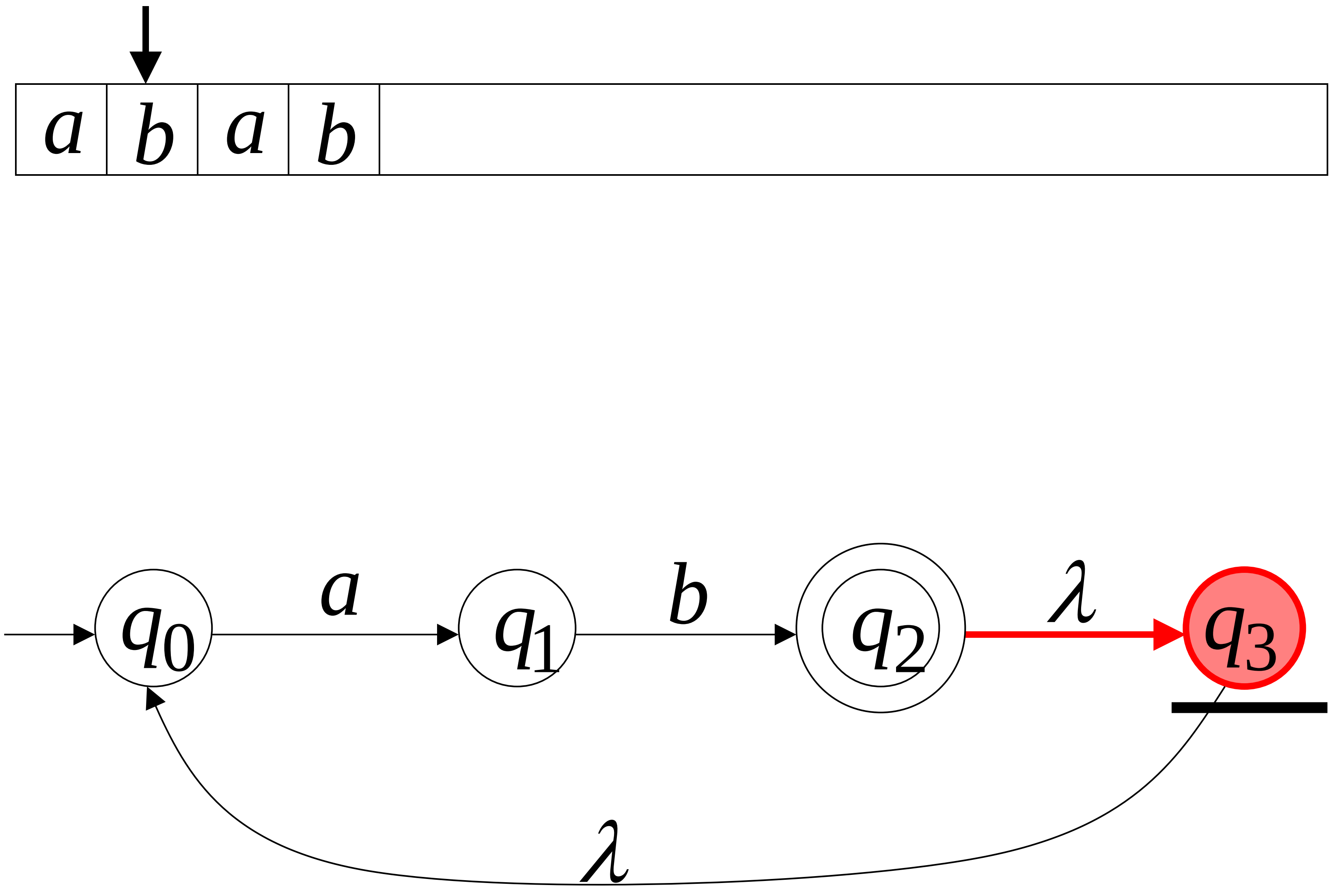
Another String

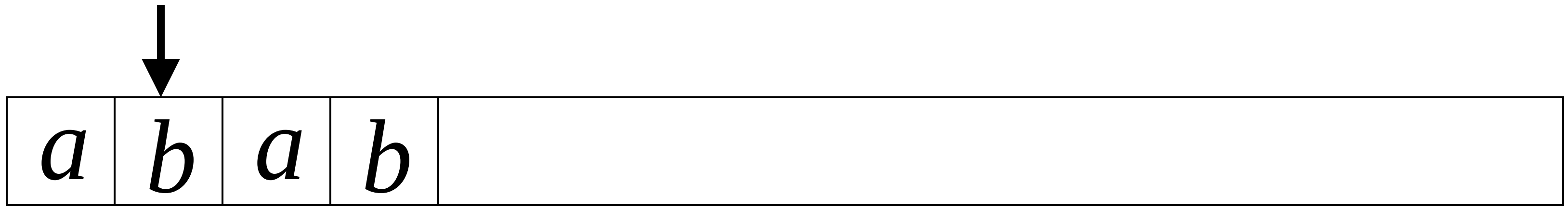




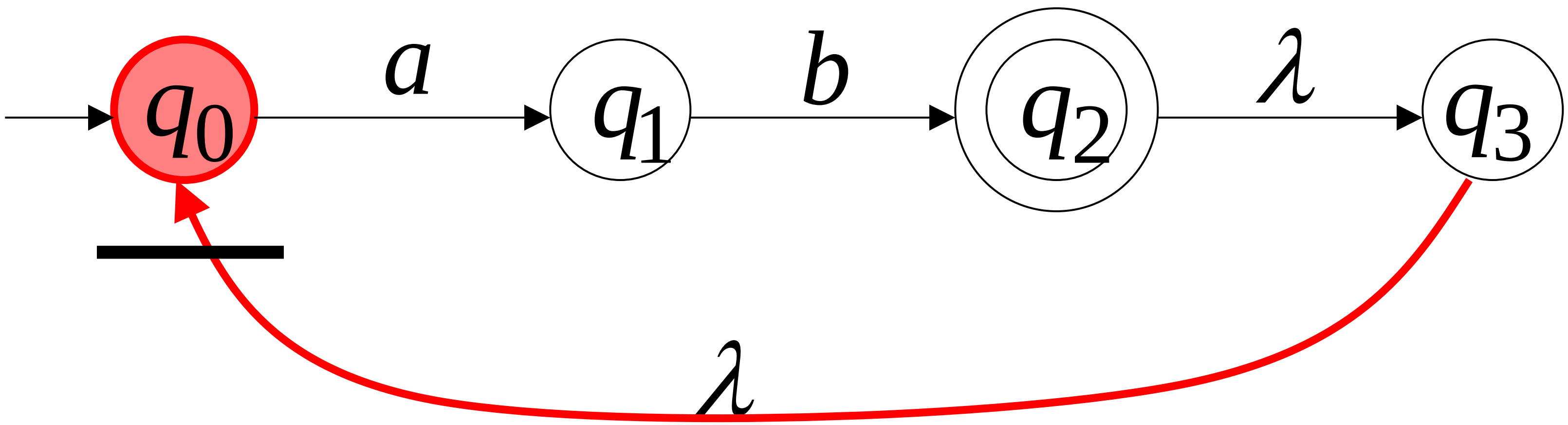

a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3


a b a b
a
b
λ
λ
q_0
q_1
q_2
q_3

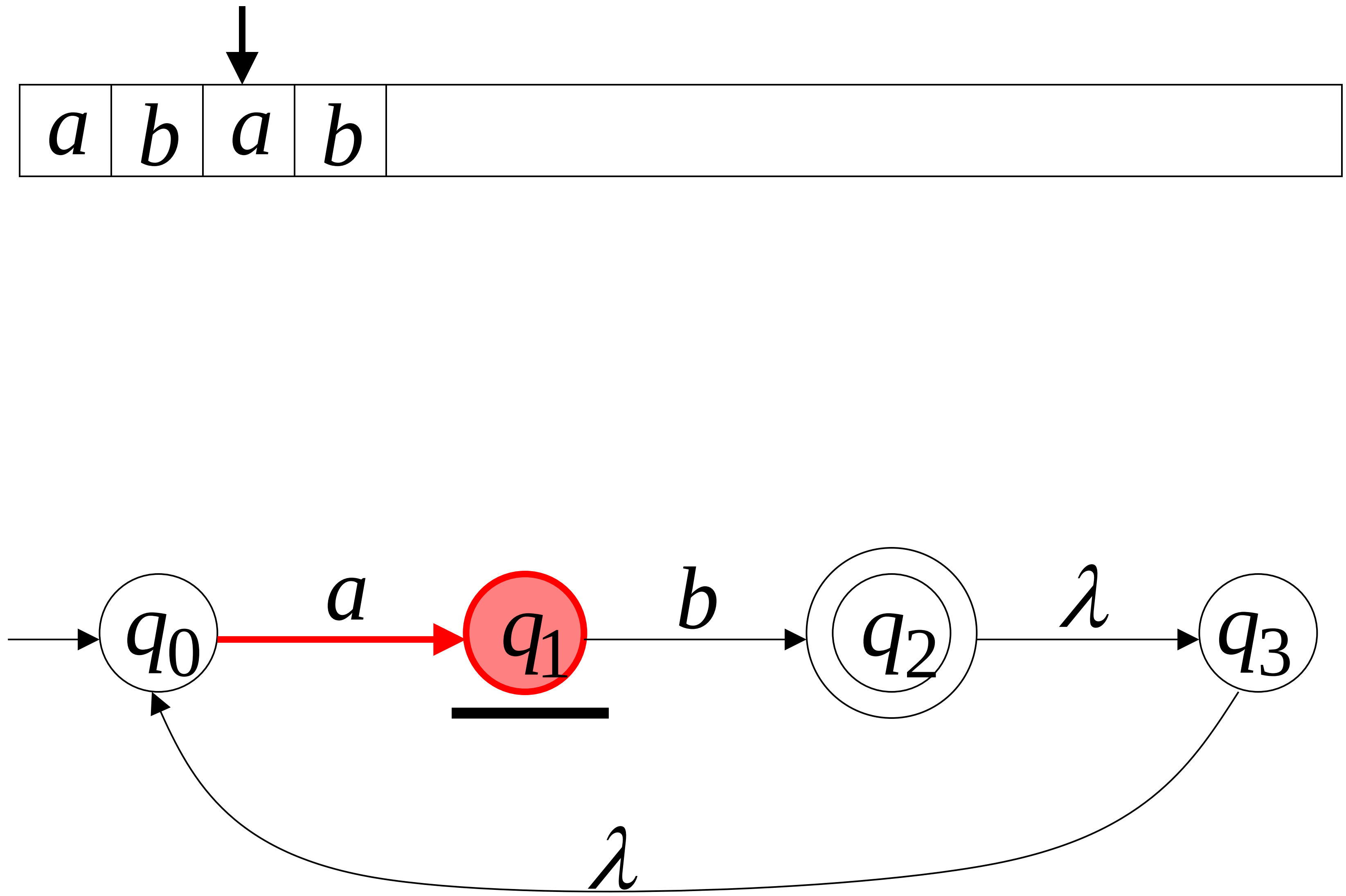

a
b
a
b
a
b
λ
λ
q0
q1
q2
q3



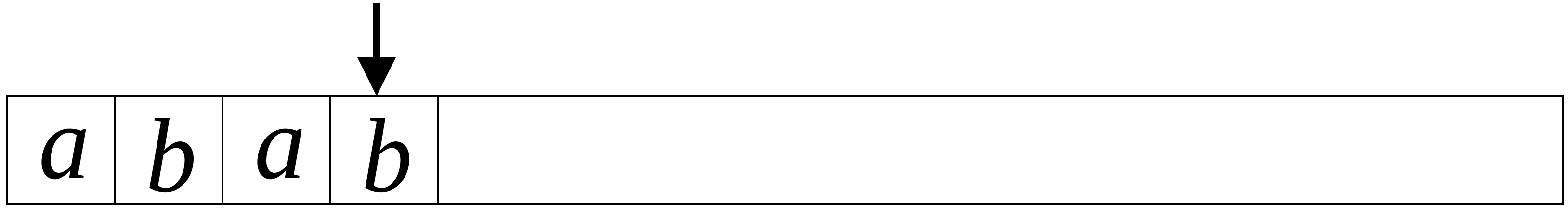
a
b
a
b

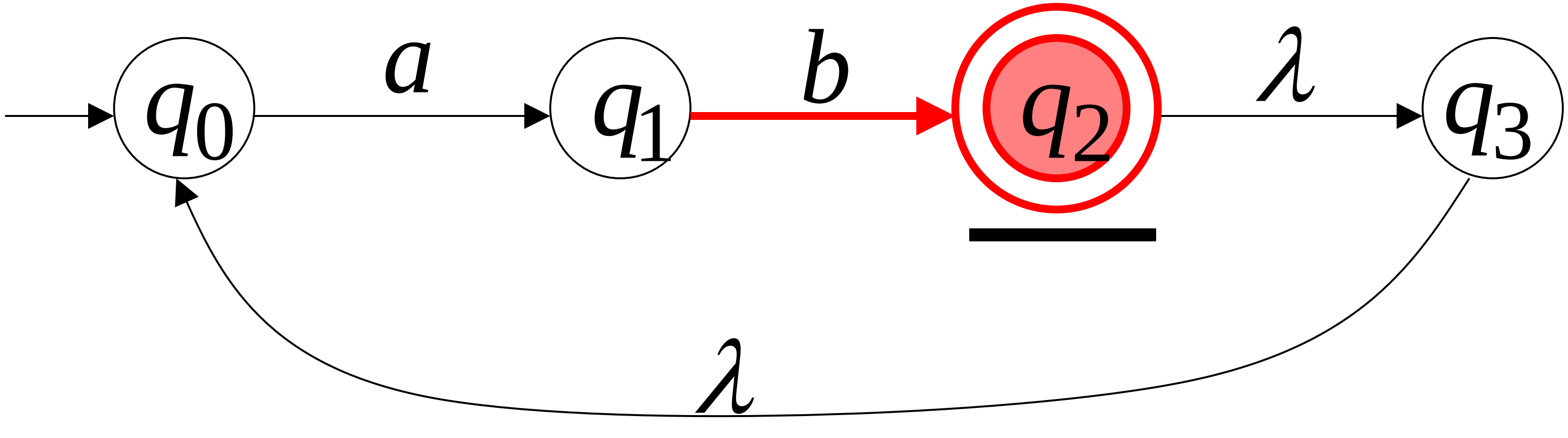


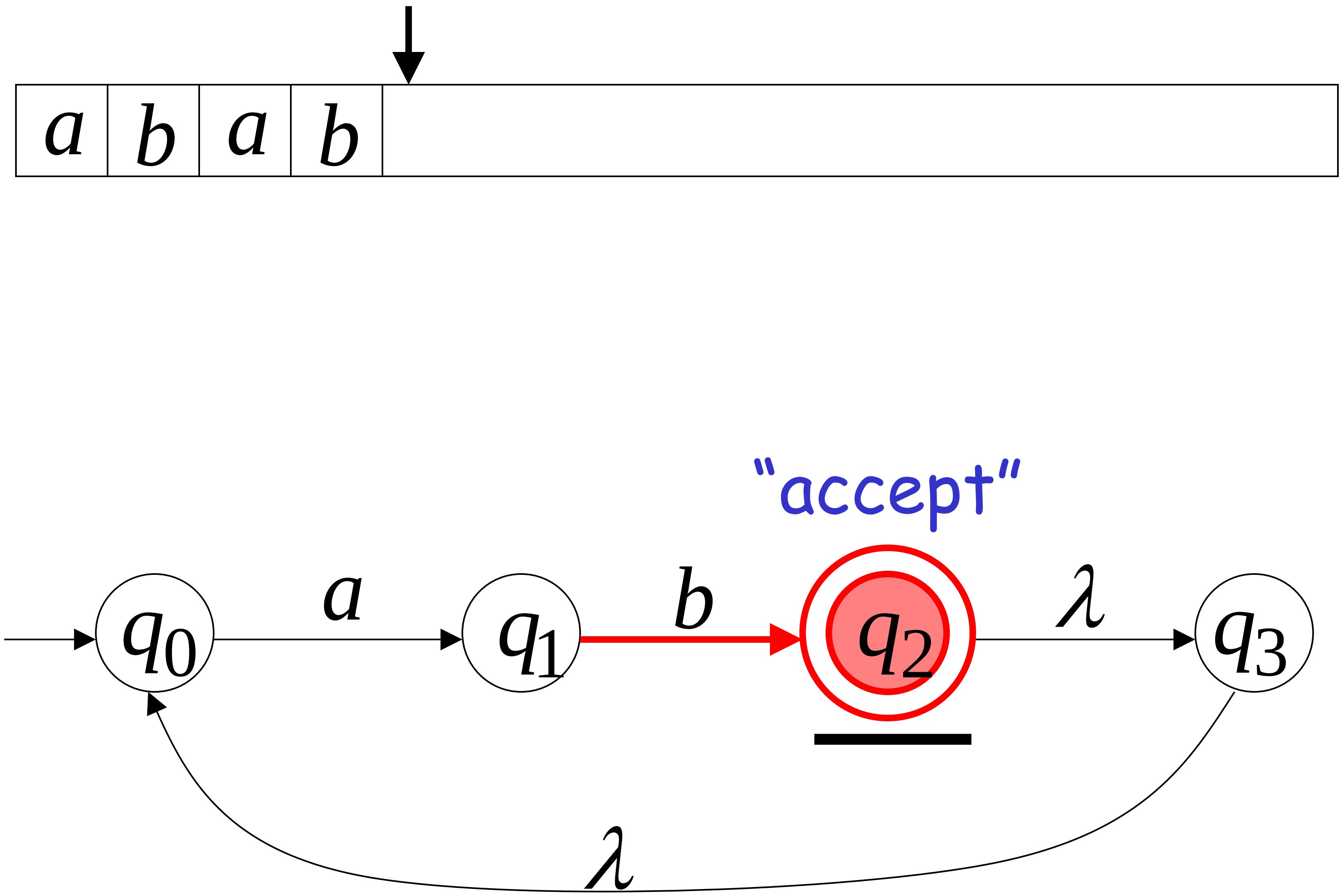
q_0
a
q_1
b
