BLM2502 Theory of Computation

BLM2502 Theory of Computation

» **Course Outline**

Week	Content
1.	Introduction to Course
2.	Computability Theory, Complexity Theory, Automata Theory, Set Theory, Relations, Proofs, Pigeonhole Principle
3.	Regular Expressions
4.	Finite Automata
5.	**Deterministic and Nondeterministic Finite Automata**
6.	**Epsilon Transition, Equivalence of Automata**
7.	Pumping Theorem
8.	Context Free Grammars
9.	Parse Tree, Ambiguity,
10.	Pumping Theorem
11.	Turing Machines, Recognition and Computation, Church-Turing Hypothesis
12.	Turing Machines, Recognition and Computation, Church-Turing Hypothesis
13.	Review

NFA
Non-Deterministic
Finite Automata

Deterministic Same inputs always Same outputs

Formal Definition of NFA

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

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

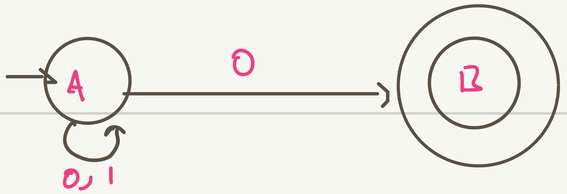
Σ: Input aplhabet, i.e. $\{a, b\}$ $\quad \varepsilon \notin \Sigma$

δ: Transition function $Q \times \Sigma \to 2^Q$

q_0: Initial state

F: Accepting states

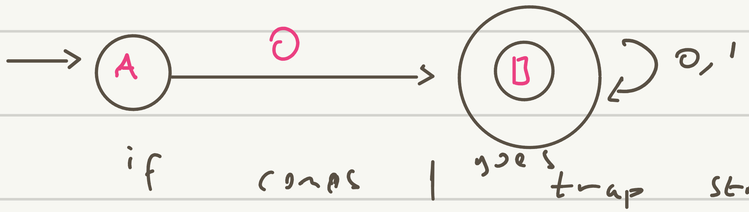
A	∅	A	B
AB	ε	AB	ε
A	B	A	B
AB	ε	AD	ε



→ Nondeterministic

$L = \{ w \mid w \text{ ends with } 0 \}$

↳ Burada ağaç olarak düşünüyoruz ve ağaçlardan herhangi biri L'yi işliyorsa yazılabilir.



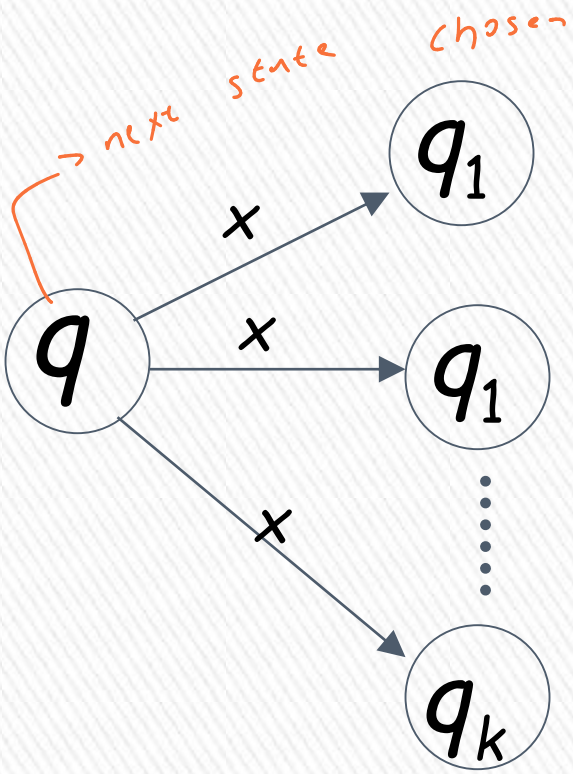
$L = \{ w \mid w \text{ starts with } 0 \}$

if reads | goes trap state

Transition Function δ

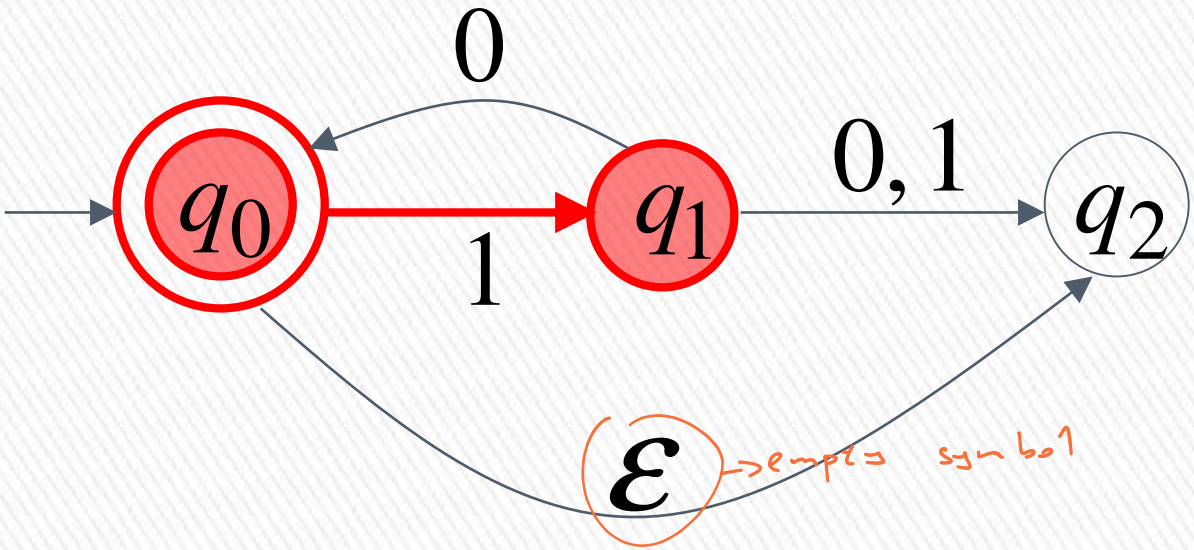
non-deterministic

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

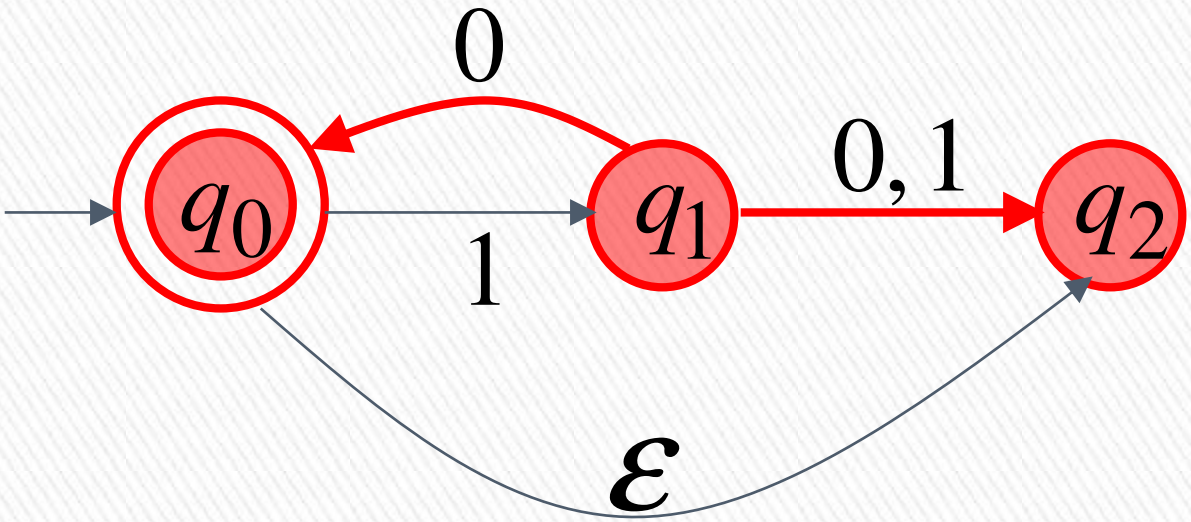


resulting states with following **one** transition with symbol x

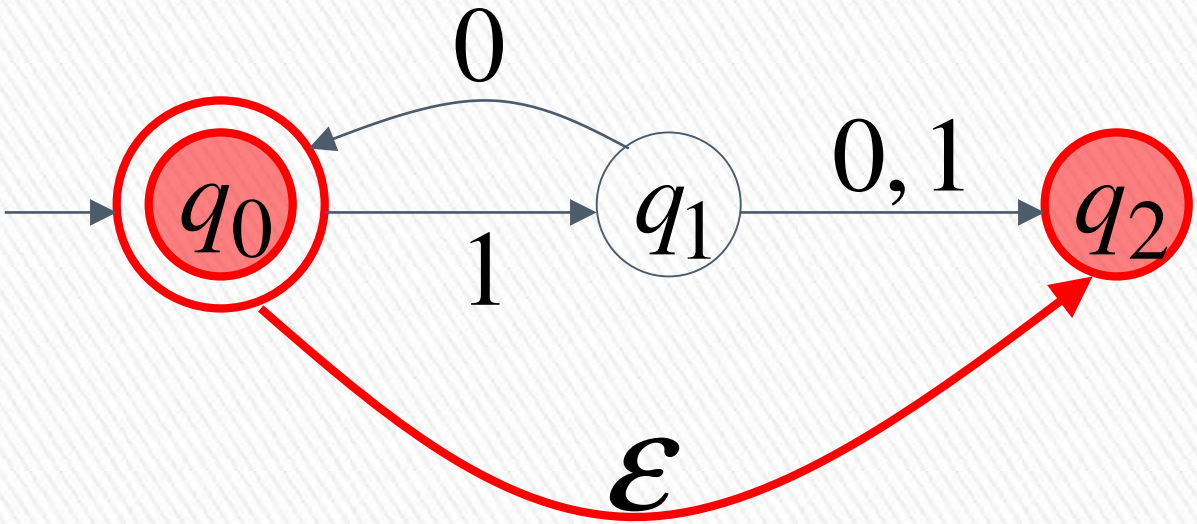
$$\delta(q_0, 1) = \{q_1\}$$



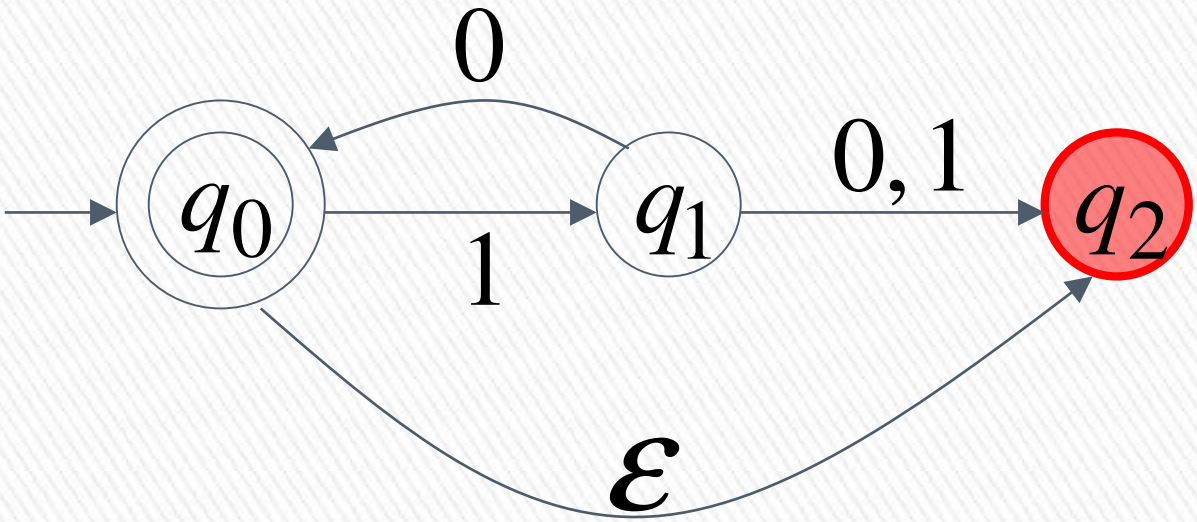
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

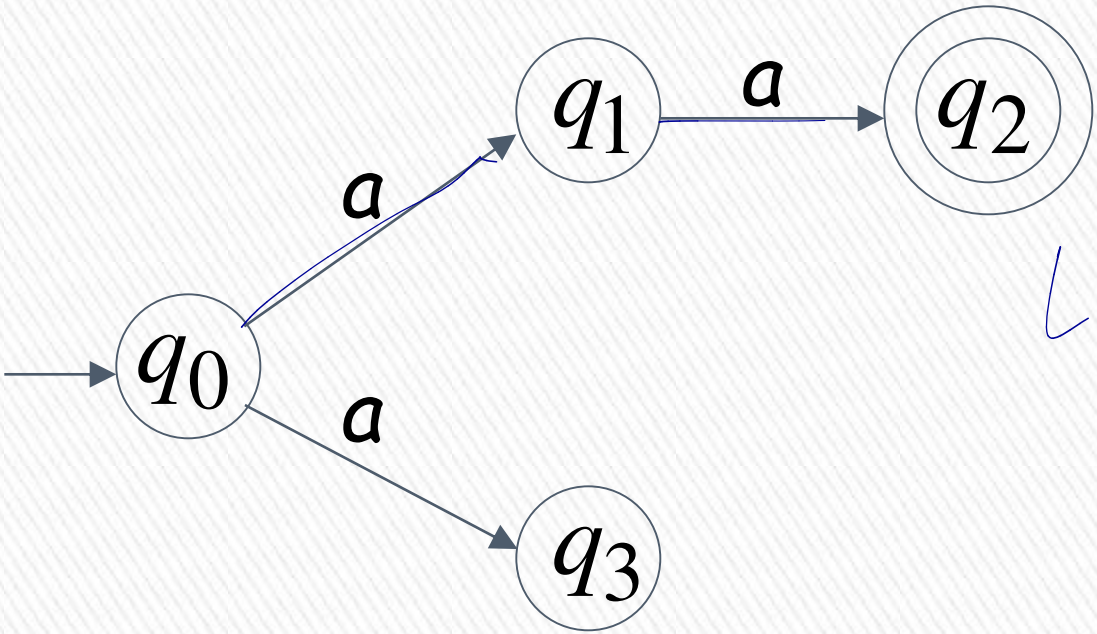


$$\delta(q_2, 1) = \emptyset$$



Nondeterministic Finite Automaton (NFA)

