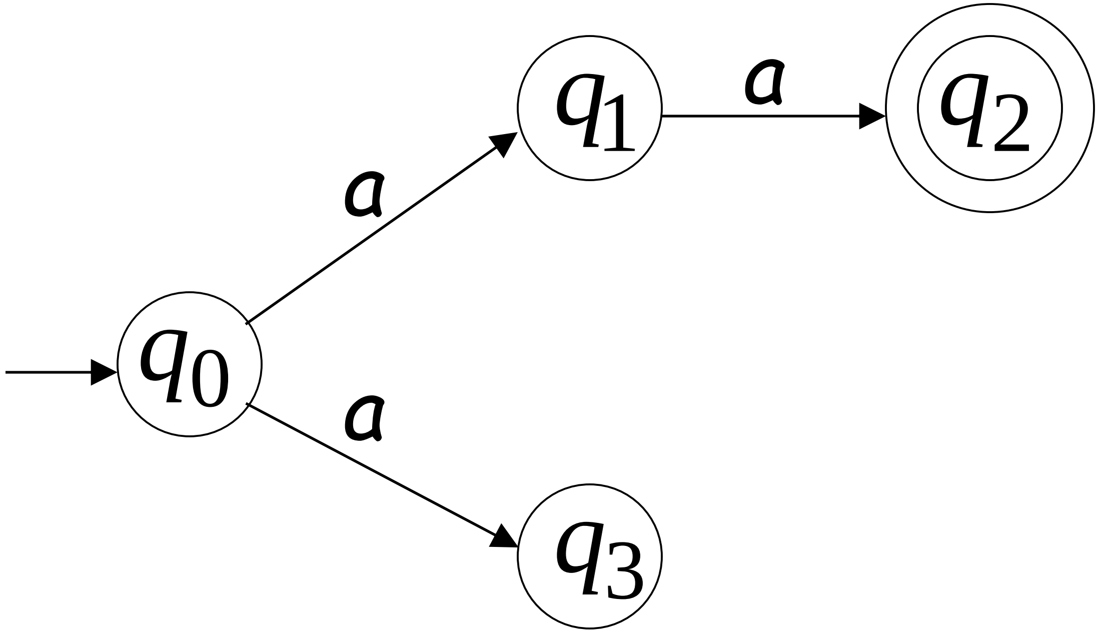
Non Deterministic Automata

Linz: Nondeterministic Finite Accepters, page 51

Nondeterministic Finite Accepter (NFA)

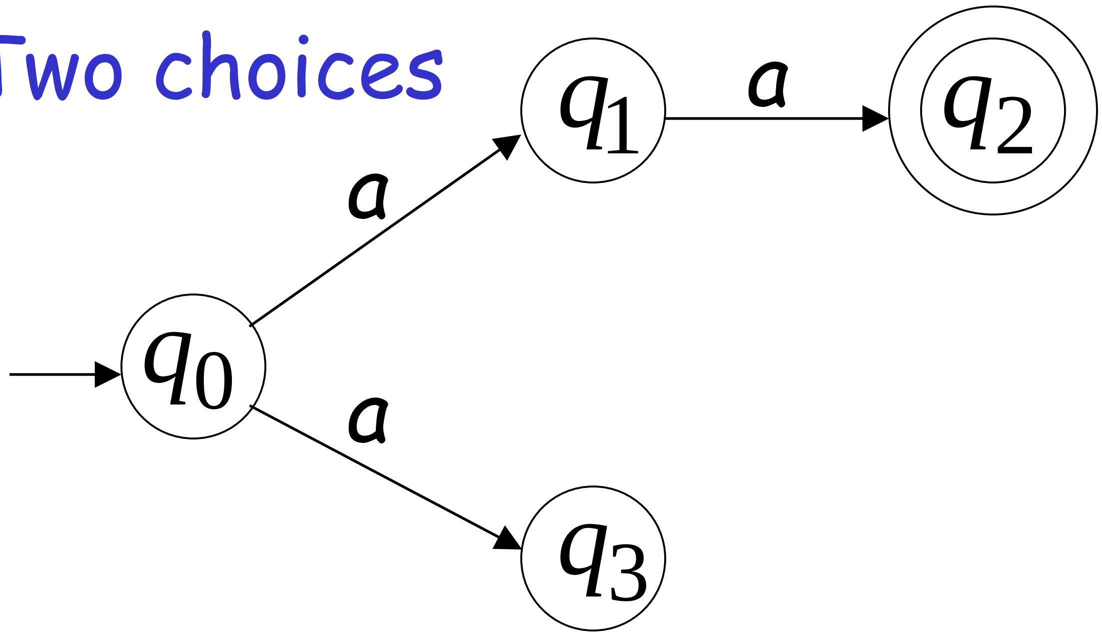
Alphabet = $\{a\}$



Nondeterministic Finite Accepter (NFA)

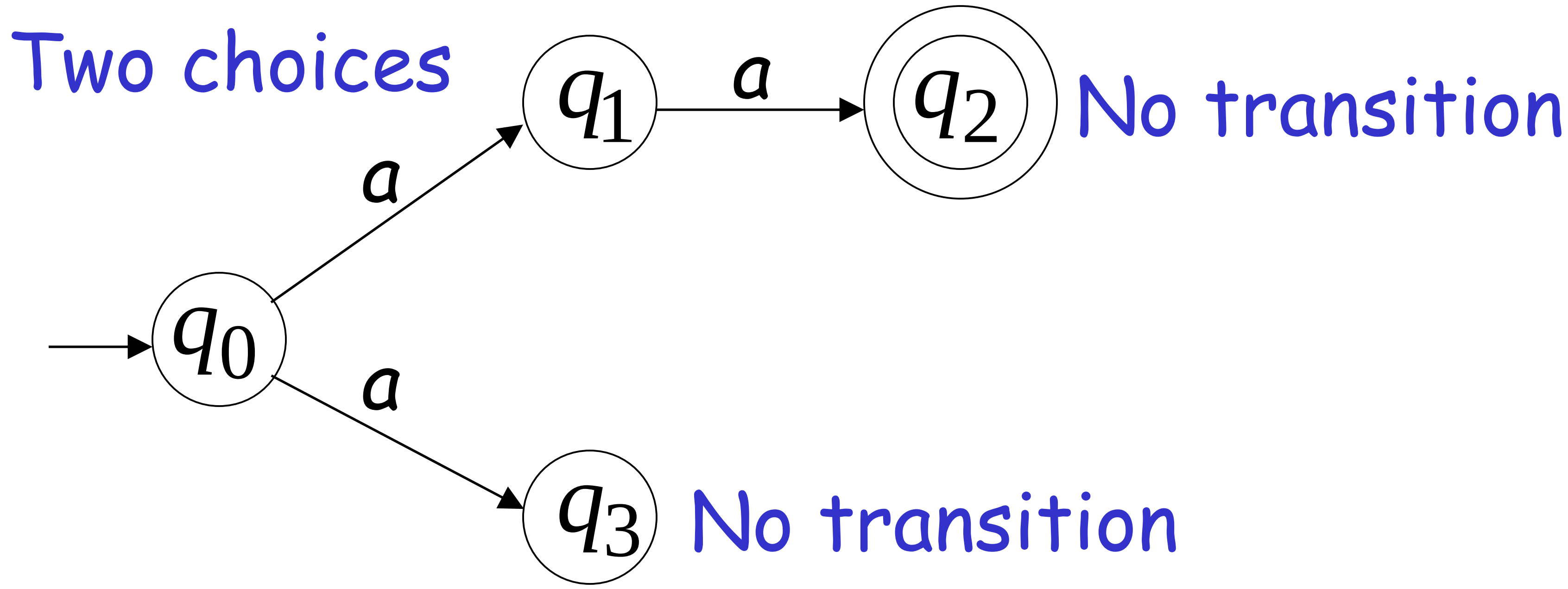
Alphabet = $\{a\}$

Two choices

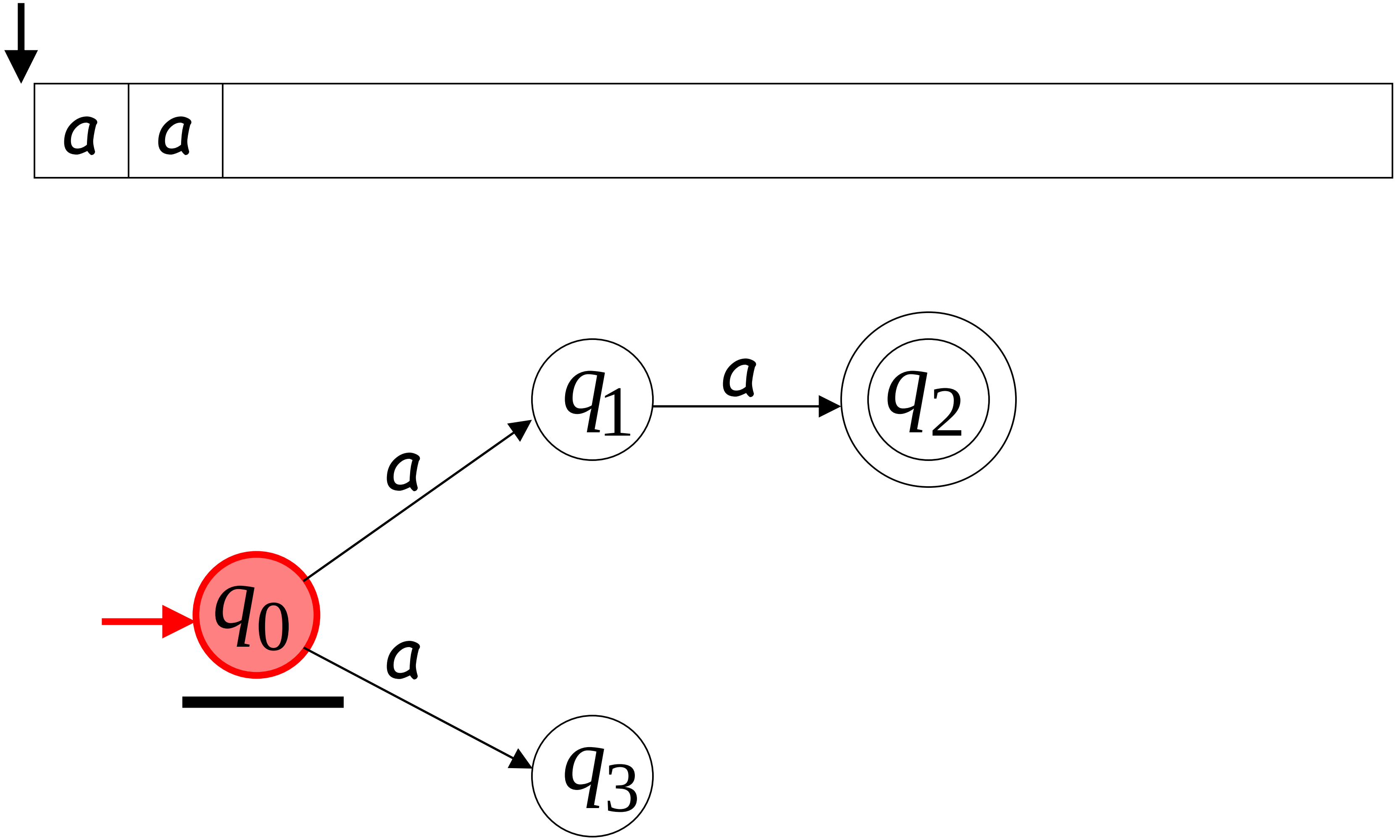


Nondeterministic Finite Accepter (NFA)

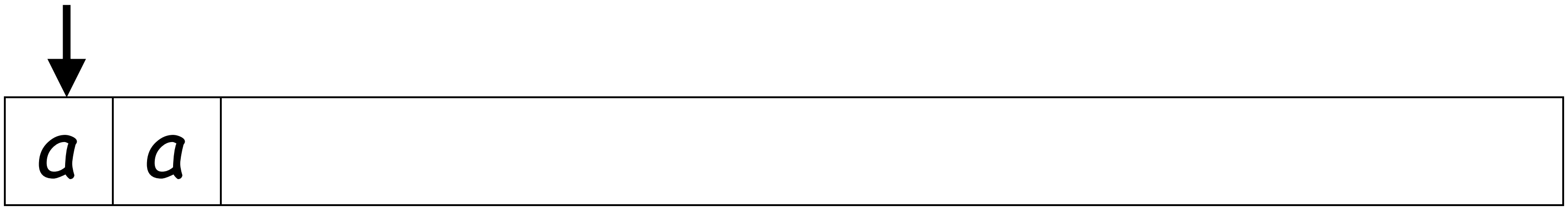
Alphabet = $\{a\}$

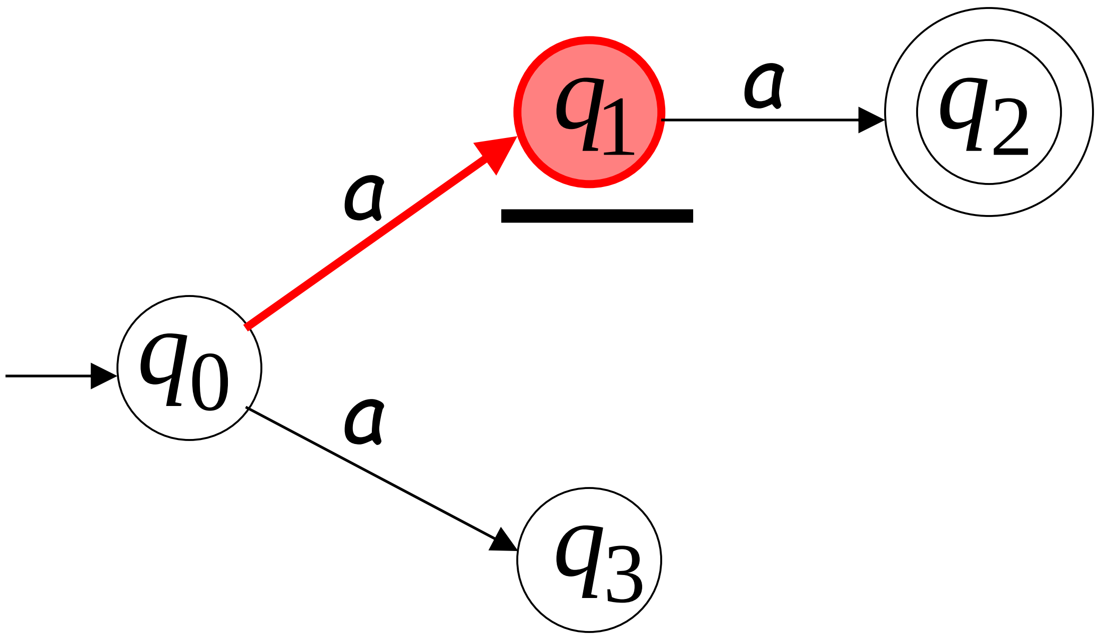


First Choice

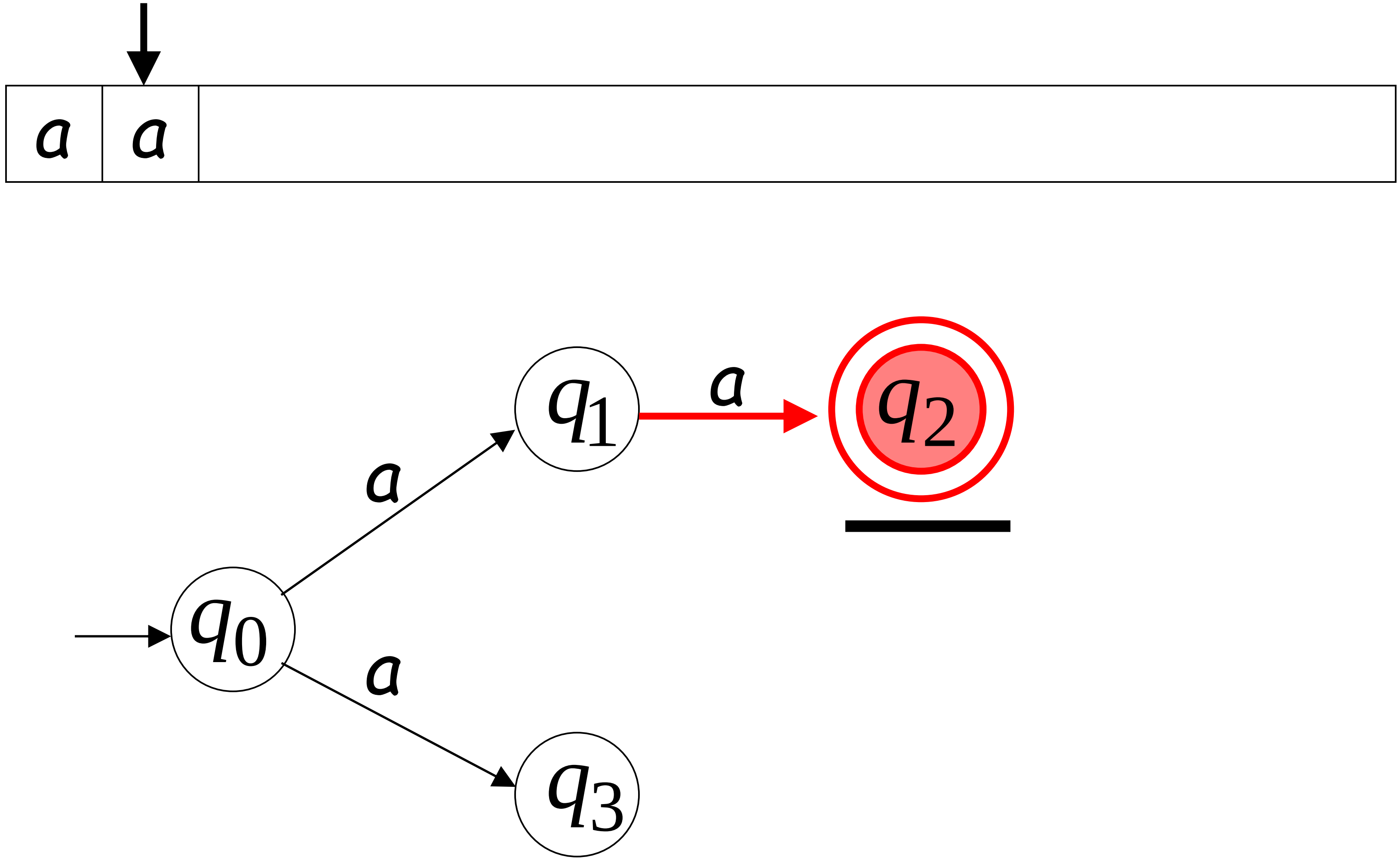


First Choice

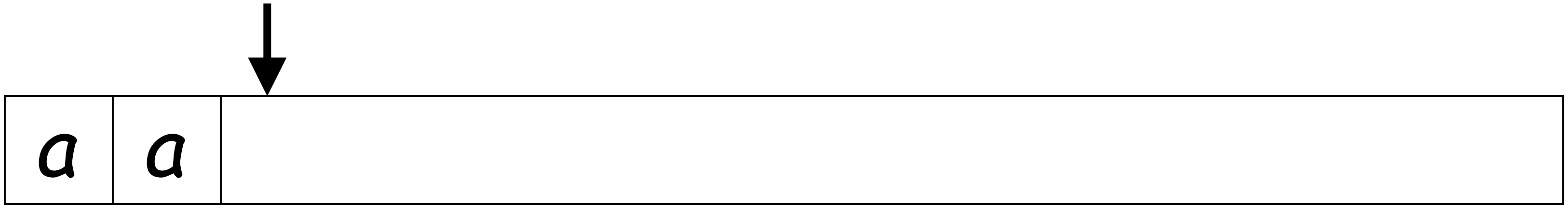




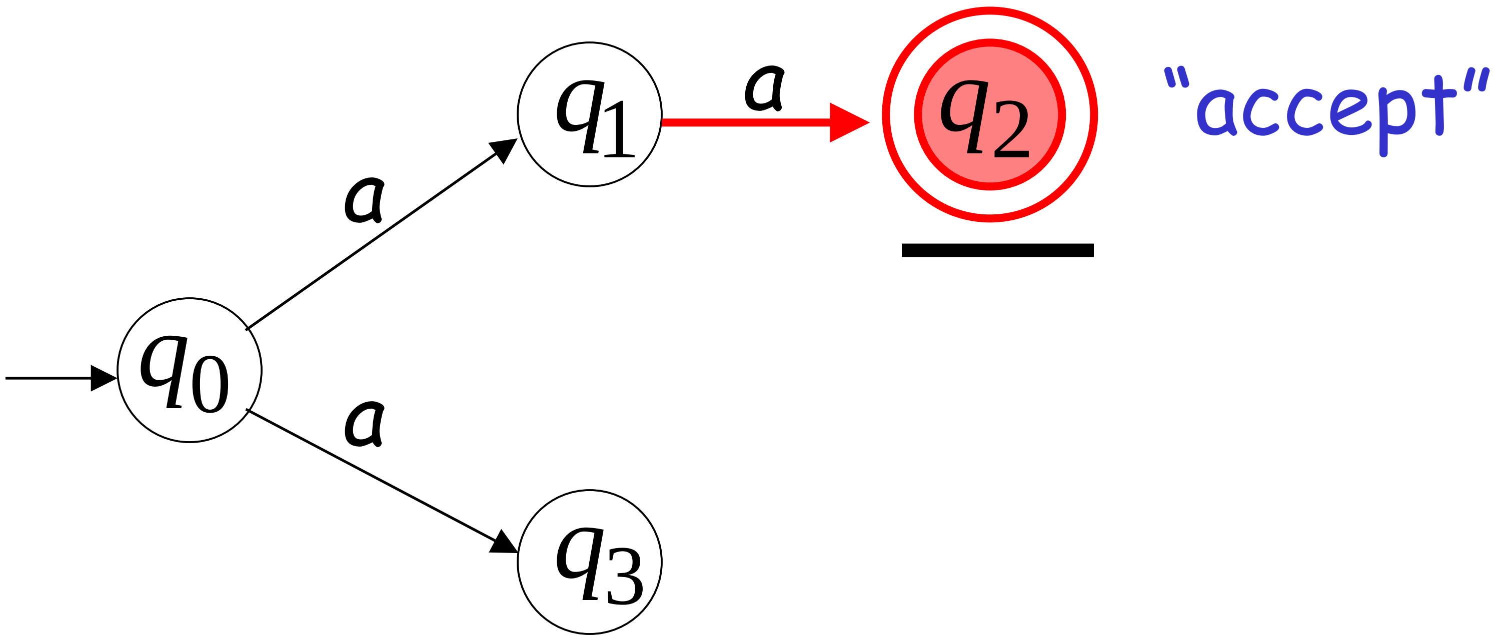
First Choice



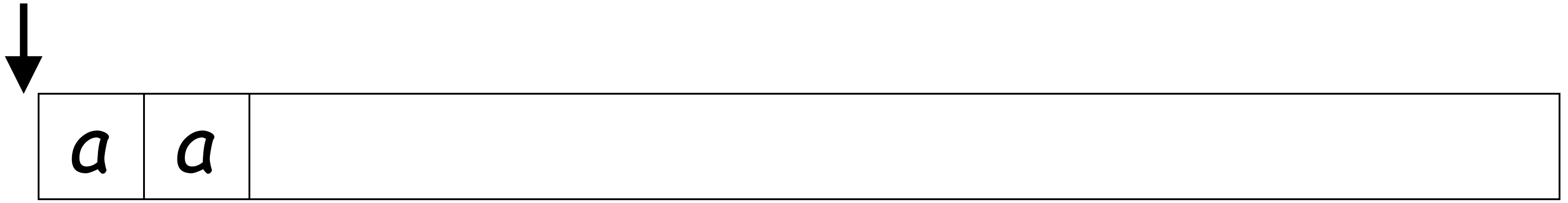
First Choice

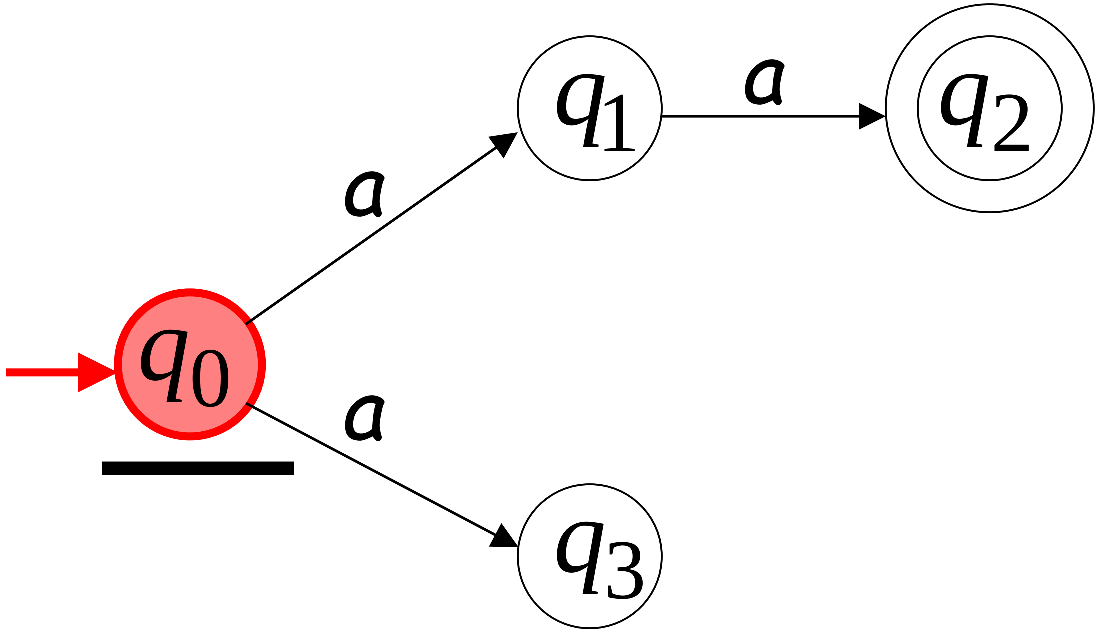


All input is consumed

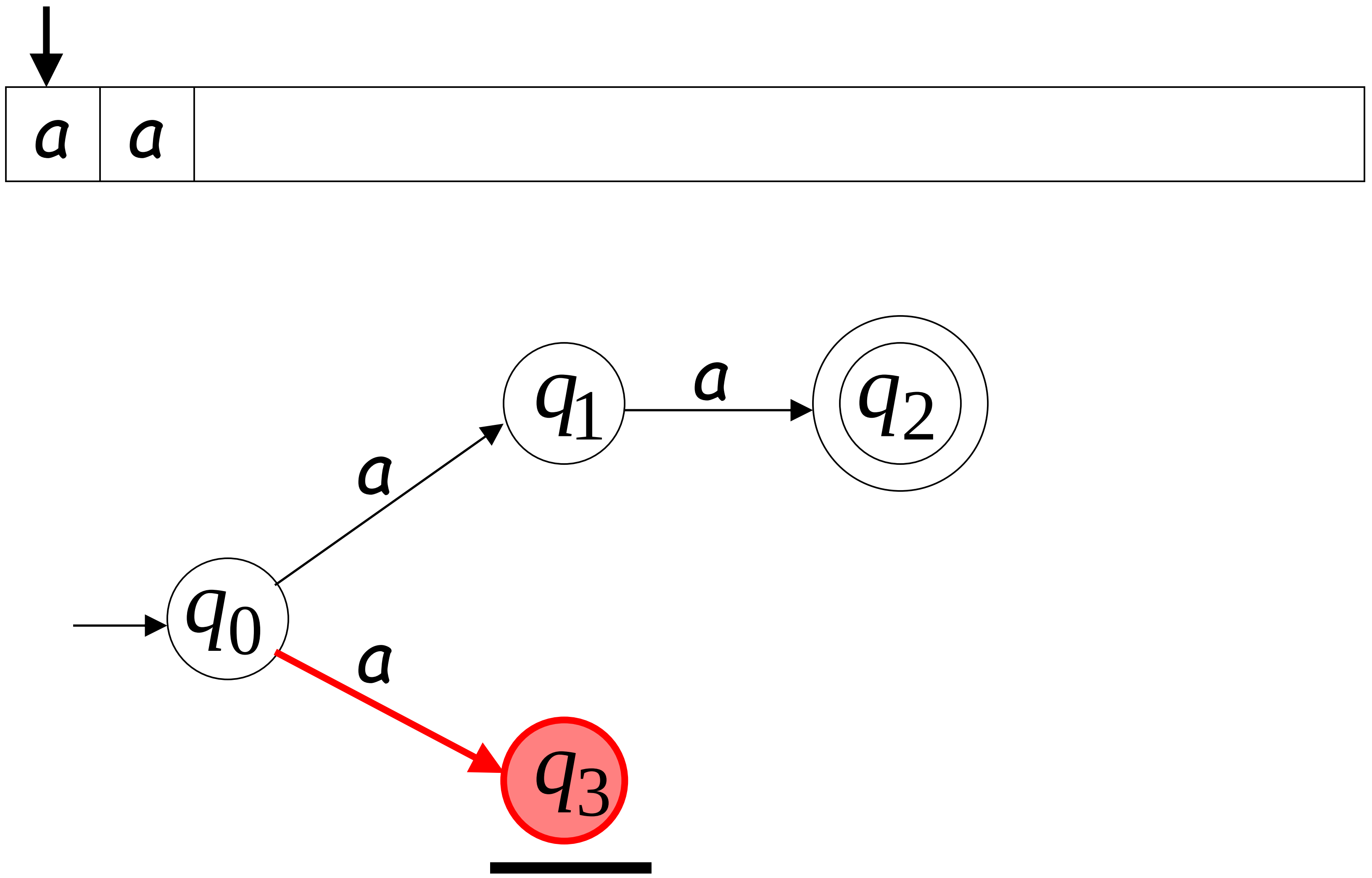


Second Choice

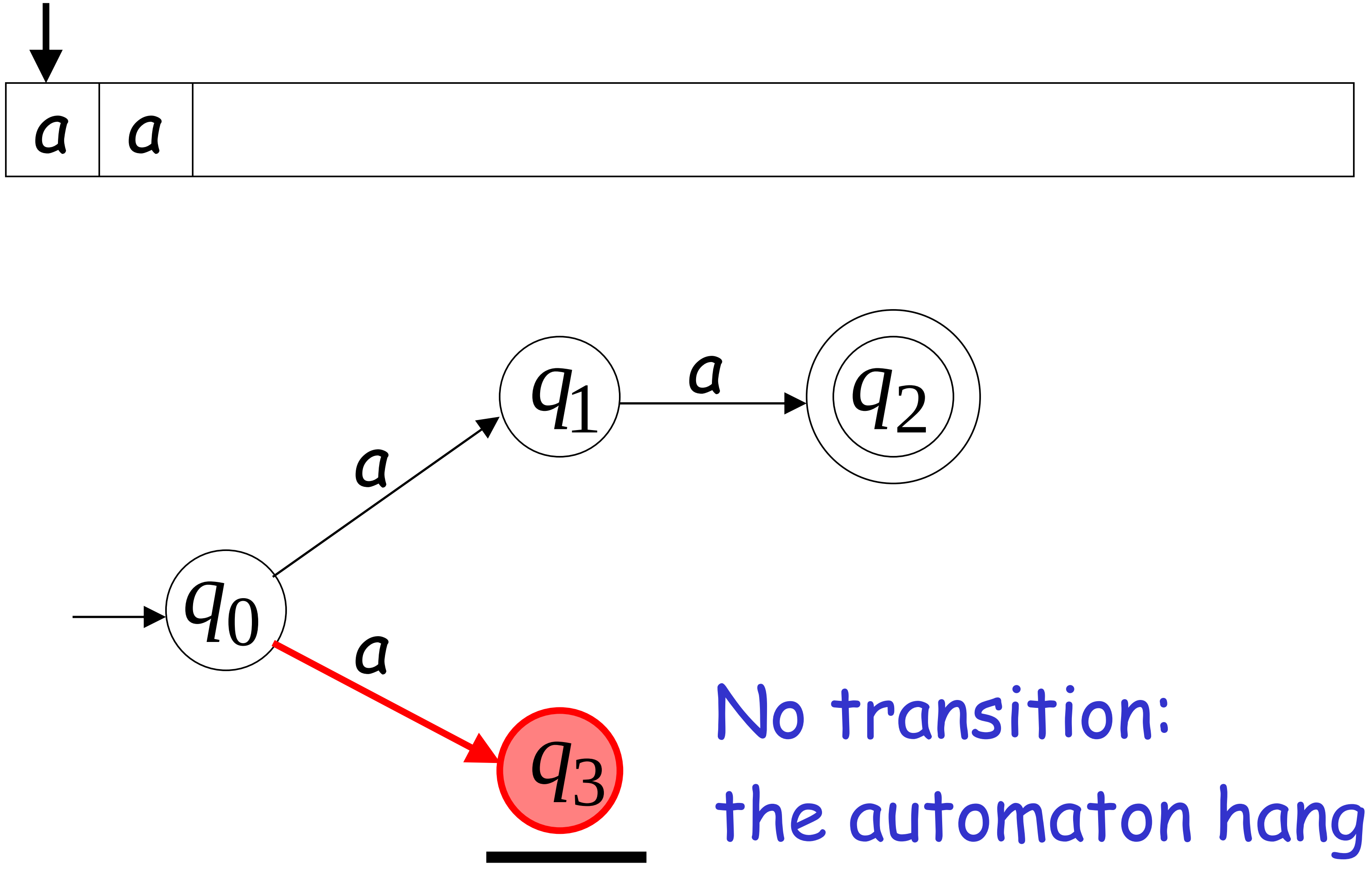




Second Choice

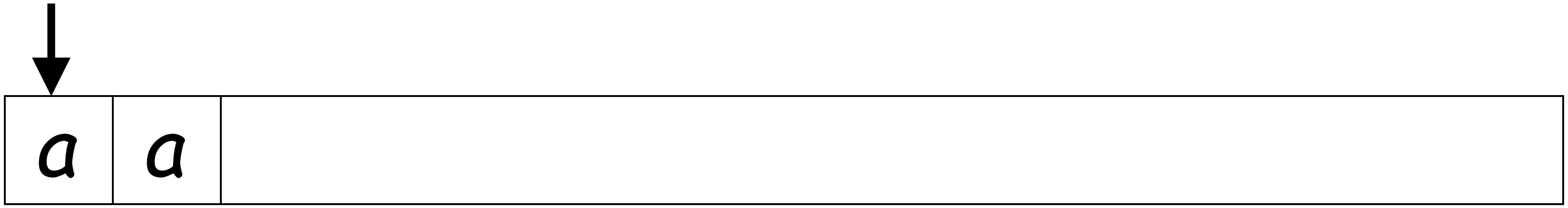


Second Choice

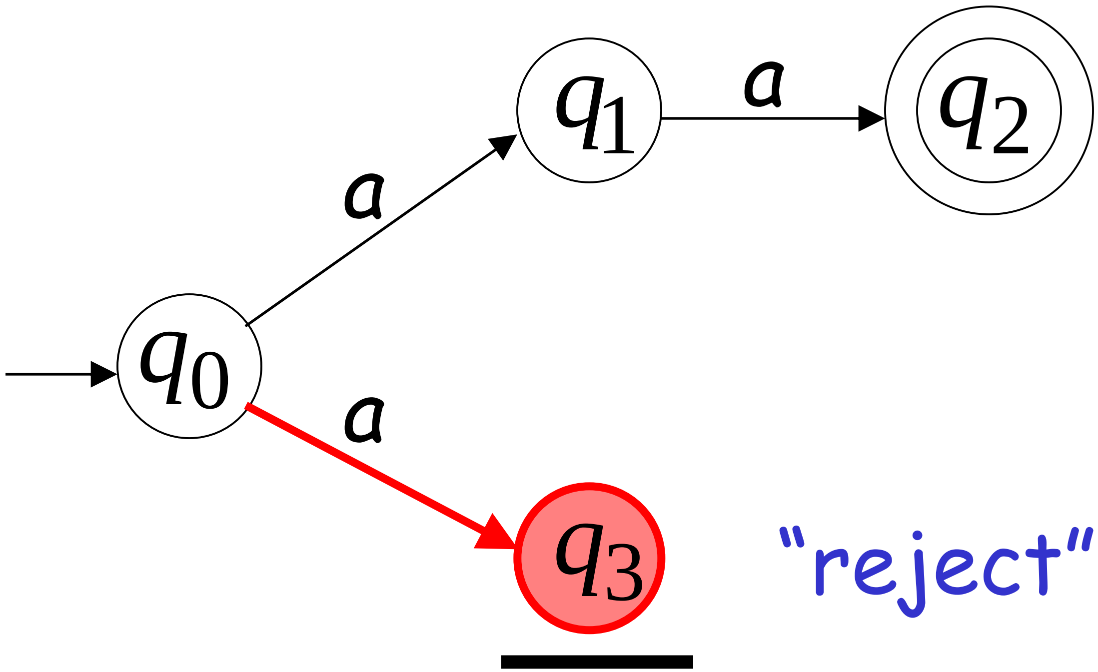


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



"reject"