q_2
λ
q_3
λ



a
b
a
b
a
q0
q1
b
q2
λ
q3
λ


a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

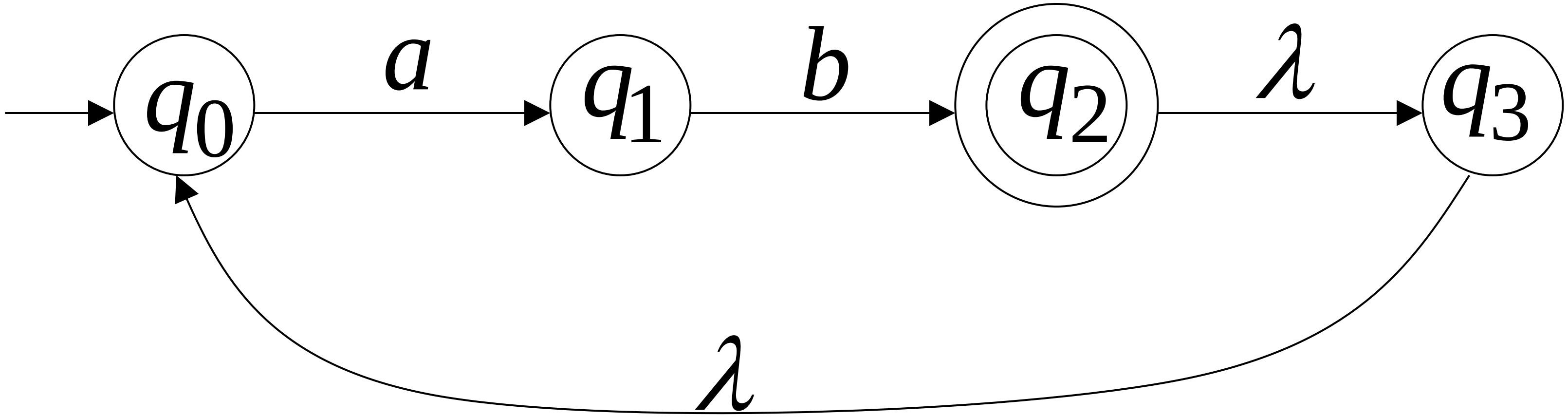


a
b
a
b
"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

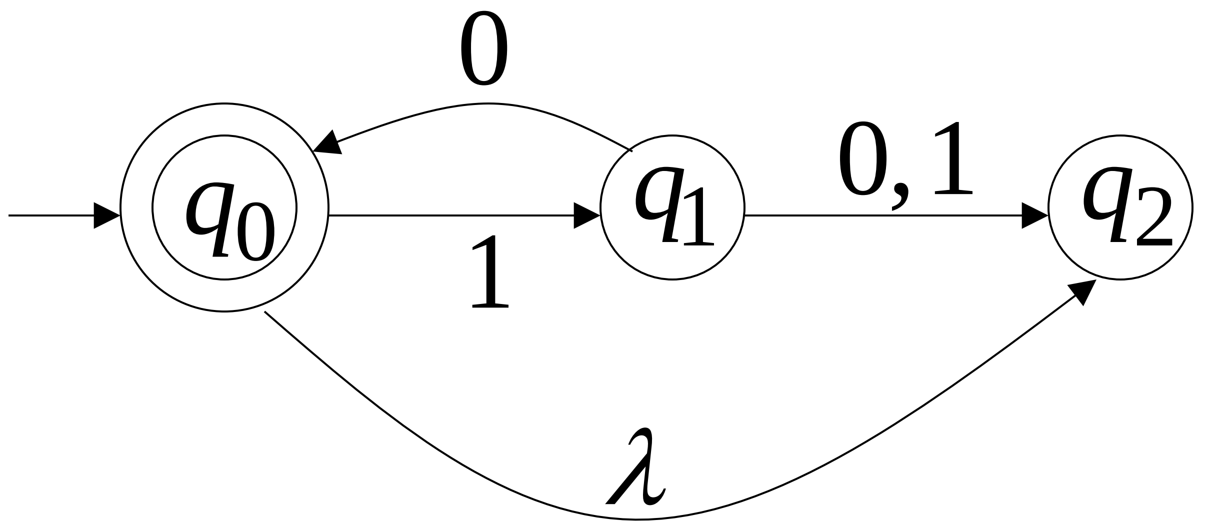
Language accepted

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

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



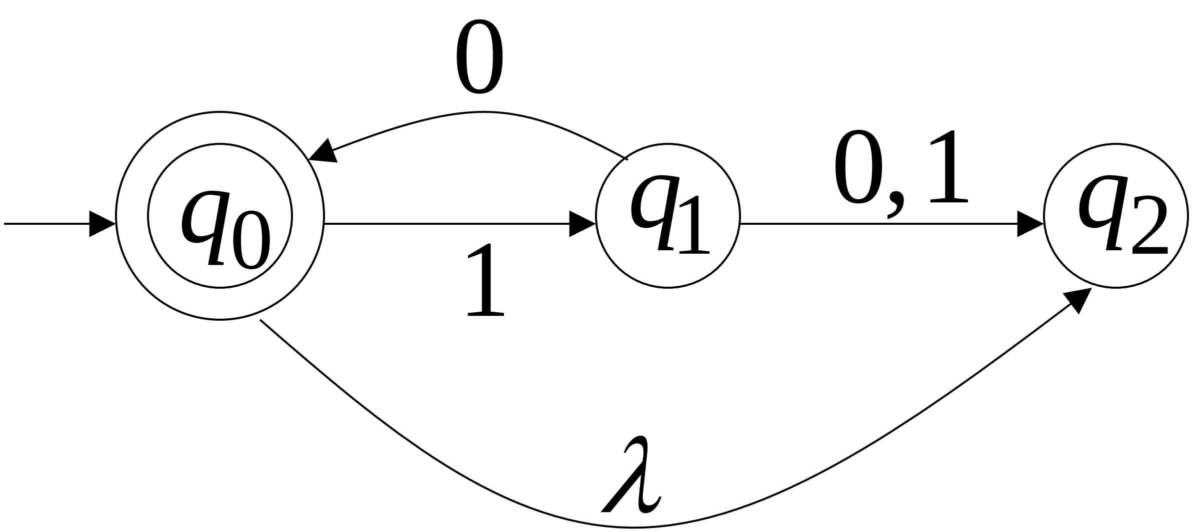
Another NFA Example: L(M)?