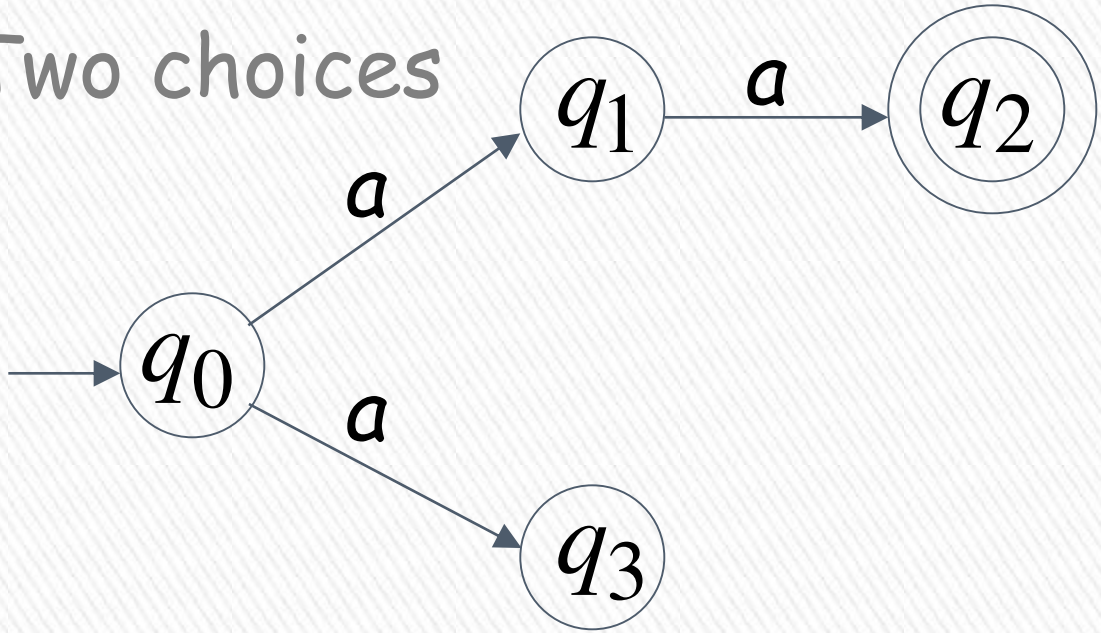
Alphabet = $\{a\}$



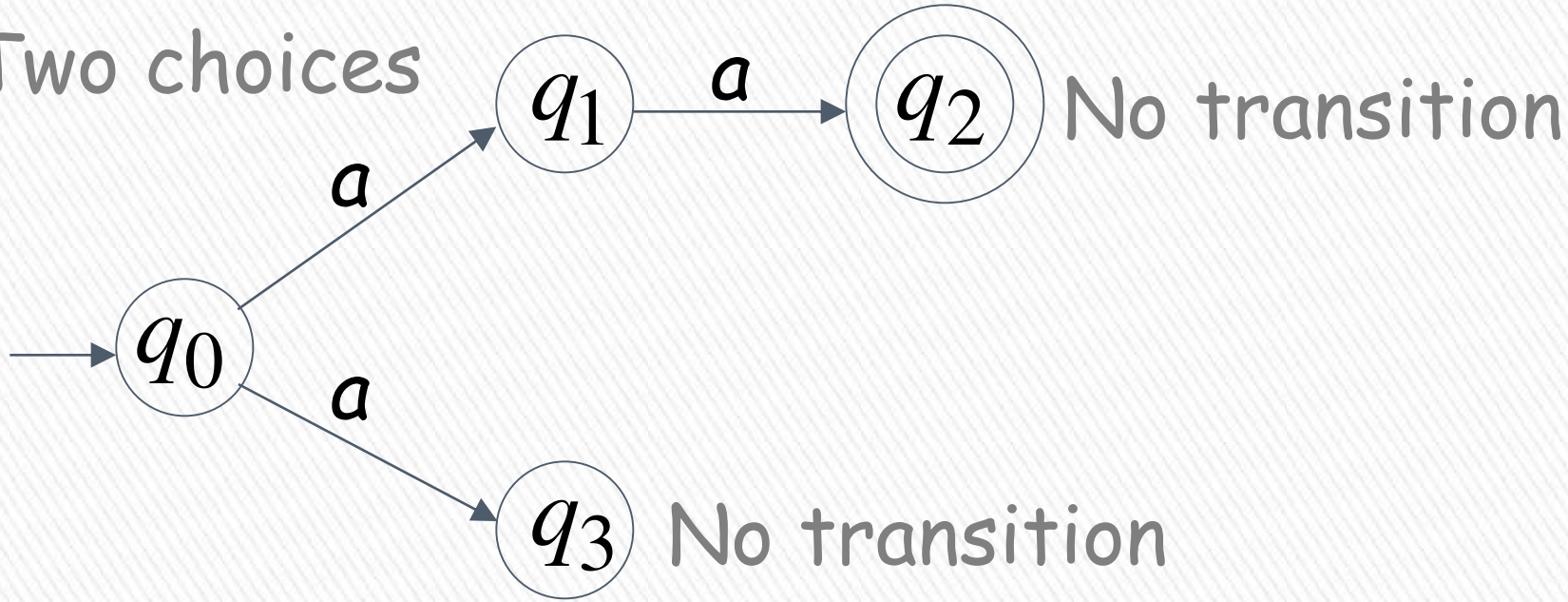
$L = \{aa\}$

Alphabet = $\{a\}$

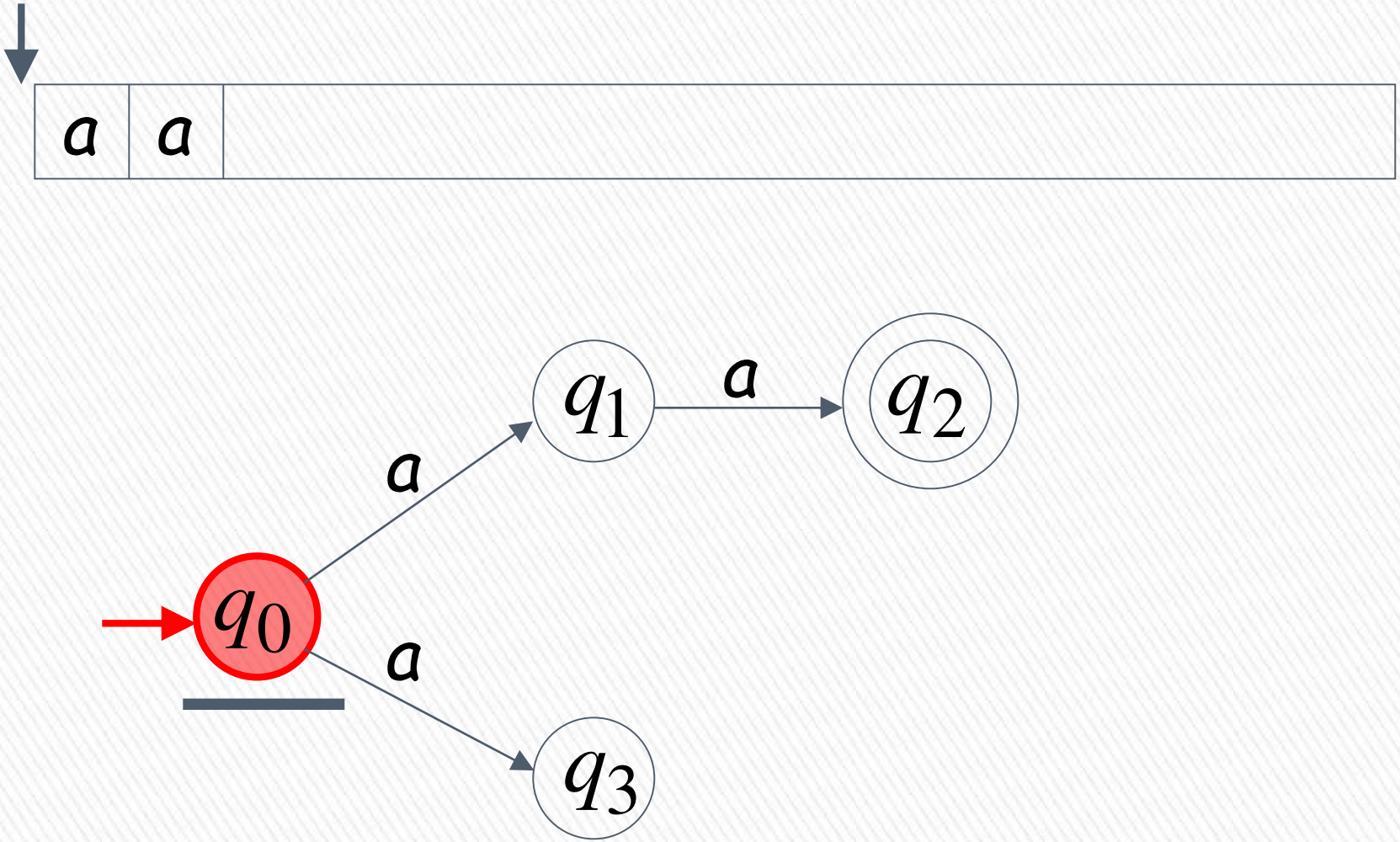
Two choices



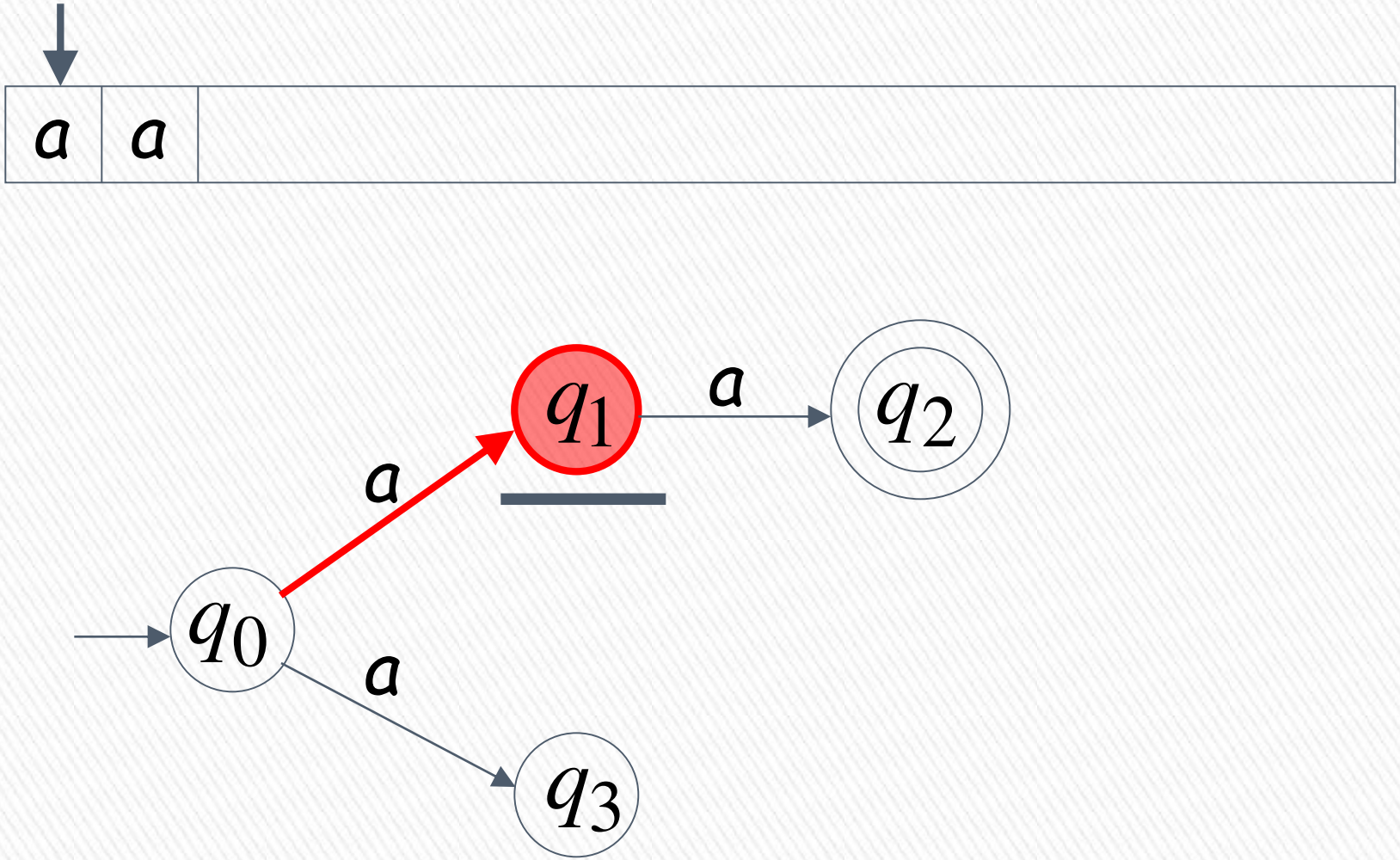
Alphabet = $\{a\}$



First Choice

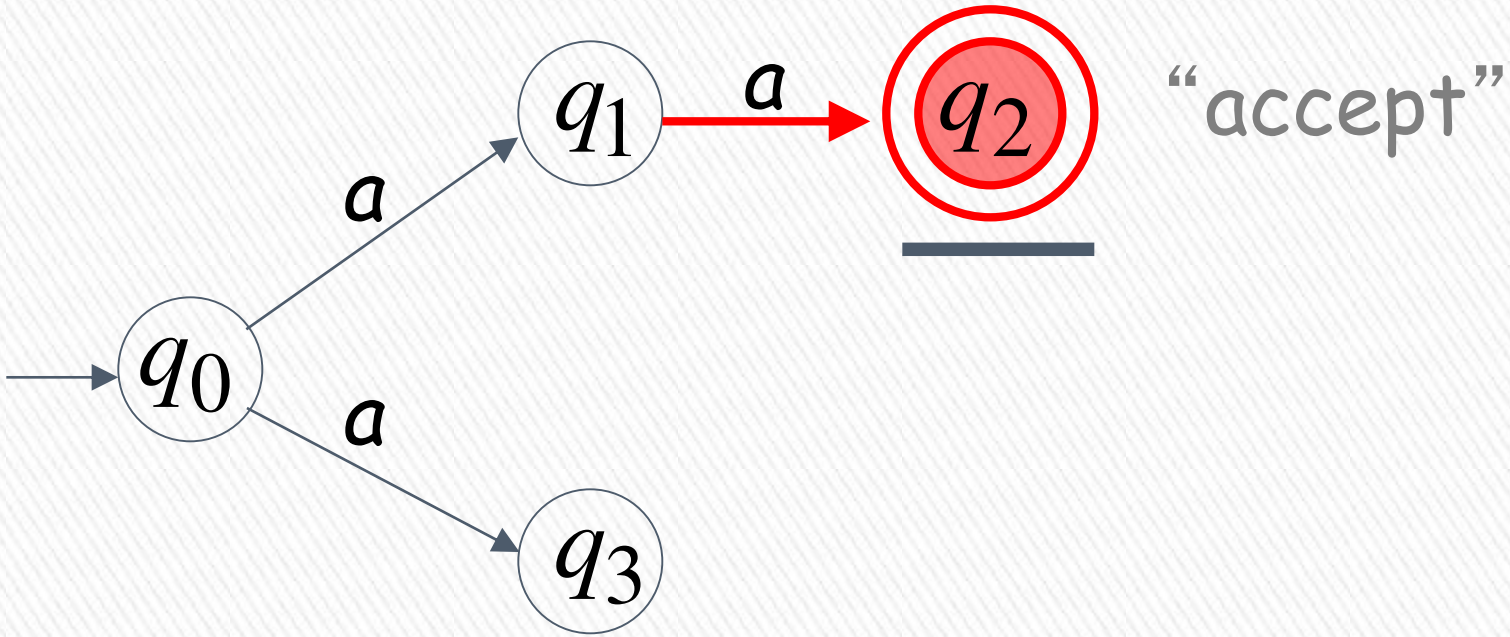


First Choice



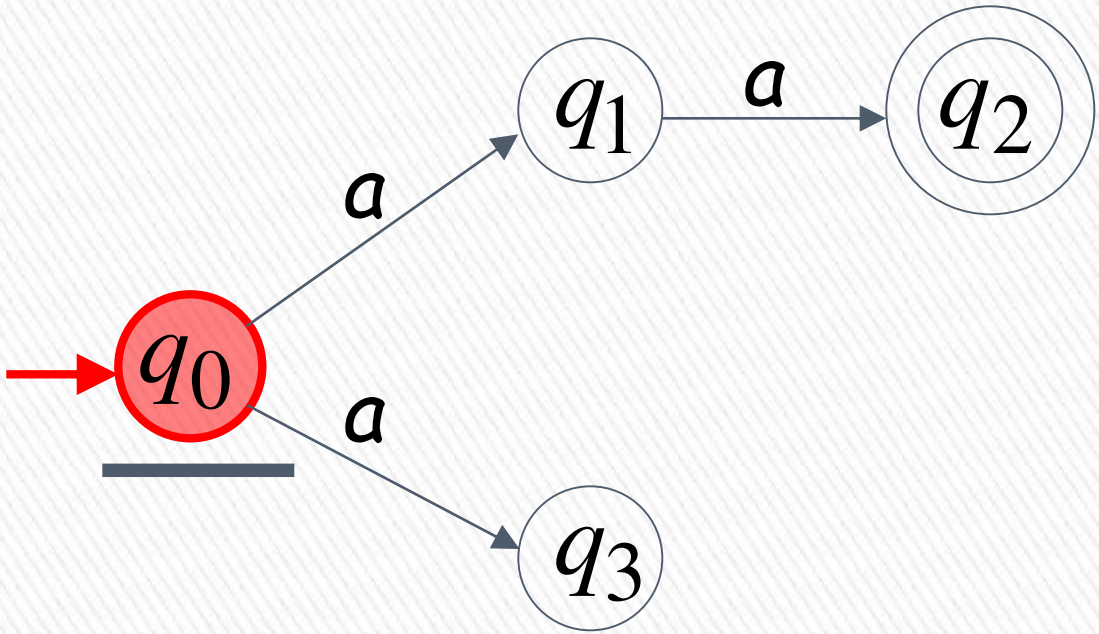
First Choice

All input is consumed



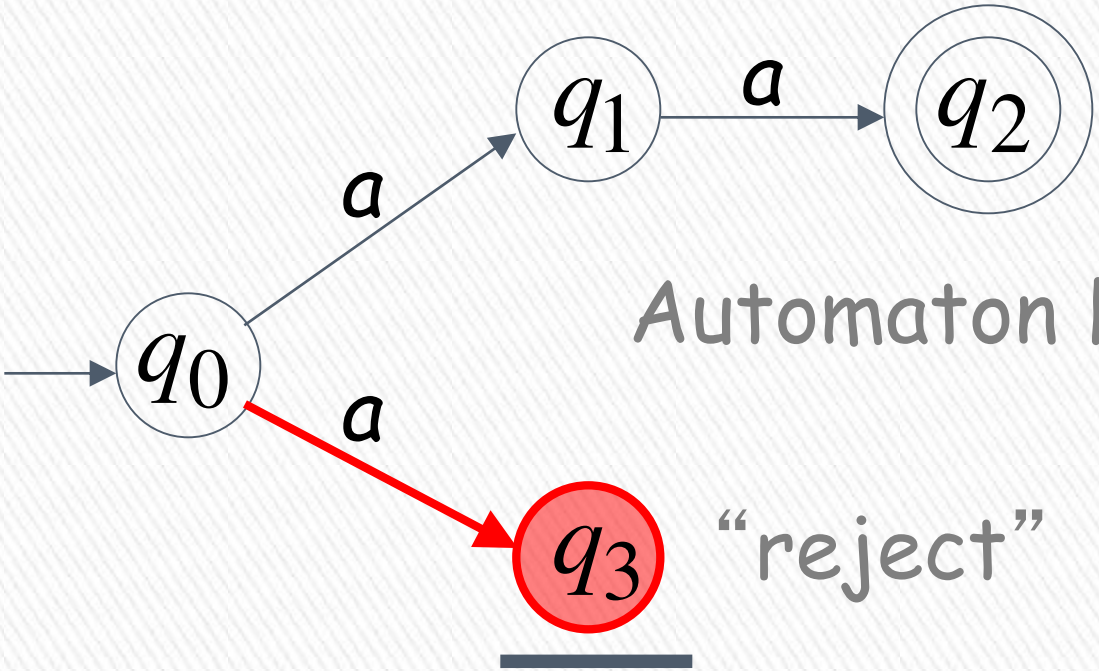
"accept"

Second Choice