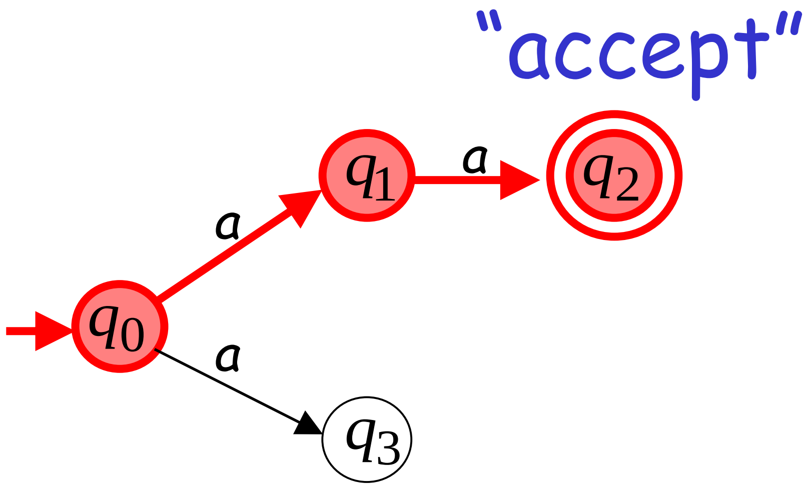
An NFA accepts a string:

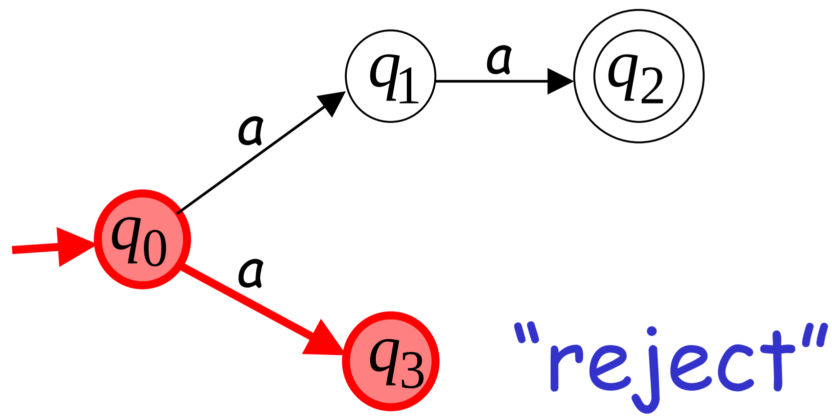
when there is a computation of the NFA that accepts the string

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

Example

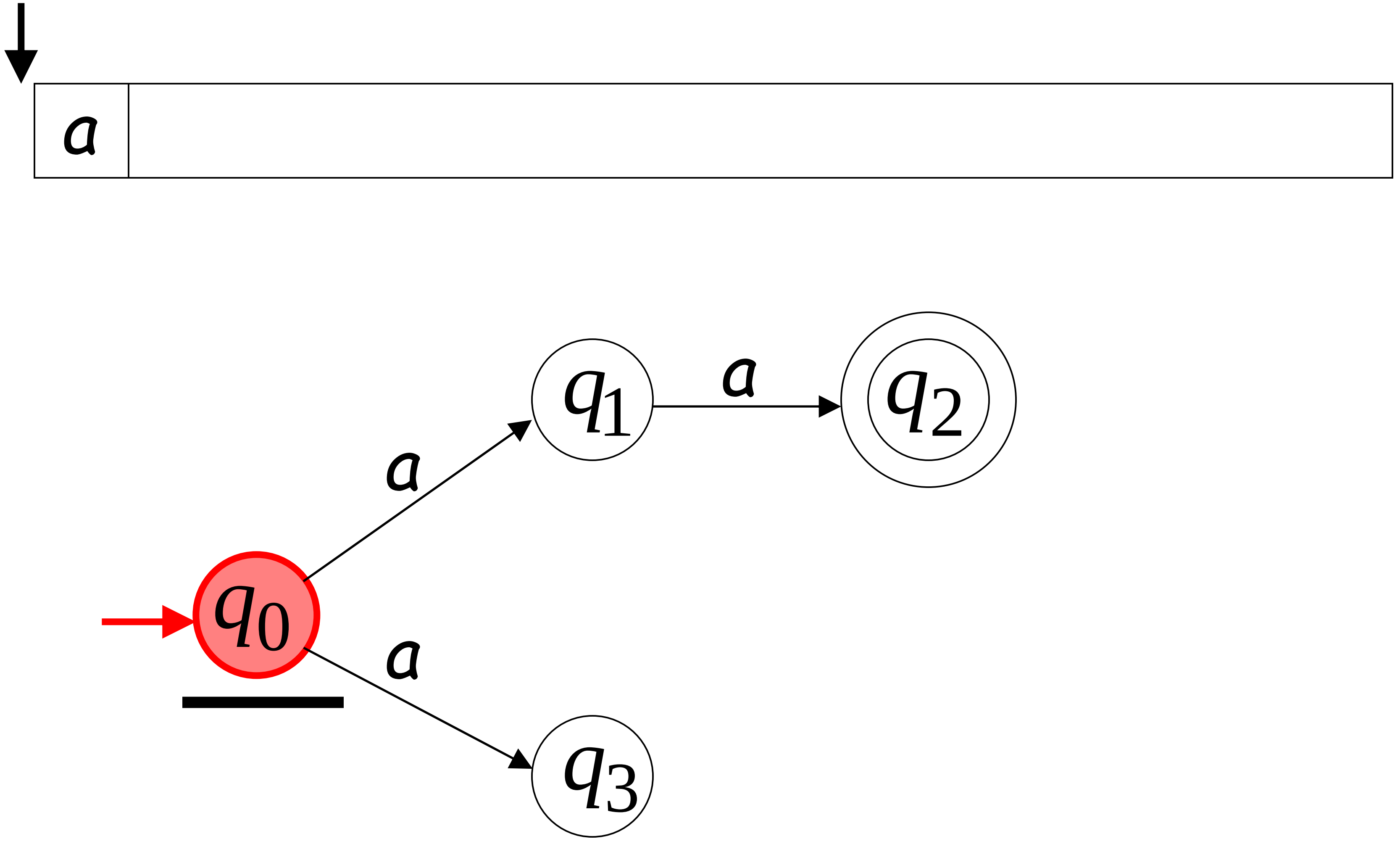
aa is accepted by the NFA:



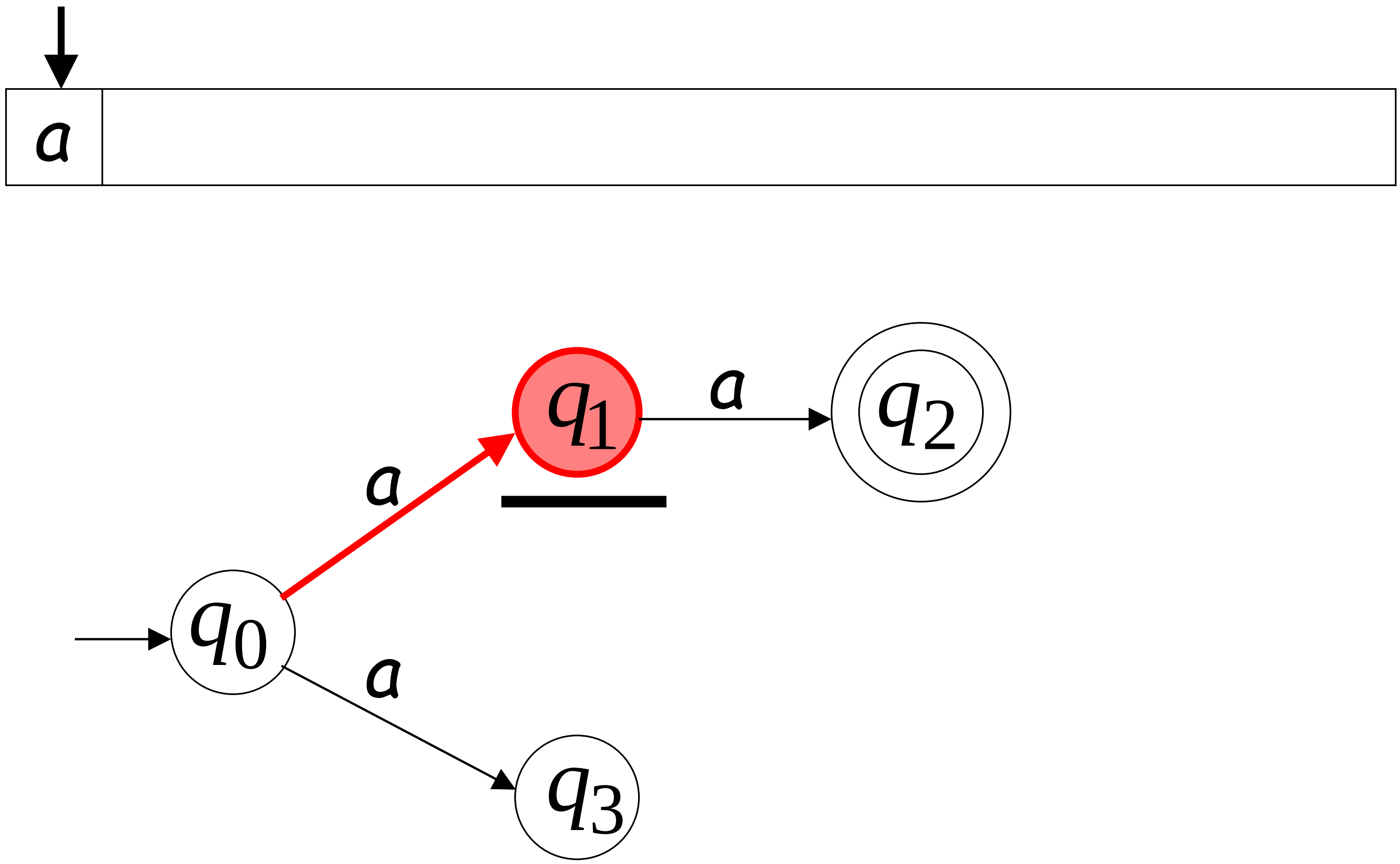


because this computation accepts aa

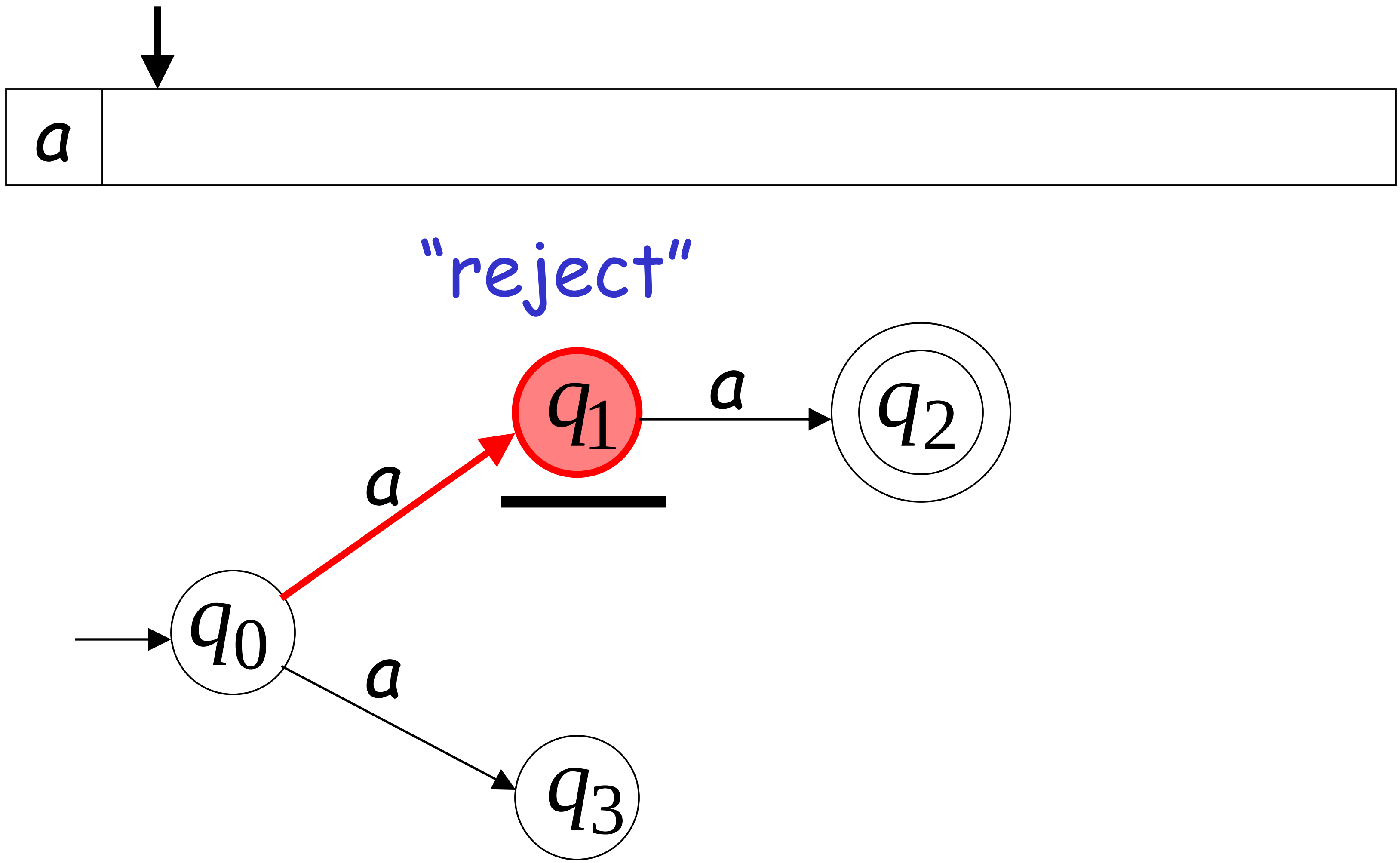
Rejection example



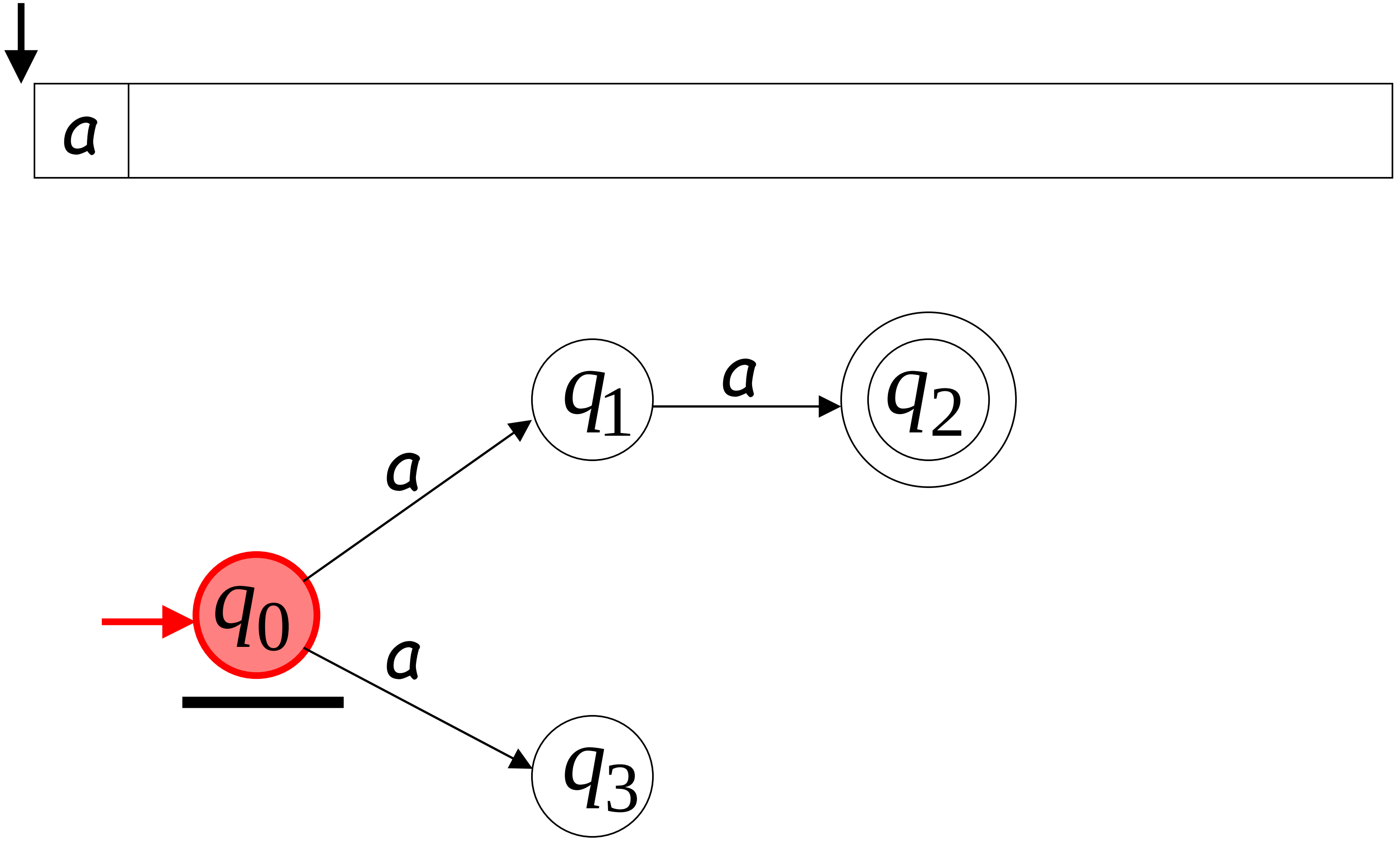
First Choice



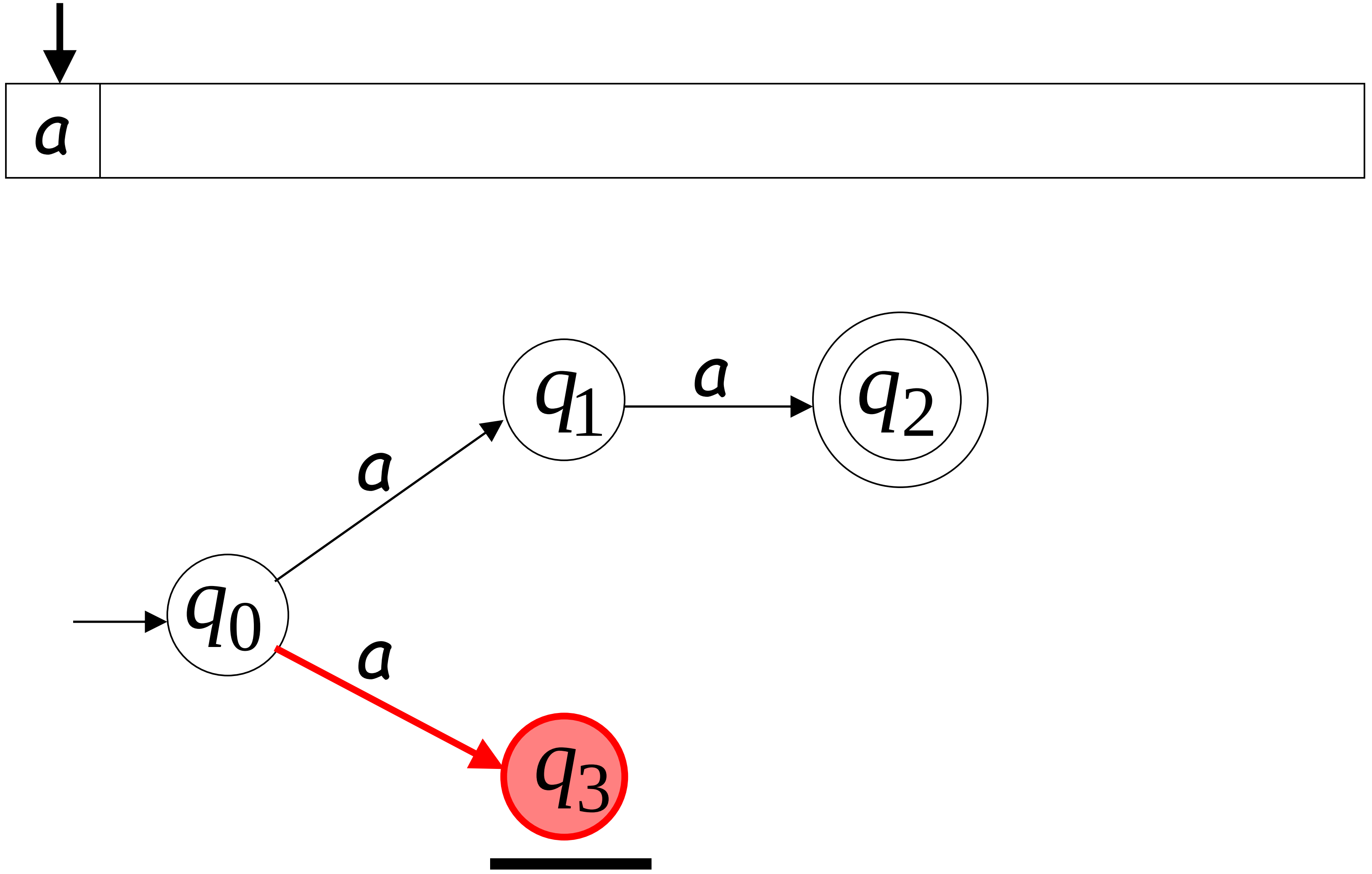
First Choice



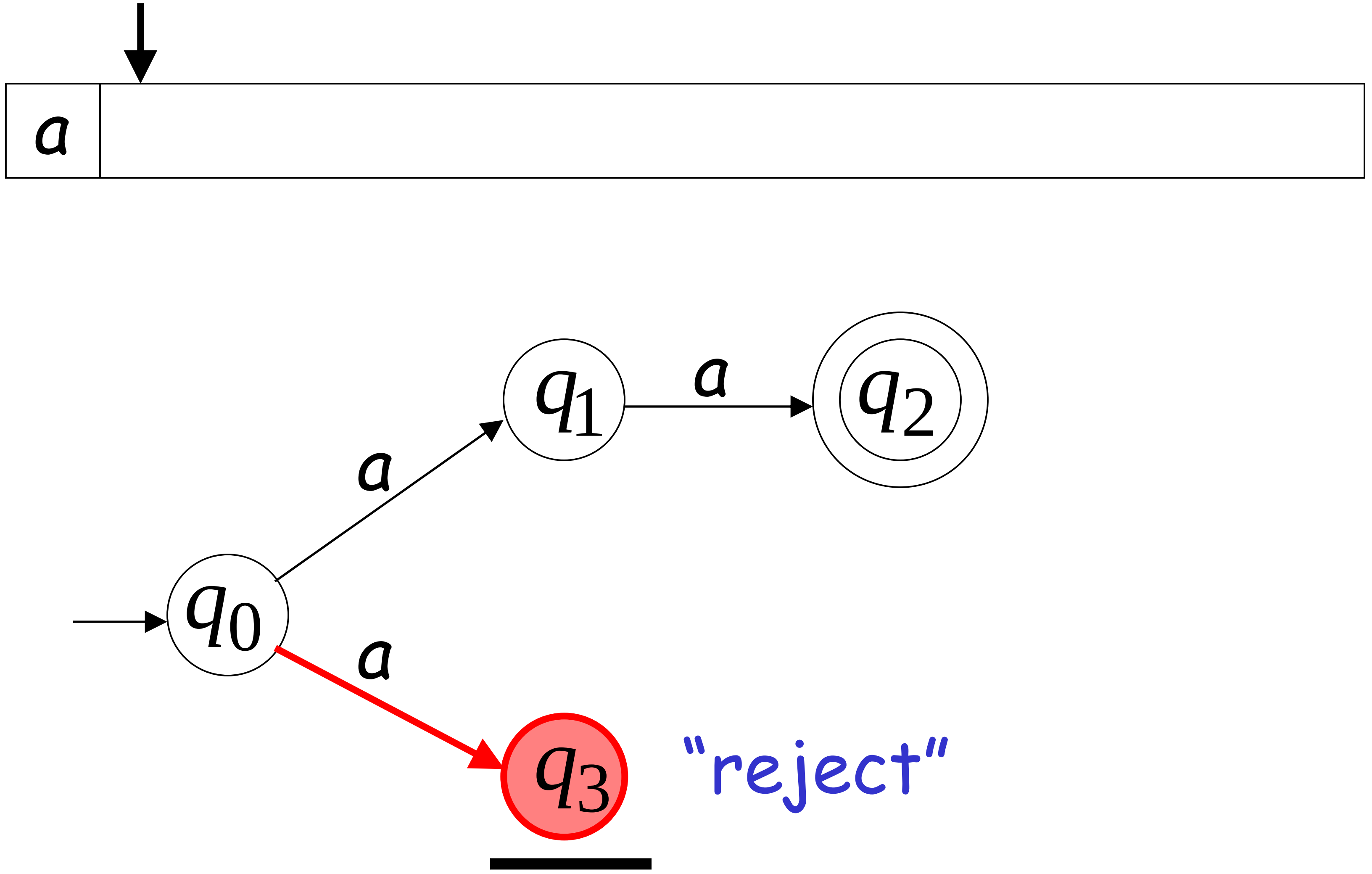
Second Choice



Second Choice



Second Choice



An NFA rejects a string:

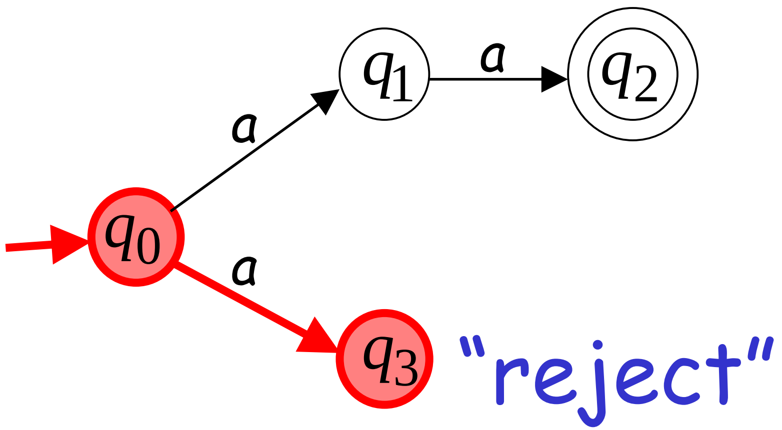
when there is no computation of the NFA that accepts the string

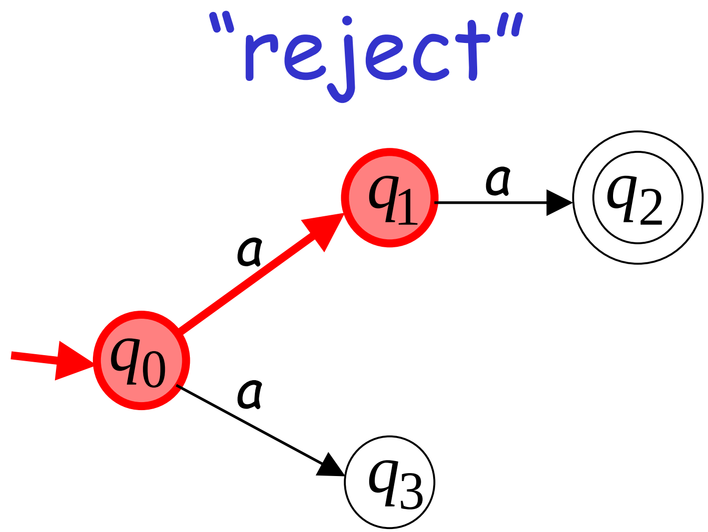
- All the input is consumed and the automaton is in a non final state