Language accepted

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

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



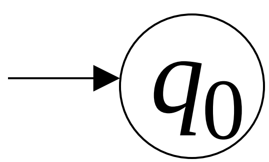
(redundant state)

Remarks:

- The λ symbol never appears on the input tape

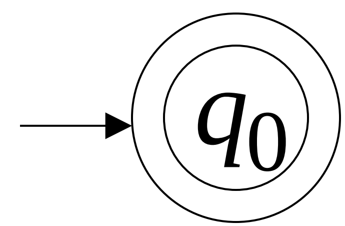
- Simple automata: Languages?

M_1



$L(M_1) = \{\}$

M_2

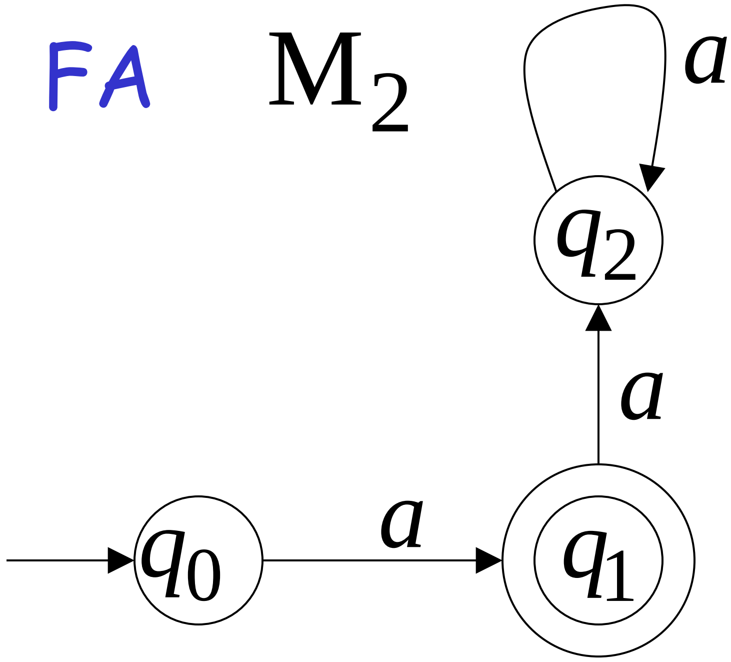


$L(M_2) = \{\lambda\}$

λ-transition in deterministic automata?