Second Choice

Input cannot be consumed



Automaton Halts

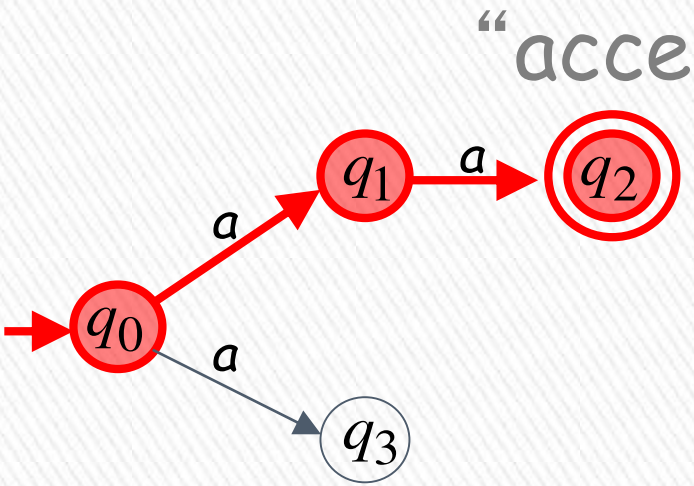
"reject"

An NFA accepts a string:

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

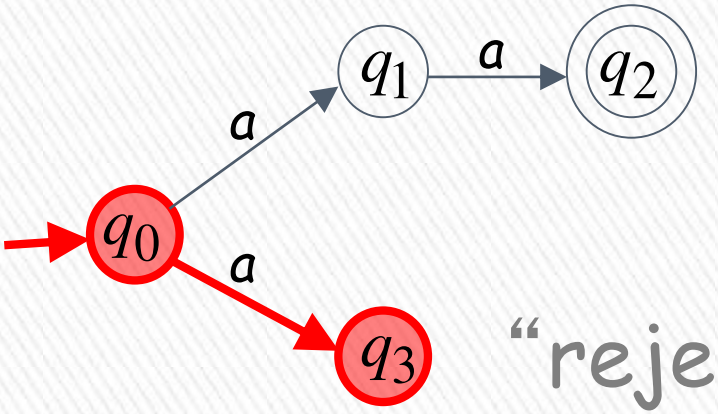
i.e., all the input string is processed and the automaton is in an accepting state

aa is accepted by the NFA:



"accept"

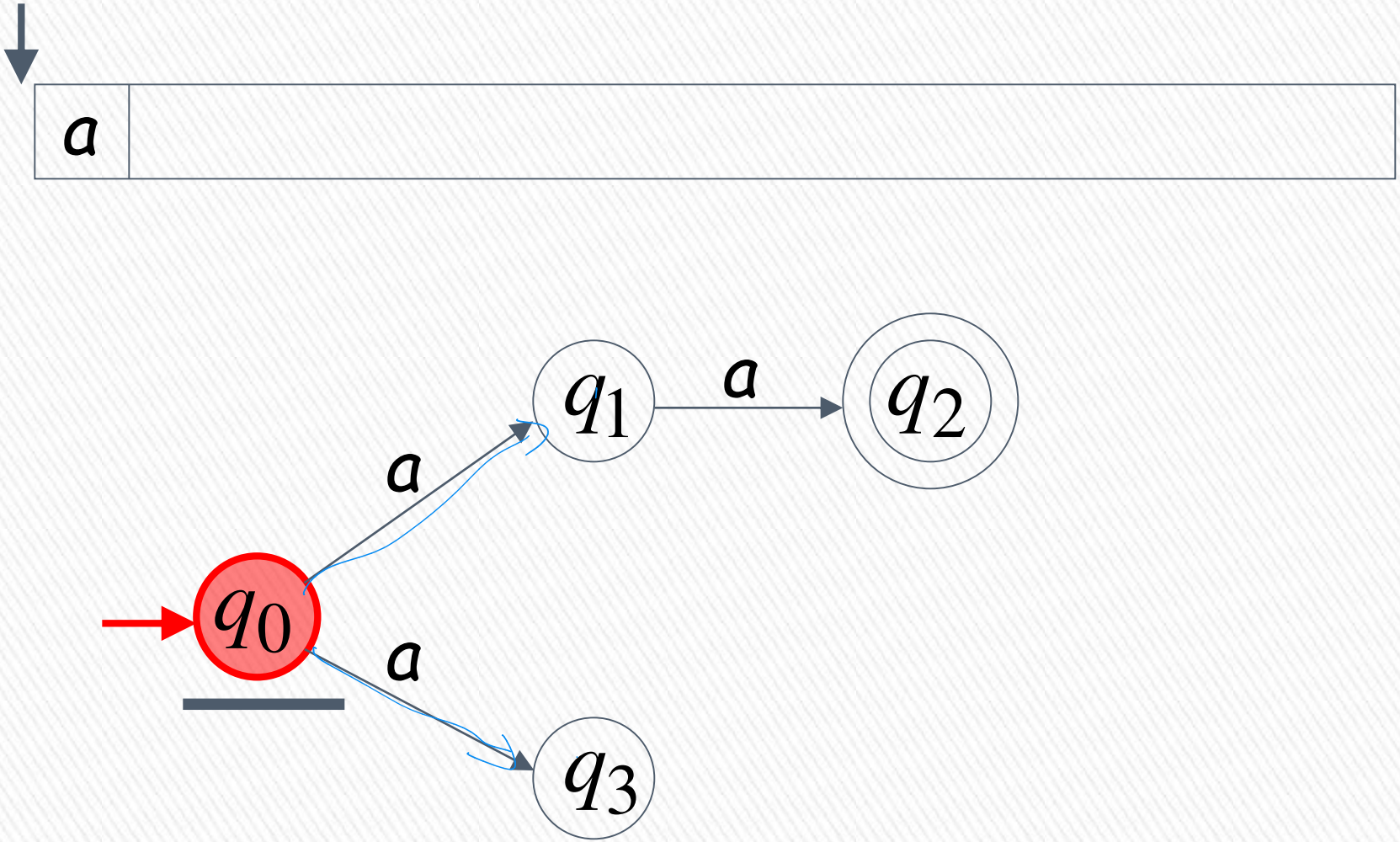
because this computation accepts aa



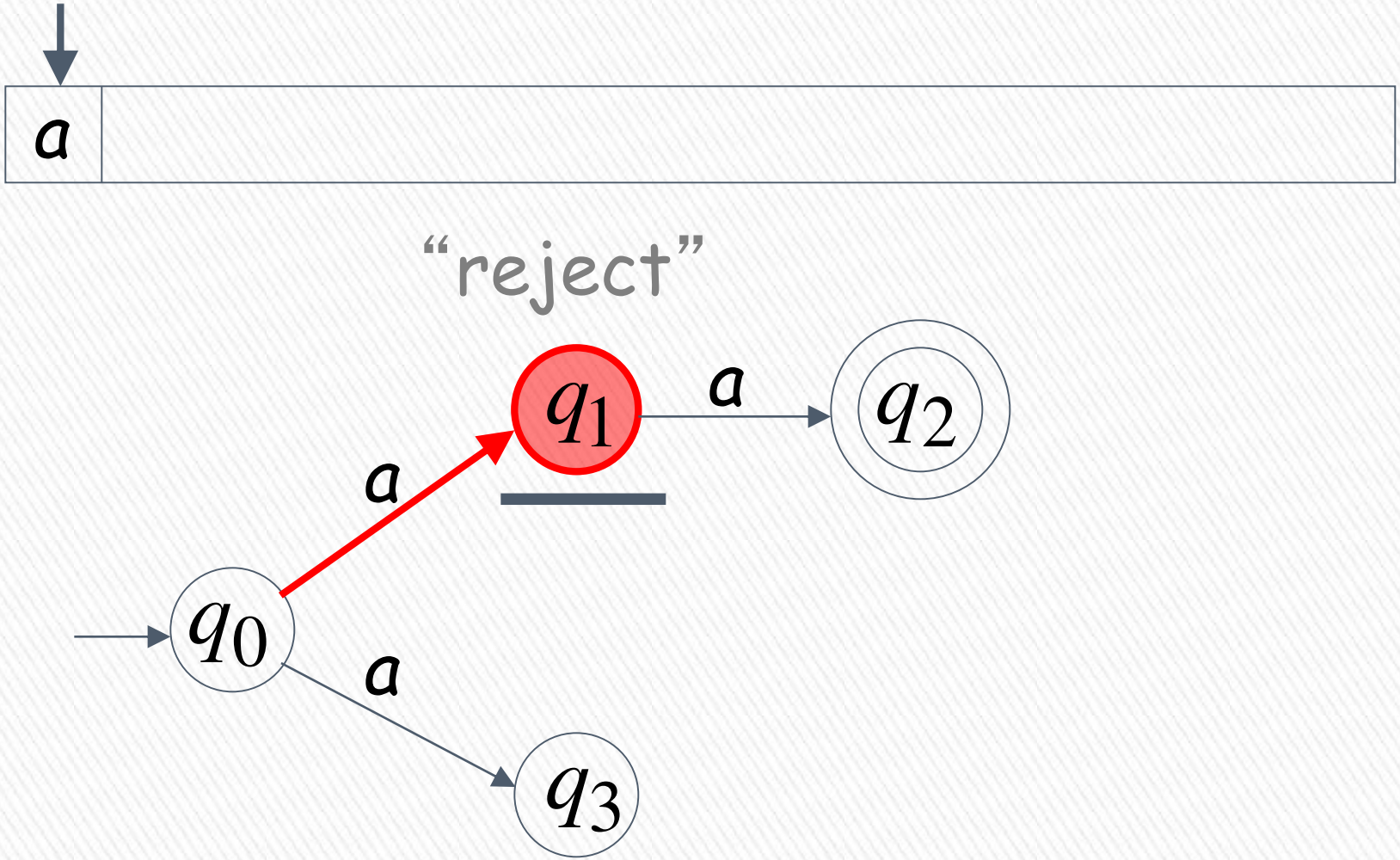
"reject"

