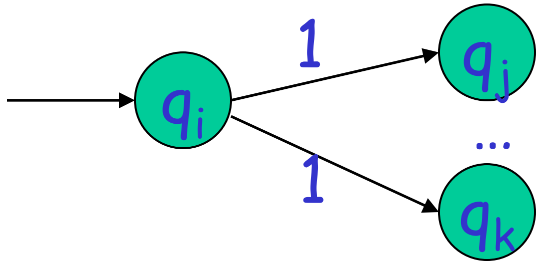
Non-Deterministic Finite Automata

Non-deterministic Finite Automata (NFA)

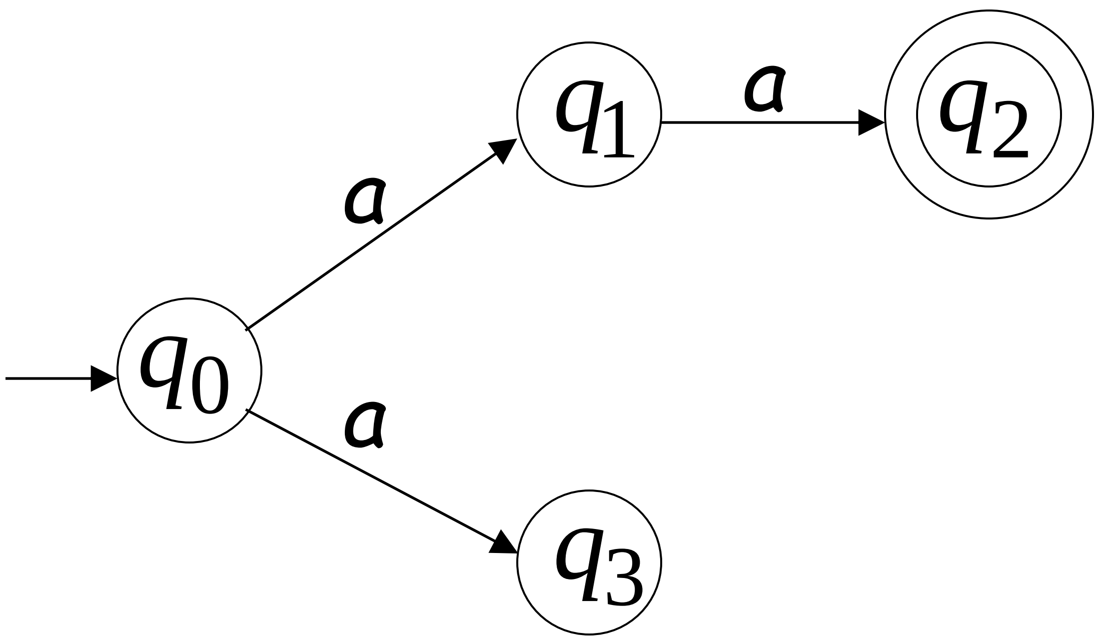
- A Non-deterministic Finite Automaton (NFA)
  - is of course "non-deterministic"
    - Implying that the machine can exist in more than one states at the same time
    - Transitions could be non-deterministic



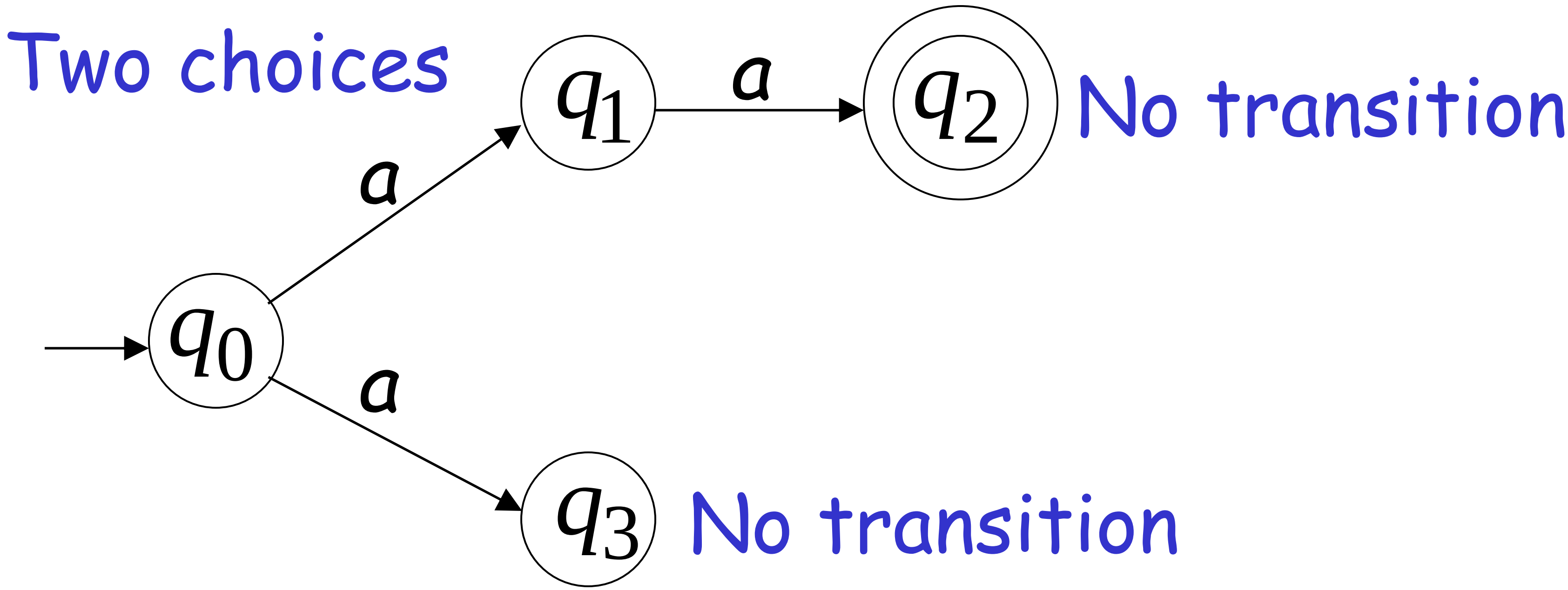
- Each transition function therefore maps to a set of states

Nondeterministic Finite Automaton (NFA)

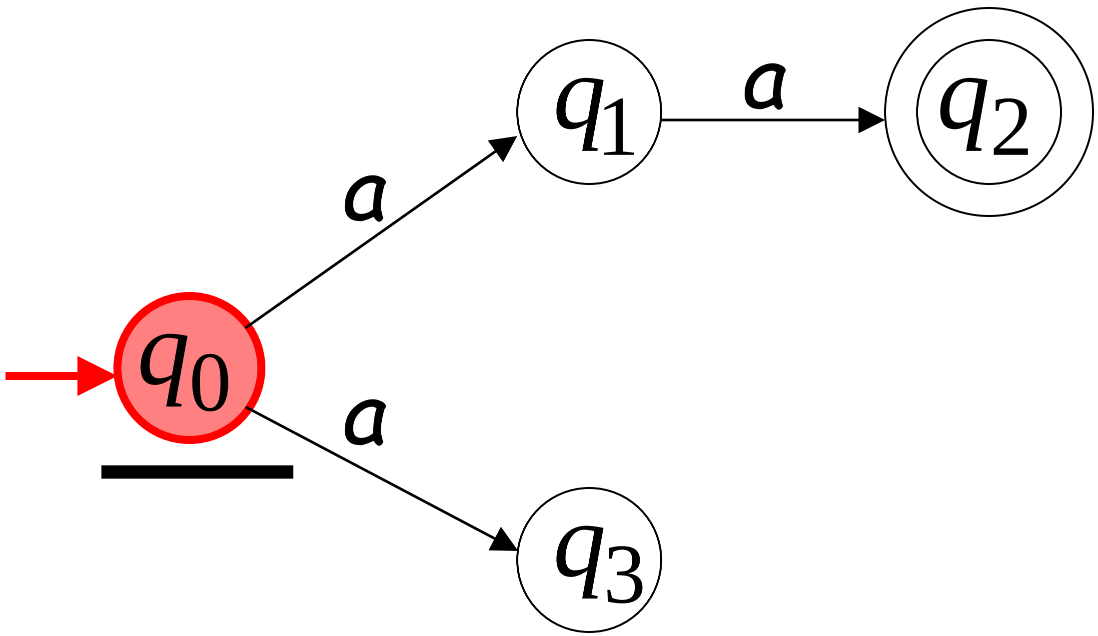
Alphabet = $\{a\}$



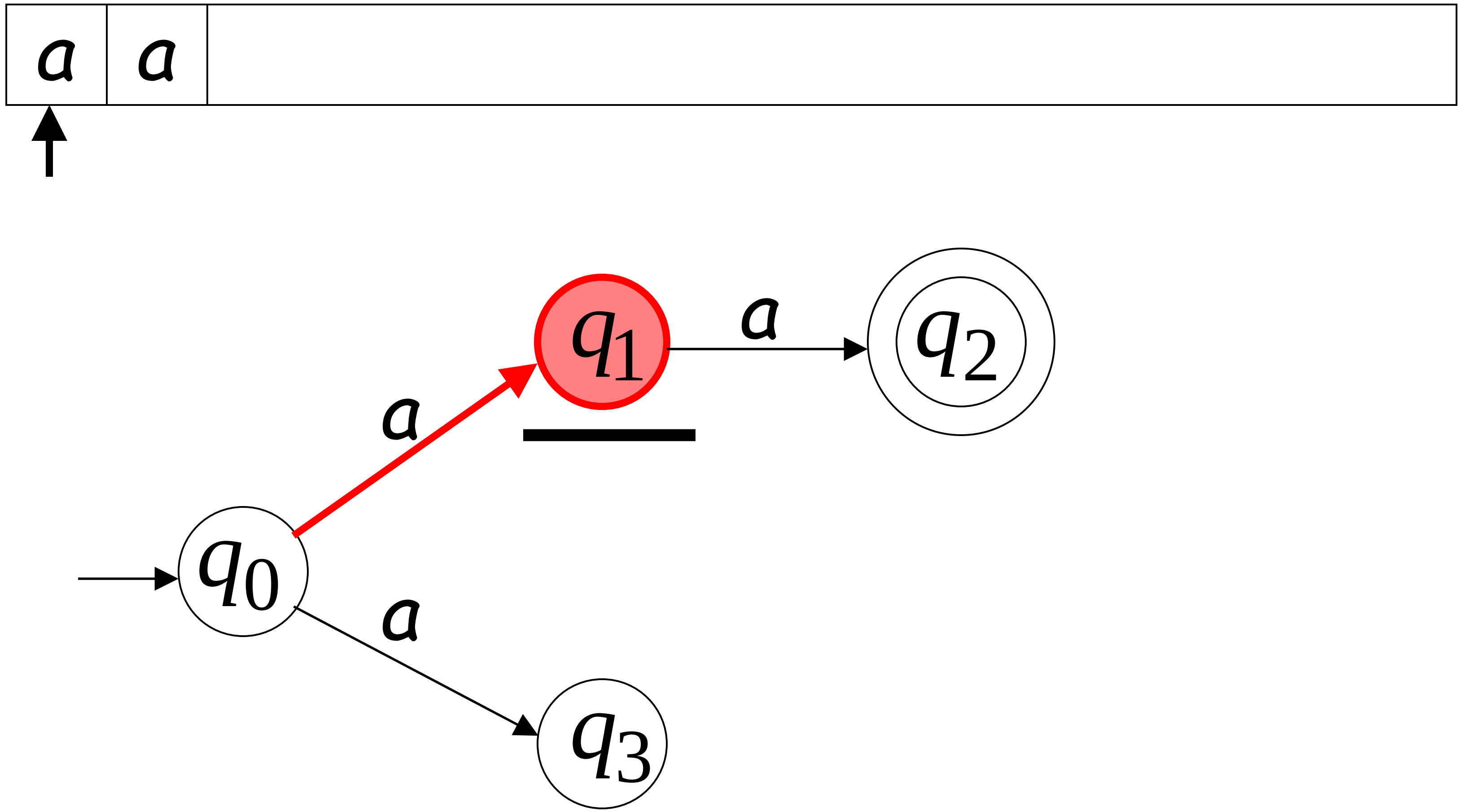
Alphabet = $\{a\}$



First Choice

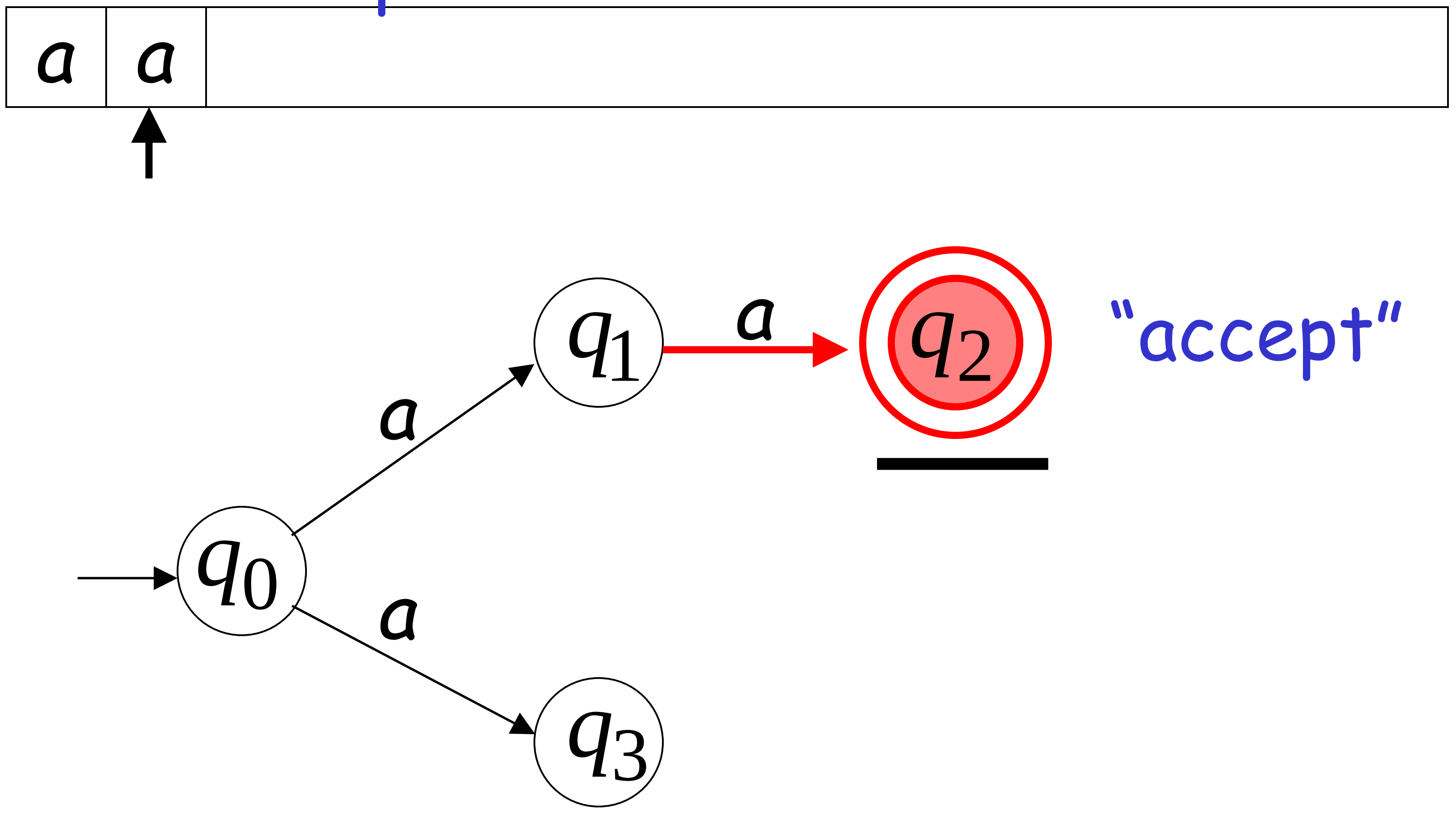


First Choice

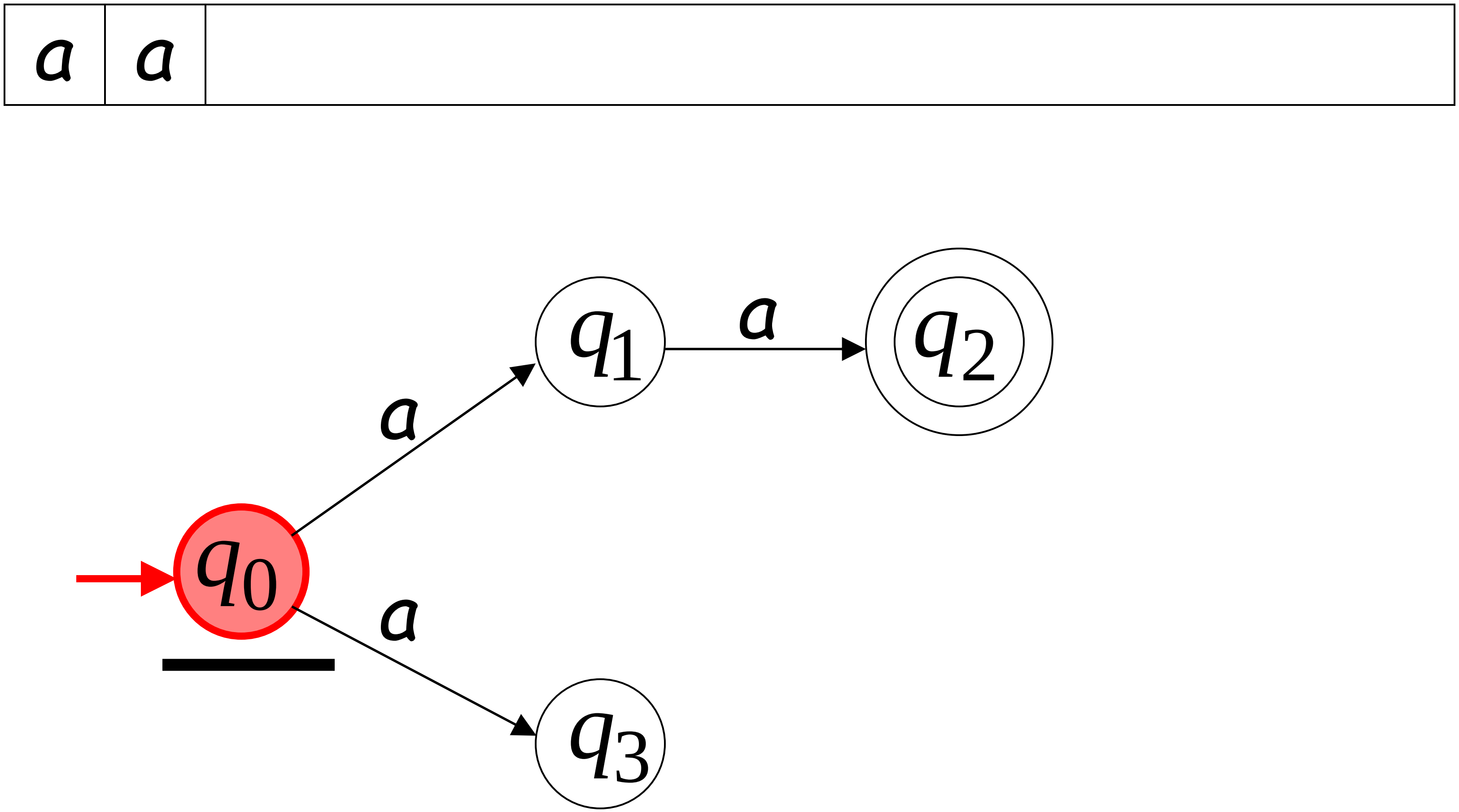


First Choice

All input is consumed

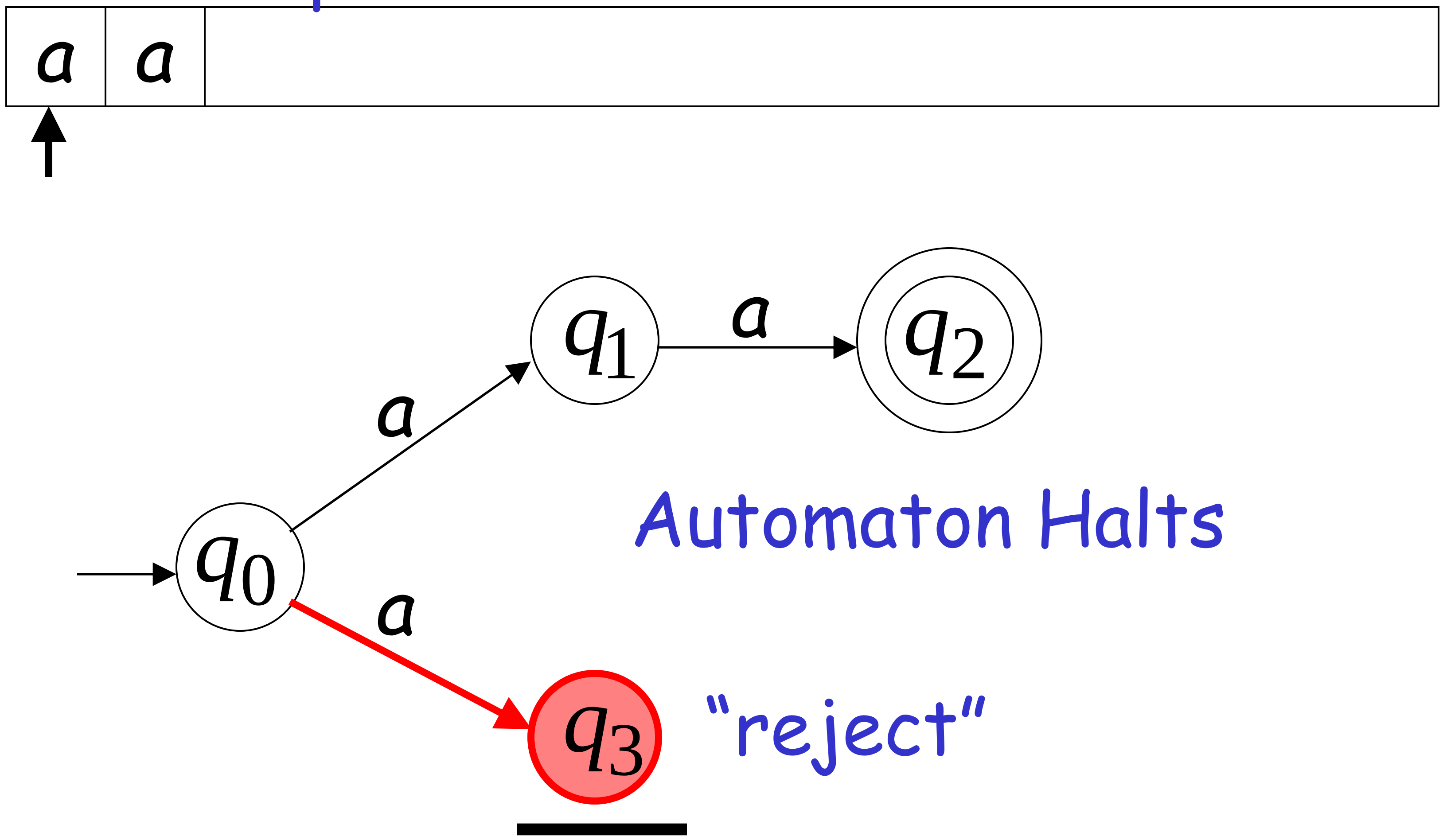


Second Choice



Second Choice

Input cannot be consumed



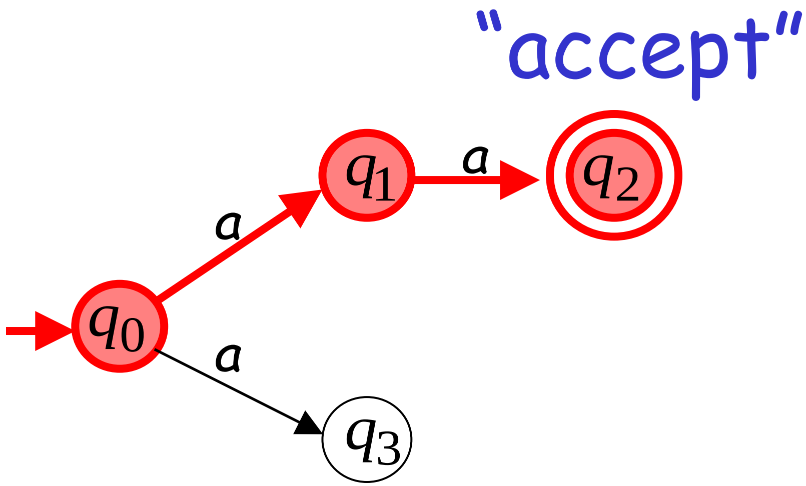
Automaton Halts

"reject"

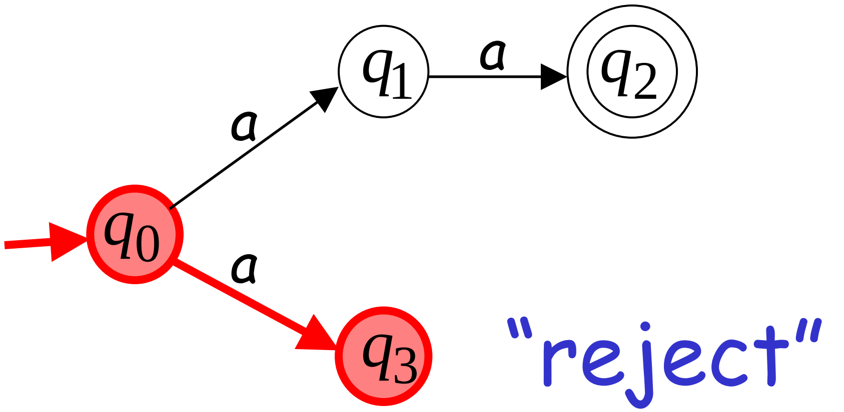
How to use an NFA?