• NFAs are interesting because we can express languages easier than FAs

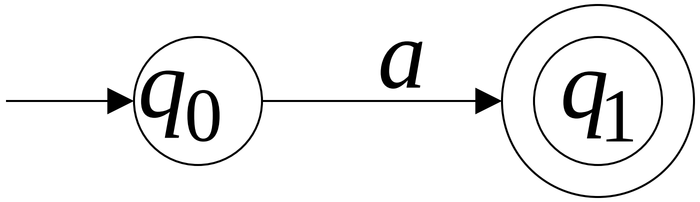
FA M_2



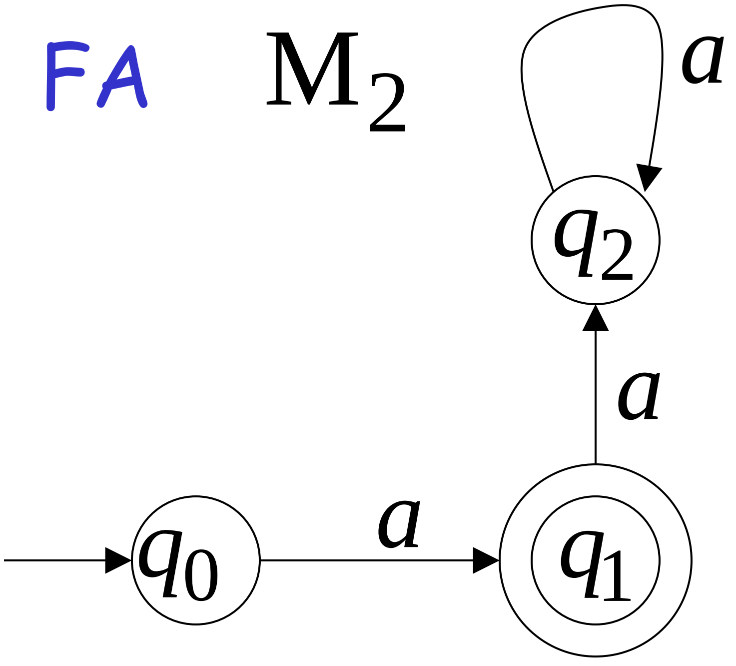
$$L(M_2) = \{a\}$$

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

NFA M_1



FA 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 alphabet, i.e. $\{a, b\}$

