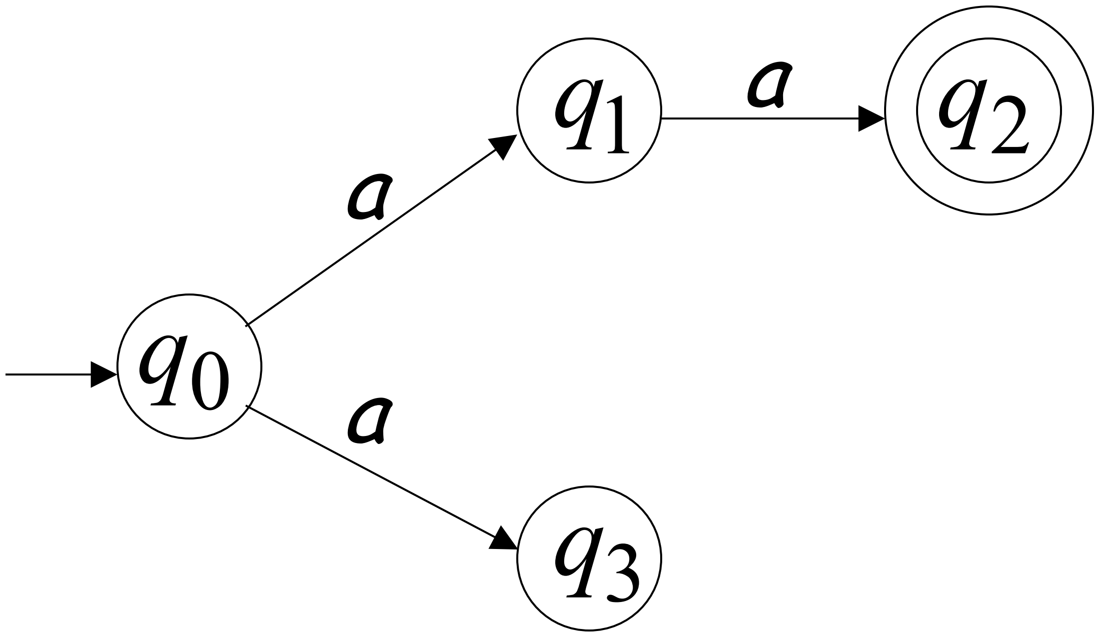
Non-Deterministic Finite Automata

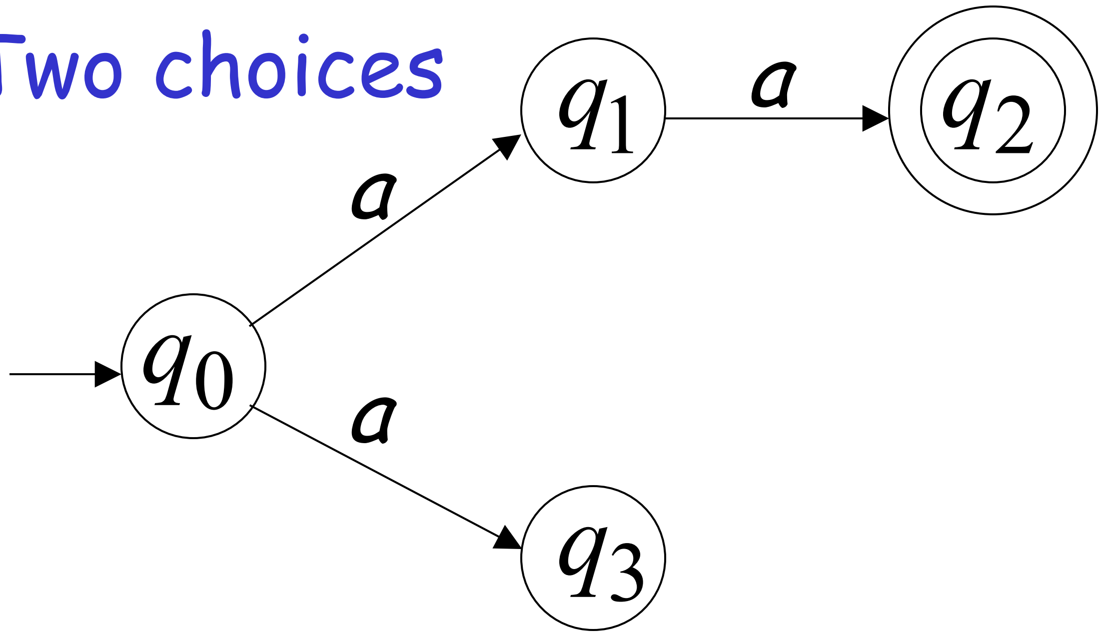
Nondeterministic Finite Automaton (NFA)

Alphabet = $\{a\}$

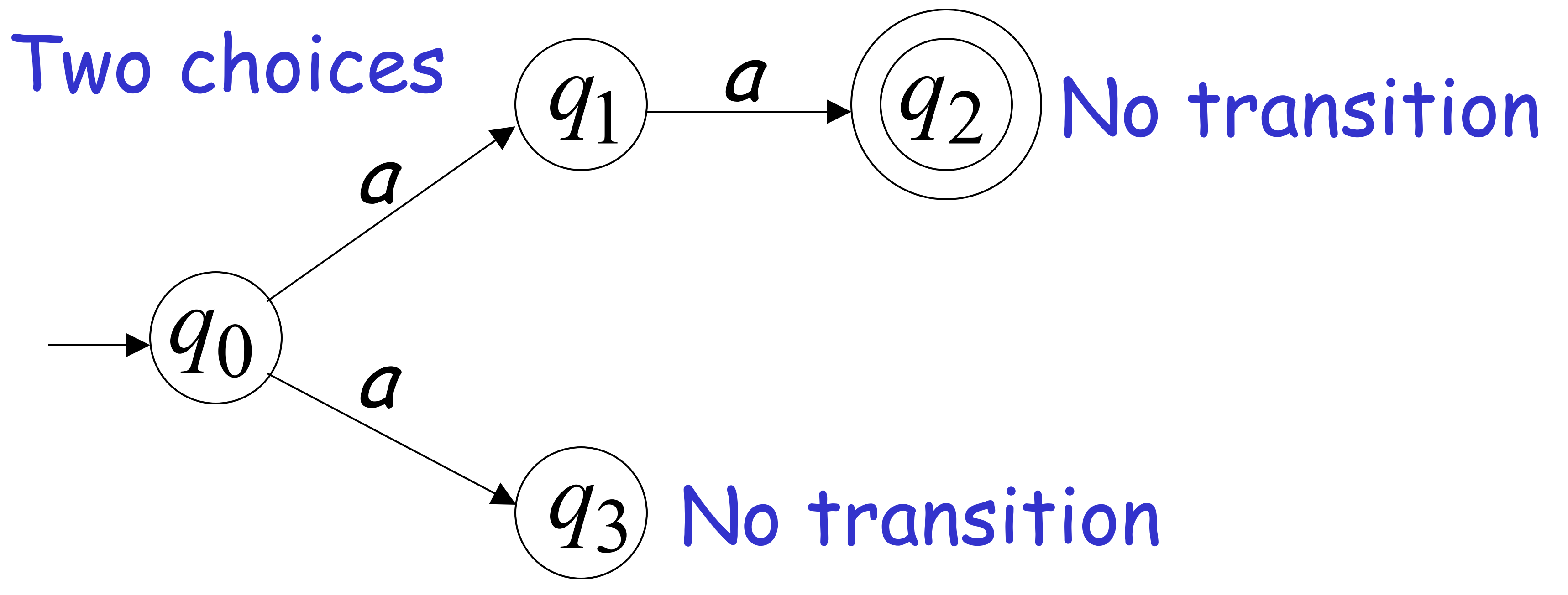


Alphabet = $\{a\}$

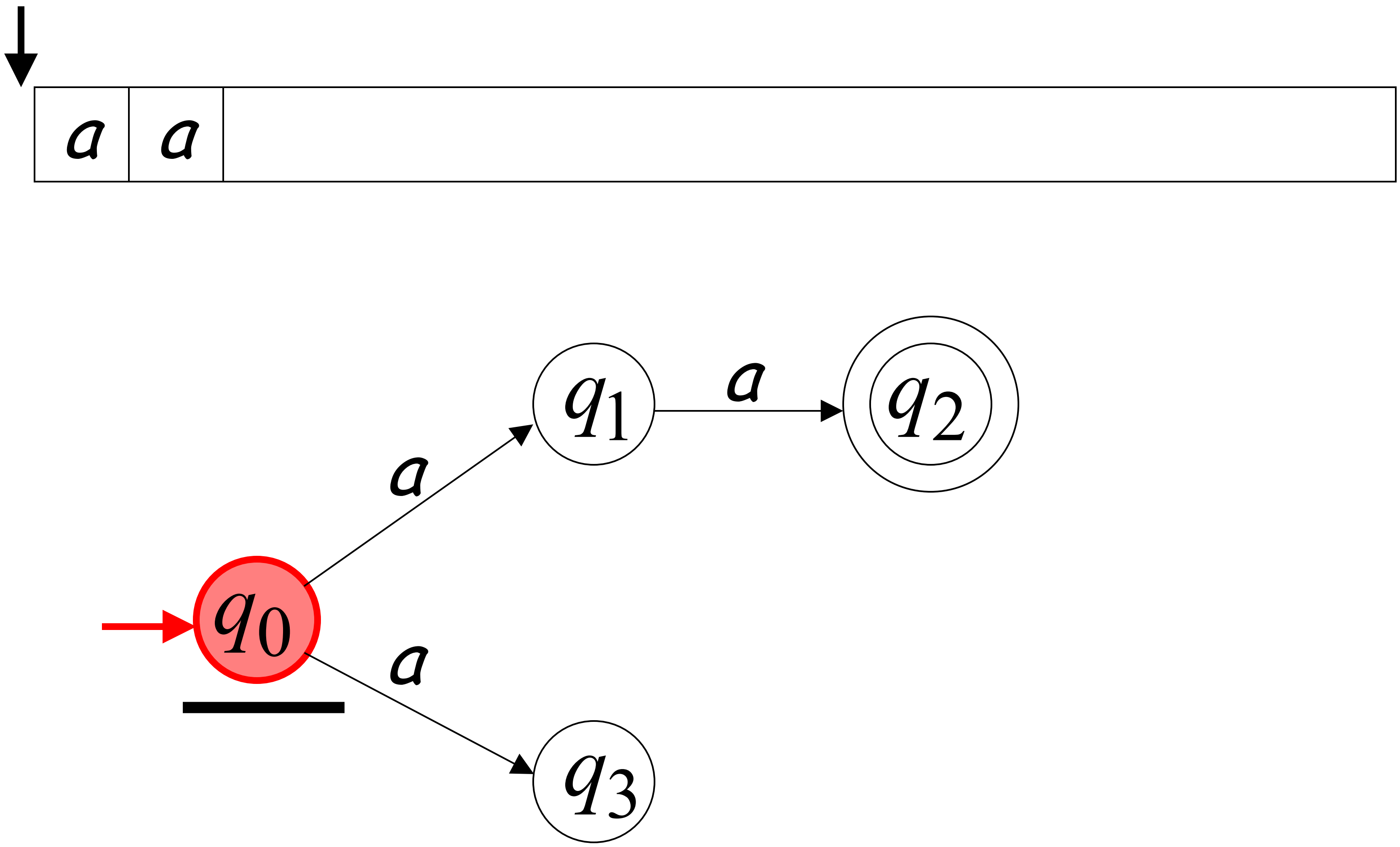
Two choices



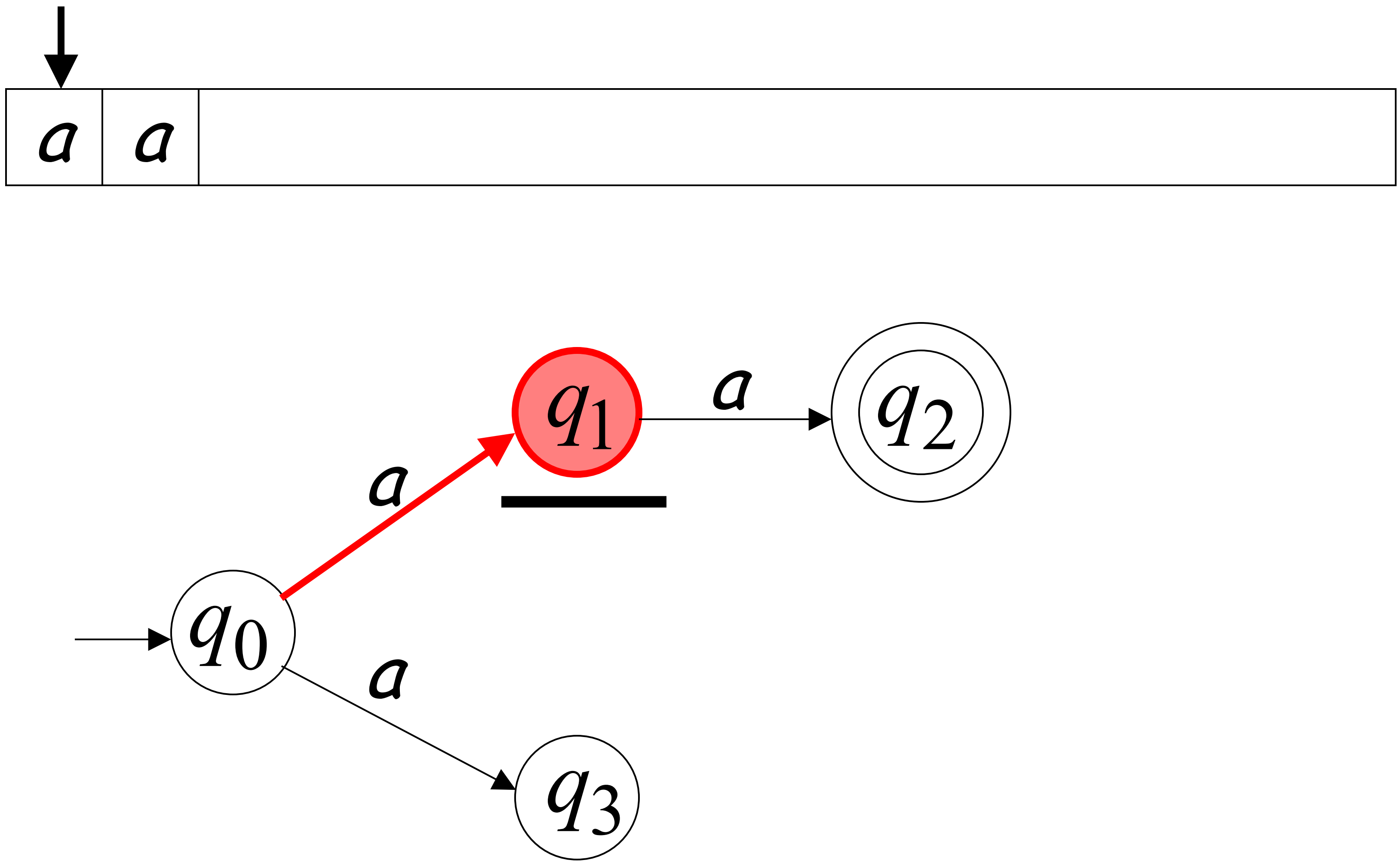
Alphabet = $\{a\}$



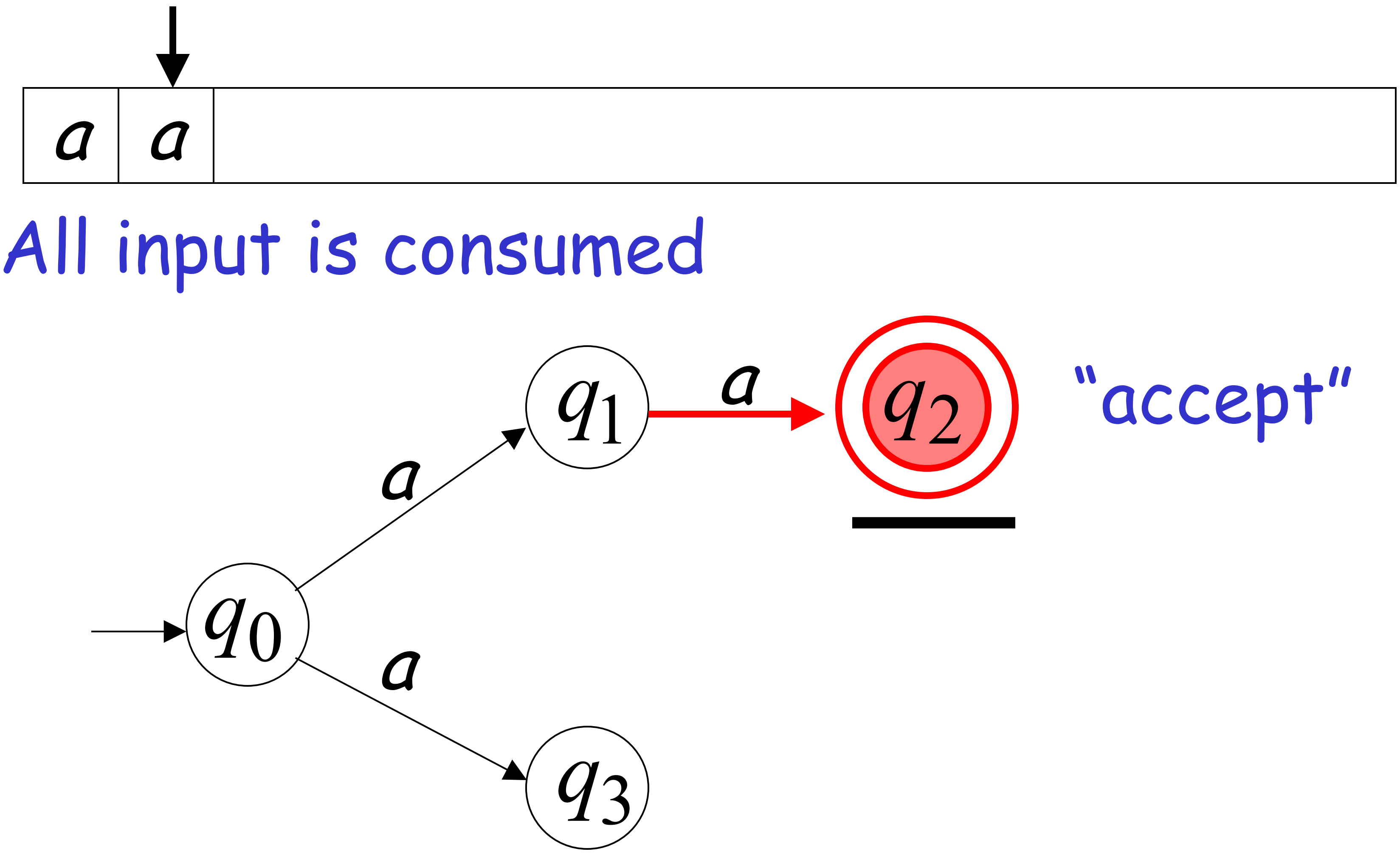
First Choice



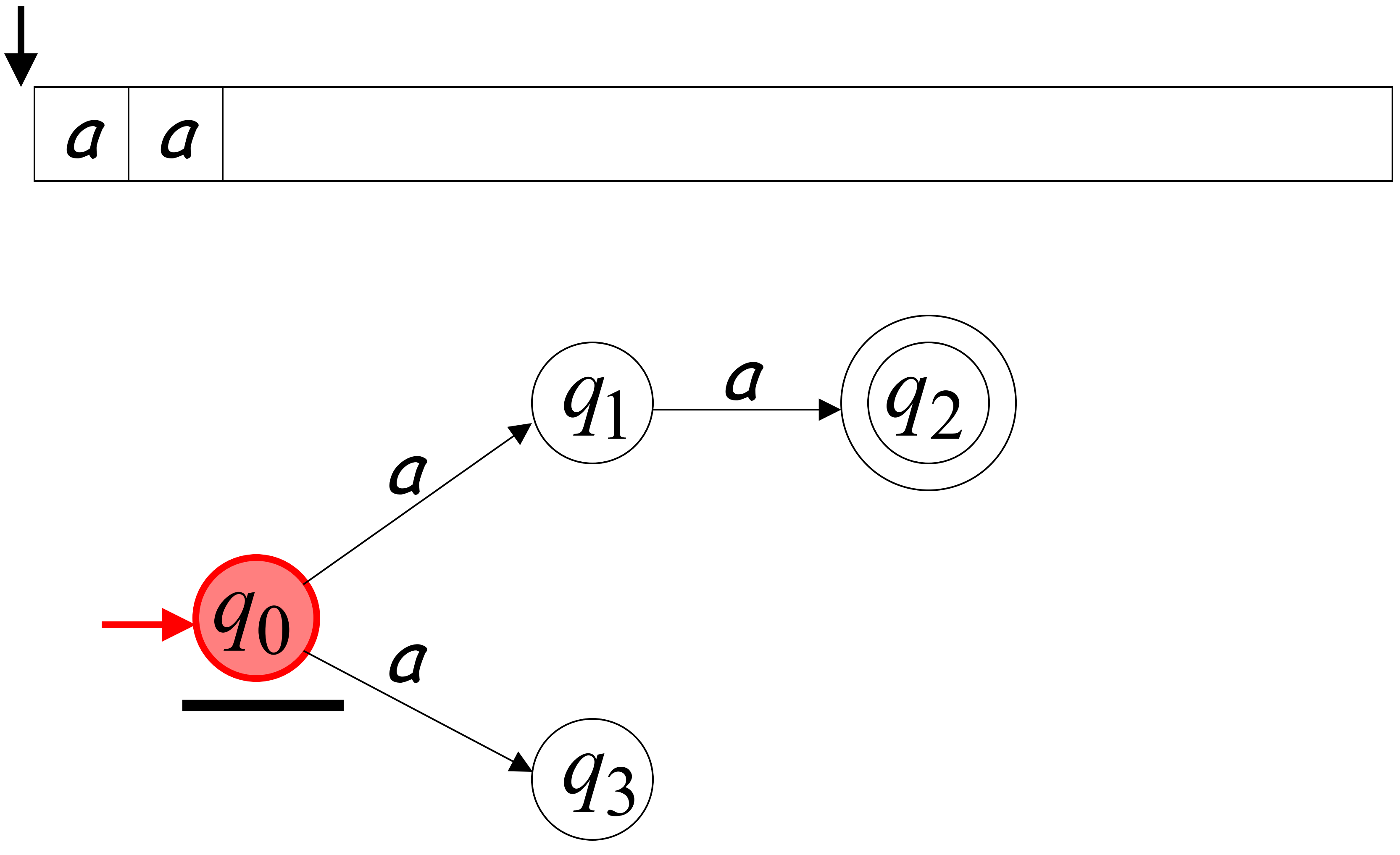
First Choice



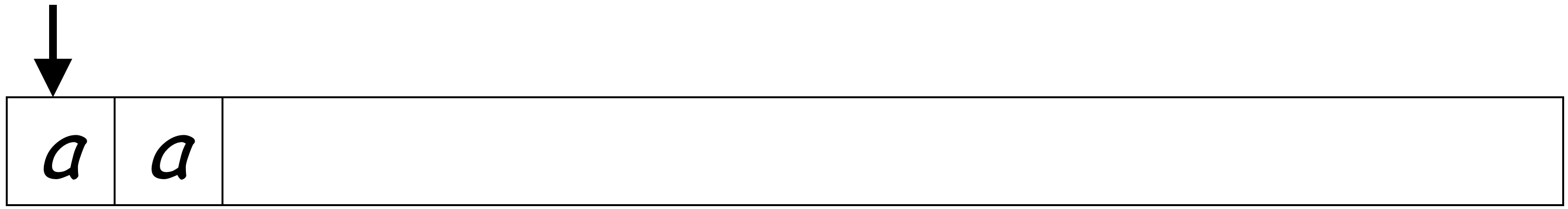
First Choice



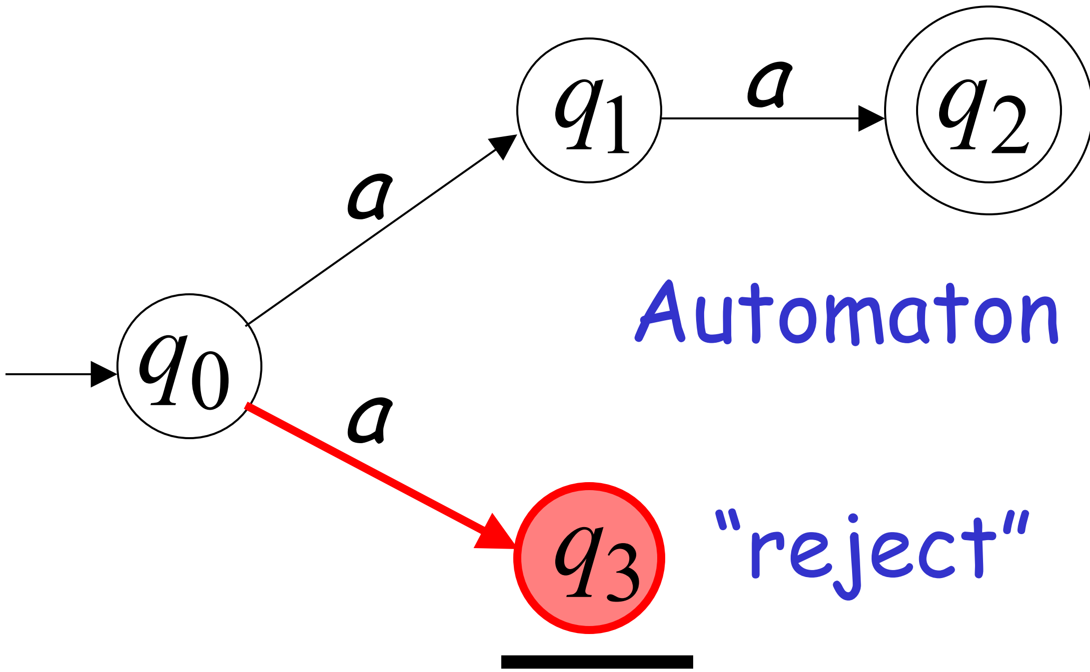
Second Choice



Second Choice



Input cannot be consumed



Automaton Halts

"reject"

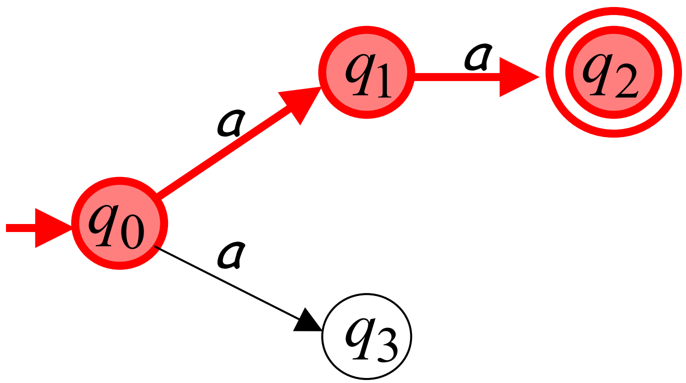
An NFA accepts a string:

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

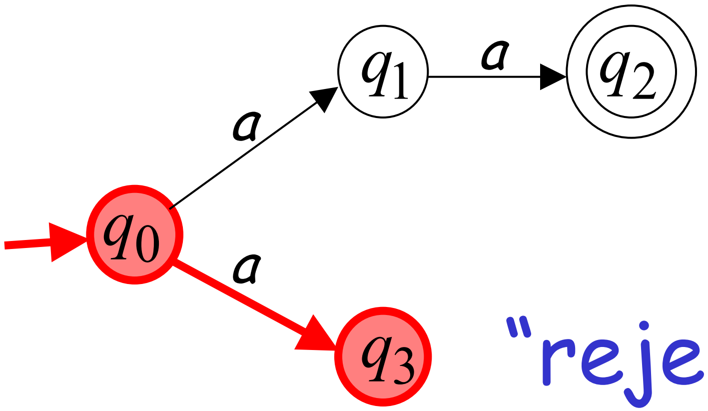
i.e., all the input string is processed and the automaton is in an accepting state

aa is accepted by the NFA:

"accept"



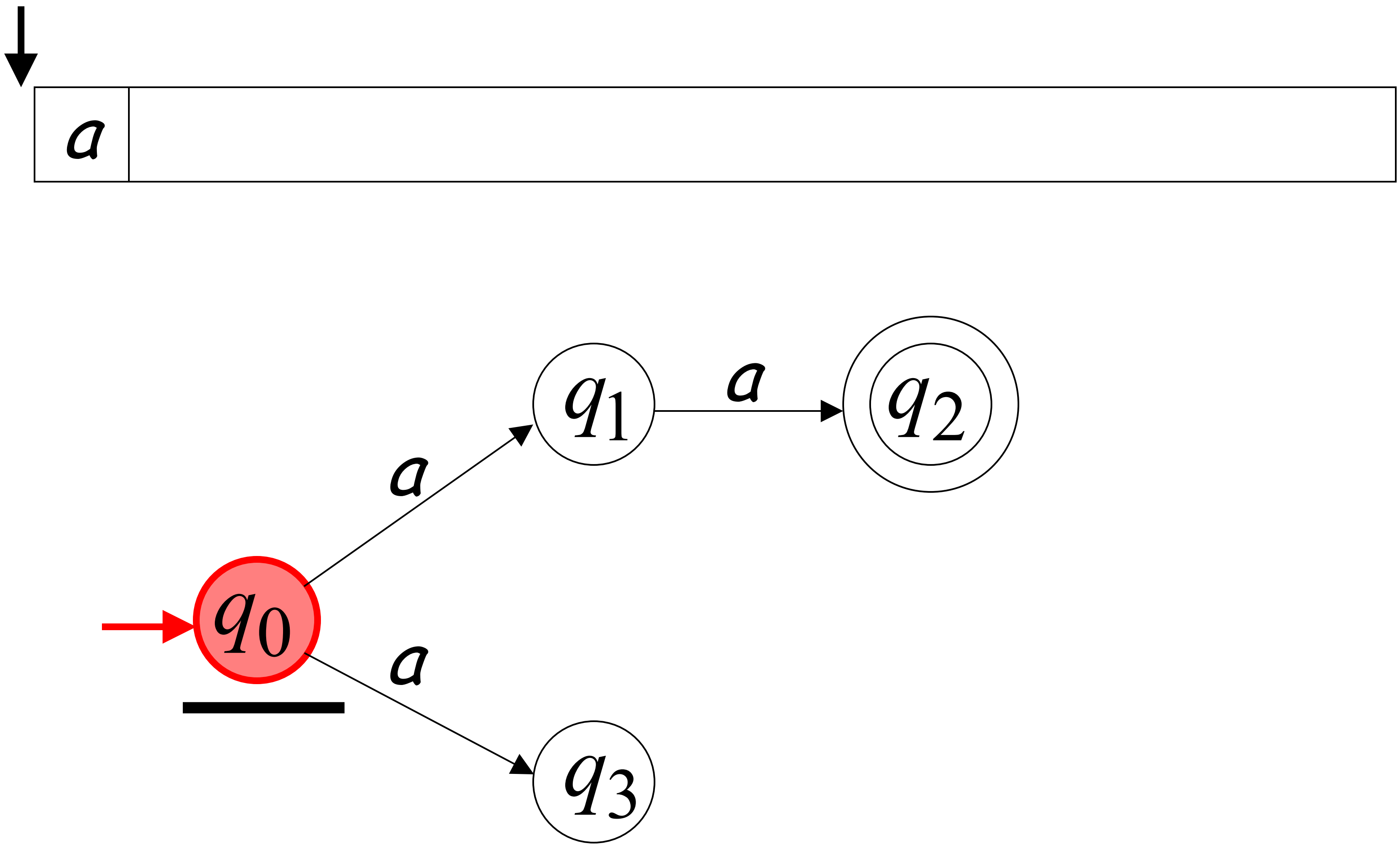
because this computation accepts aa



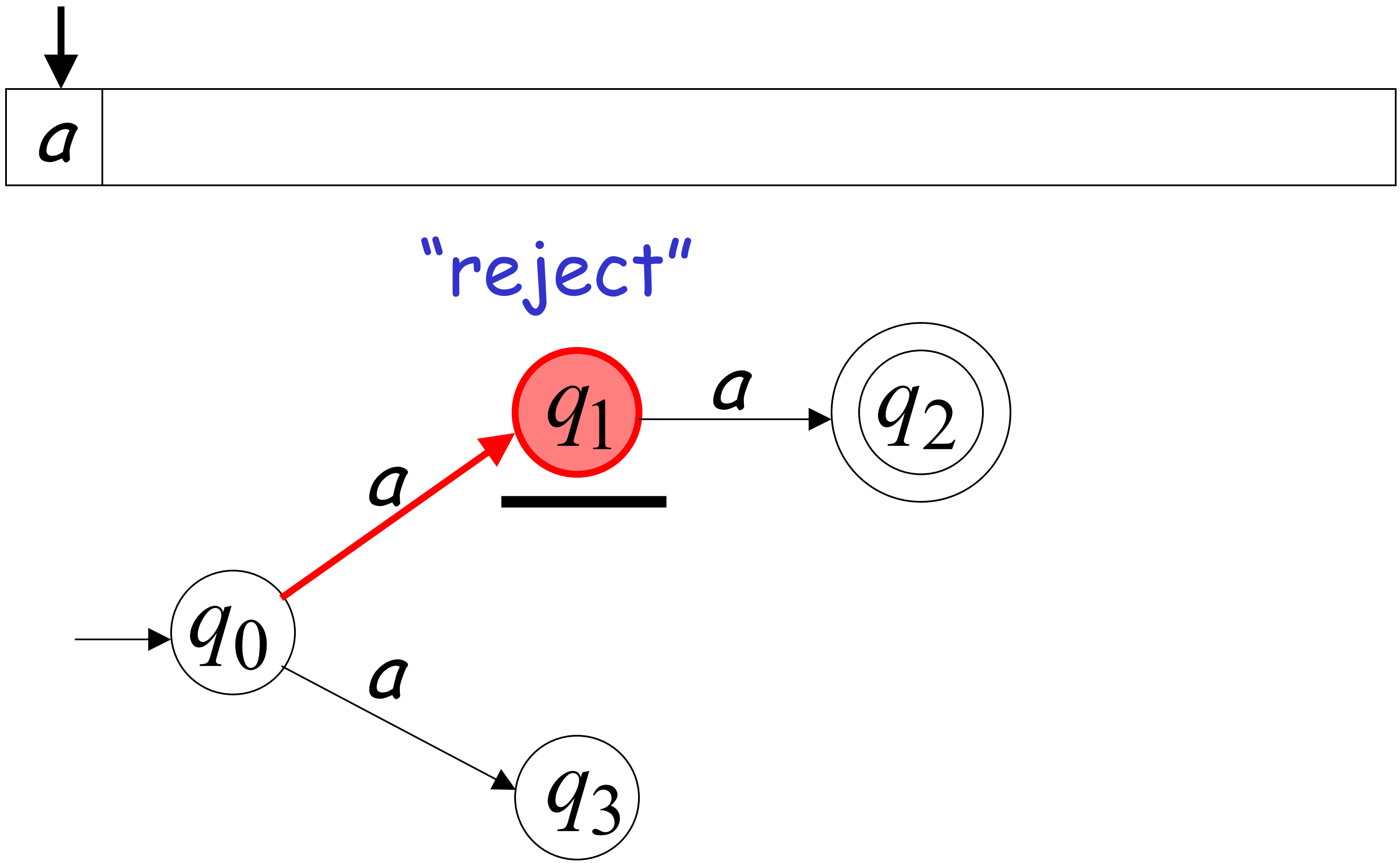
"reject"

this computation is ignored

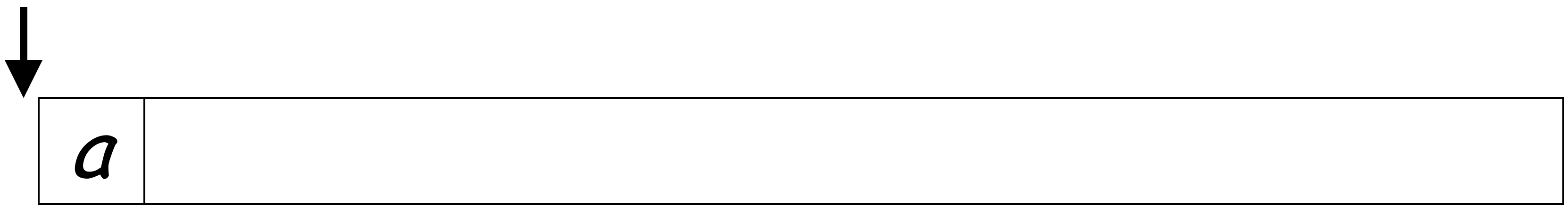
Rejection example

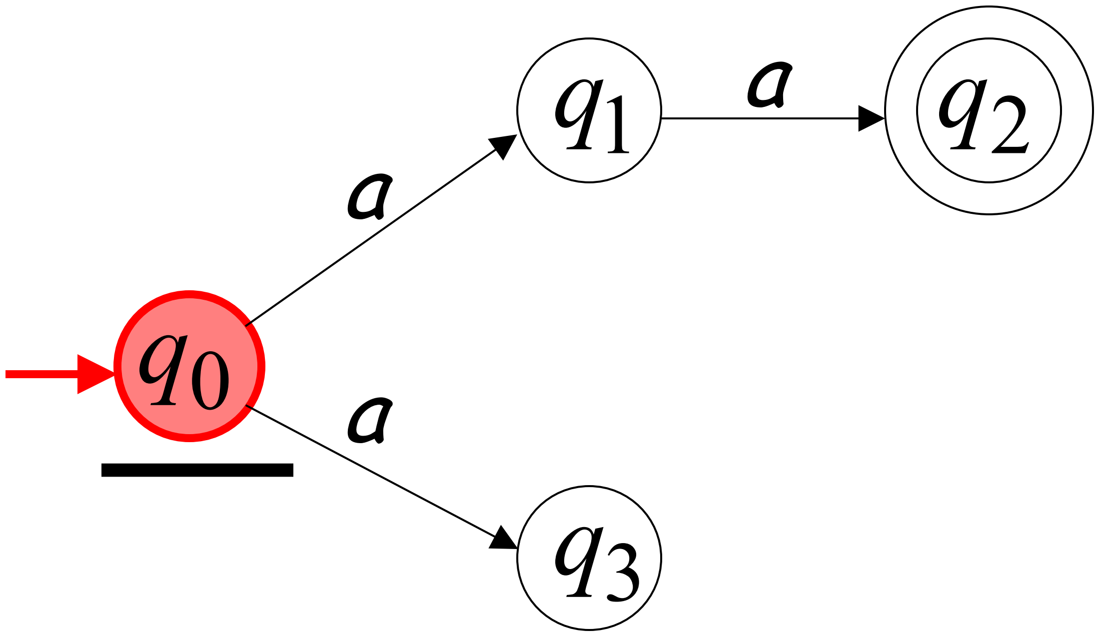


First Choice

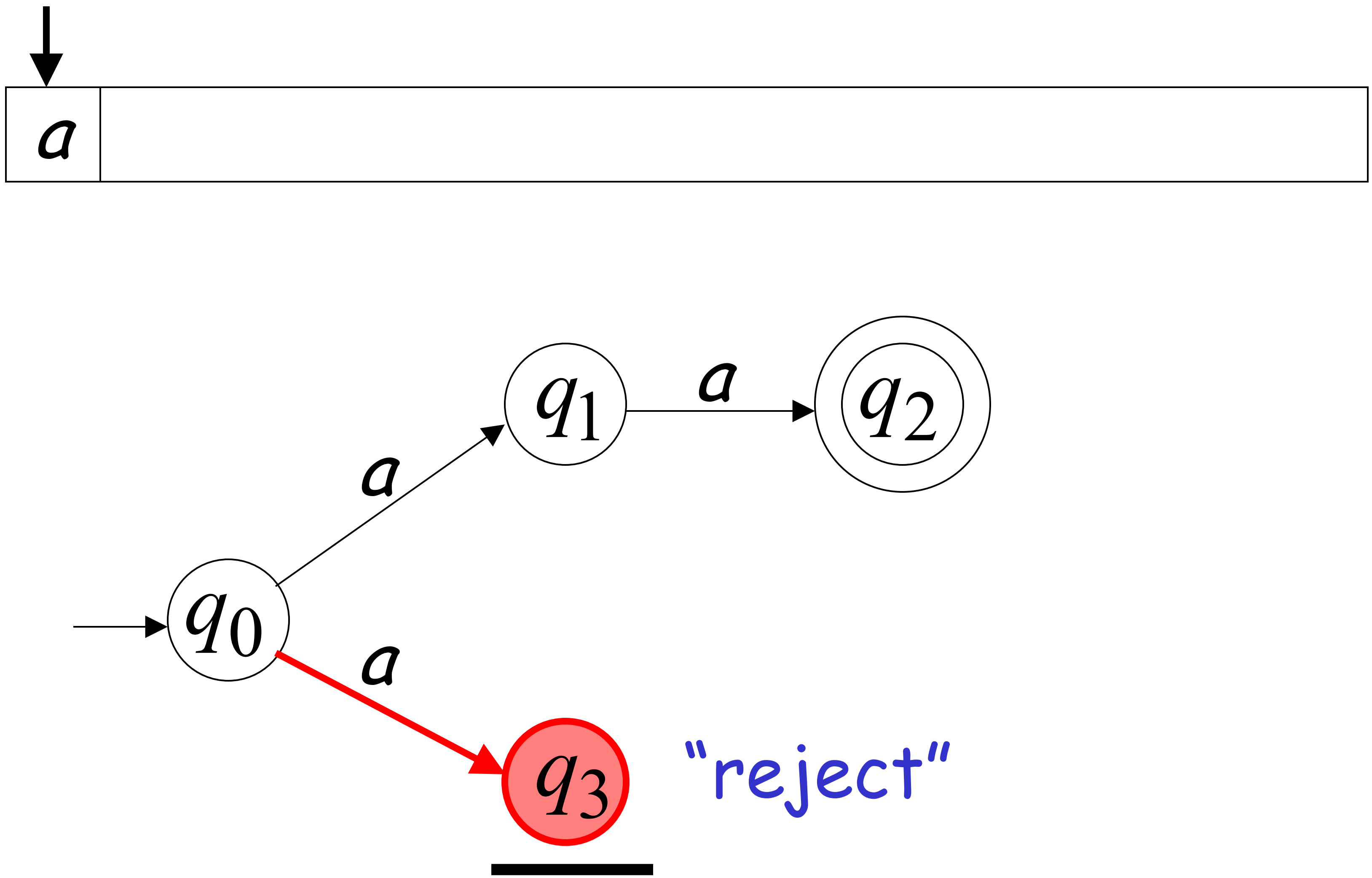


Second Choice

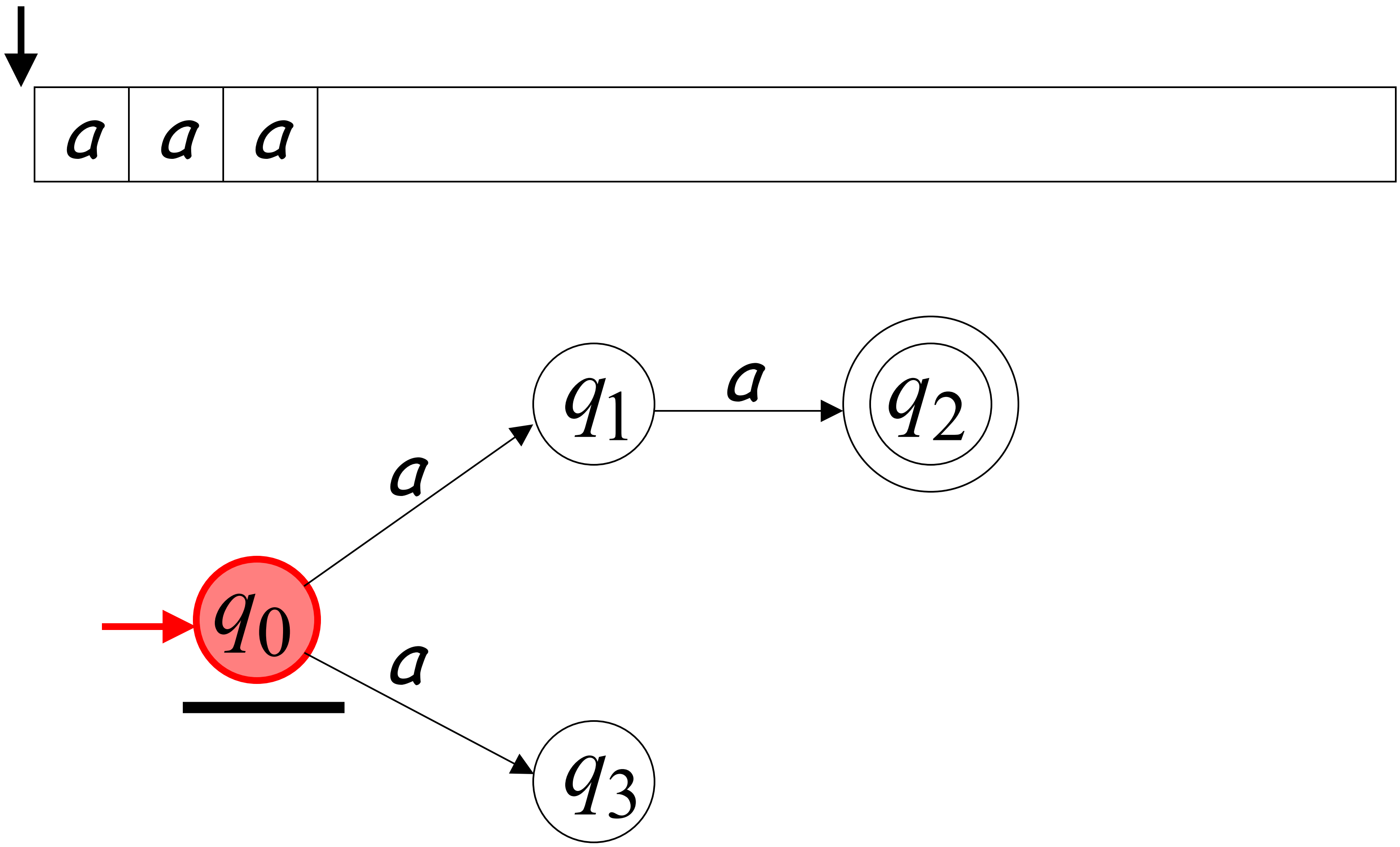




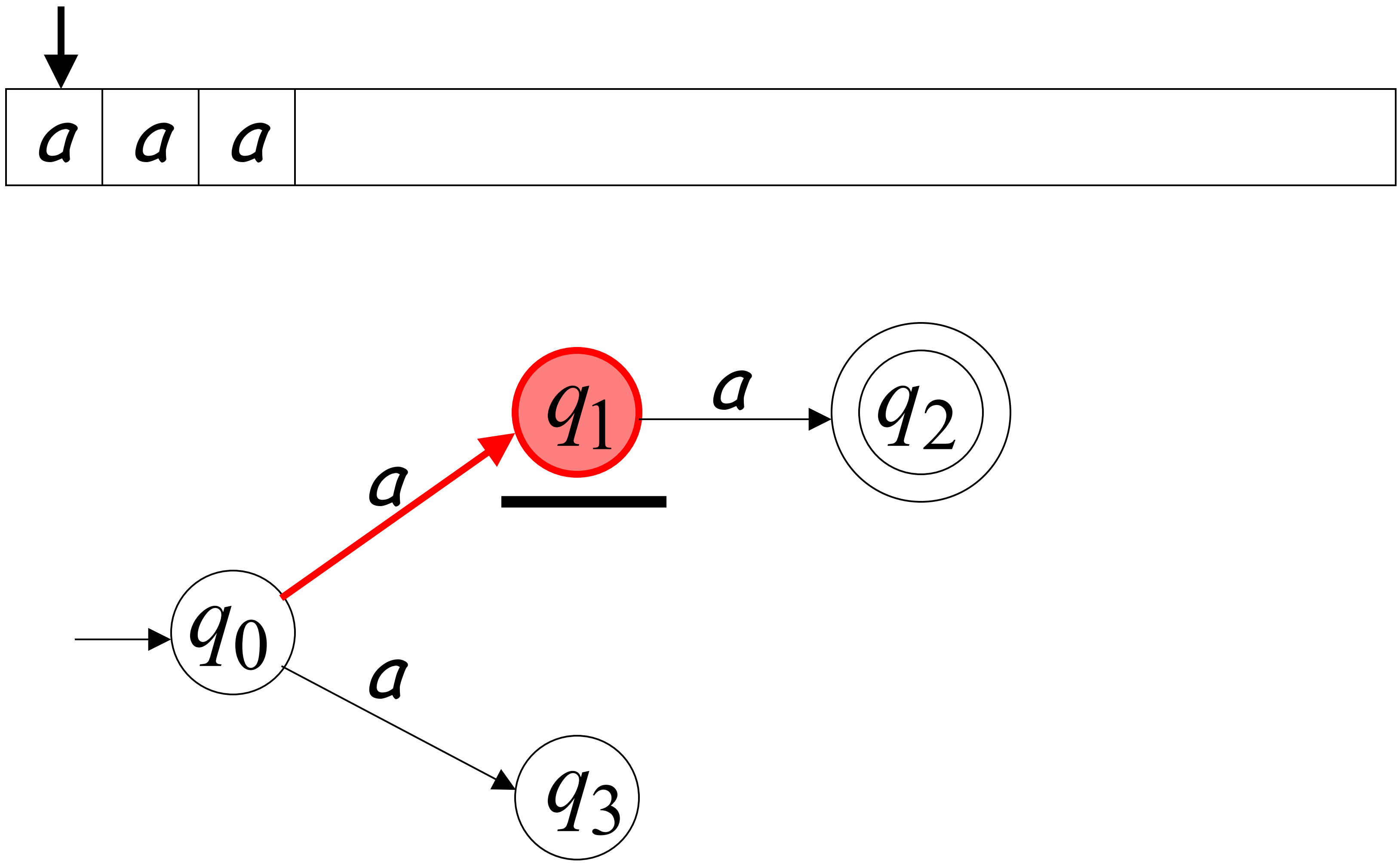
Second Choice



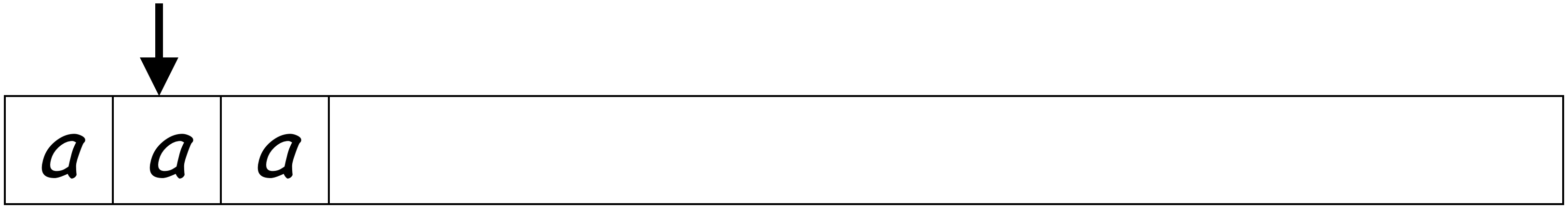
Another Rejection example



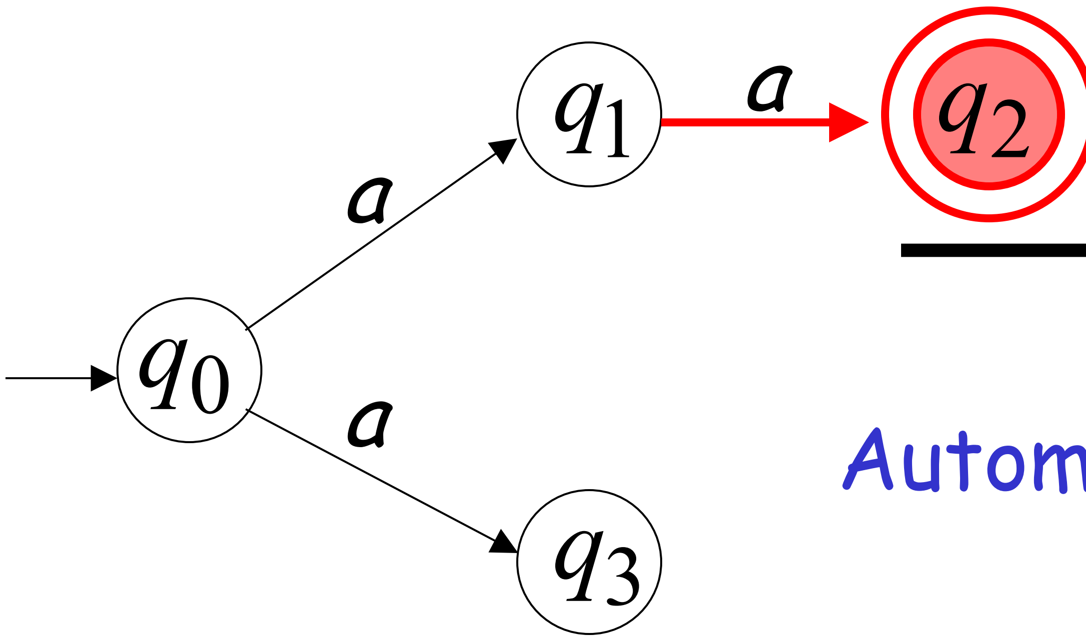
First Choice



First Choice



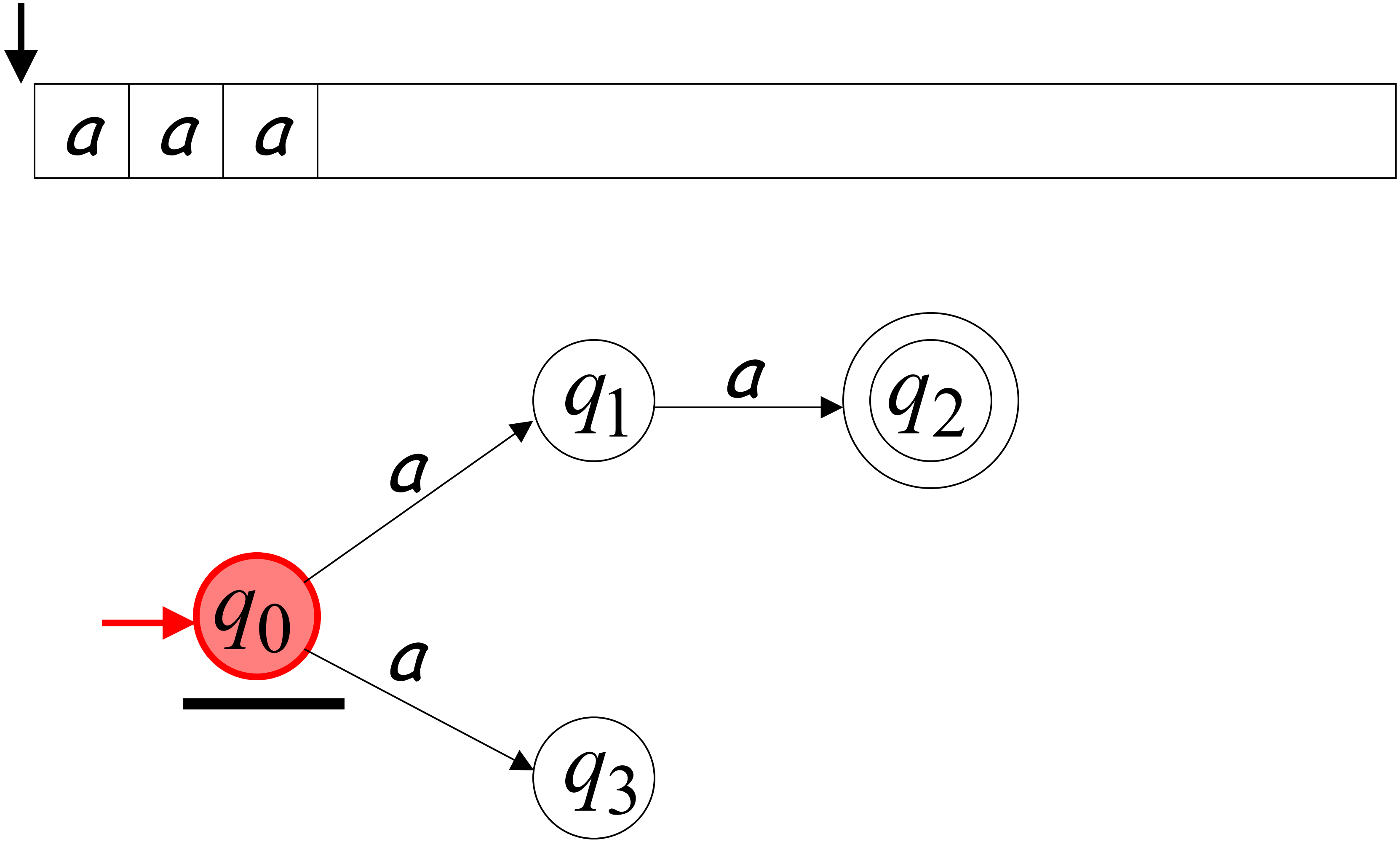
Input cannot be consumed



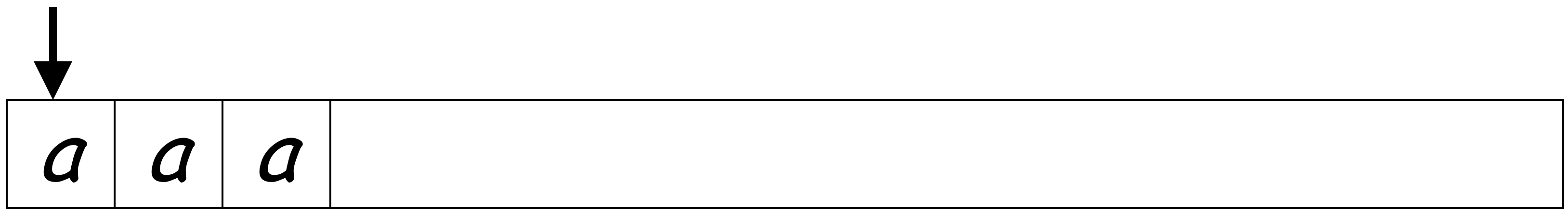
"reject"