- Input: a word w in Σ^*
- Question: Is w acceptable by the NFA?
- Steps:
  - Start at the "start state", q_0
  - For every input symbol in the sequence w do
    - Determine all possible next states from all current states, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed and if at least one of the current states is a final state then *accept w;*
  - Otherwise, *reject w.*

aa is accepted by the NFA:

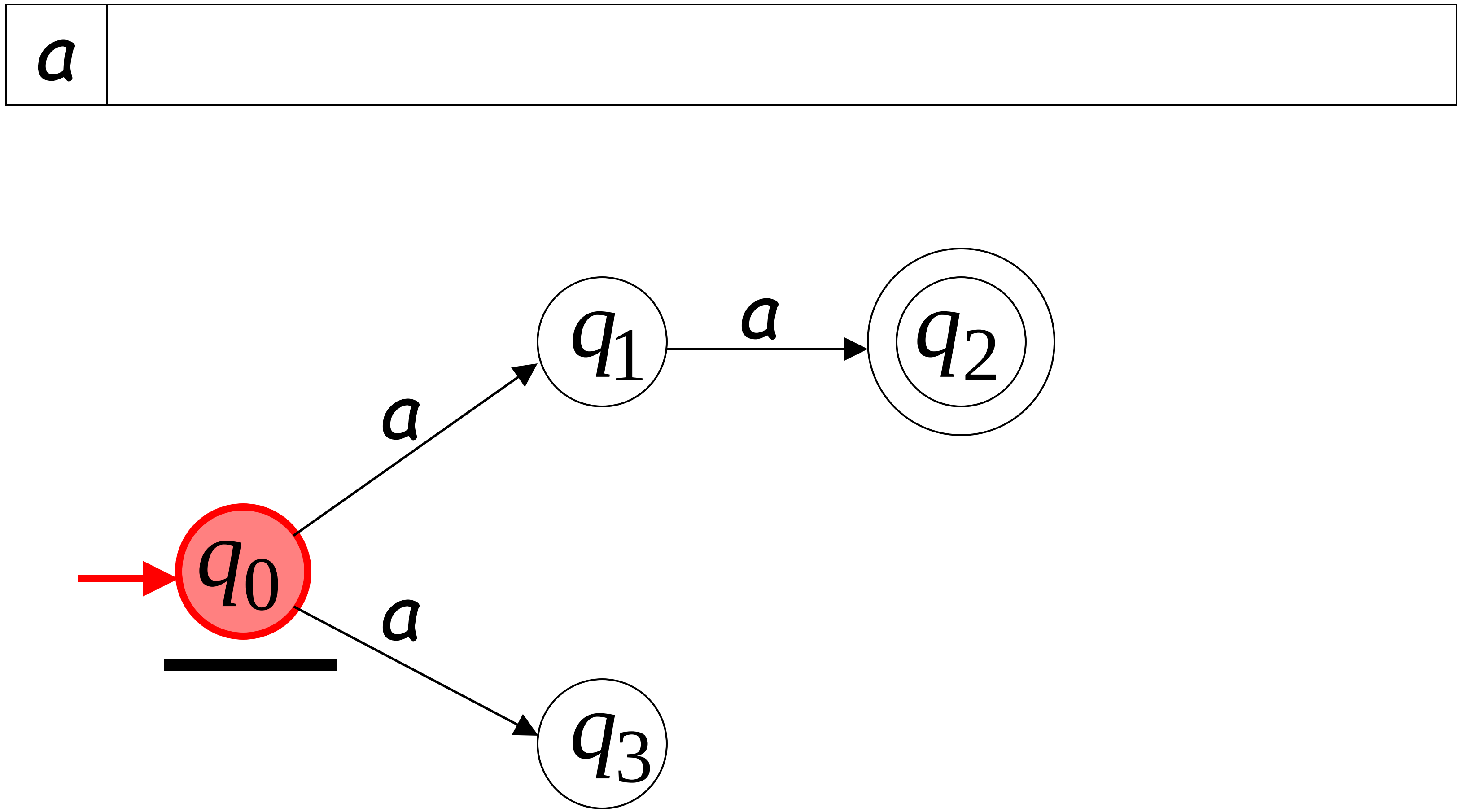


because this computation accepts aa

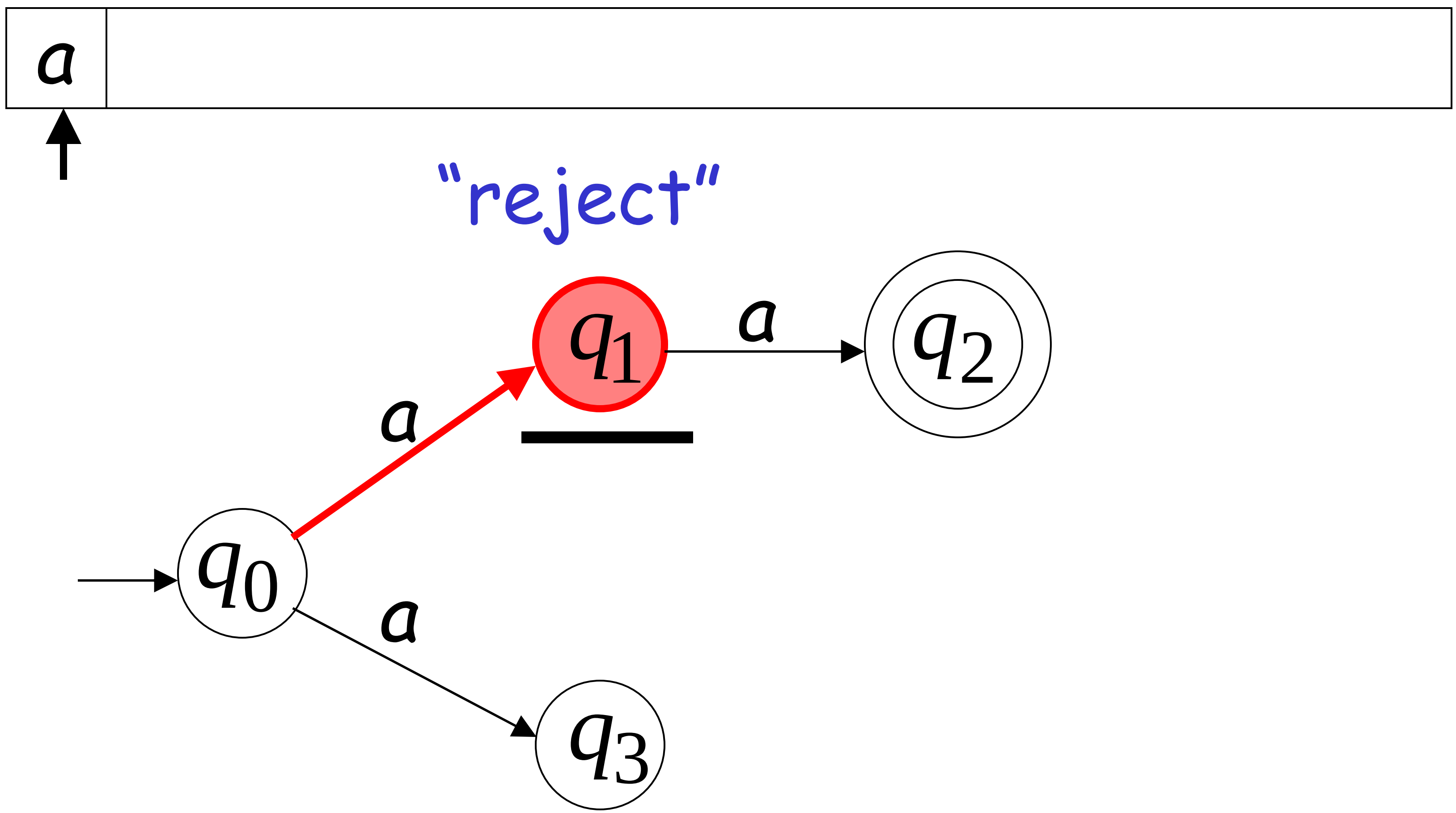


this computation is ignored

Rejection example

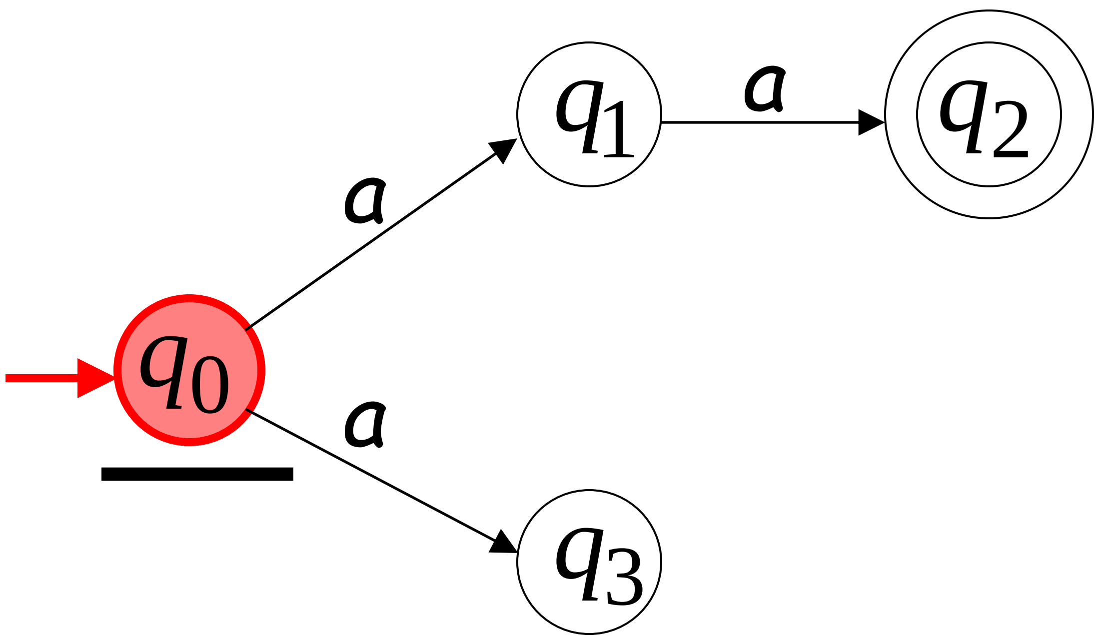


First Choice

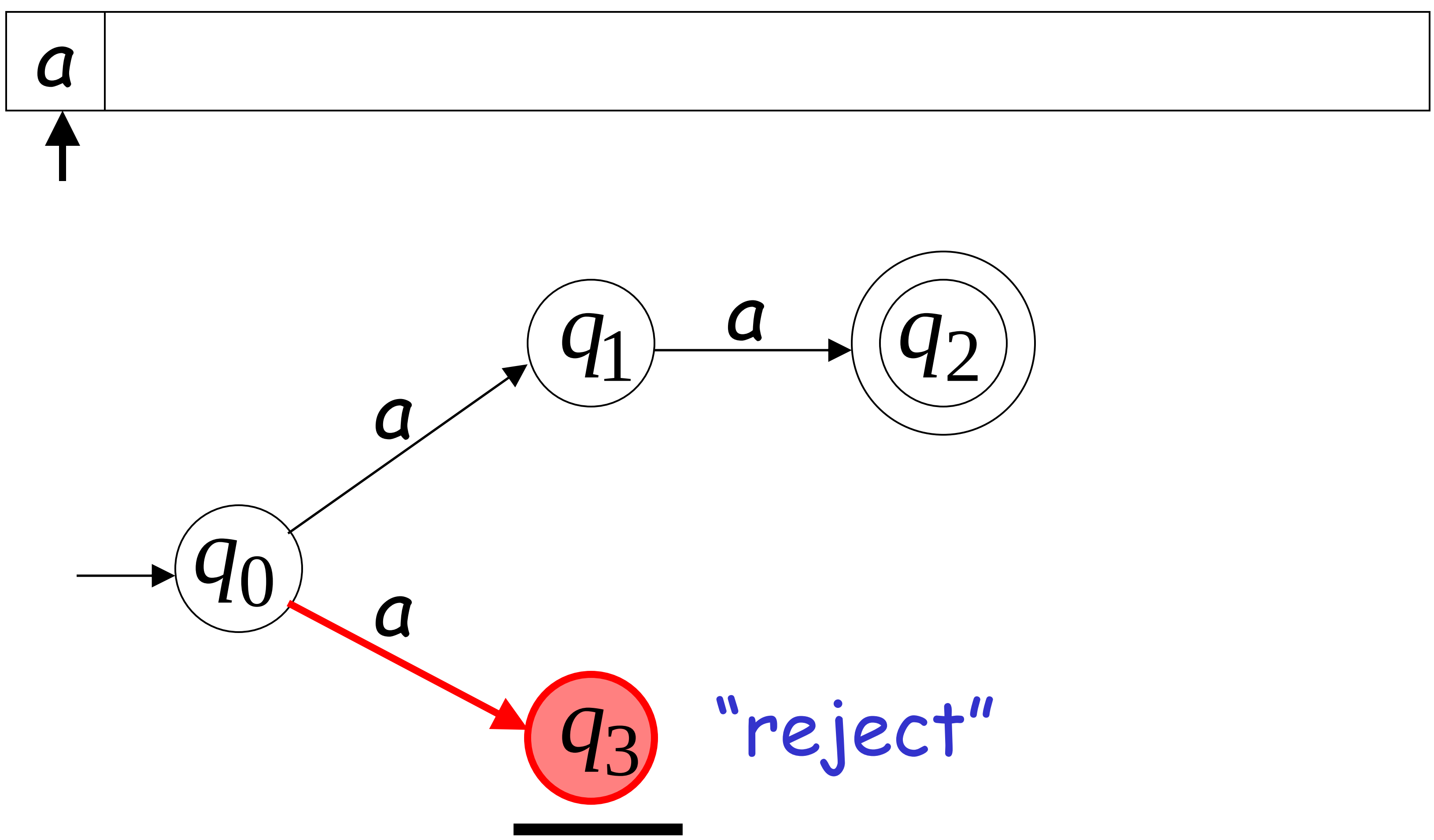


Second Choice

a	

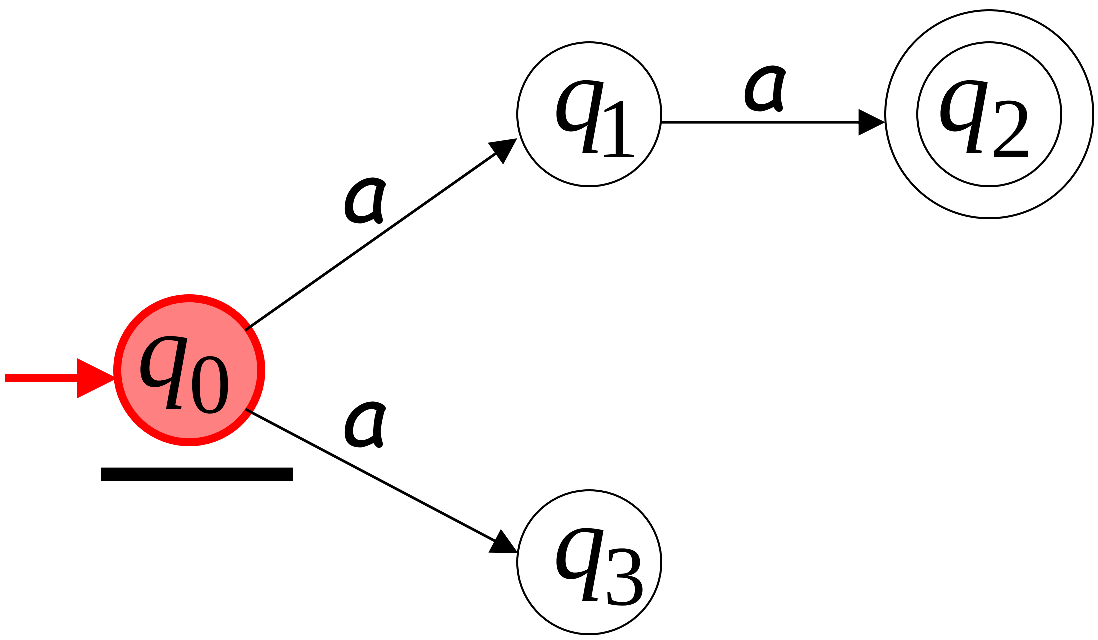


Second Choice

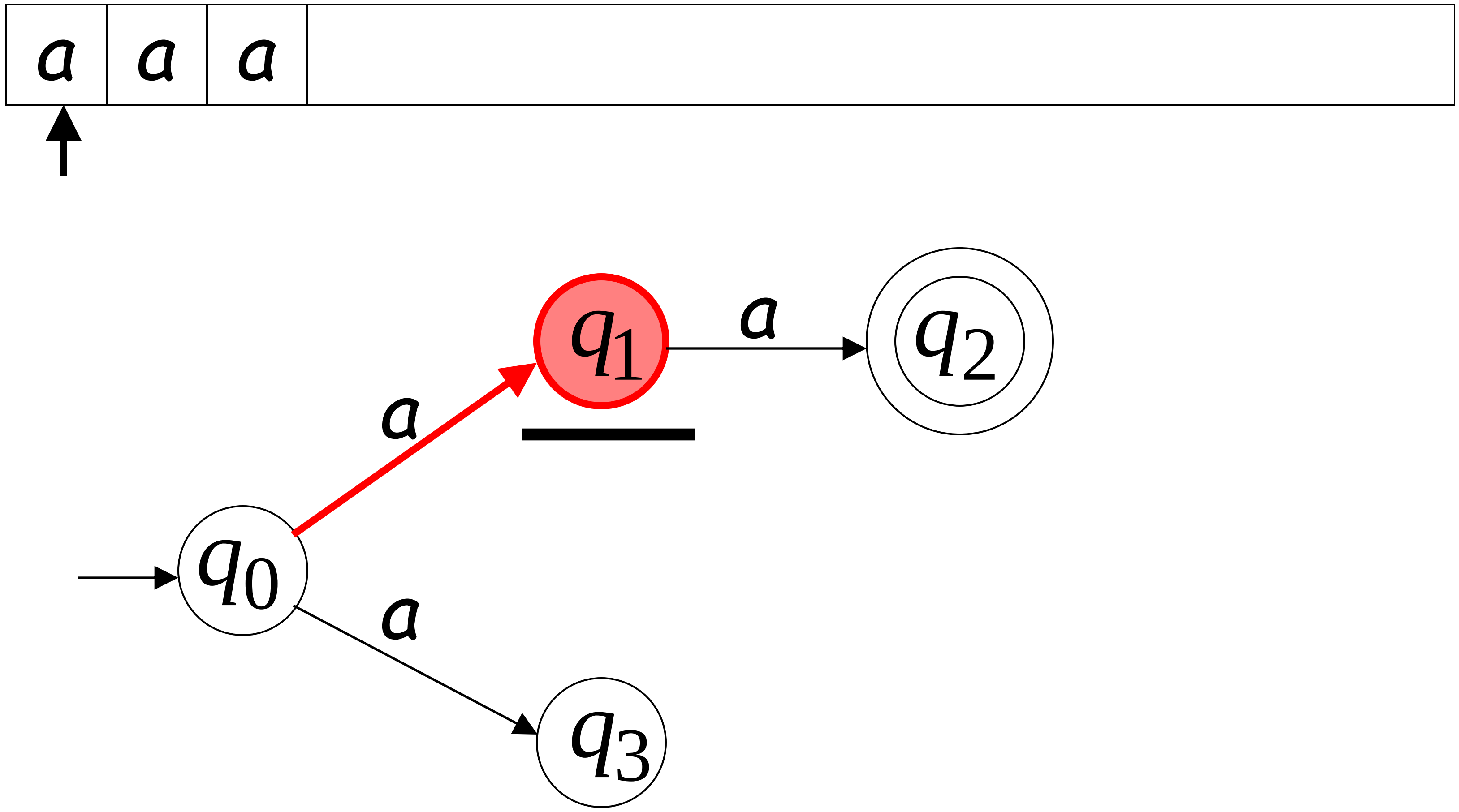


Another Rejection example

a	a	a	

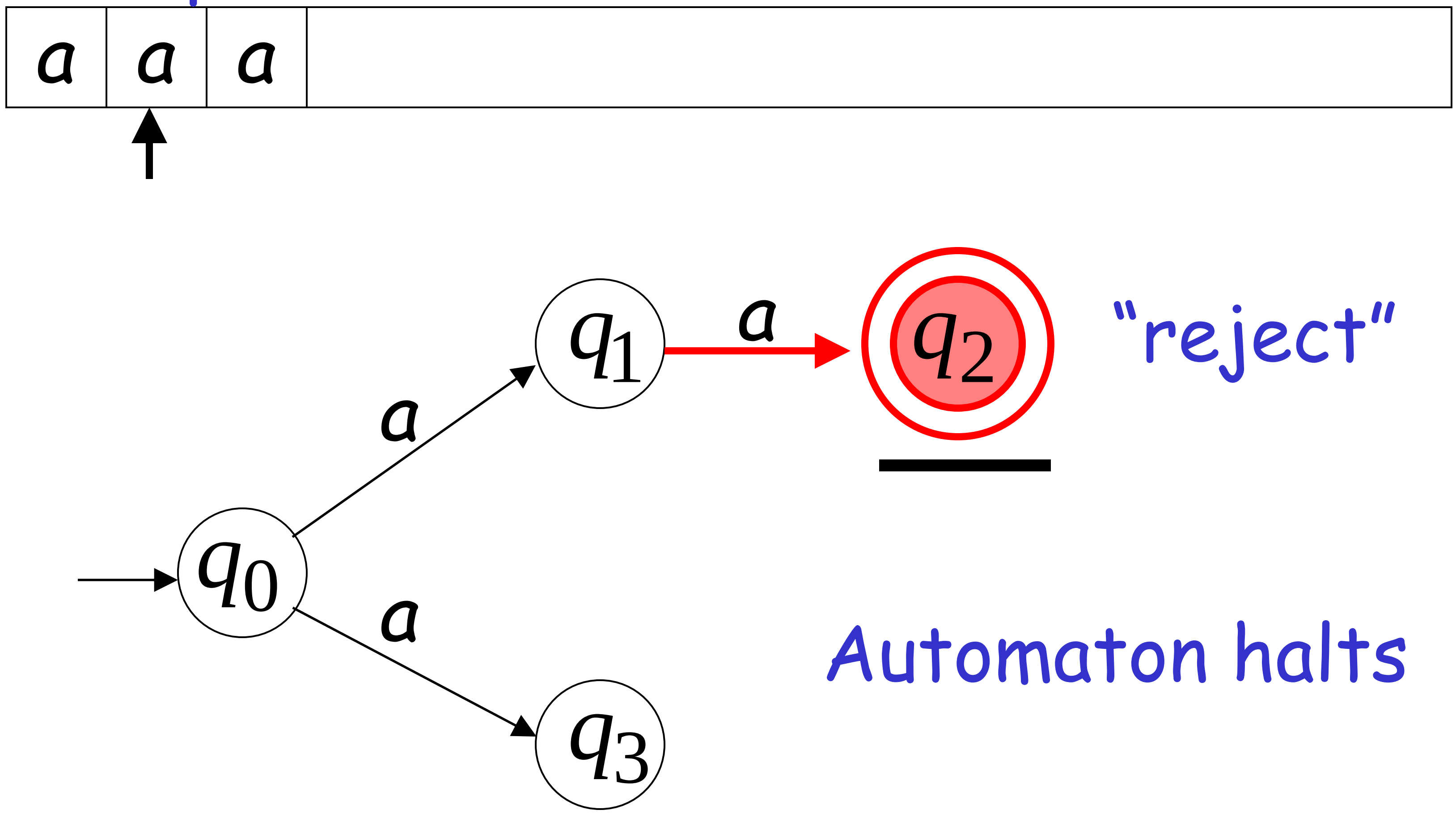


First Choice



First Choice

Input cannot be consumed

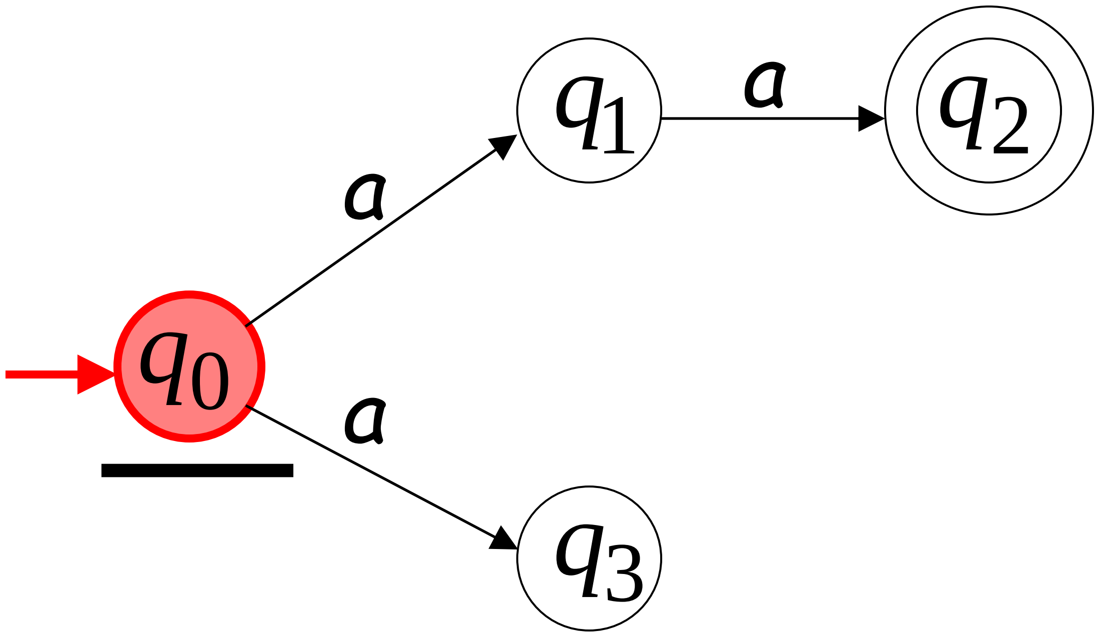


"reject"

Automaton halts

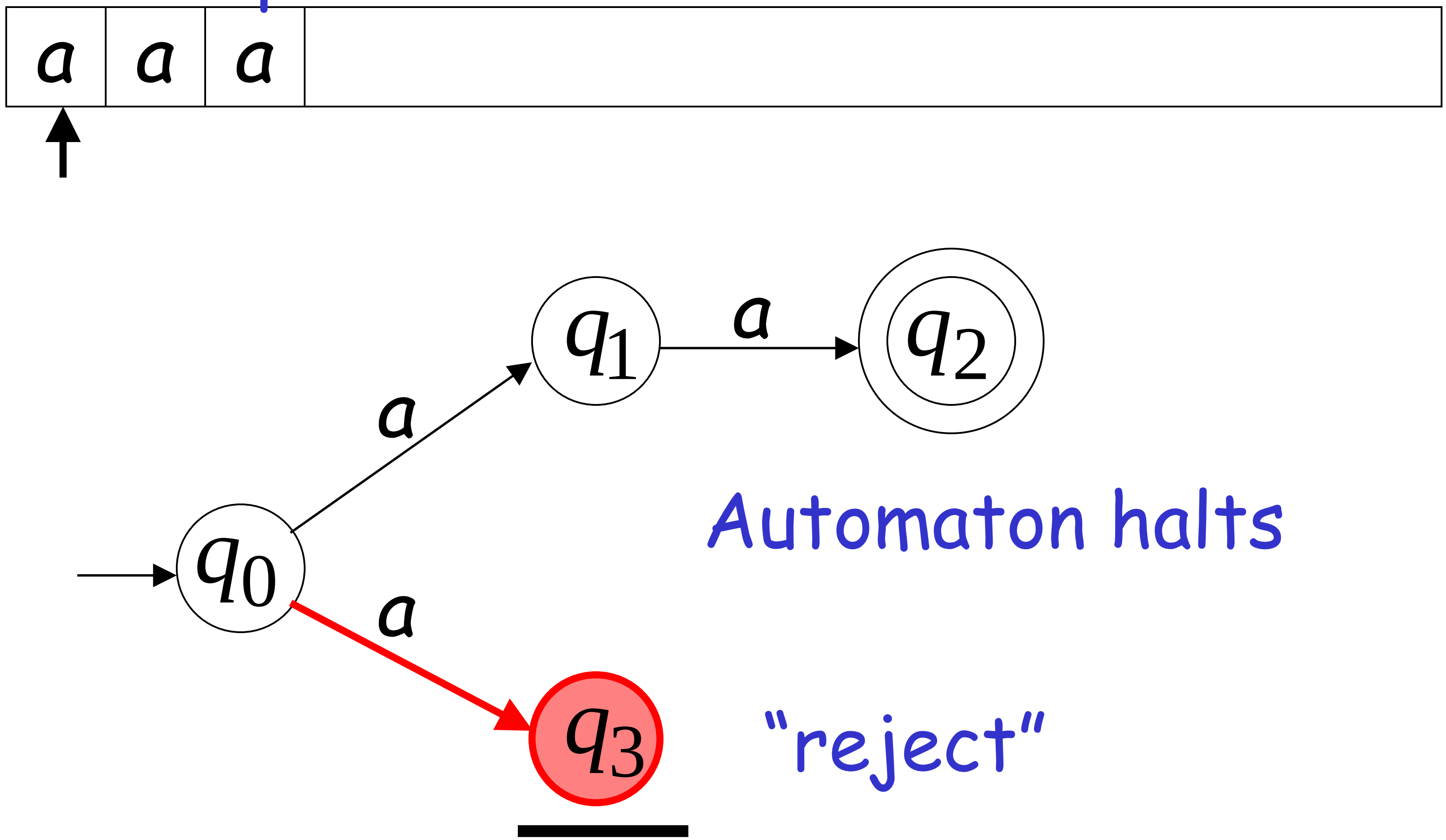
Second Choice

a	a	a	



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.