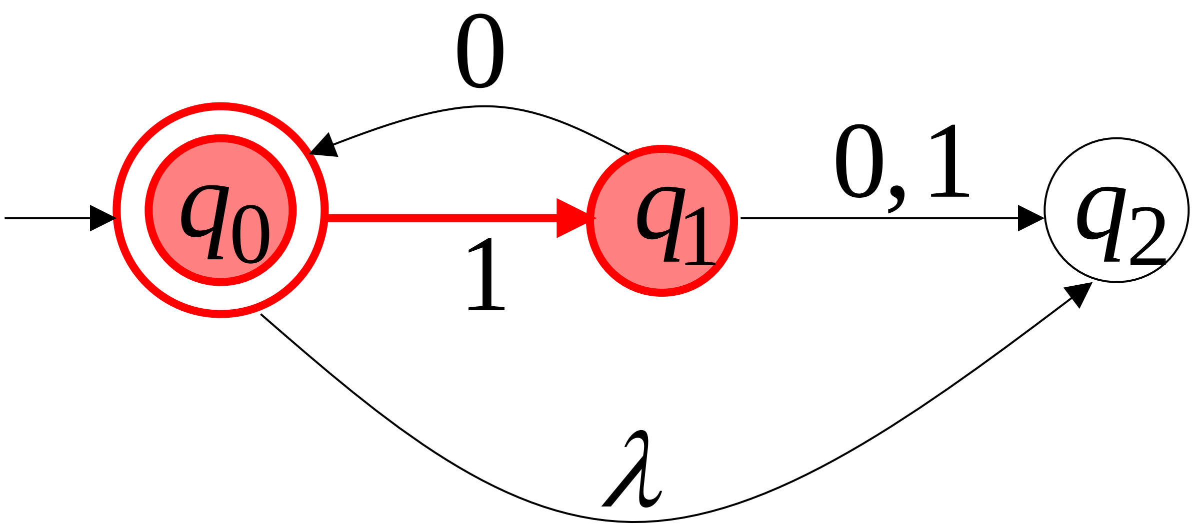
$\delta:$ Transition function

$q_0:$ Initial state

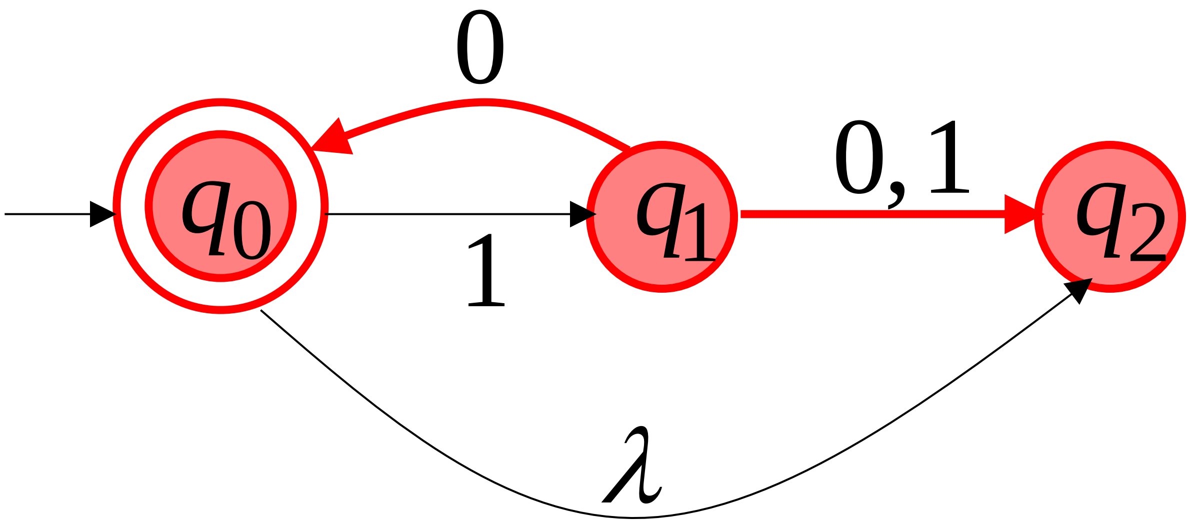
$F:$ Accepting states

Transition Function δ

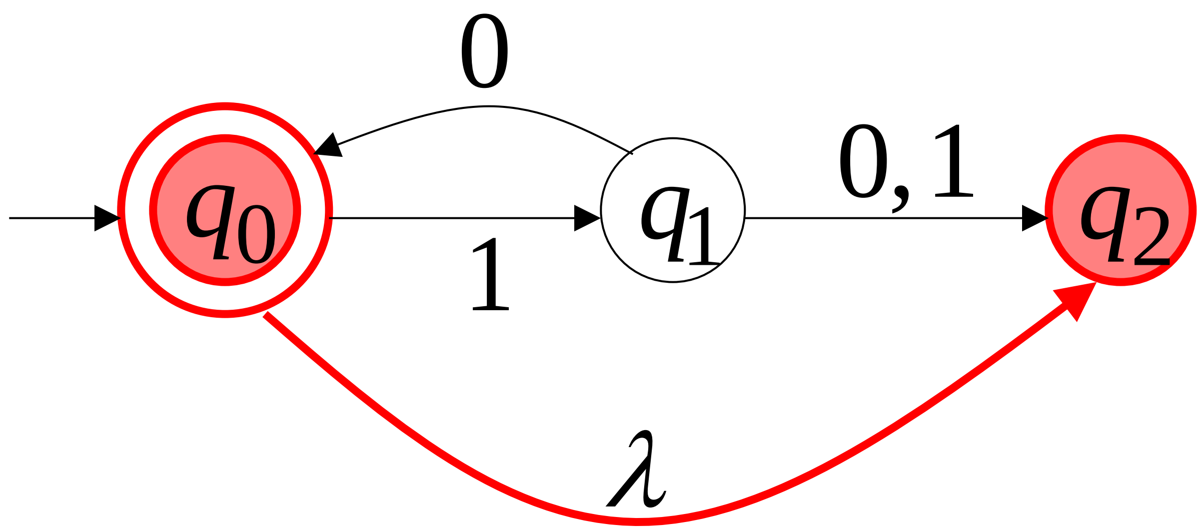
$$\delta(q_0, 1) = \{q_1\}$$



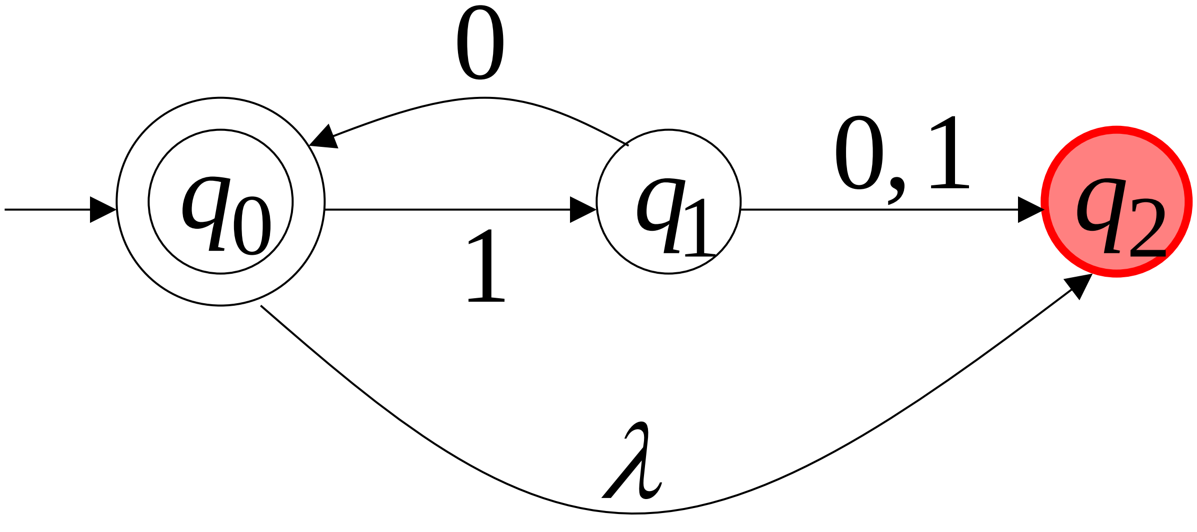
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