Automaton halts

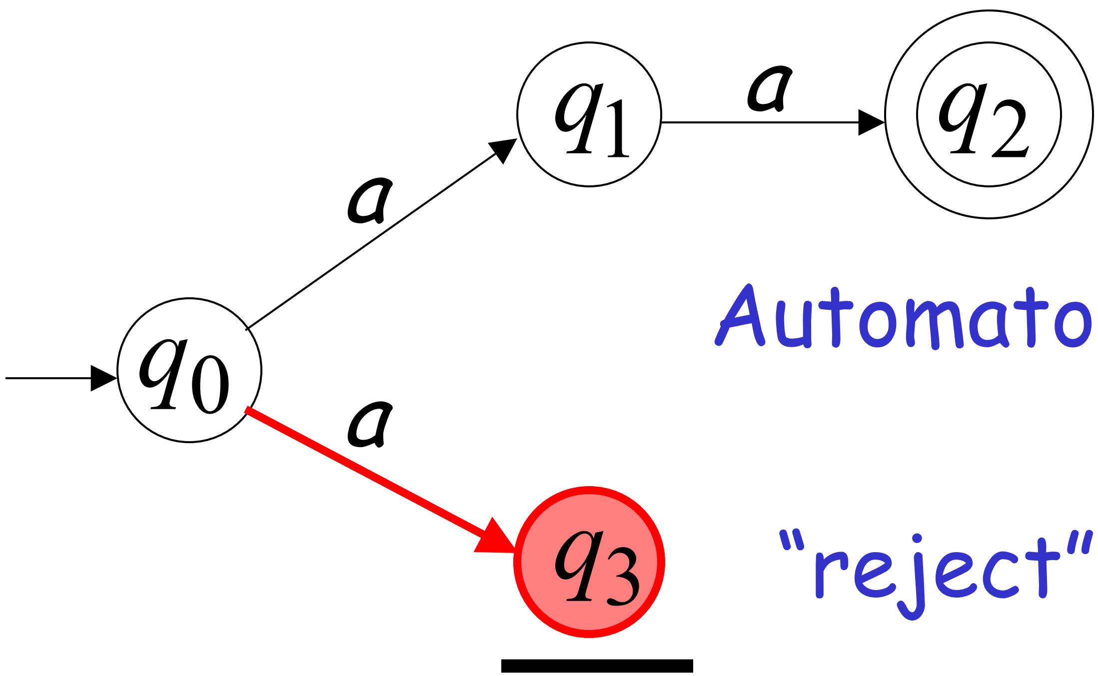
Second Choice



Second Choice



Input cannot be consumed



Automaton halts

"reject"

An NFA rejects a string:

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

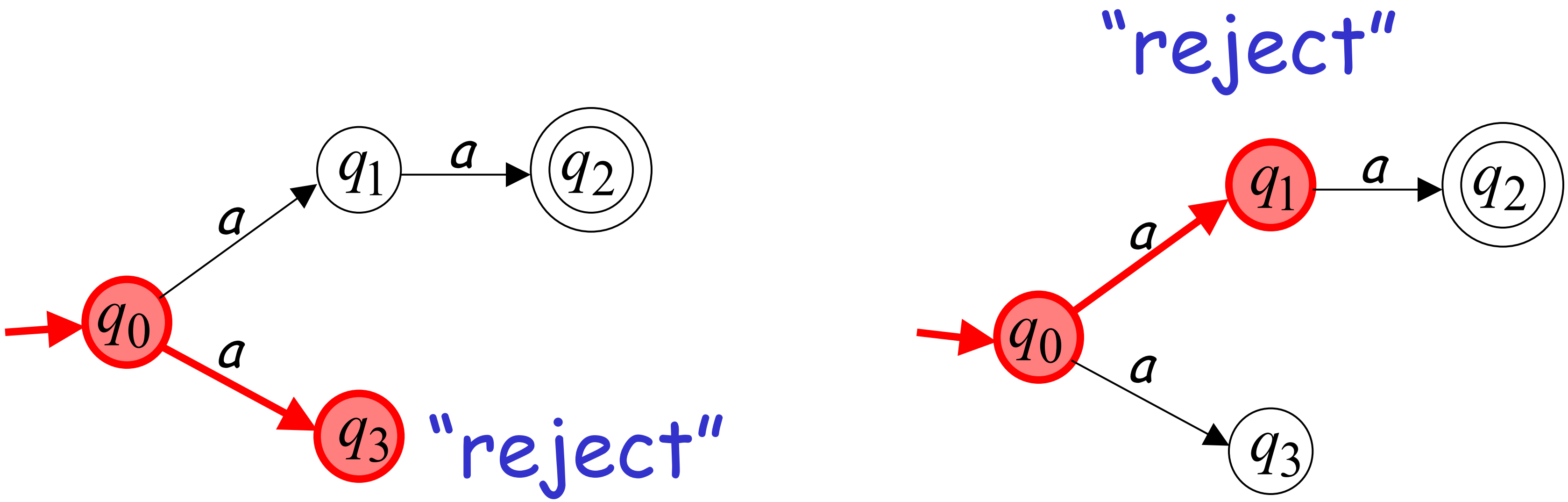
For each computation:

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

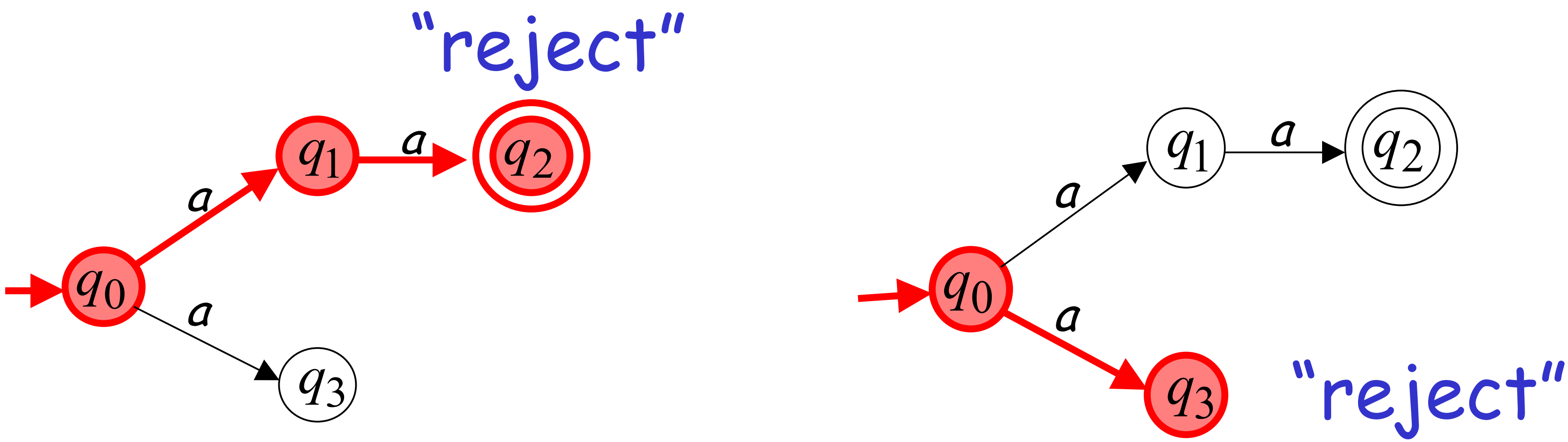
OR

- The input cannot be consumed

a is rejected by the NFA:

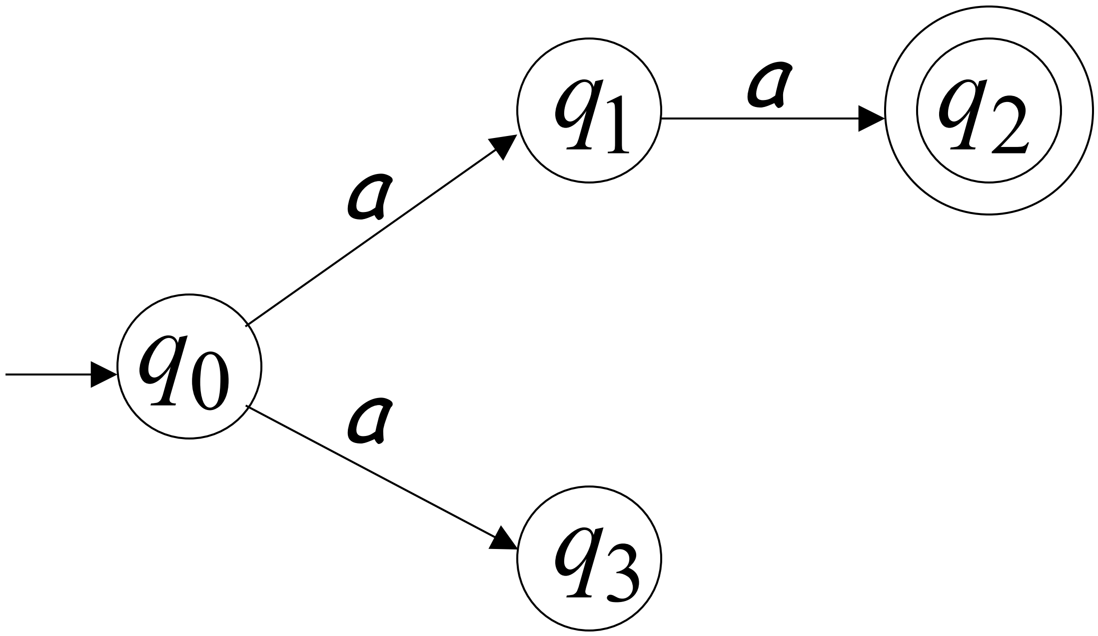


All possible computations lead to rejection aaa is rejected by the NFA:

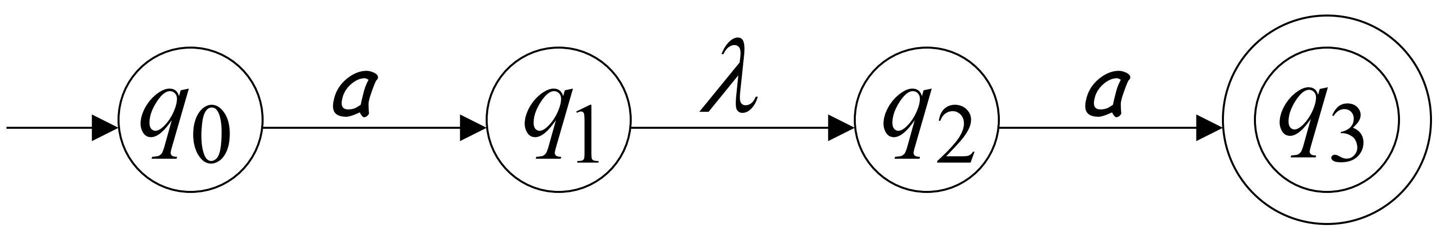


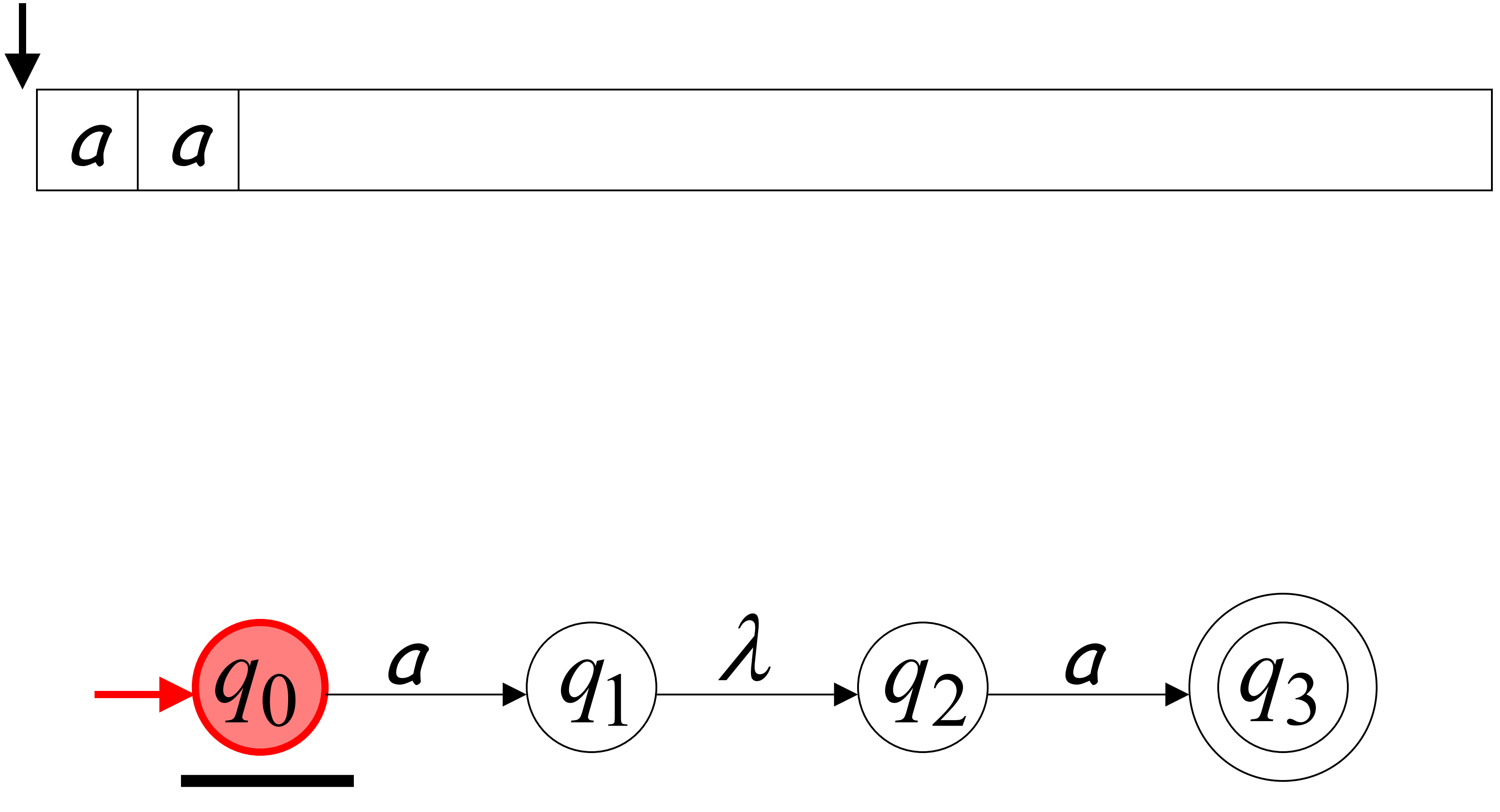
All possible computations lead to rejection