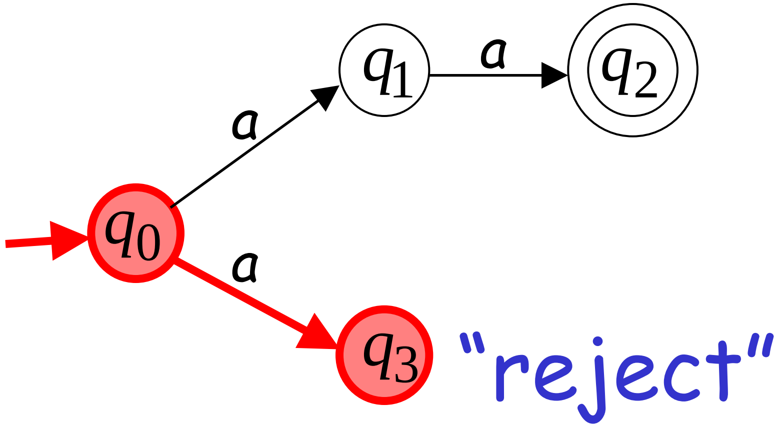
For each possible computation path:

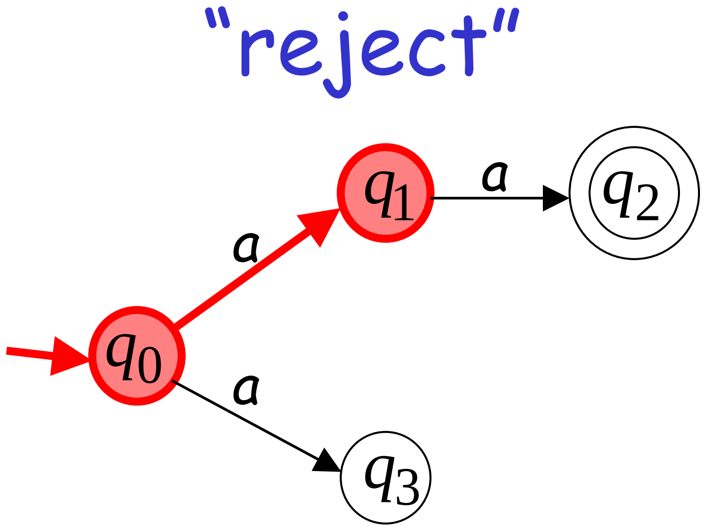
- All the input is consumed and the automaton is in a non-accepting state

OR

- The input cannot be consumed

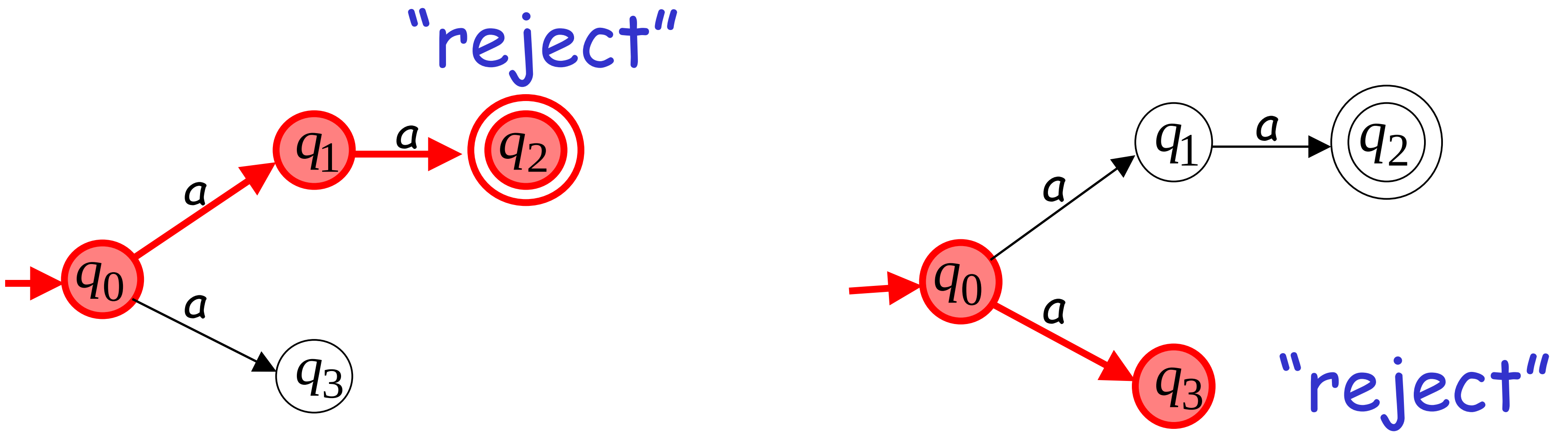
a is rejected by the NFA:





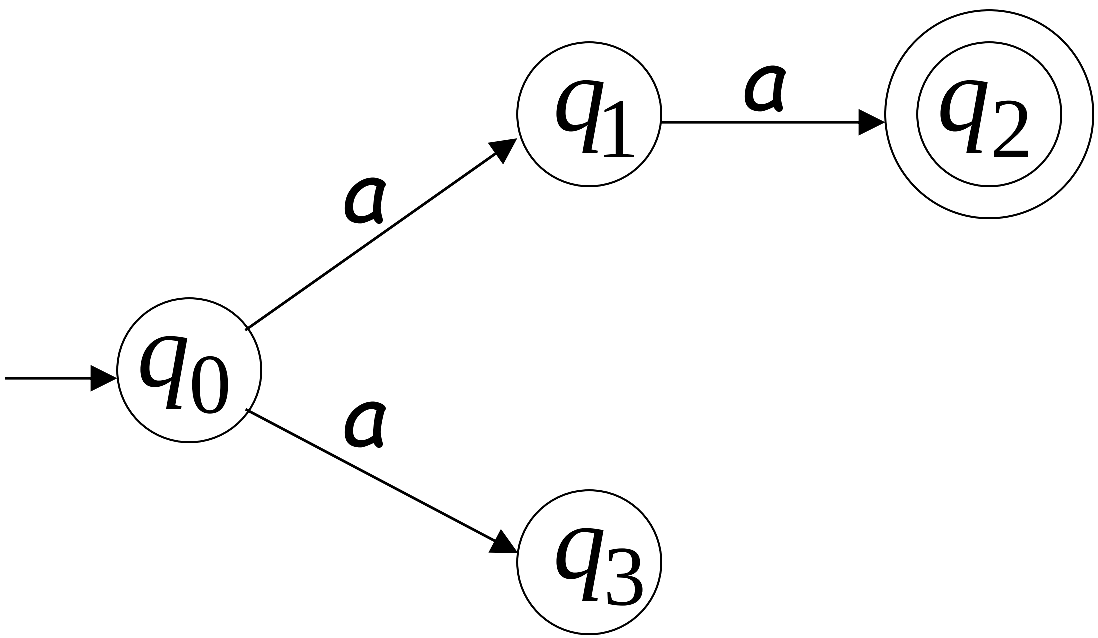
All possible computations lead to rejection

aaa is rejected by the NFA:

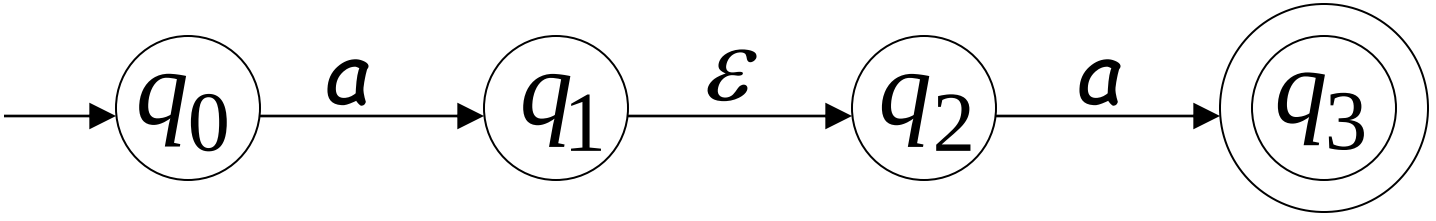


All possible computations lead to rejection

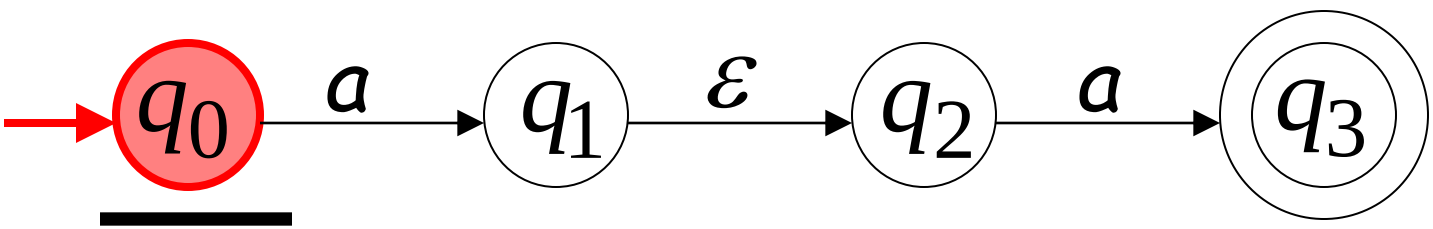
Language accepted: $L = \{aa\}$



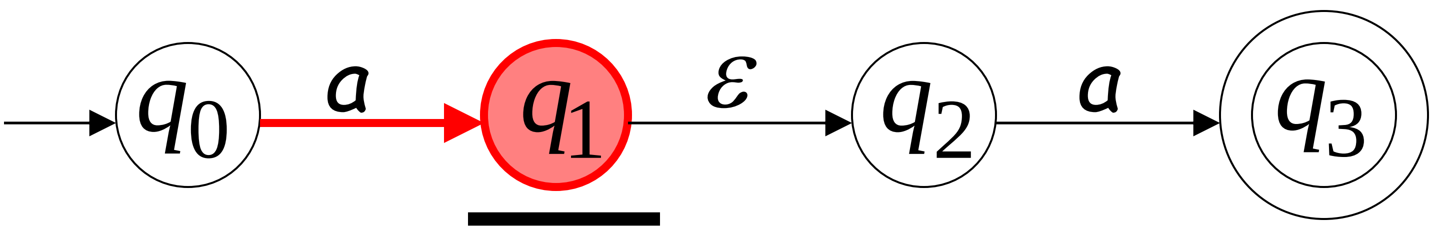
Epsilon Transitions



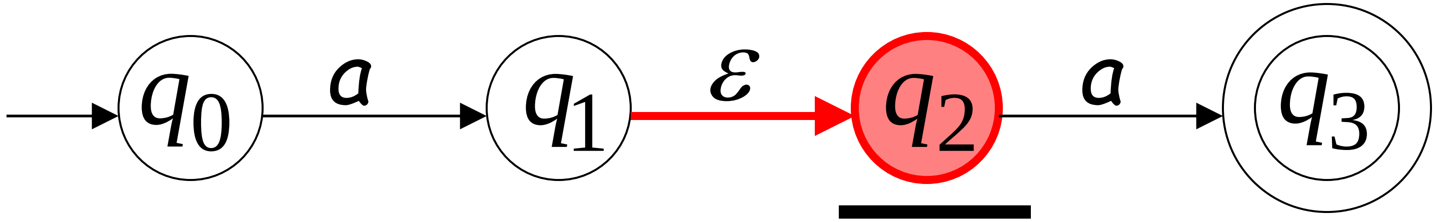

a
a


q_0
a
q_1
ε
q_2
a
q_3


a
a

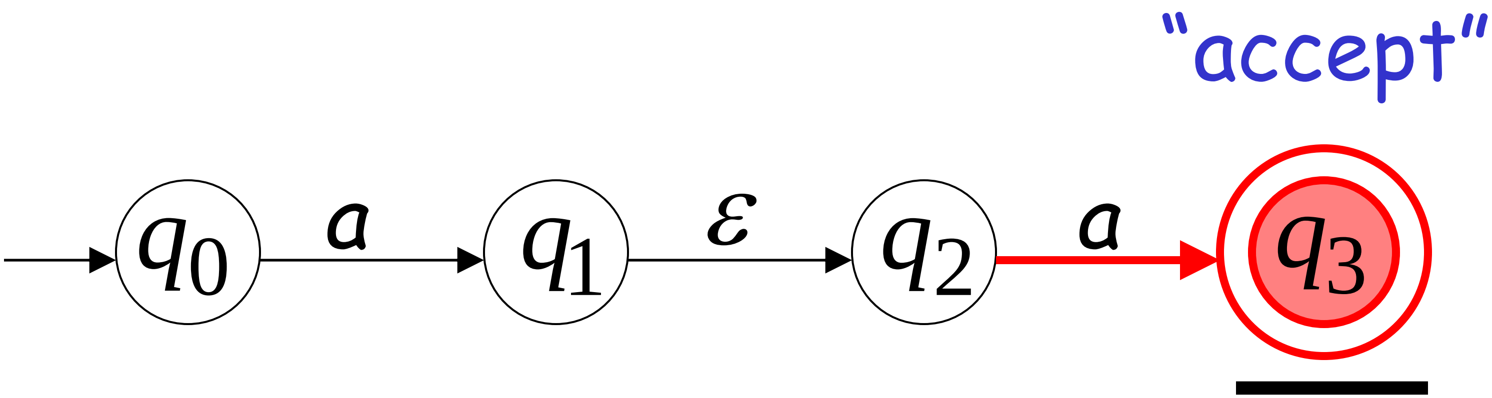

q_0
a
q_1
ε
q_2
a
q_3

input tape head does not move



Automaton changes state

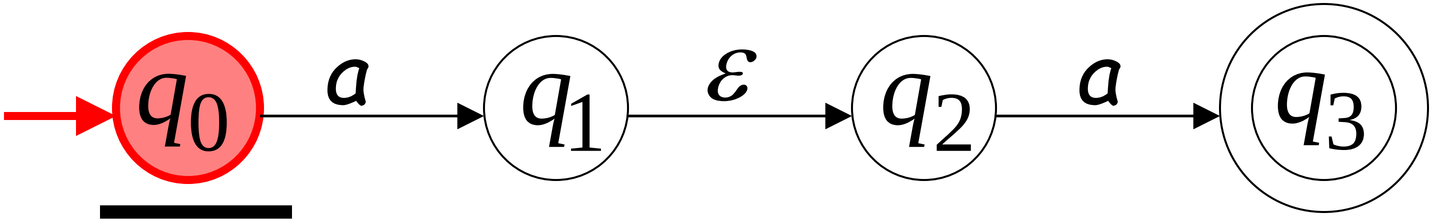
all input is consumed

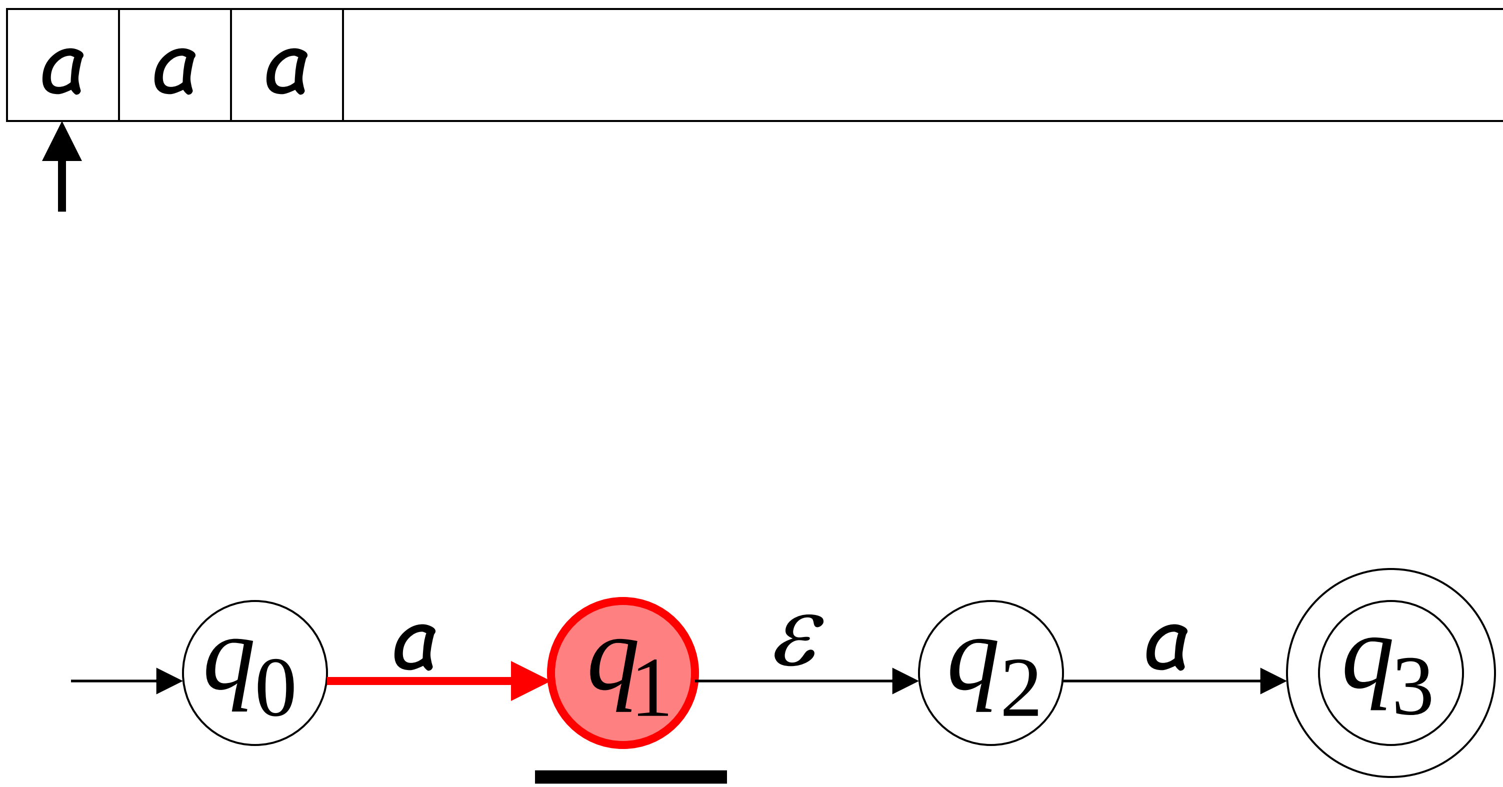


String aa is accepted

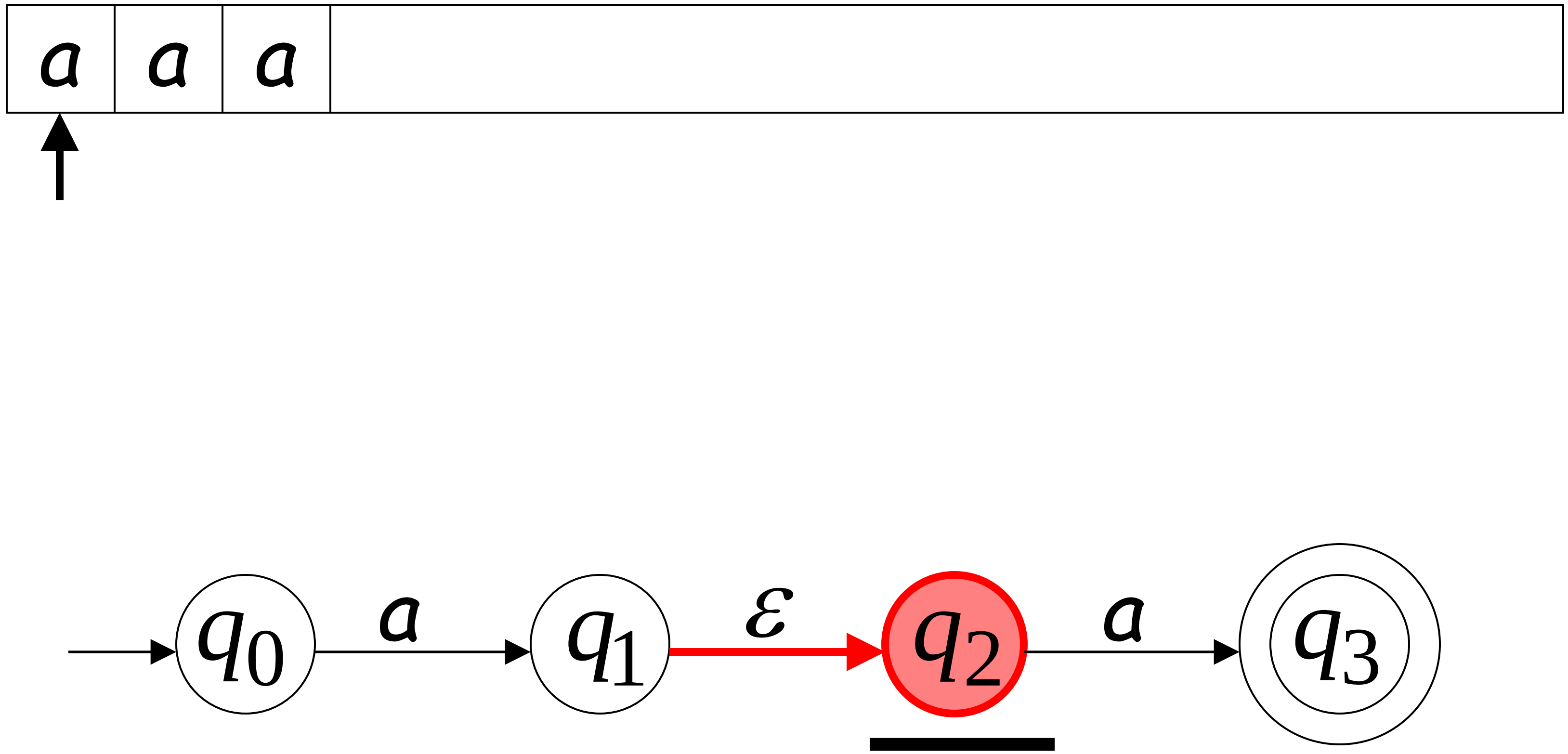
Rejection Example

a	a	a	



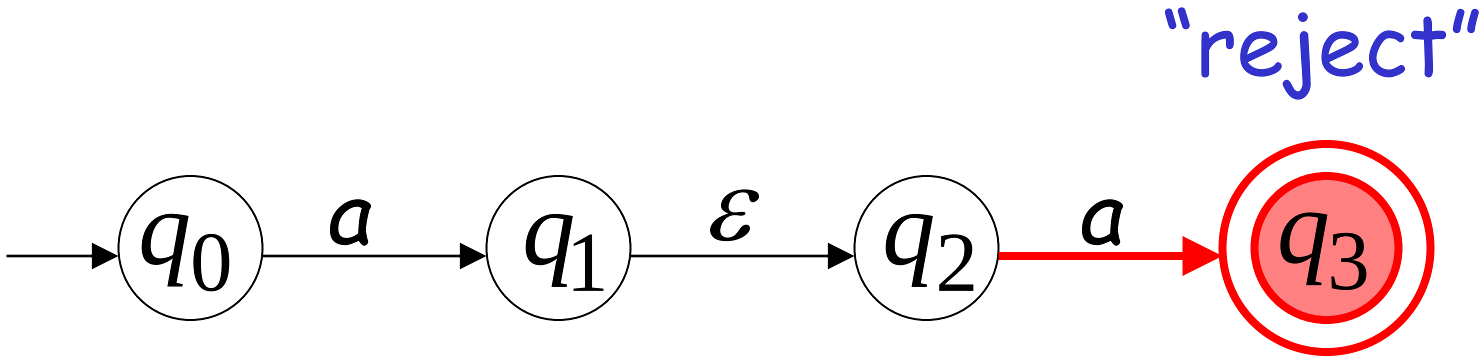

a a a
a
ε
a
q_0
q_1
q_2
q_3

(read head doesn't move)