- The input cannot be consumed

Example

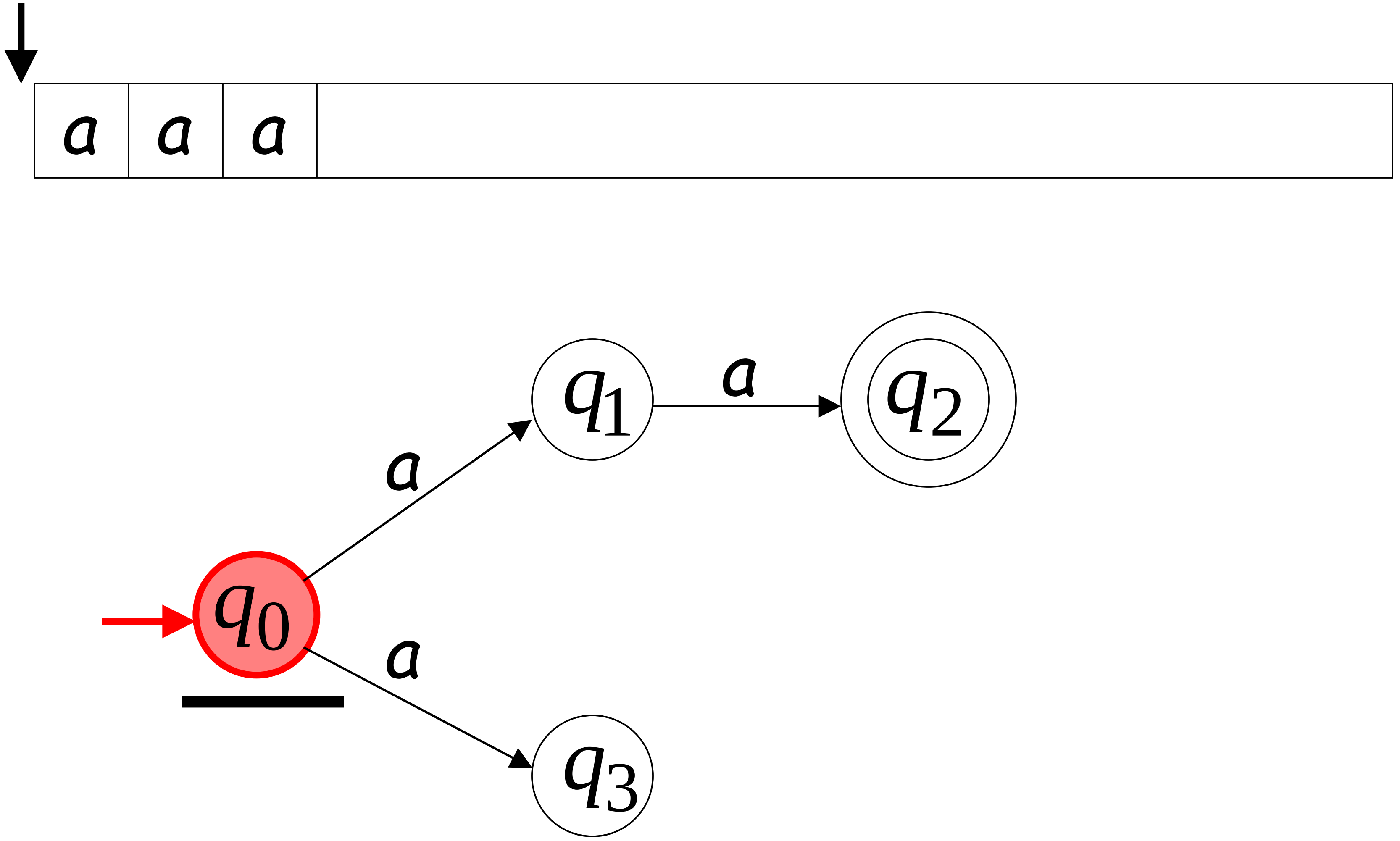
a is rejected by the NFA:



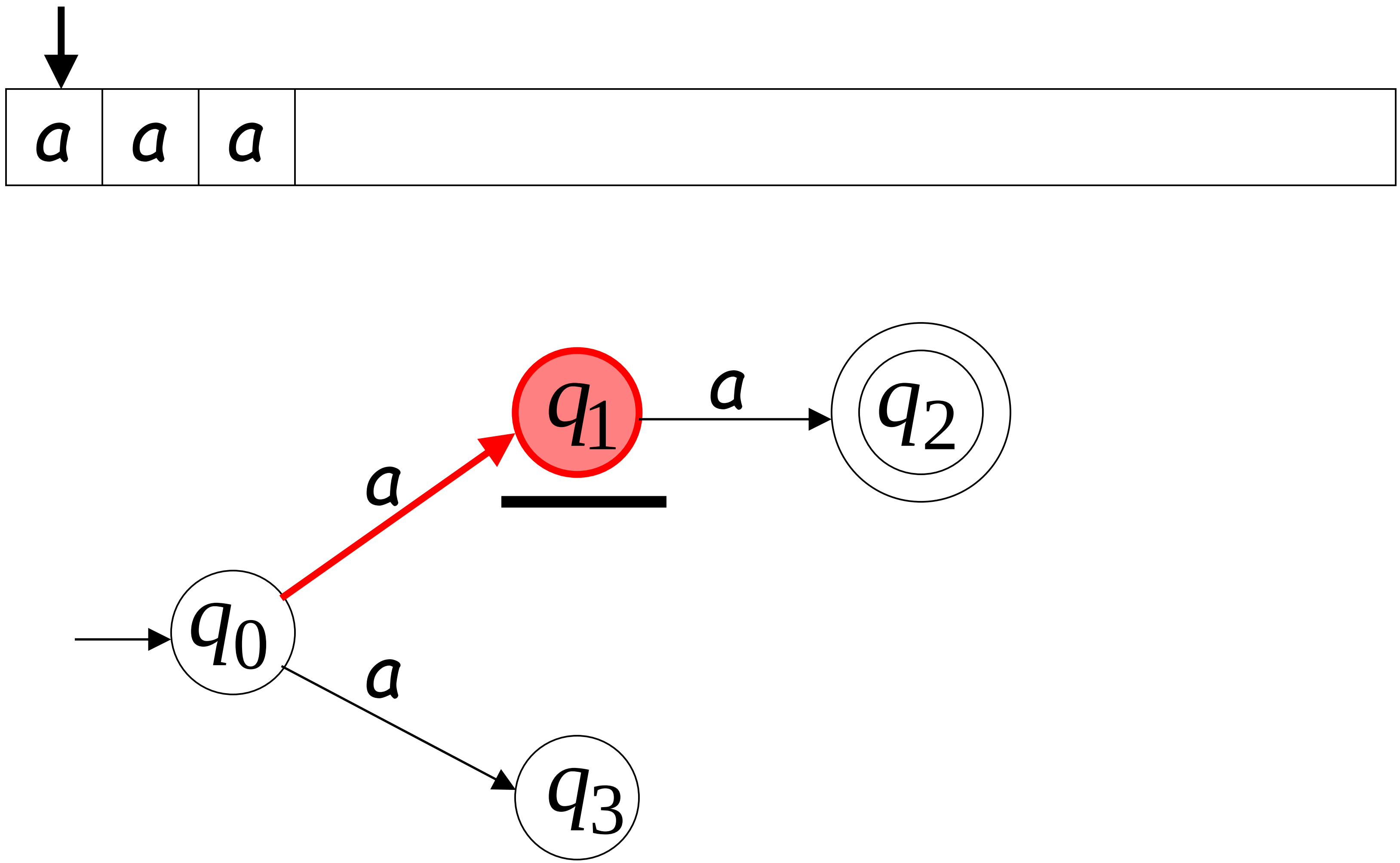


All possible computations lead to rejection

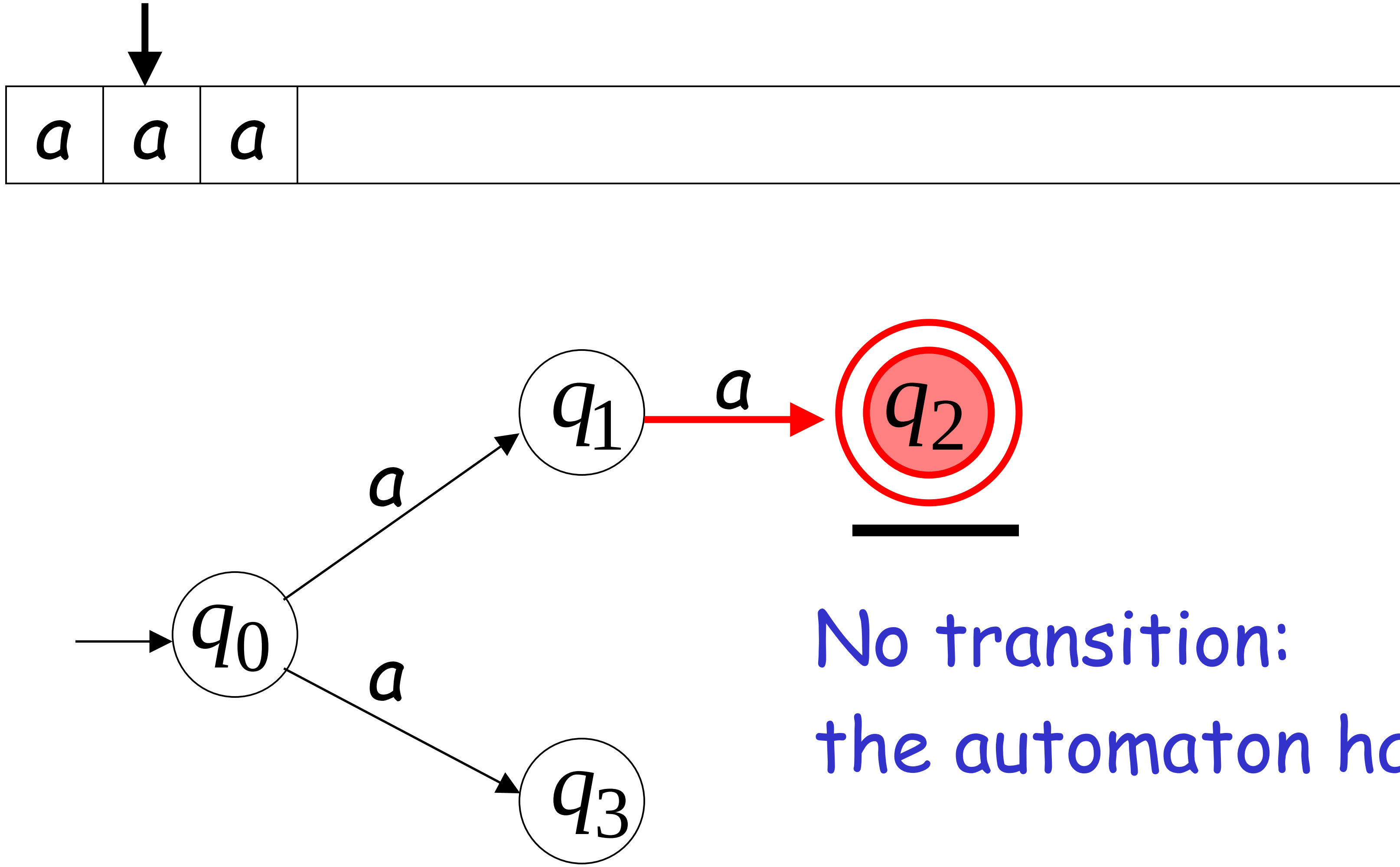
Rejection example



First Choice



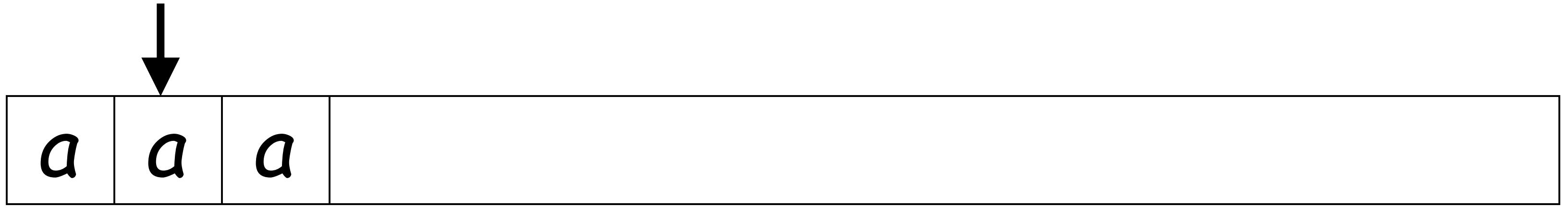
First Choice



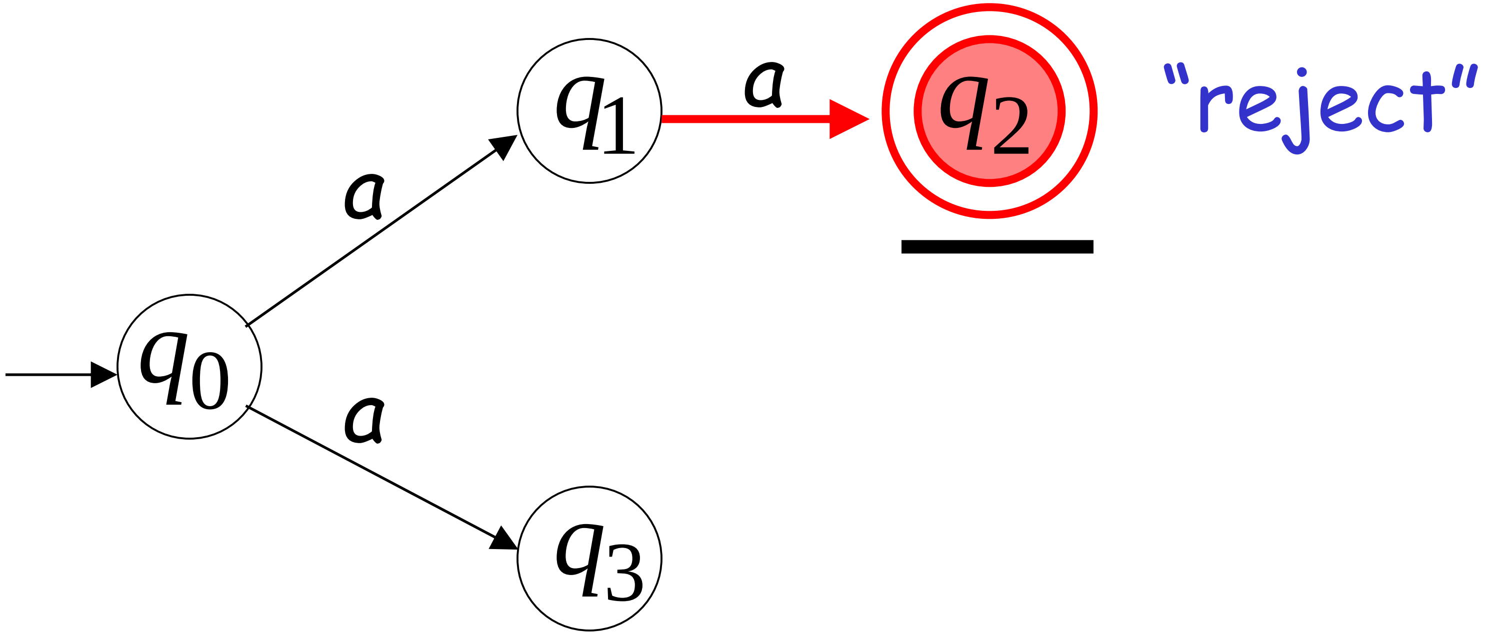
No transition:

the automaton hangs

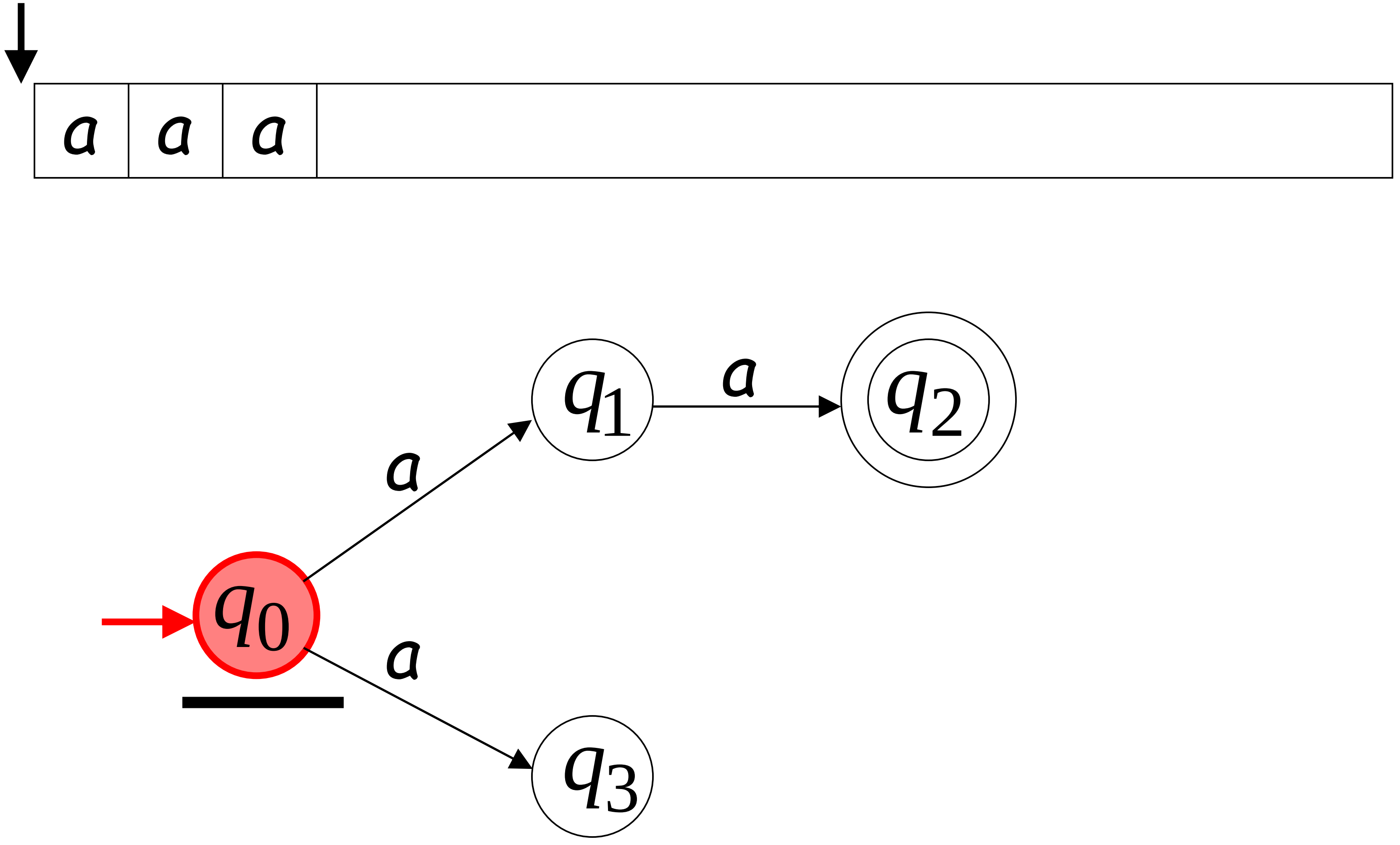
First Choice



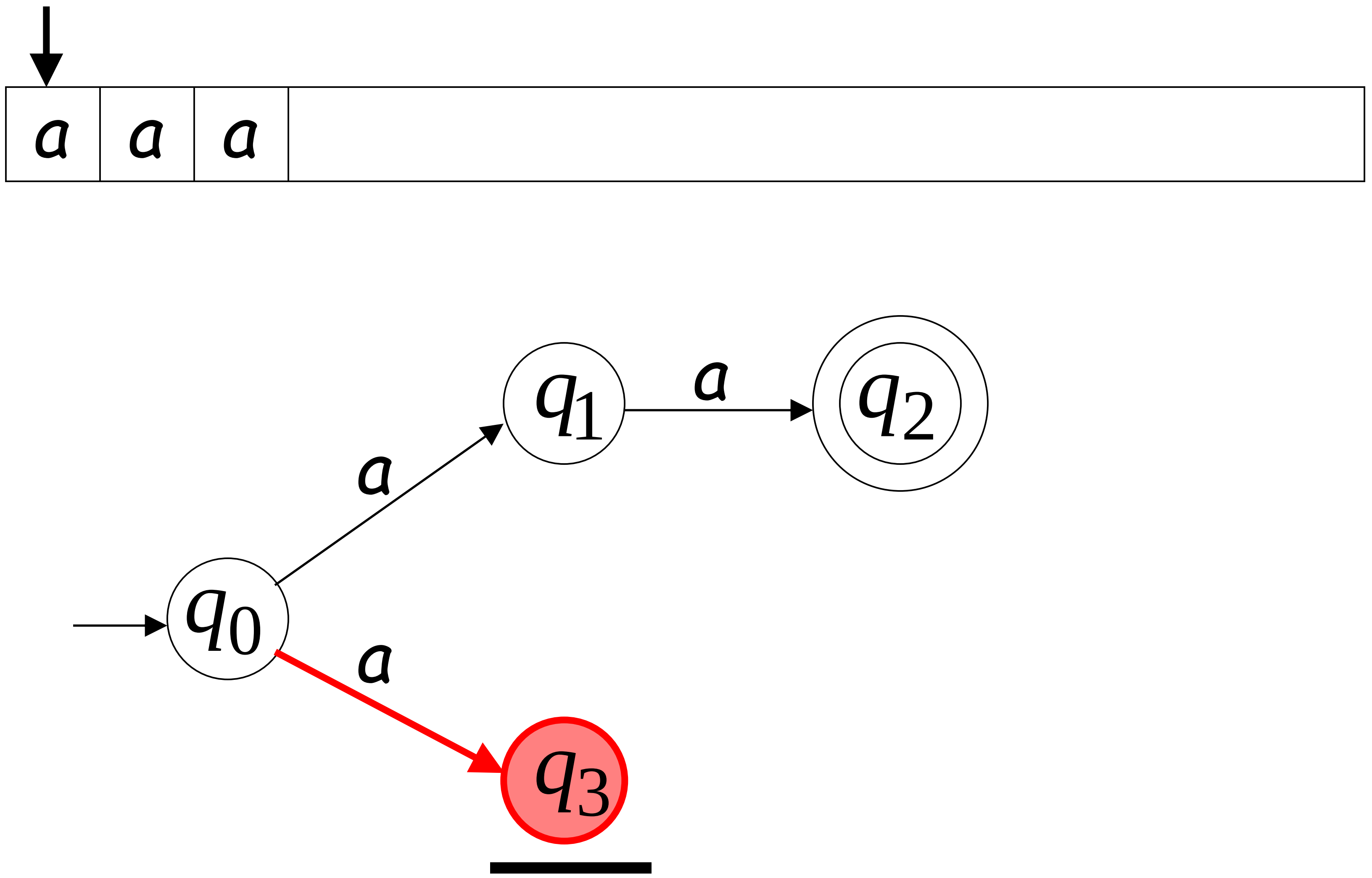
Input cannot be consumed



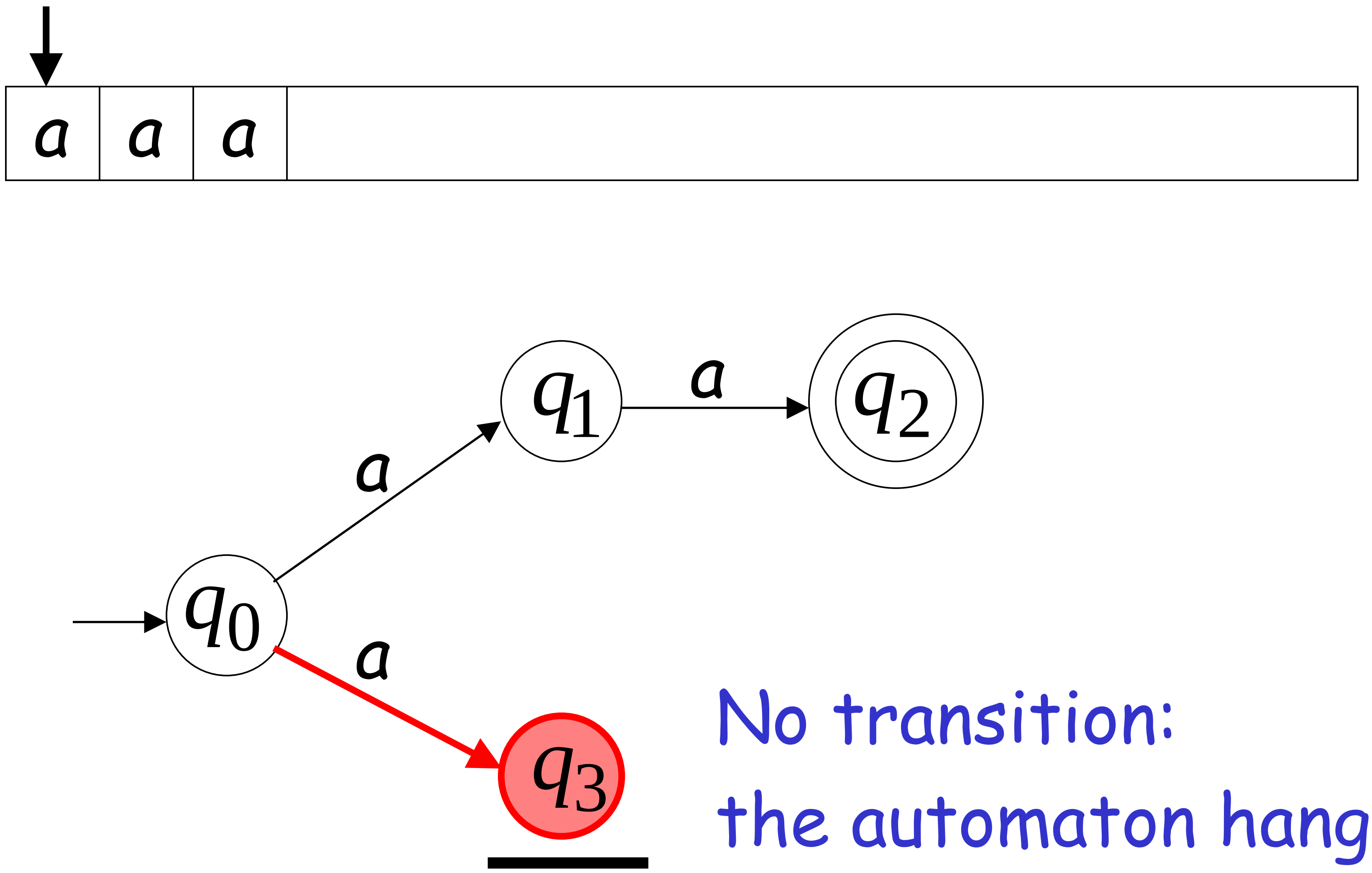
Second Choice



Second Choice

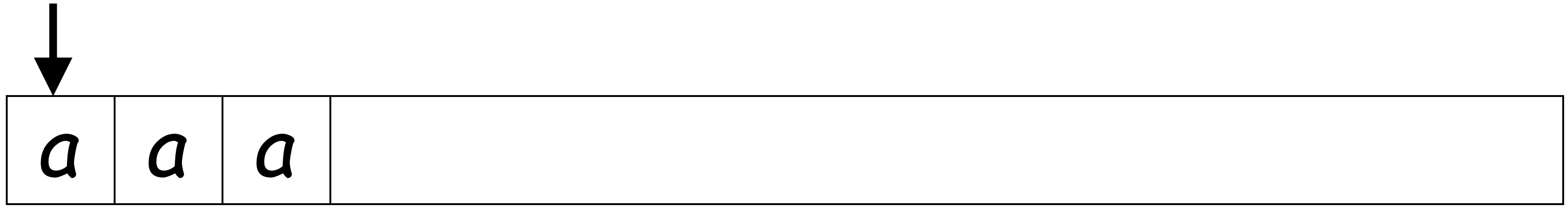


Second Choice

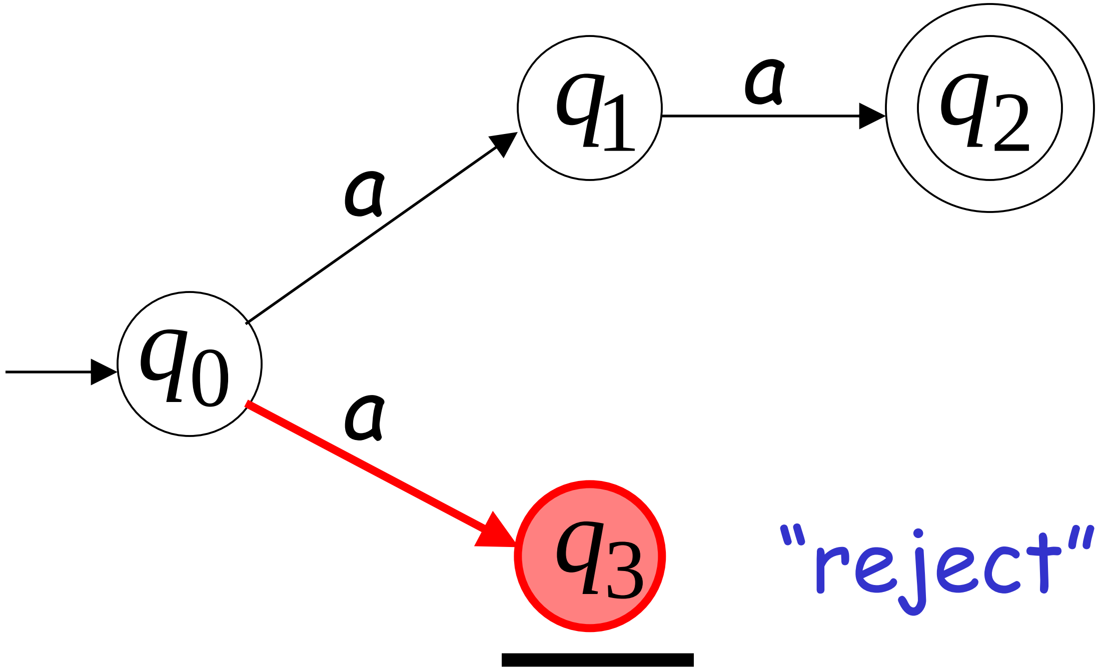


No transition:
the automaton hangs

Second Choice

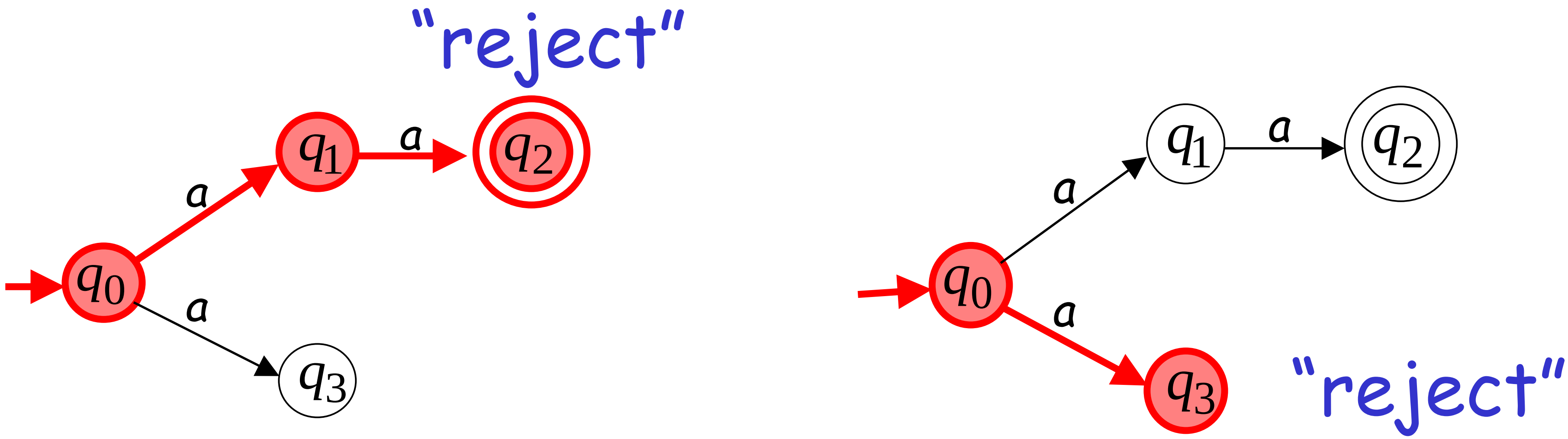


Input cannot be consumed



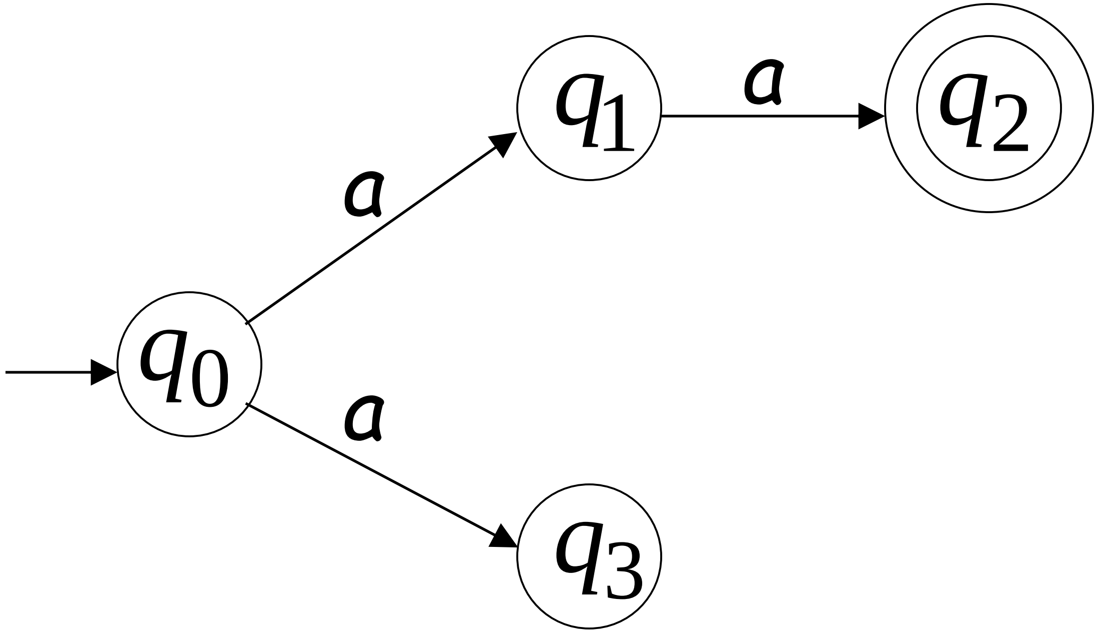
"reject"

aaa is rejected by the NFA:

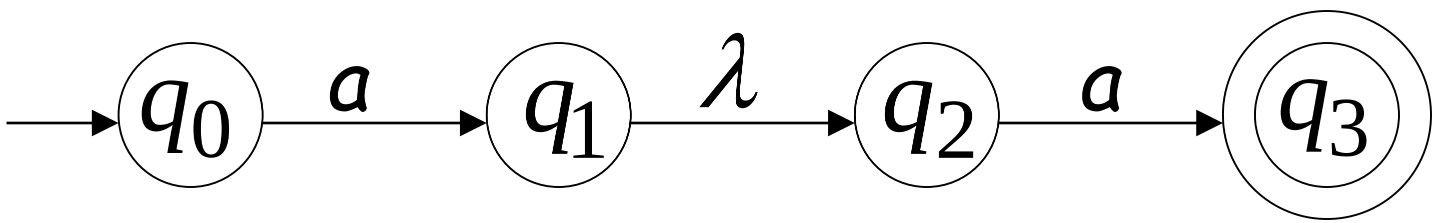


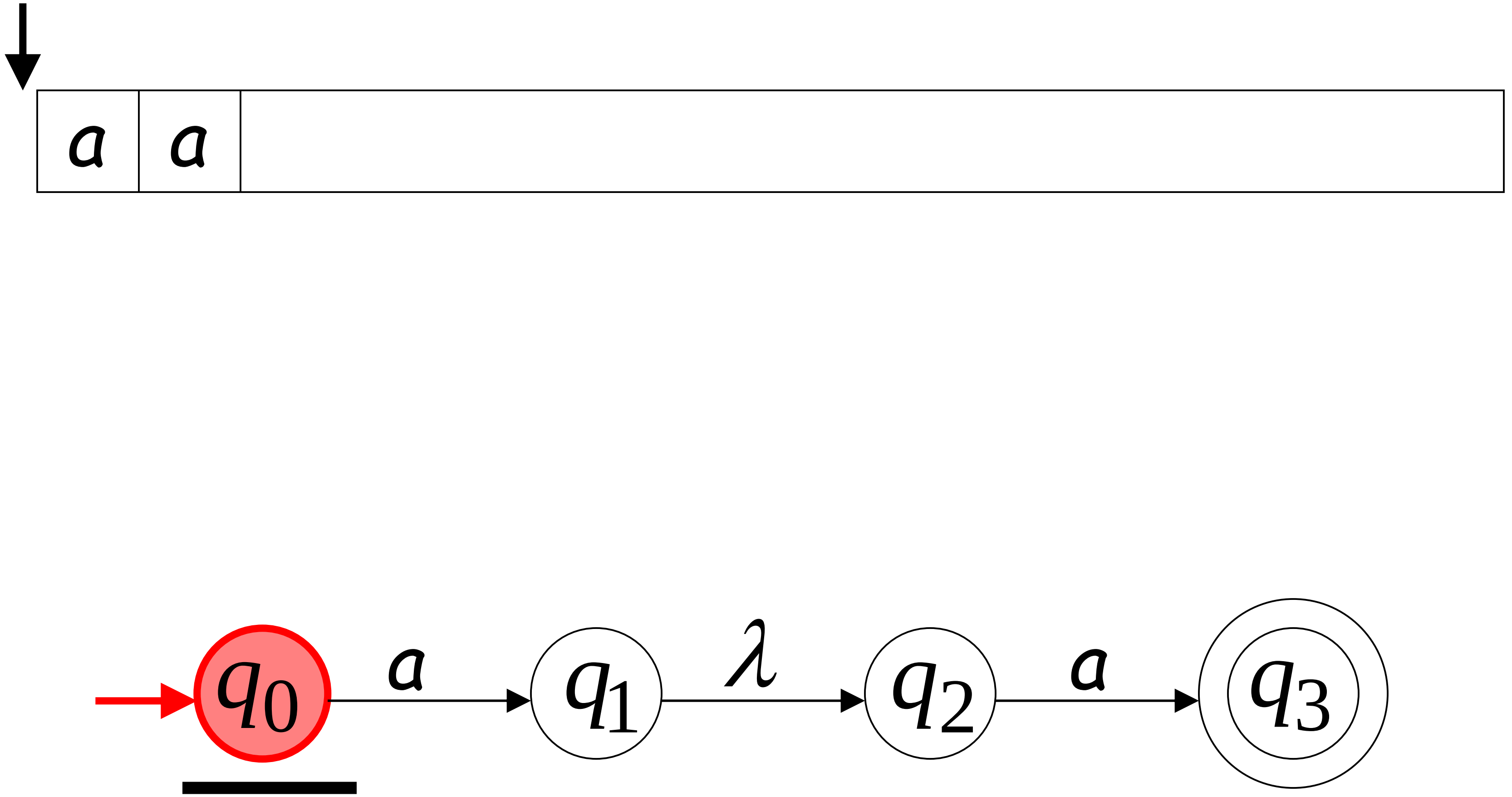
All possible computations lead to rejection

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

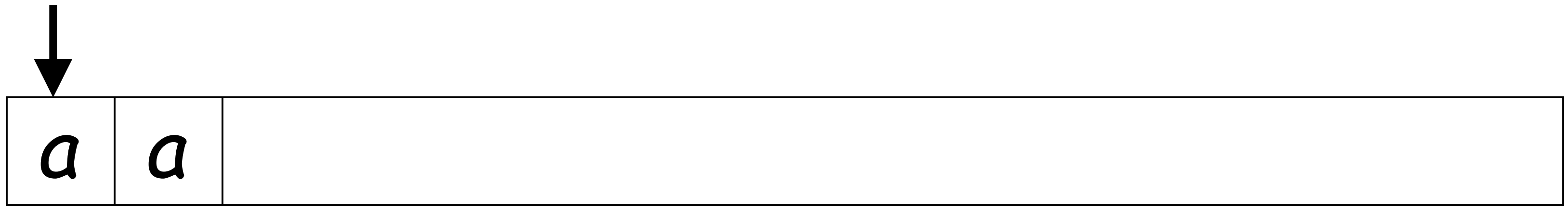


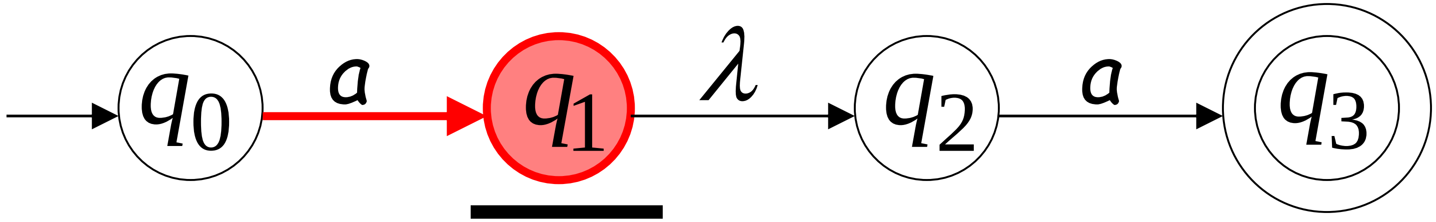
Lambda Transitions



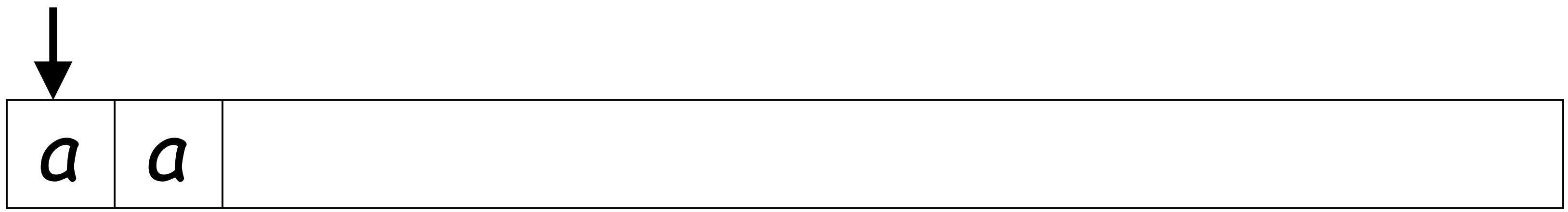


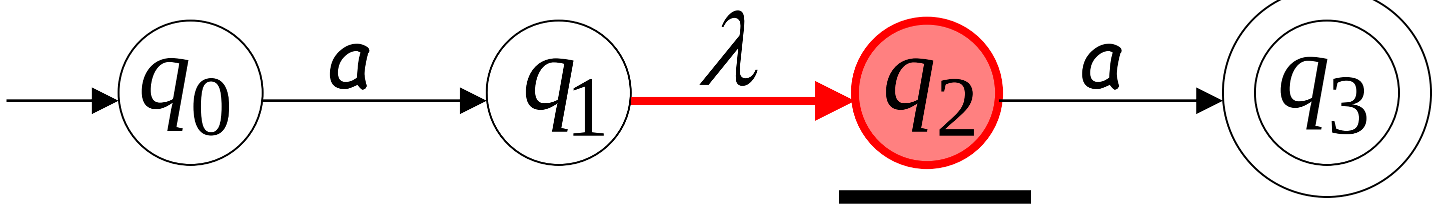
a
a
q_0
a
q_1
λ
q_2
a
q_3

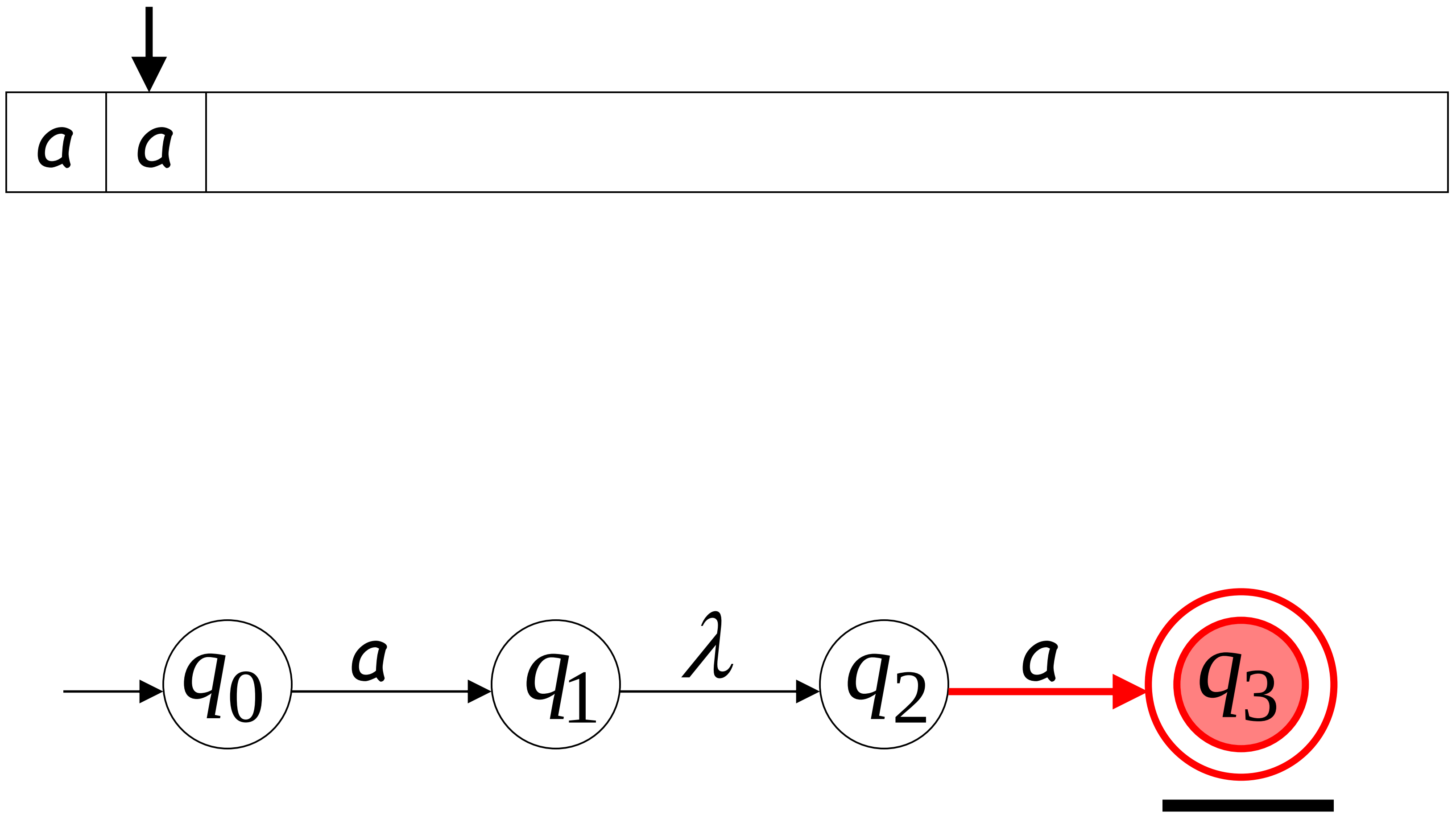

a
a


q_0
a
q_1
λ
q_2
a
q_3

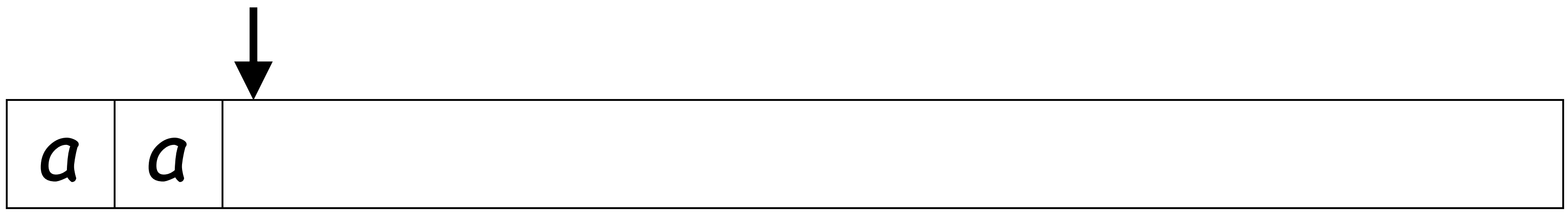
(read head doesn't move)



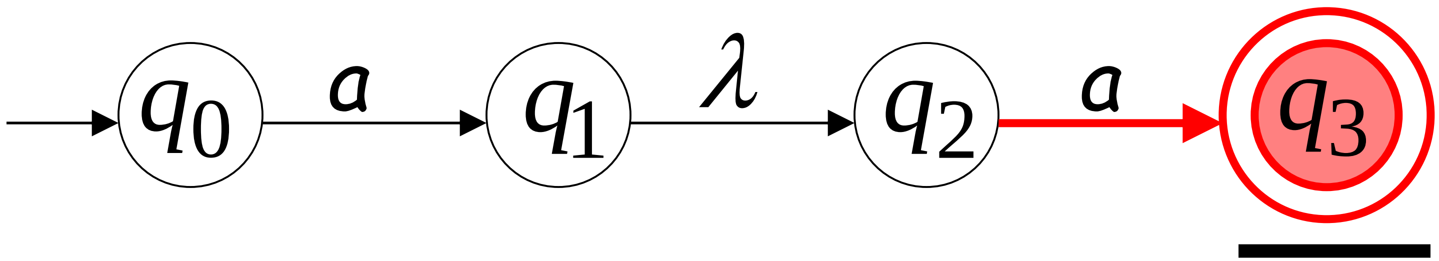



a
a
q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

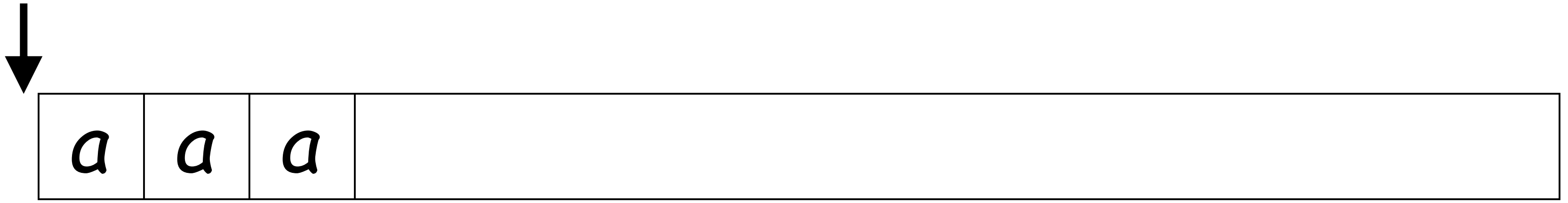


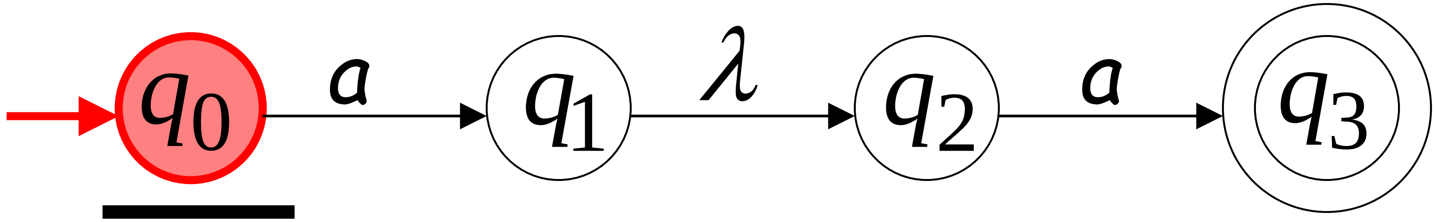
“accept”

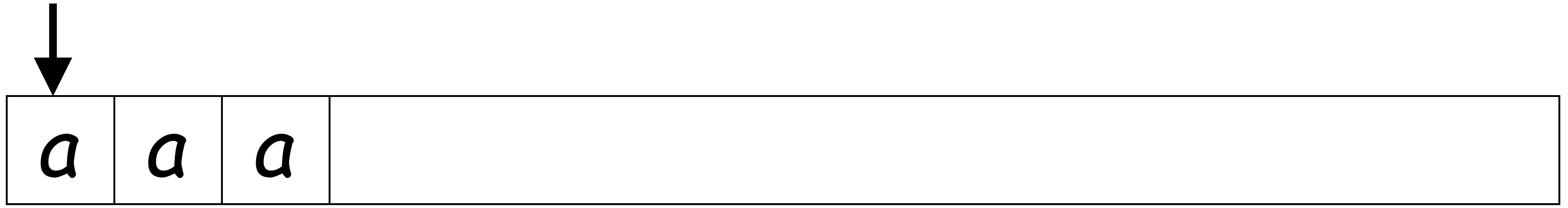


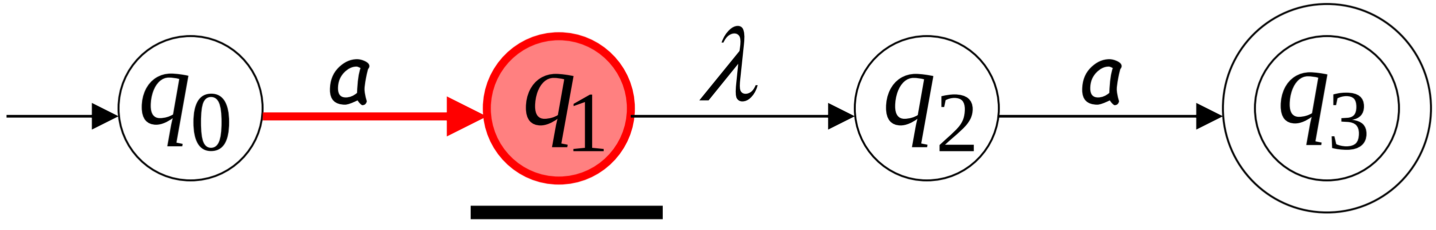
String aa is accepted

Rejection Example

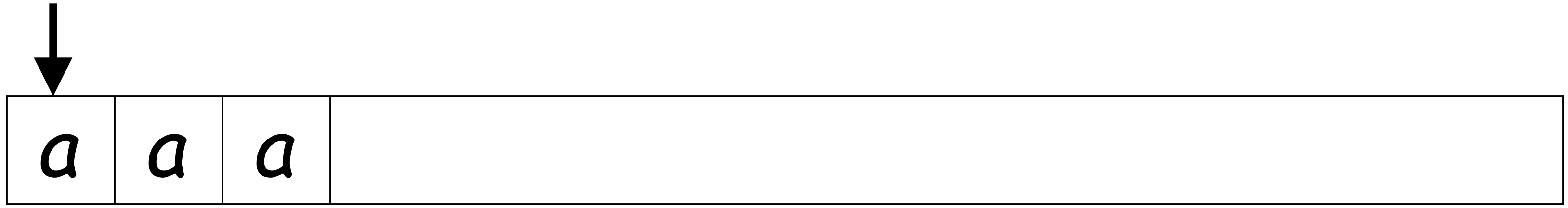


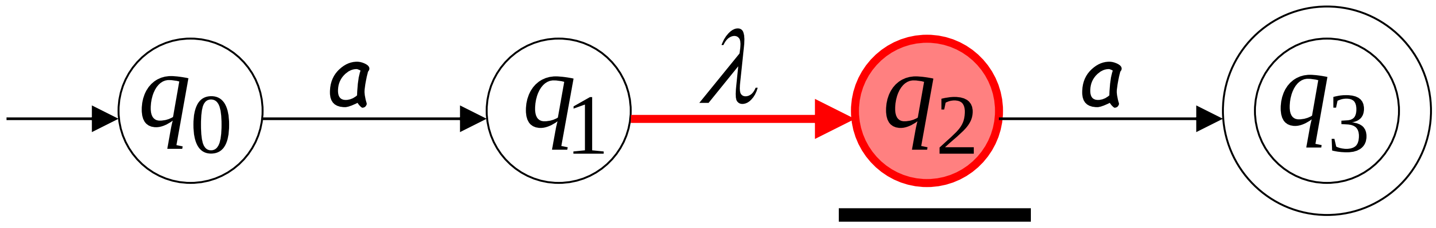


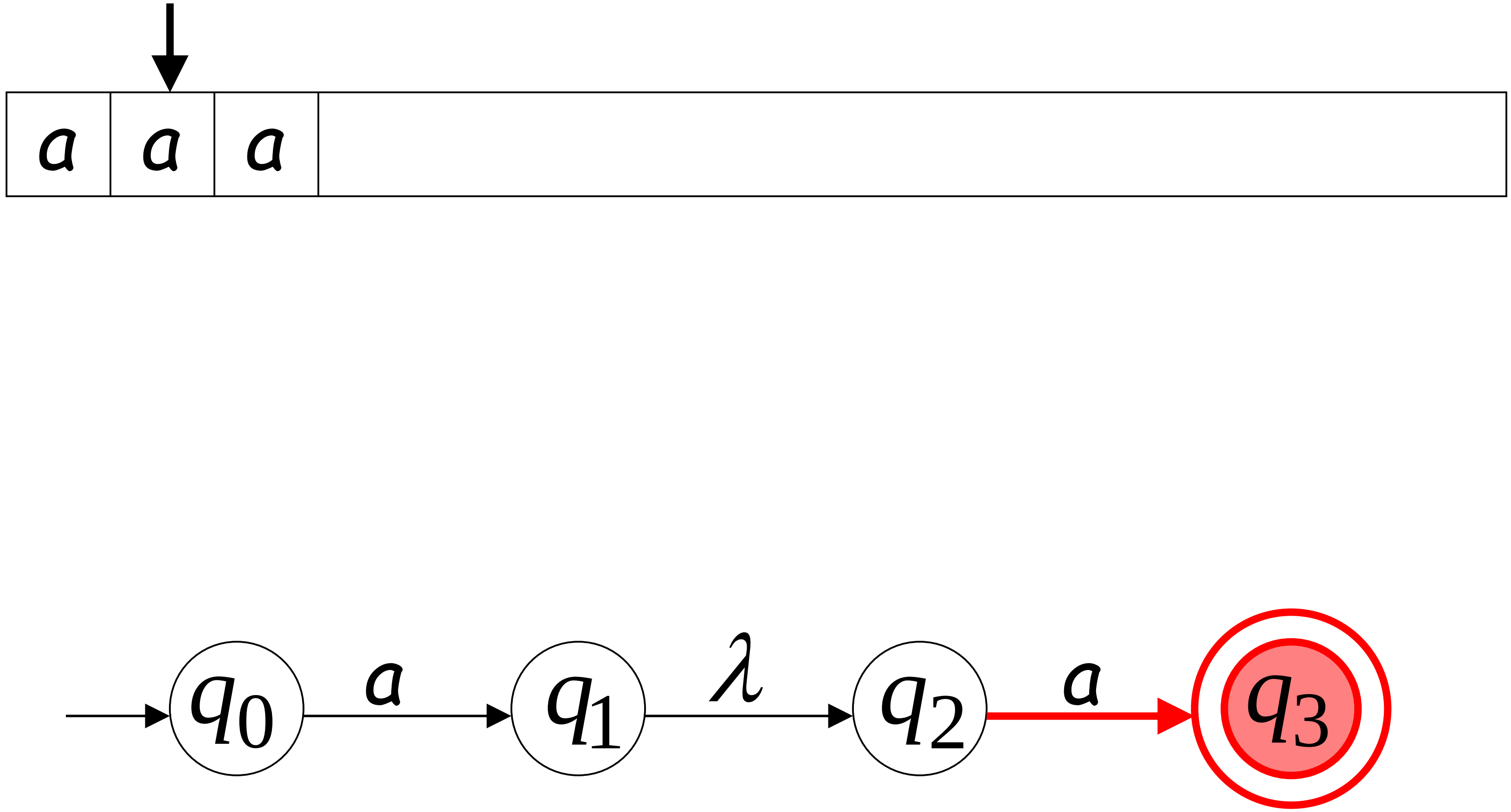

a
a
a


q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)