this computation is ignored

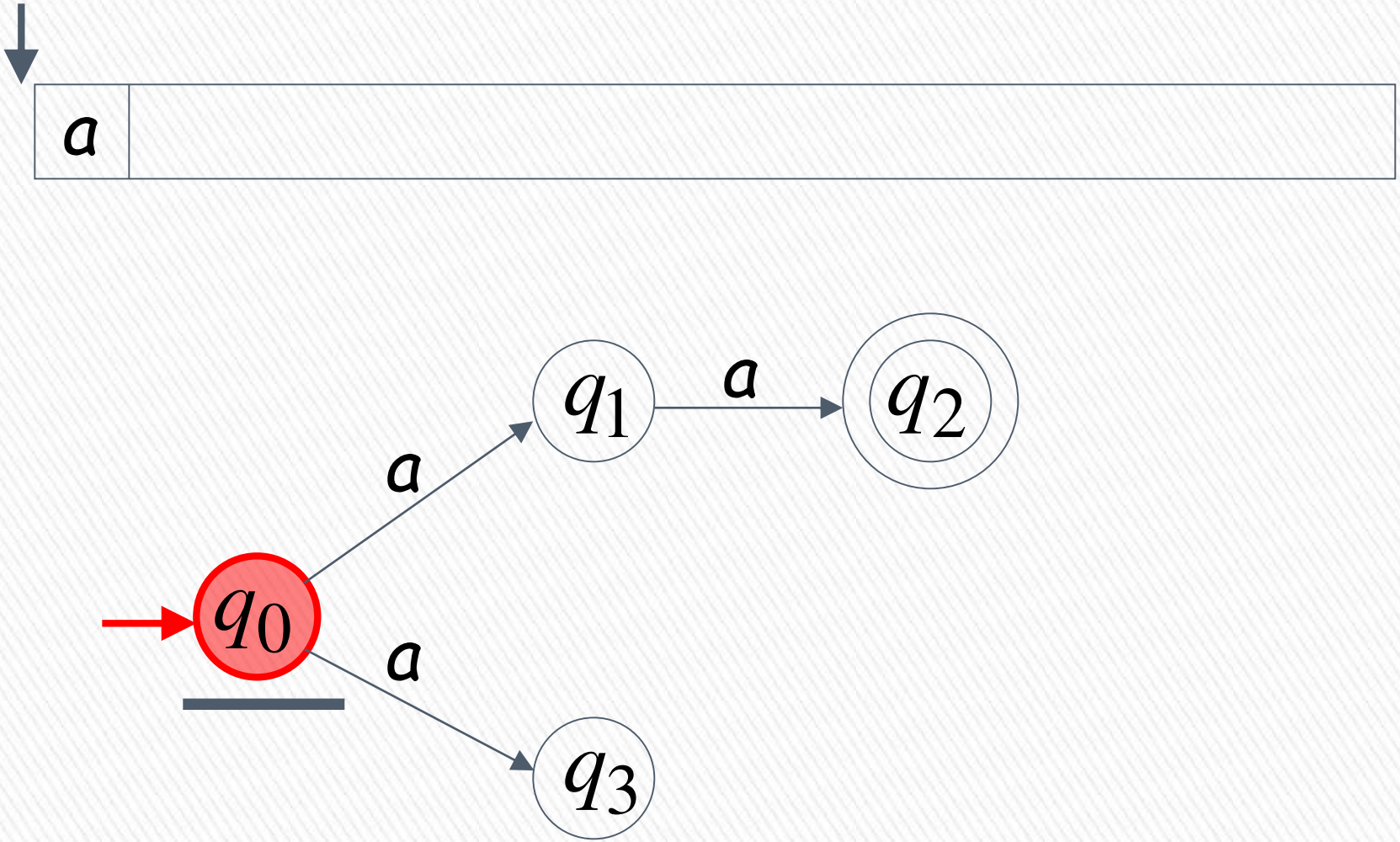
Rejection example



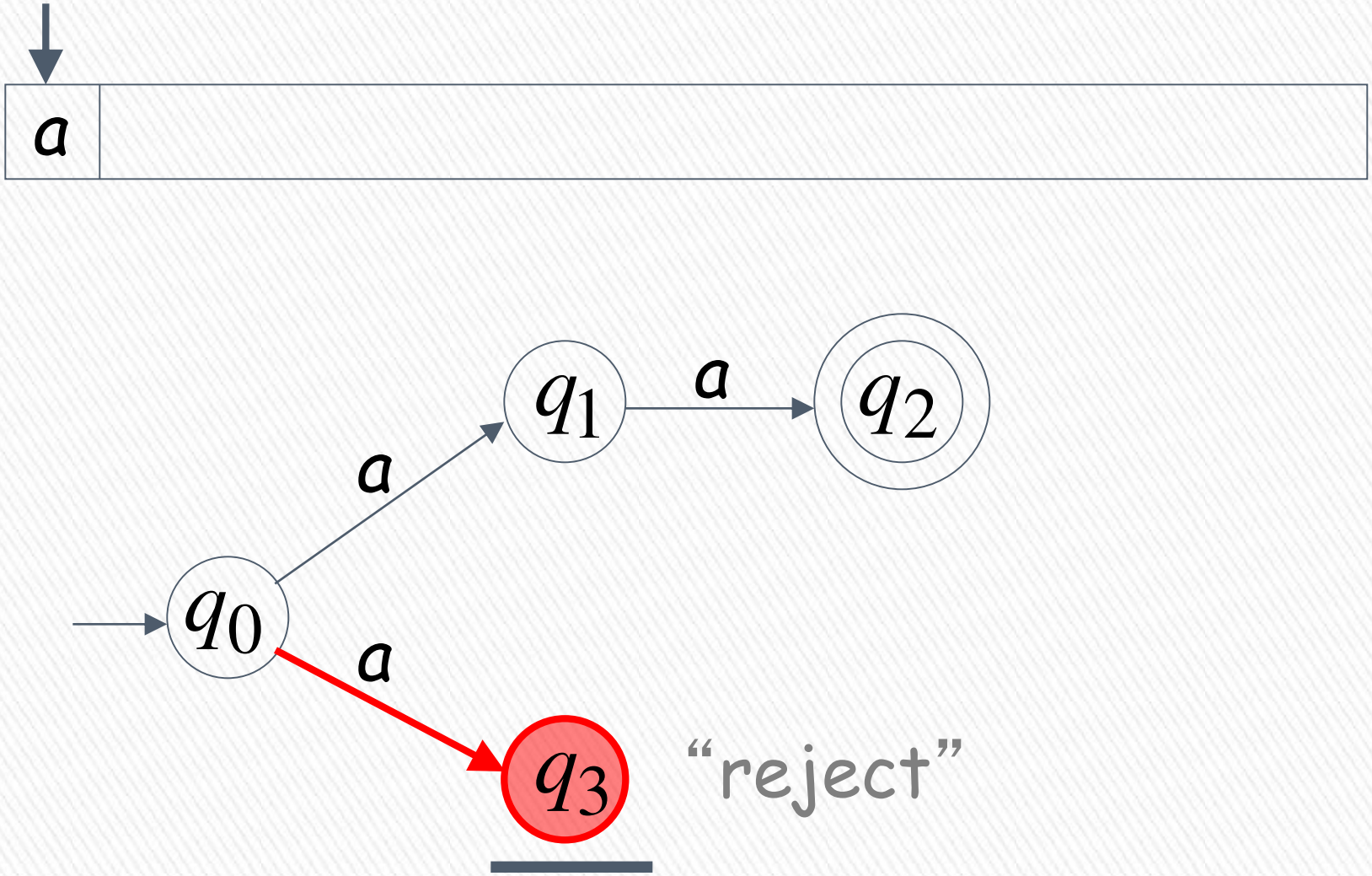
First Choice



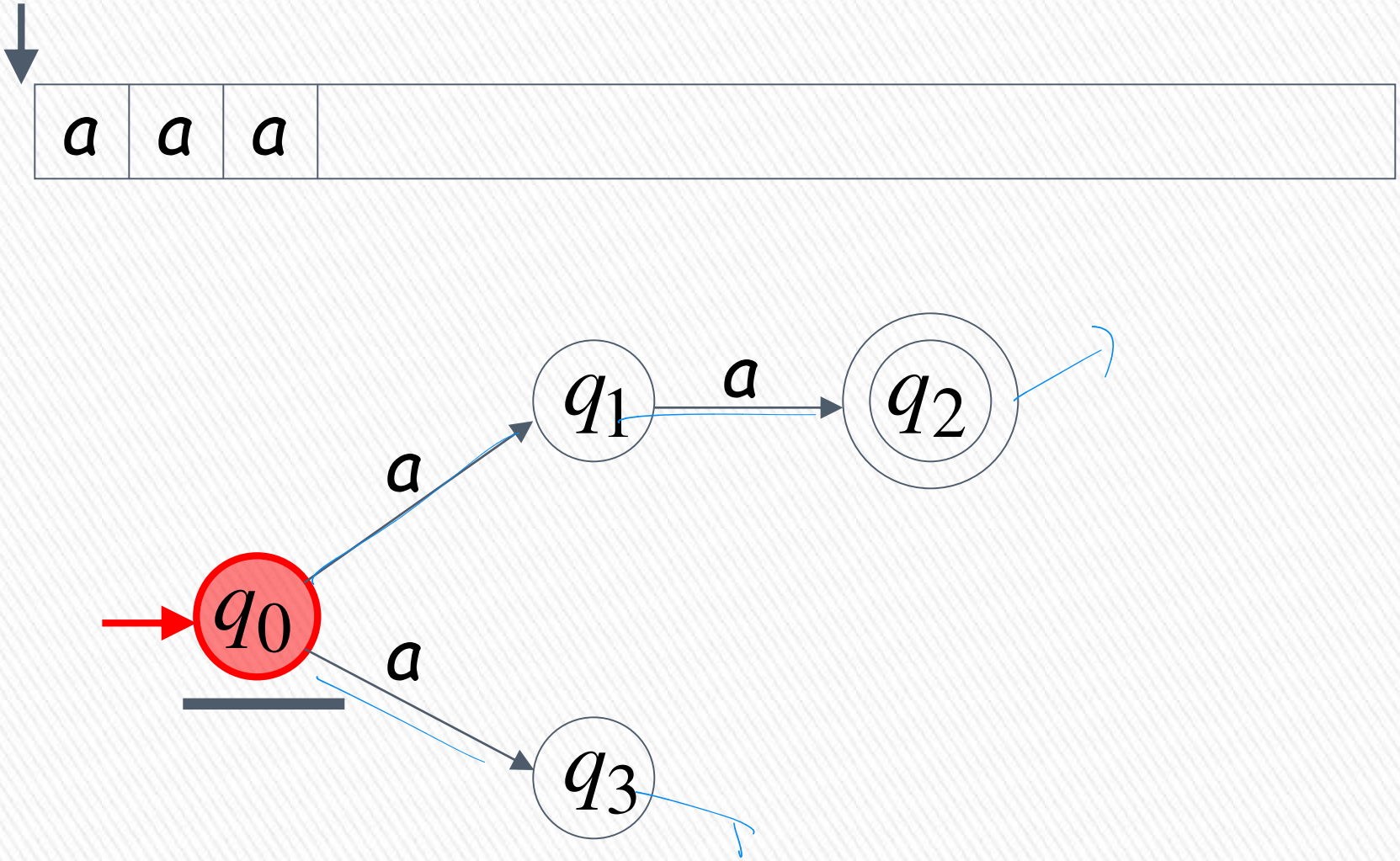
Second Choice



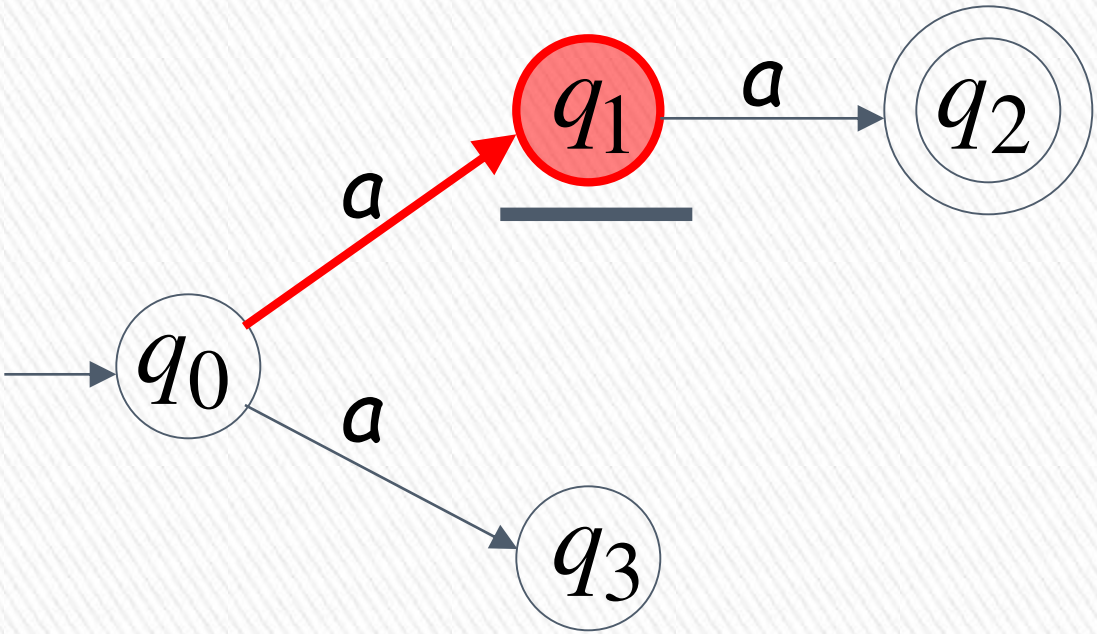
Second Choice



Another Rejection example

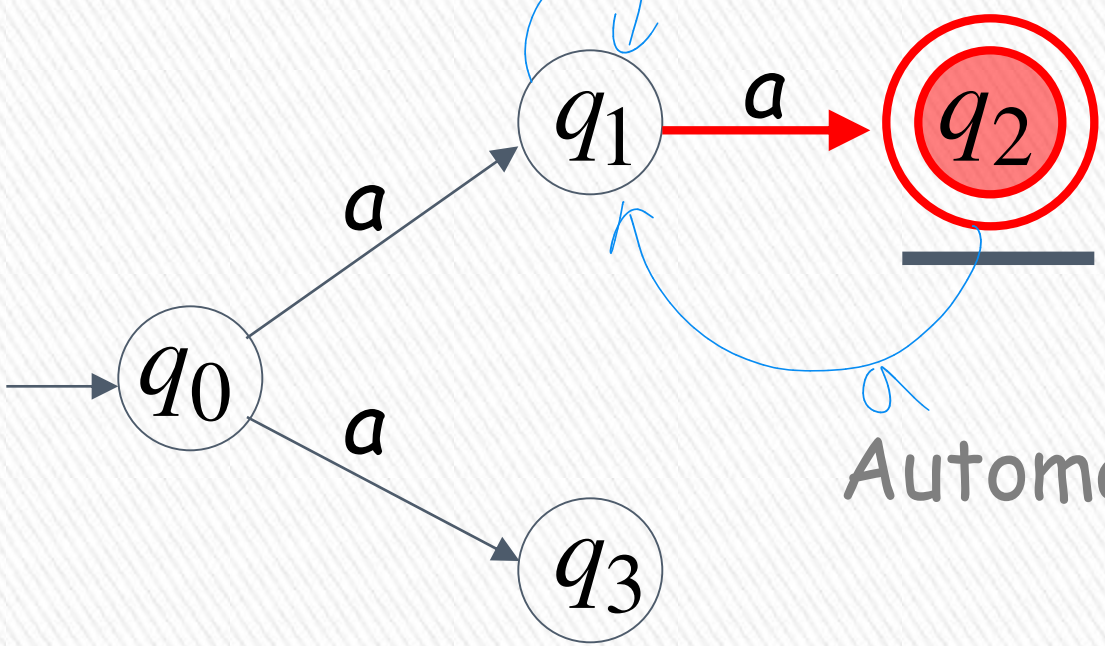


First Choice



First Choice

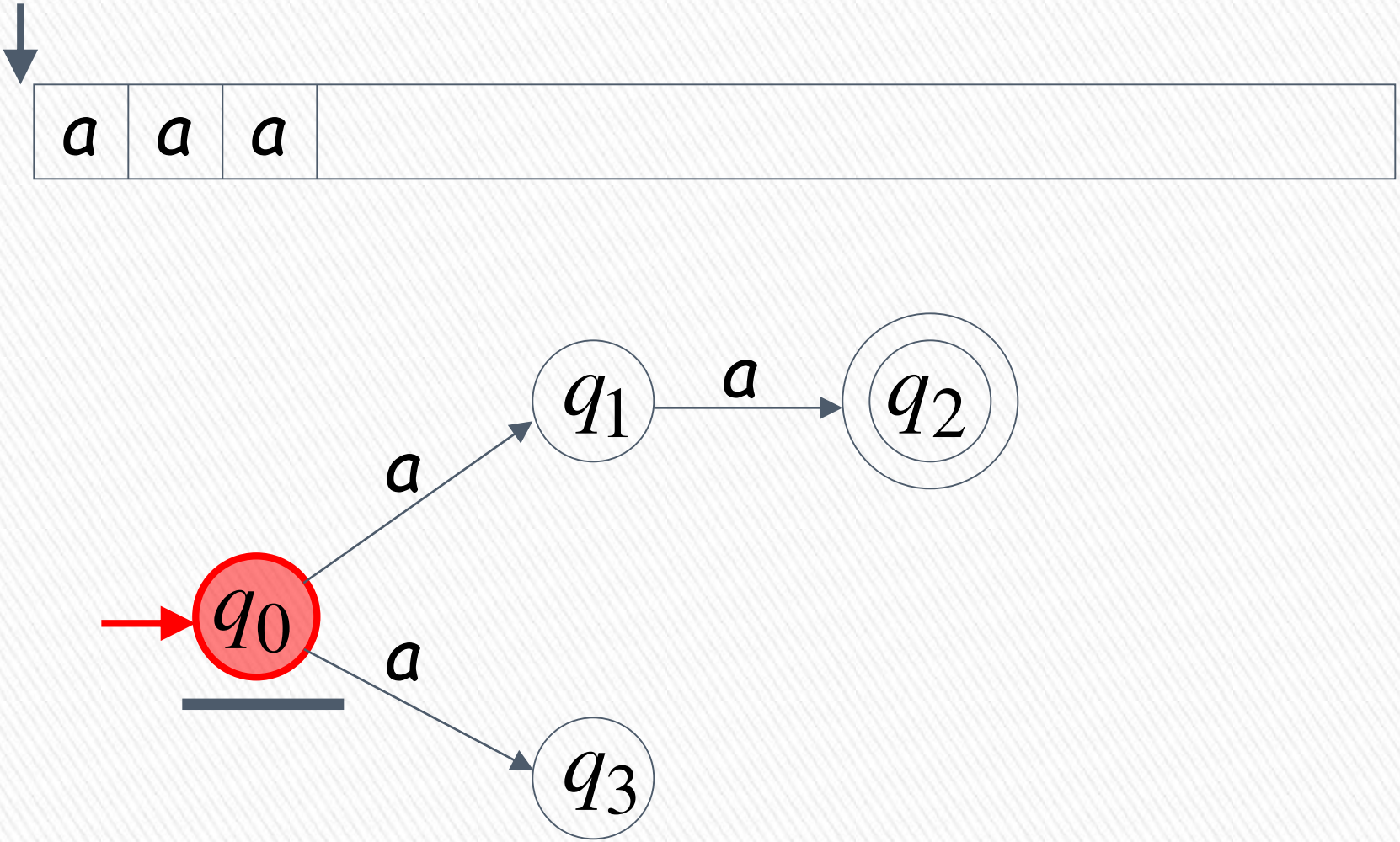
Input cannot be consumed



"reject"

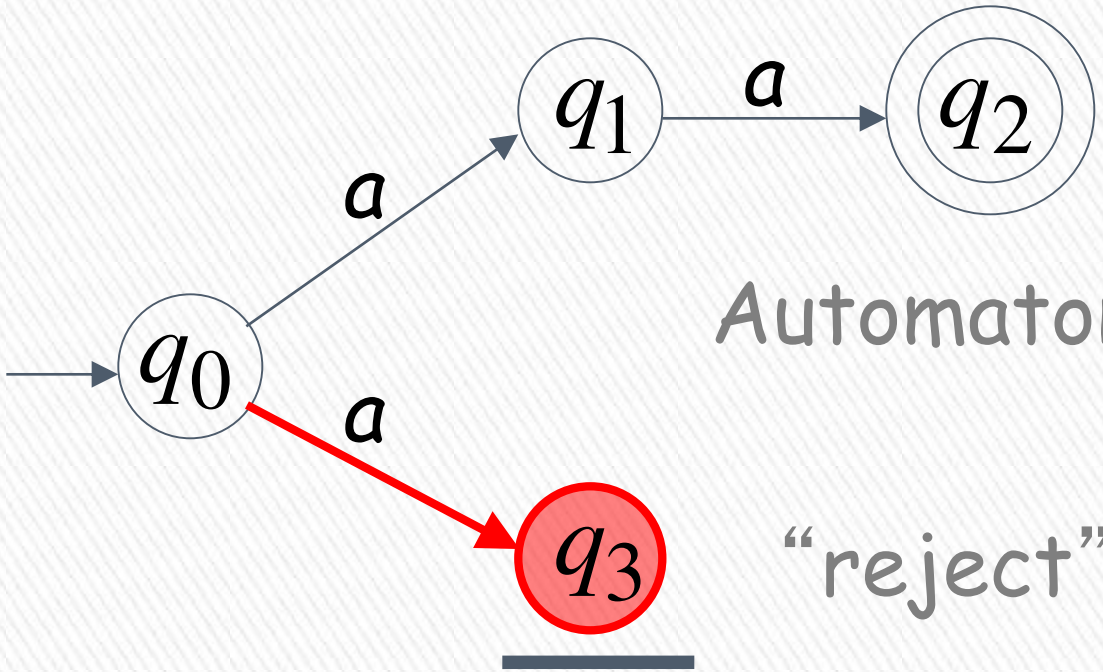
Automaton halts