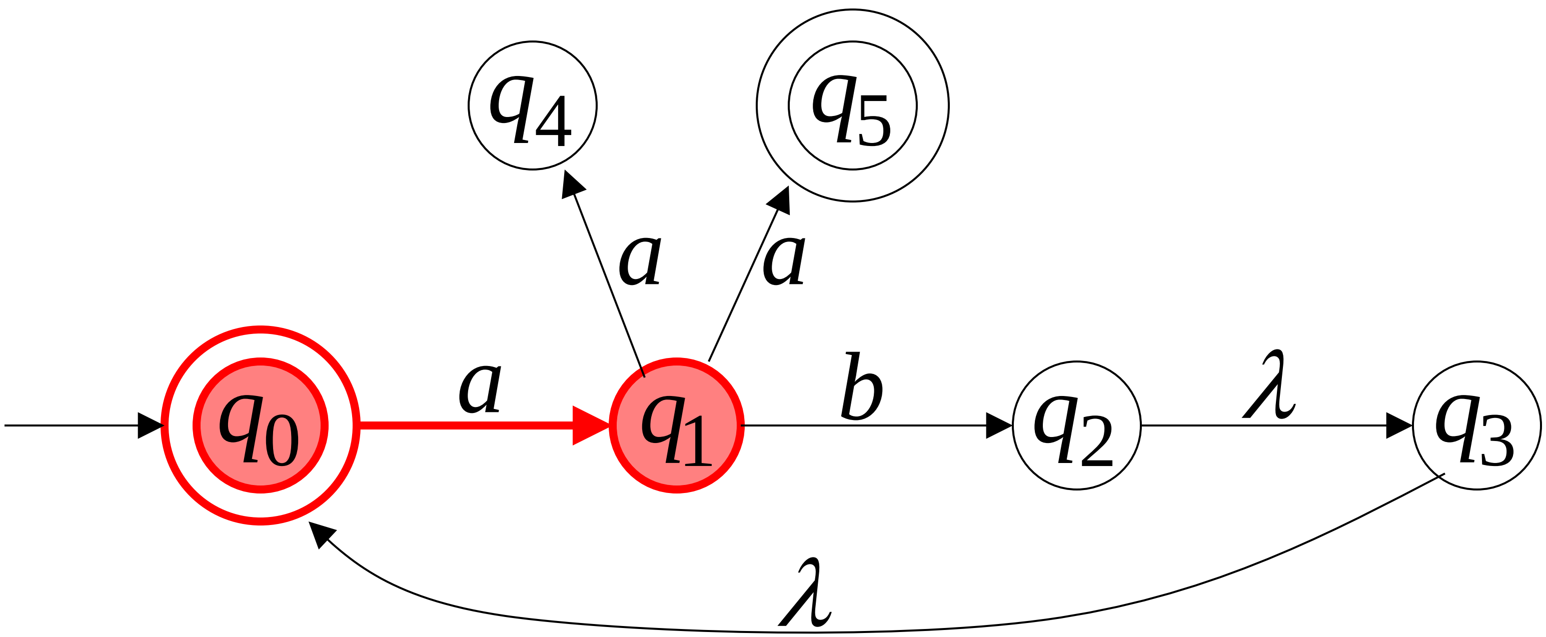


$$\delta(q_2, 1) = \varnothing$$

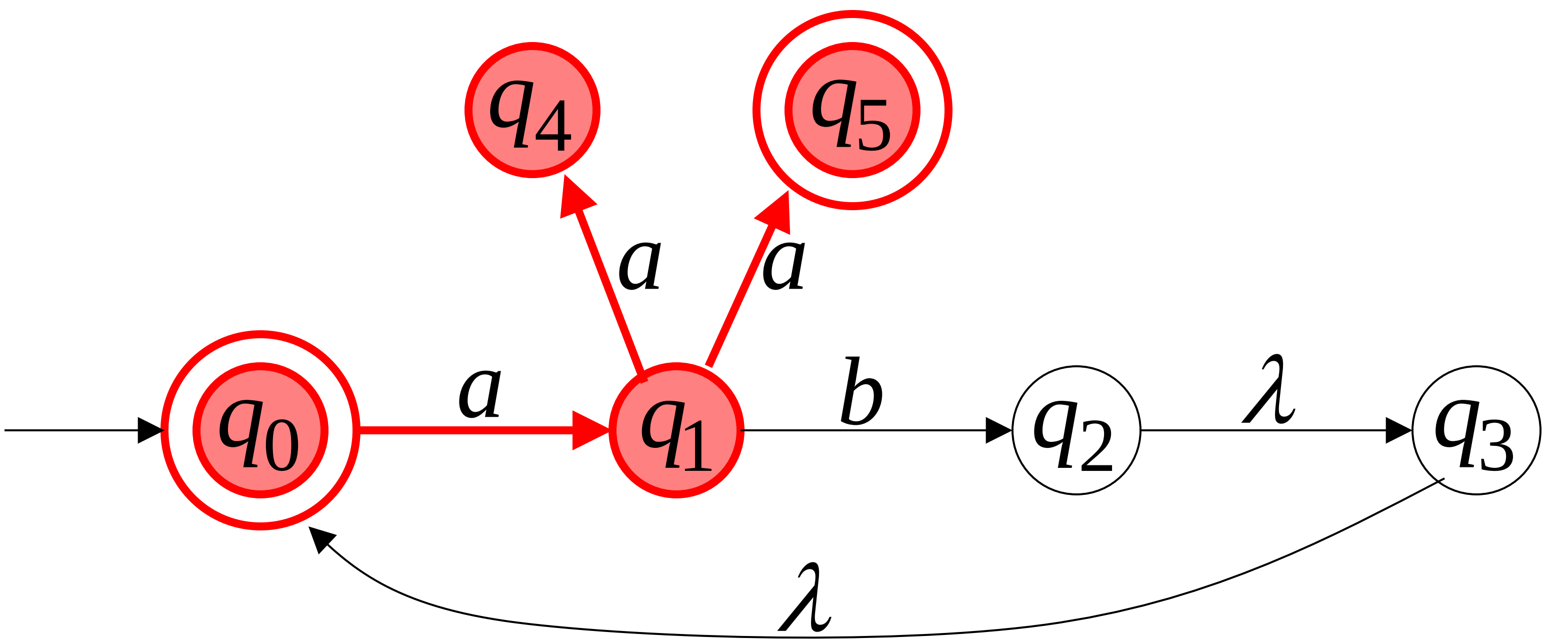


Extended Transition Function δ^*

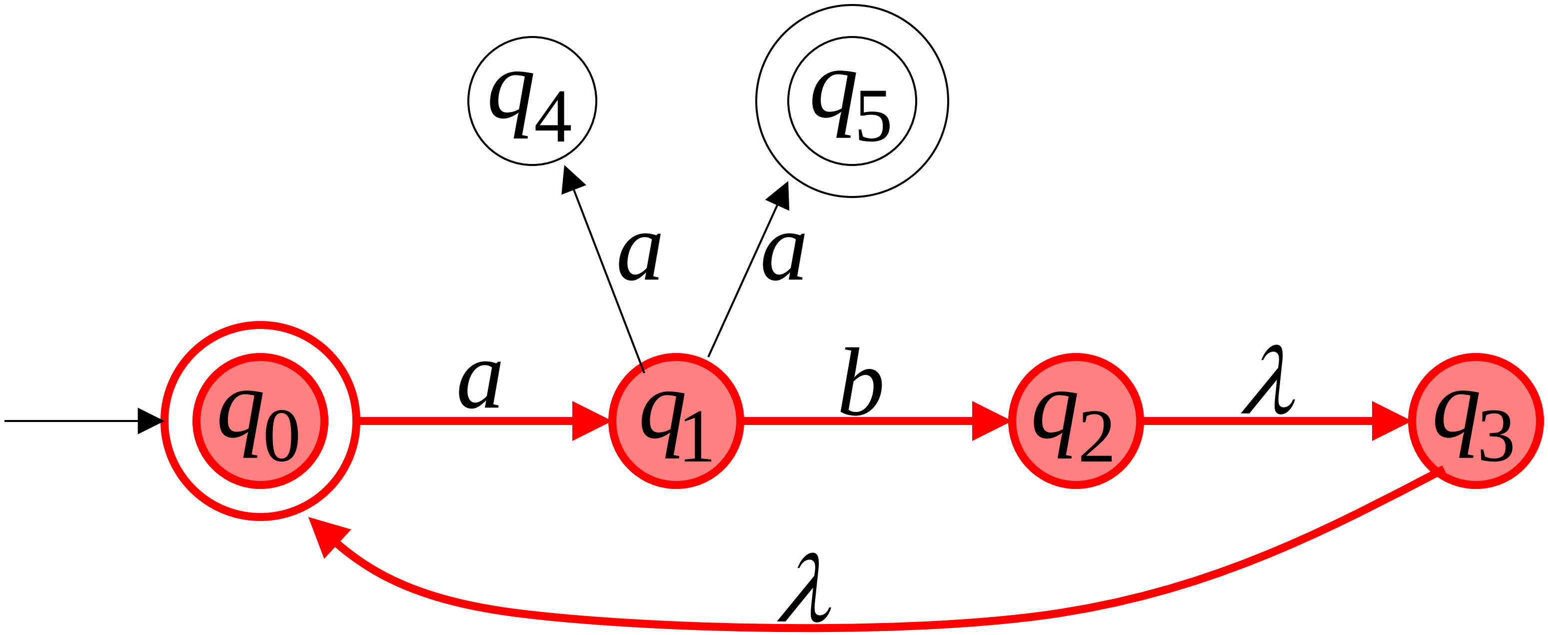
$$\delta^*(q_0, a) = \{q_1\}$$



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

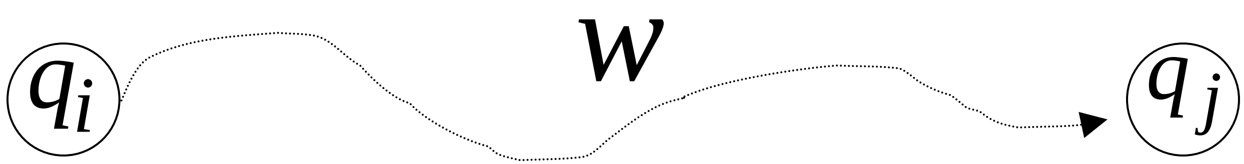


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

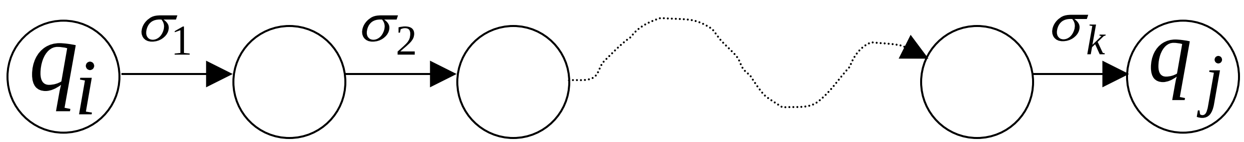


Formally

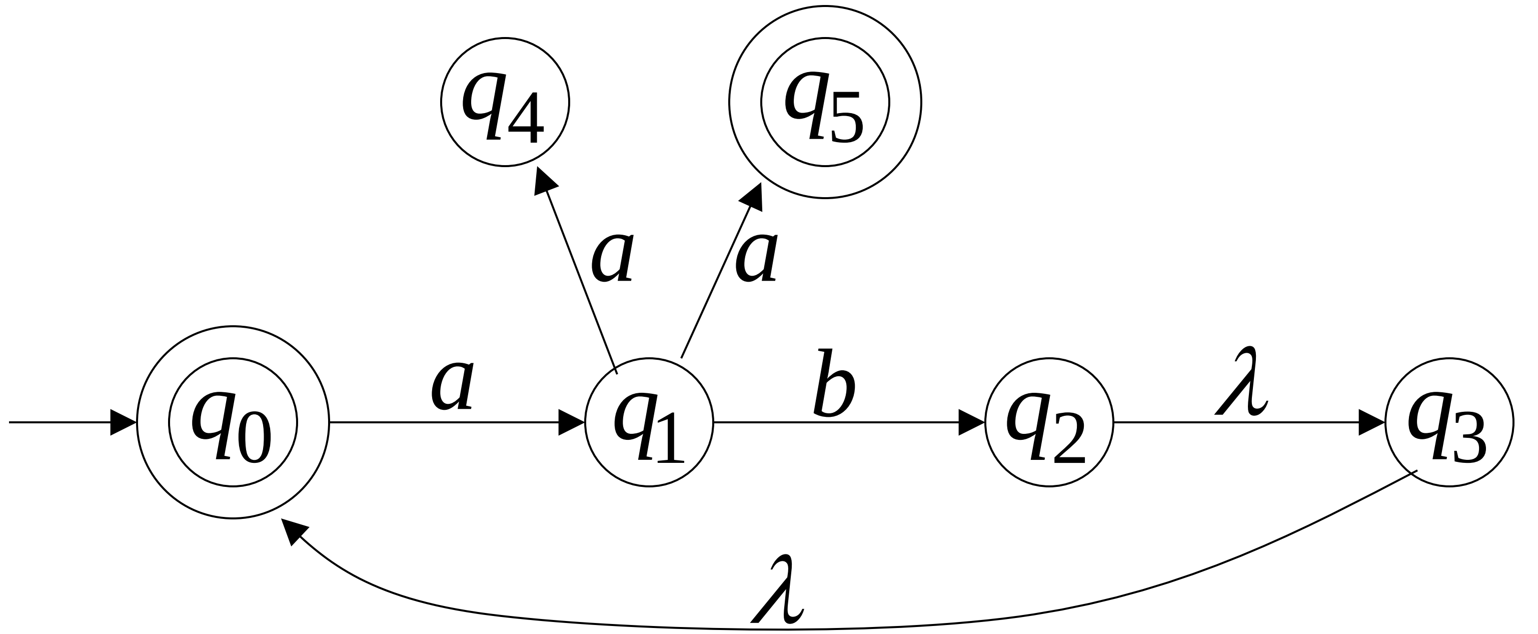
$q_j \in \delta^*(q_i, w)$: there is a walk from q_i to q_j with label w