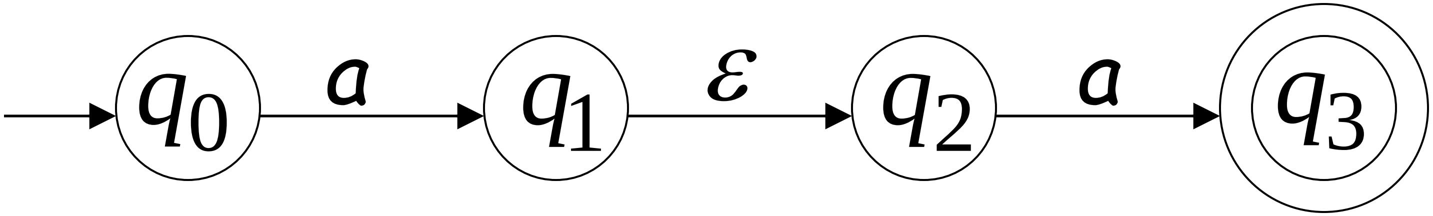
Input cannot be consumed

Automaton halts

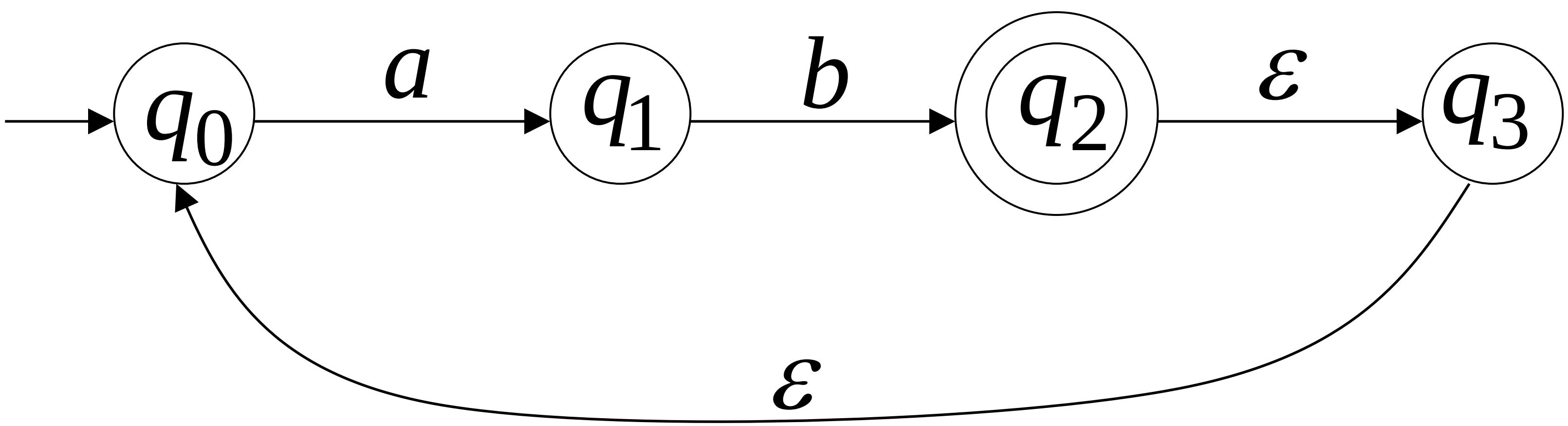


String aaa is rejected

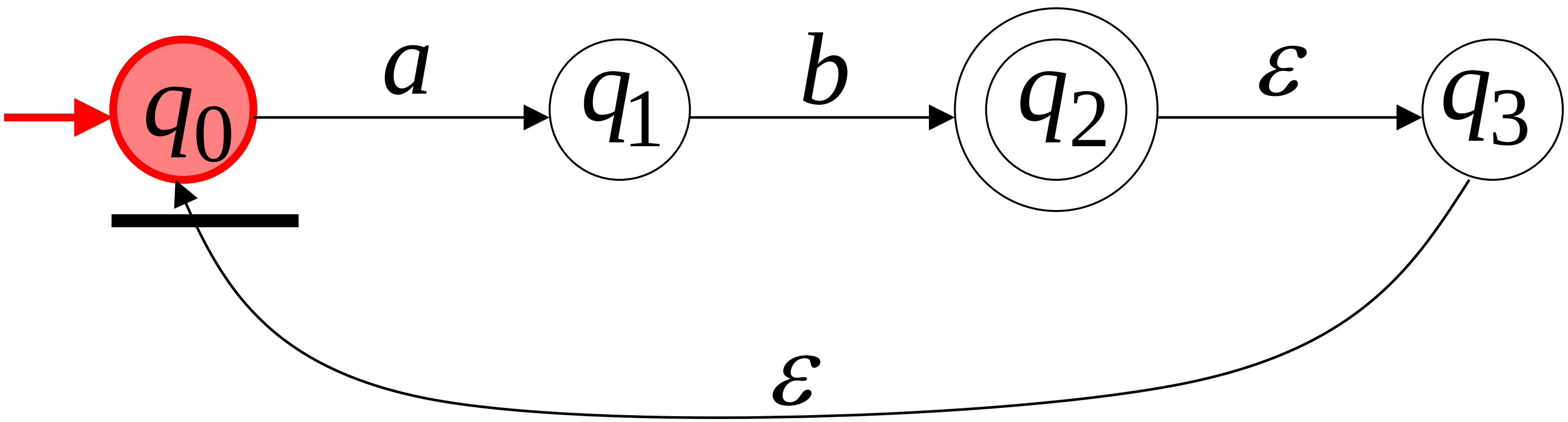
Language accepted: $L = \{aa\}$

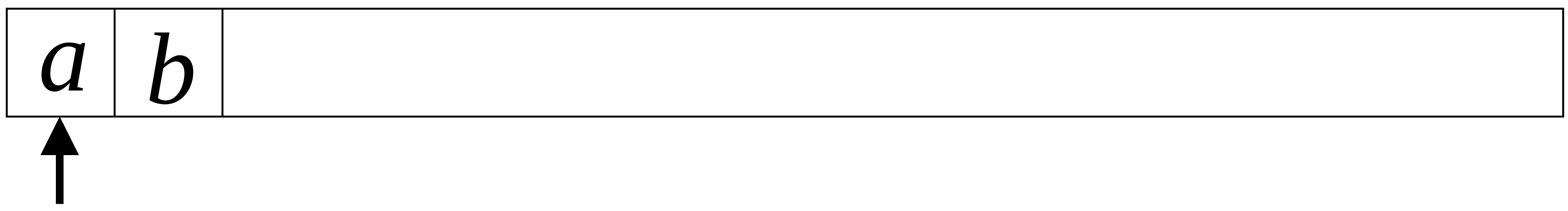


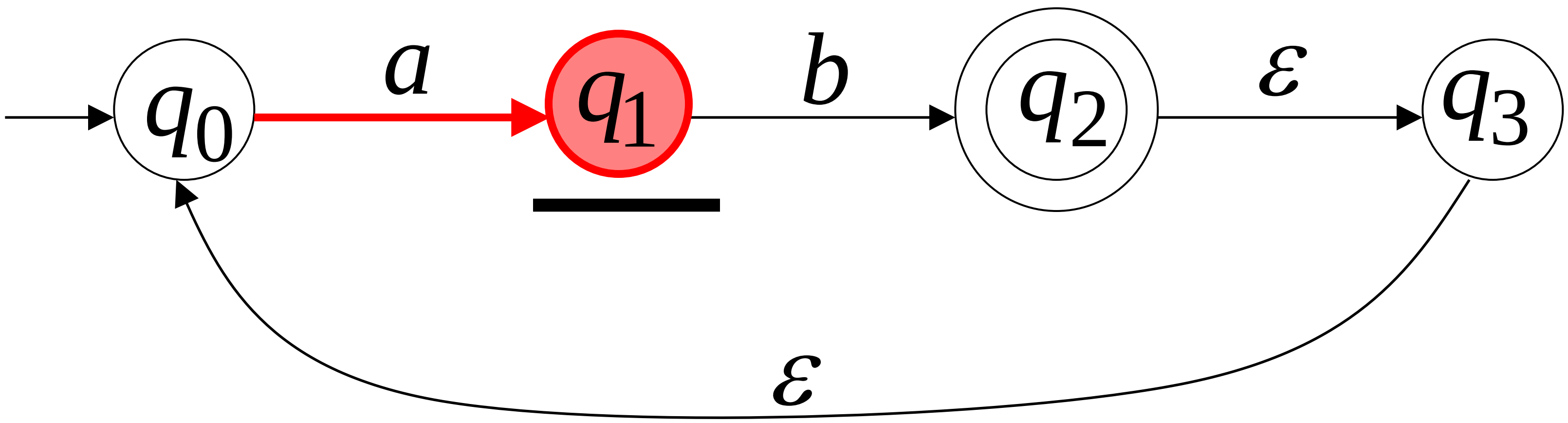
Another NFA Example

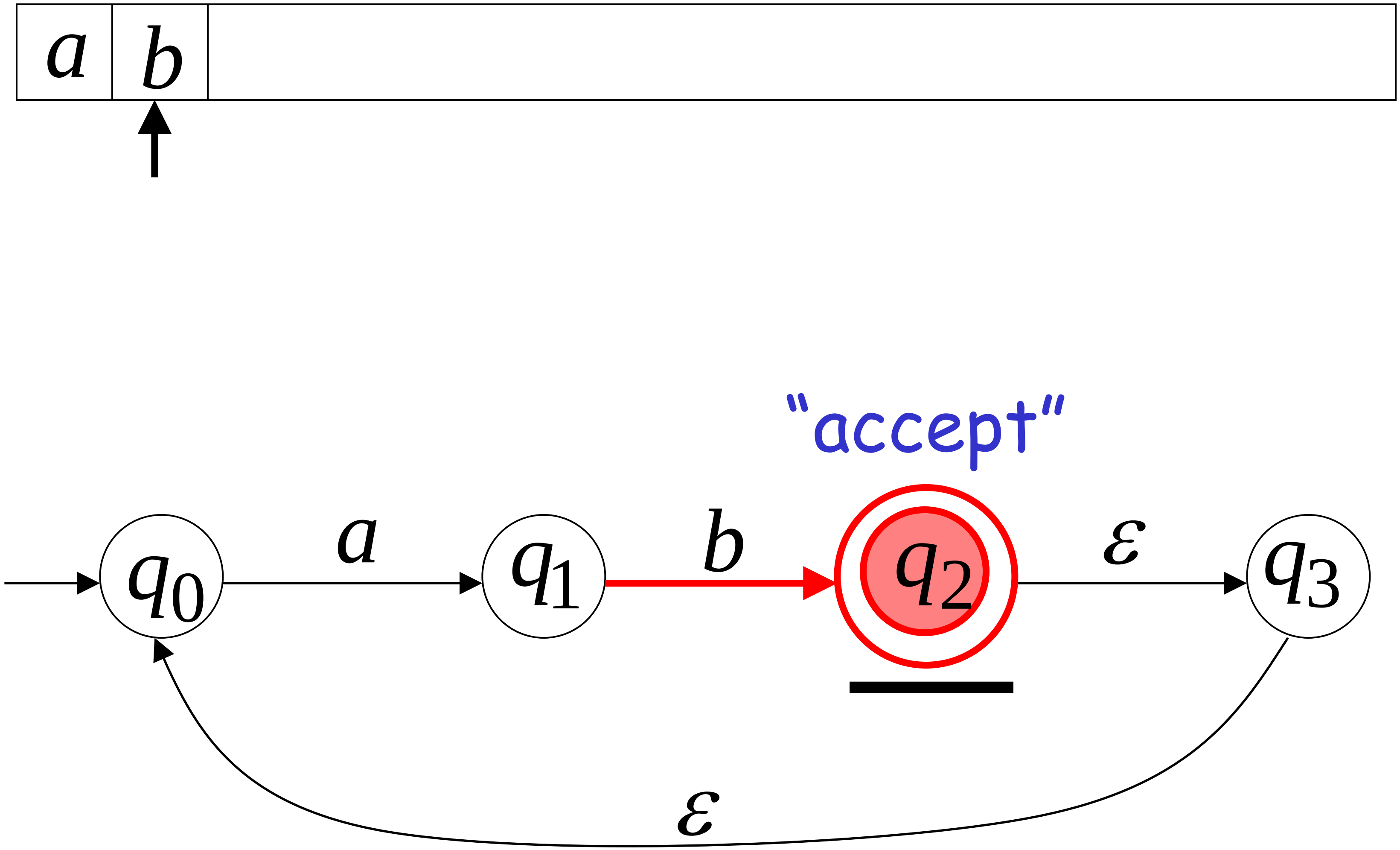



a
b


q_0
a
q_1
b
q_2
ε
q_3
ε


a
b

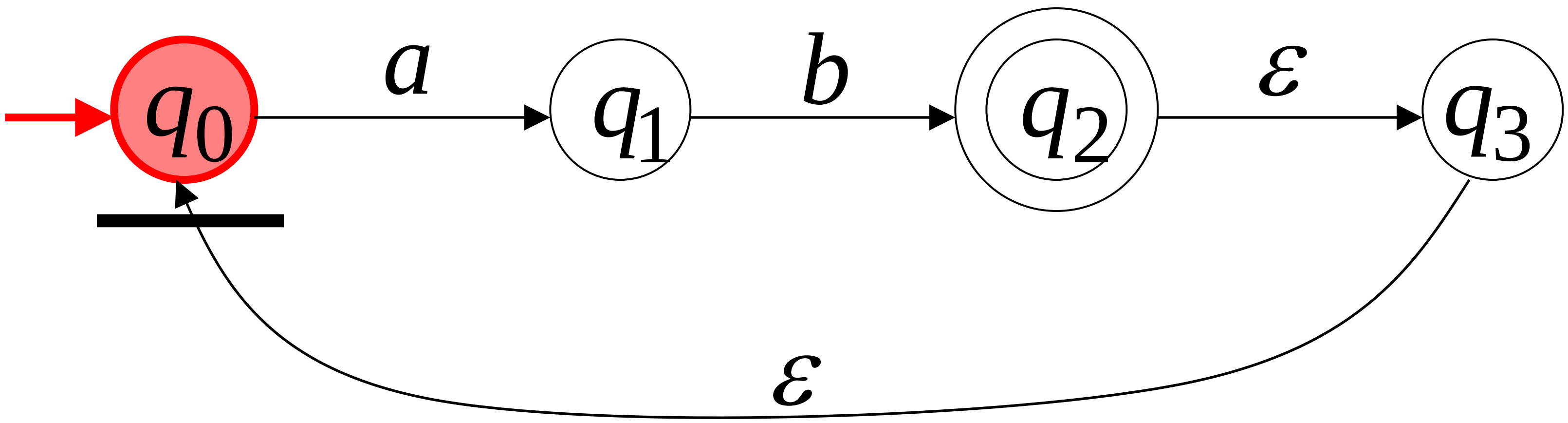

q_0
a
q_1
b
q_2
ε
q_3
ε

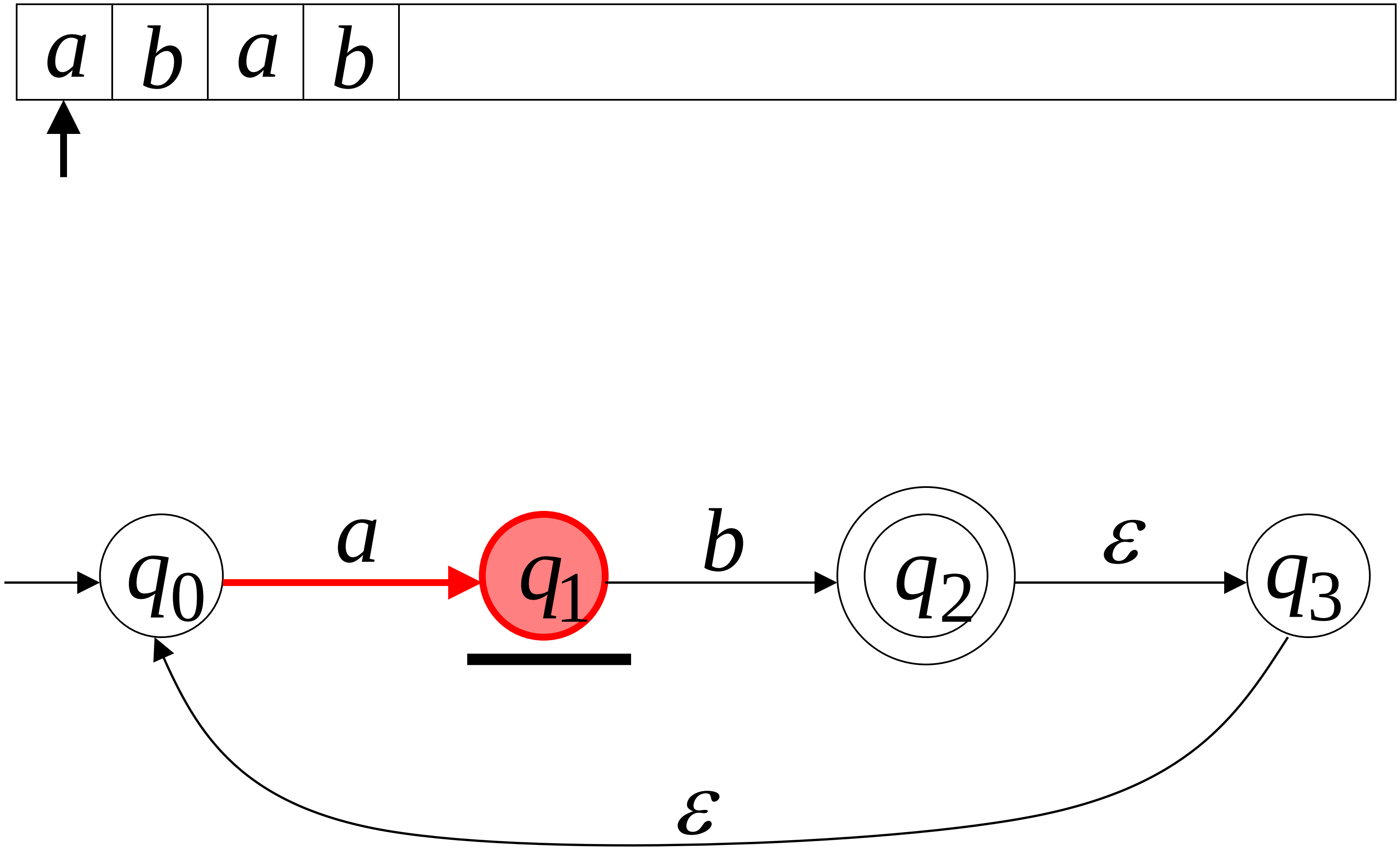


a
b
"accept"
a
b
q_0
q_1
q_2
q_3
ε
ε

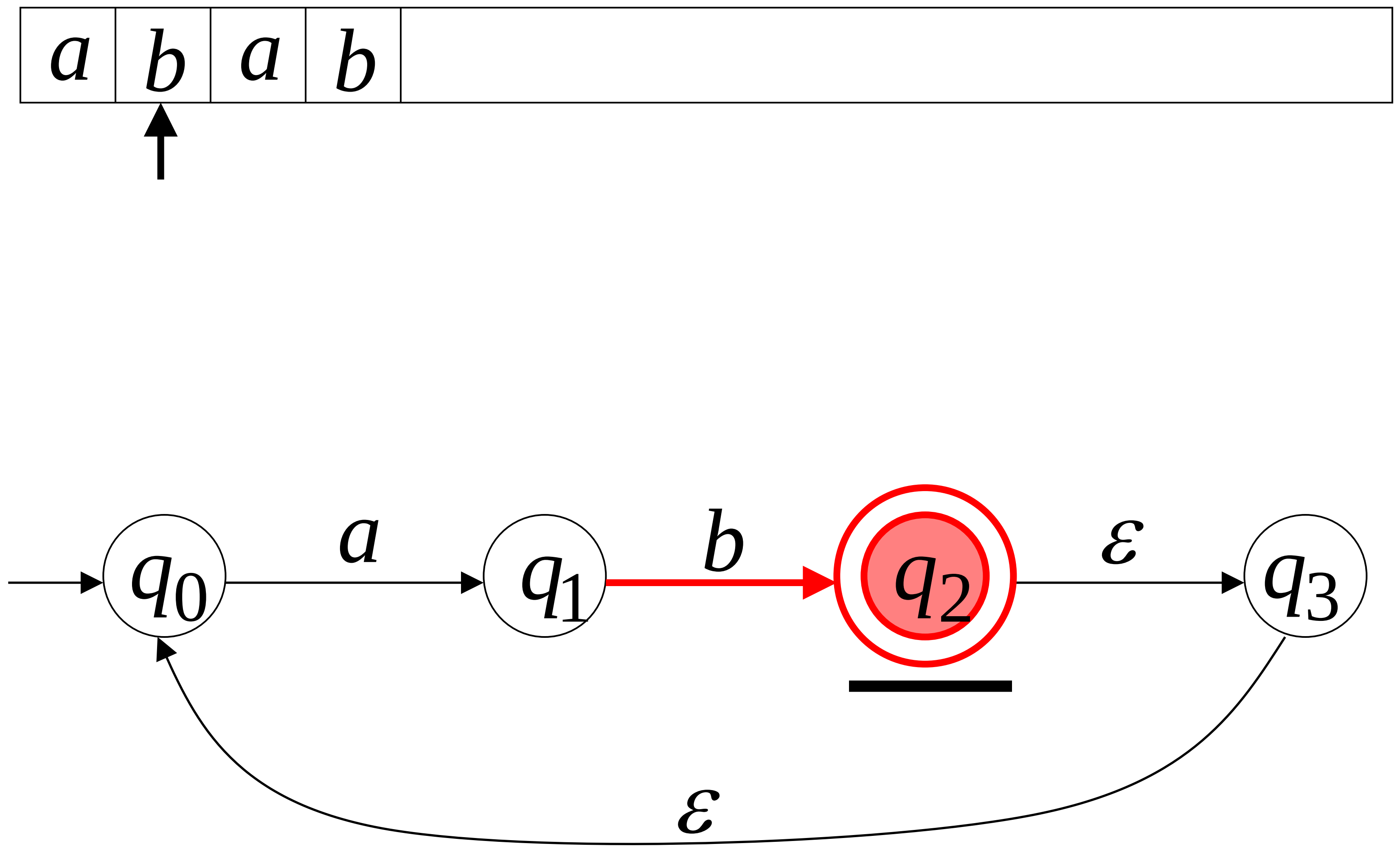
Another String

a	b	a	b	

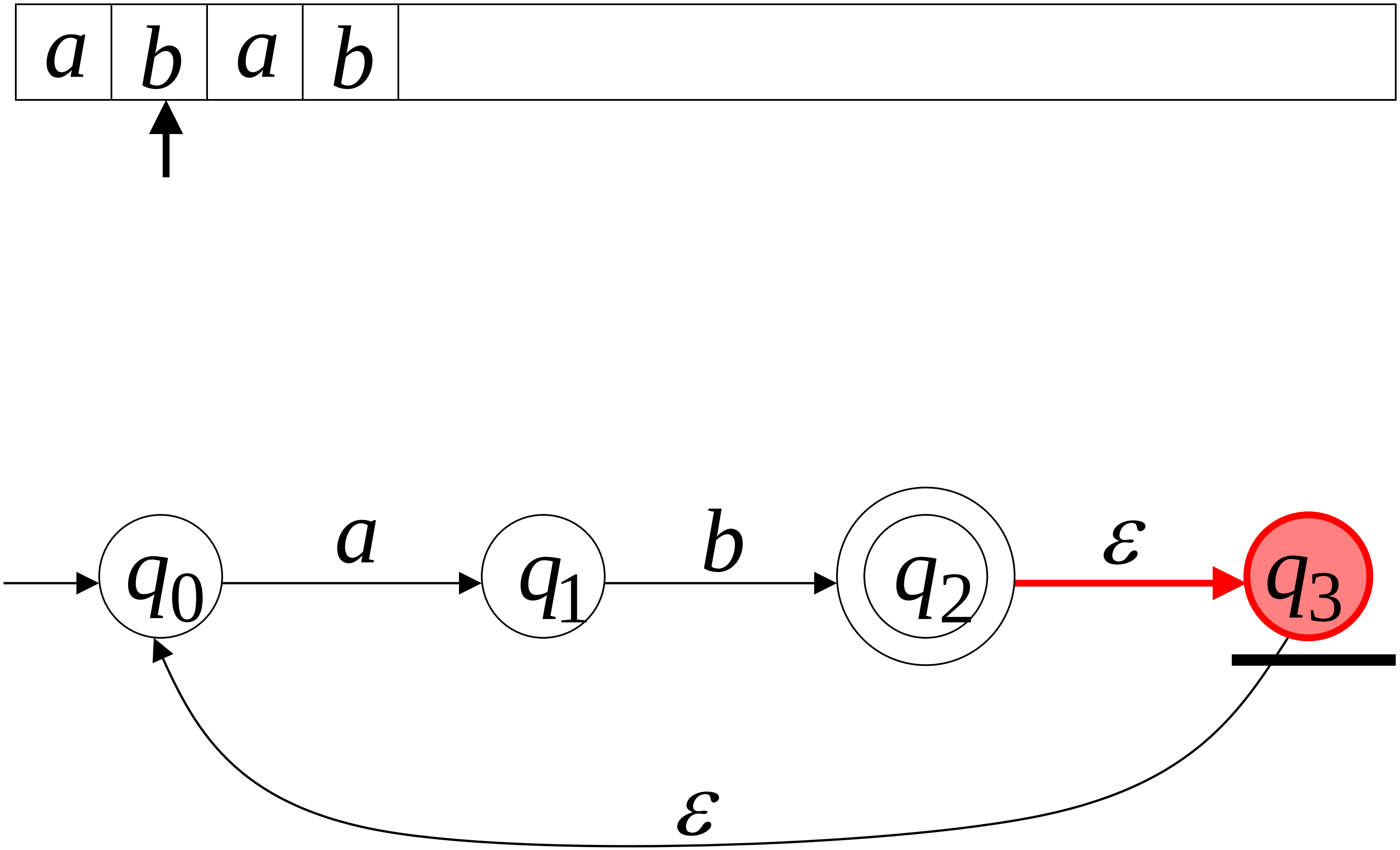




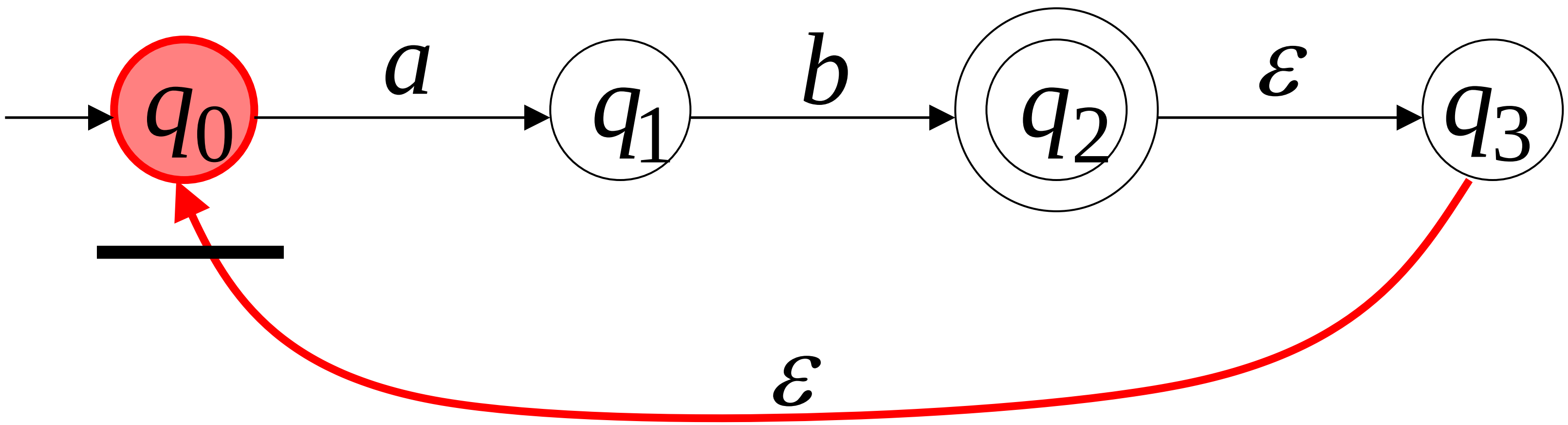
a
b
a
b
a
b
ε
ε
q_0
q_1
q_2
q_3

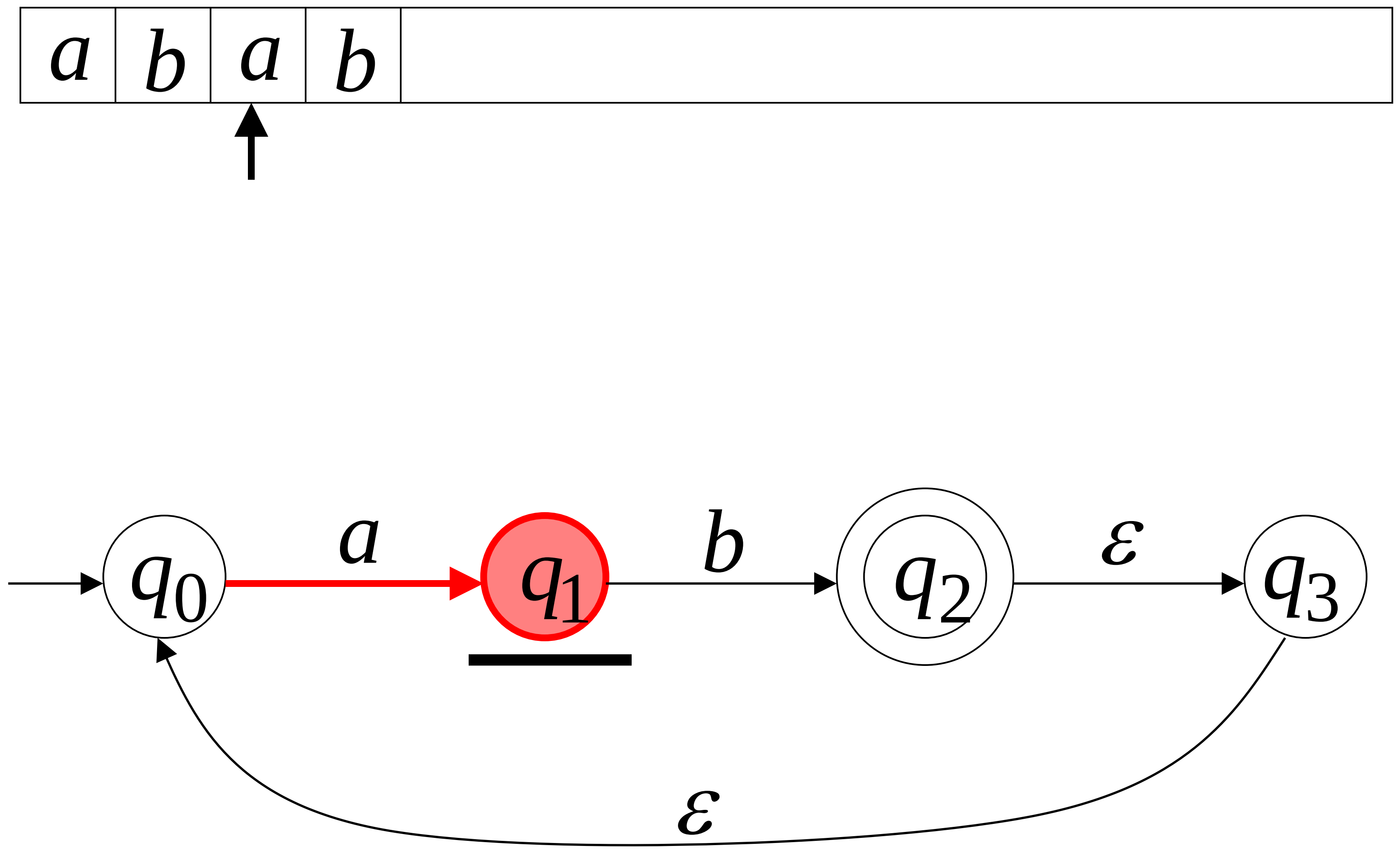


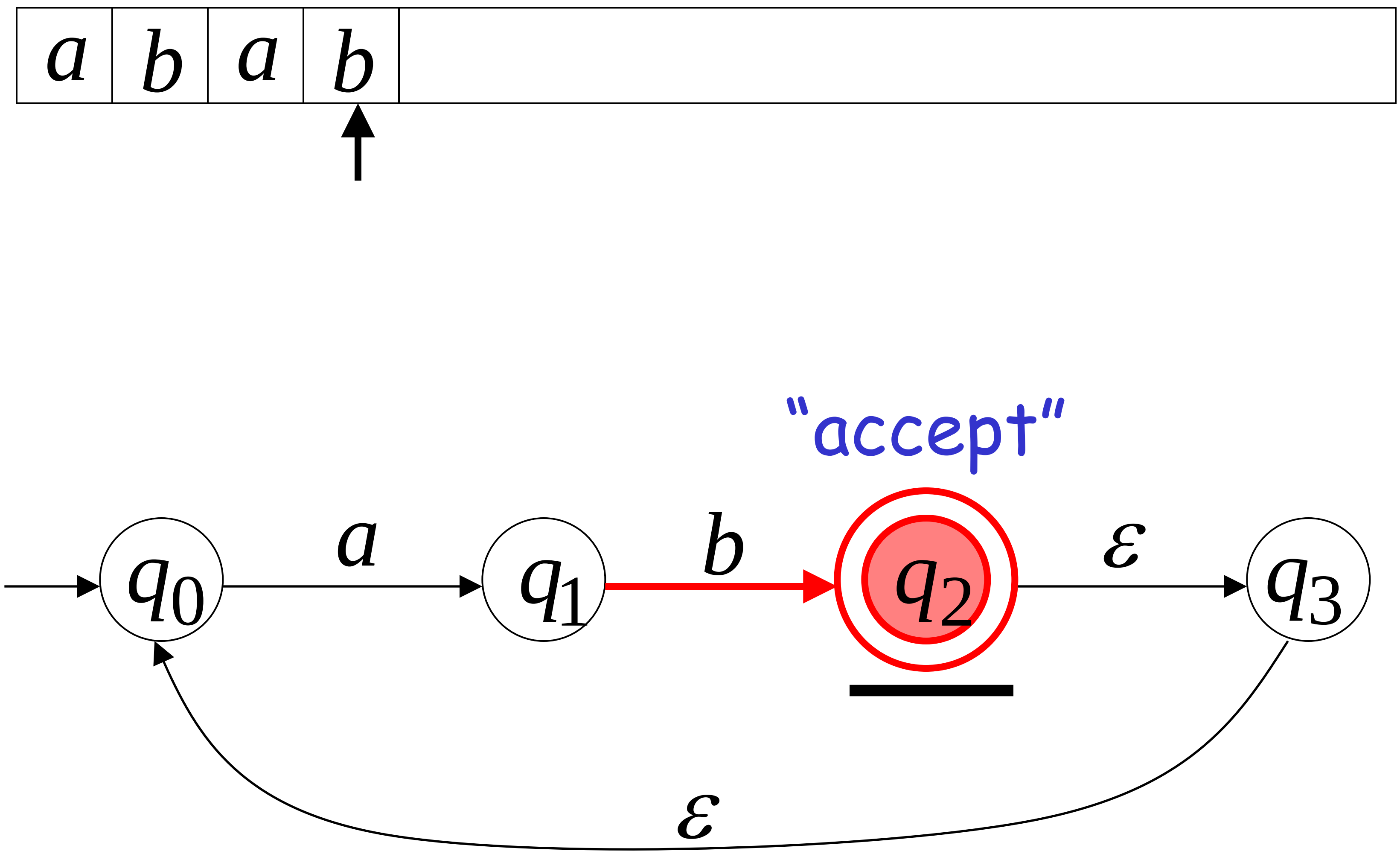
a
b
a
b
a
b
ε
ε


a
b
a
b
q0
a
q1
b
q2
ε
q3
ε


a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

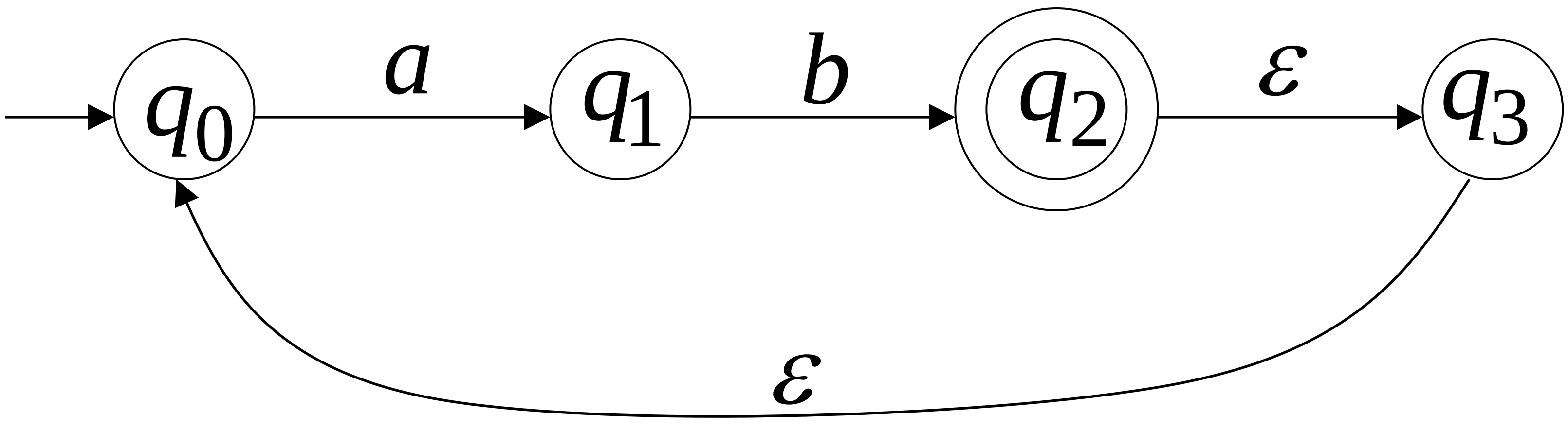

a
b
a
b
a
b
ε
ε
q_0
q_1
q_2
q_3



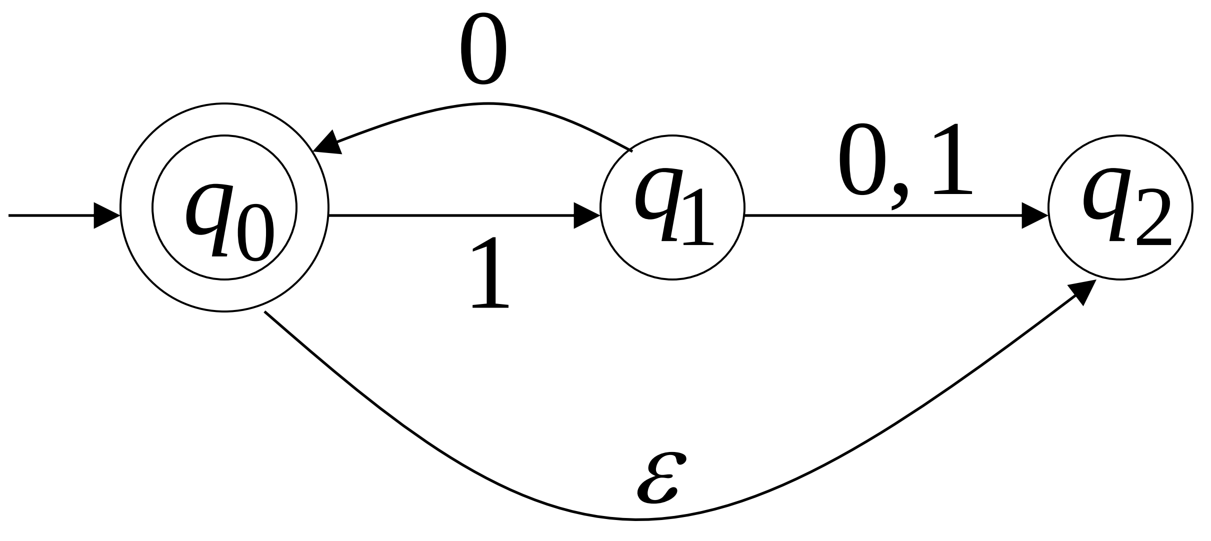
a
b
a
b
"accept"
a
b
ε
ε
q_0
q_1
q_2
q_3

Language accepted

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



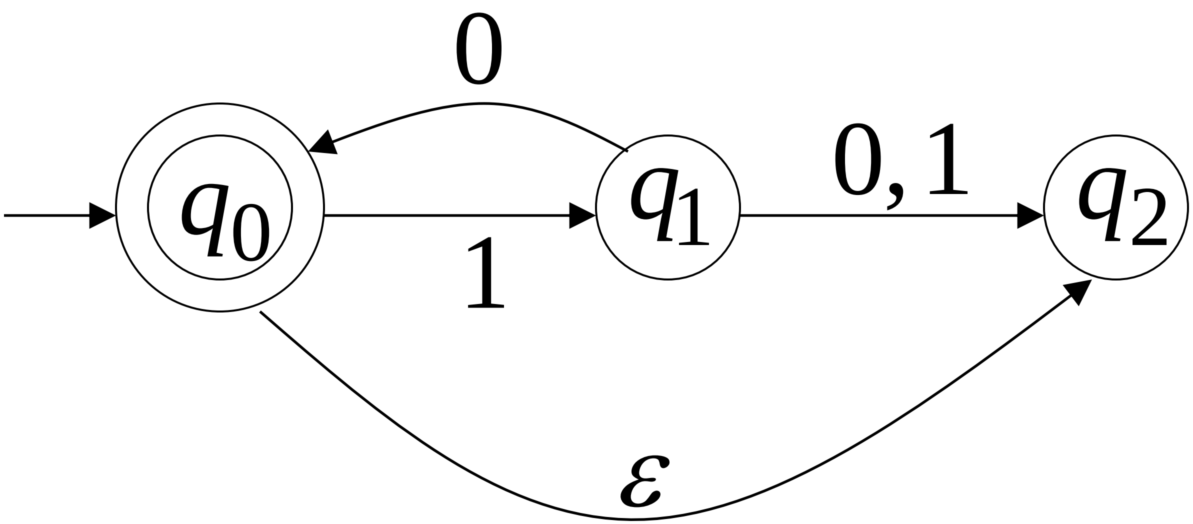
Another NFA Example



Language accepted

$$L = \{\varepsilon,\ 10,\ 1010,\ 101010,\ \ldots\}$$

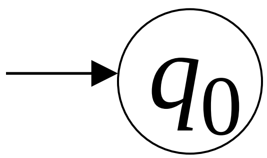
$$= \{10\}^*$$



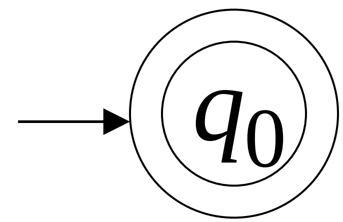
(redundant state)

Simple Automata

M_1



M_2

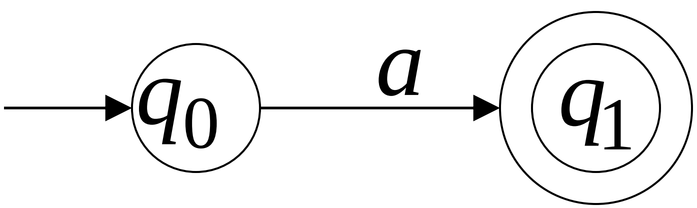