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.

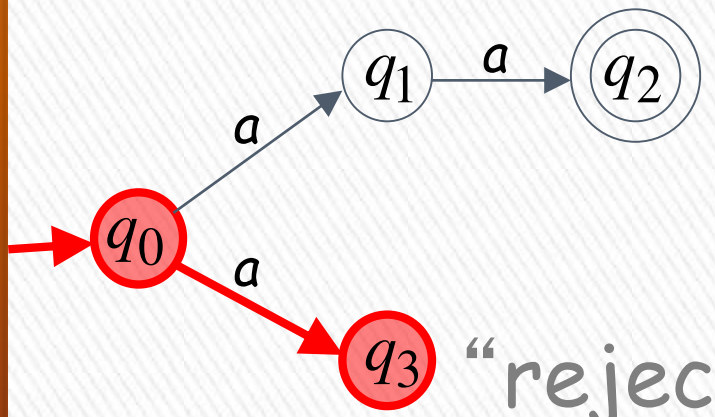
For each computation:

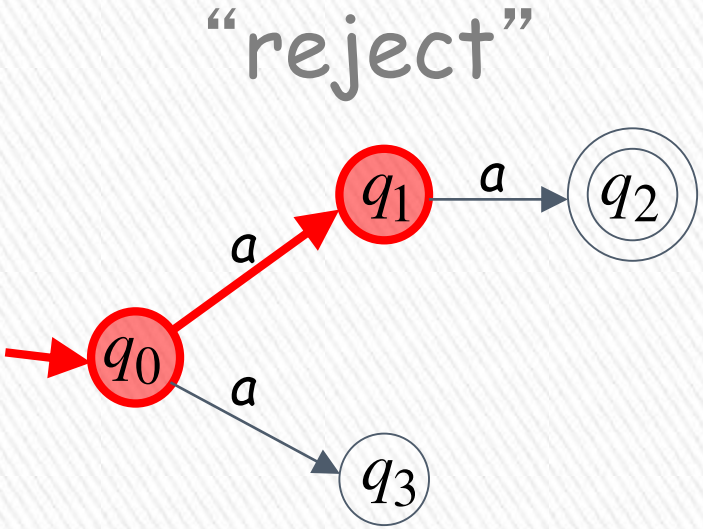
- All the input is consumed and the automaton is in a non final state

OR

- The input cannot be consumed

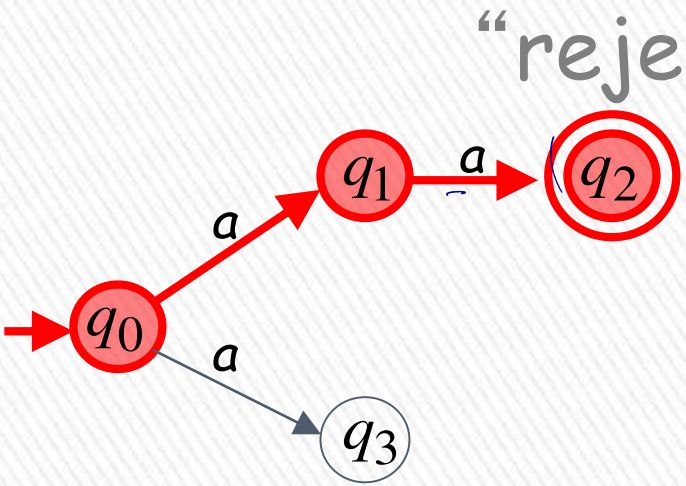
a is rejected by the NFA:

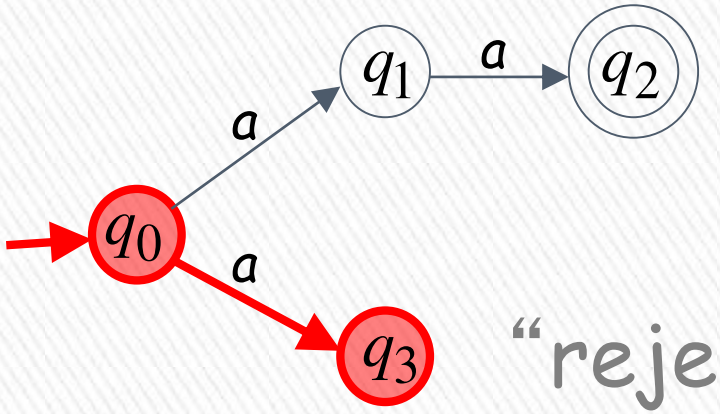




All possible computations lead to rejection

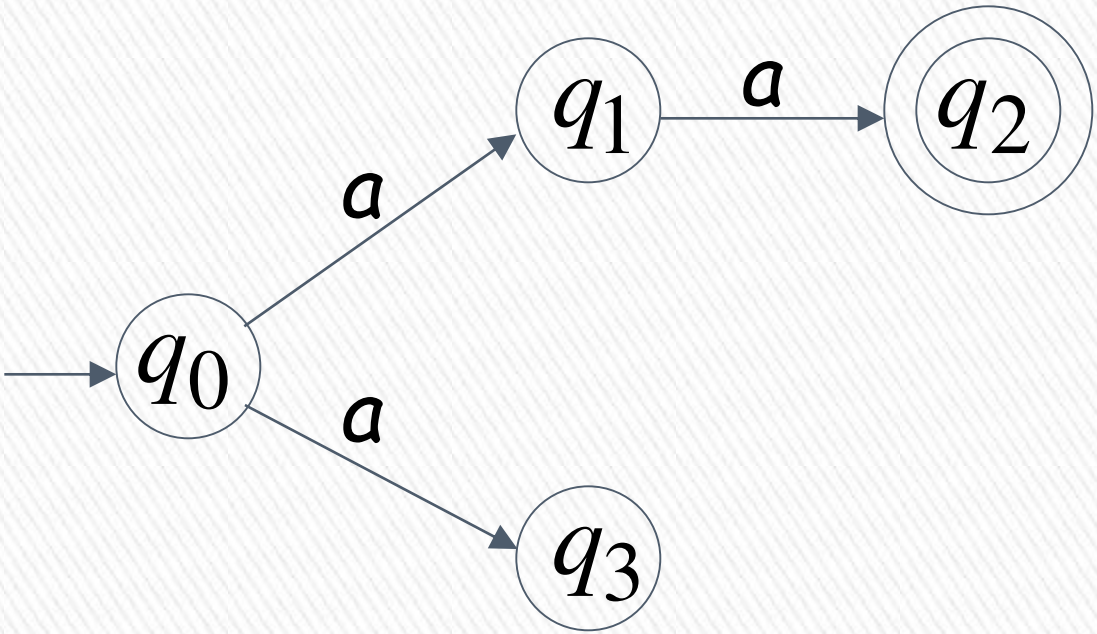
aaa is rejected by the NFA:



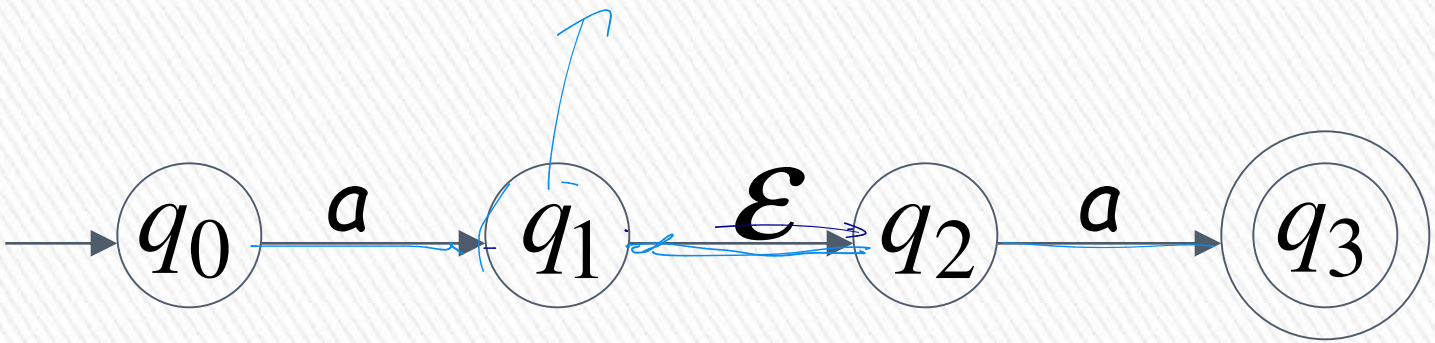


All possible computations lead to rejection

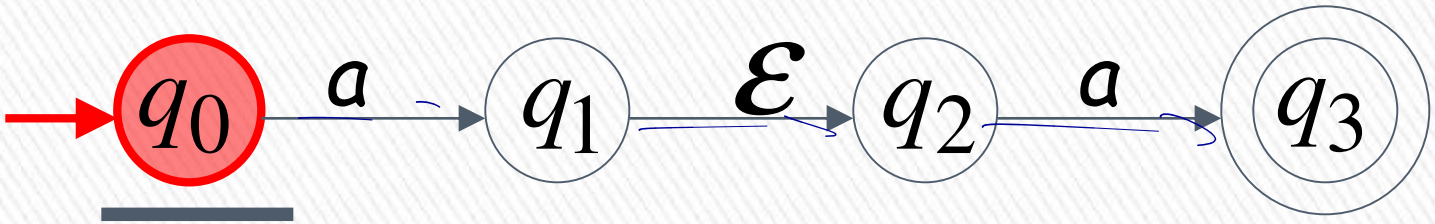
Language accepted: $L = \{aa\}$



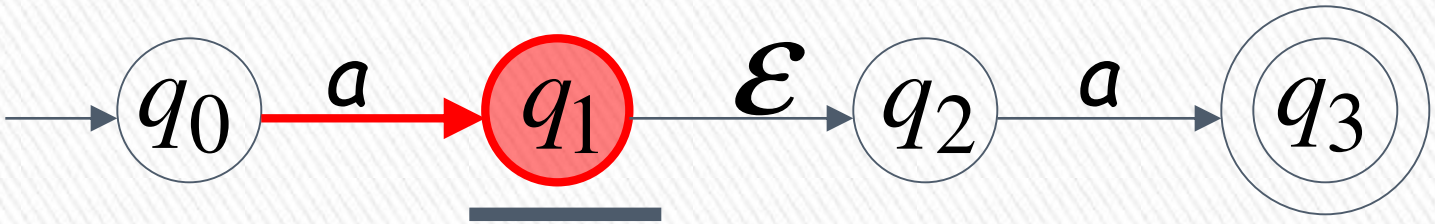
Epsilon Transition



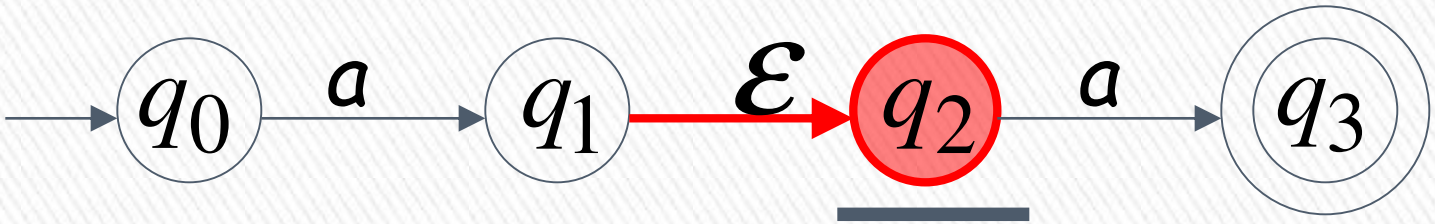

a
a


q_0
a
q_1
ε
q_2
a
q_3


a
a

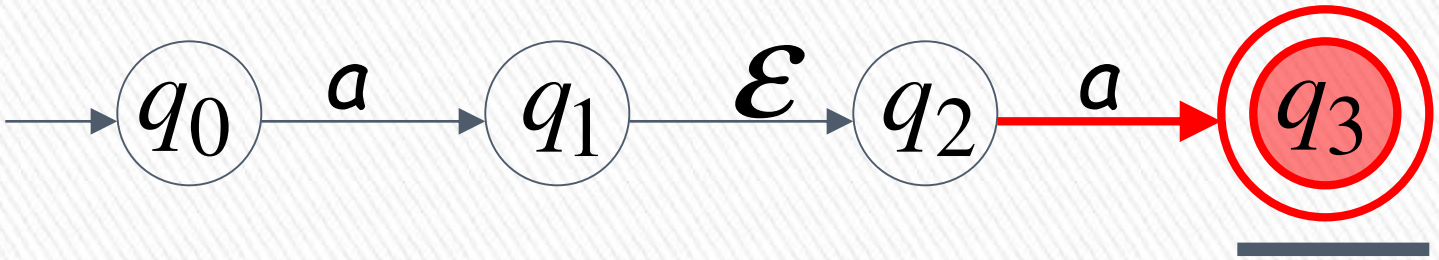

q_0
a
q_1
ε
q_2
a
q_3

input tape head does not move



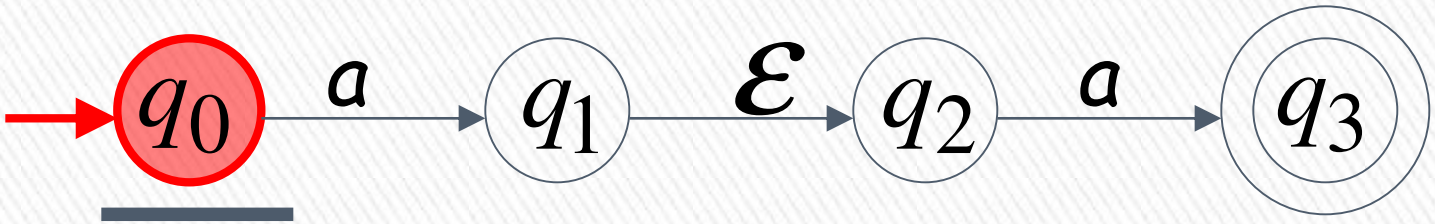
all input is consumed

"accept"

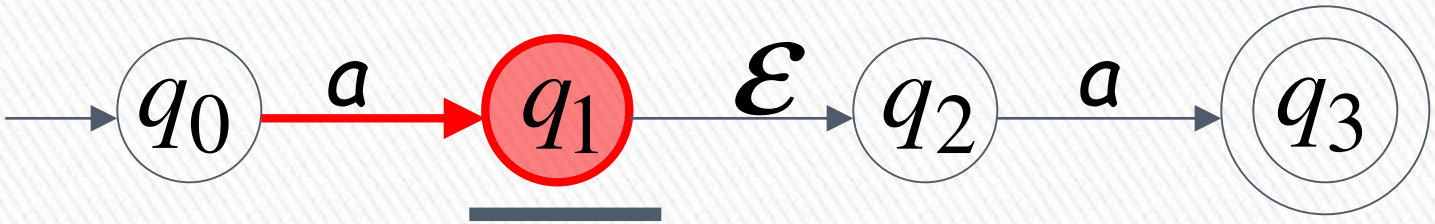


String aa is accepted

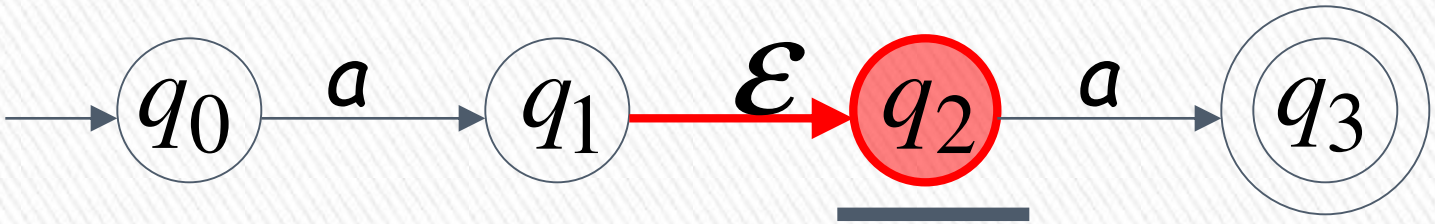
Rejection Example




a
a
a

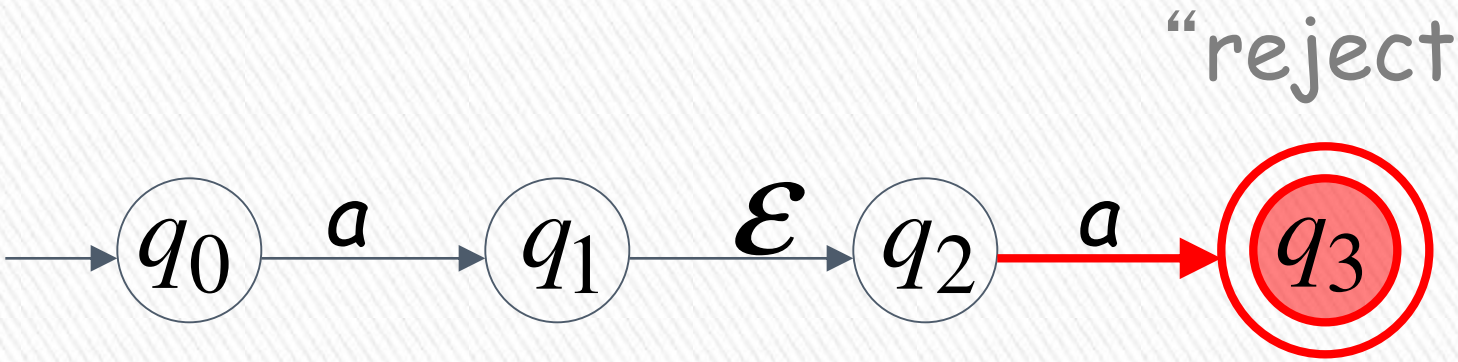

q_0
a
q_1
ε
q_2
a
q_3

(read head doesn't move)