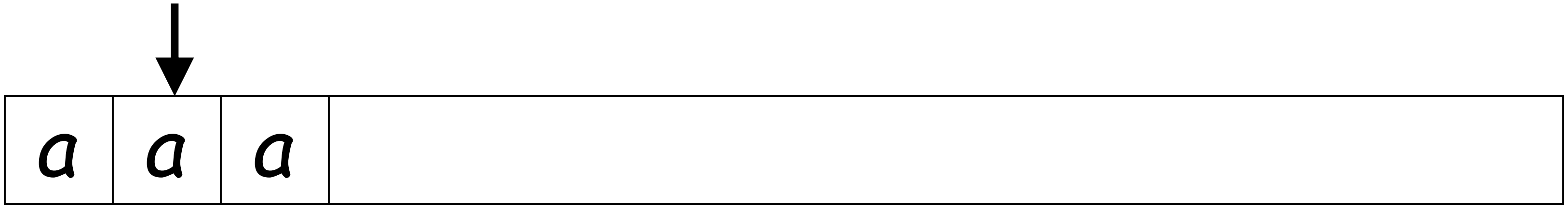


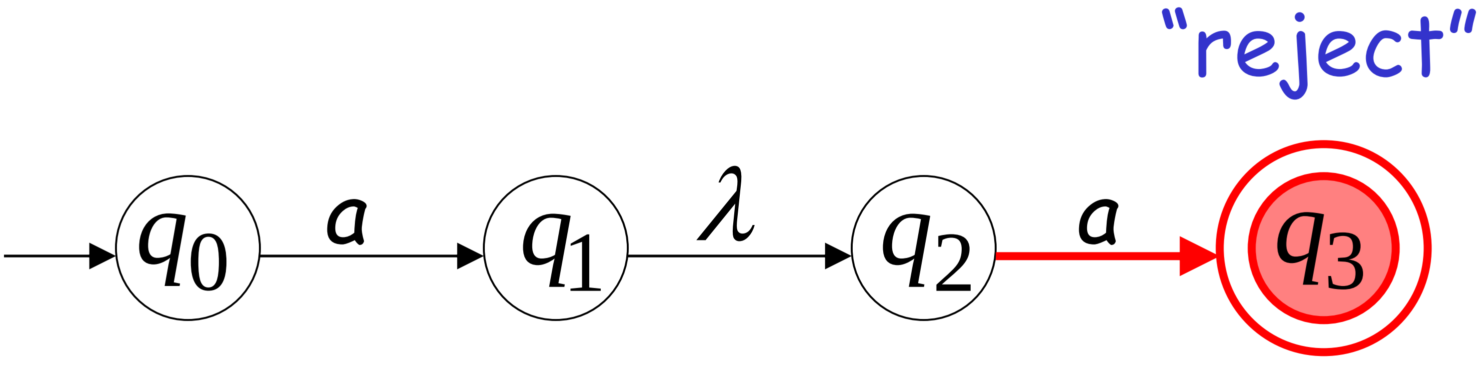




No transition:
the automaton hangs

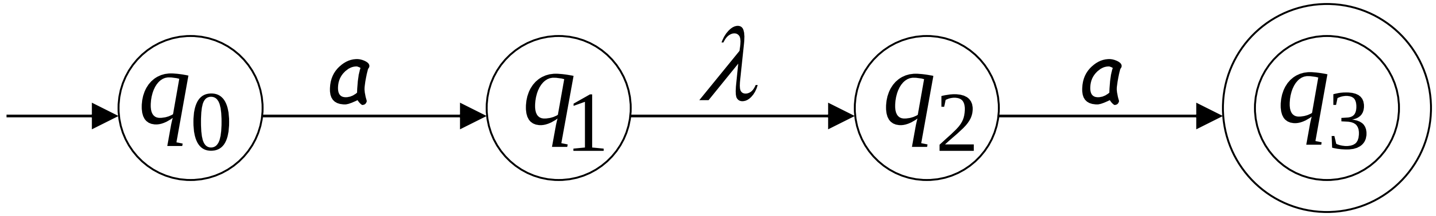
Input cannot be consumed



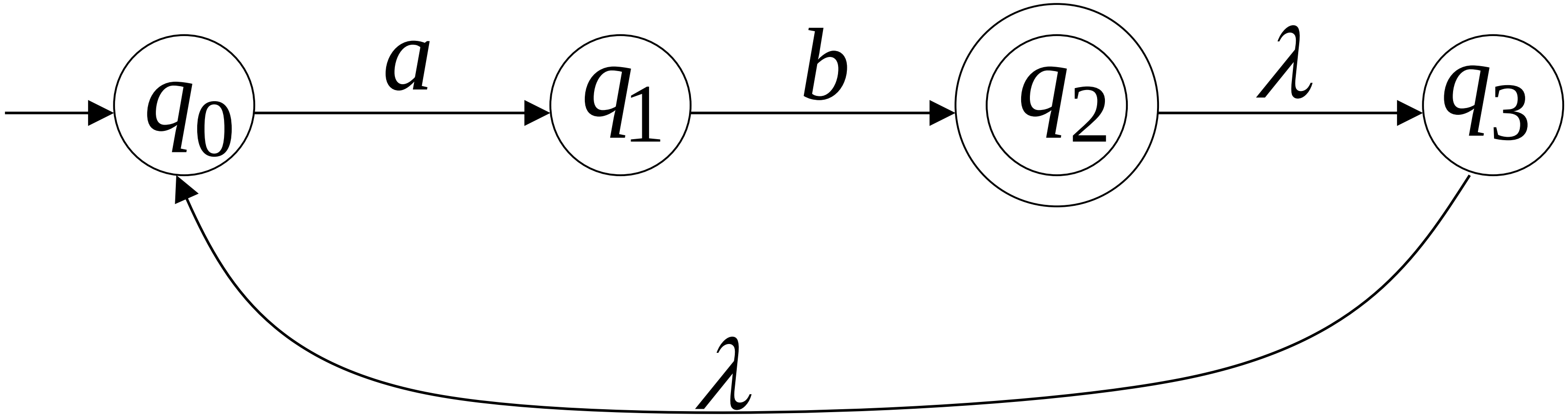


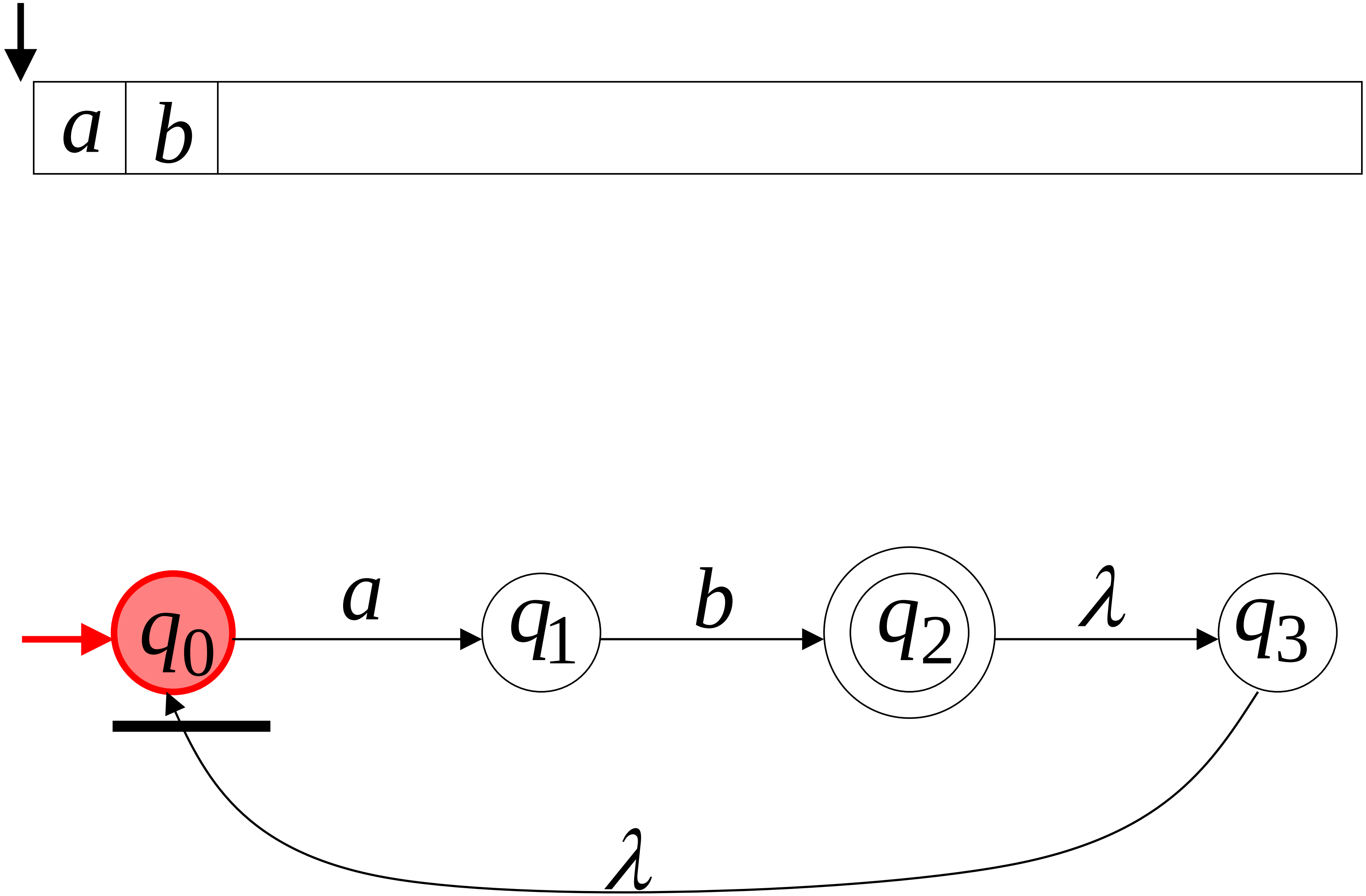
String aaa is rejected

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

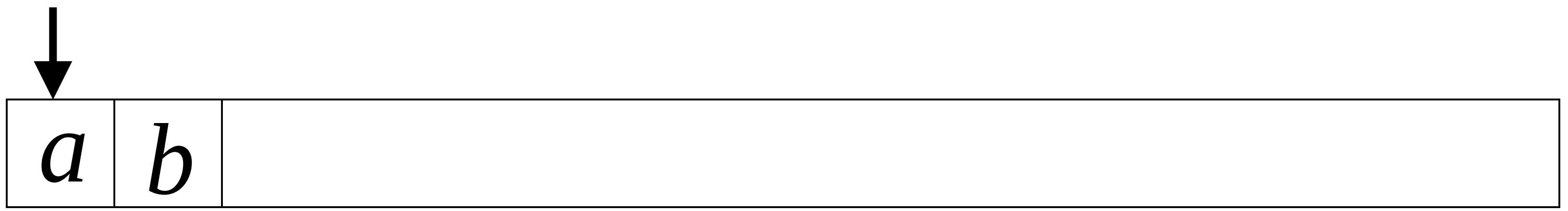


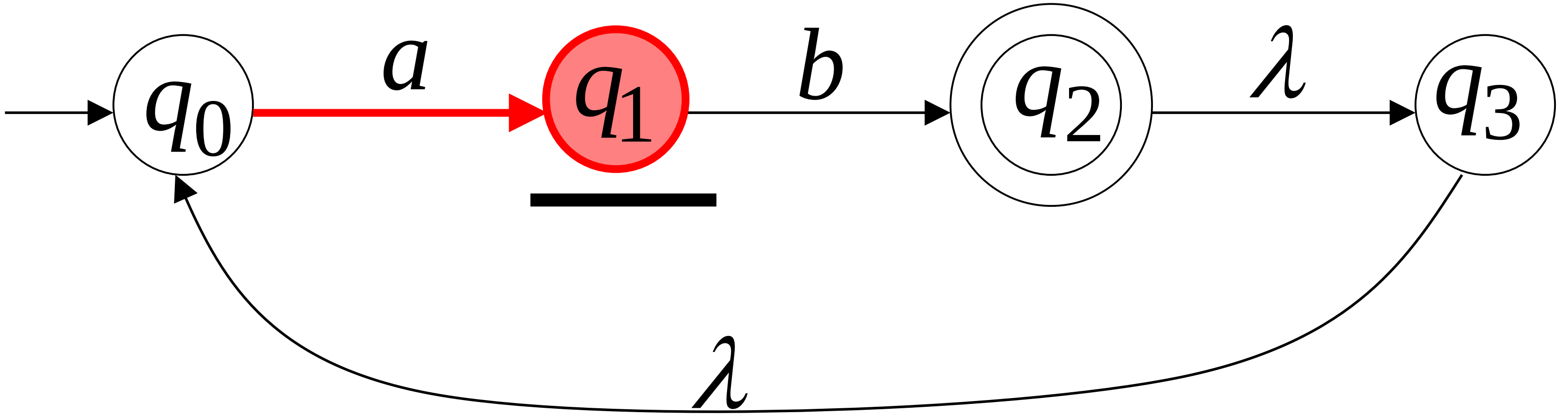
Another NFA Example

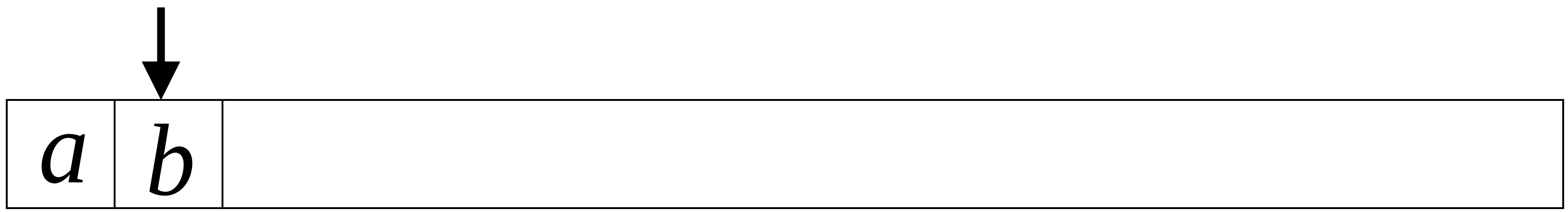


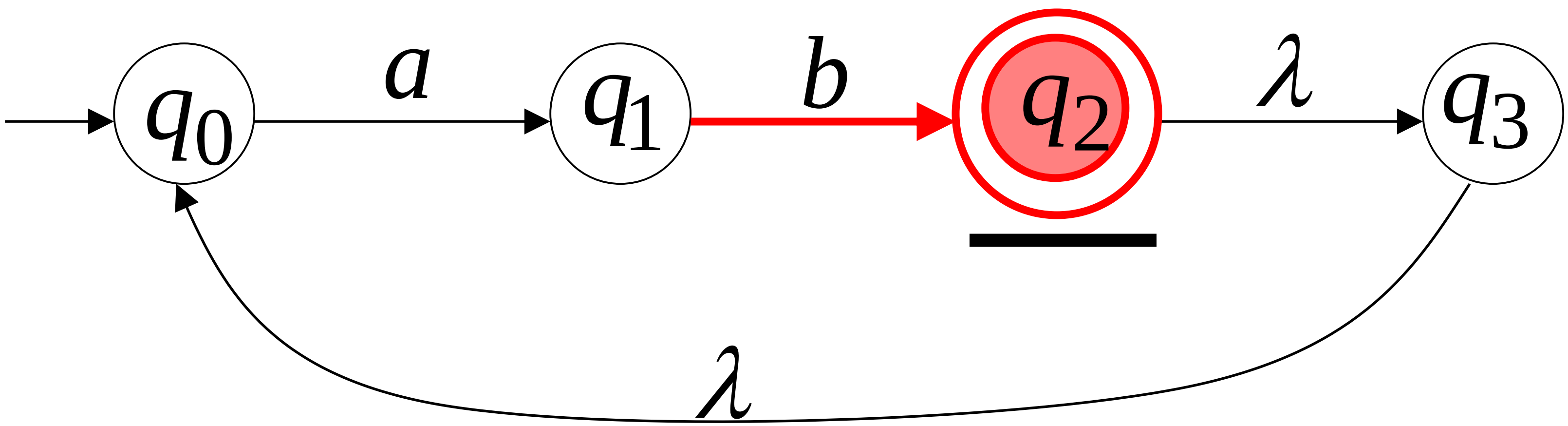


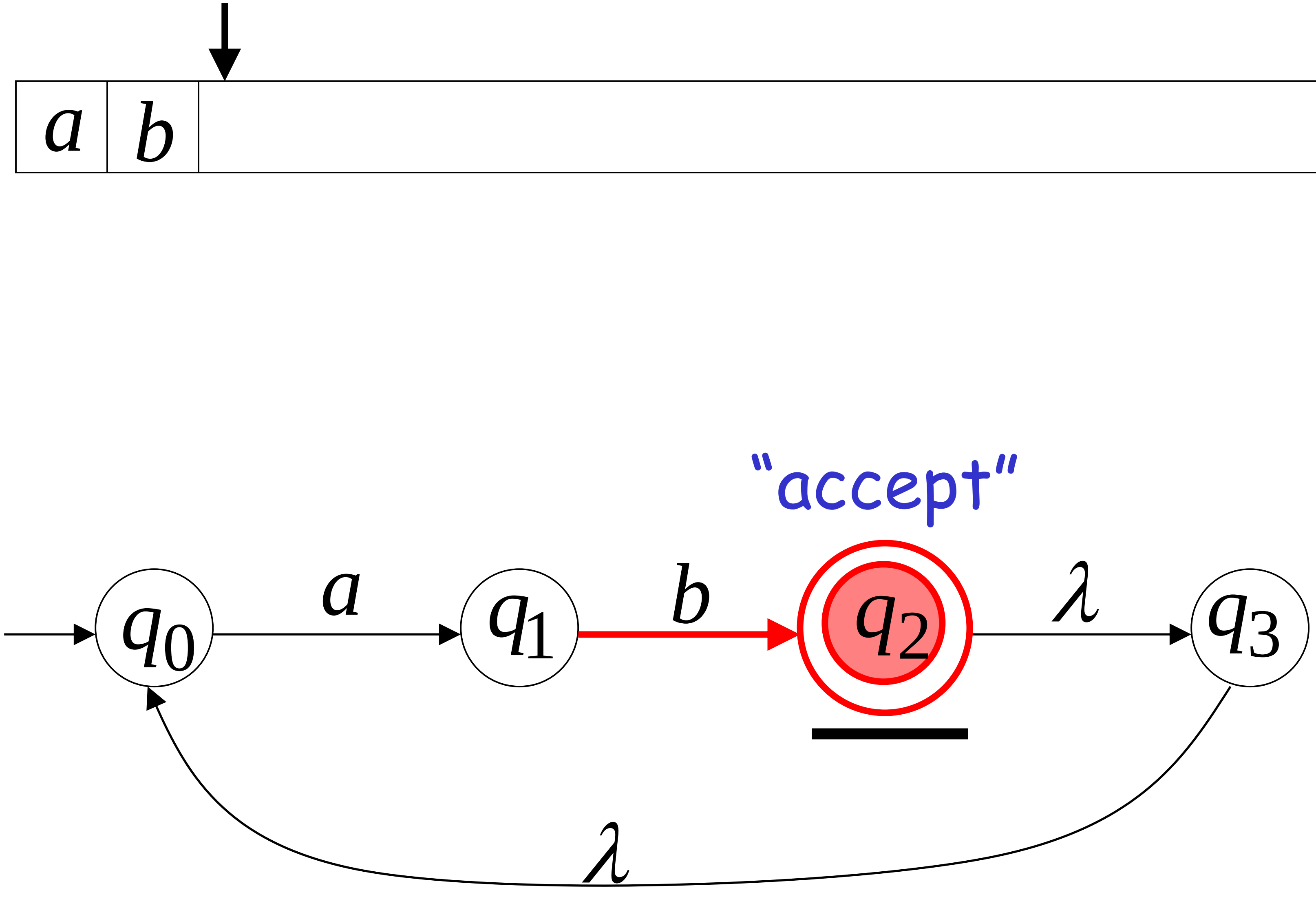
a
b
a
q0
q1
b
q2
λ
q3
λ


a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

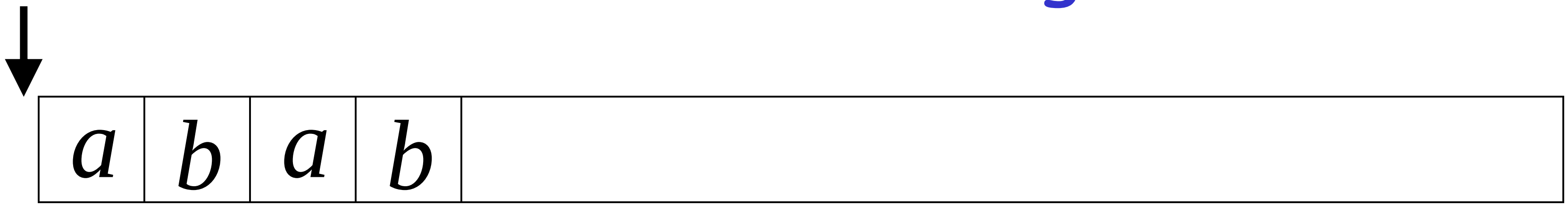

a
b

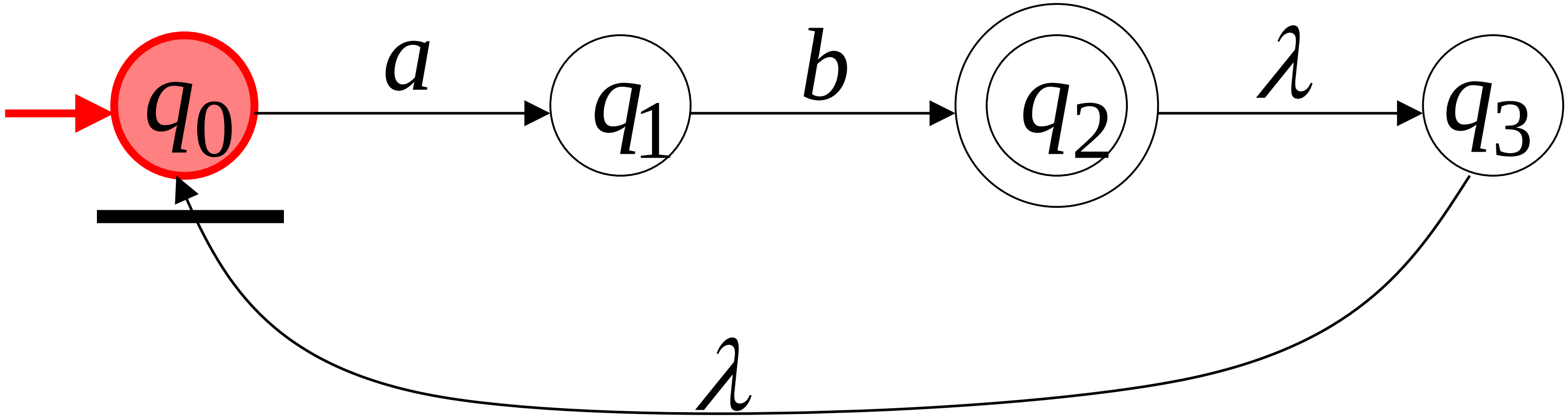

q_0
a
q_1
b
q_2
λ
q_3
λ

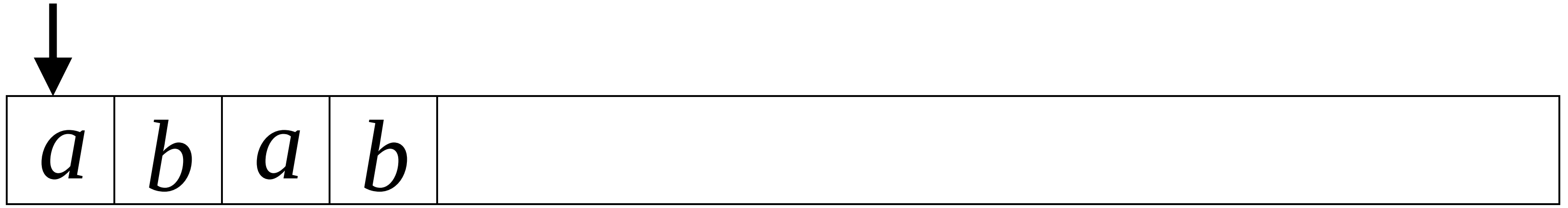


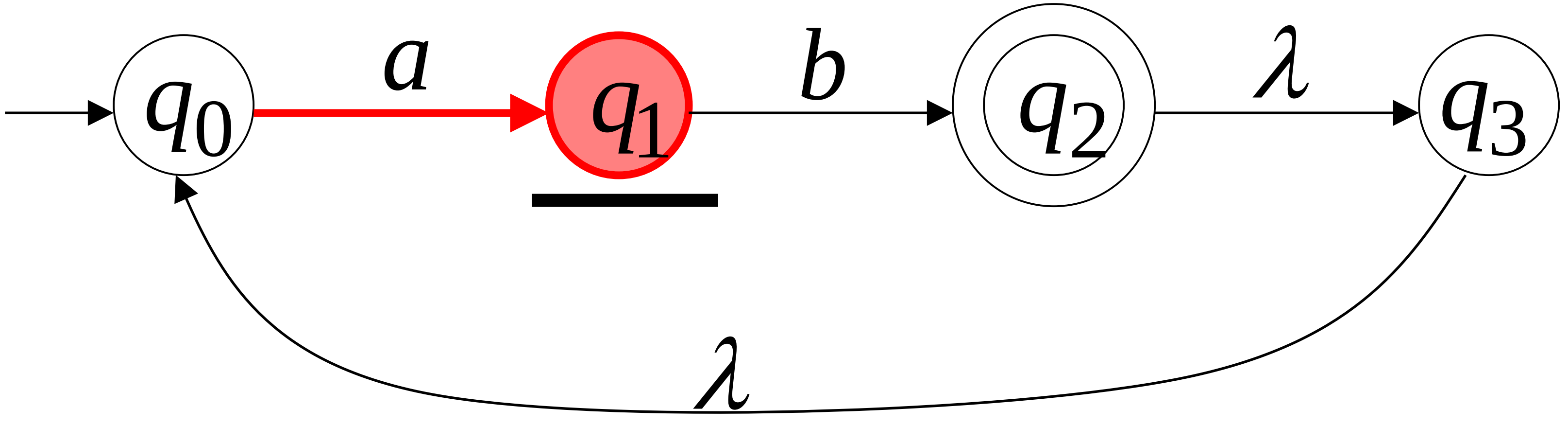
a
b
"accept"
q_0
q_1
q_2
q_3
a
b
λ
λ

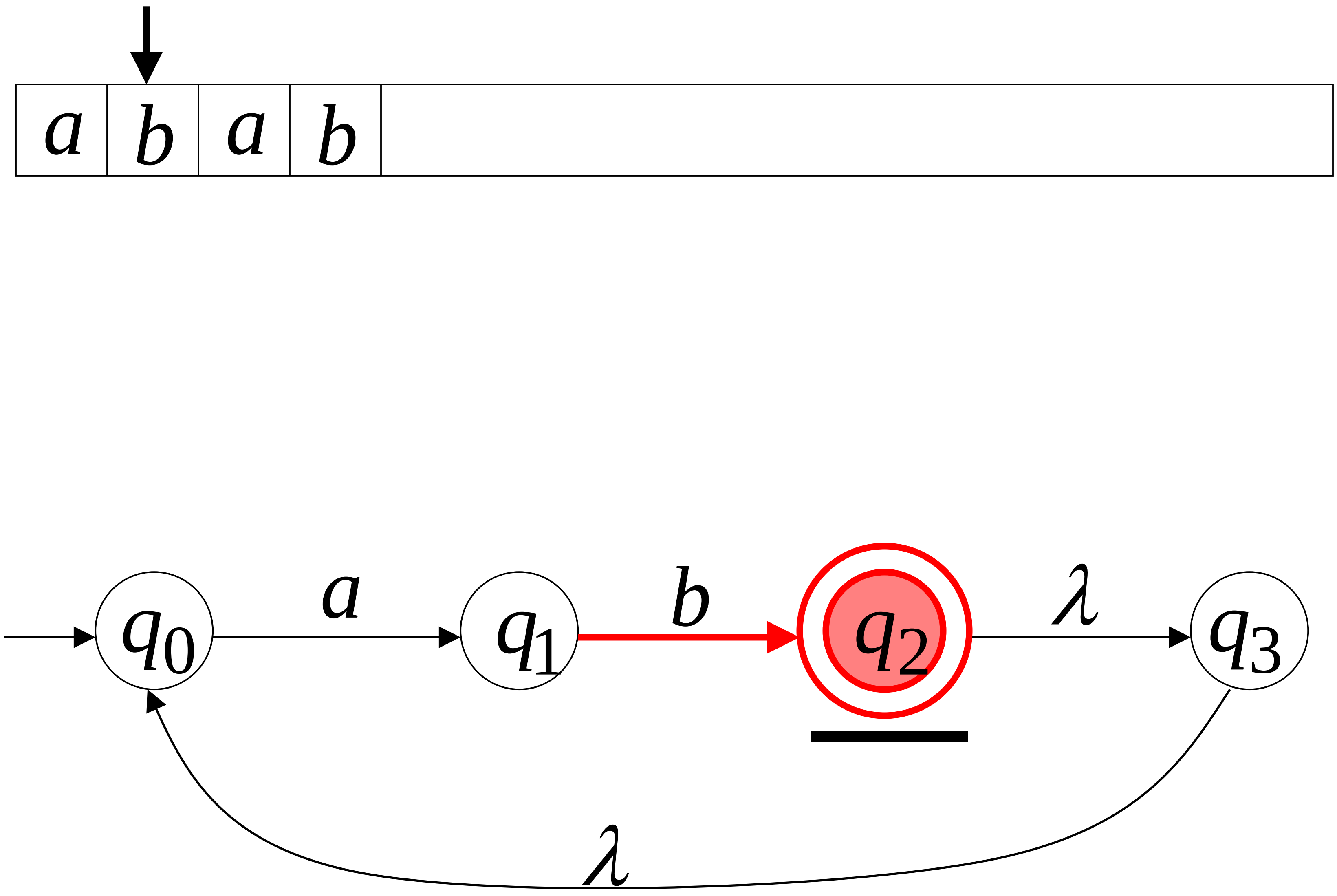
Another String

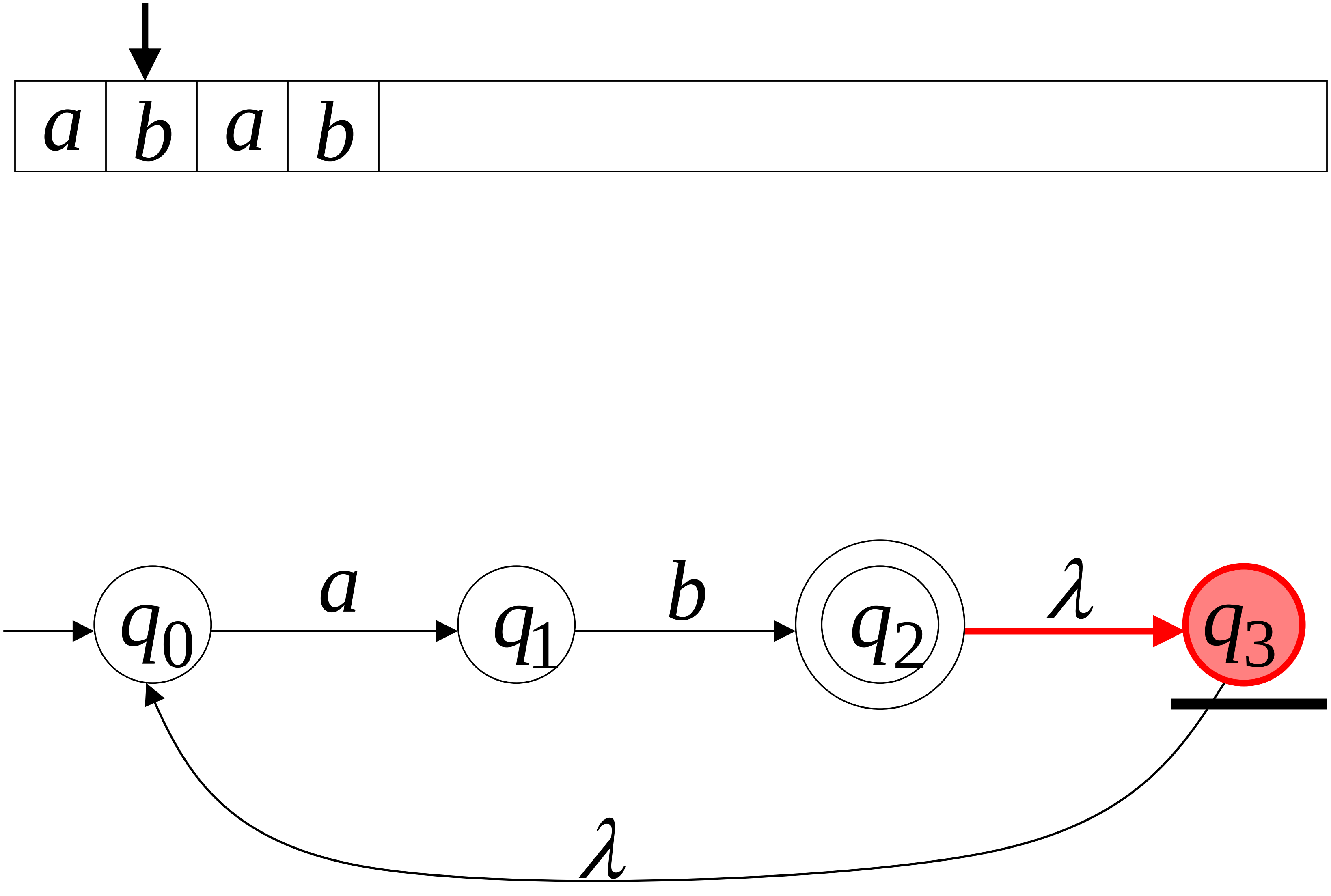


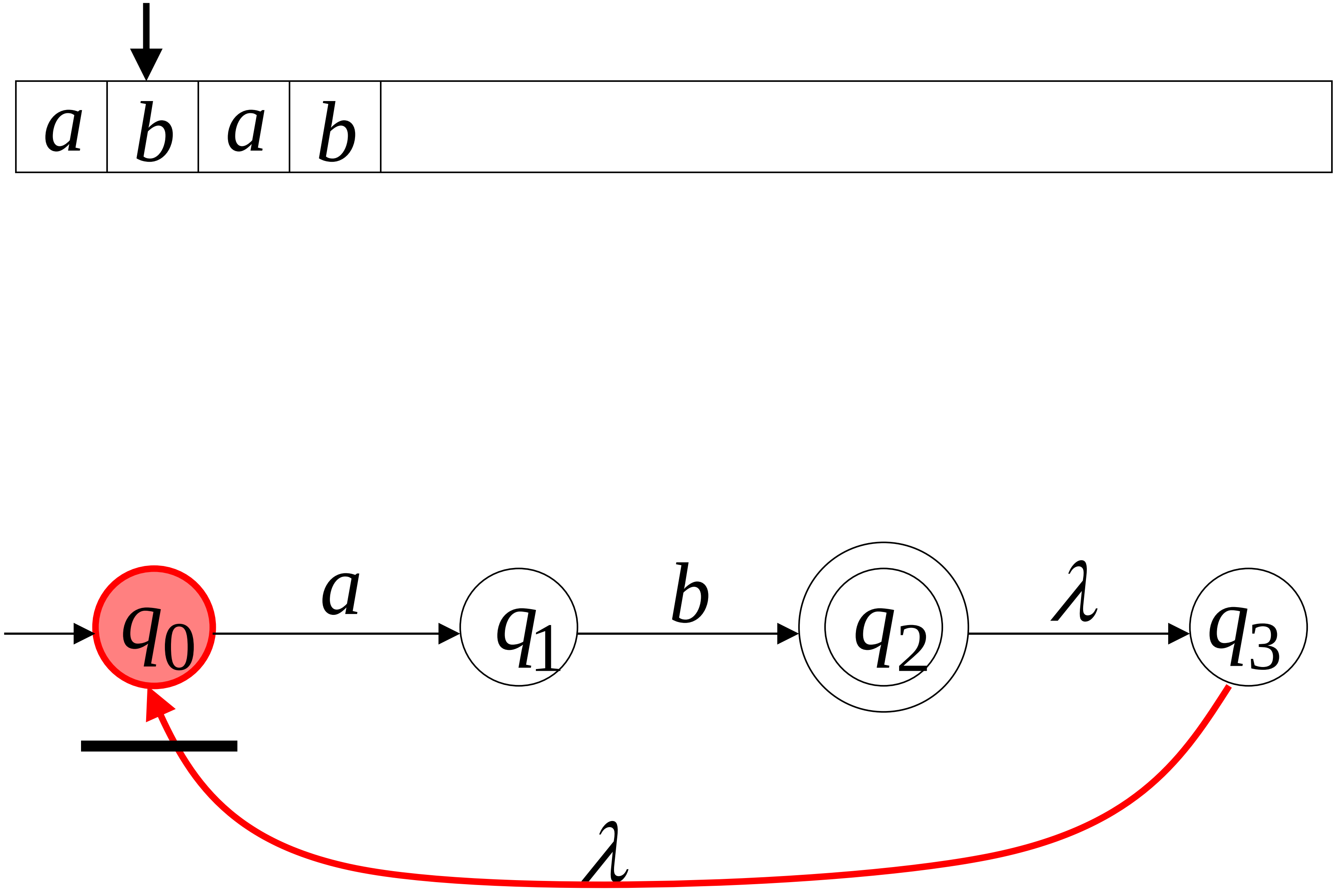


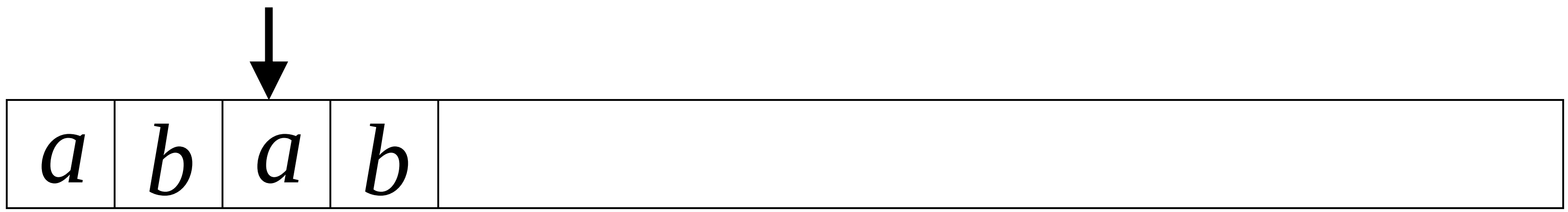

a
b
a
b

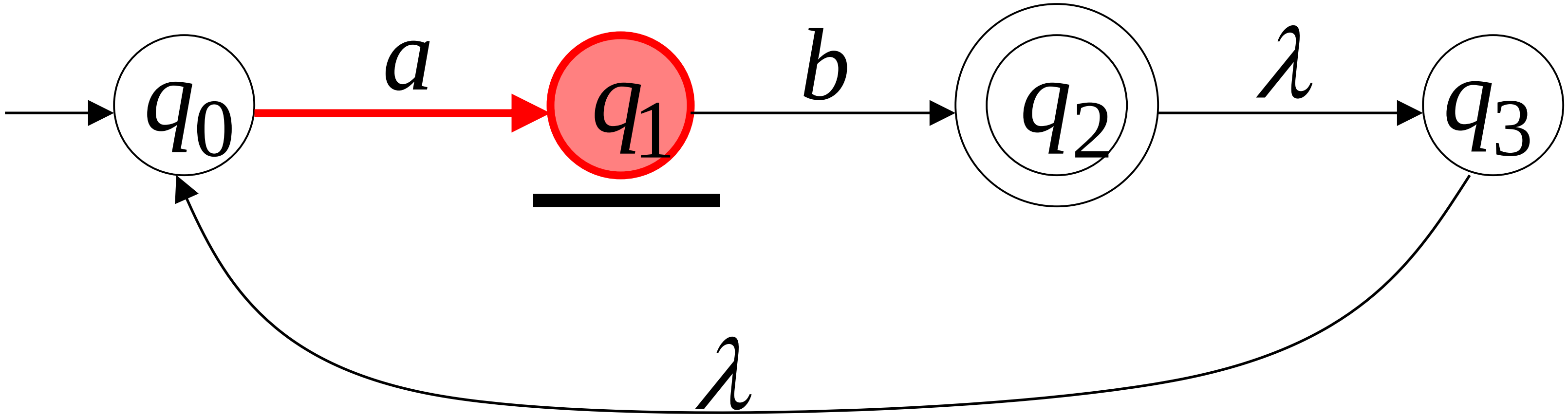

q_0
a
q_1
b
q_2
λ
q_3
λ

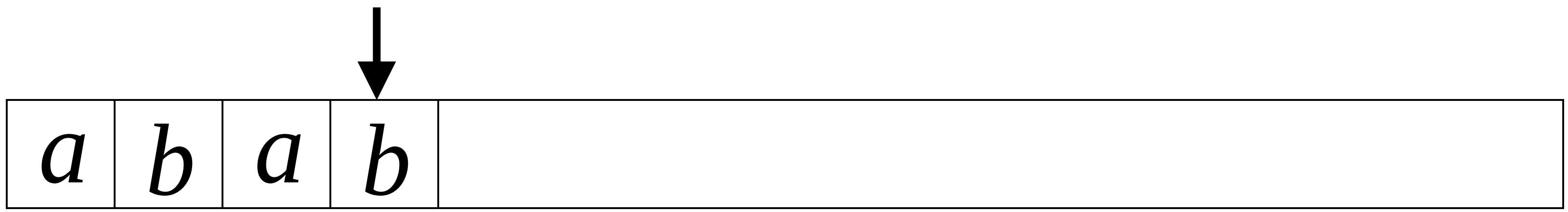

a
b
a
b
a
b
λ
q0
q1
q2
q3
λ

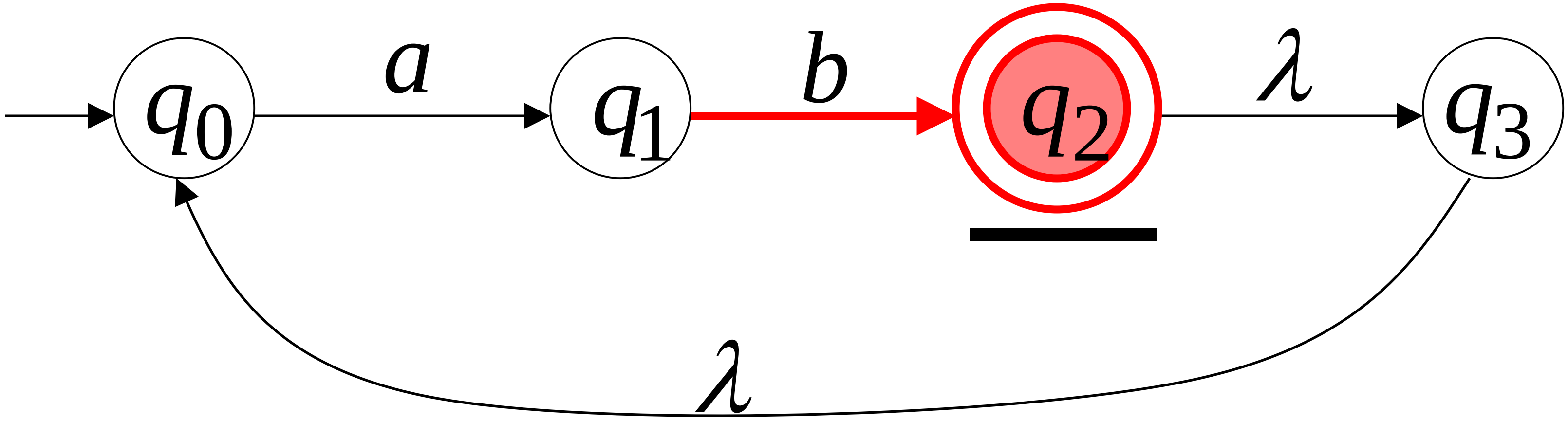

a
b
a
b
a
b
λ
q0
q1
q2
q3
λ

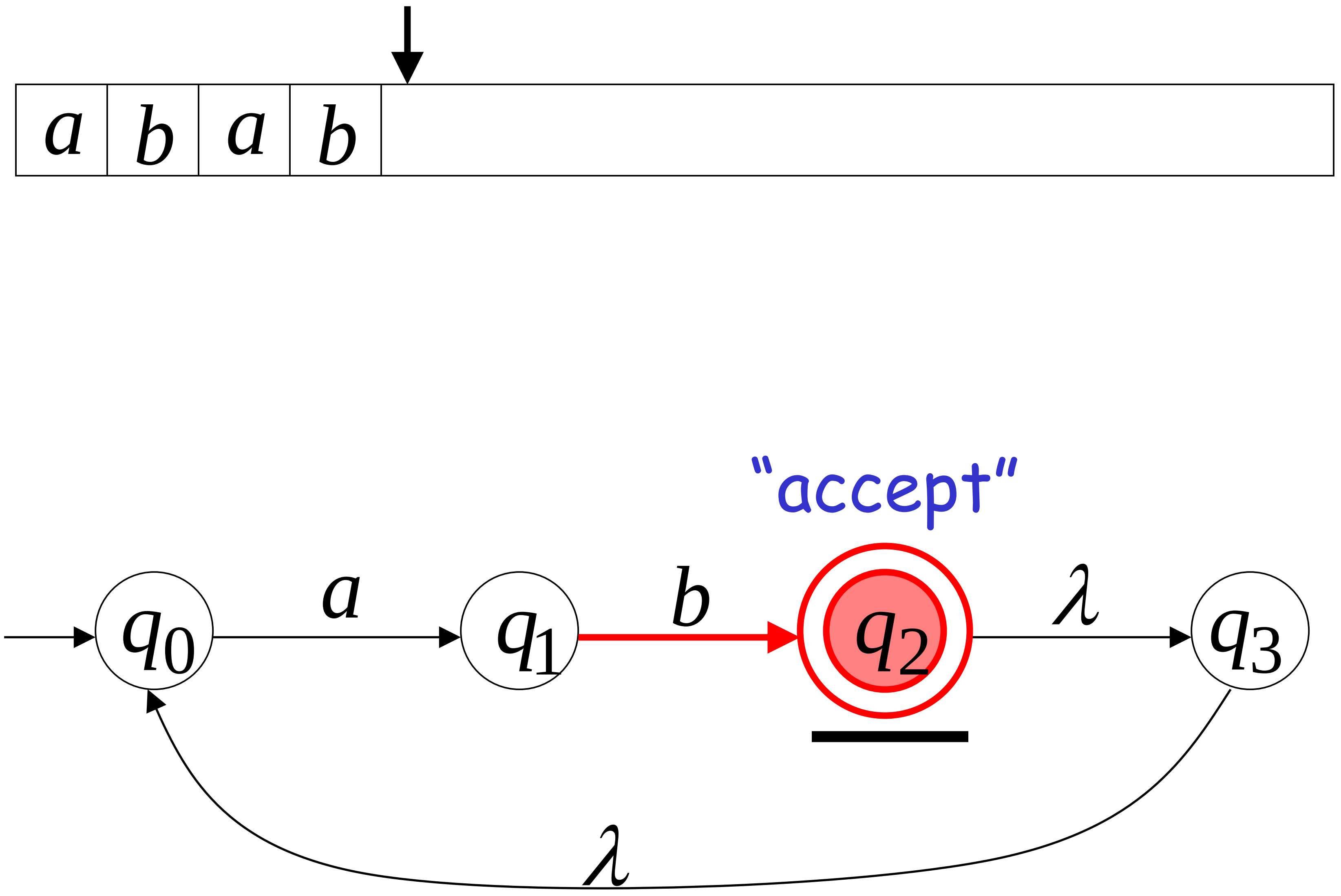