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

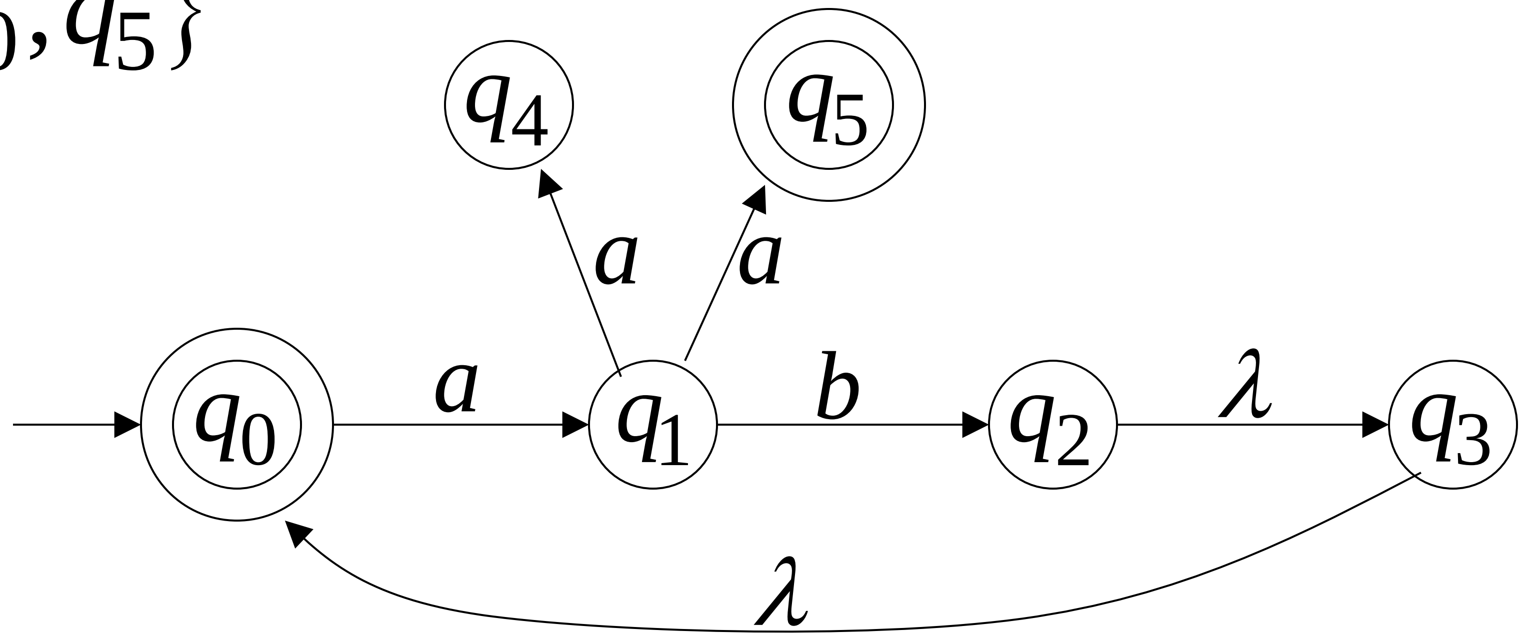


L(M)?