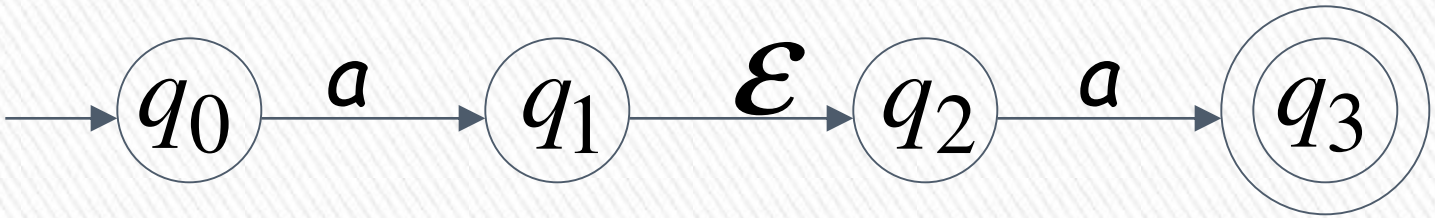
Input cannot be consumed

Automaton halts

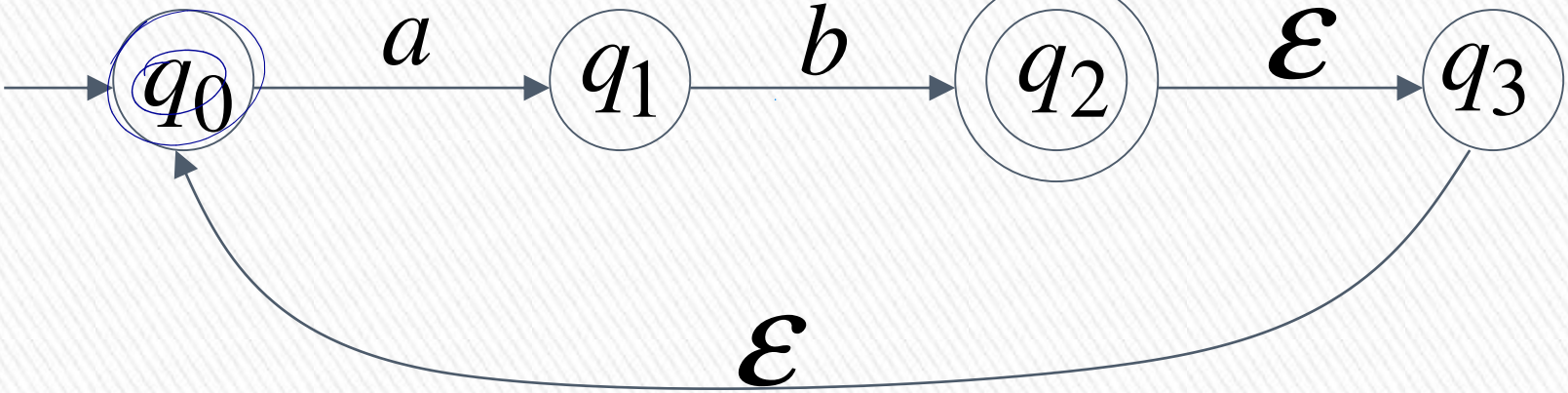


String **aaa** is rejected

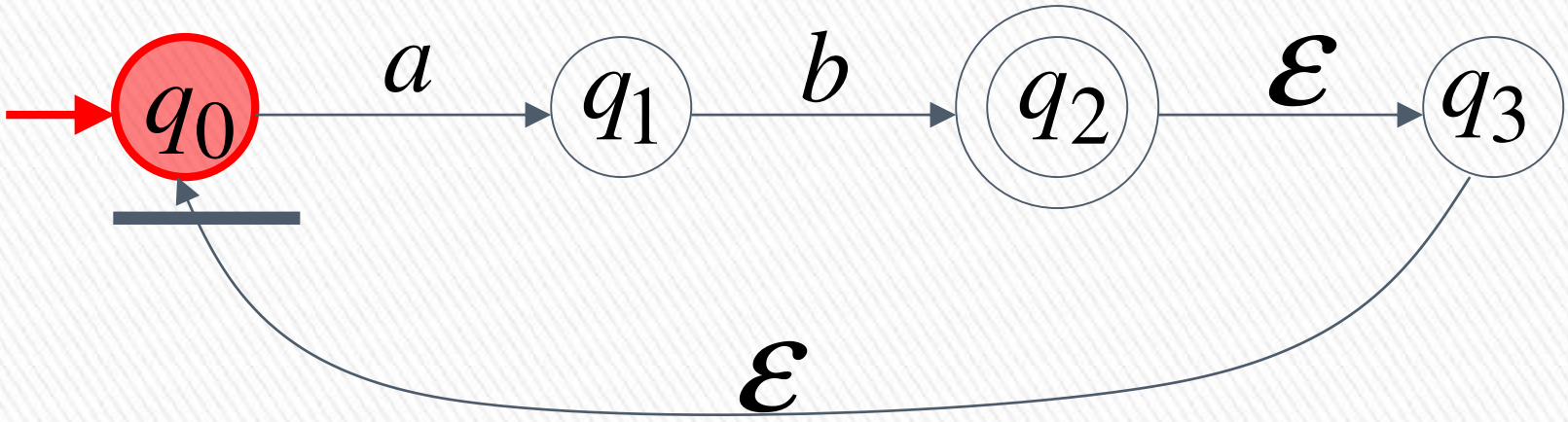
Language accepted: $L = \{aa\}$



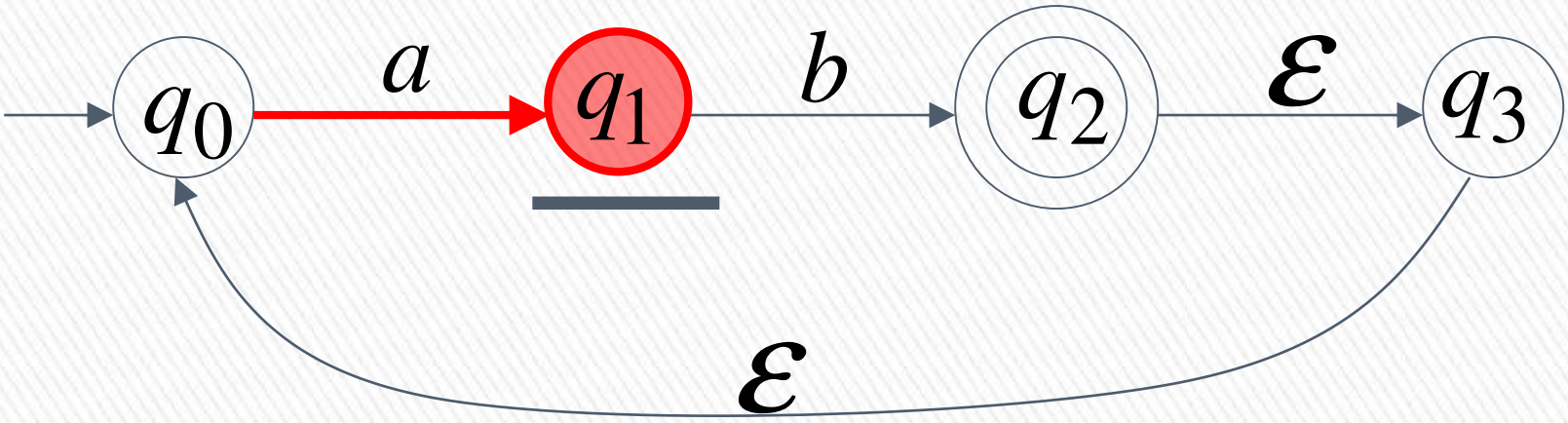
Another NFA Example



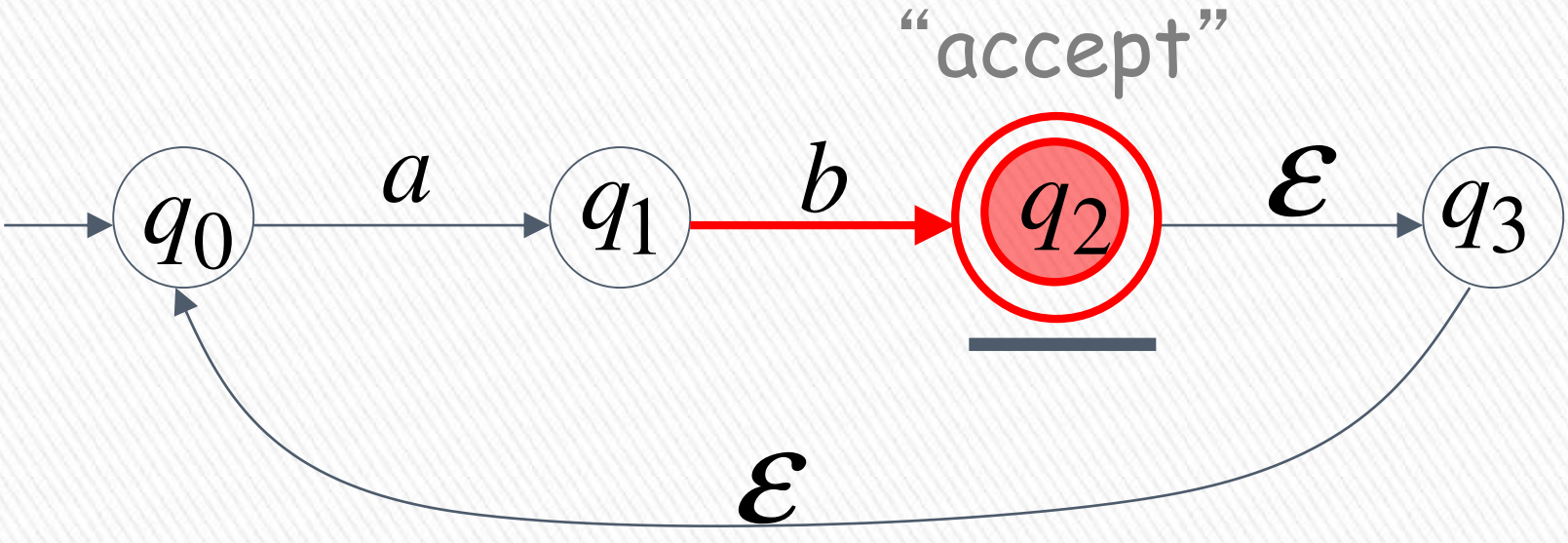

a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

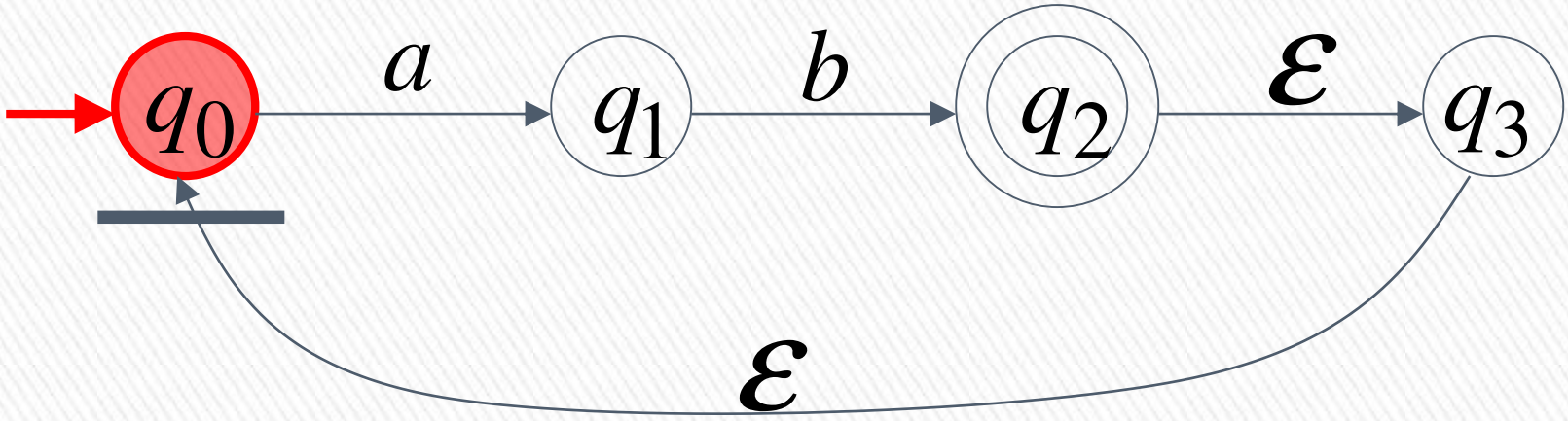

a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

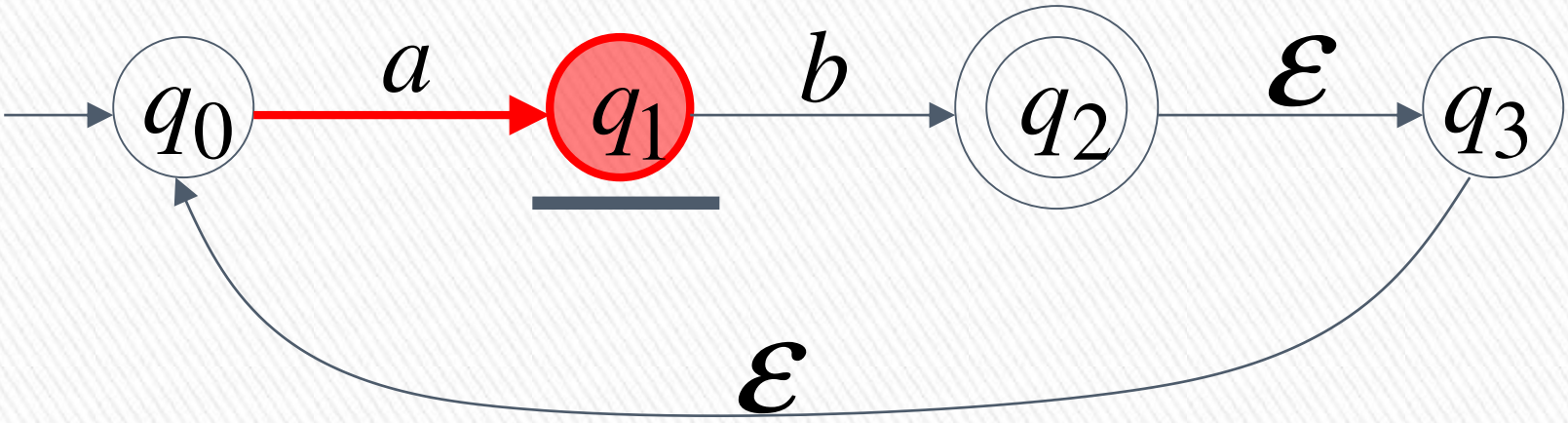

a
b


"accept"
q_0
a
q_1
b
q_2
ε
q_3
ε

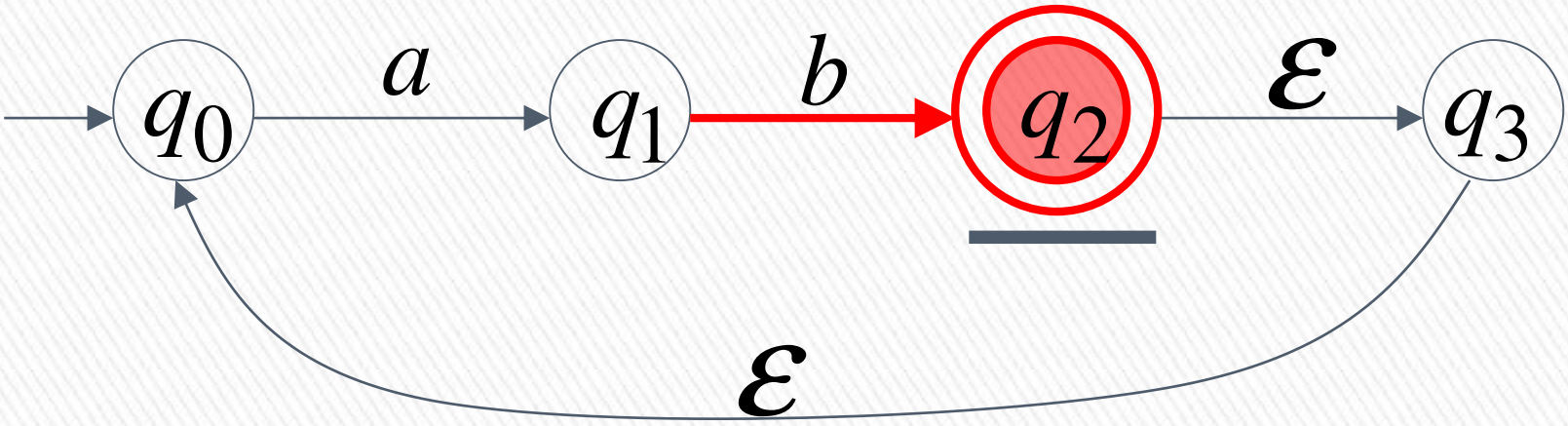
Another String



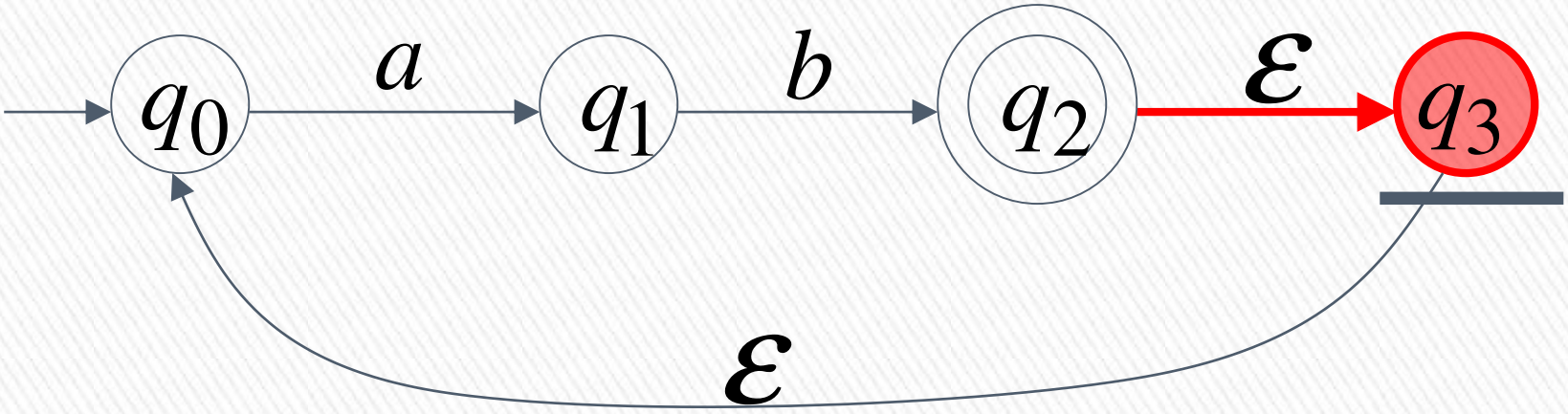

a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

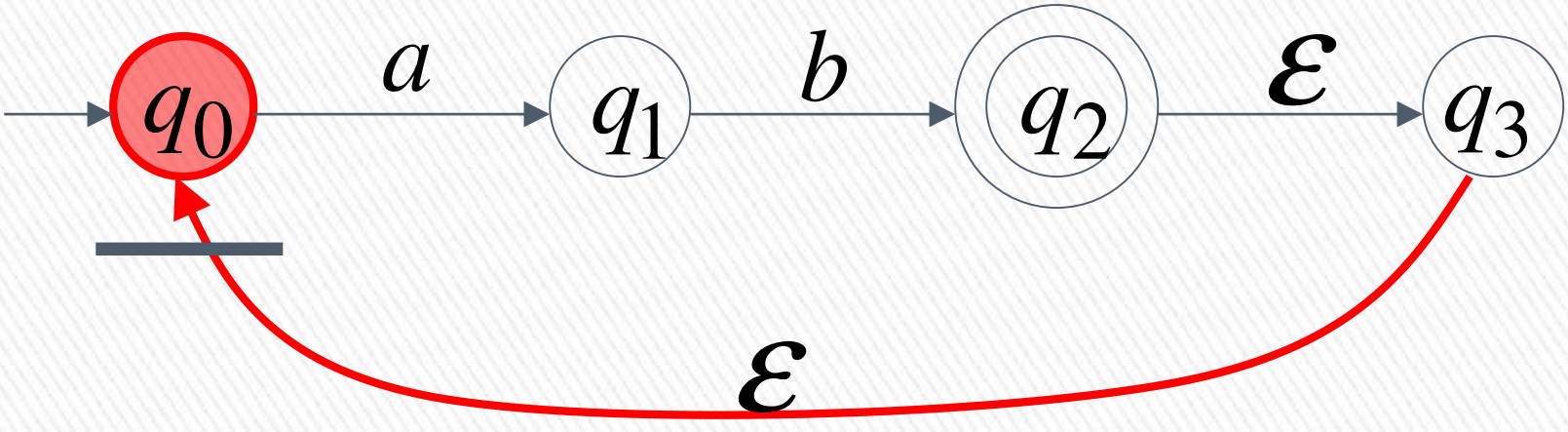

a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

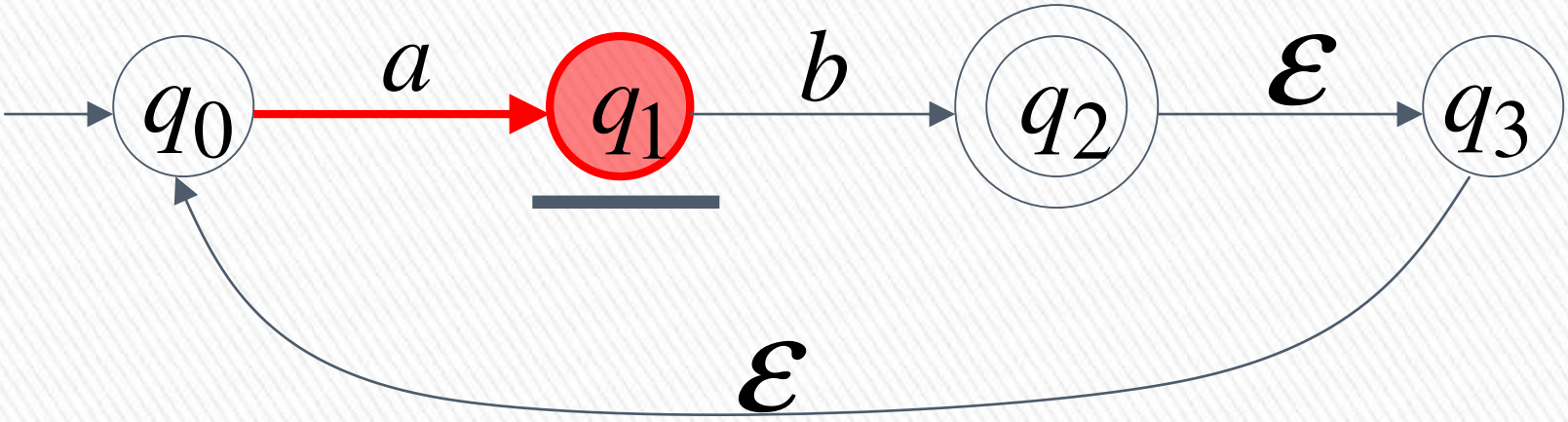

a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

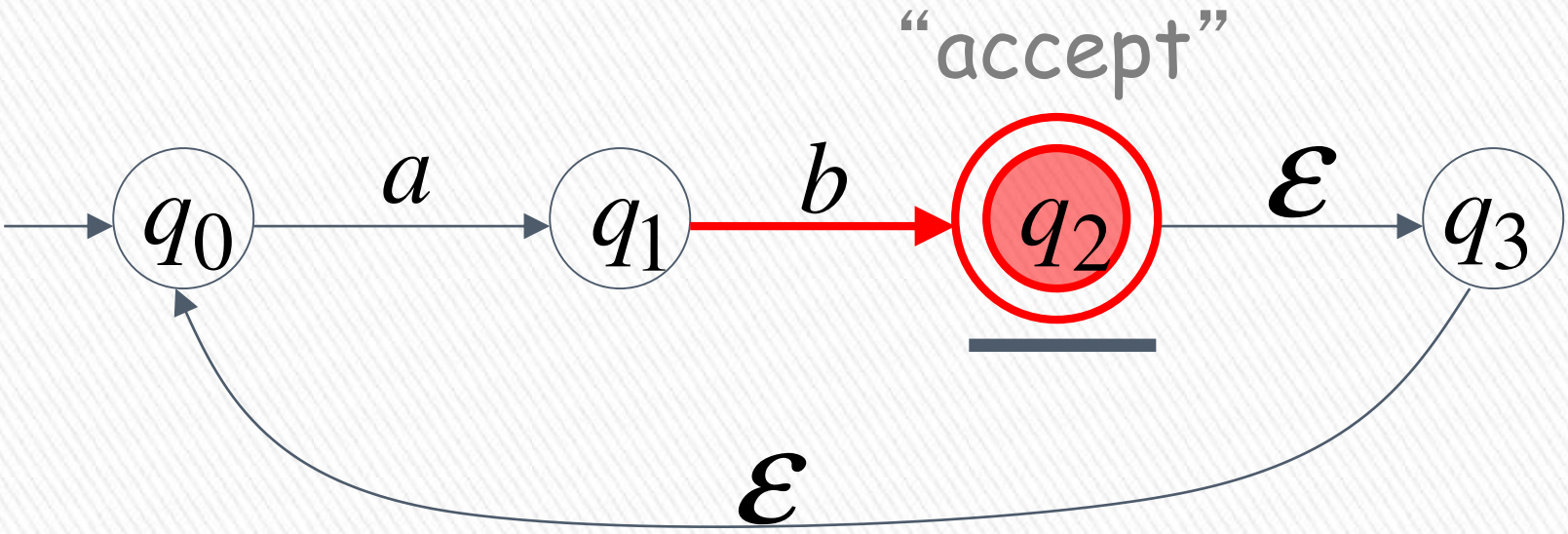

a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε


a
b