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

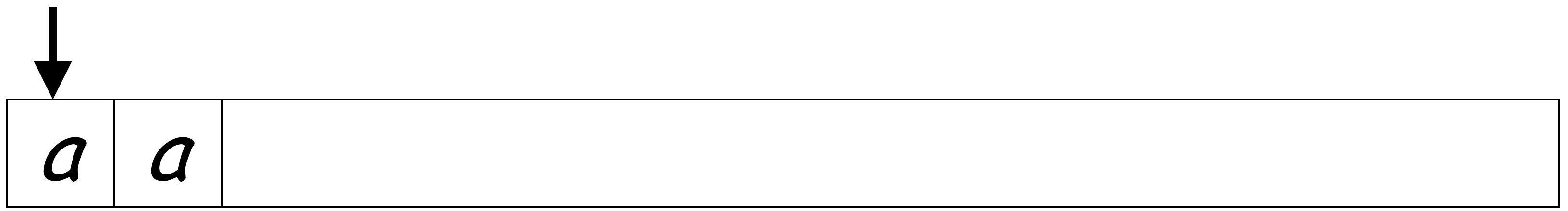


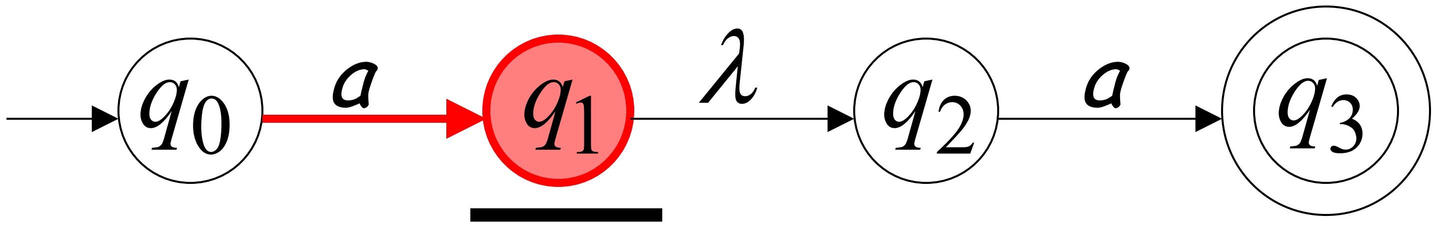
Lambda Transitions



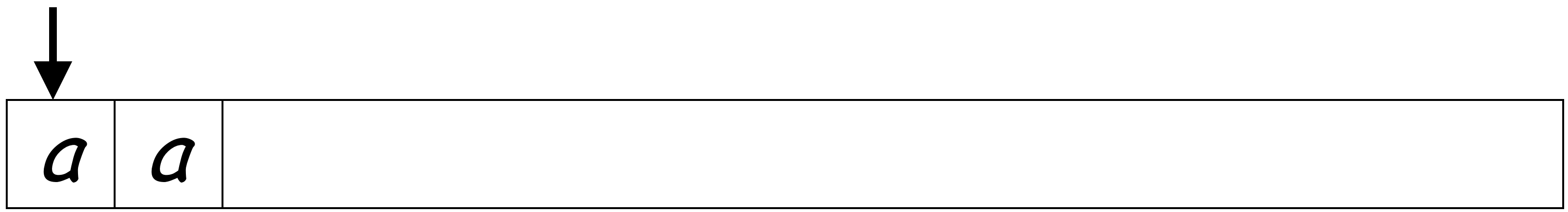


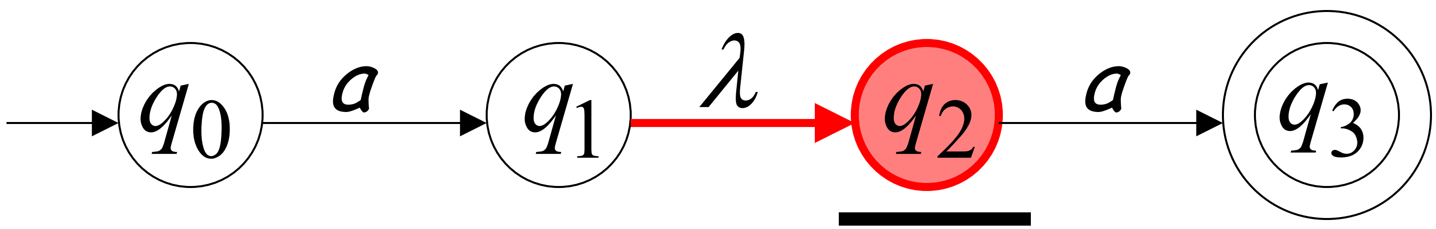
a
a
a
λ
a
q_0
q_1
q_2
q_3


a
a

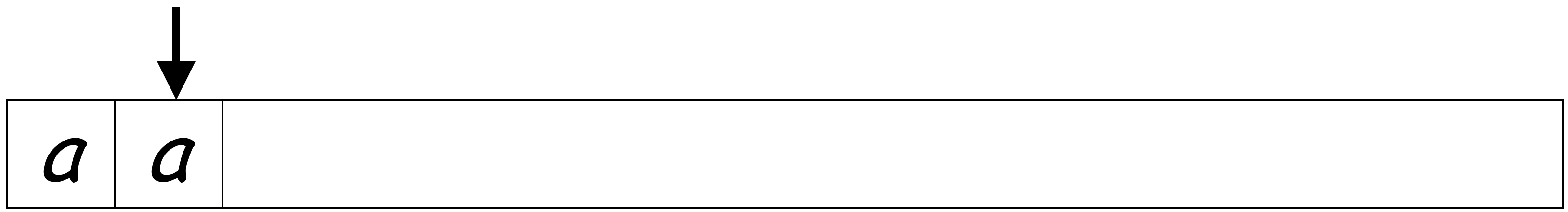

q_0
a
q_1
λ
q_2
a
q_3

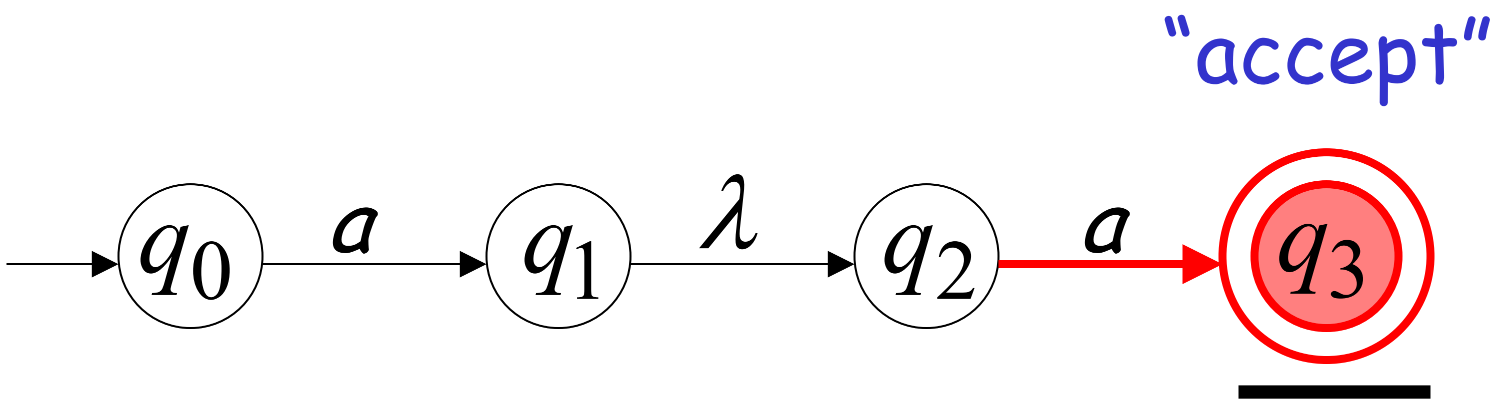
input tape head does not move





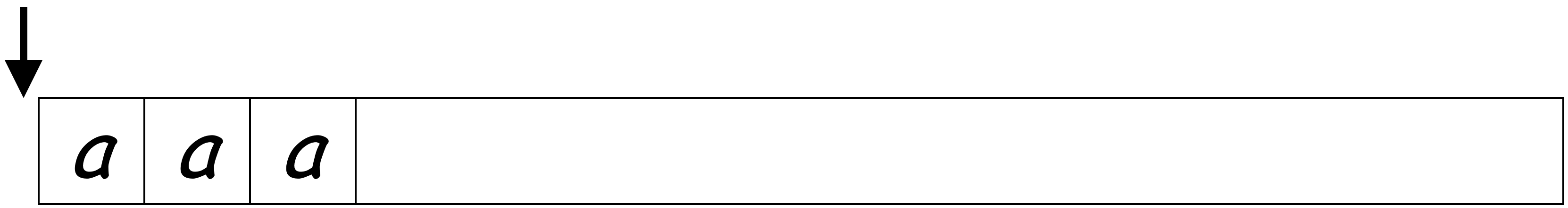
all input is consumed

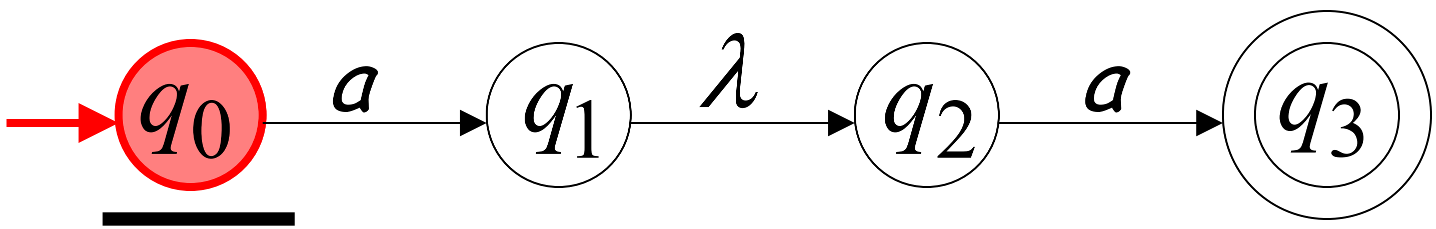


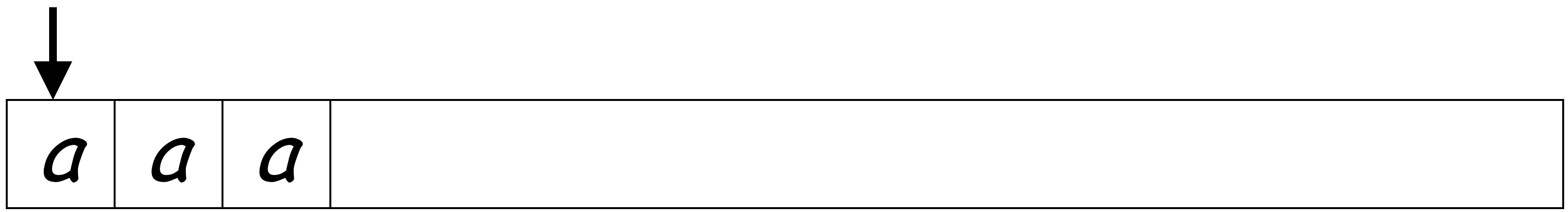


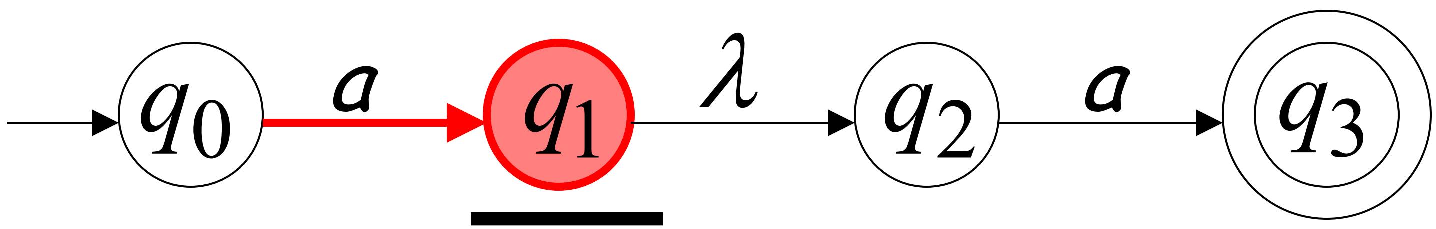
String aa is accepted

Rejection Example

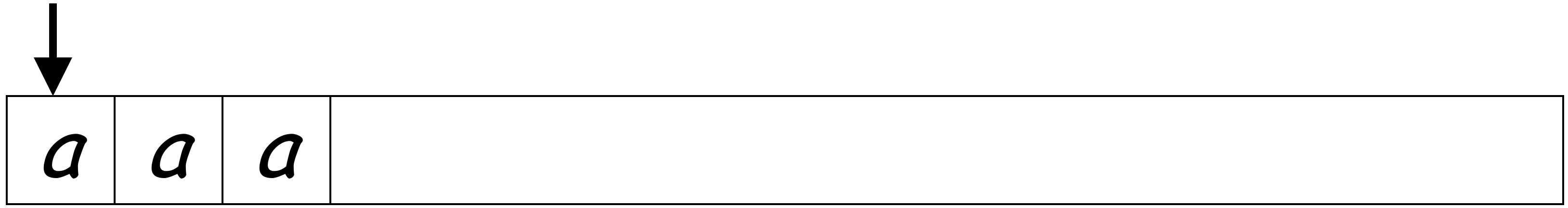


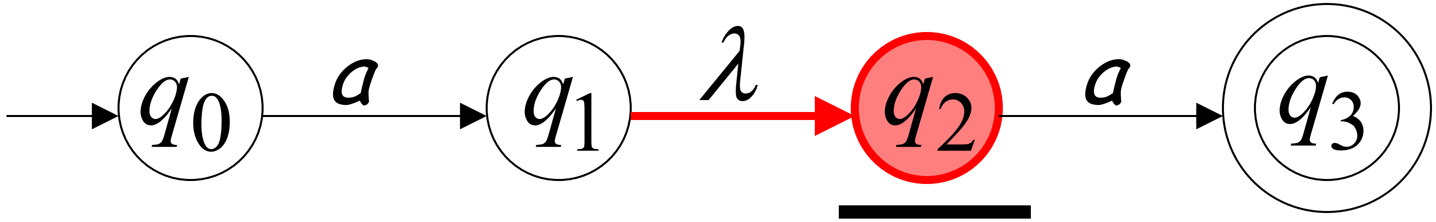



a
a
a

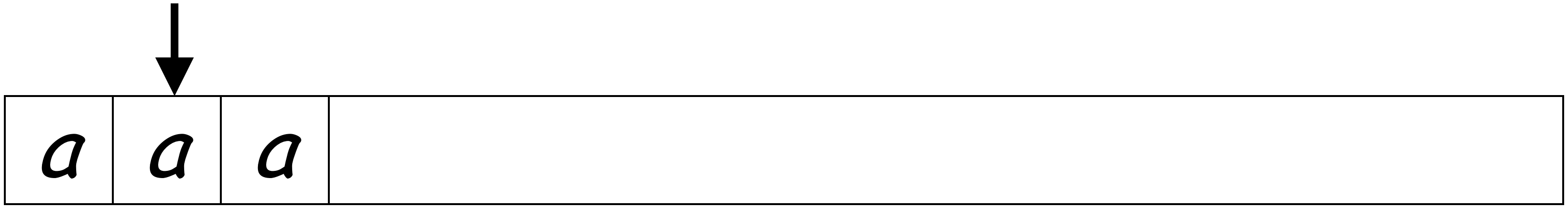

q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)



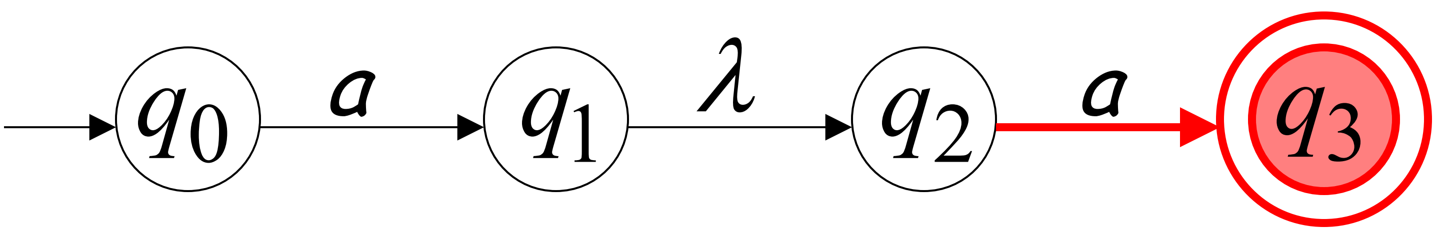


Input cannot be consumed



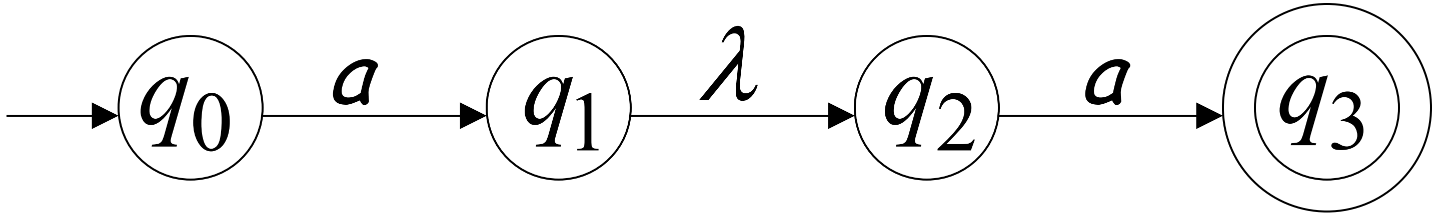
Automaton halts

"reject"