a b a b
q0 a q1 b q2 λ q3
λ


a
b
a
b


q0
a
q1
b
q2
λ
q3
λ


a
b
a
b

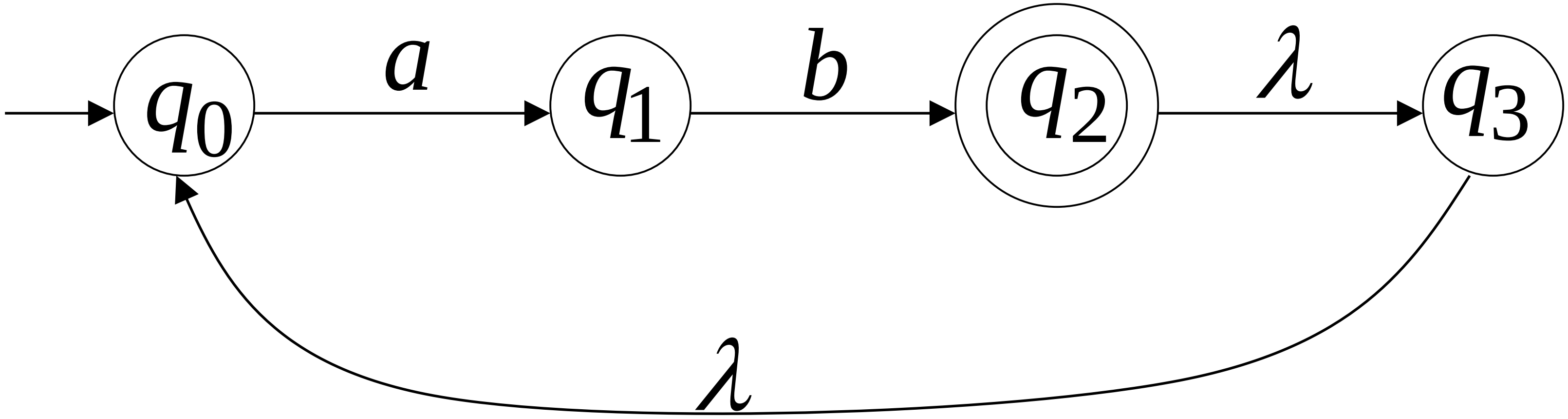

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b
"accept"
a
b
λ
λ
q0
q1
q2
q3

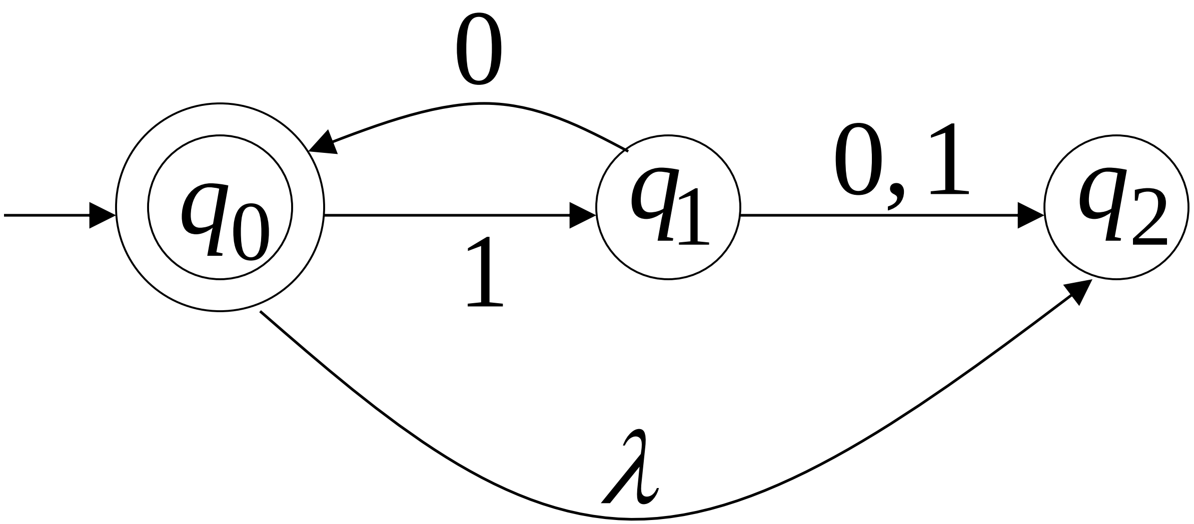
Language accepted

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

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



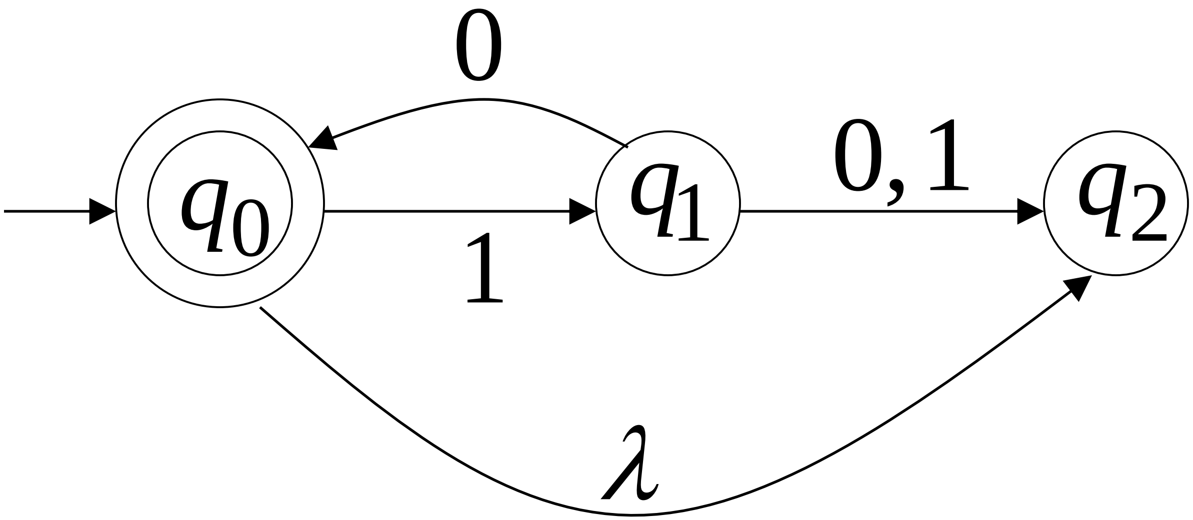
Another NFA Example



Language accepted

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

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

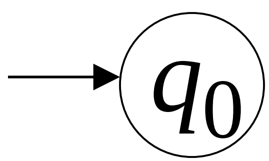


Remarks:

- The λ symbol never appears on the input tape

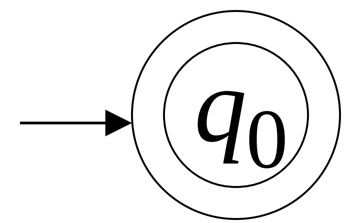
- Extreme automata:

M_1



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

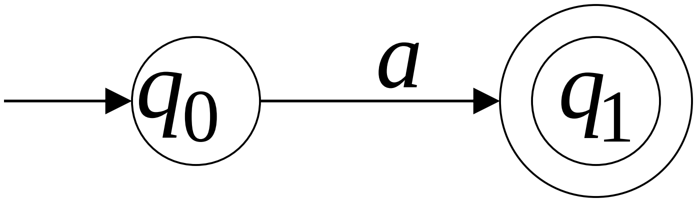
M_2



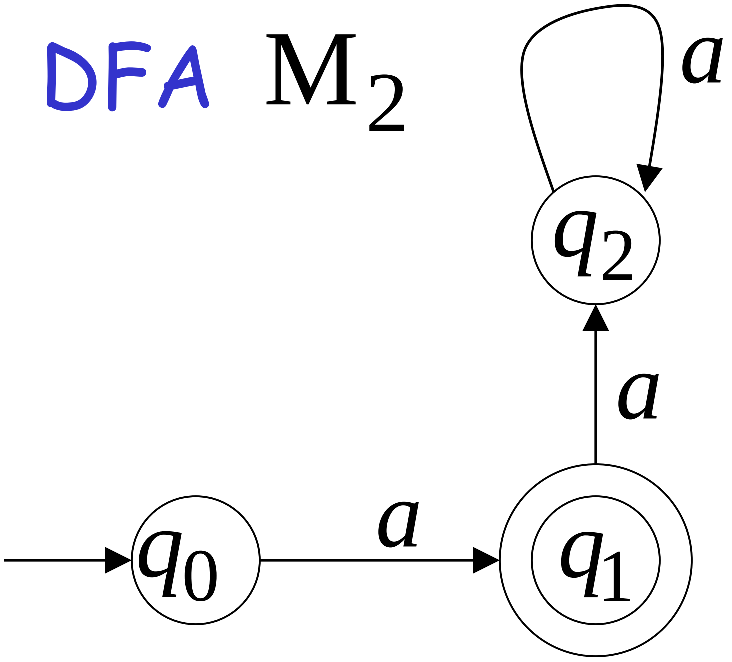
$$L(M_2) = \{\lambda\}$$

- 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. $\{a, b\}$

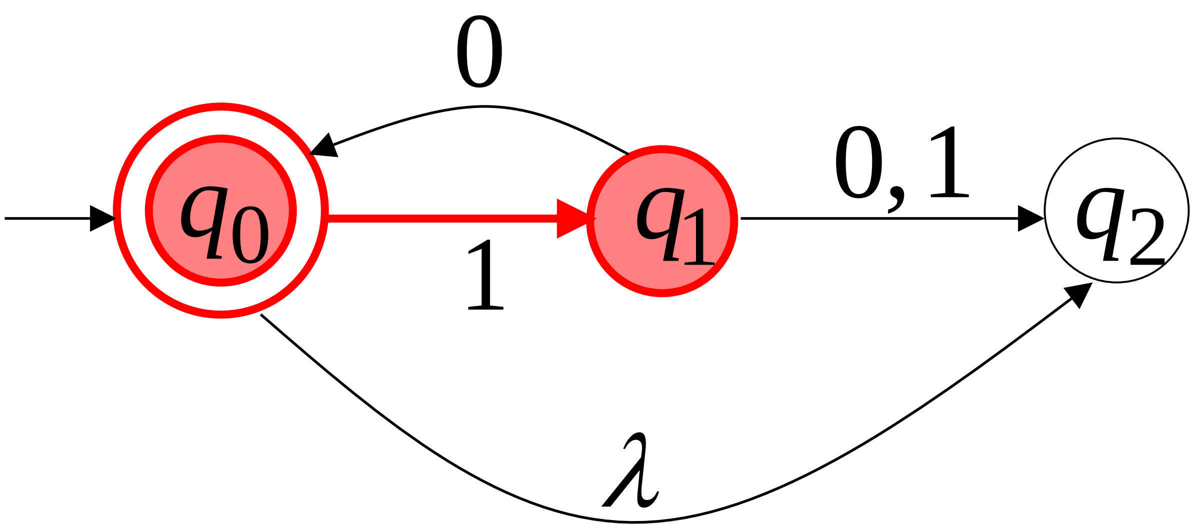
$\delta:$ Transition function

$q_0:$ Initial state

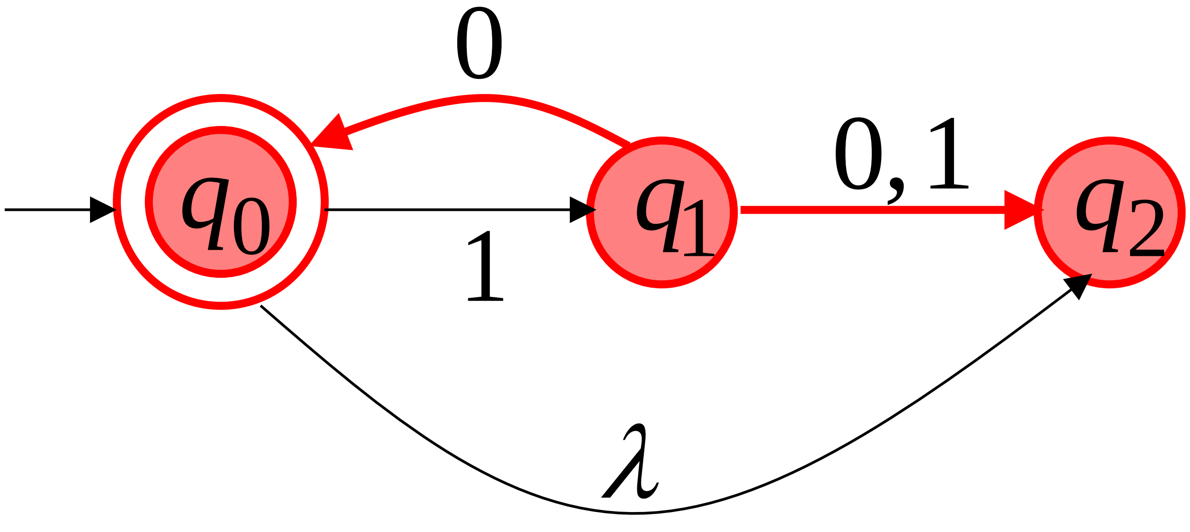
$F:$ Final states

Transition Function δ

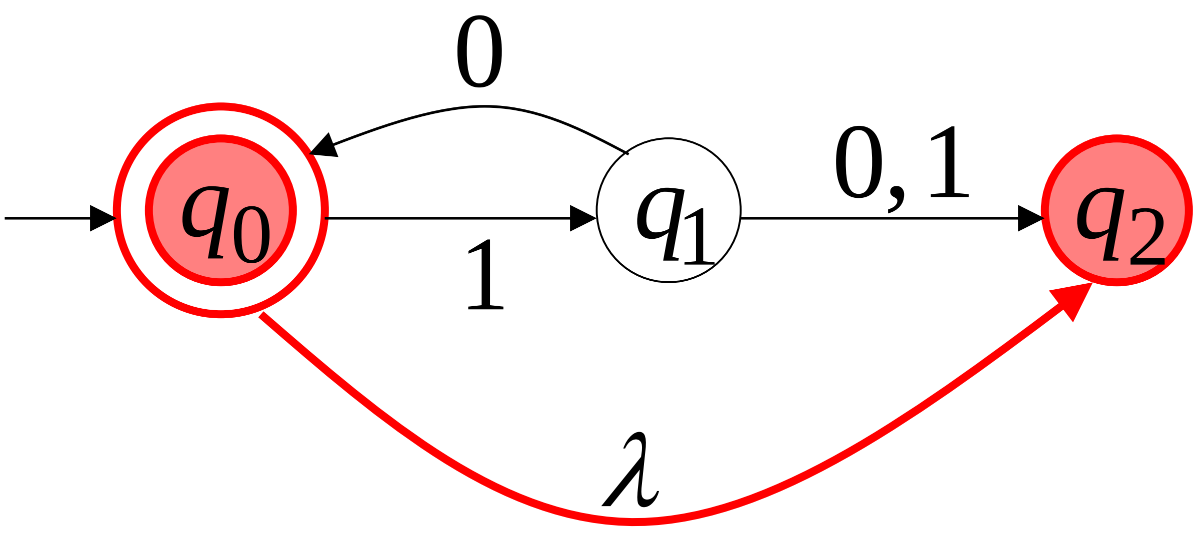
$$\delta(q_0, 1) = \{q_1\}$$



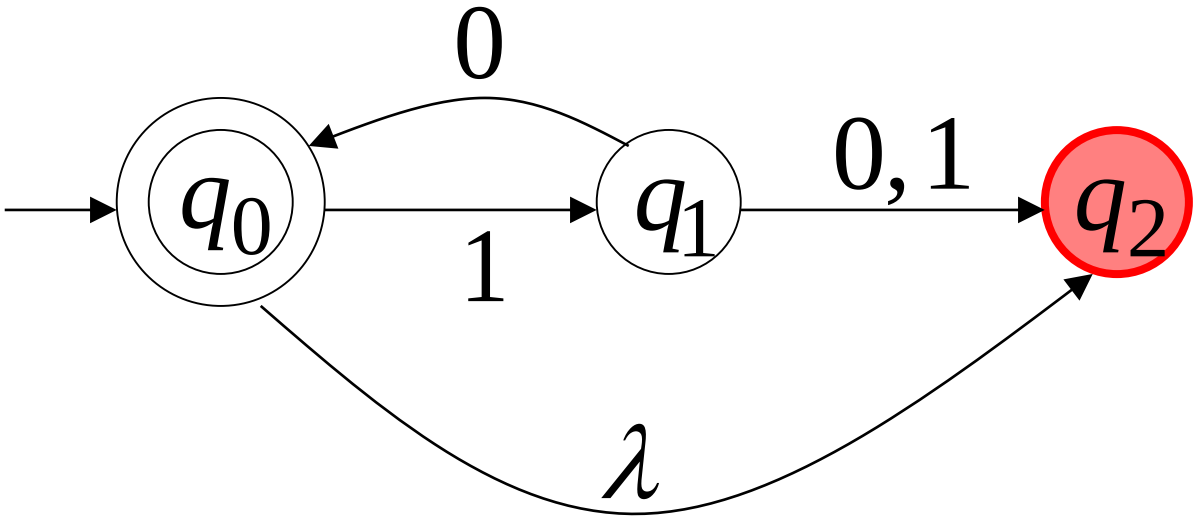
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