$$L(M_1) = \{\}$$

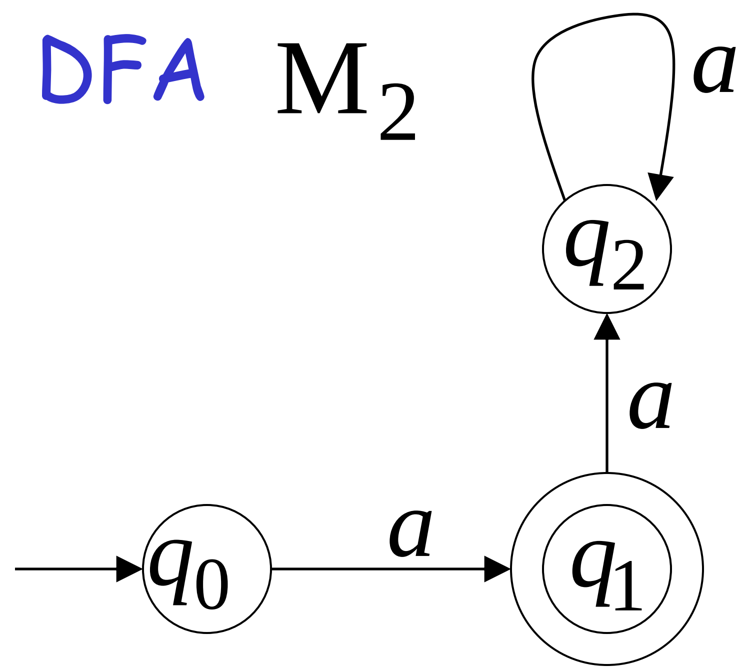
$$L(M_2) = \{\varepsilon\}$$

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. $\{a, b\}$ $\qquad \varepsilon \notin \Sigma$

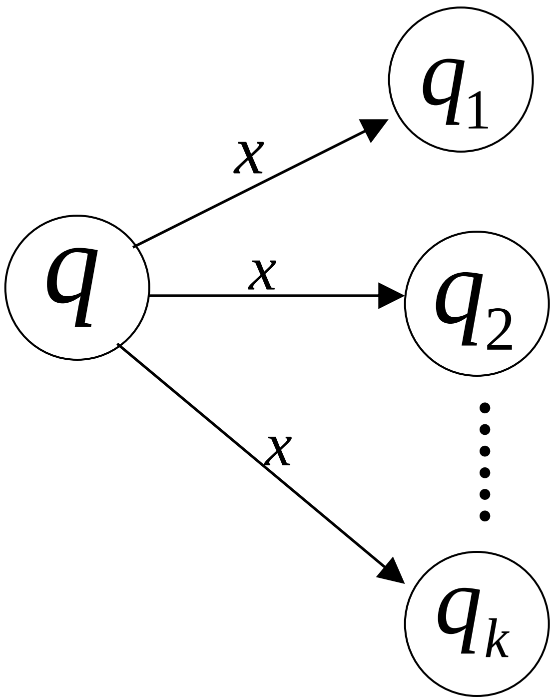
$\delta:$ Transition function

$q_0:$ Initial state

$F:$ Accepting states

Transition Function δ

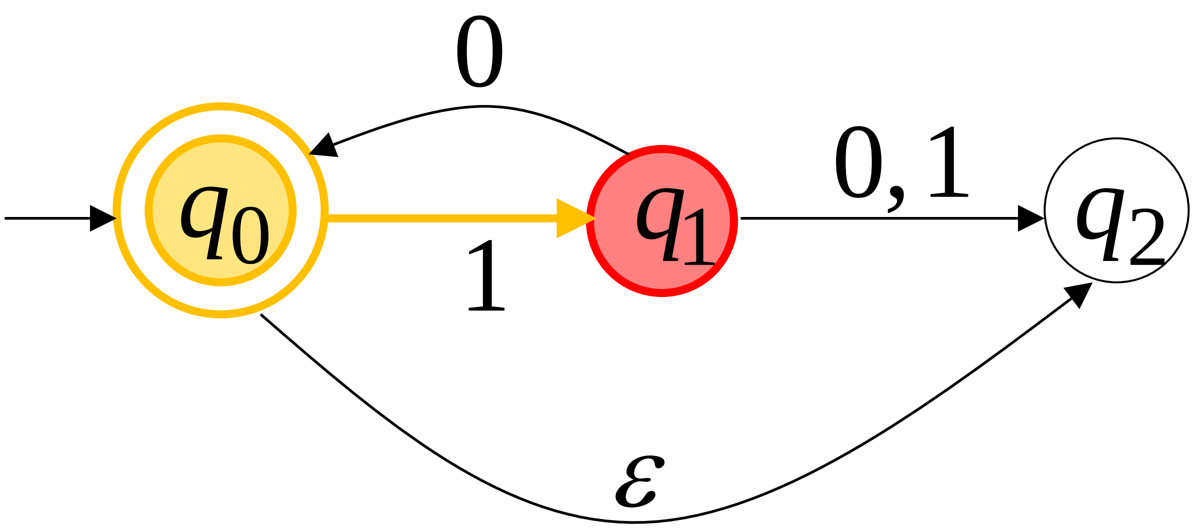
$$\delta(q, x) = \{q_1, q_2, \ldots, q_k\}$$



resulting states reached by following one transition with input symbol x

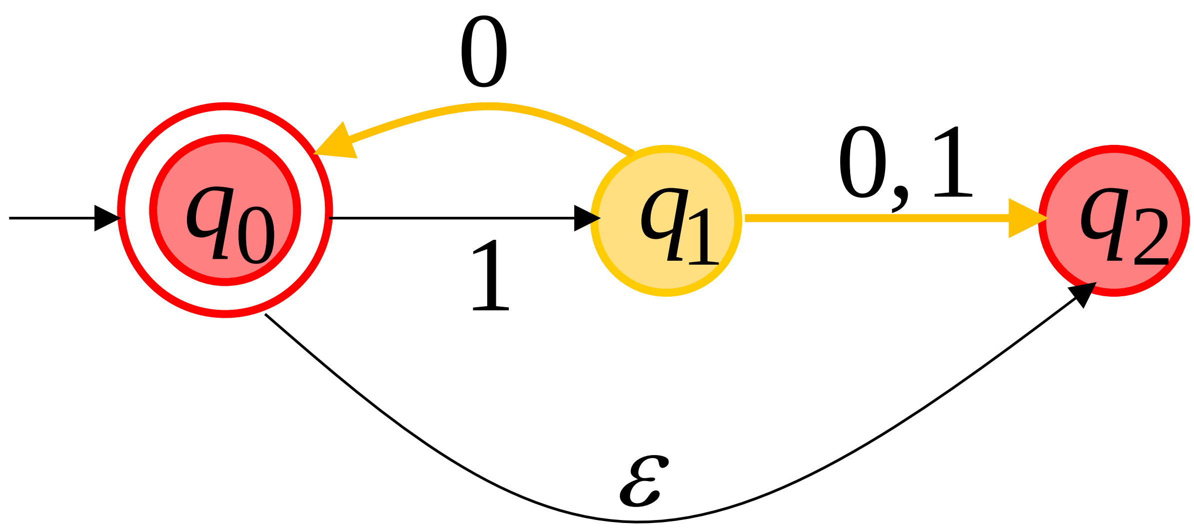
States reachable from q_0 scanning 1

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



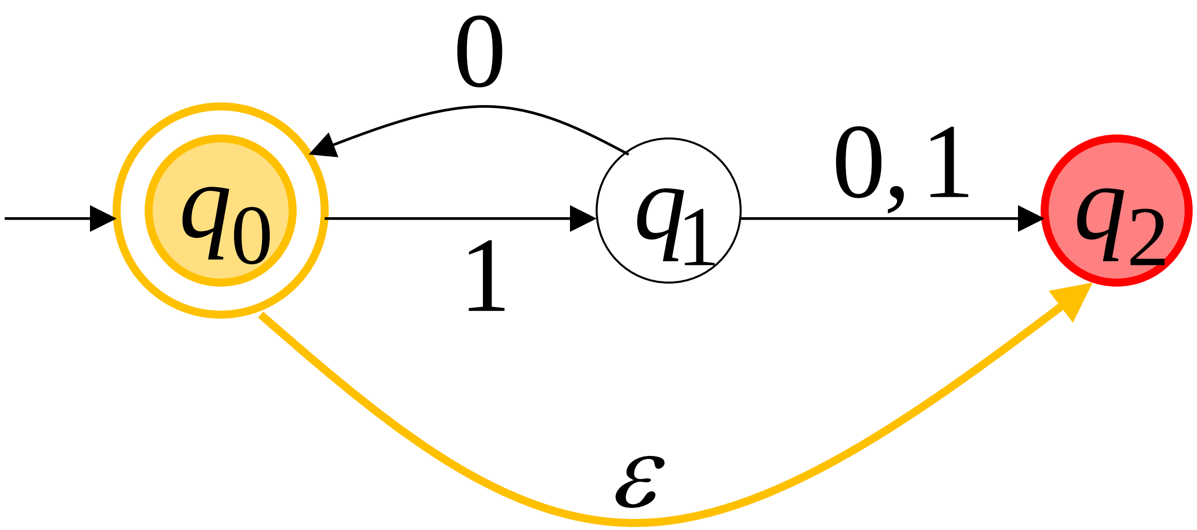
States reachable from q_1 scanning 0

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



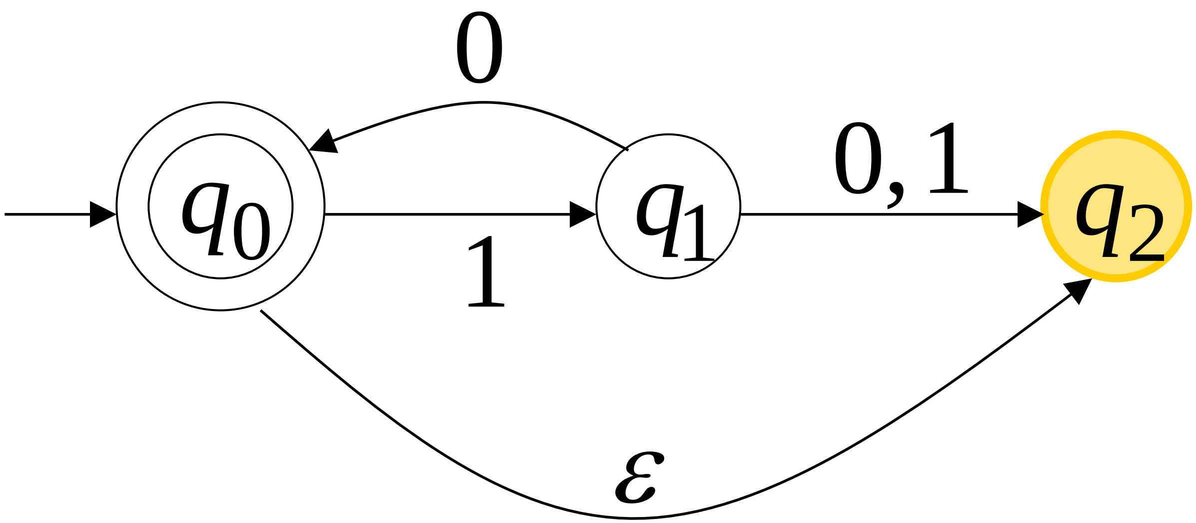
States reachable from q_0 with one transition scanning no input symbol

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



States reachable from q_2 scanning 1

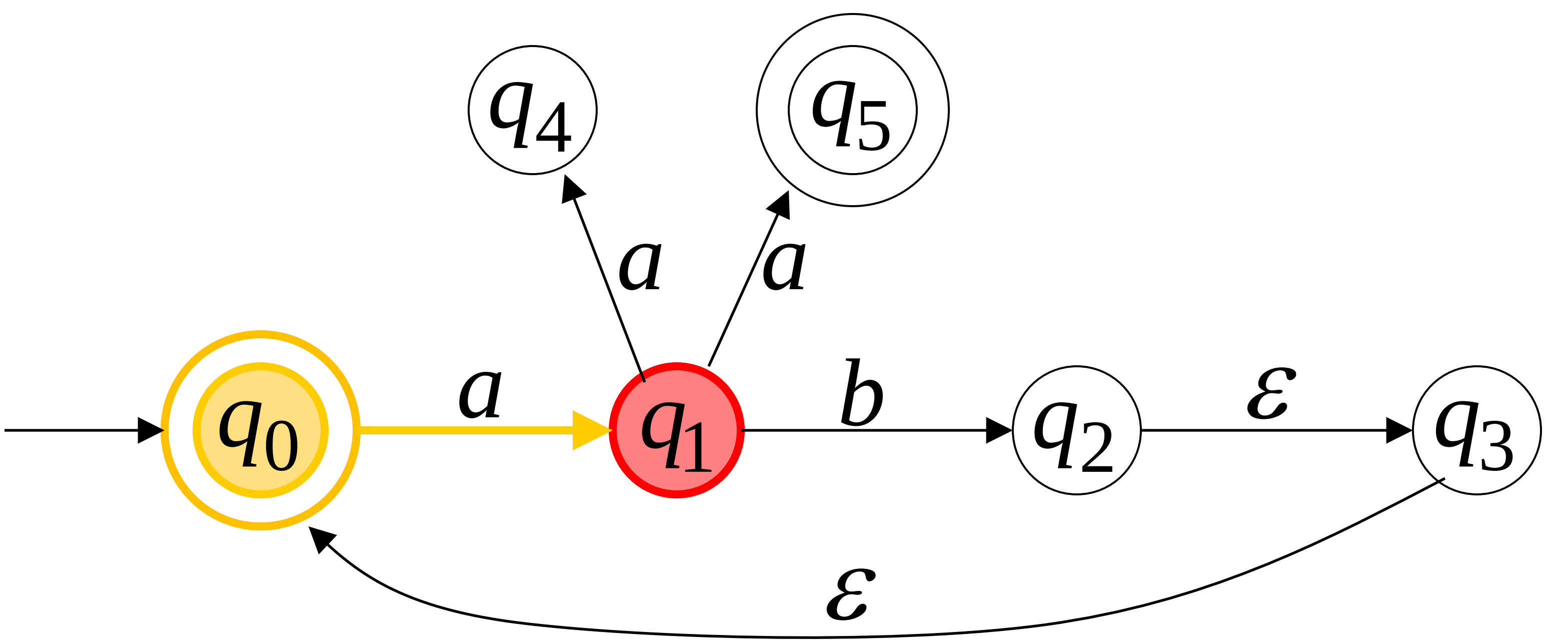
$$\delta(q_2, 1) = \emptyset$$



Extended Transition Function δ^*

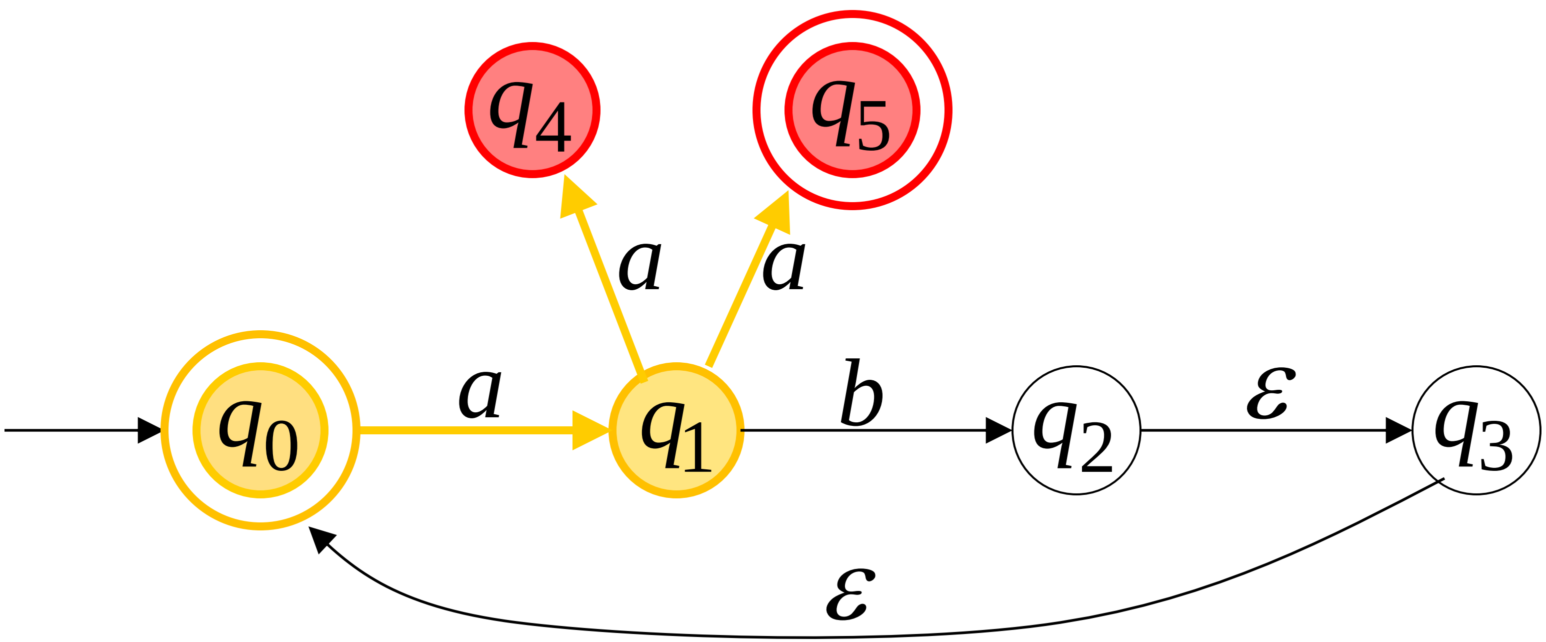
Similar with δ but applied on strings

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



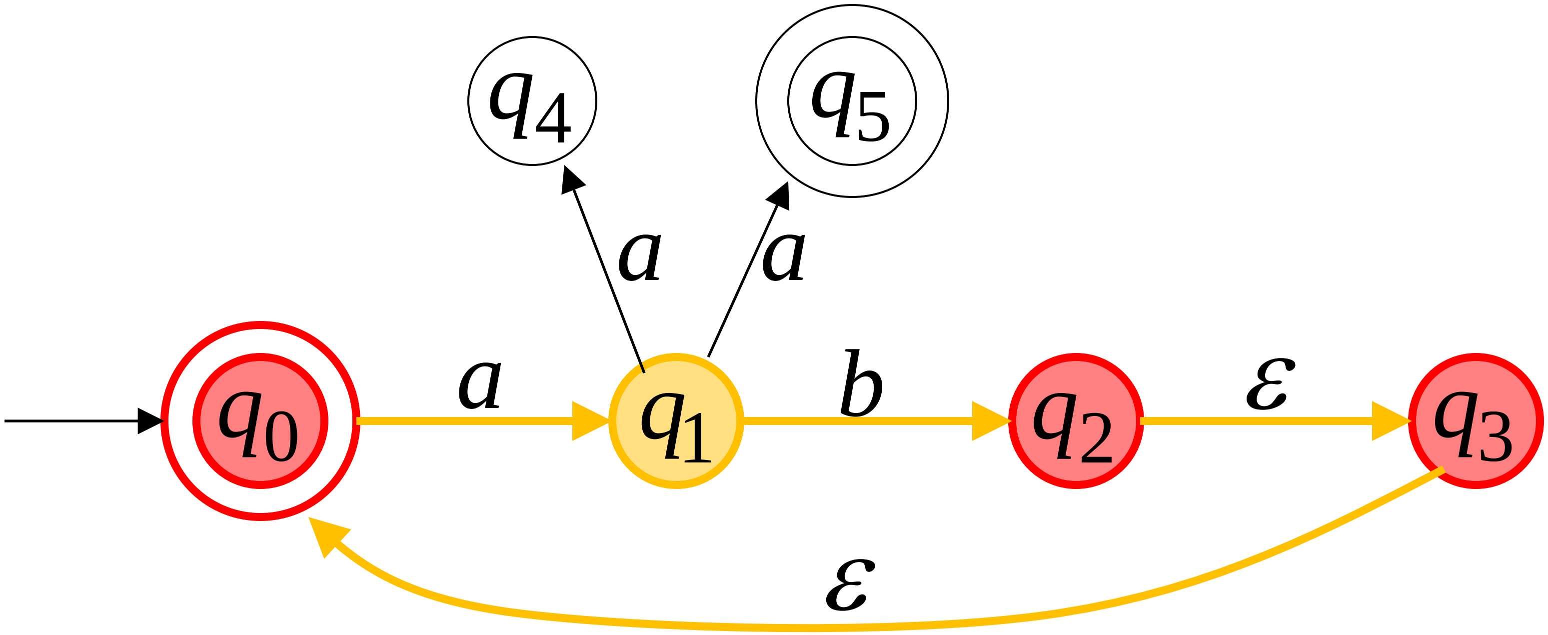
States reachable from q_0 scanning aa

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



States reachable from q_0 scanning ab

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



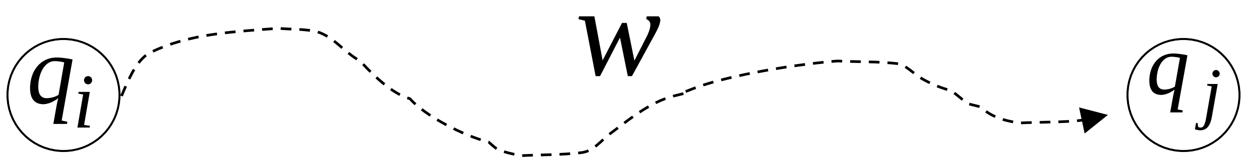
Special case:

for any state q

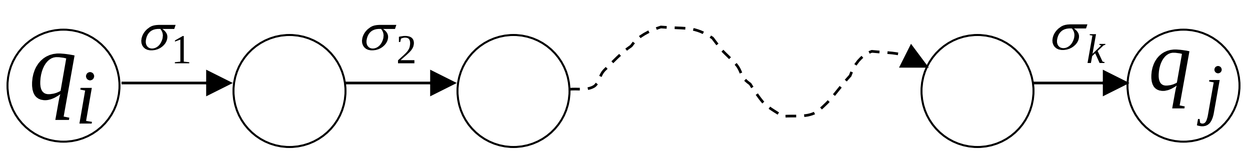
$$q \in \delta^*(q, \varepsilon)$$

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:

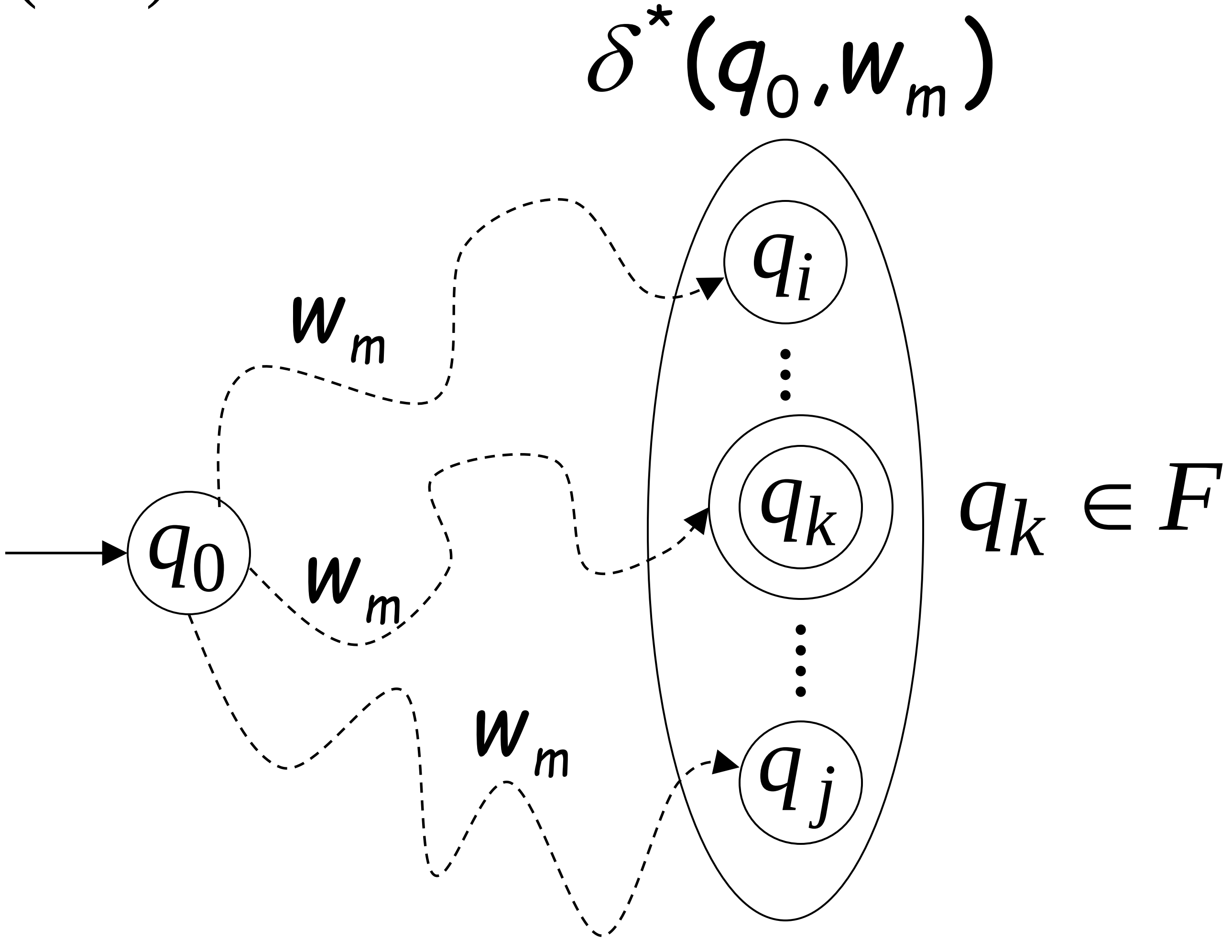
$$L(M) = \{w_1, w_2, \ldots, w_n\}$$

Where for each w_m

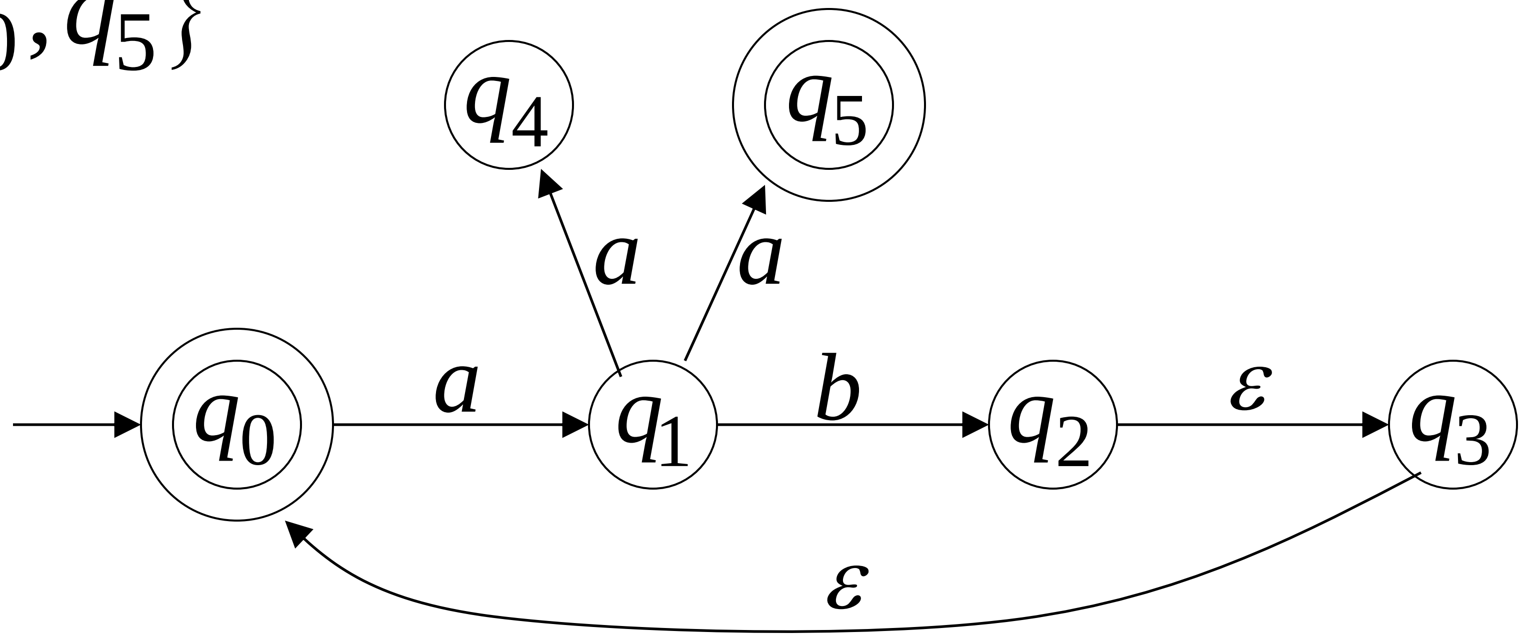
$$\delta^*(q_0, w_m) = \{q_i, \ldots, q_k, \ldots, q_j\}$$

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

$w_m \in L(M)$



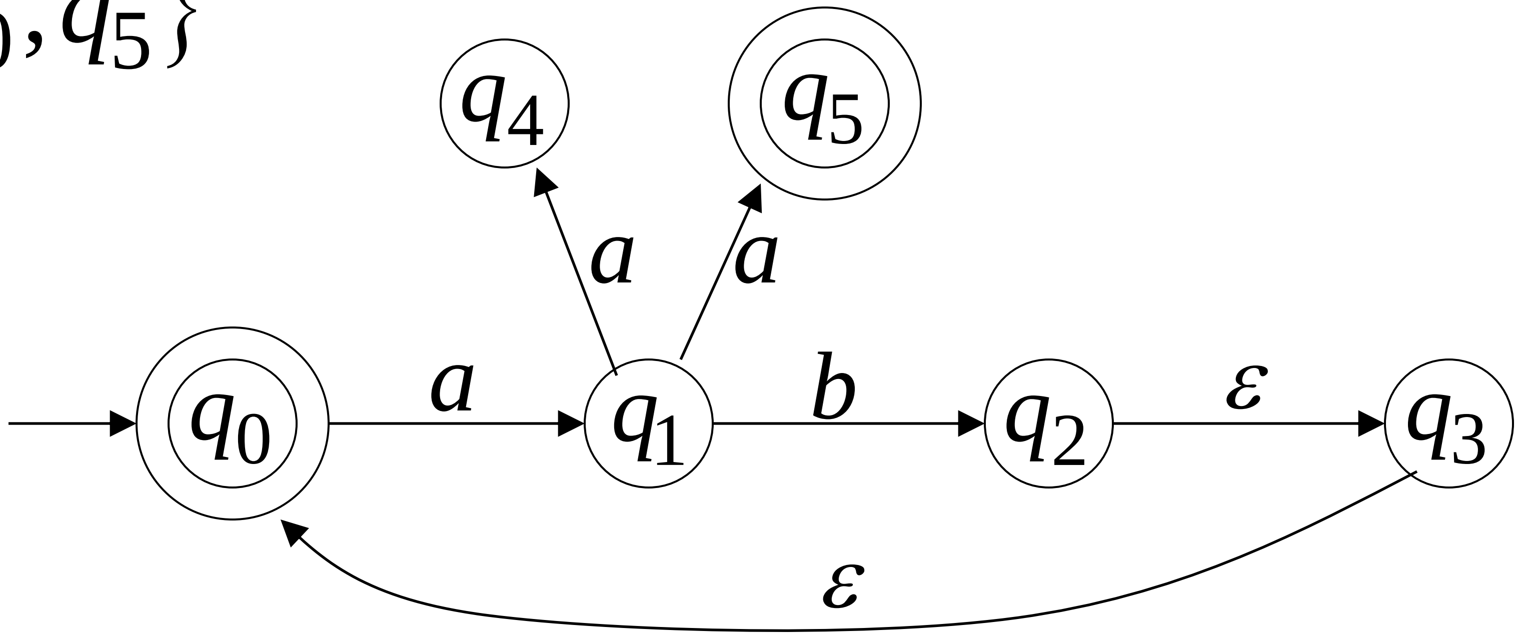
$$F = \{q_0, q_5\}$$



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

$\in F$

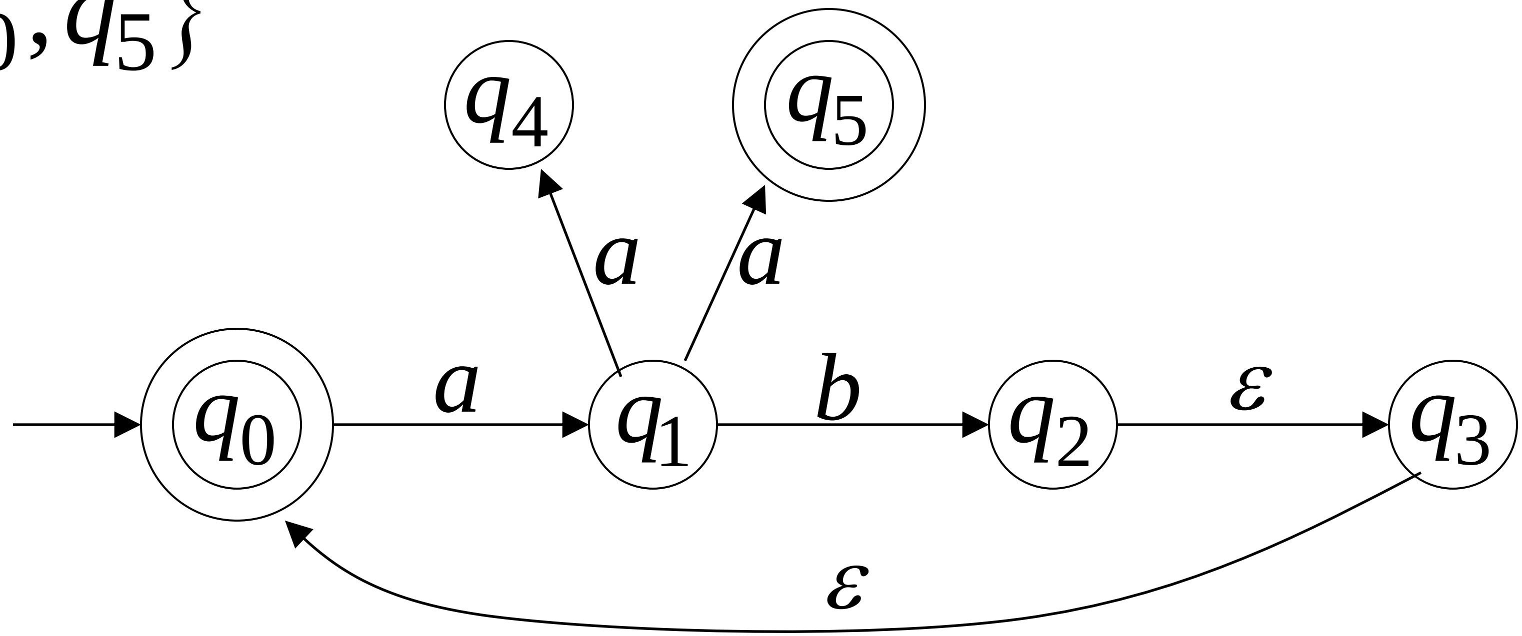
$$F = \{q_0, q_5\}$$



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

$$\in F$$

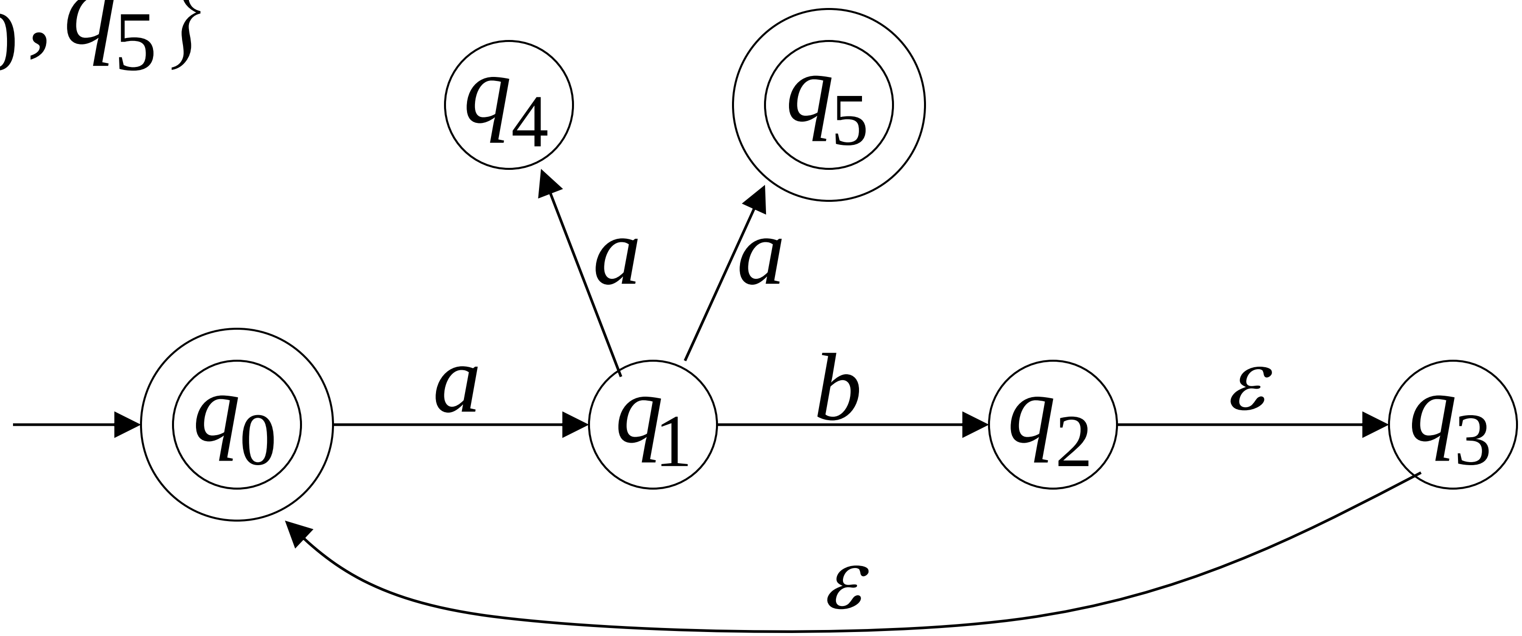
$F = \{q_0, q_5\}$



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

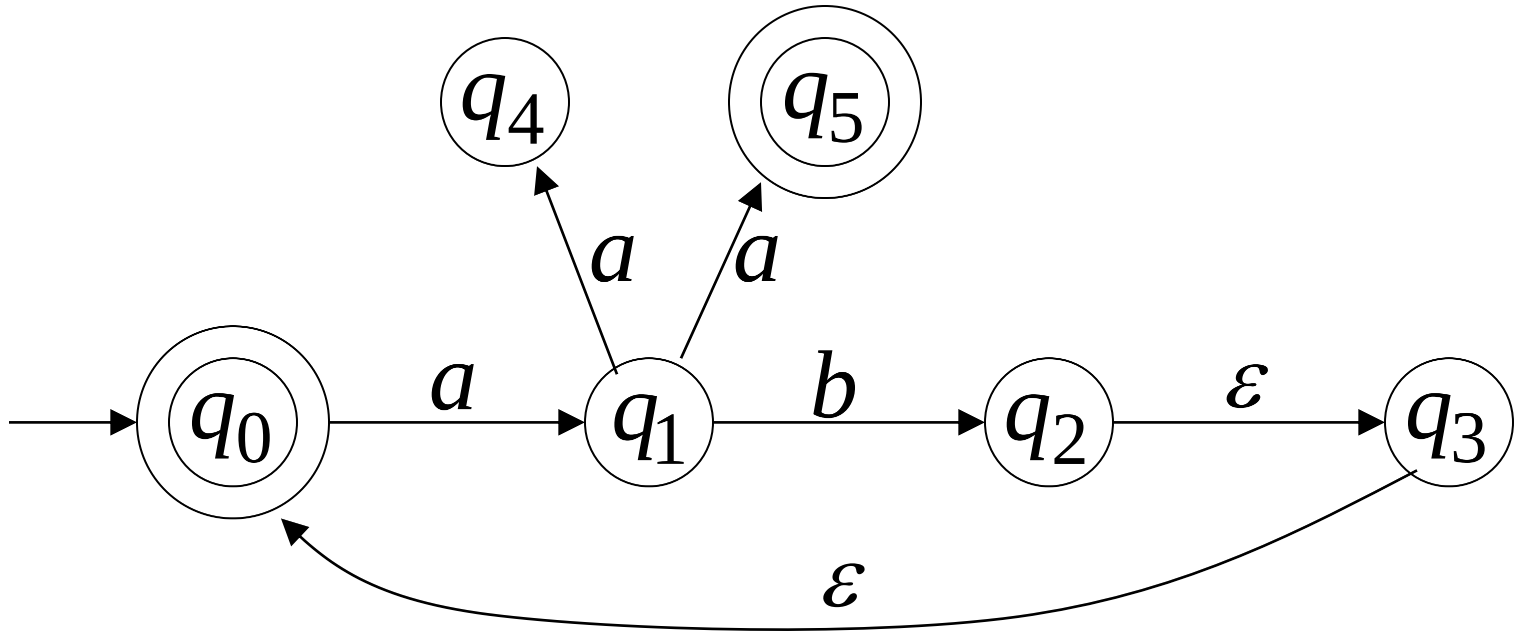
$$\in F$$

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



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

$$\{q_1\} \notin F$$



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

NFAs accept the Regular Languages

Equivalence of Machines

- Next session

Equivalence of Machines

Definition:

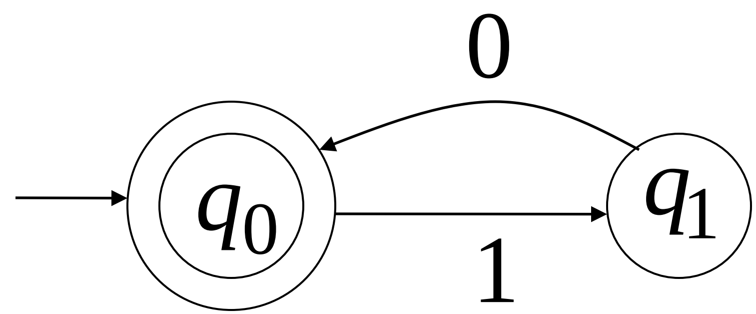
Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

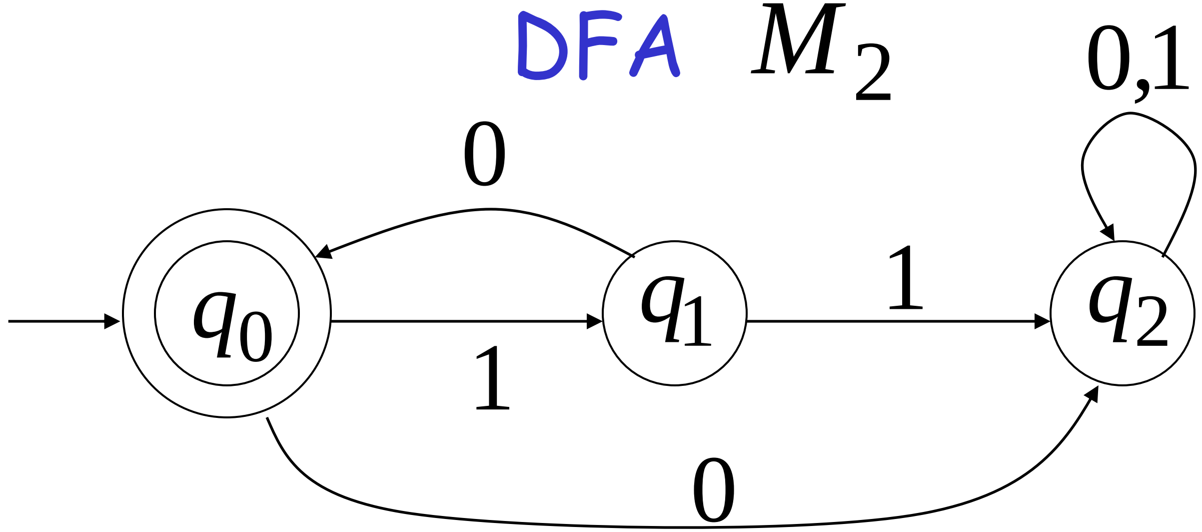
NFA M_1

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



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, namely, they accept the same set of languages

Proof: we need to show

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

AND

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

Proof-Step 1

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

Every DFA is trivially a NFA

$$\Downarrow$$

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

Proof-Step 2

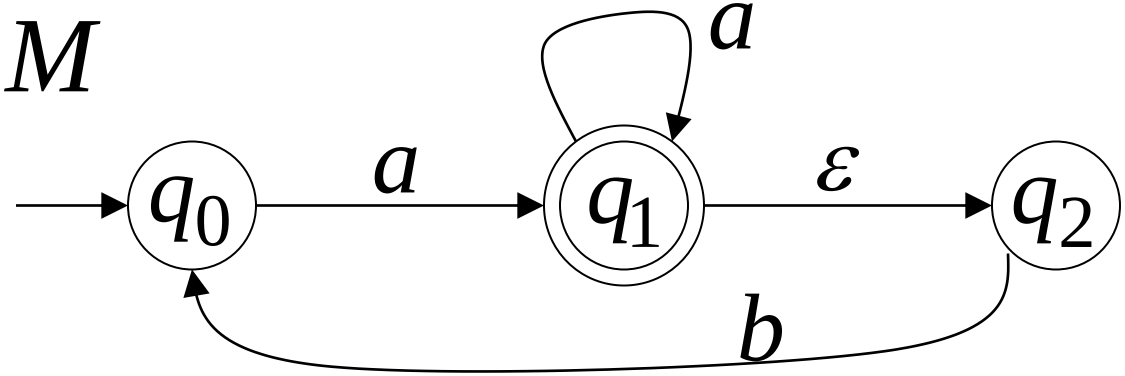
$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Any NFA can be converted to an equivalent DFA