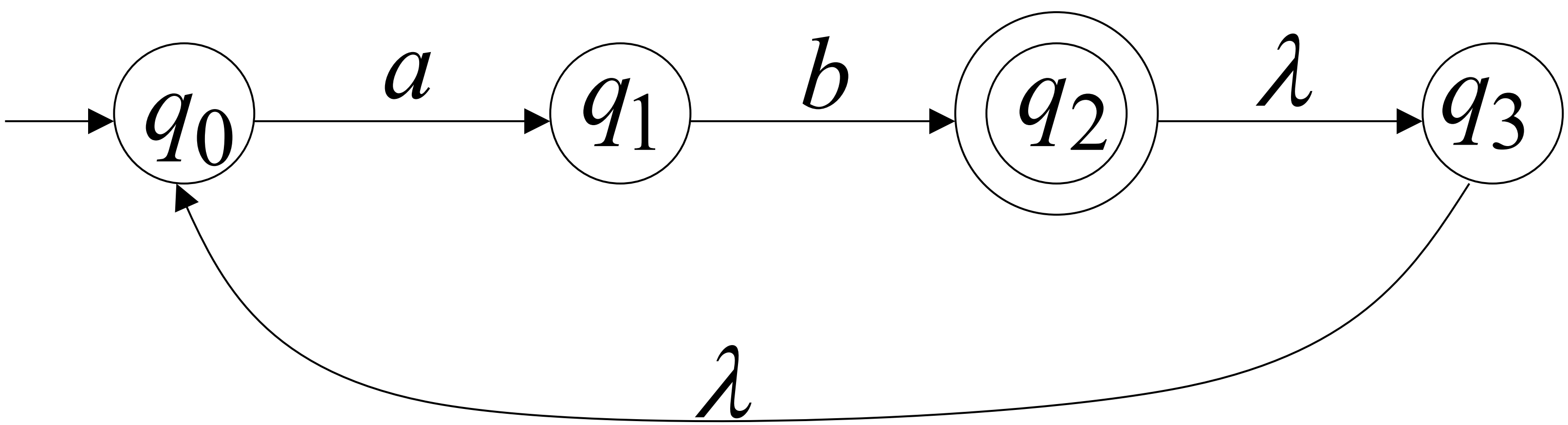


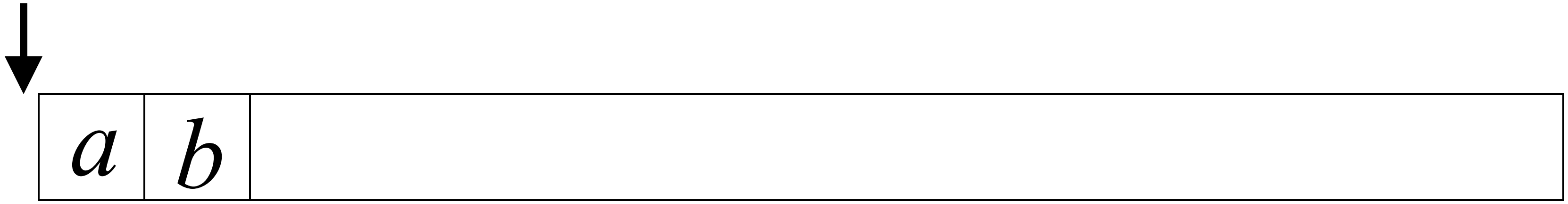
String **aaa** is rejected

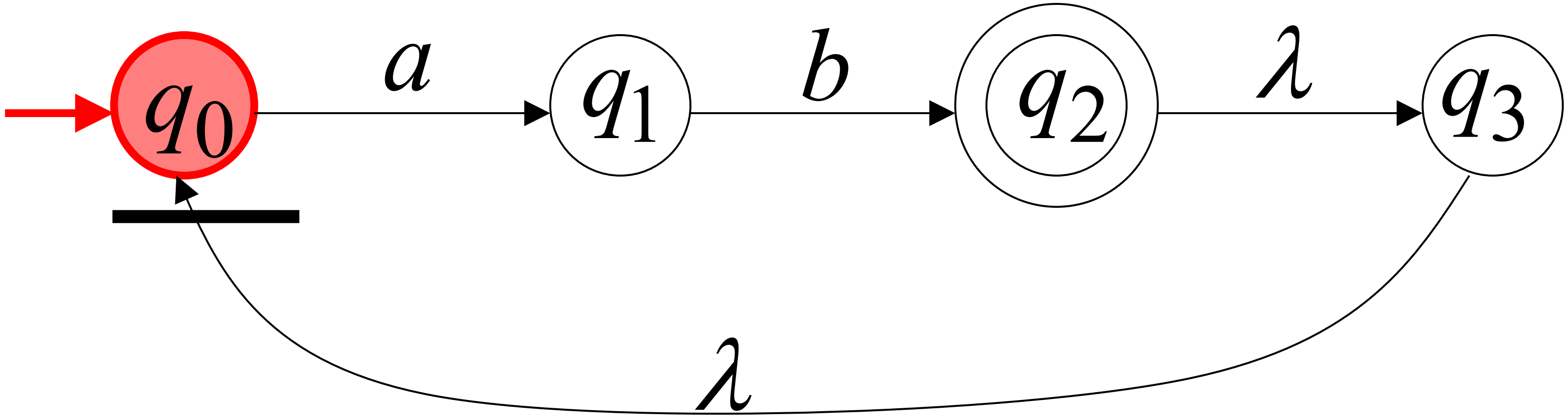
Language accepted: $L = \{aa\}$

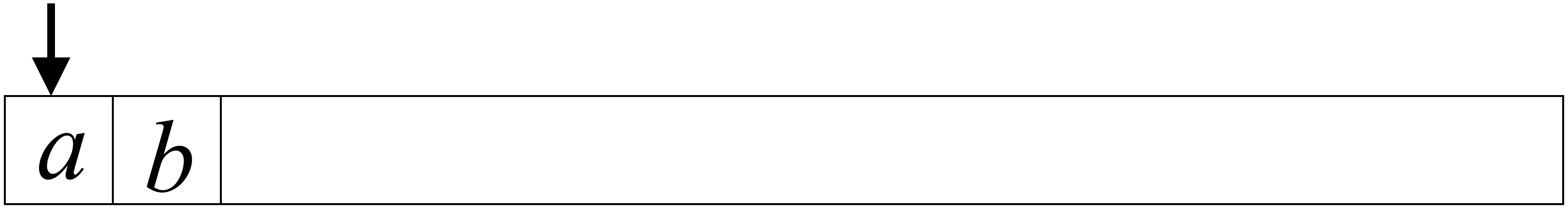


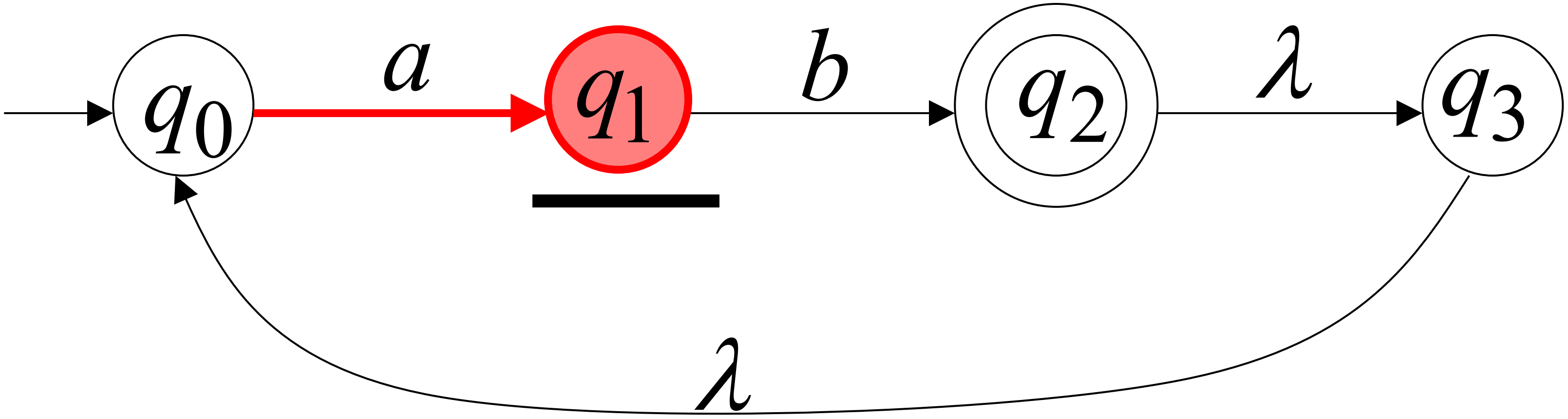
Another NFA Example

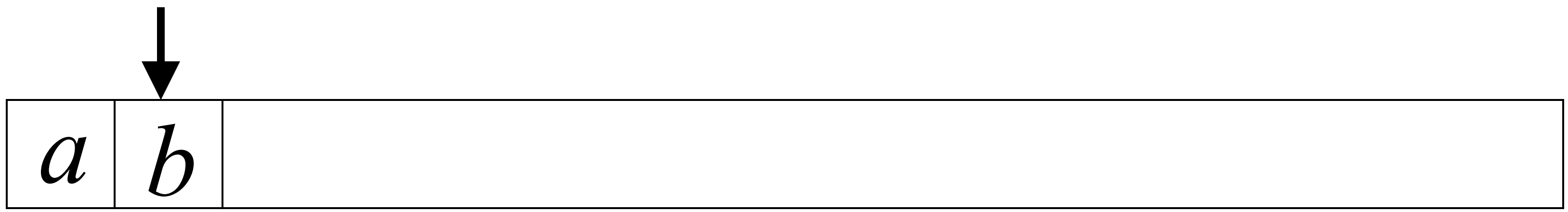


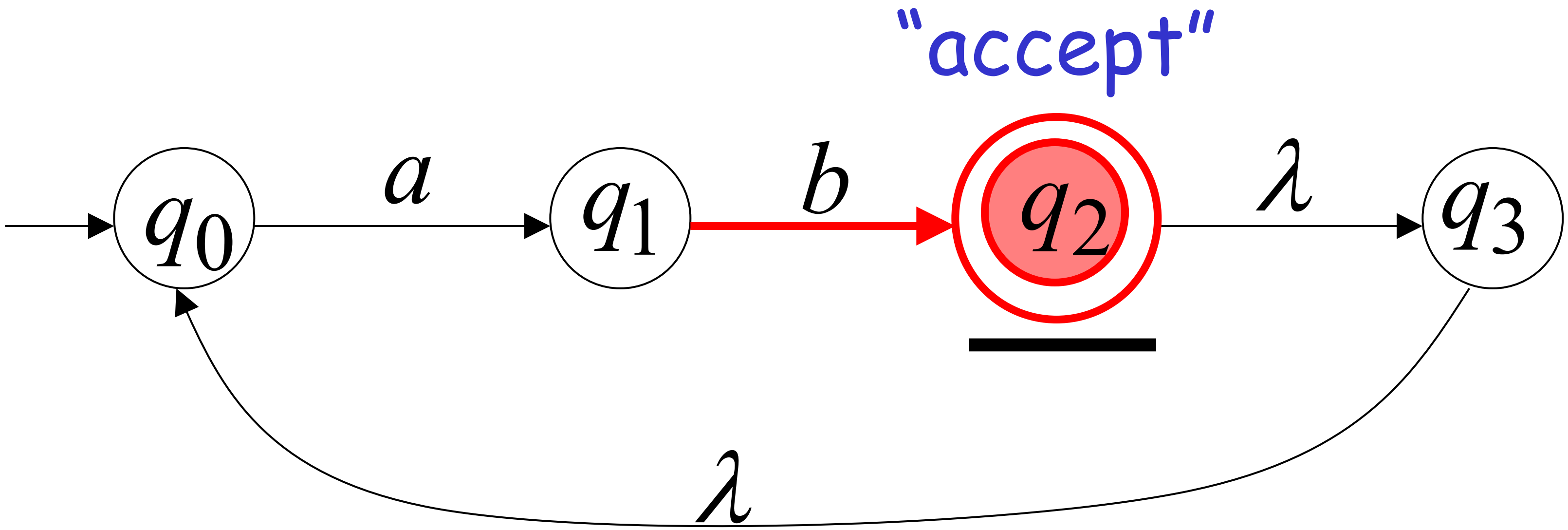

a
b


a
b
λ
λ


a
b

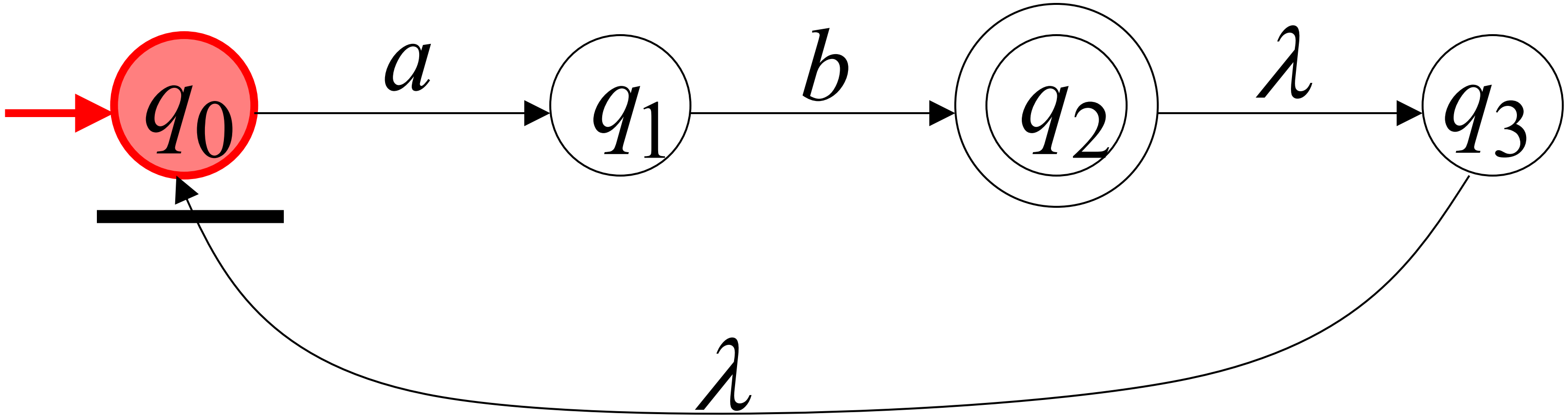

a
b
λ
λ

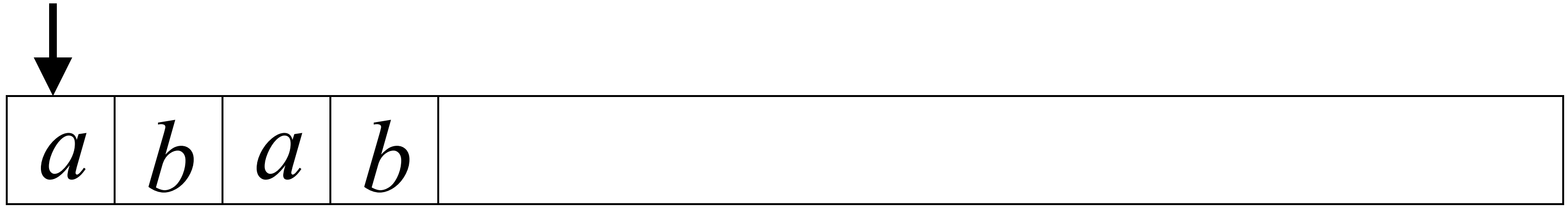

a
b


"accept"
q0
a
q1
b
q2
λ
q3
λ

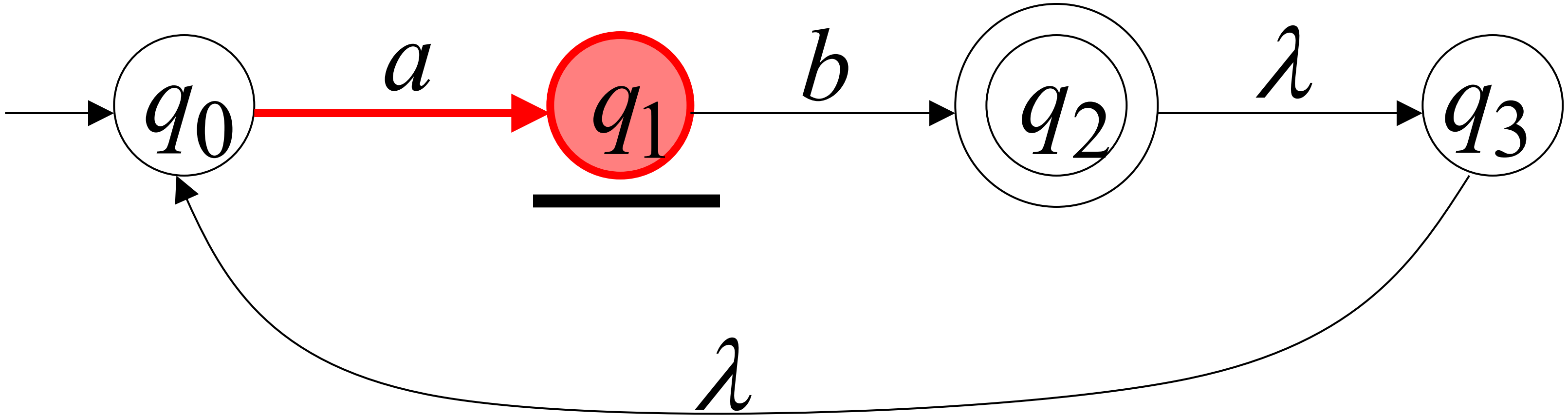
Another String

a b a b

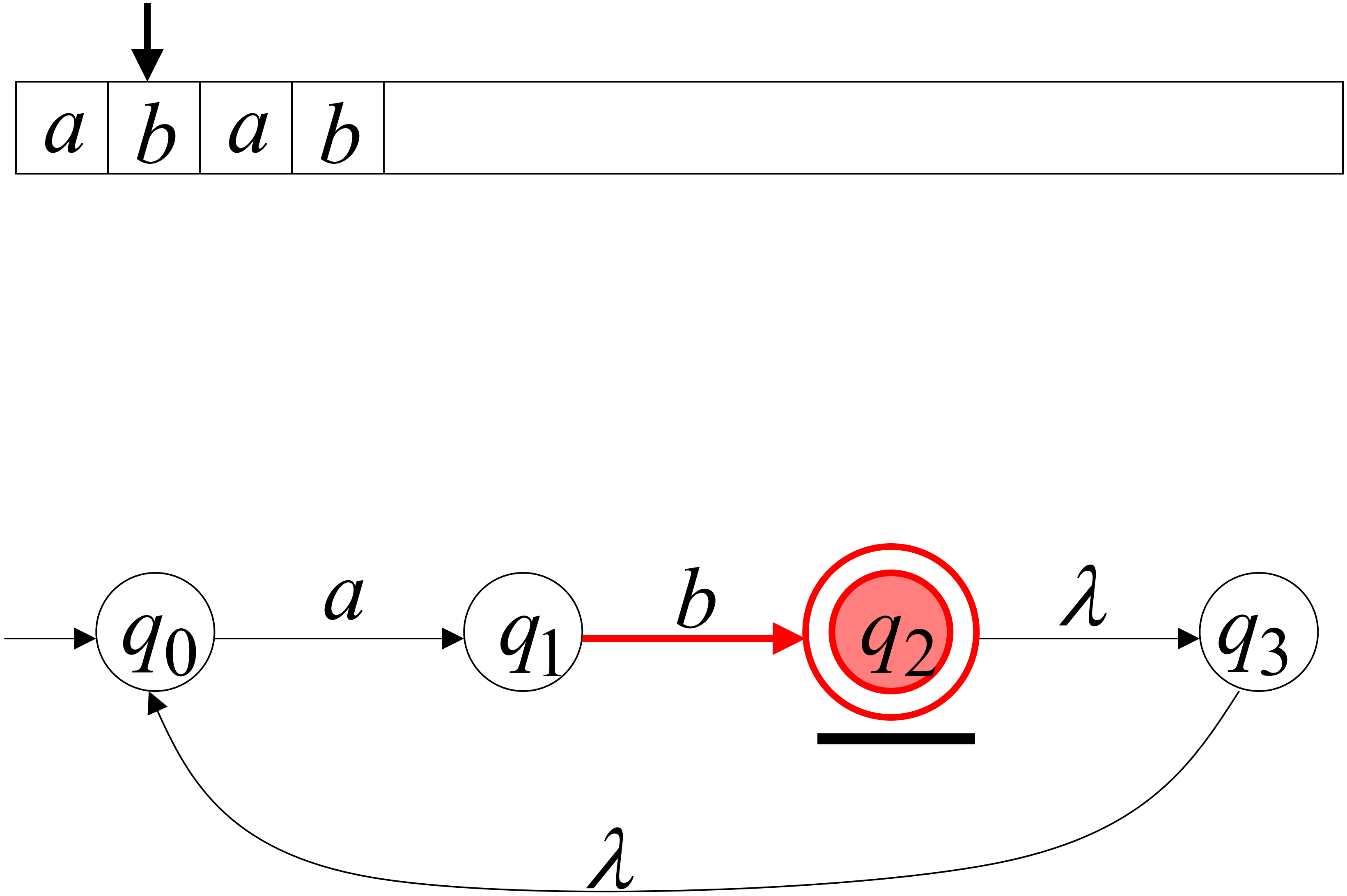




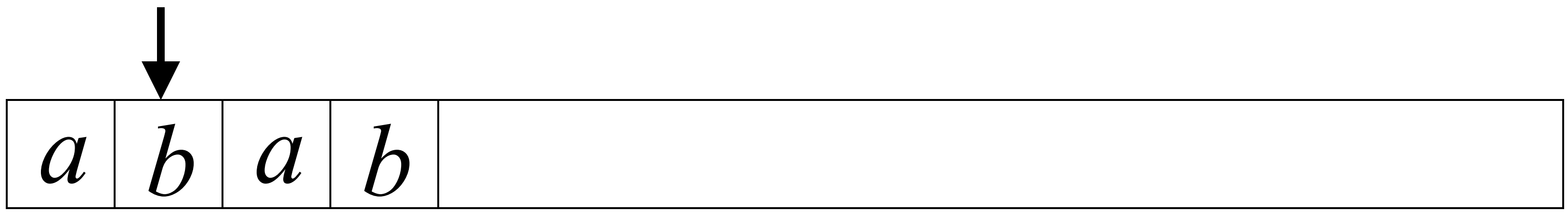
a
b
a
b

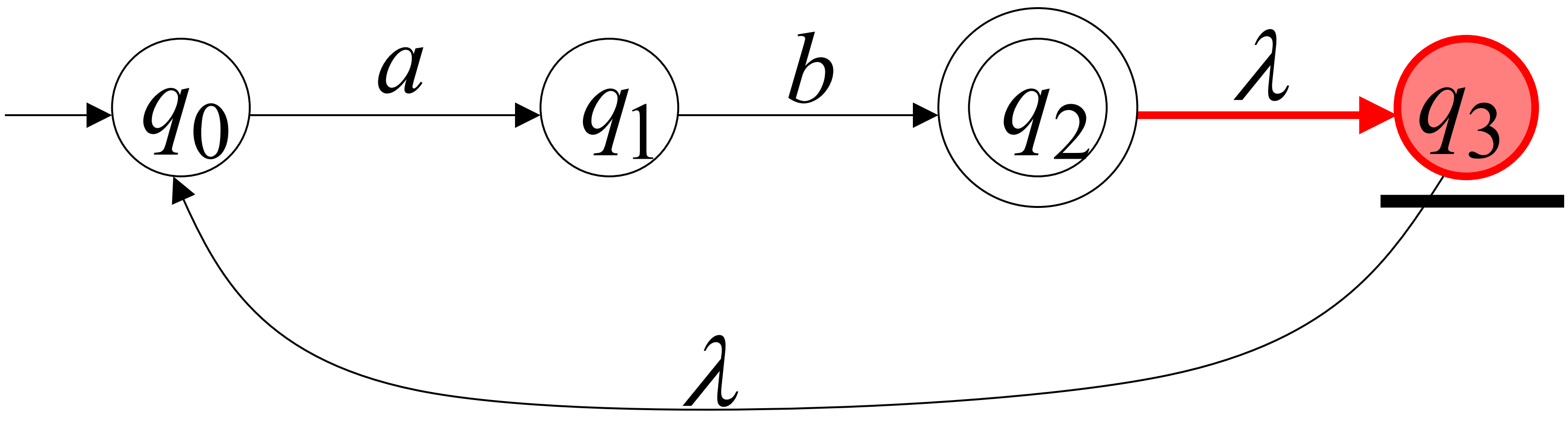


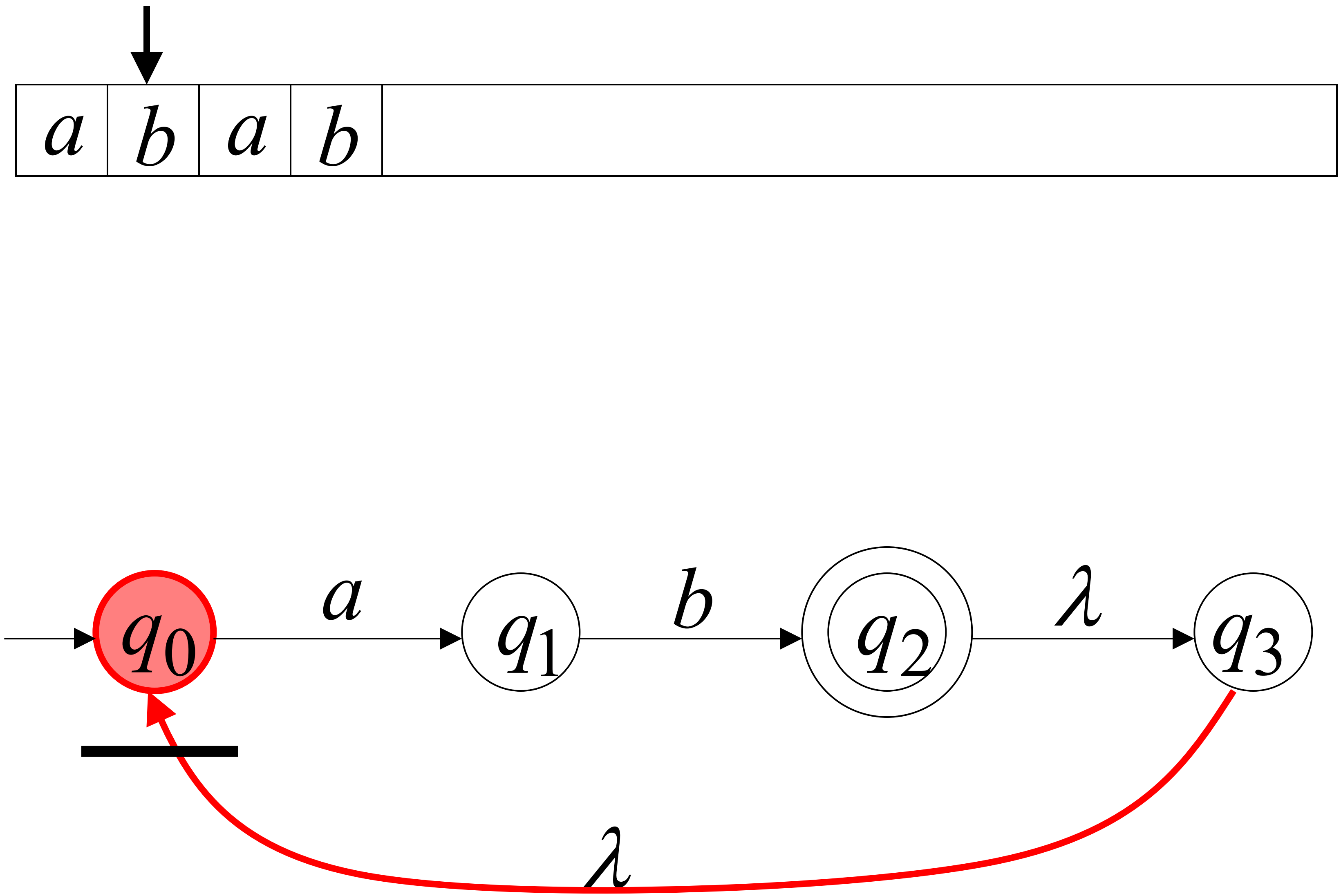
q_0
a
q_1
b
q_2
λ
q_3
λ



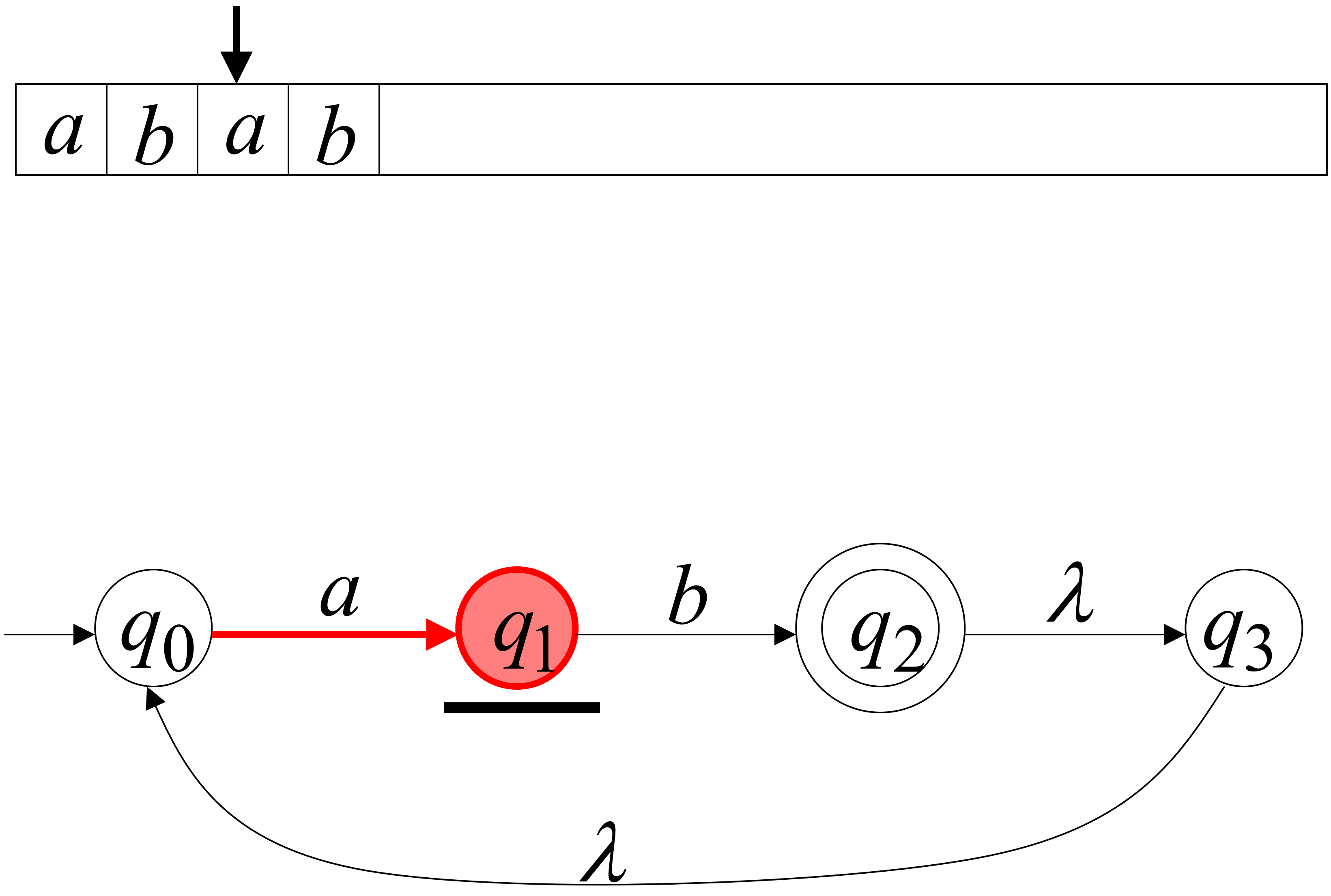
a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

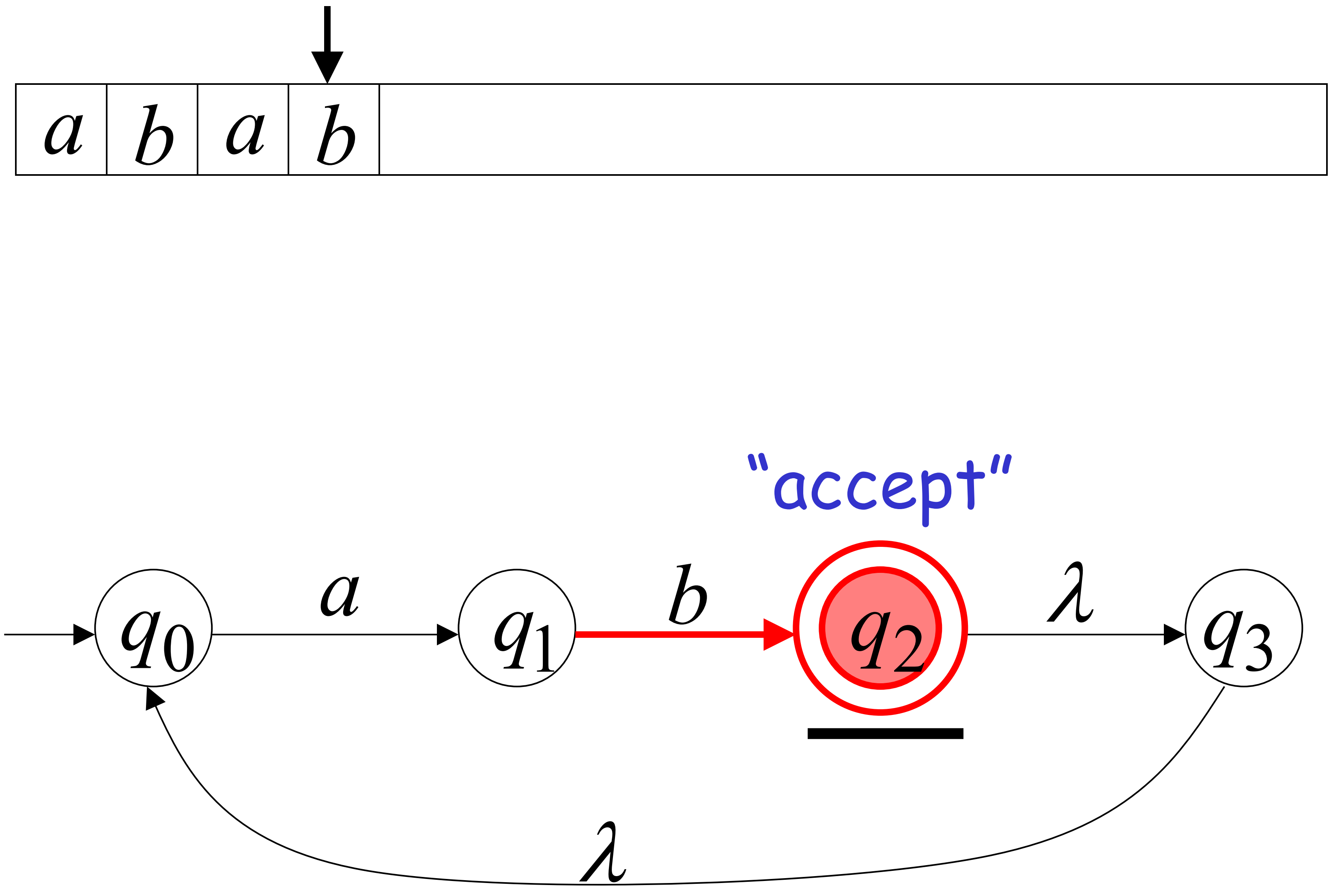

a
b
a
b


a
b
λ
λ
q_0
q_1
q_2
q_3



a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

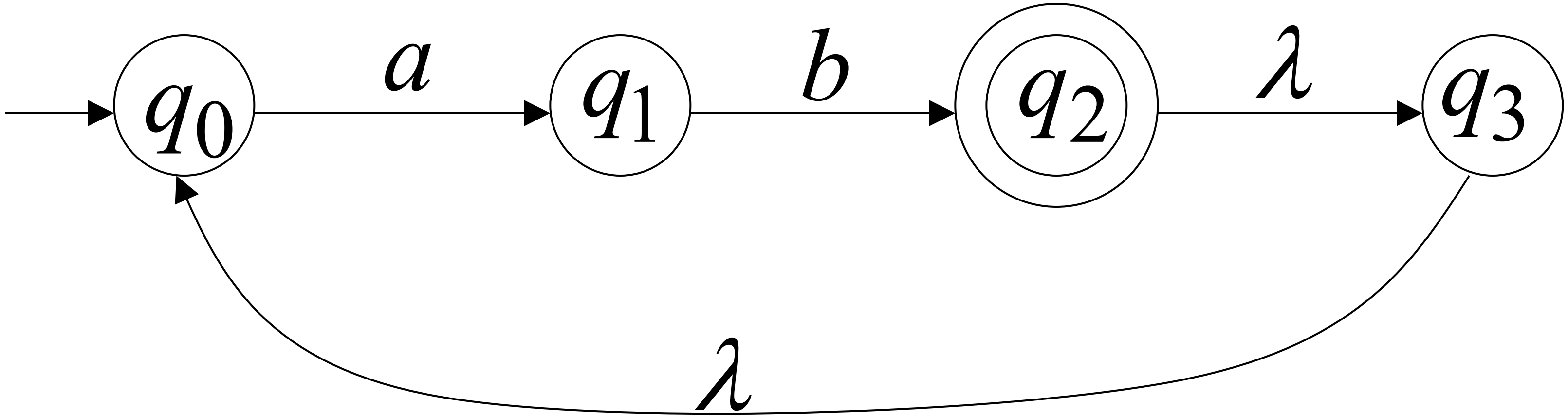

a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3


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

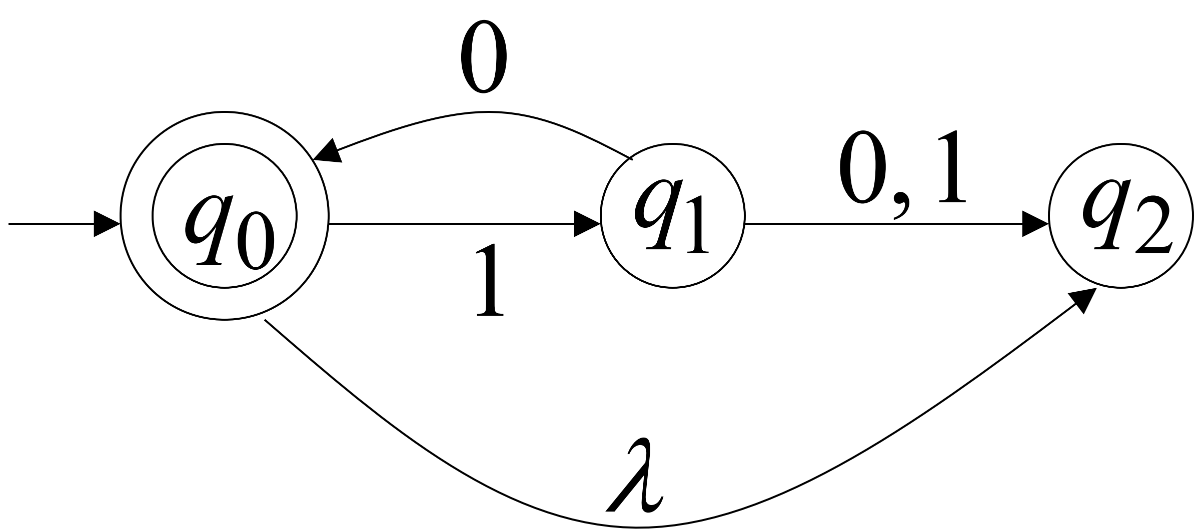
Language accepted

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

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

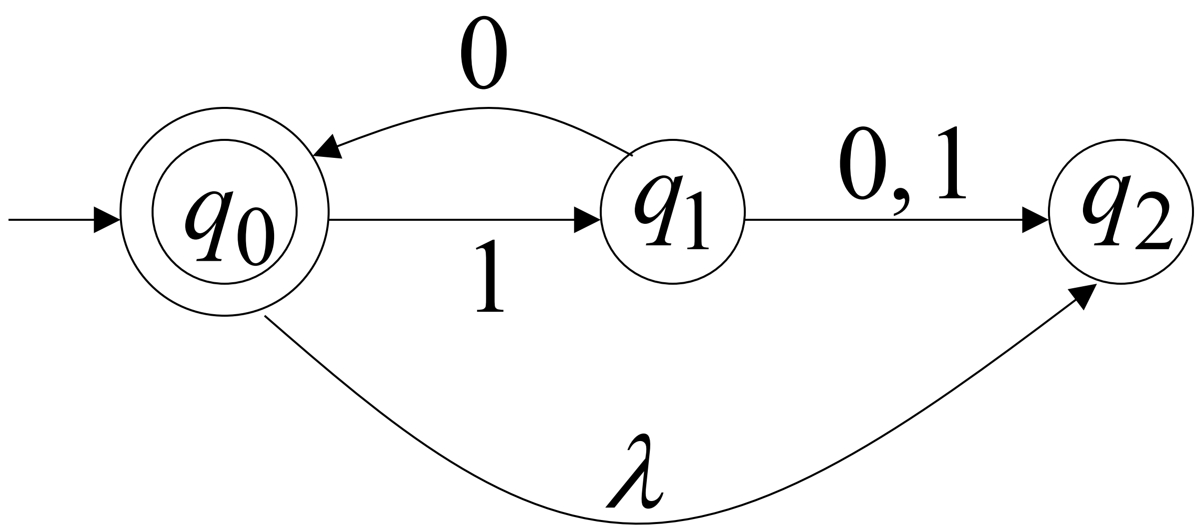


Another NFA Example



Language accepted

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



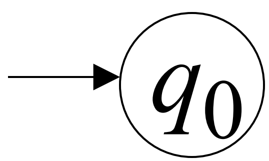
(redundant state)

Remarks:

- The λ symbol never appears on the input tape

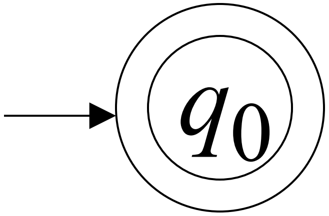
- Simple automata:

M_1



$L(M_1) = \{\}$

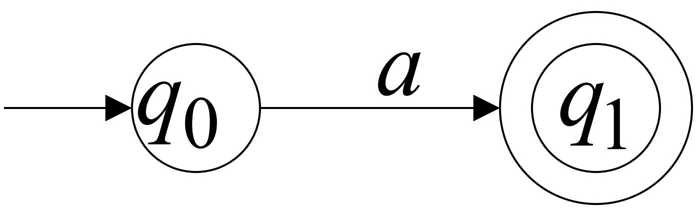
M_2



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

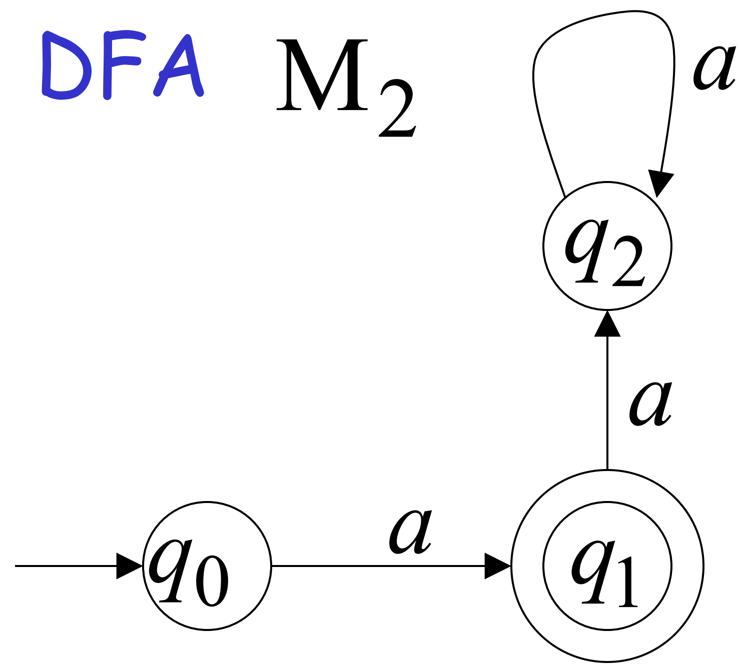
• NFAs are interesting because we can express languages easier than DFAs