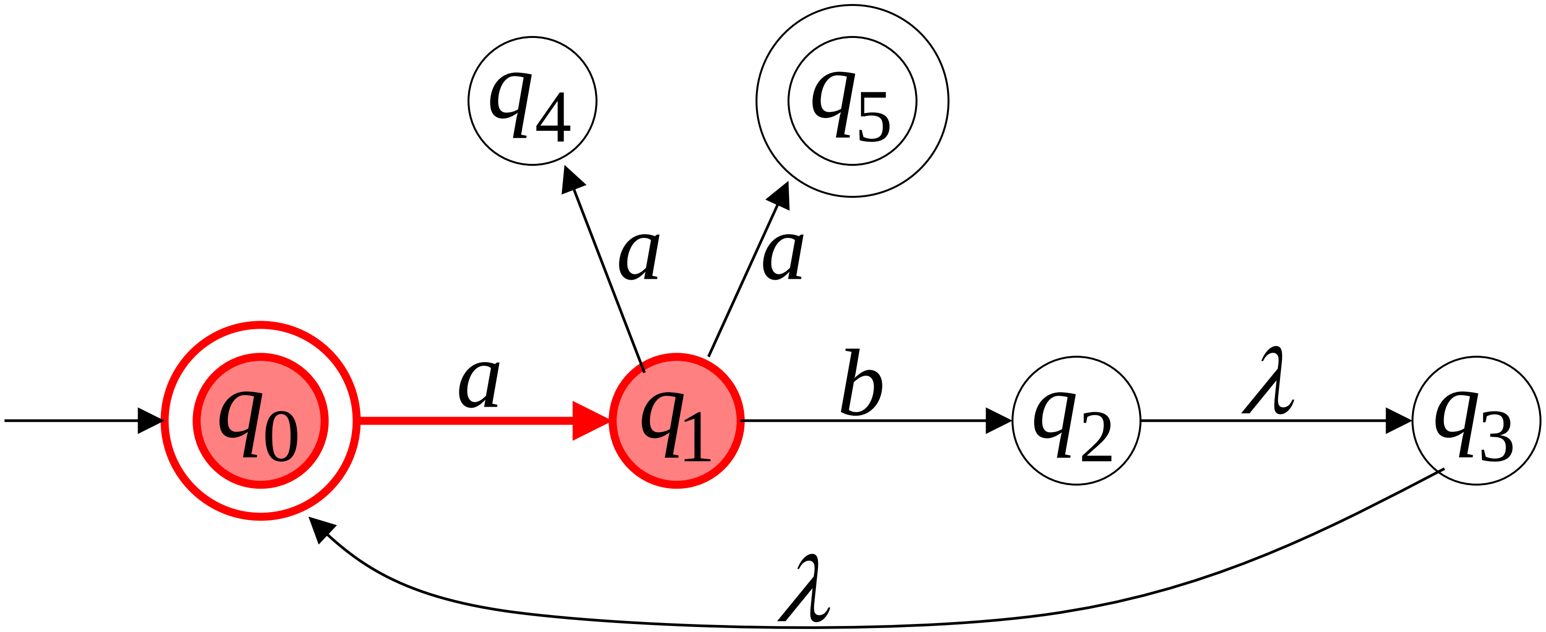


$$\delta(q_2, 1) = \varnothing$$

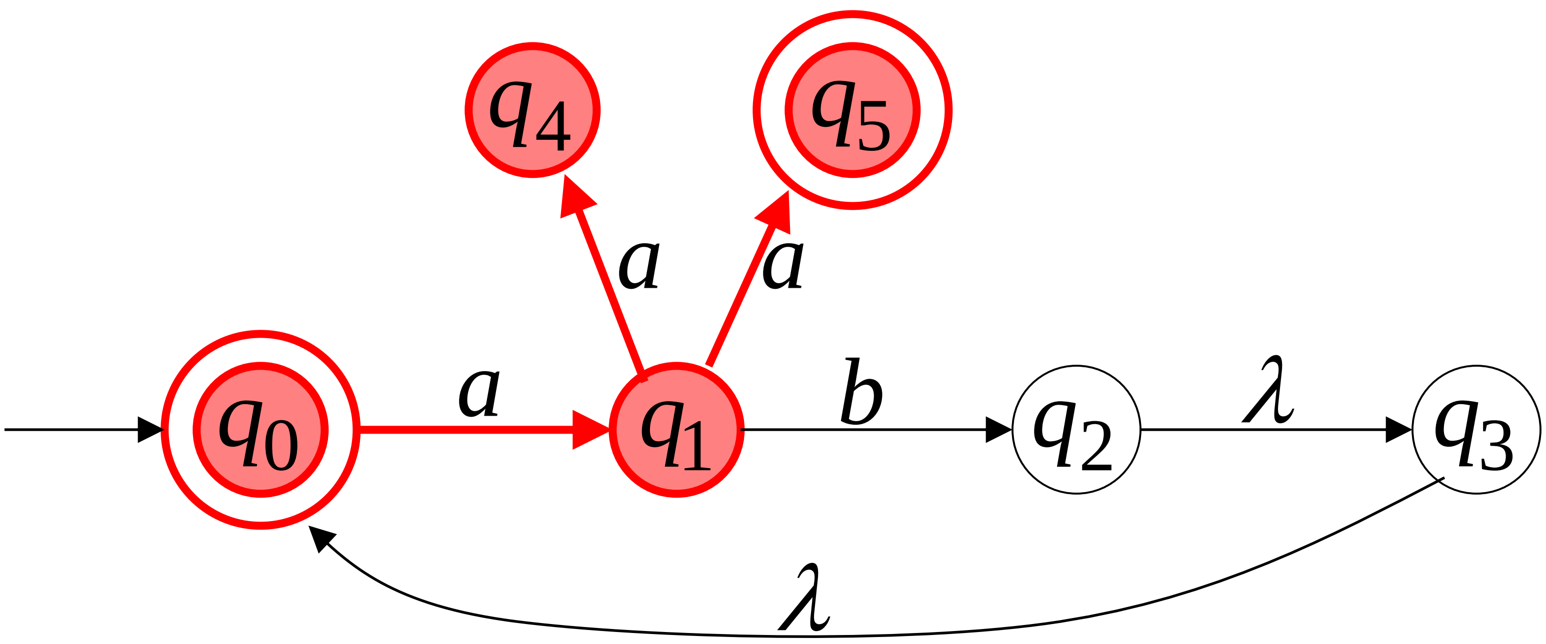


Extended Transition Function δ^*

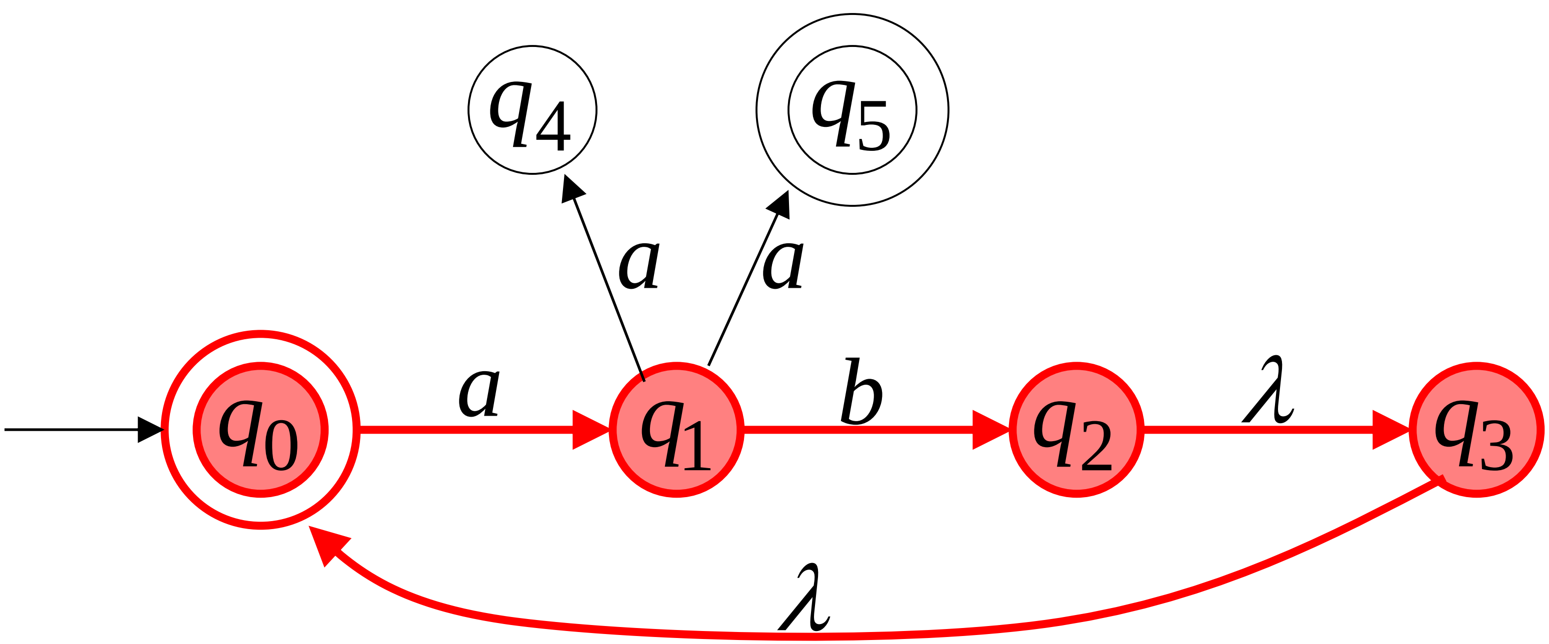
$$\delta^*(q_0, a) = \{q_1\}$$



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



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



Formally

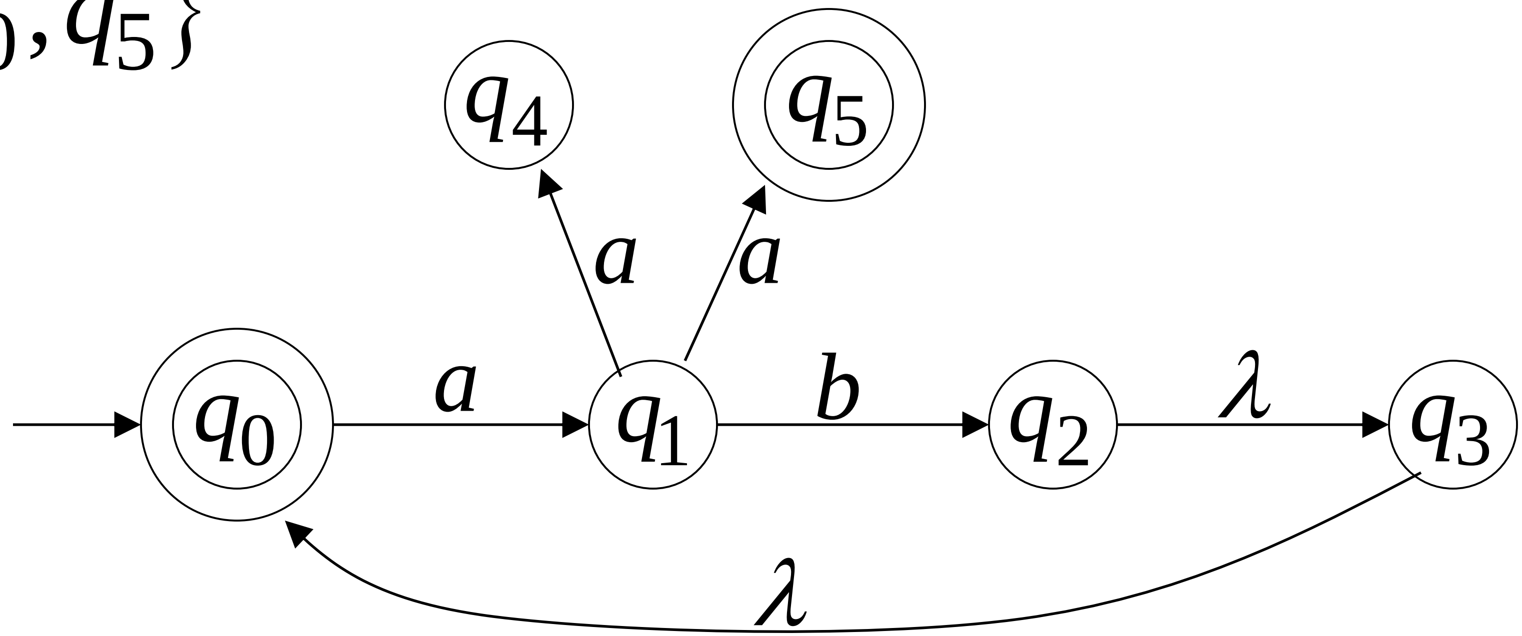
It holds $q_j \in \delta^*(q_i, w)$

if and only if

there is a walk from q_i to q_j
with label w

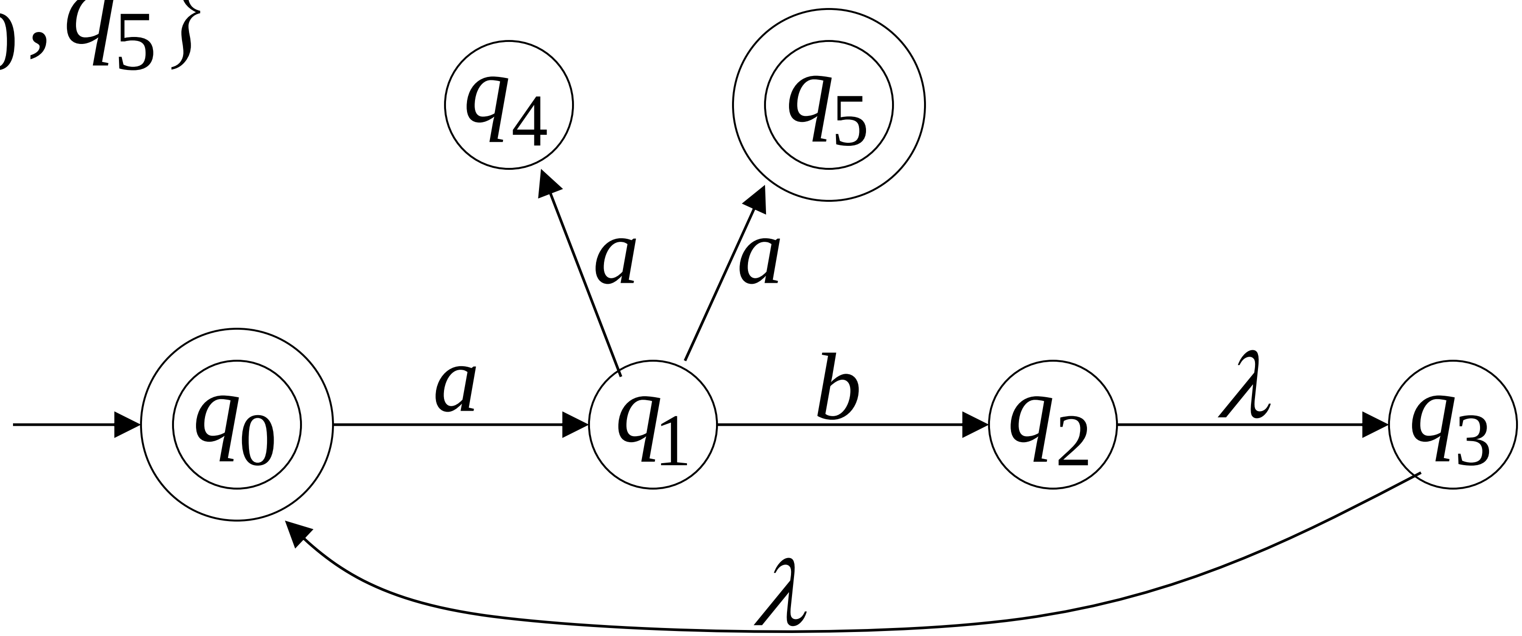
The Language of an NFA M

$F = \{q_0, q_5\}$



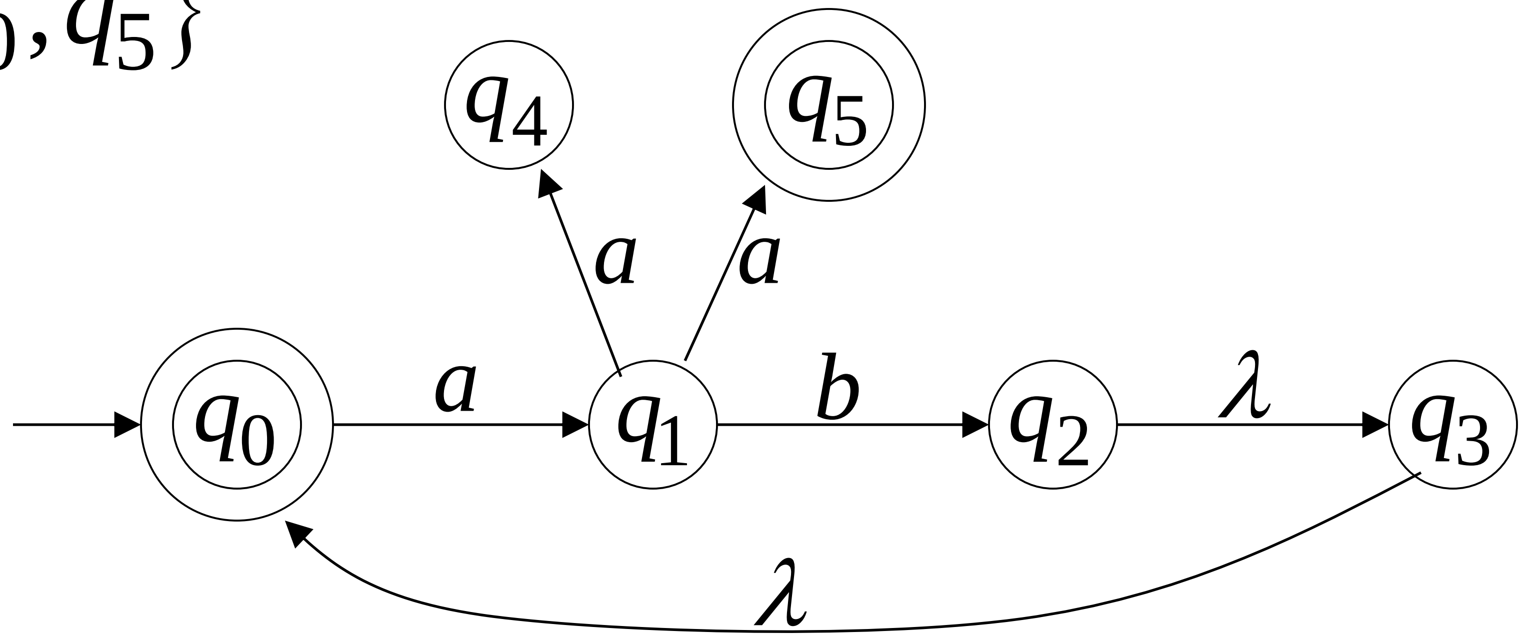
$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\}$ $\qquad aa \in L(M)$

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



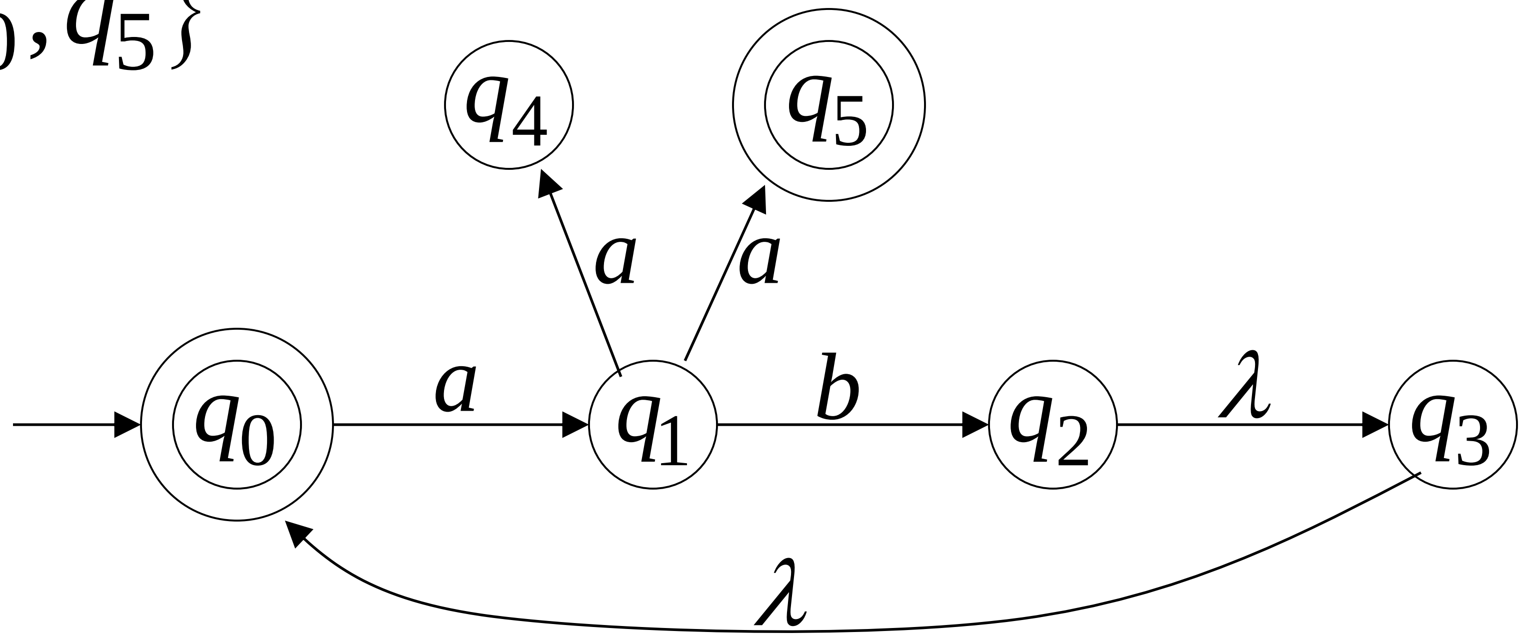
$$\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\} \qquad ab \in L(M)$$

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

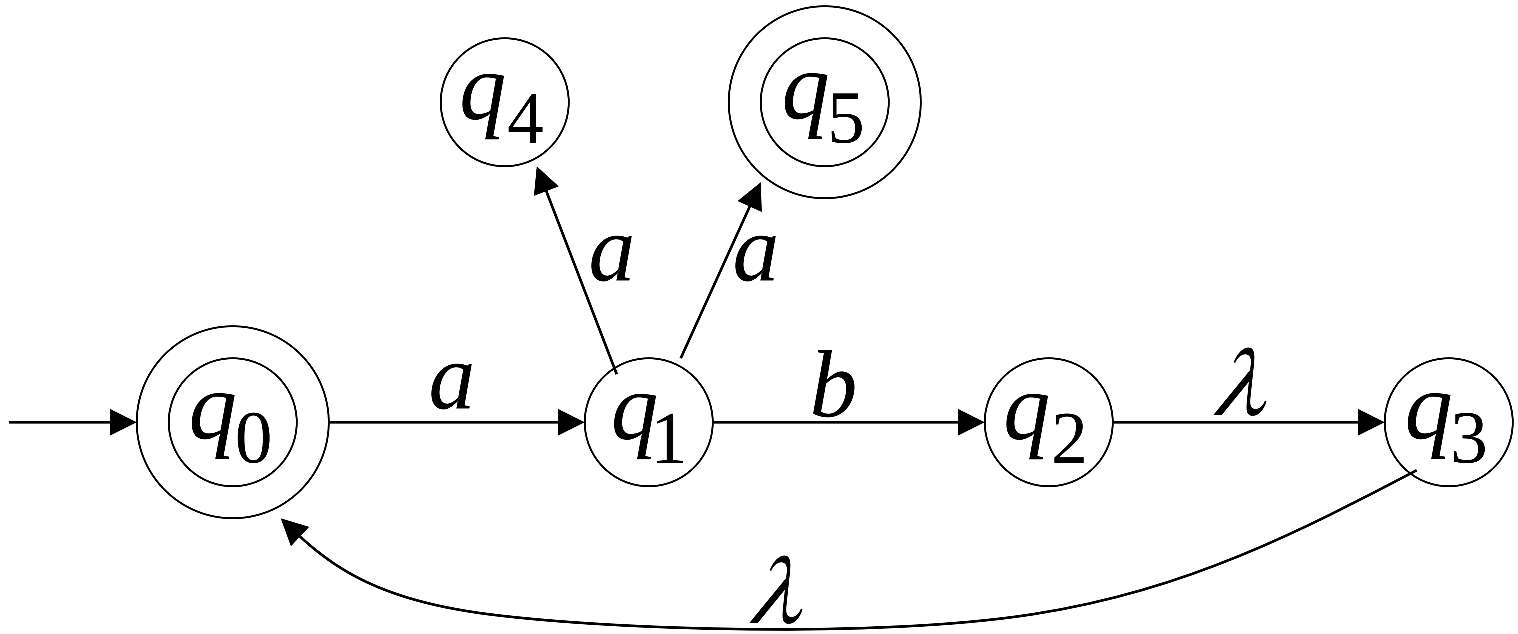


$$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\} \qquad aaba \in L(M)$$

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



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



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

Formally

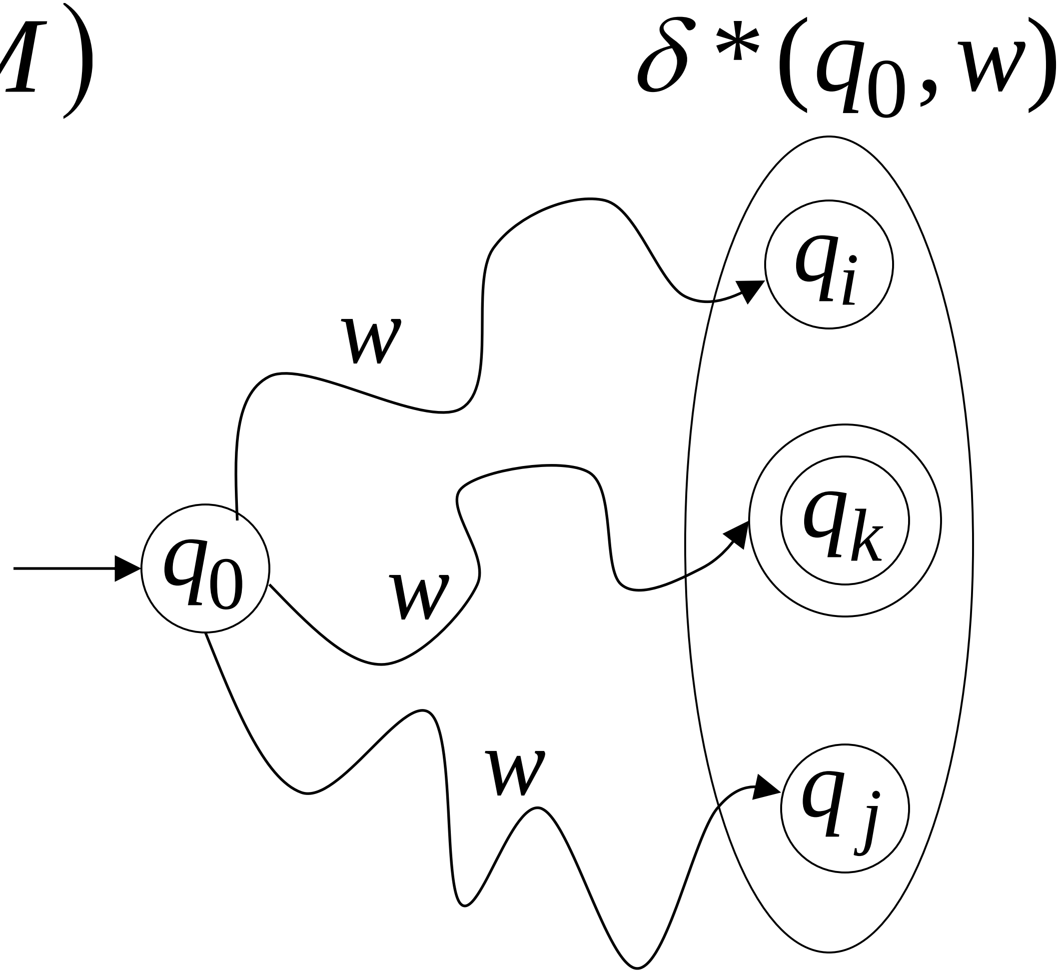
The language accepted by NFA M is:

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

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

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

$w \in L(M)$



$q_k \in F$

Equivalence of NFAs and DFAs

Linz: 2.3 Equivalence of Deterministic and Nondeterministic Finite Accepters, page 58

Equivalence of Machines

For DFAs or NFAs:

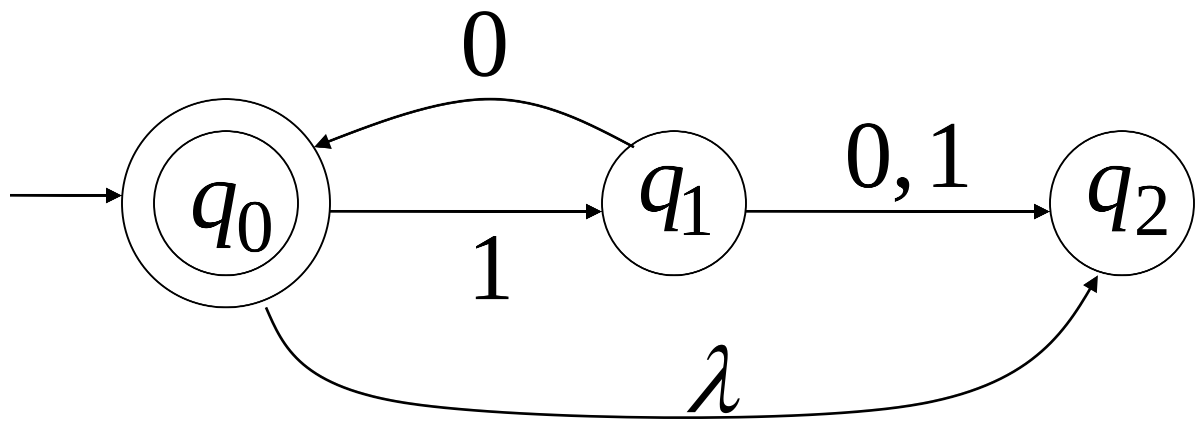
Machine M_1 is equivalent to machine M_2

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

Example

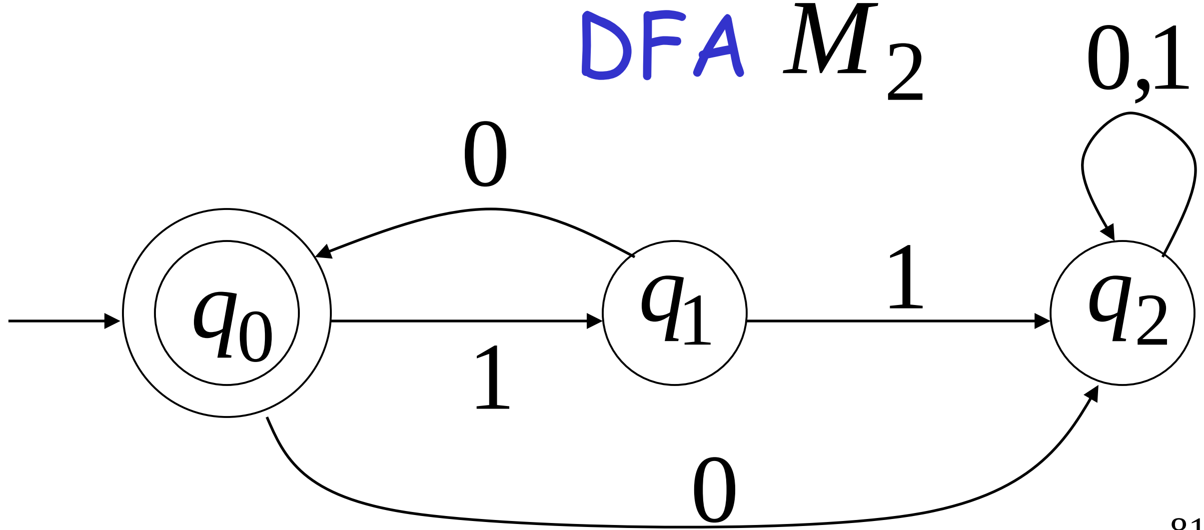
NFA M_1

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



DFA M_2

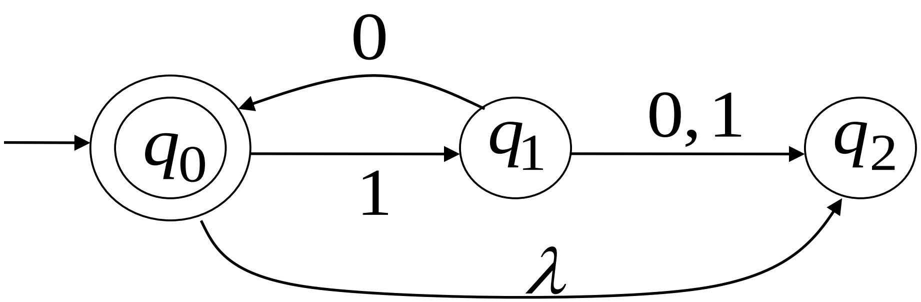
$$L(M_2) = \{10\}^*$$



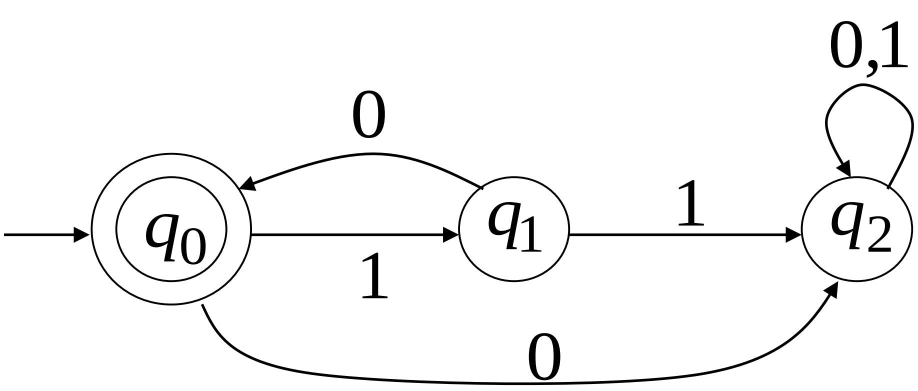
Since $L(M_1) = L(M_2) = \{10\}^*$

machines M_1 and M_2 are equivalent

NFA M_1



DFA M_2



Equivalence of NFAs and DFAs

Question: NFAs = DFAs ?

Same power?

Accept the same languages?

Equivalence of NFAs and DFAs

Question: NFAs = DFAs ? **YES!**

Same power?

Accept the same languages?

We will prove:

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

NFAs and DFAs have the same computation power

Step 1

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

Proof: Every DFA is trivially an NFA

A language accepted by a DFA
is also accepted by an NFA

Step 2

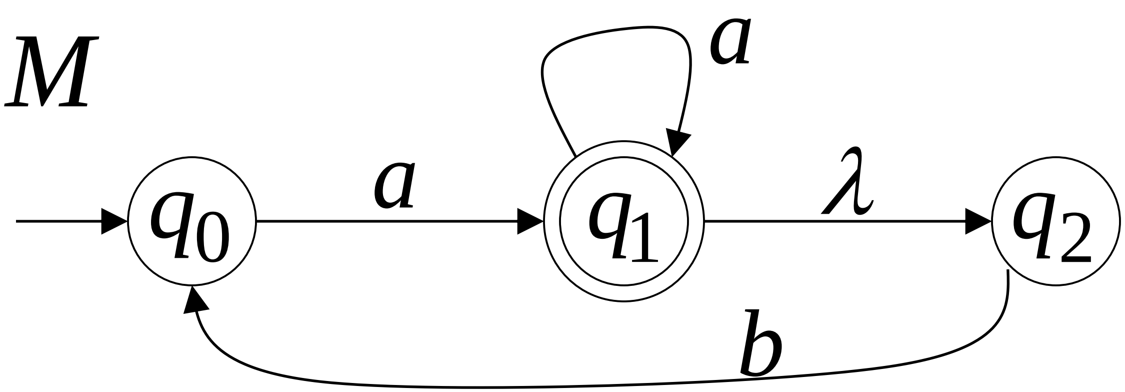
$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by DFAs}\end{array}\right\}$$

Proof: Any NFA can be converted to an equivalent DFA

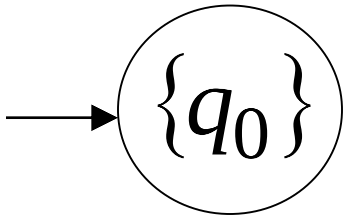
A language accepted by an NFA is also accepted by a DFA

NFA to DFA

NFA M

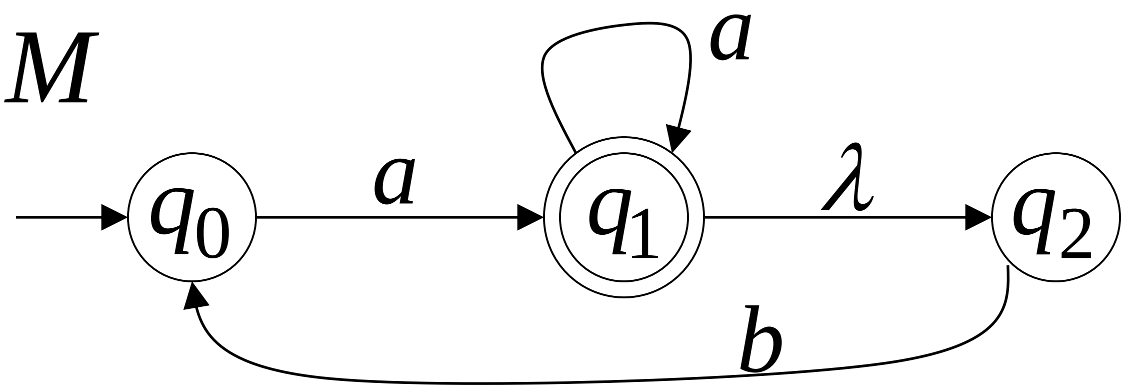


DFA M'

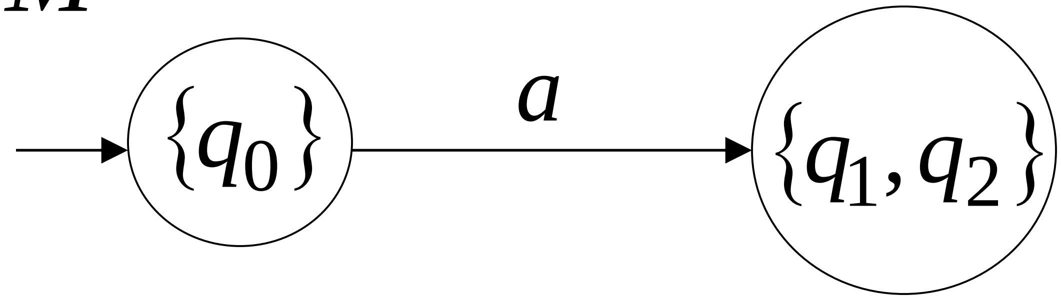


NFA to DFA

NFA M

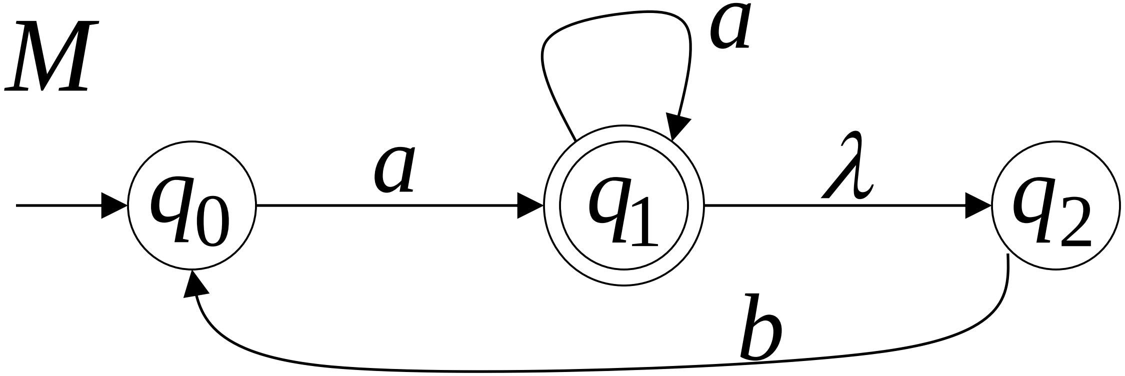


DFA M'