The Language of an NFA M

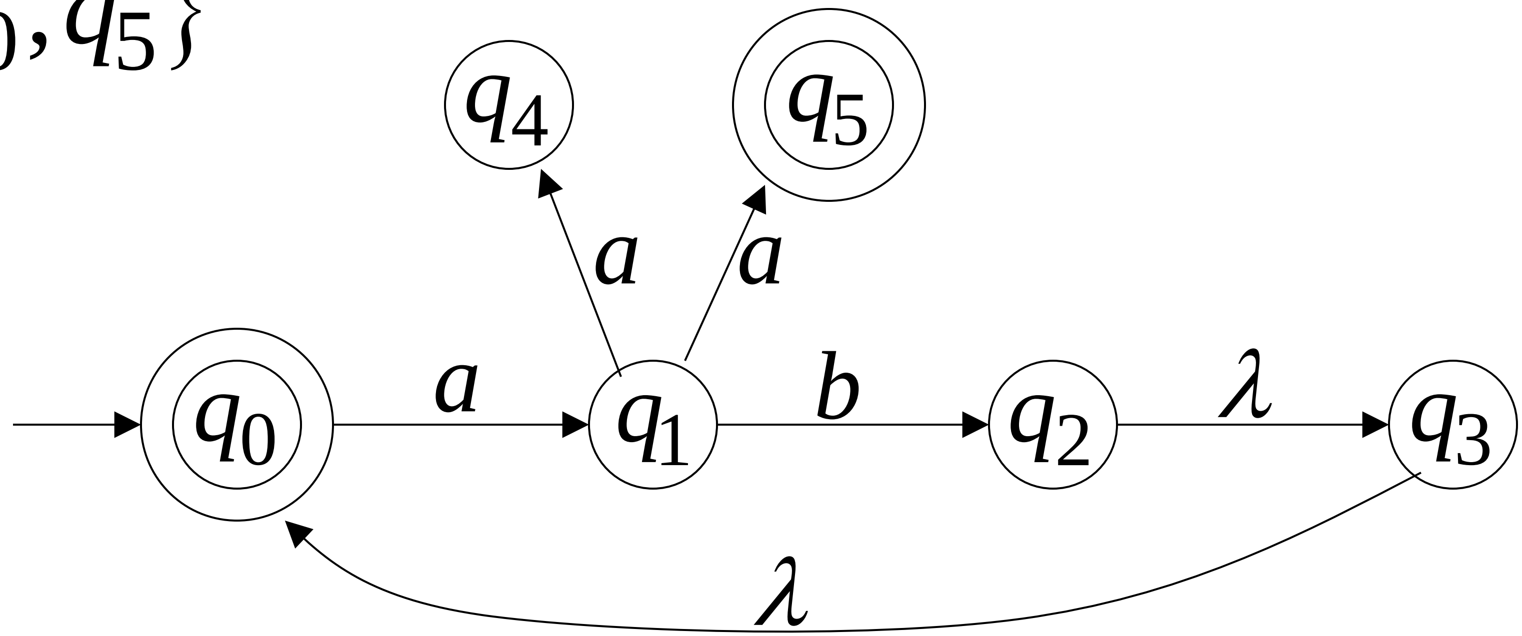
$F=\{q_0, q_5\}$



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

$$\in F$$

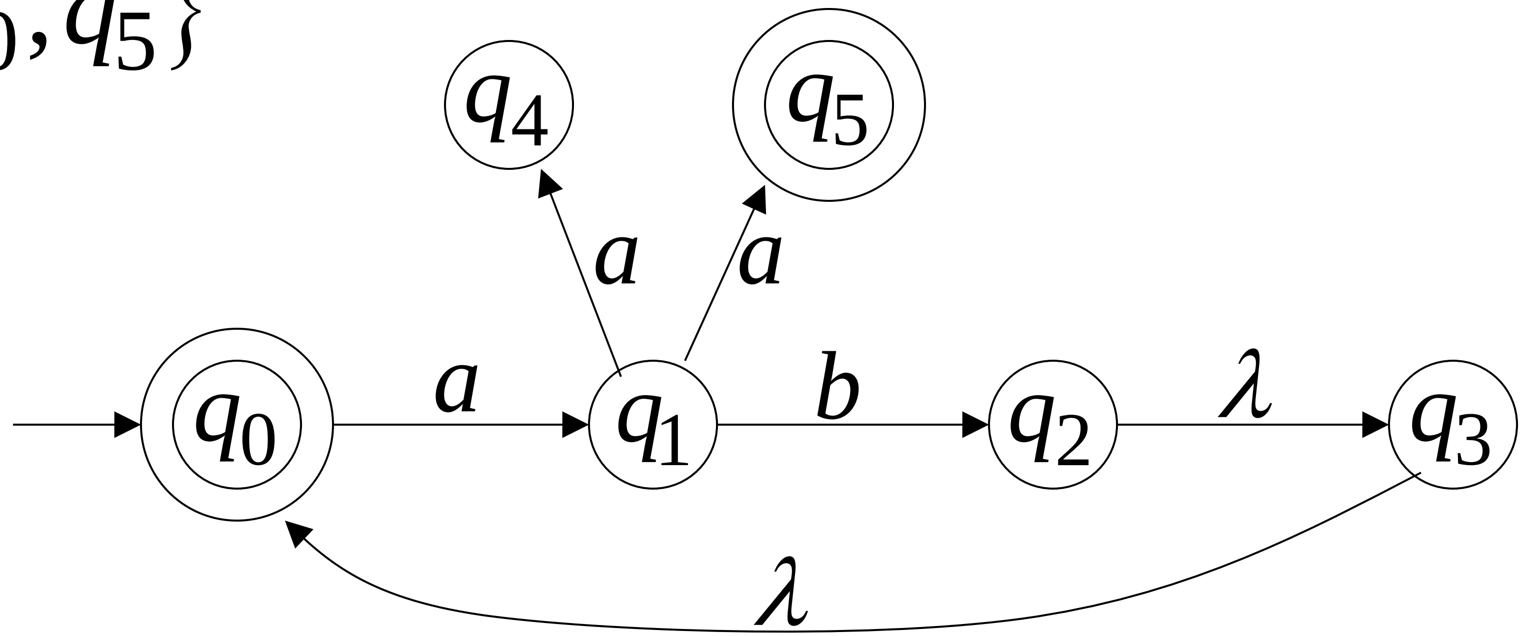
$$F = \{q_0, q_5\}$$



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

$$\searrow \in F$$

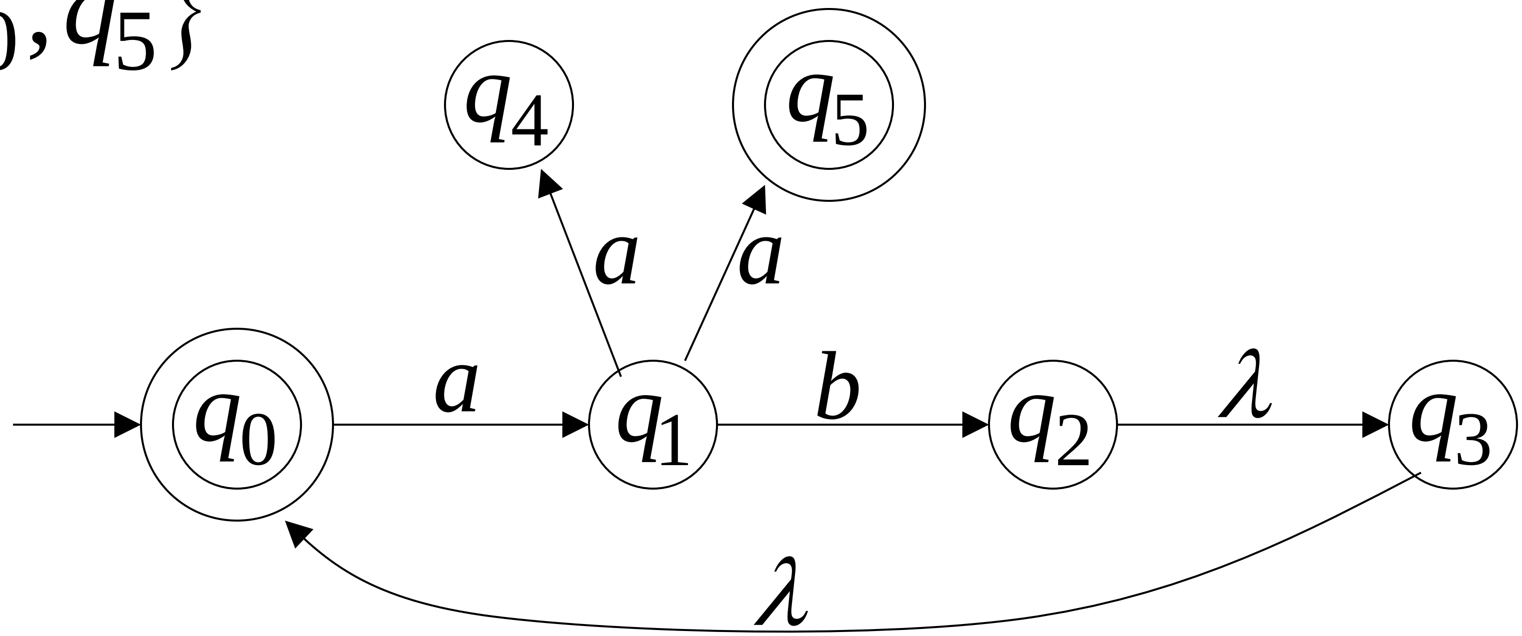
$$F = \{q_0, q_5\}$$



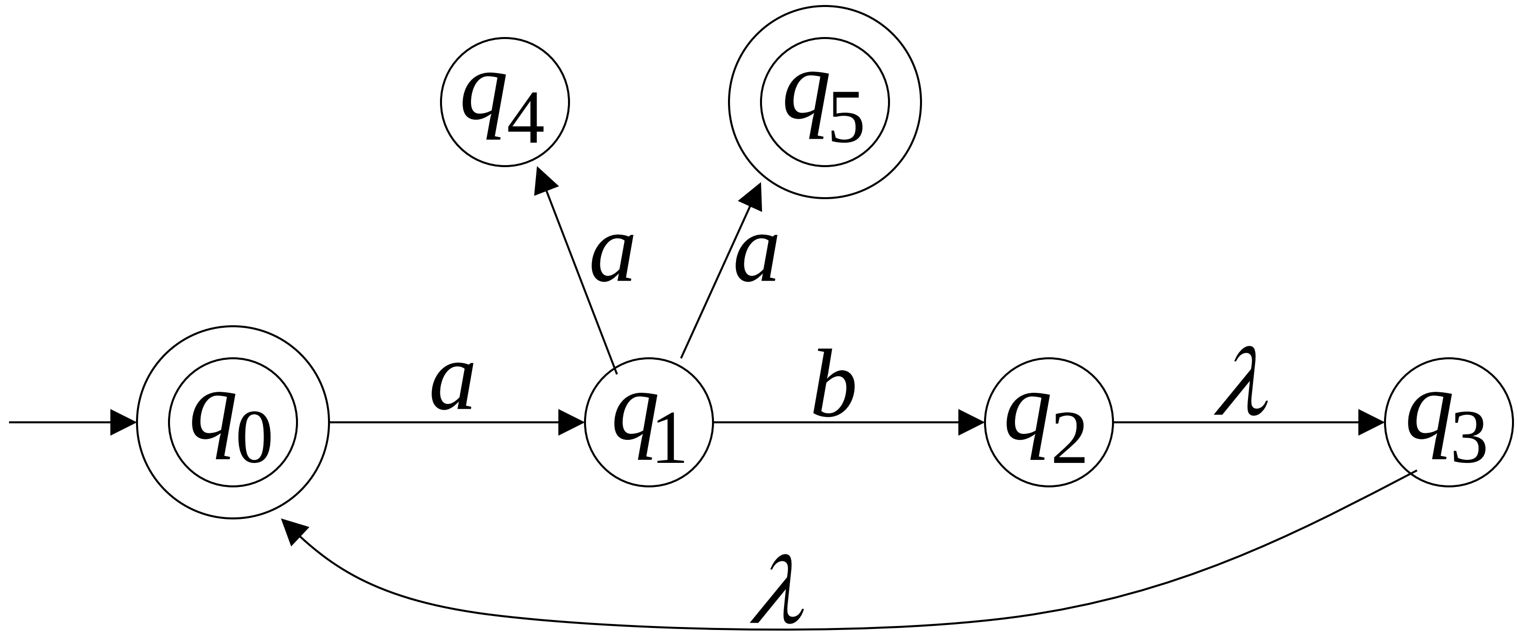
$$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\} \qquad aaba \in L(M)$$

$$\in F$$

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



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



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

Formally

The language accepted by NFA M is:

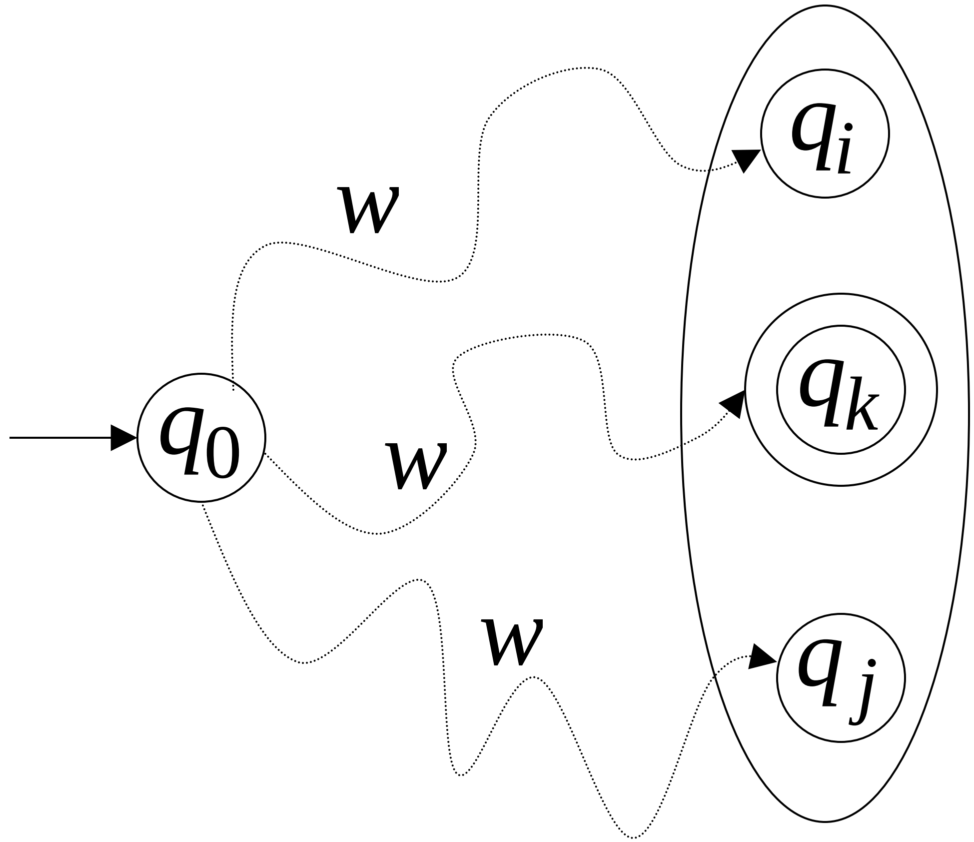
$$L(M) = \{w_1, w_2, w_3, \ldots\}$$

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

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

$$w \in L(M)$$

$$\delta^*(q_0, w)$$



$$q_k \in F$$