NFA M_1



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

DFA M_2



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

Formal Definition of NFAs

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

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

Σ : Input aplhabet, i.e. $\{a, b\}$ $\qquad \lambda \notin \Sigma$

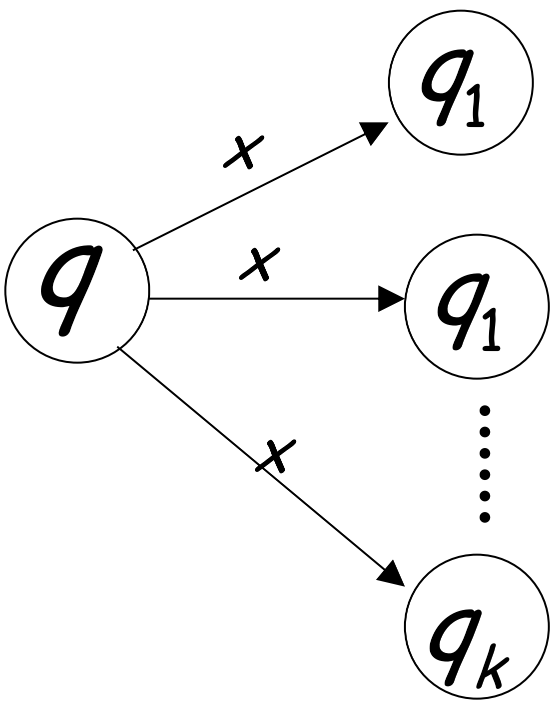
δ : Transition function

q_0 : Initial state

F : Accepting states

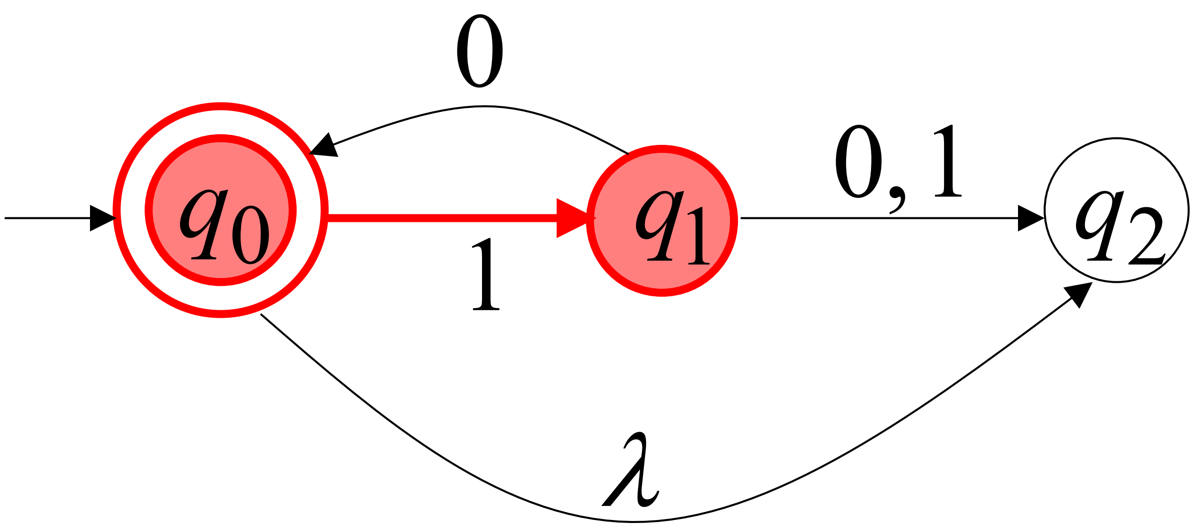
Transition Function δ

$$\delta(q, x) = \{q_1, q_2, \ldots, q_k\}$$

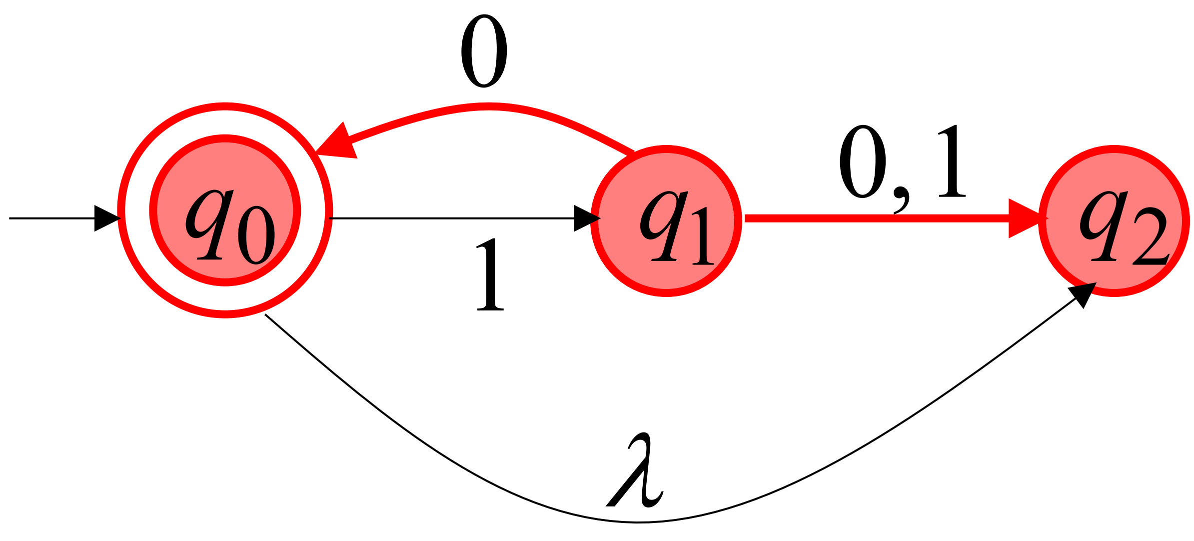


resulting states with following one transition with symbol x

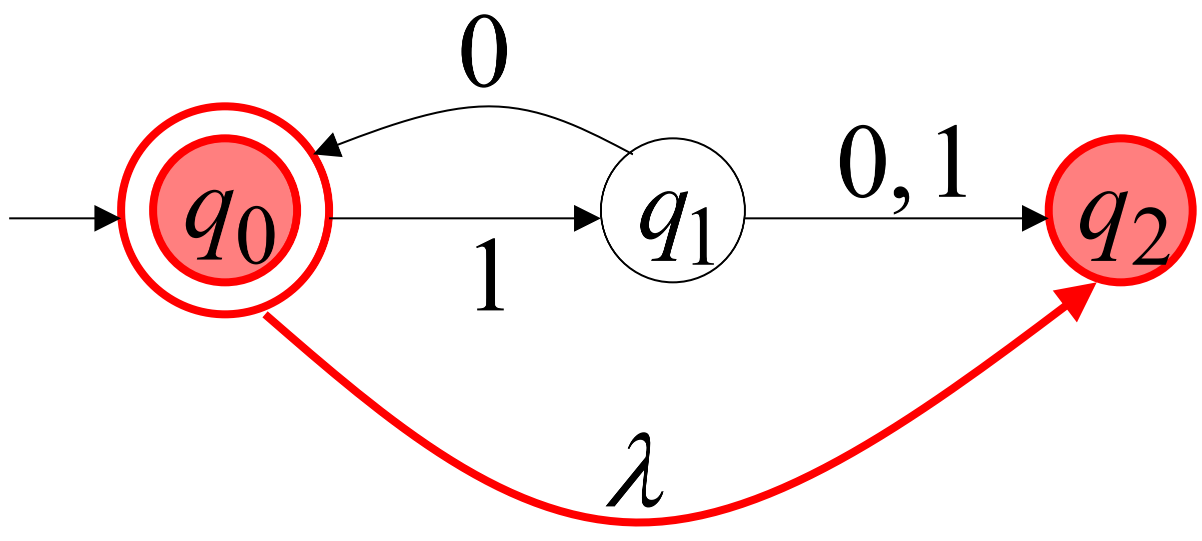
$$\delta(q_0, 1) = \{q_1\}$$



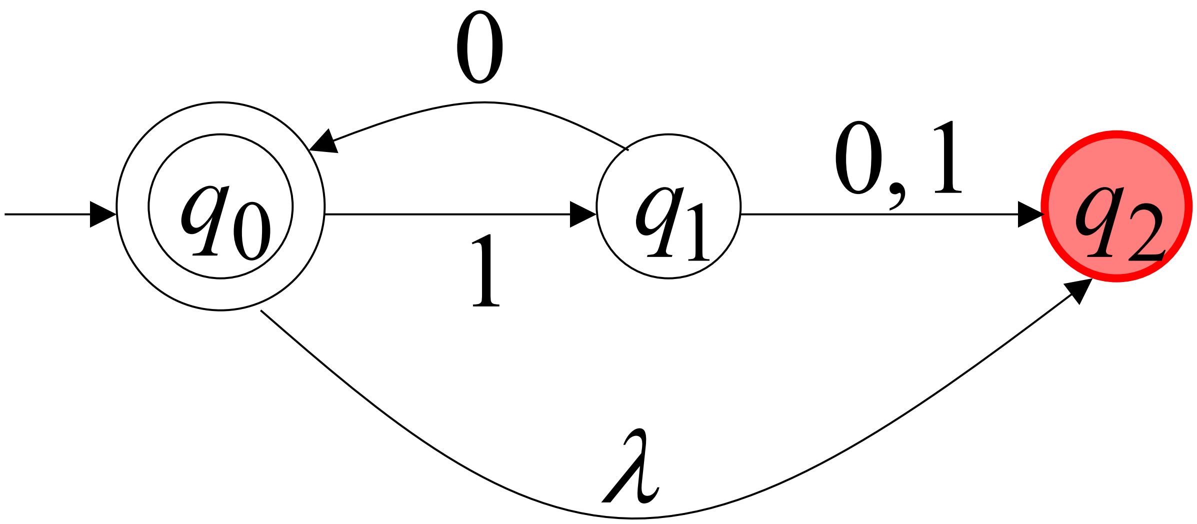
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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



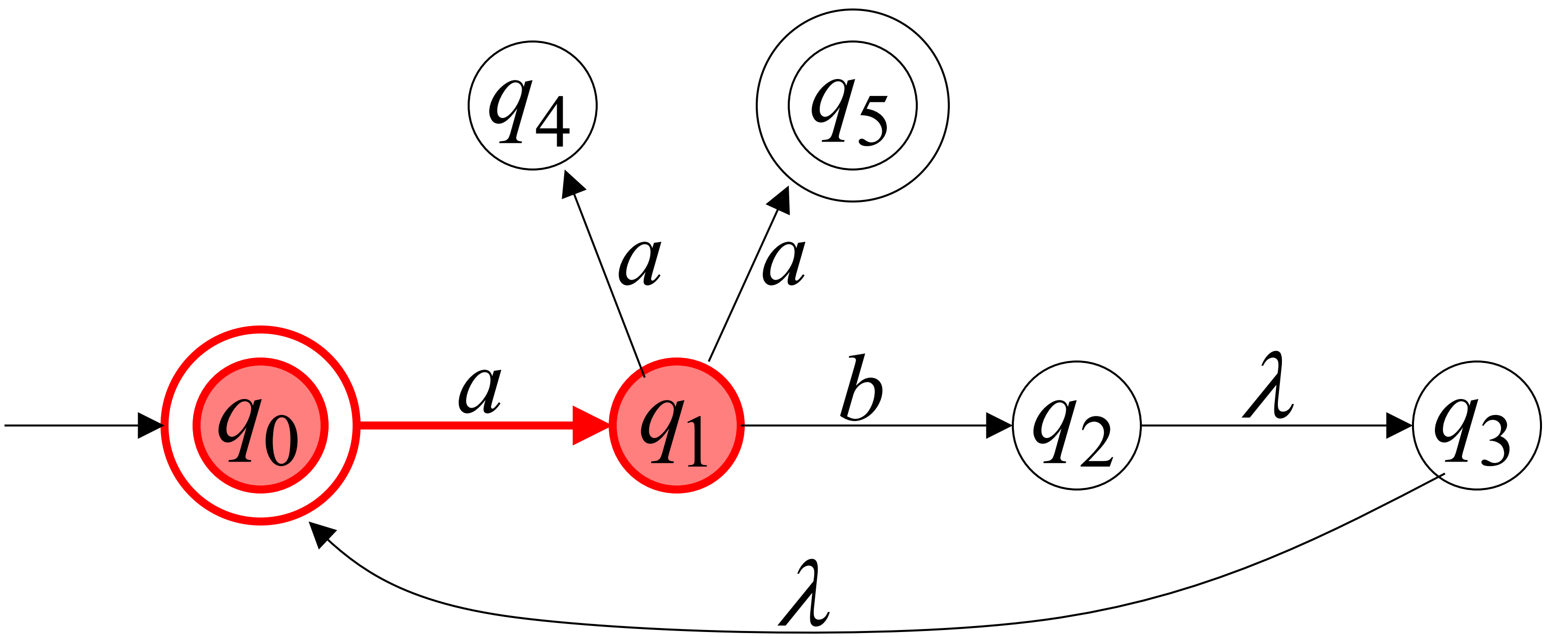
$$\delta(q_2, 1) = \varnothing$$



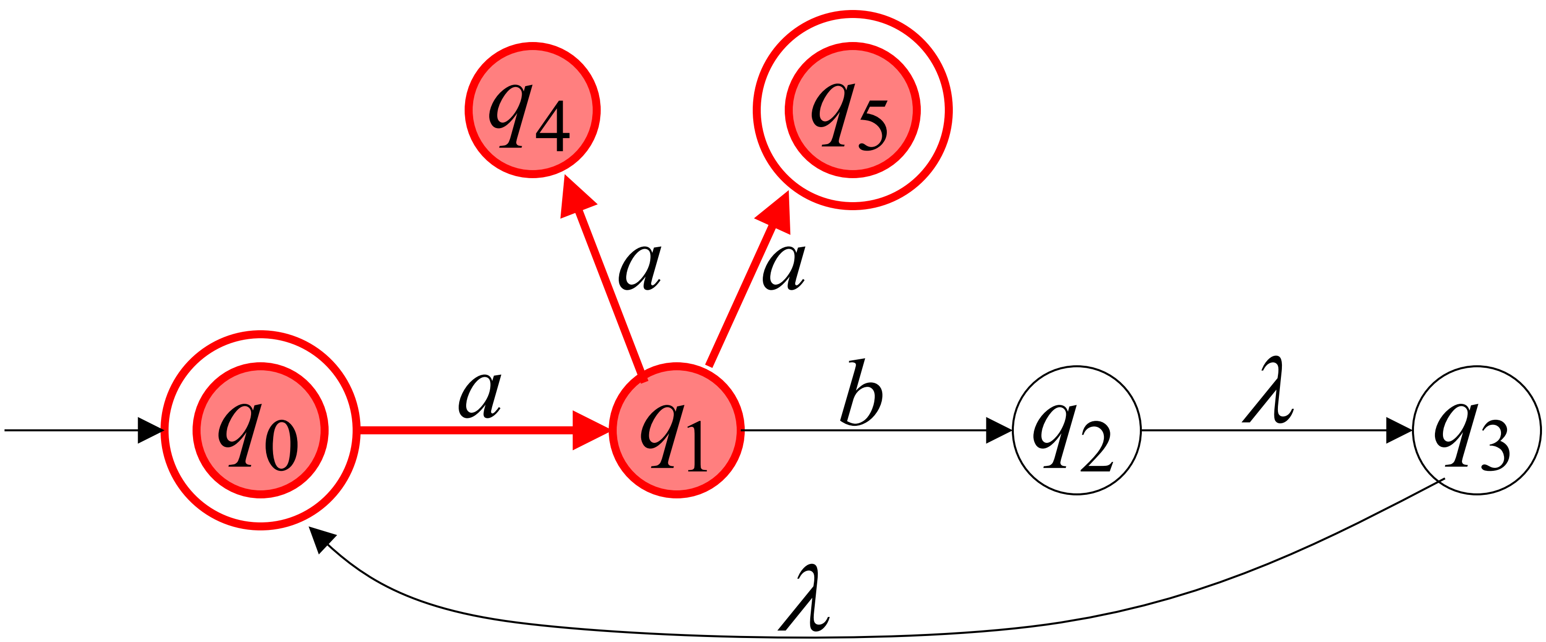
Extended Transition Function δ^*

Same with δ but applied on strings

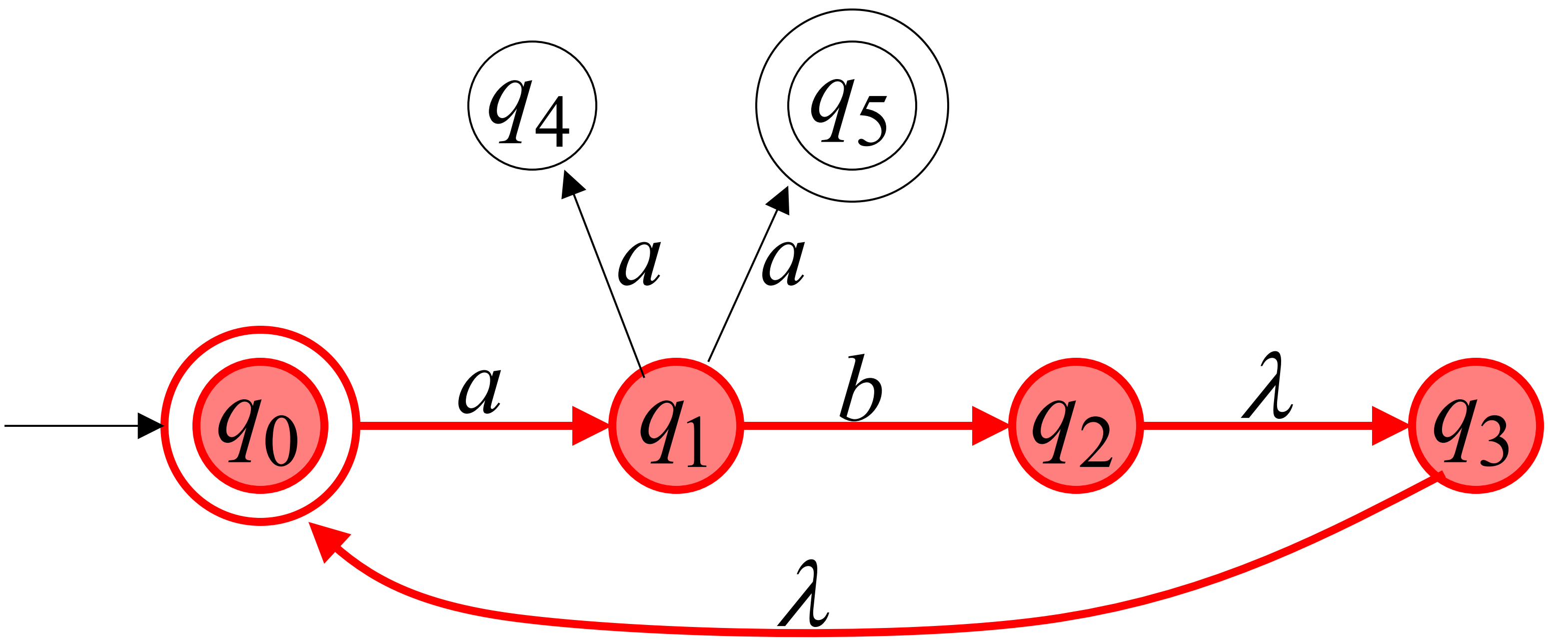
$$\delta^*(q_0, a) = \{q_1\}$$



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



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



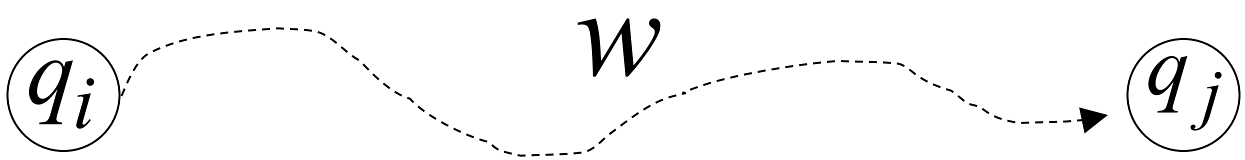
Special case:

for any state q

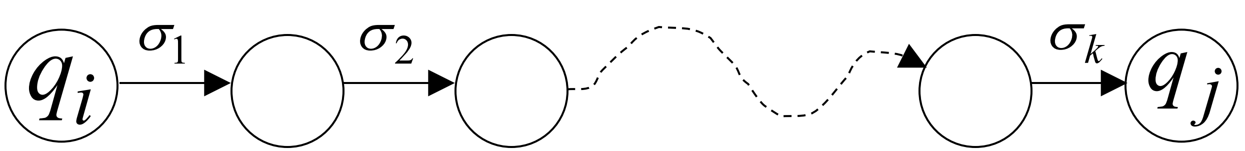
$$q \in \delta^*(q, \lambda)$$

In general

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

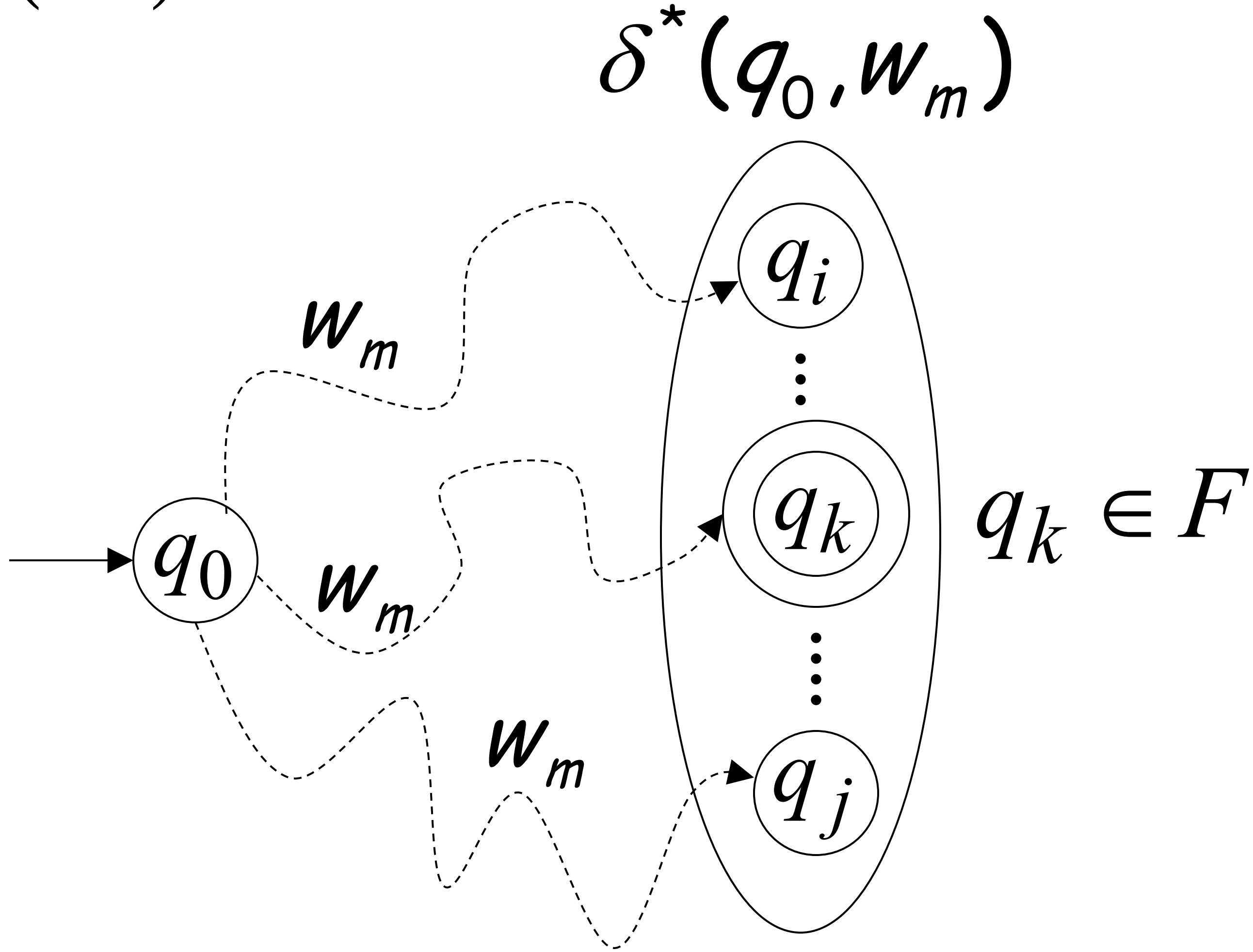
The language accepted by M is:

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

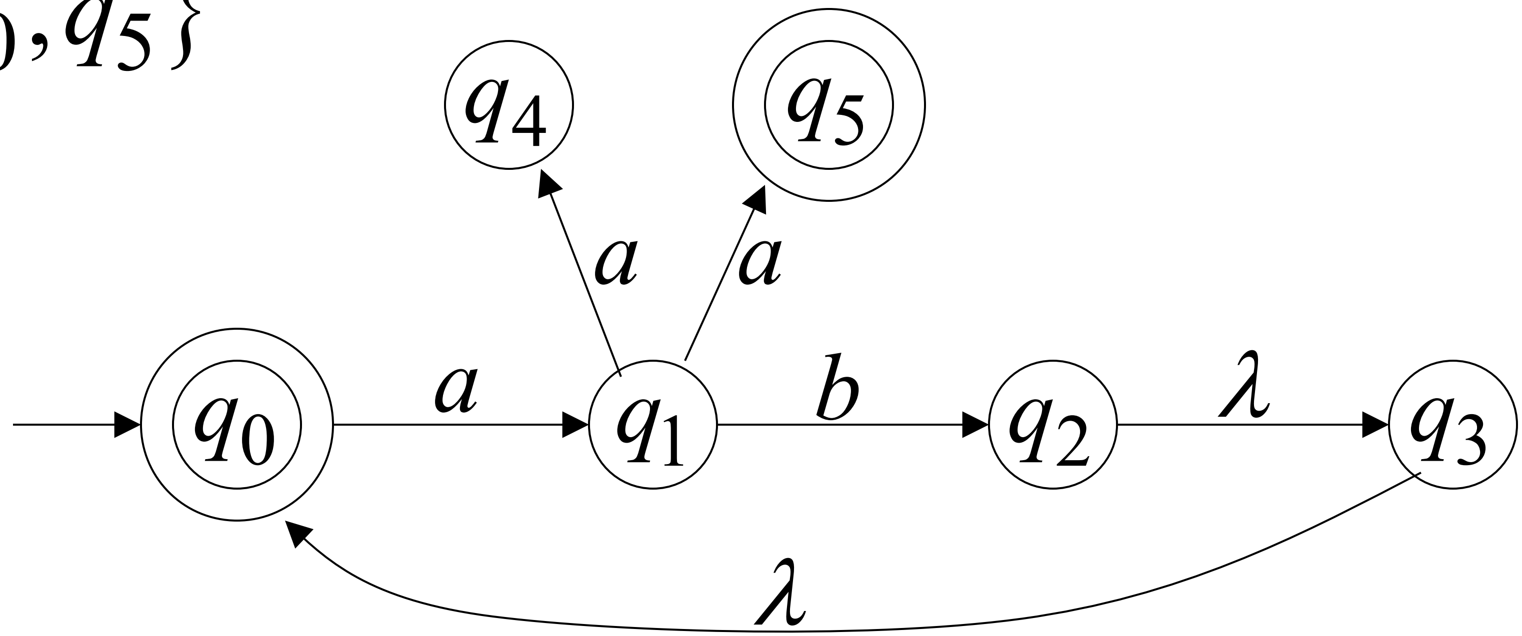
where $\delta^*(q_0, w_m) = \{q_i, \ldots, q_k, \ldots, q_j\}$