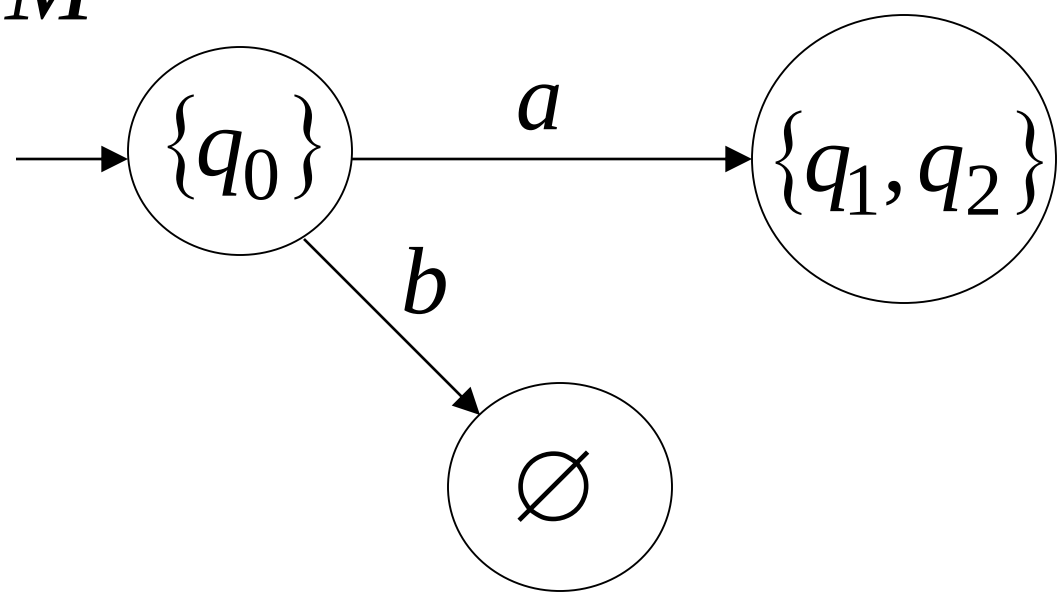


NFA to DFA

NFA M

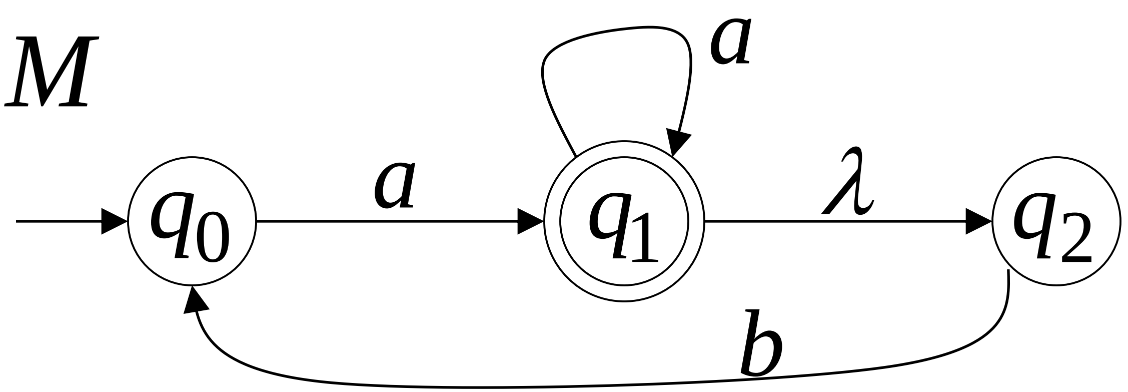


DFA M'

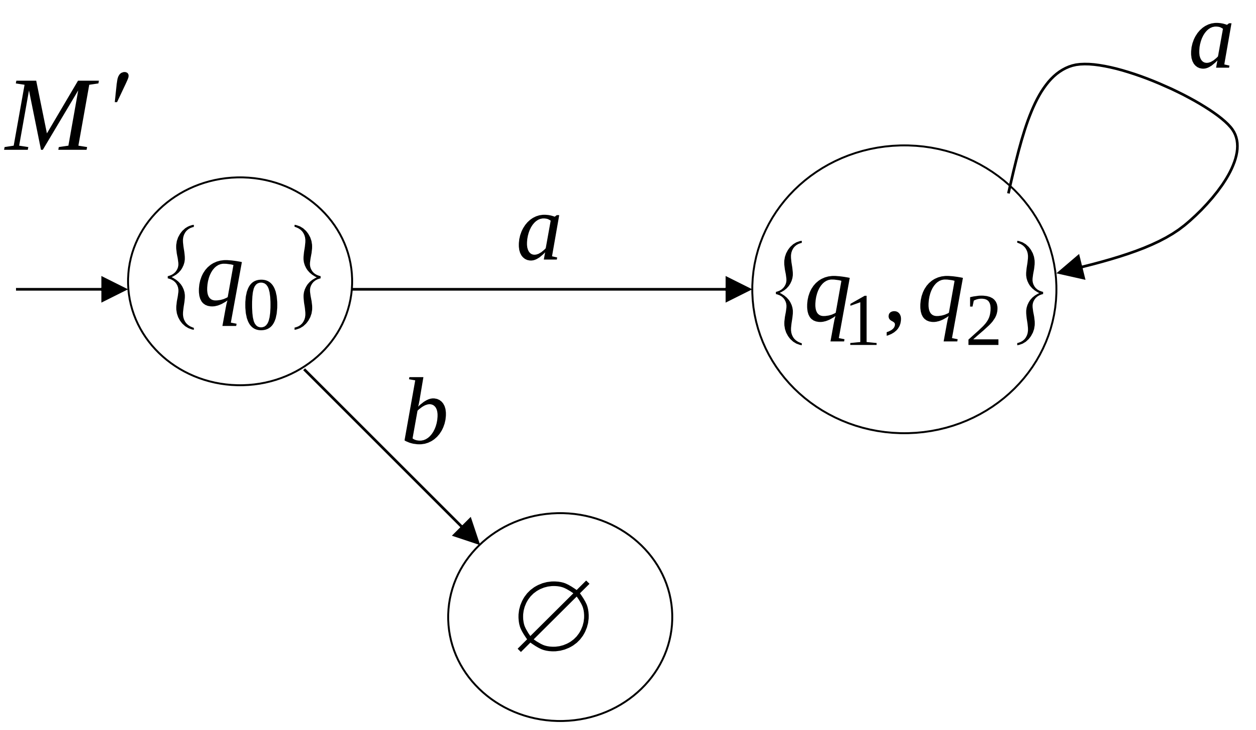


NFA to DFA

NFA M

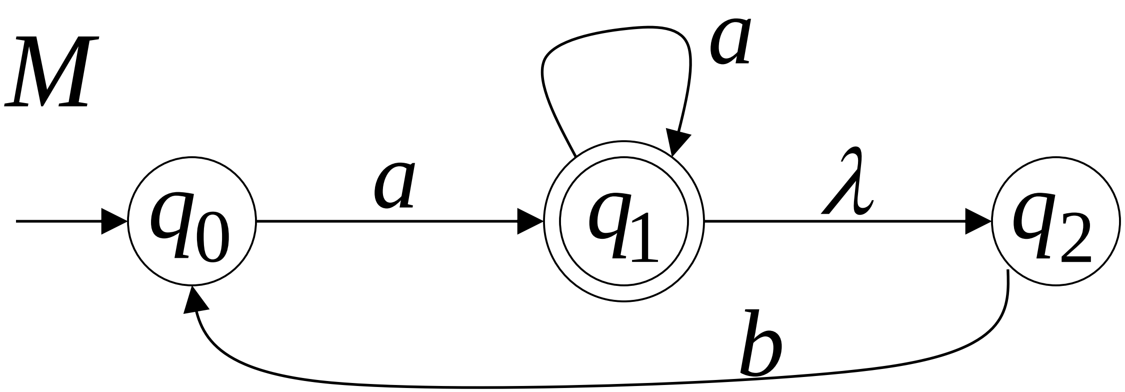


DFA M'

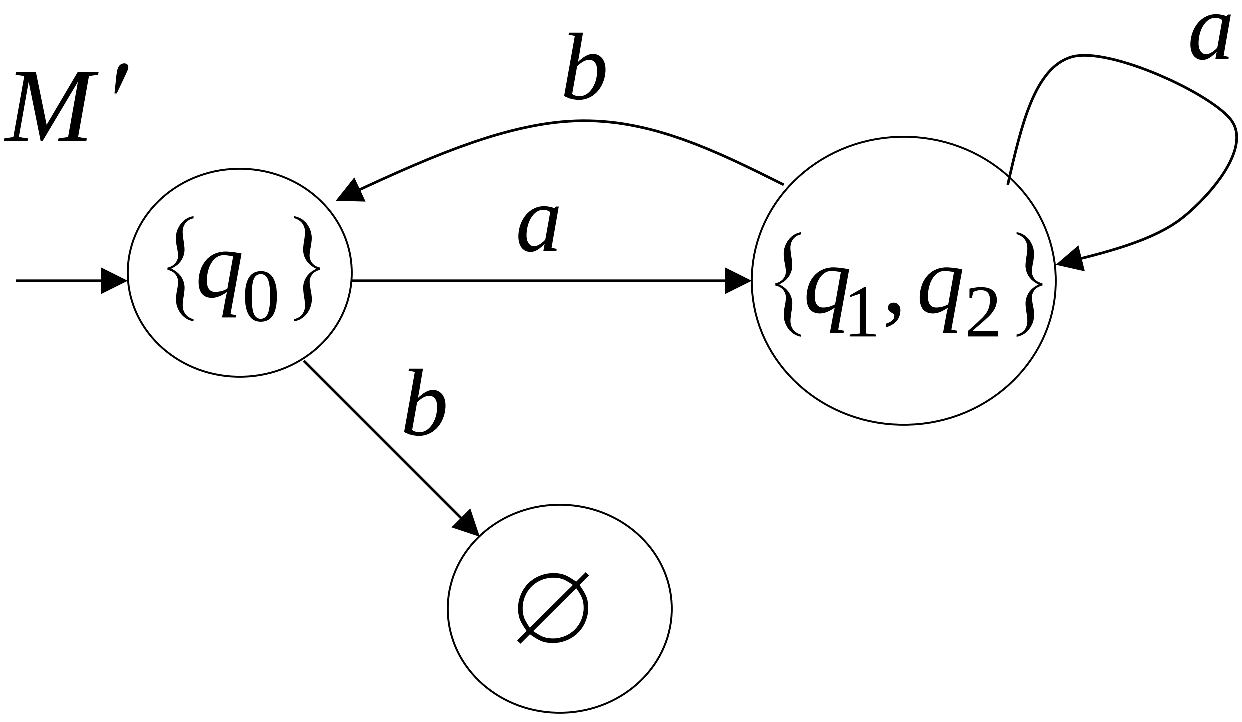


NFA to DFA

NFA M

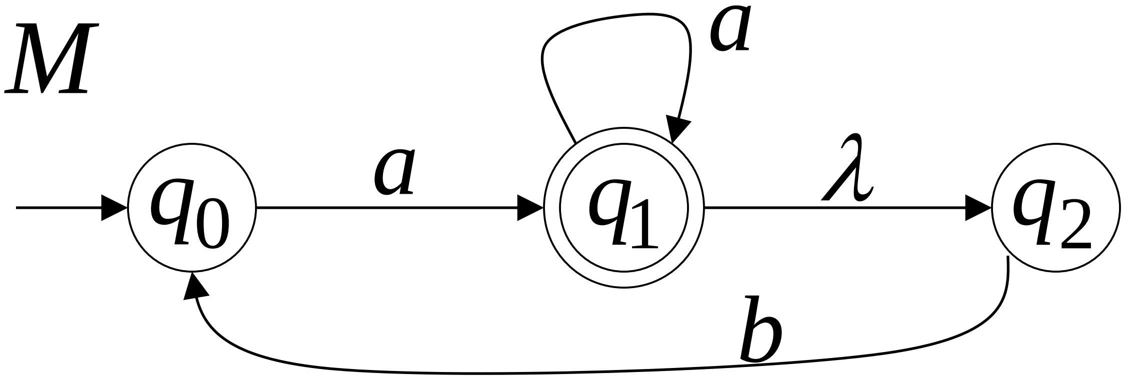


DFA M'

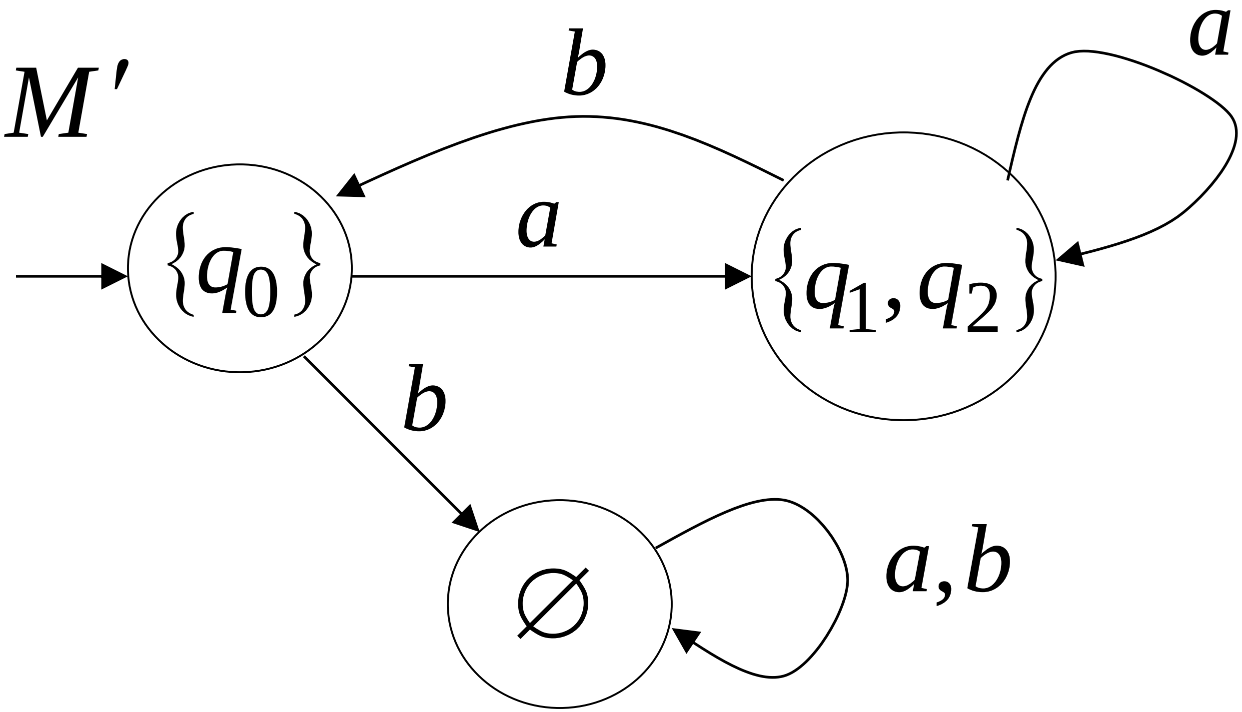


NFA to DFA

NFA M



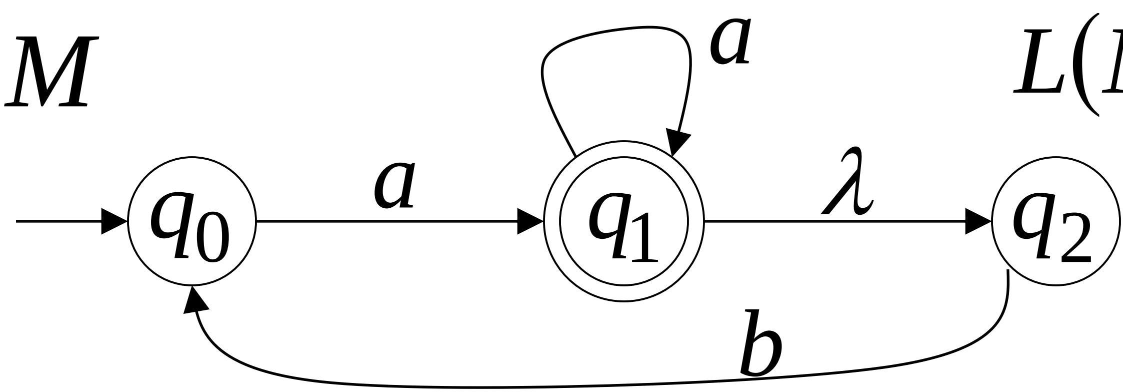
DFA M'



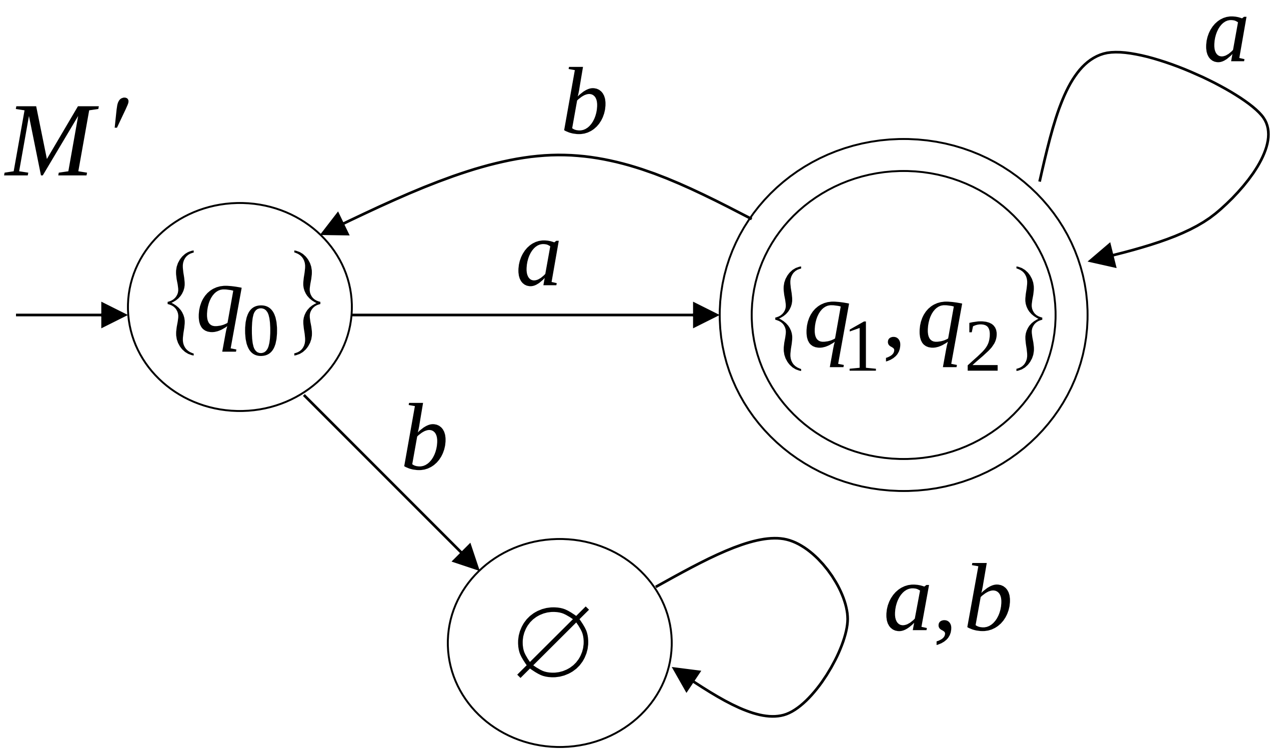
NFA to DFA

NFA M

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



DFA M'



NFA to DFA: Remarks

We are given an NFA M

We want to convert it
to an equivalent DFA M'

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

If the NFA has states

$$q_0, q_1, q_2, \ldots$$

the DFA has states in the powerset

$$\emptyset, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots$$

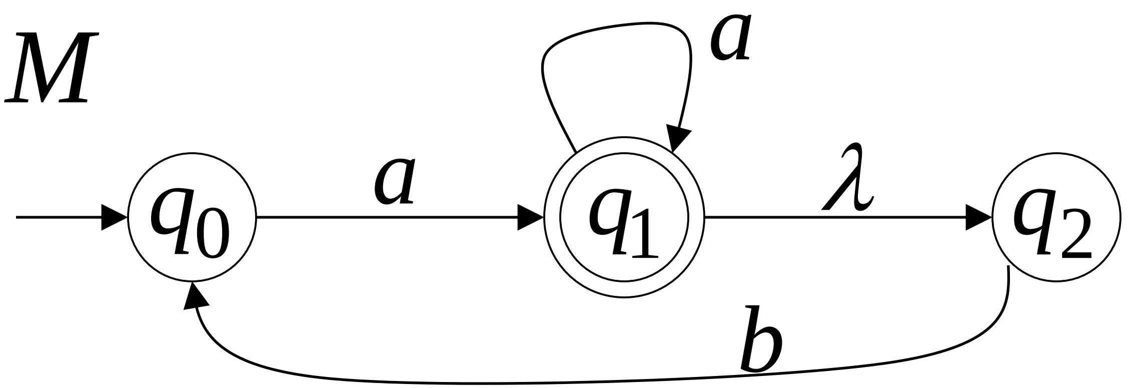
Procedure NFA to DFA

1. Initial state of NFA: q_0

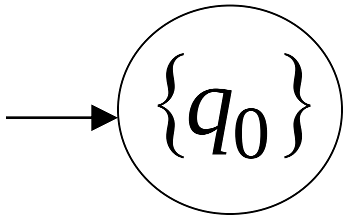
Initial state of DFA: $\{q_0\}$

Example

NFA M



DFA M'



Procedure NFA to DFA

2. For every DFA's state $\{q_i, q_j, \ldots, q_m\}$

Compute in the NFA

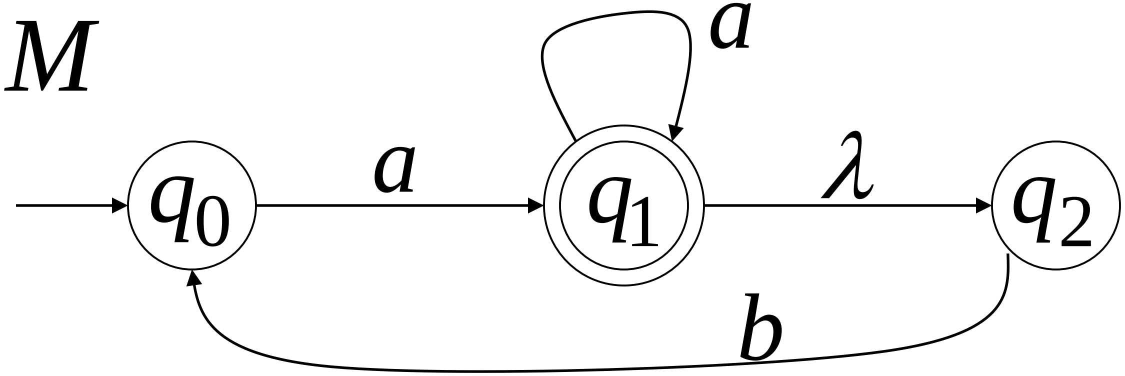
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to DFA

$$\delta(\{q_i, q_j, \ldots, q_m\},\ a) = \{q'_i, q'_j, \ldots, q'_m\}$$

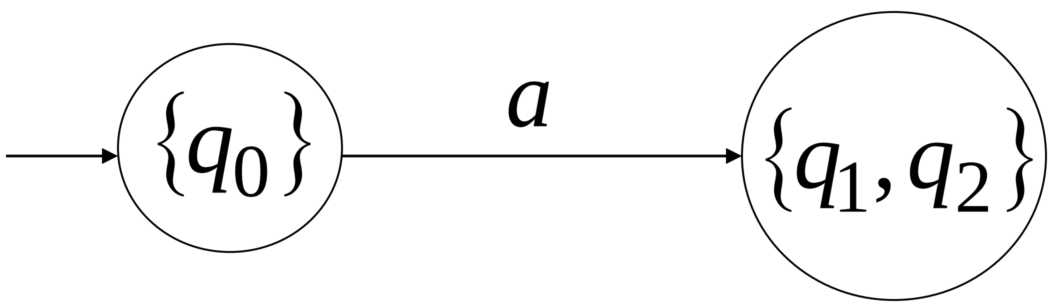
Exampe

NFA M



$$\delta^*(q_0, a) = \{q_1, q_2\}$$

DFA M'



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

Procedure NFA to DFA

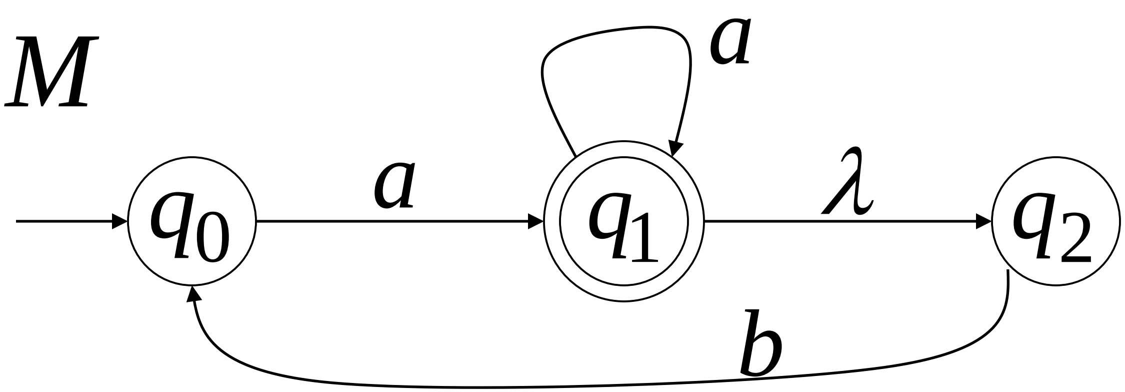
Repeat Step 2 for all letters in alphabet,

until

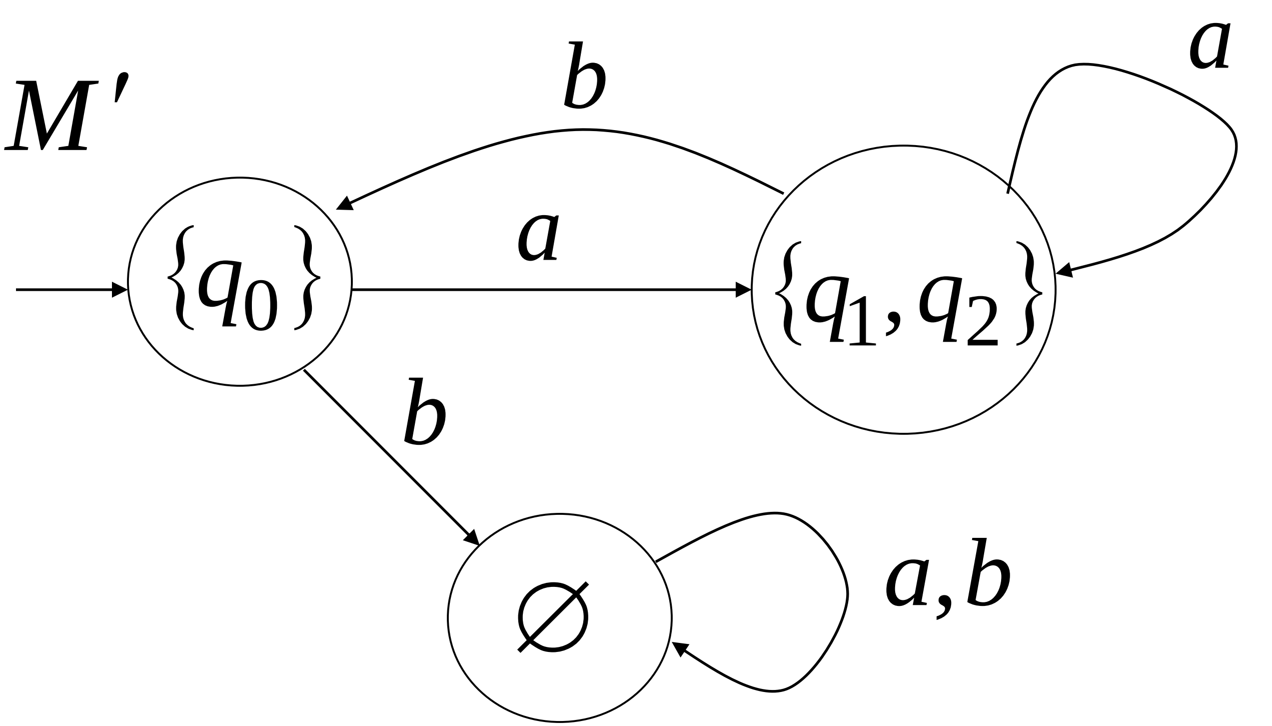
no more transitions can be added.

Example

NFA M



DFA M'



Procedure NFA to DFA

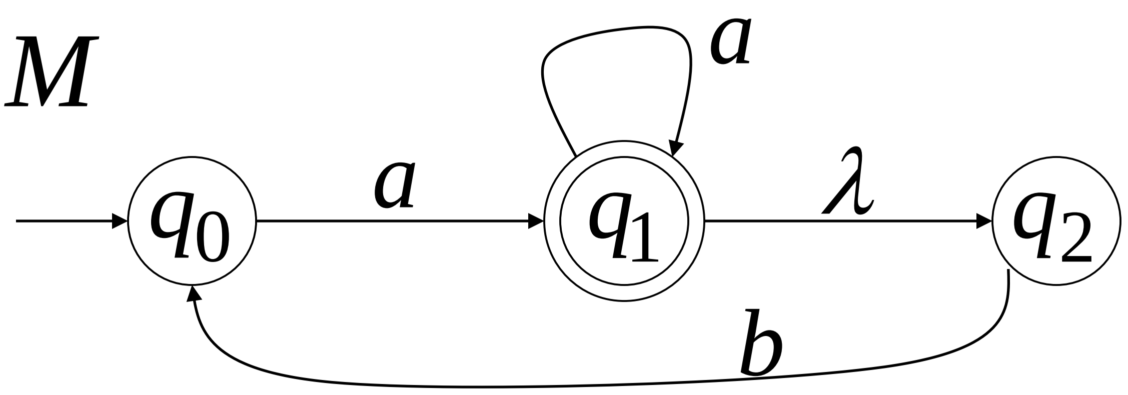
3. For any DFA state $\{q_i, q_j, \ldots, q_m\}$

If some q_j is a final state in the NFA

Then, $\{q_i, q_j, \ldots, q_m\}$
is a final state in the DFA

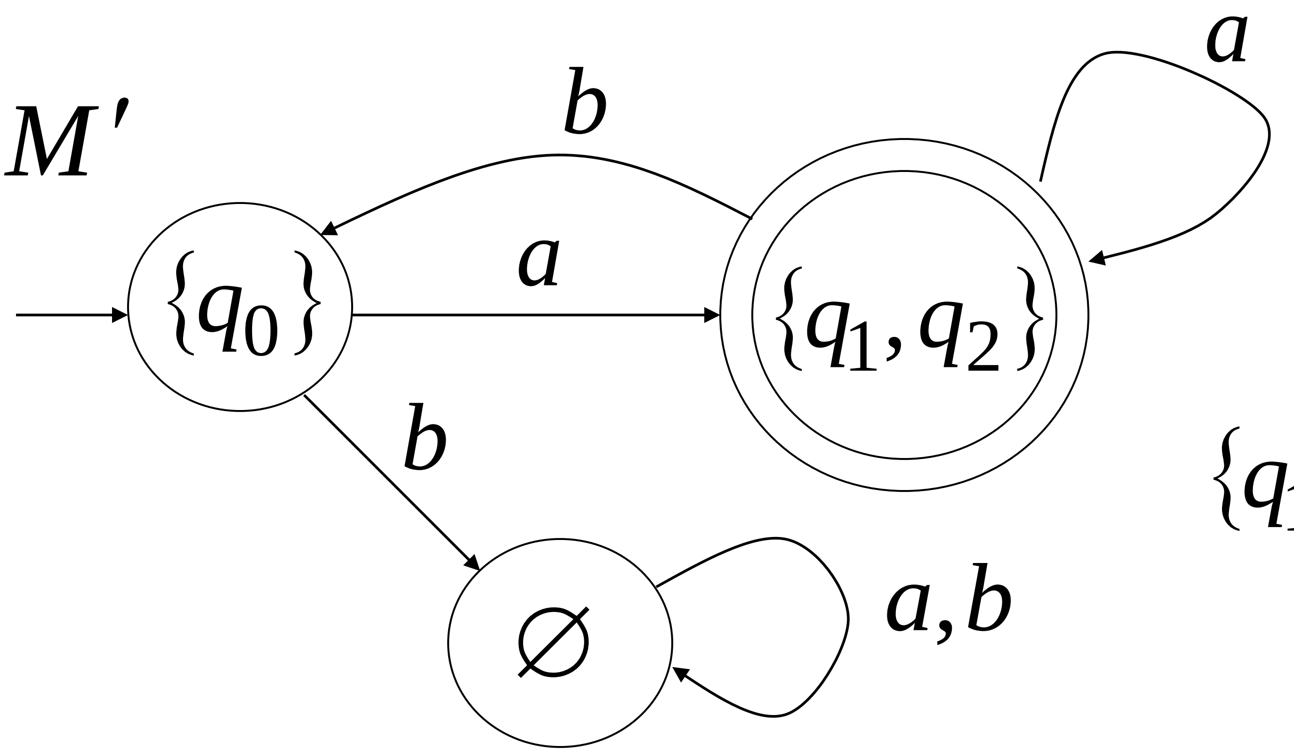
Example

NFA M



$q_1 \in F$

DFA M'



$\{q_1, q_2\} \in F'$

Theorem

Take NFA M

Apply procedure to obtain DFA M'

Then M and M' are equivalent :

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

Finally

We have proven

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

We have proven

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

Regular Languages

We have proven

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

Regular Languages　　　　Regular Languages

We have proven

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

Regular Languages

Regular Languages

Thus, NFAs accept the regular languages