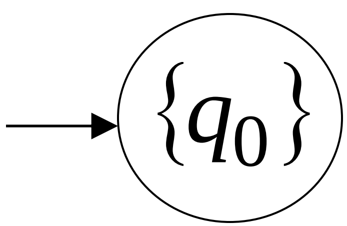
Any language L accepted by a NFA is also accepted by a DFA

Conversion of NFA to DFA

NFA M

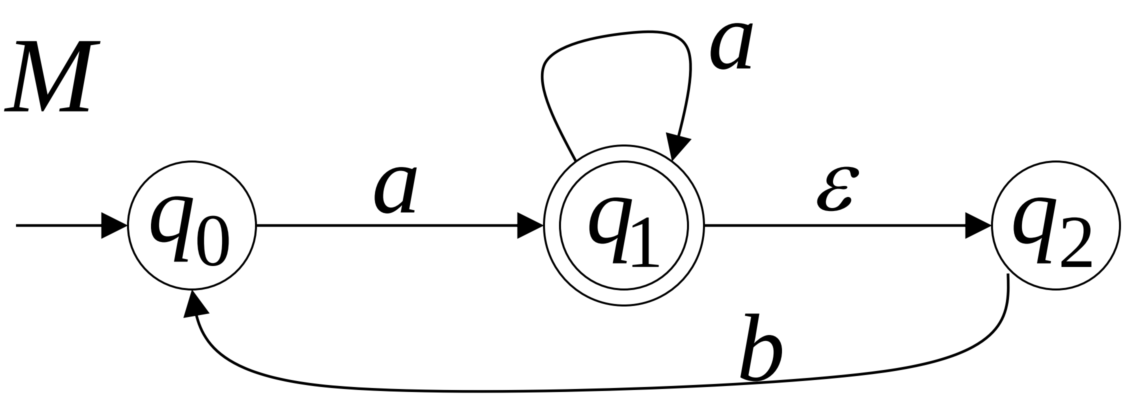


DFA M'

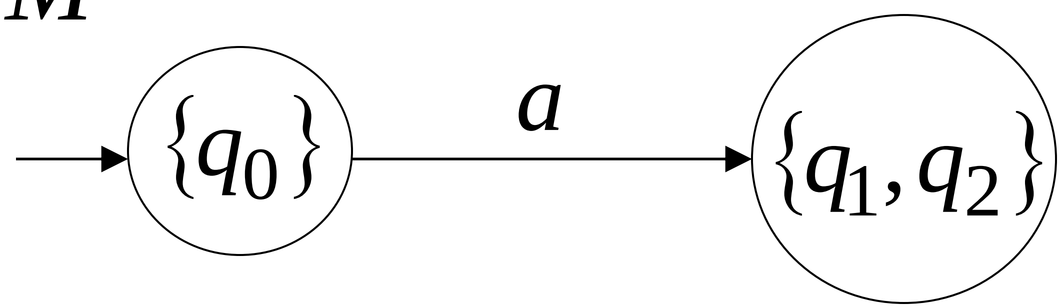


$$\delta^*(q_0, a) = \{q_1, q_2\}$$

NFA M

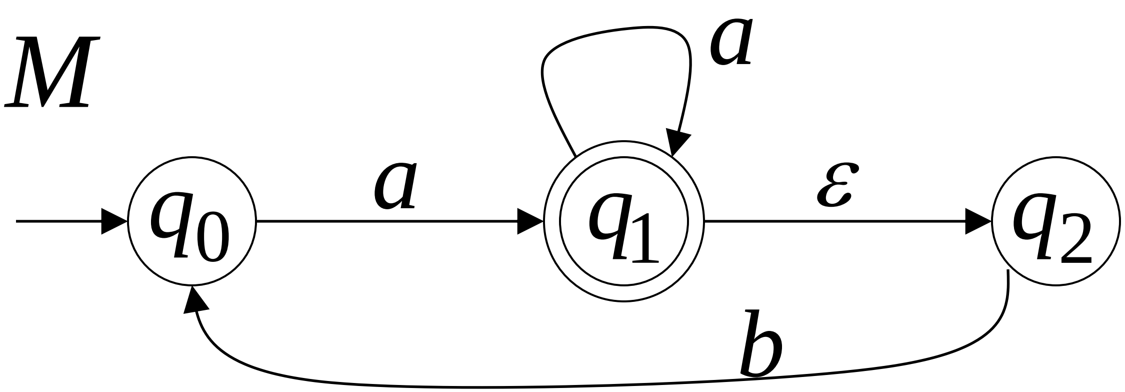


DFA M'

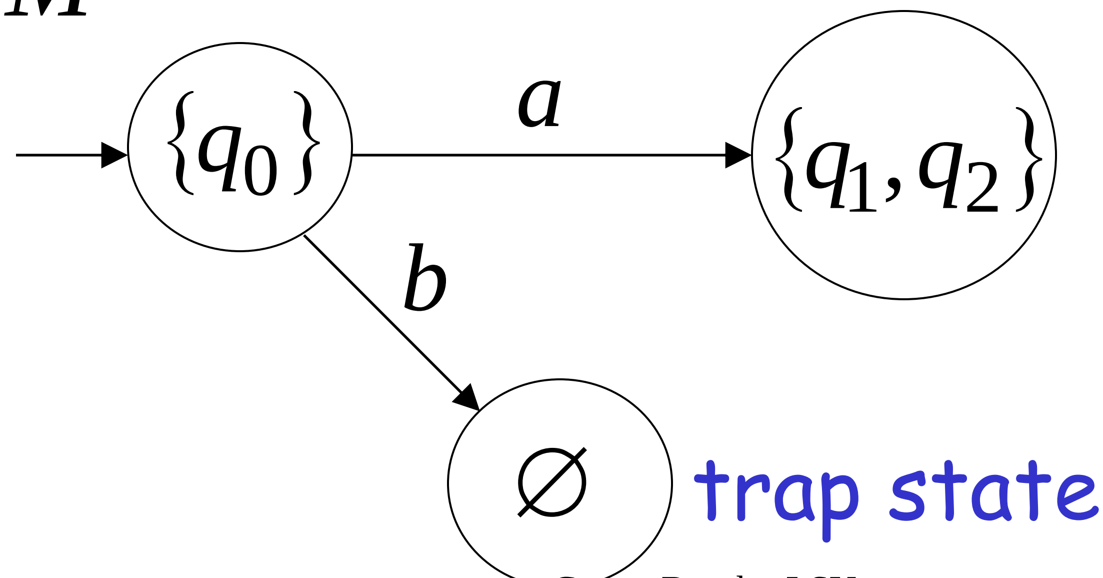


$$\delta^*(q_0, b) = \varnothing \quad \text{empty set}$$

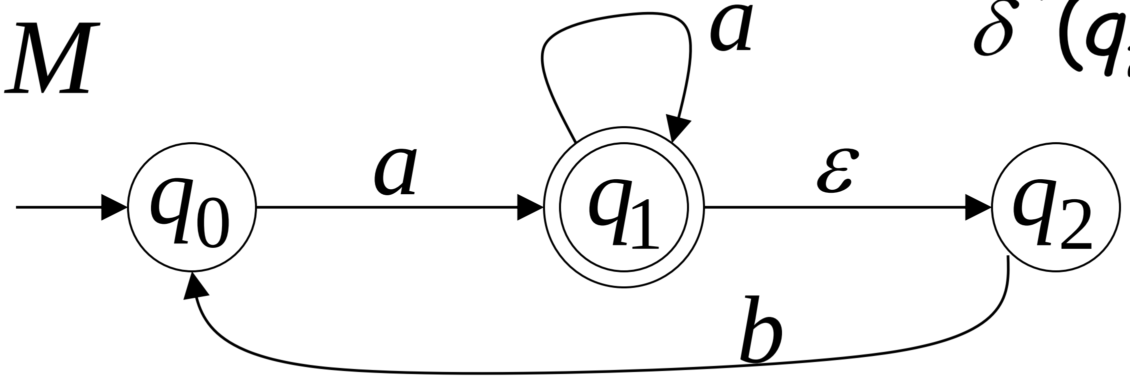
NFA M



DFA M'



NFA M



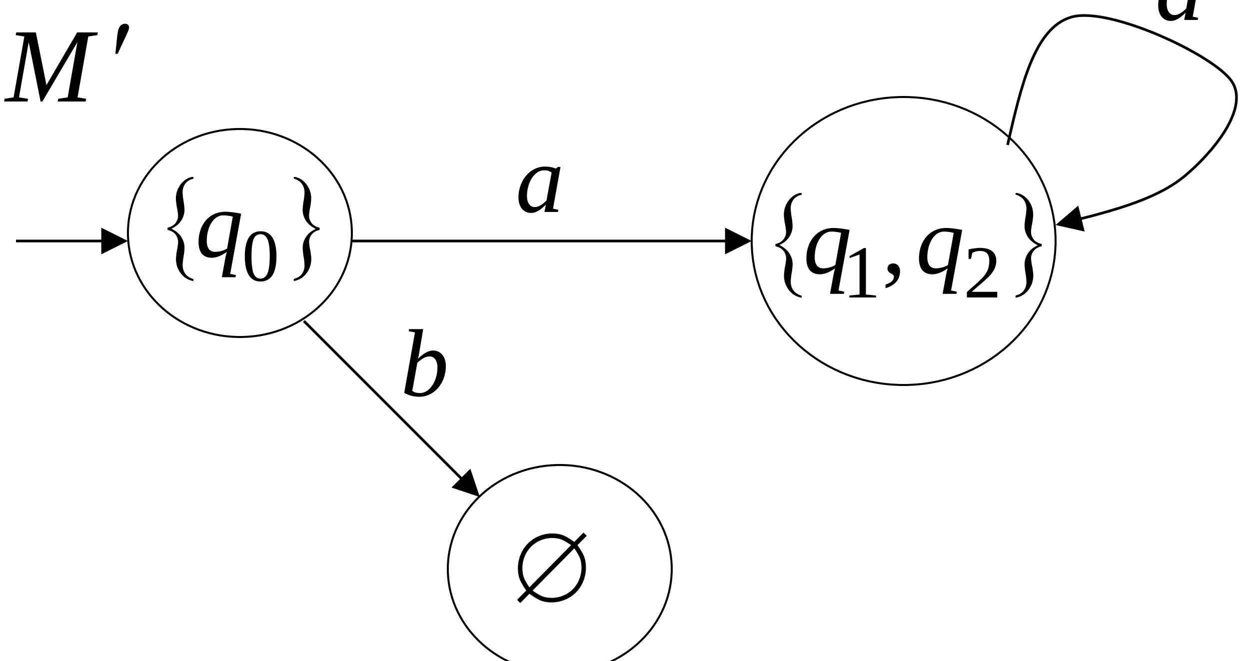
$\delta^*(q_1, a) = \{q_1, q_2\}$

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

union

$\{q_1, q_2\}$

DFA M'



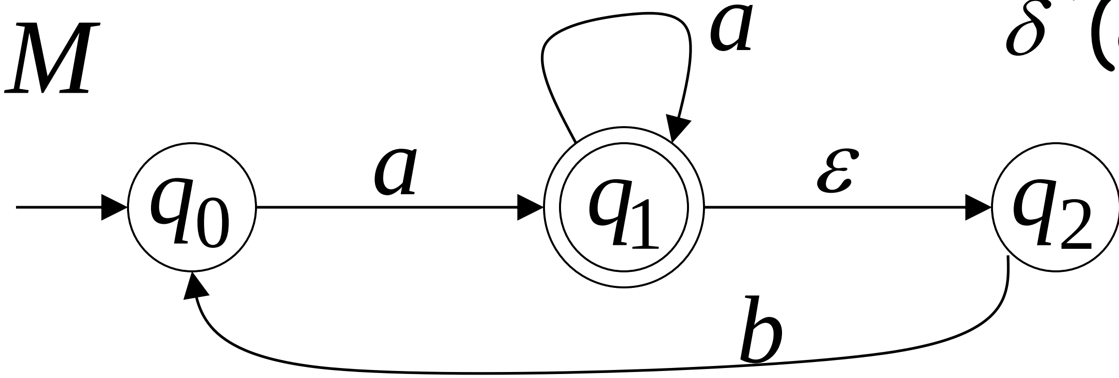
$\delta^*(q_1, b) = \{q_0\}$

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

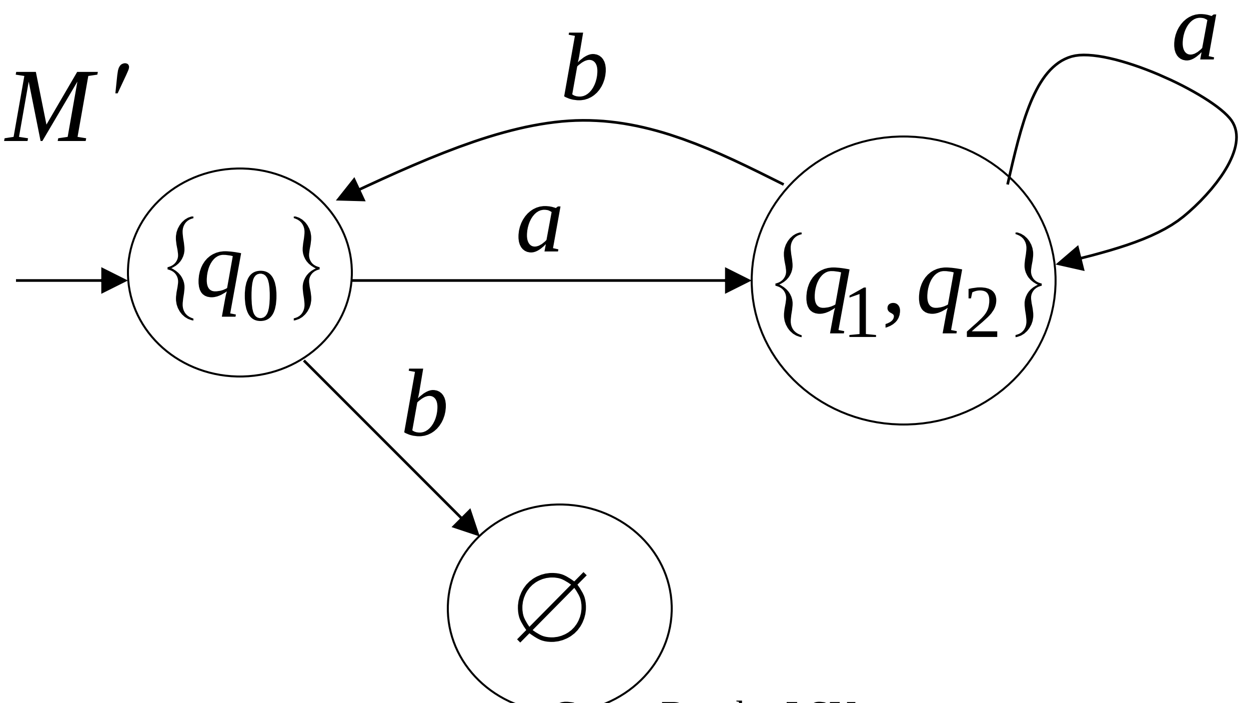
union

$\{q_0\}$

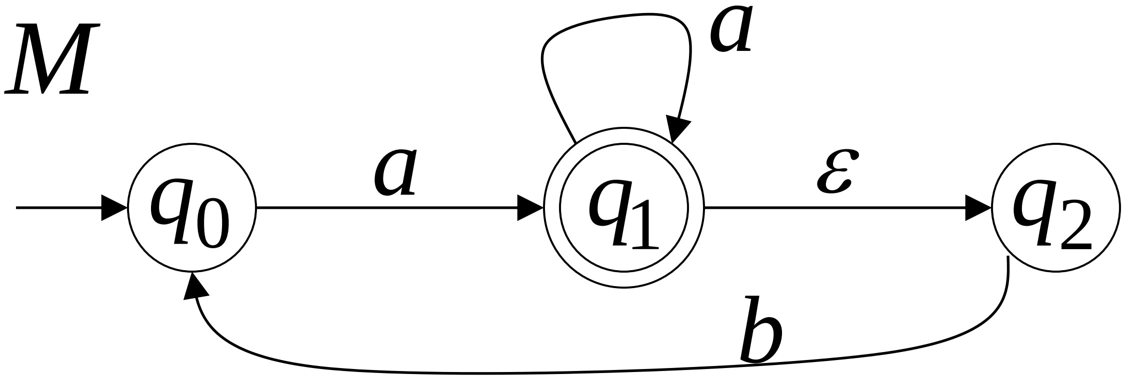
NFA M



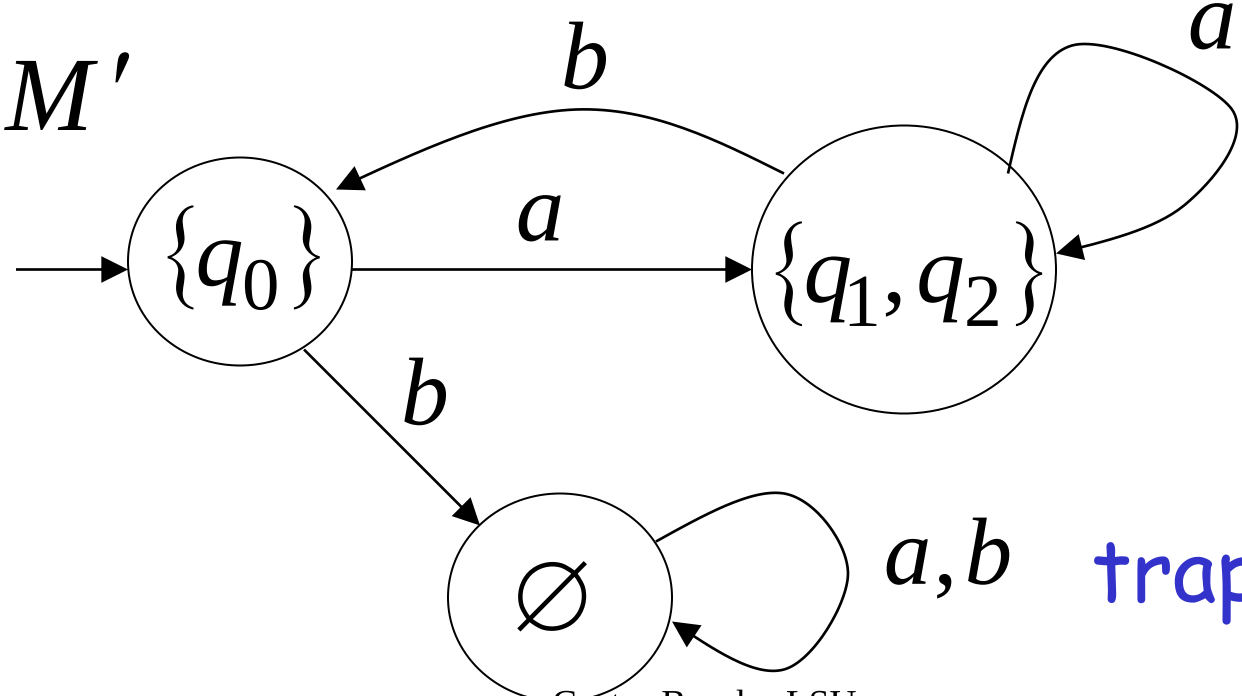
DFA M'



NFA M



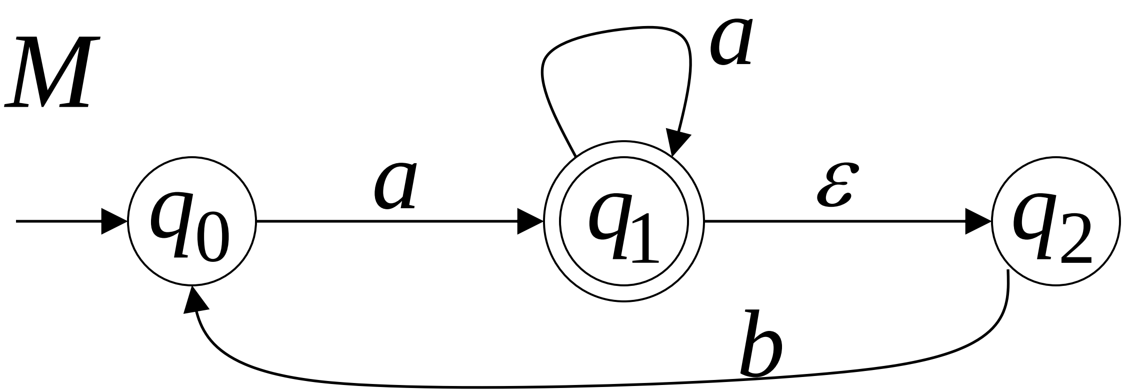
DFA M'



trap state

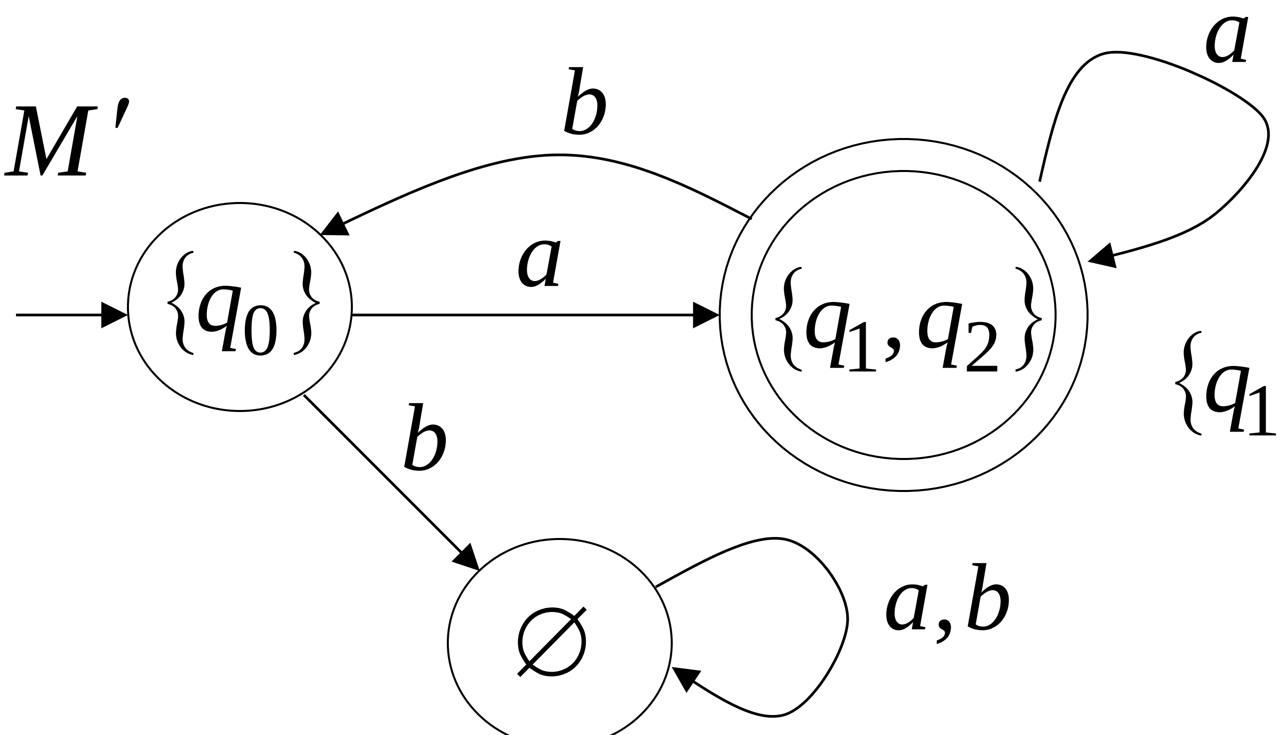
END OF CONSTRUCTION

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

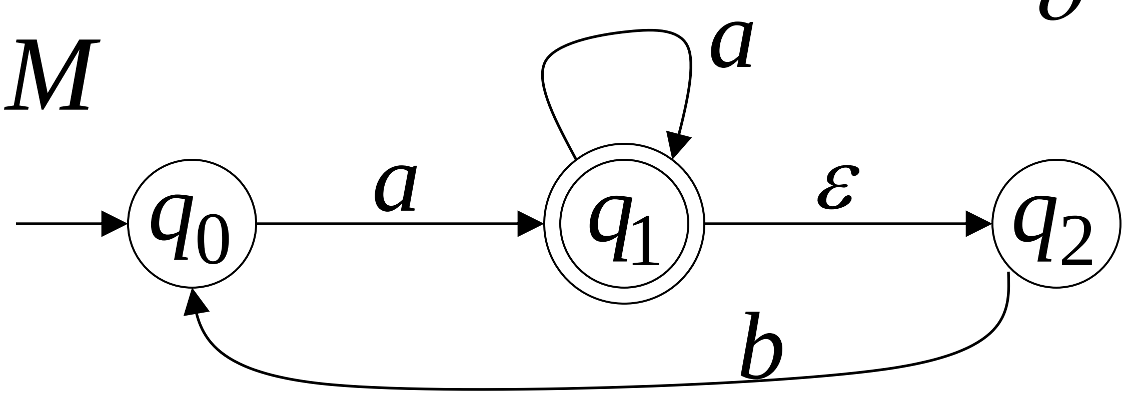
$$\delta^*(q_0, \varepsilon) = \{q_0, \ldots\}$$

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

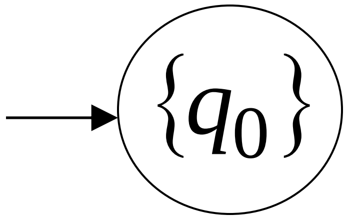
Example

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

NFA M



DFA M'



step

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

compute in the NFA

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

Union

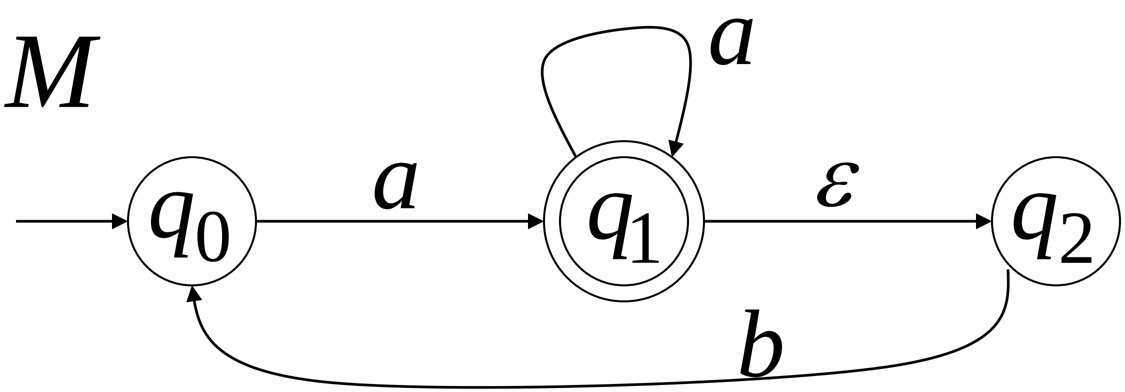
add transition to DFA

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

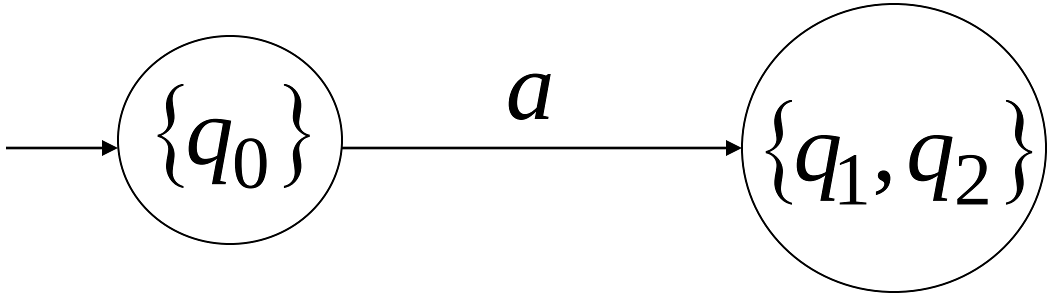
Example

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

NFA M



DFA M'