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

$w_m \in L(M)$



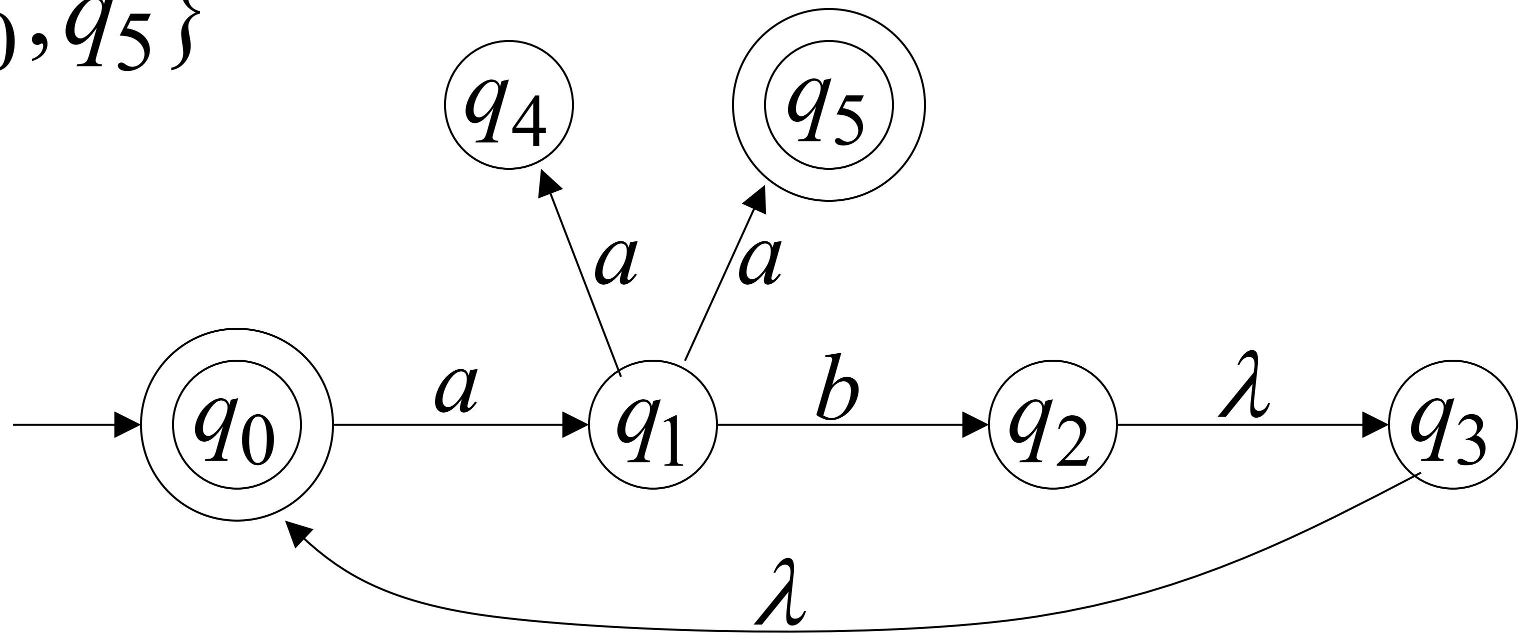
$F = \{q_0, q_5\}$



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

$\in F$

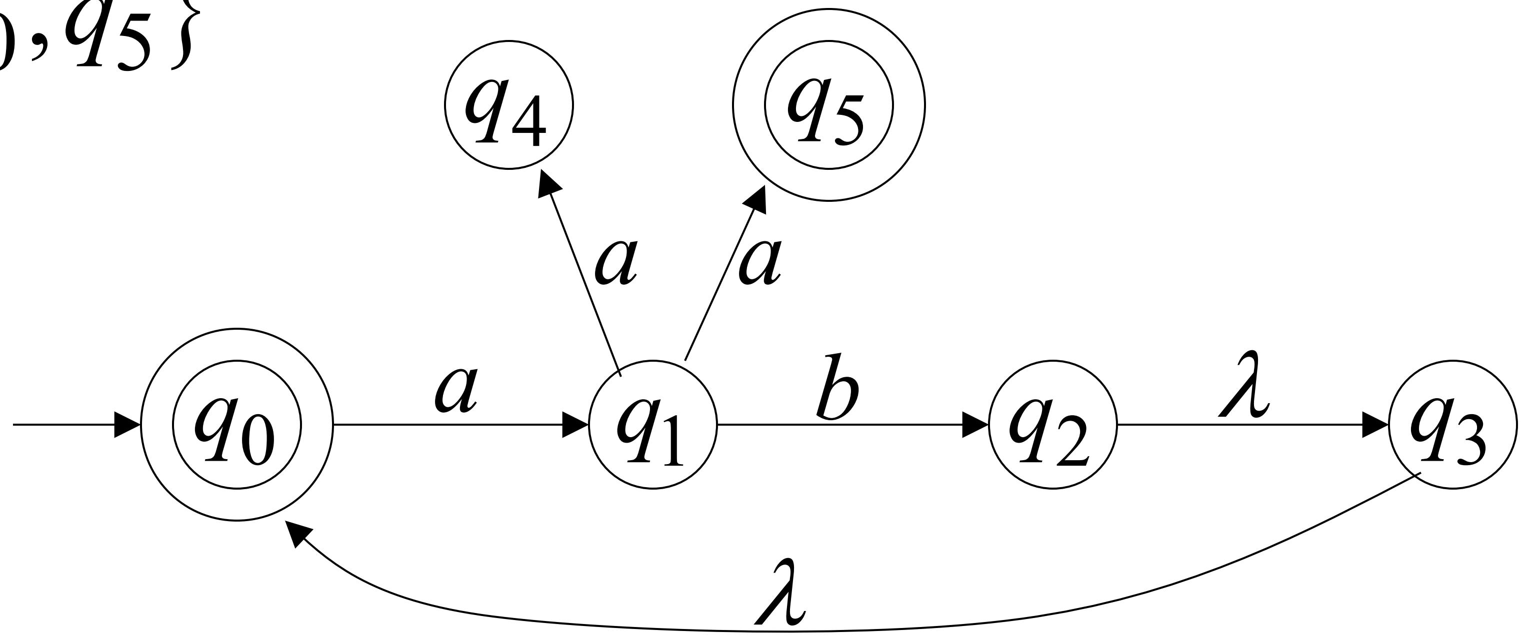
$$F = \{q_0, q_5\}$$



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

$$\in F$$

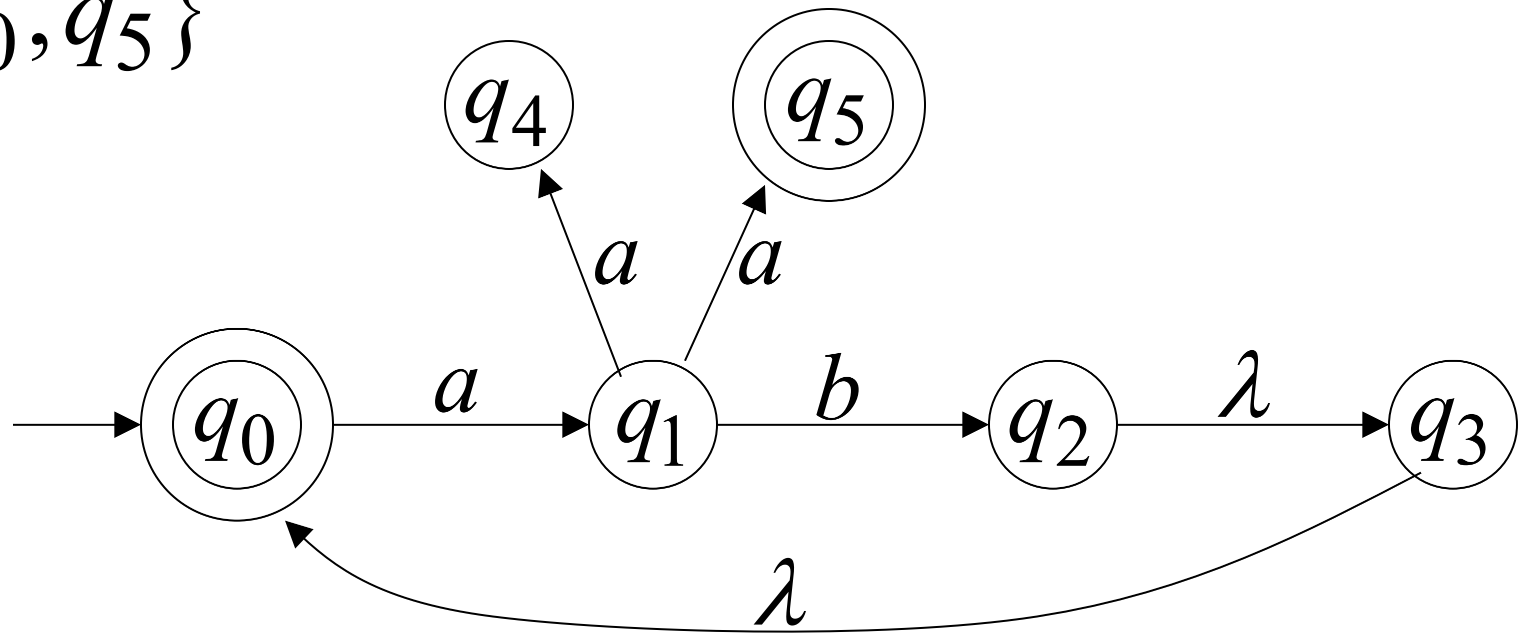
$$F = \{q_0, q_5\}$$



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

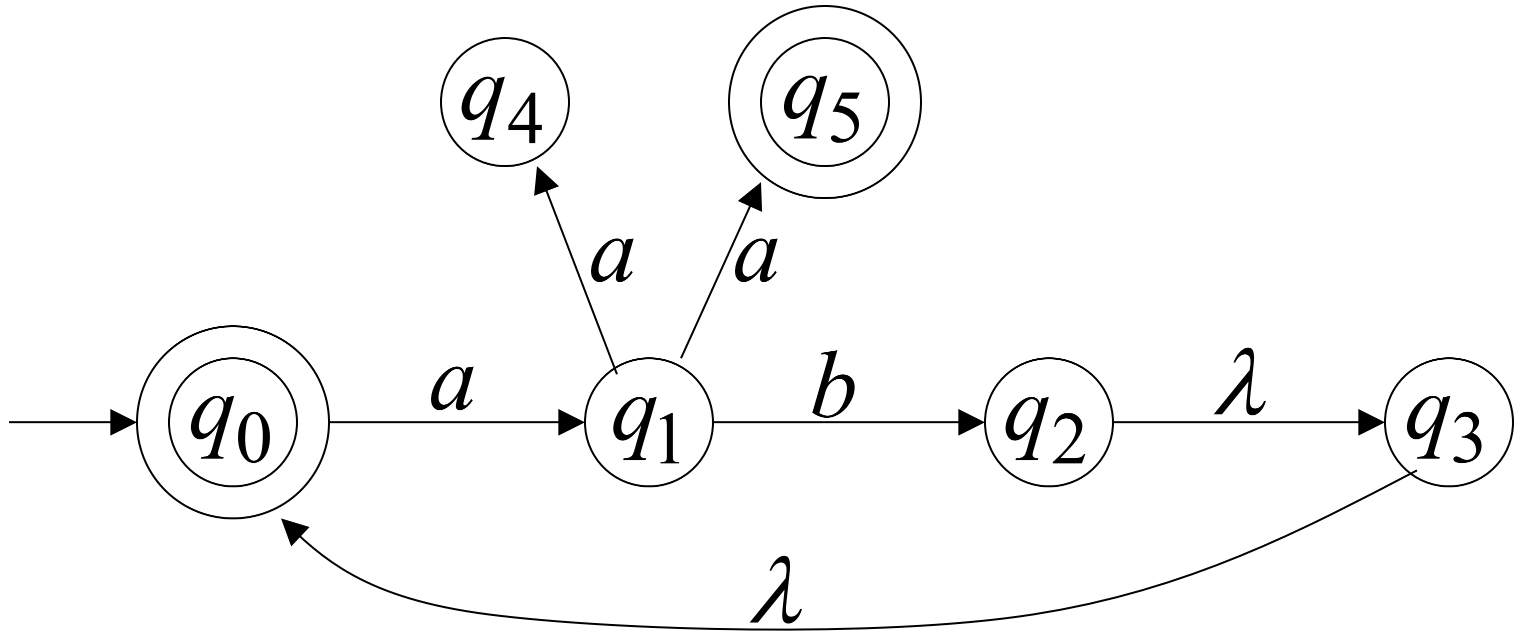
$$\in F$$

$F = \{q_0, q_5\}$



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

$\notin F$



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

NFAs accept the Regular Languages

Equivalence of Machines

Definition:

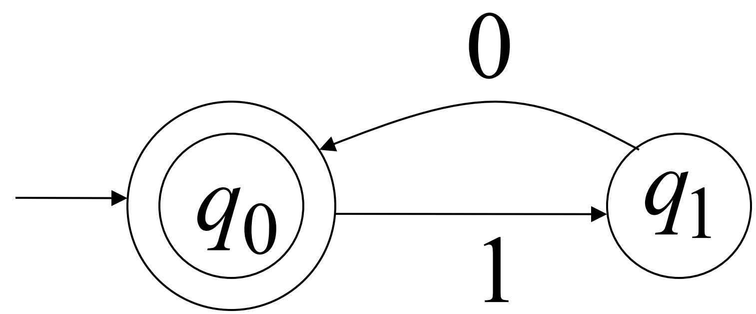
Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

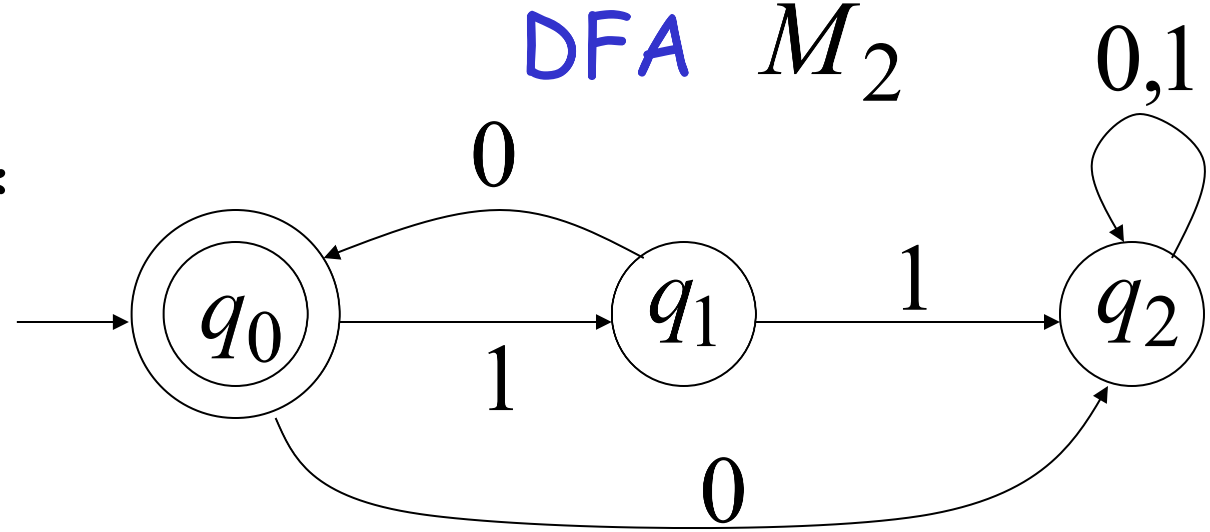
NFA M_1

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



DFA M_2

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



Theorem:

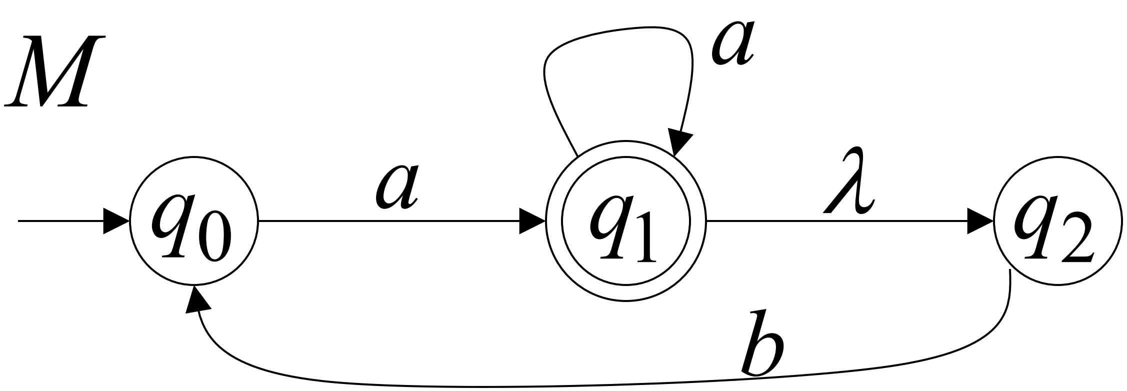
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Languages
accepted
by DFAs

NFAs and DFAs have the same computation power, accept the same set of languages

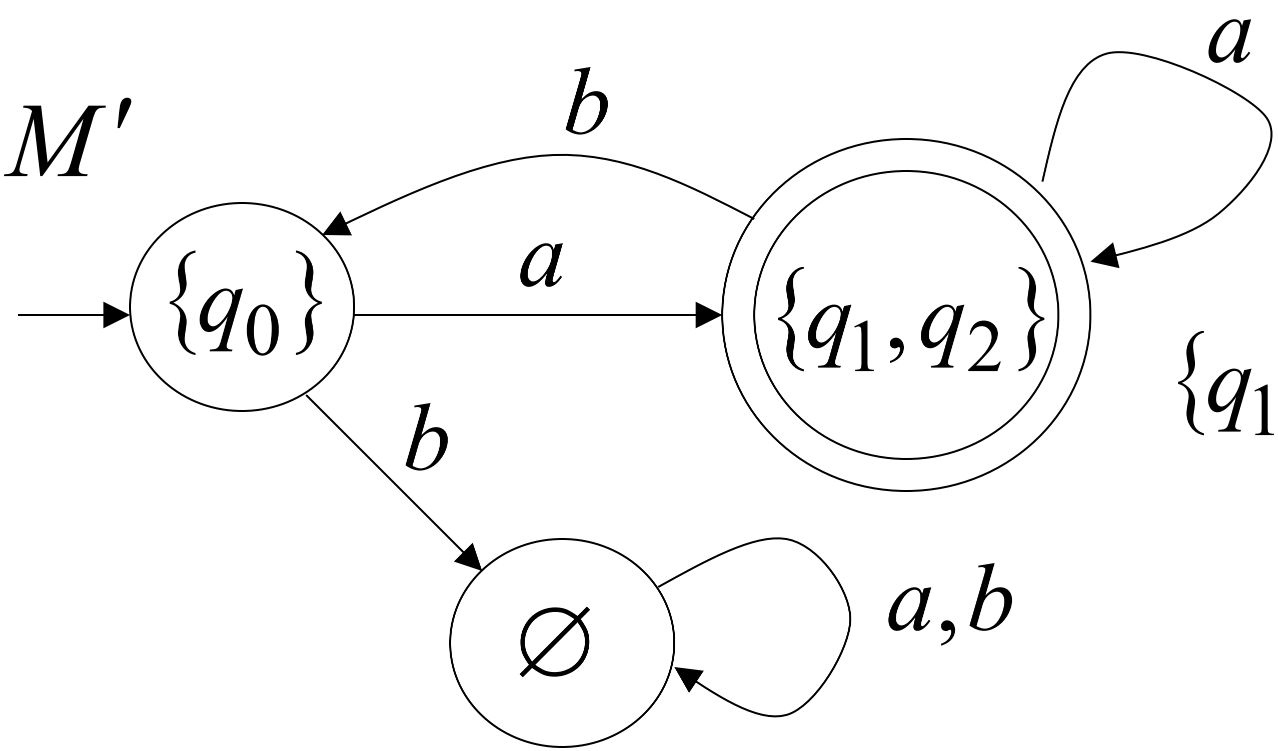
Conversion NFA to DFA

NFA M



$q_1 \in F$

DFA M'



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

General Conversion Procedure

Input: an NFA M

Output: an equivalent DFA M'

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

The NFA has states $q_0, q_1, q_2, \ldots$

The DFA has states from the power set

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

Conversion Procedure Steps

step

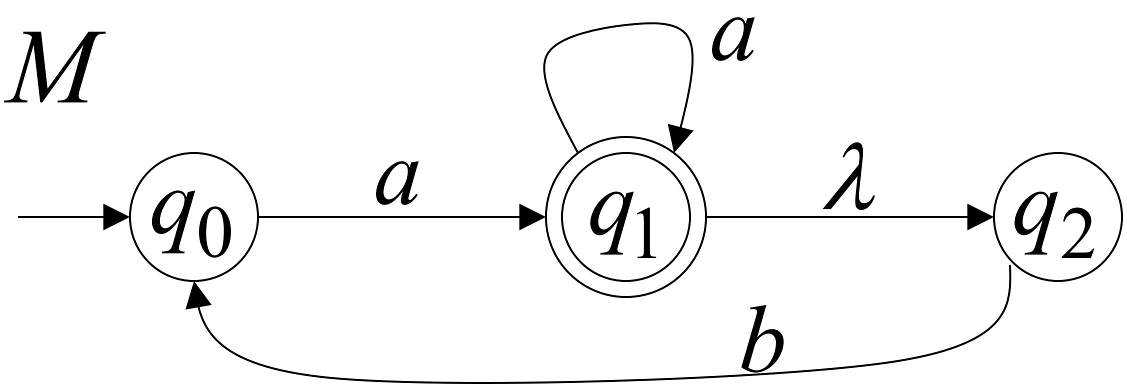
1. Initial state of NFA: q_0

$\Downarrow$

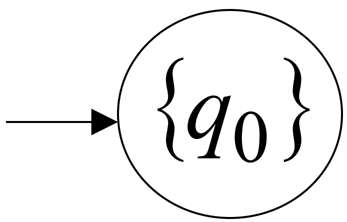
Initial state of DFA: $\{q_0\}$

Example

NFA M



DFA M'



step

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

compute in the NFA

$$\left.\begin{aligned} &\delta^*(q_i, a) \\ \cup\, &\delta^*(q_j, a) \\ &\ldots \\ \cup\, &\delta^*(q_m, a) \end{aligned}\right\} = \{q'_k, q'_l, \ldots, q'_n\}$$

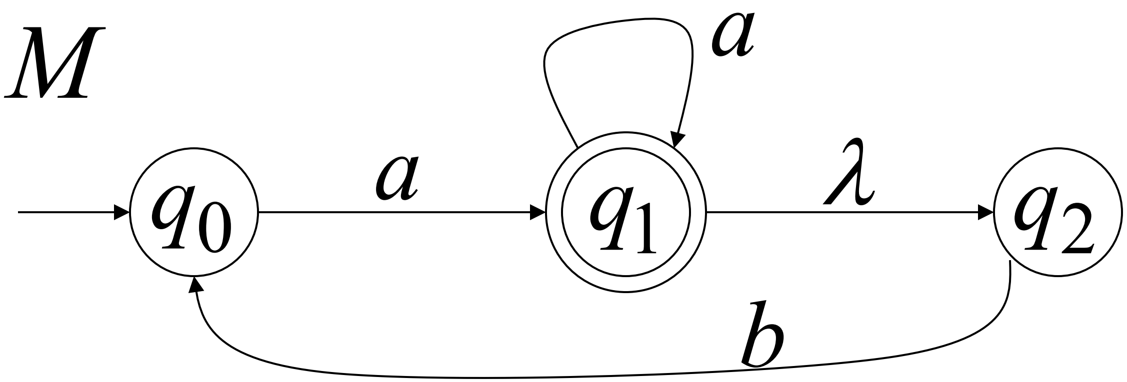
Union

add transition to DFA

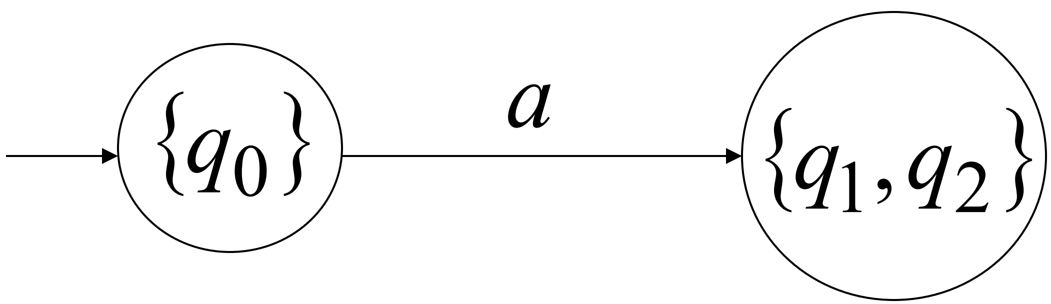
$$\delta(\{q_i, q_j, \ldots, q_m\},\ a) = \{q'_k, q'_l, \ldots, q'_n\}$$

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

NFA M



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

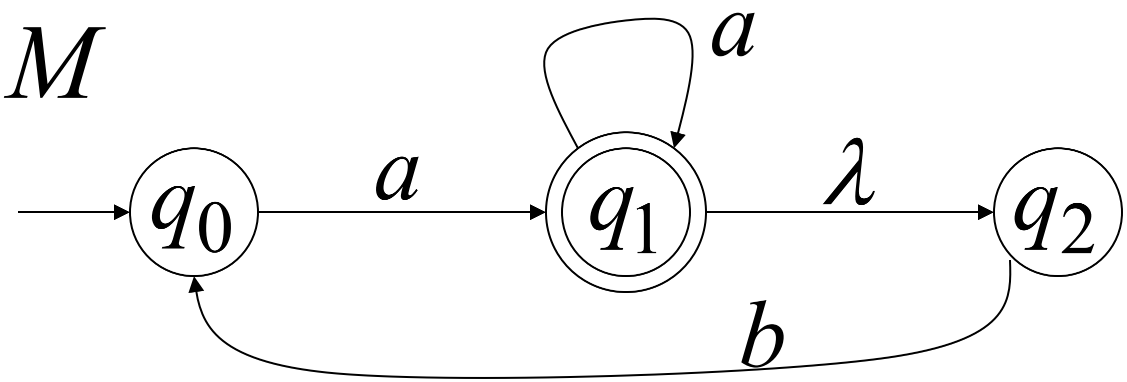


step

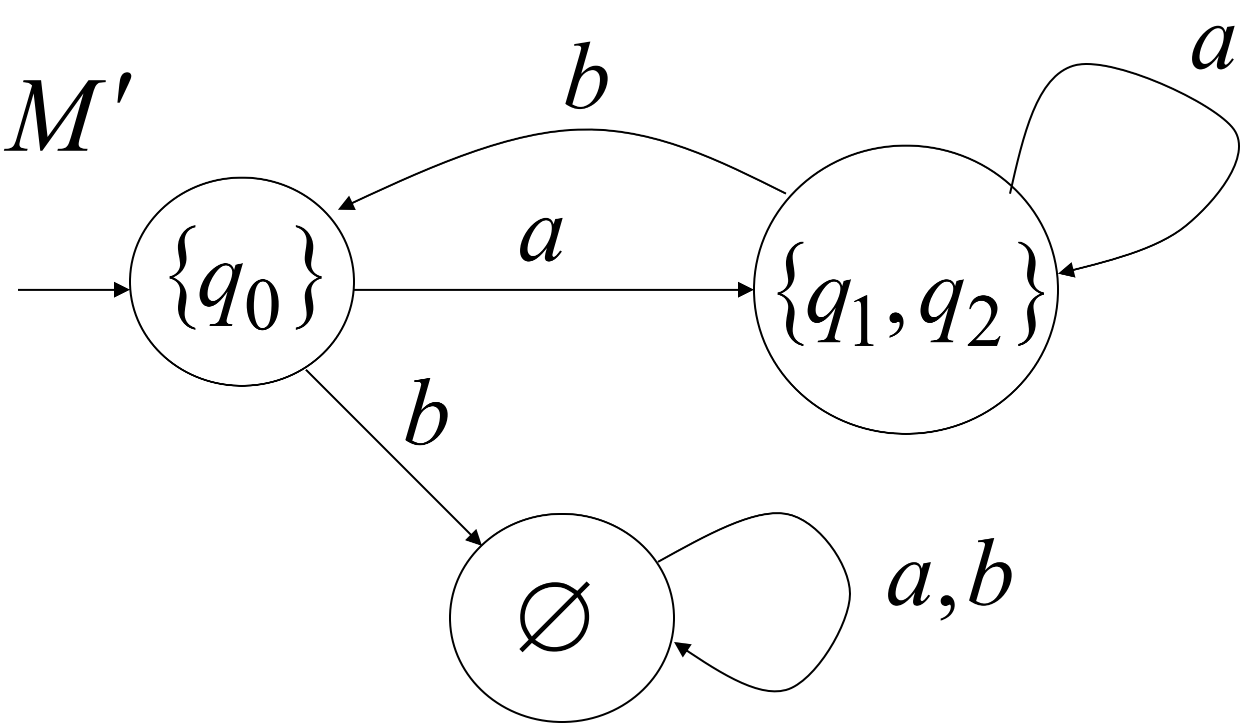
3. Repeat Step 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA

Example

NFA M



DFA M'



step

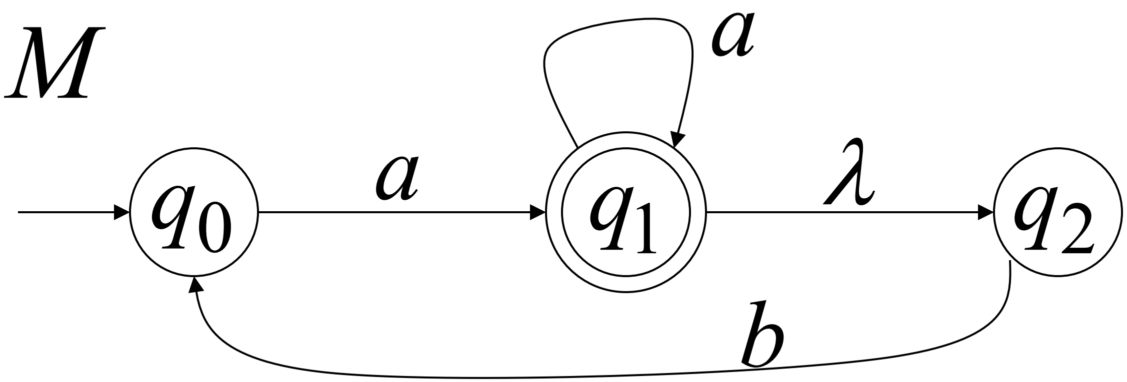
4. For any DFA state $\{q_i, q_j, \ldots, q_m\}$

if some q_j is accepting state in NFA

Then, $\{q_i, q_j, \ldots, q_m\}$
is accepting state in DFA

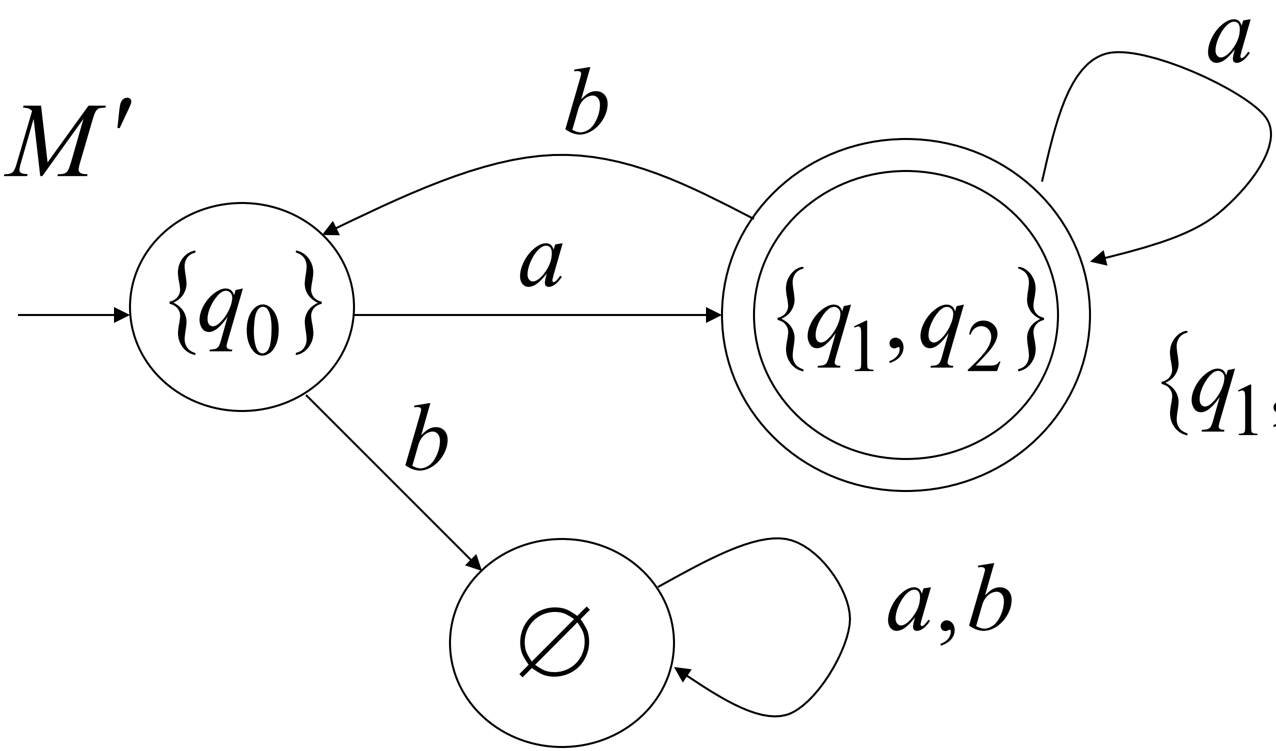
Example

NFA M



$q_1 \in F$

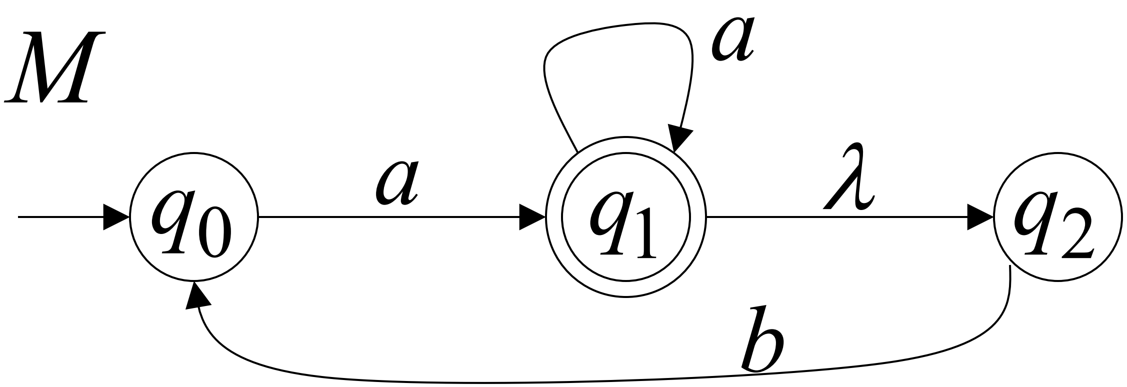
DFA M'



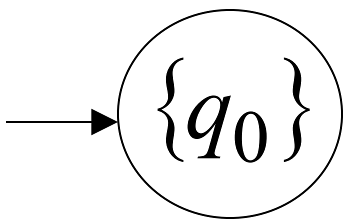
$\{q_1, q_2\} \in F'$

Conversion NFA to DFA

NFA M

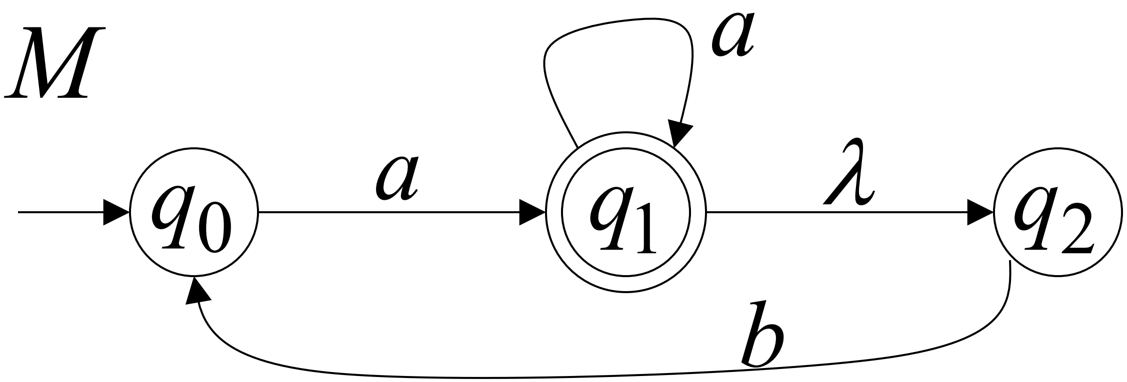


DFA M'

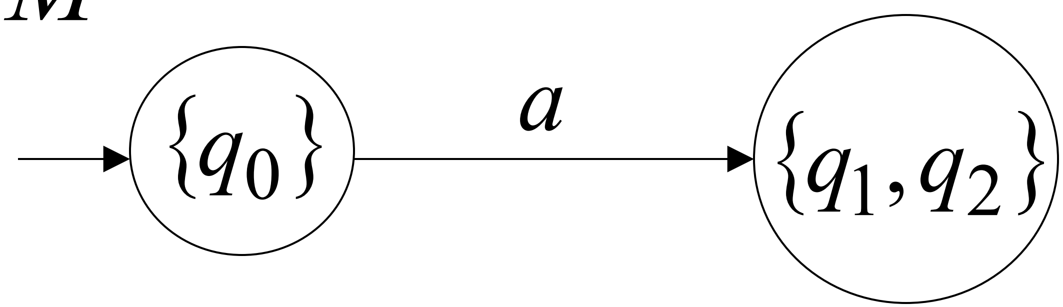


$$\delta^*(q_0, a) = \{q_1, q_2\}$$

NFA M

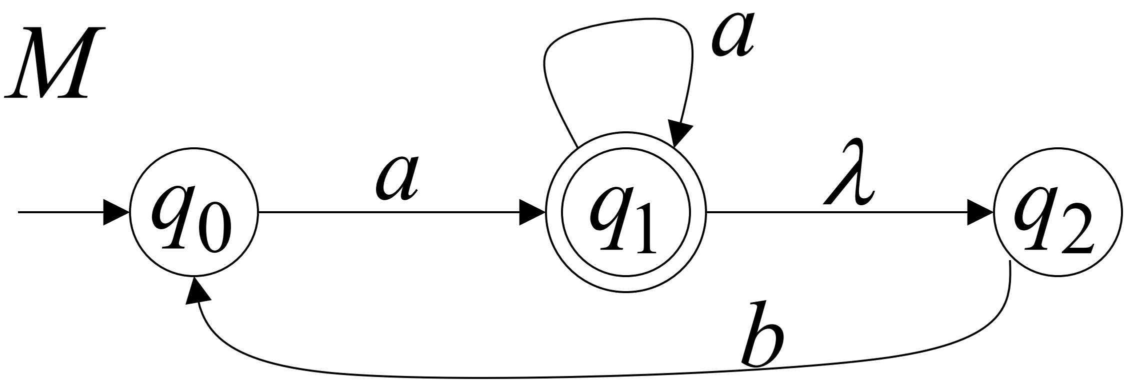


DFA M'

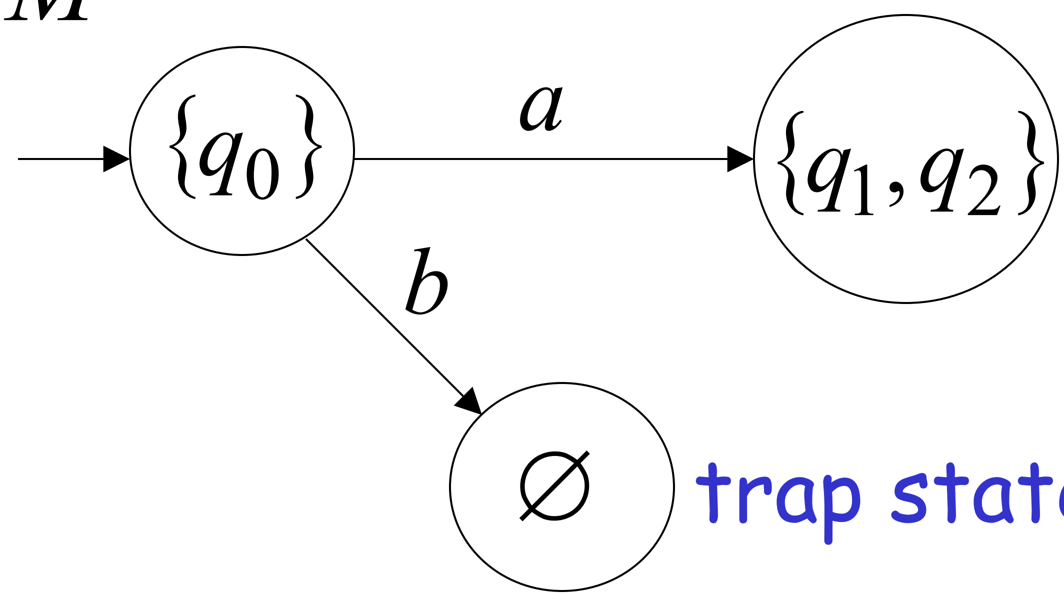


$$\delta^*(q_0, b) = \varnothing \quad \text{empty set}$$

NFA M

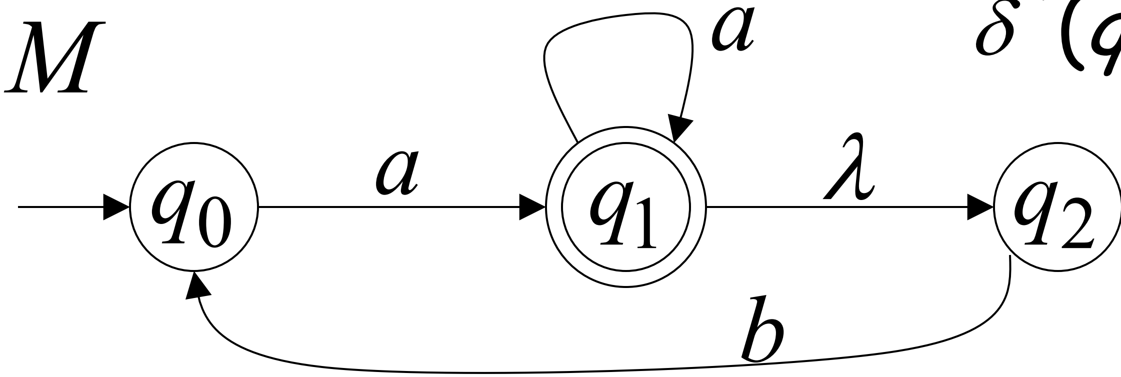


DFA M'



trap state

NFA M

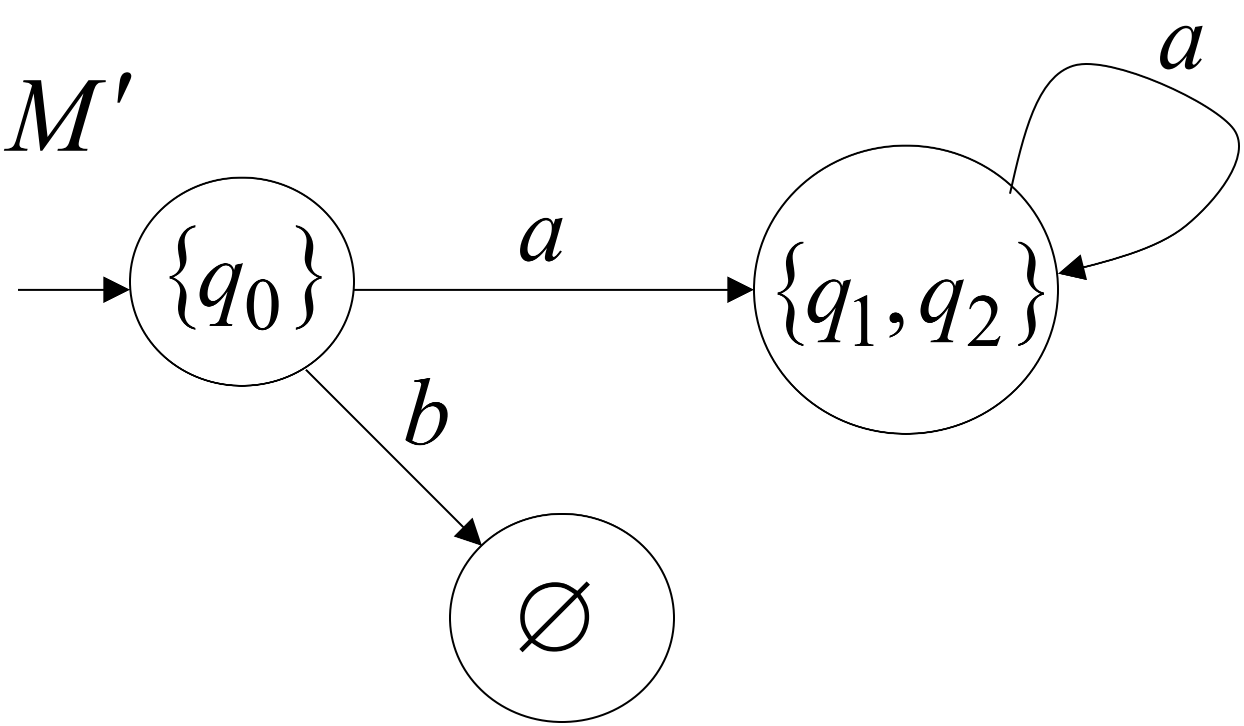


$$\delta^*(q_1,a)=\{q_1,q_2\}$$
$$\delta^*(q_2,a)=\varnothing$$

union

$$\{q_1,q_2\}$$

DFA M'

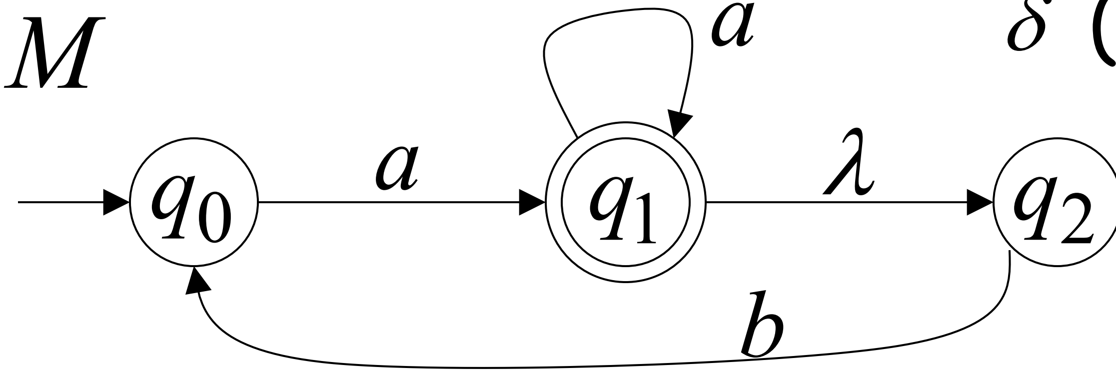


$$\delta^*(q_1, b) = \{q_0\}$$
$$\delta^*(q_2, b) = \{q_0\}$$

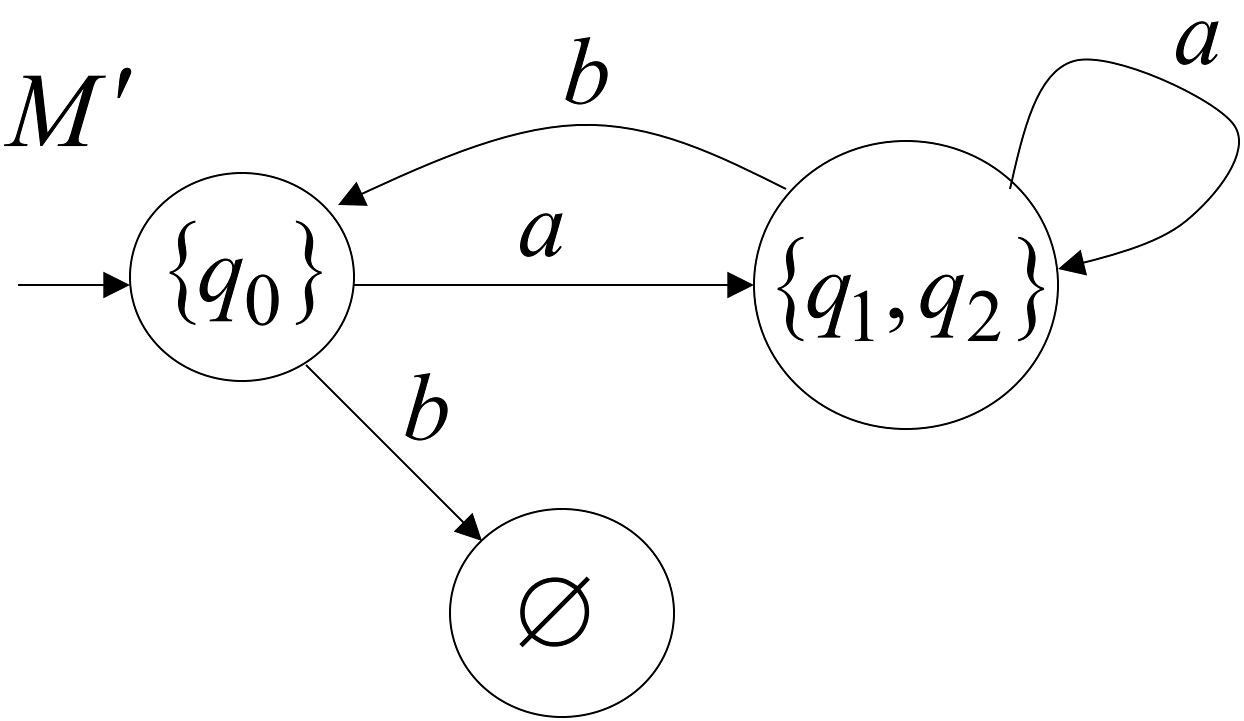
union

$$\{q_0\}$$

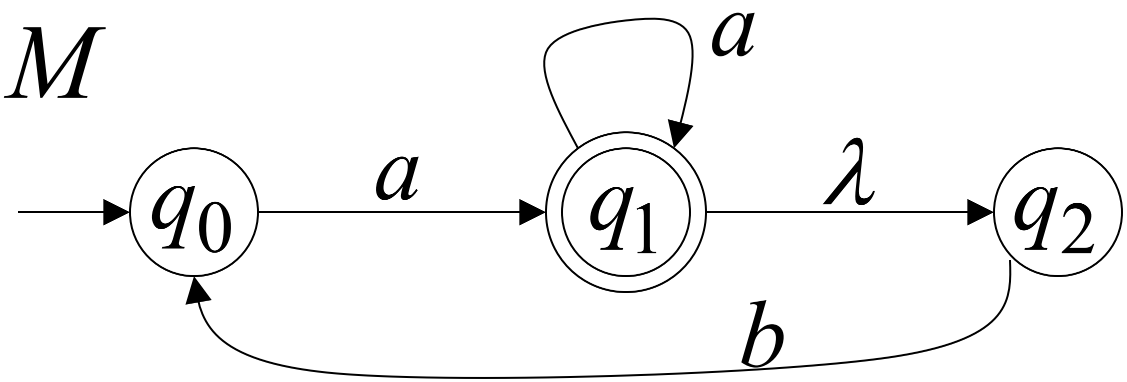
NFA M



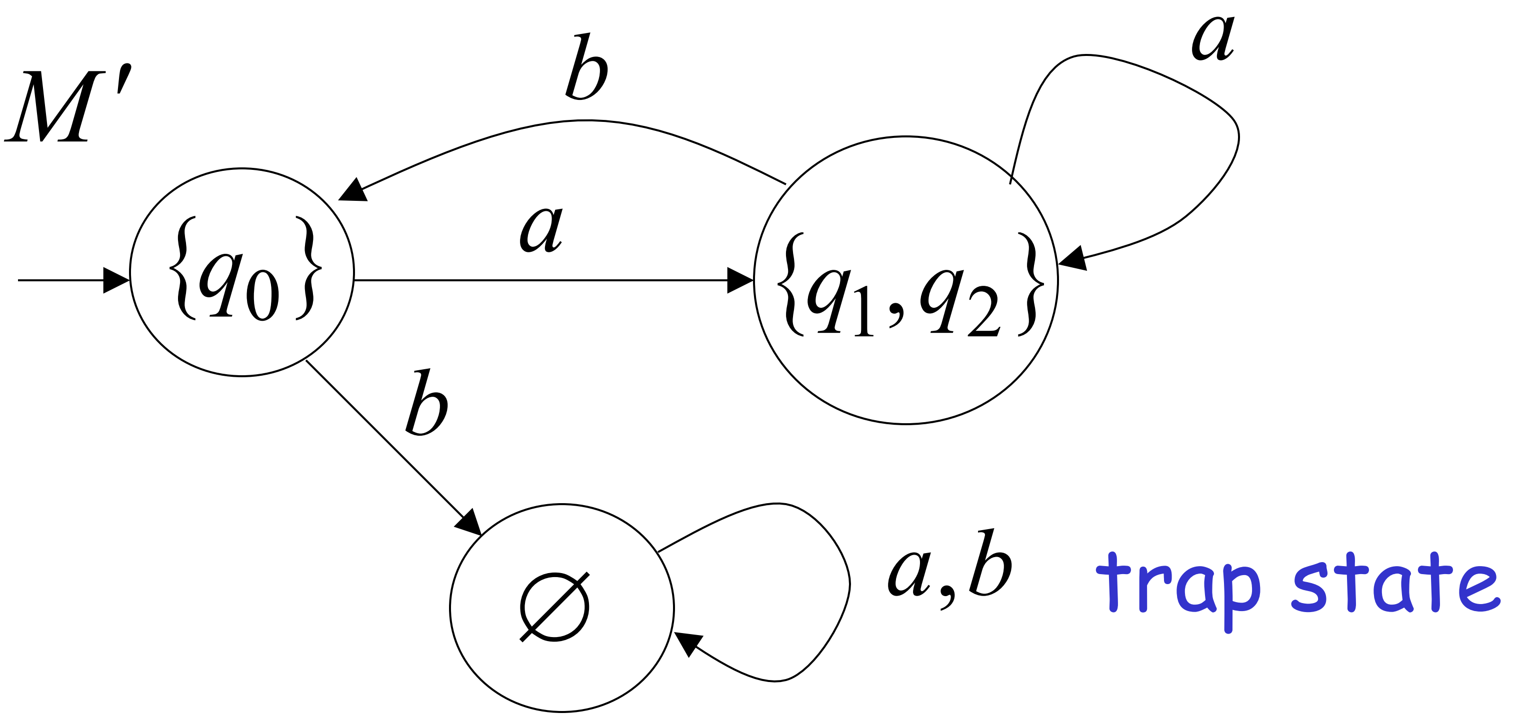
DFA M'



NFA M

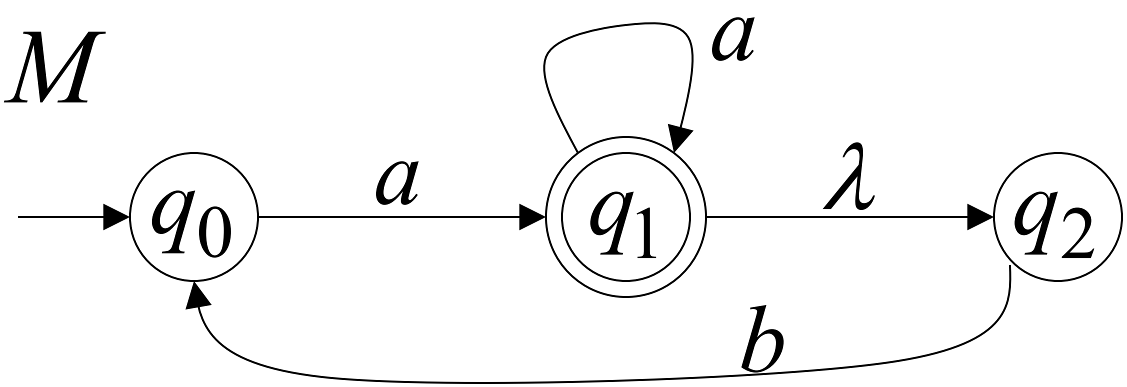


DFA M'



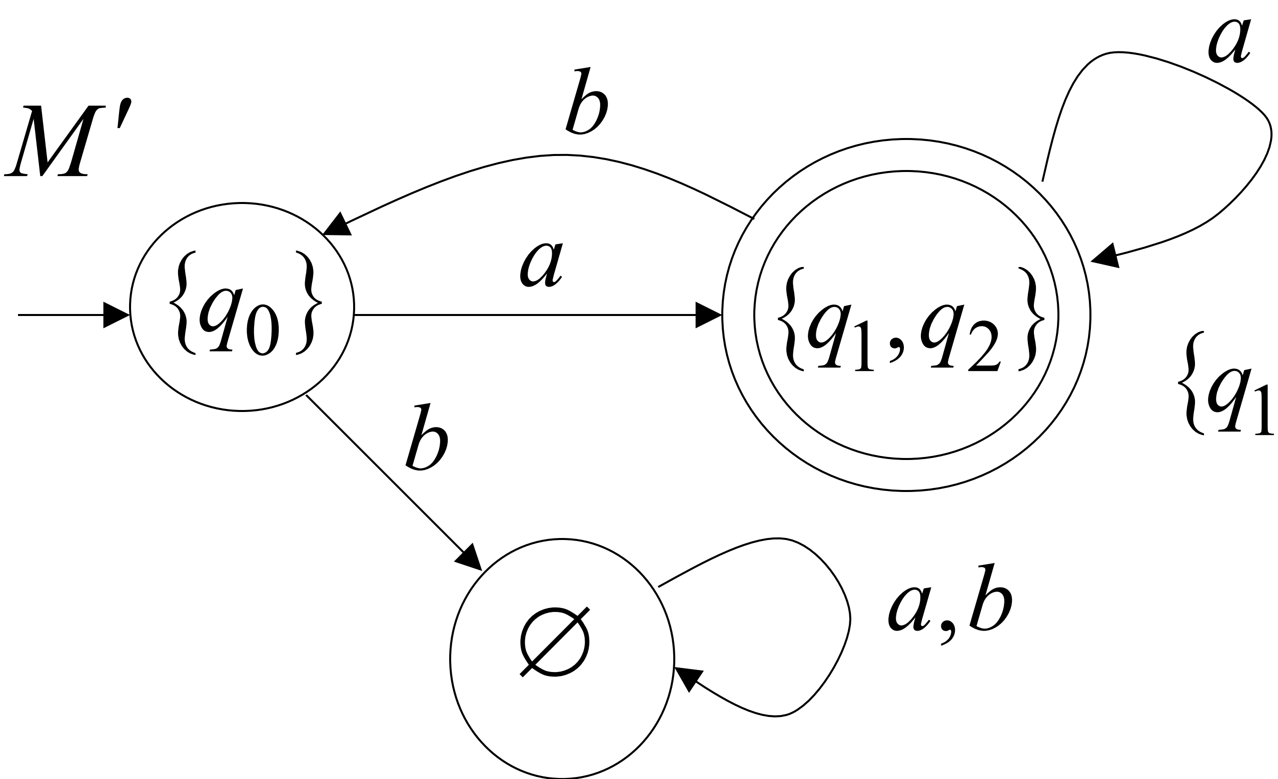
END OF CONSTRUCTION

NFA M



$q_1 \in F$

DFA M'



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

Lemma:

If we convert NFA M to DFA M' then the two automata are equivalent:

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