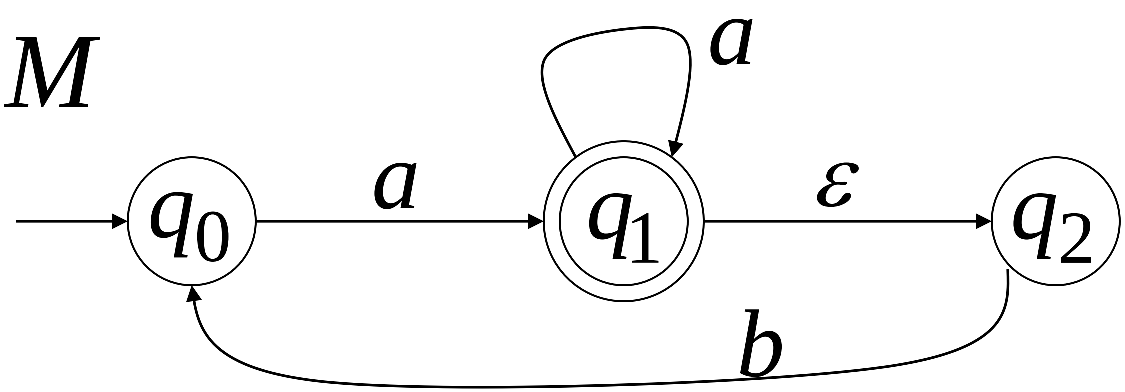
$$\delta(\{q_0\}, a) = \{q_1, q_2\}$$

step

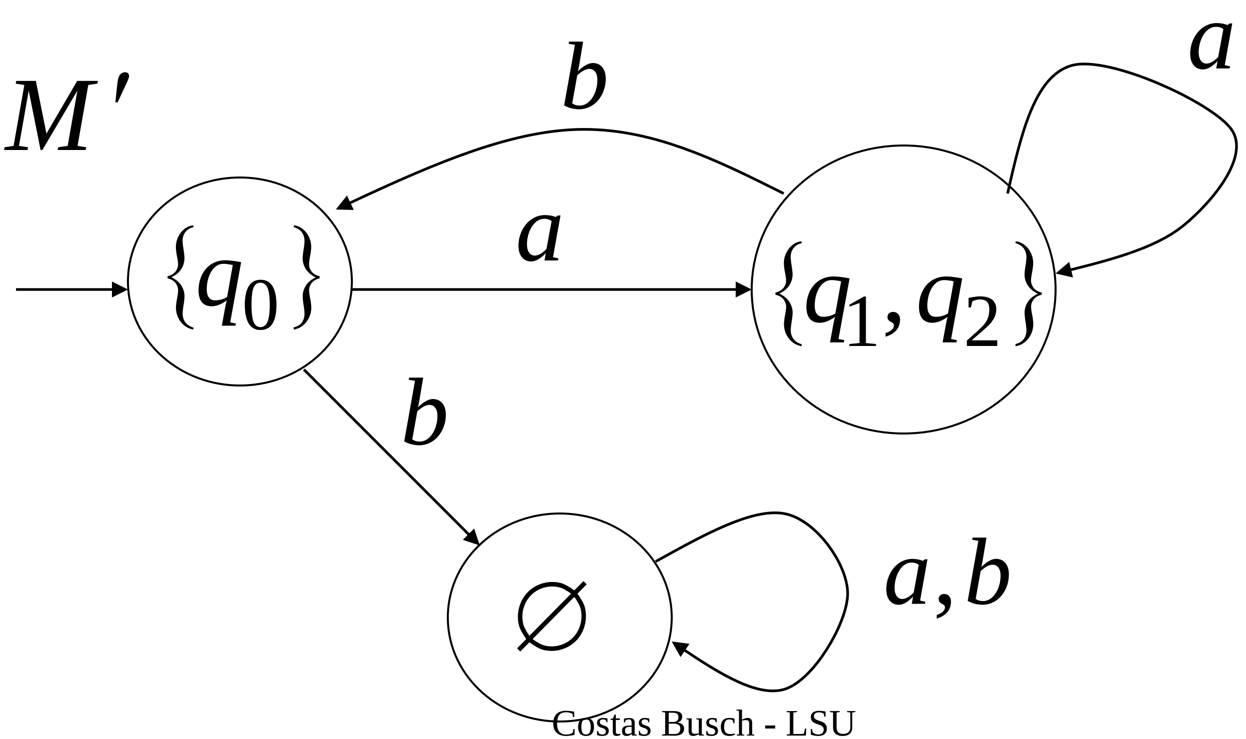
3. Repeat Step 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA

Example

NFA M



DFA M'



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

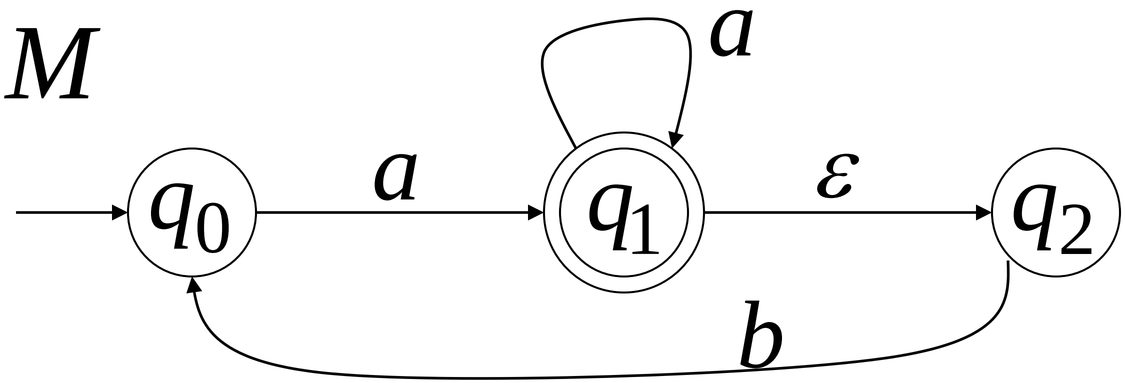
if some q_j is accepting state in NFA

Then, $\{q_i, q_j, \ldots, q_m\}$

is accepting state in DFA

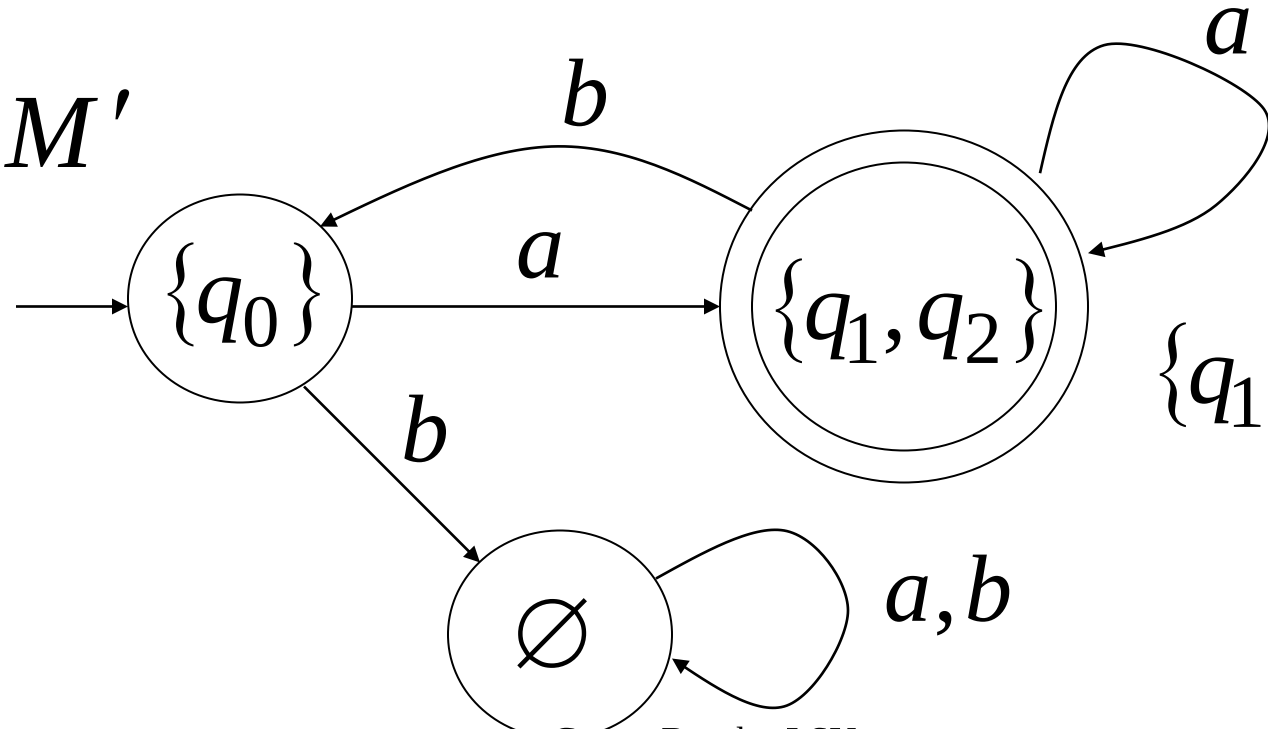
Example

NFA M



$q_1 \in F$

DFA M'



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

Lemma:

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

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

Proof:

We need to show: $L(M) \subseteq L(M')$

AND

$$L(M) \supseteq L(M')$$

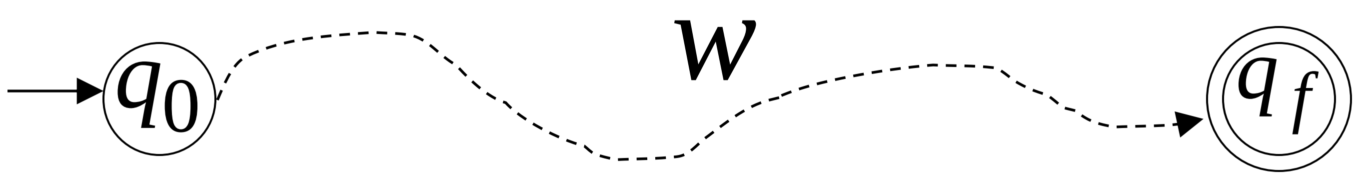
First we show: $L(M) \subseteq L(M')$

We only need to prove:

$$w \in L(M) \quad \Rightarrow \quad w \in L(M')$$

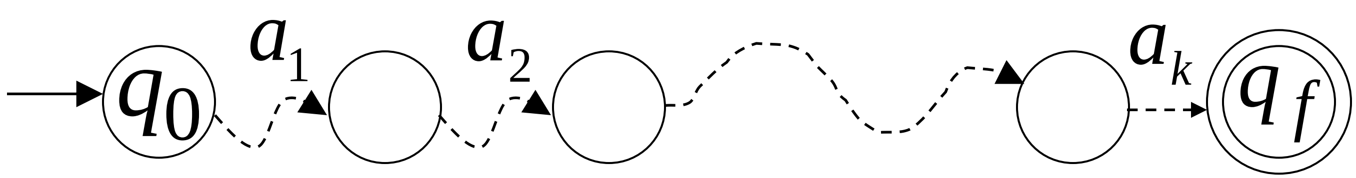
NFA

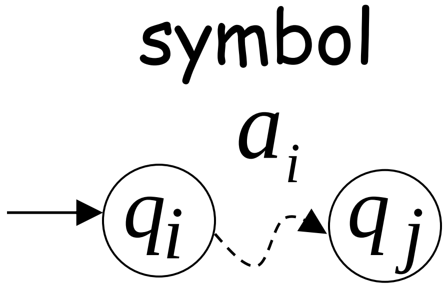
Consider $w \in L(M)$



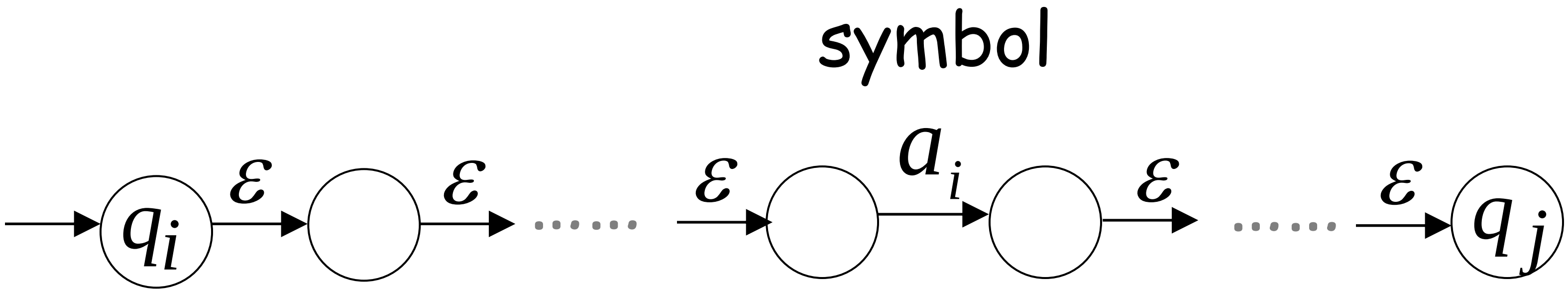
symbols

$$w = a_1 a_2 \cdots a_k$$



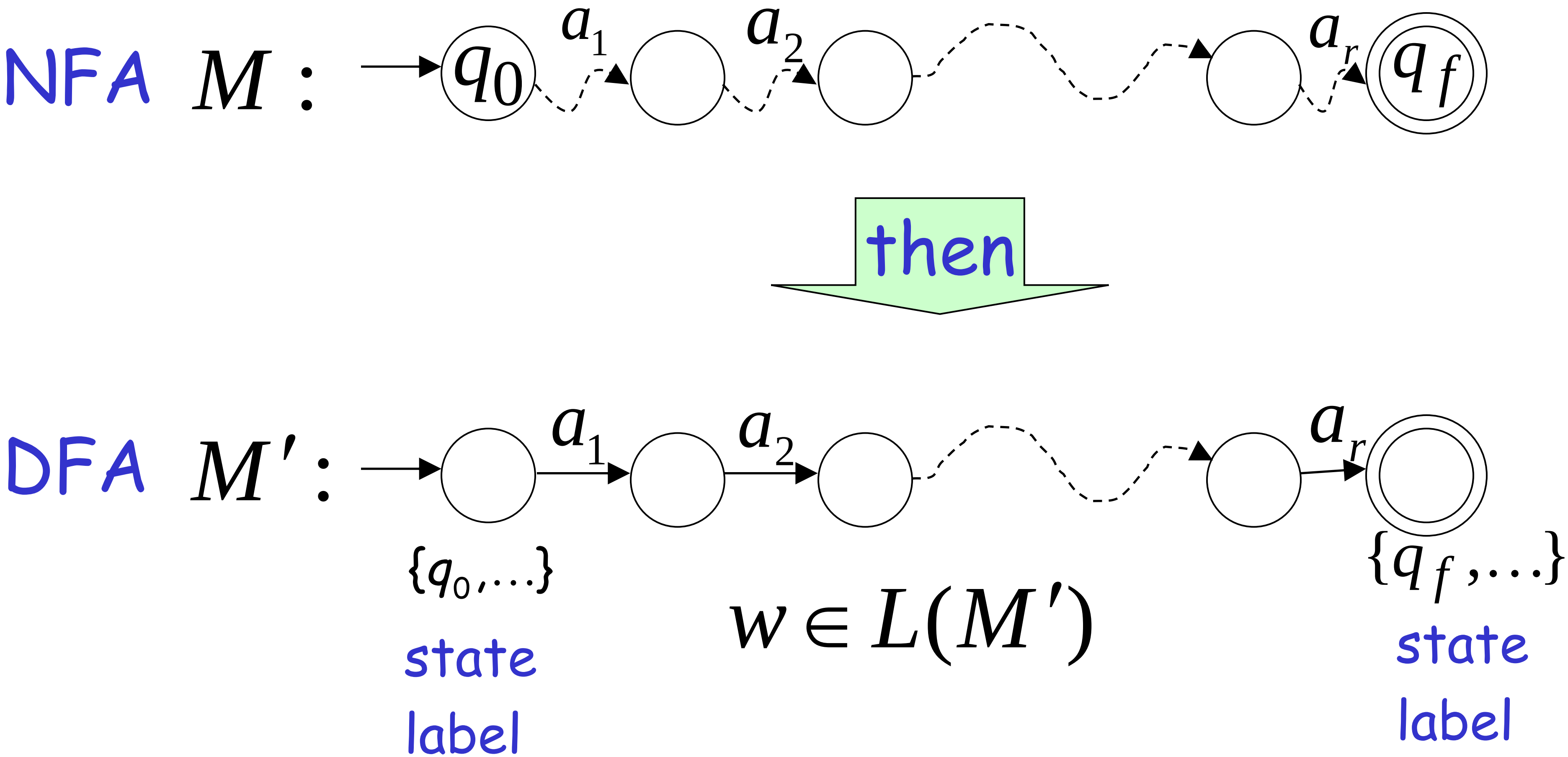


denotes a possible sub-path like



We will show that if $w \in L(M)$

$w = a_1 a_2 \cdots a_r$

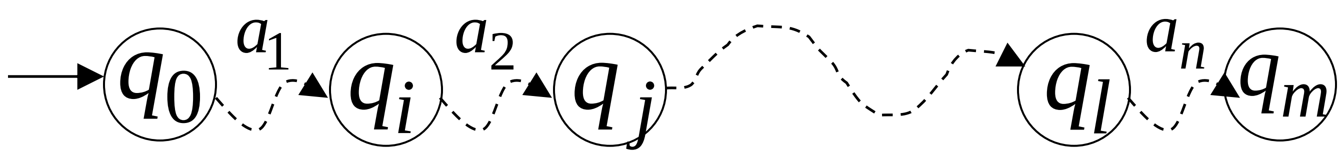


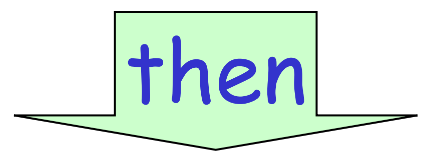
$w \in L(M')$

More generally, we will show that if in M:

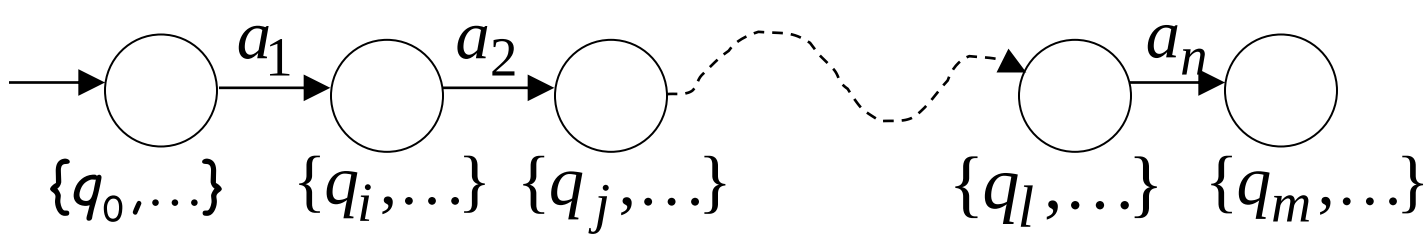
(arbitrary prefix) $v = a_1 a_2 \cdots a_n$ $\quad n \leq r$

NFA $M:$





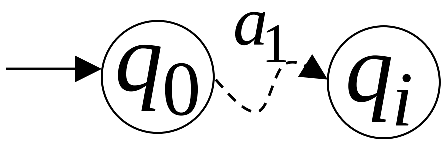
DFA $M':$



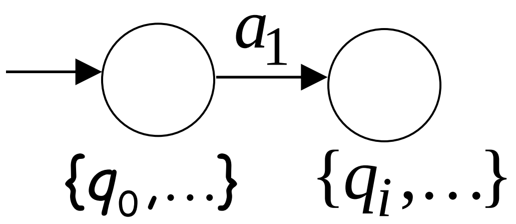
Proof by induction on $|v|$

Induction Basis: $|v| = 1$ $\qquad v = a_1$

NFA $M:$



DFA $M':$



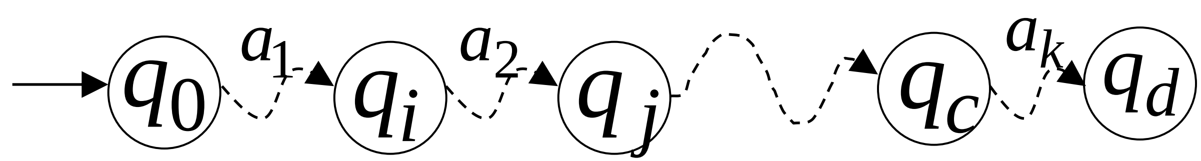
is true by construction of M'

Induction hypothesis: $1 \leq |v| \leq k$

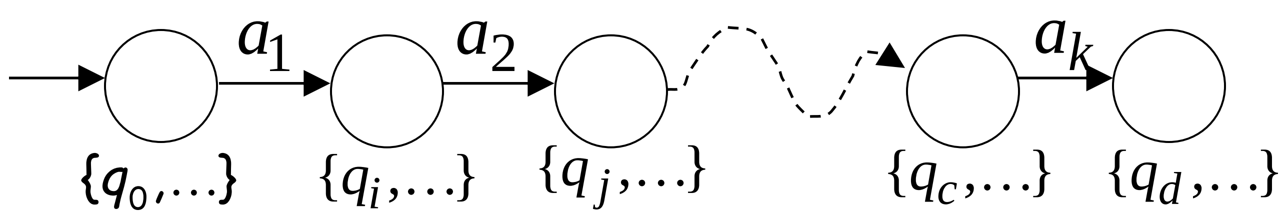
$$v = a_1 a_2 \cdots a_k$$

Suppose that the following hold

NFA M :



DFA M' :

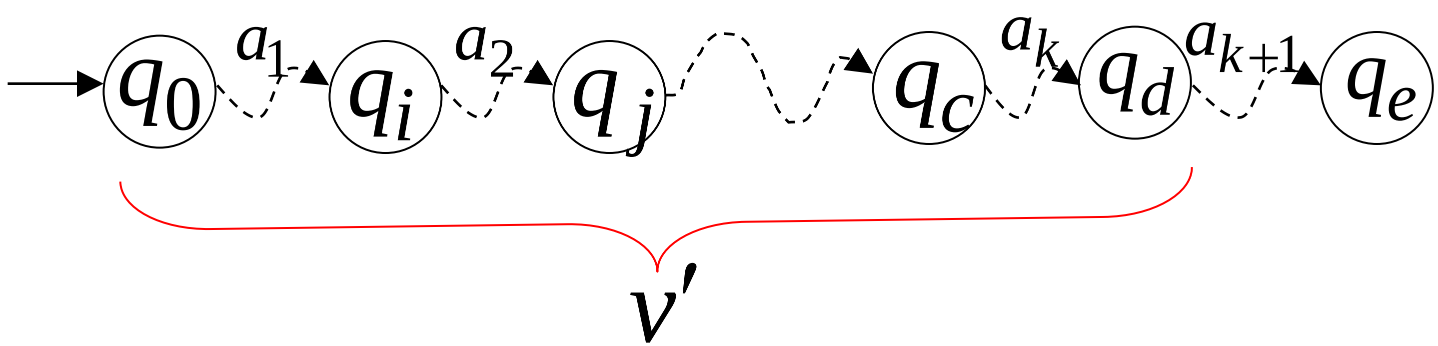


Induction Step: $|v| = k+1$

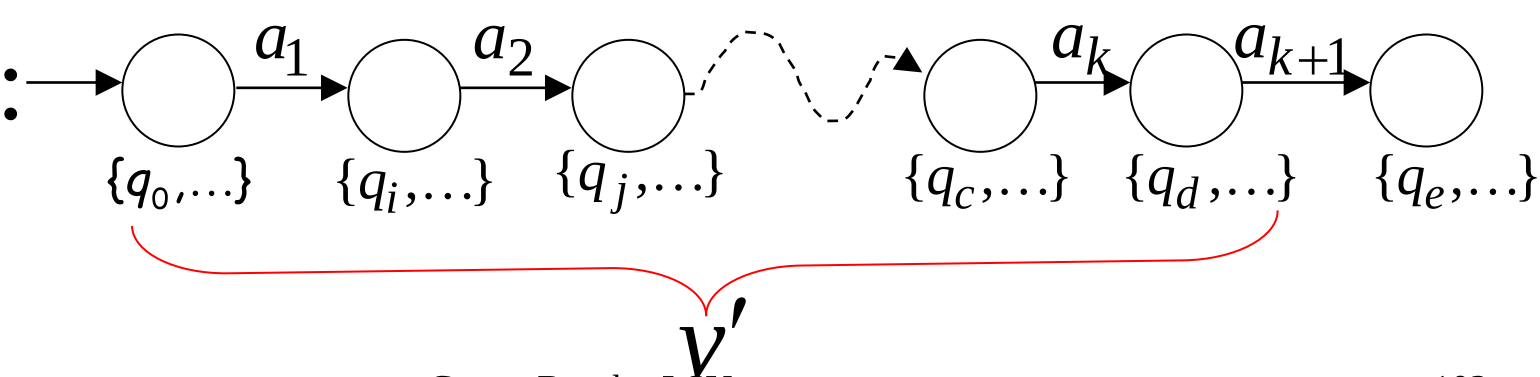
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

Then this is true by construction of M'

NFA M :



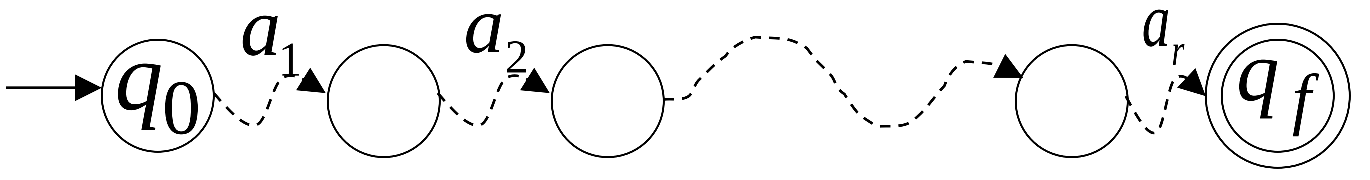
DFA M' :



Therefore if $w \in L(M)$

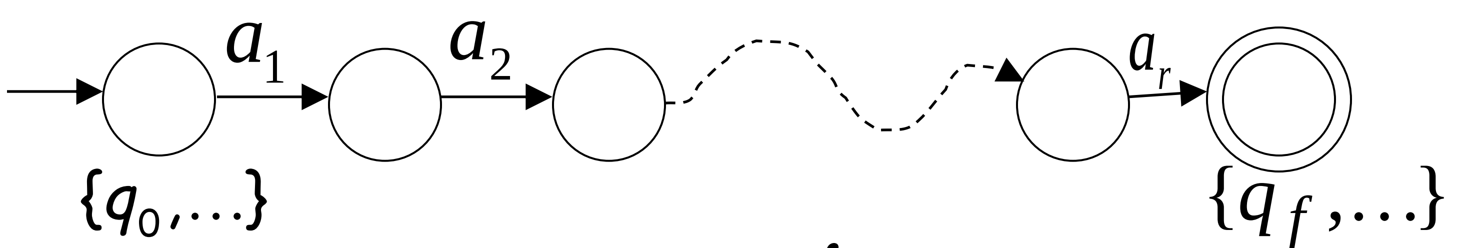
$$w = a_1 a_2 \cdots a_r$$

NFA $M:$



then

DFA $M':$



$$w \in L(M')$$

We have shown: $L(M) \subseteq L(M')$

With a similar proof

we can show: $L(M) \supseteq L(M')$

Therefore: $L(M) = L(M')$

END OF PROOF