a
b


q_0
a
q_1
b
q_2
ε
q_3
ε

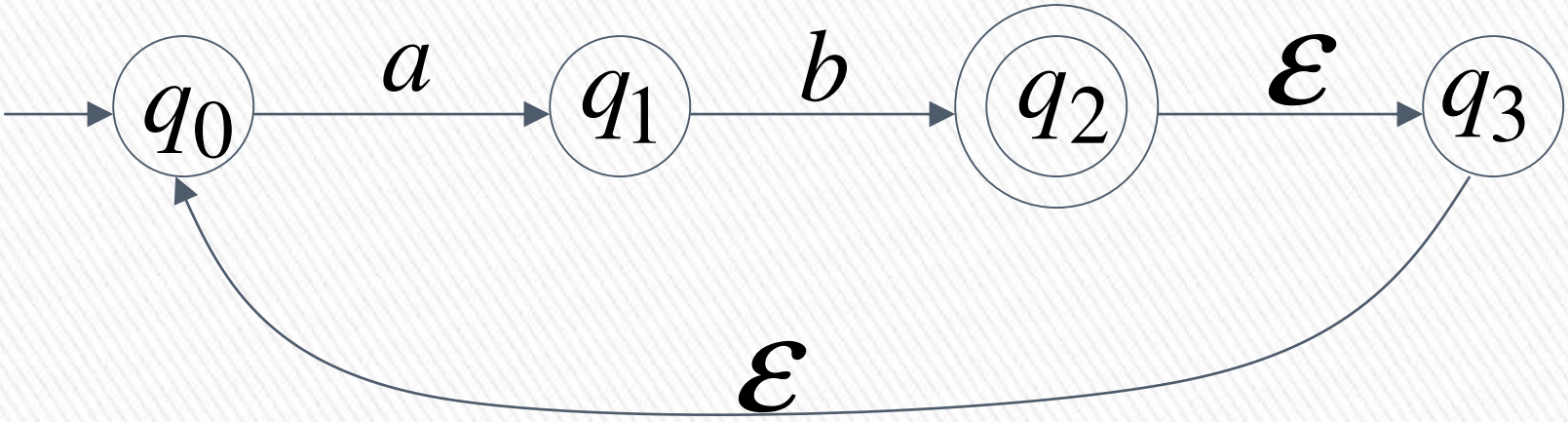

a
b
a
b


"accept"
q_0
a
q_1
b
q_2
ε
q_3
ε

Language accepted

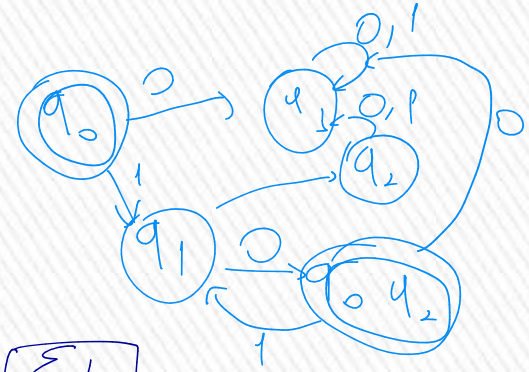
$$L = \{ab,\ abab,\ ababab,\ ...\}$$

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

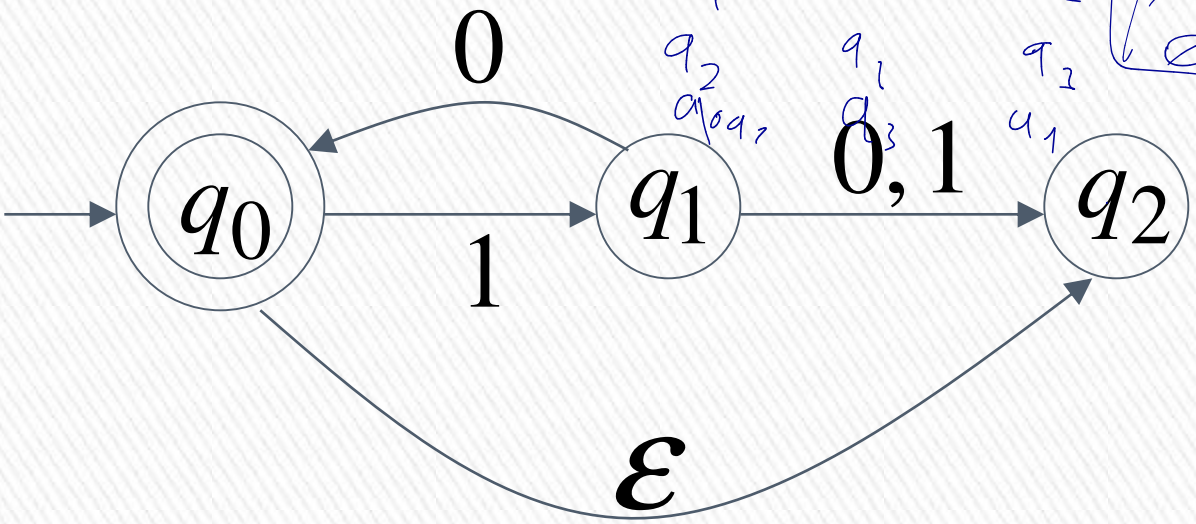


Another NFA Example

	0	1	ε
q_0	q_2	q_1	q_2
q_1	$q_0 q_2$	q_2	q_1
$q_0 q_2$	q_3	q_1	q_2

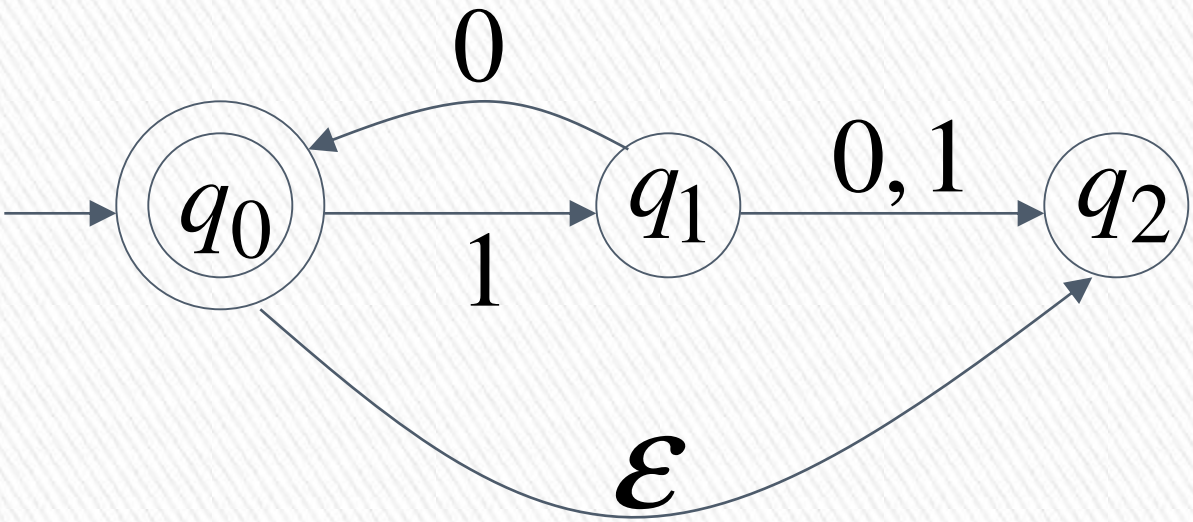


	0	1	ε
q_0	q_2	q_1	q_2
q_1	q_3 q_2	q_2	q_1
q_2	q_1	q_3	q_1
$q_0 q_1$	q_3	q_1	



Language accepted

$$L(M) = \{\varepsilon,\ 10,\ 1010,\ 101010,\ ...\}$$
$$= \{10\}^*$$

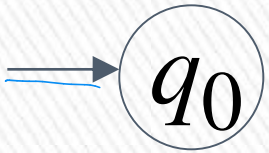


(redundant state)

Remarks:

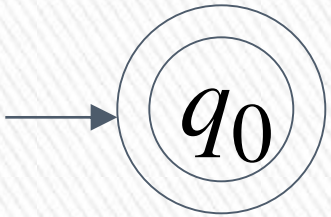
- The ε symbol never appears on the input tape
- Simple automata:

M_1



$L(M_1) = \{\}$

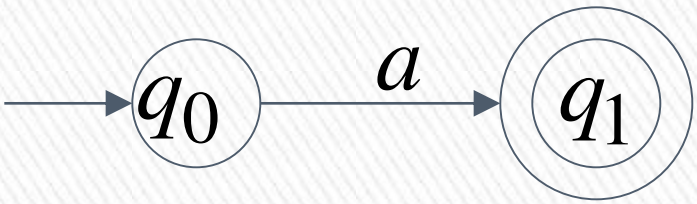
M_2



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

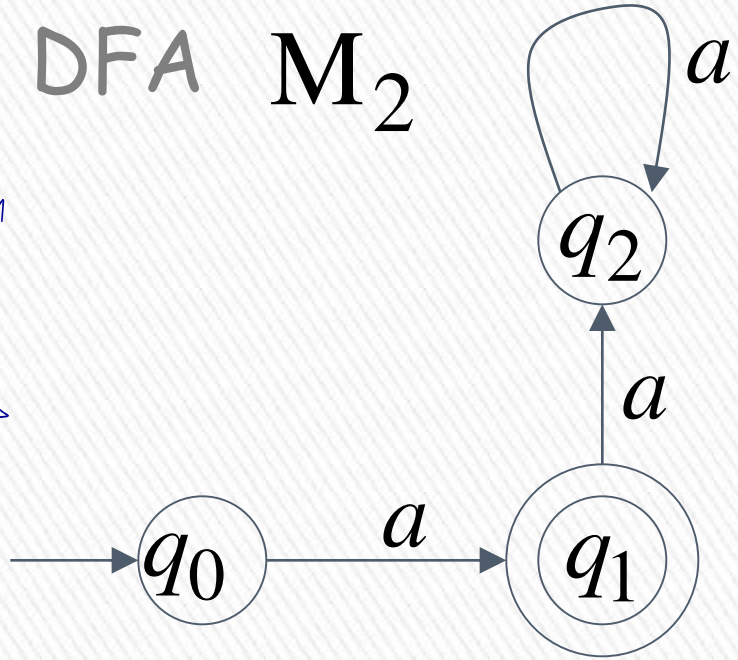
- NFAs are interesting because we can express languages easier than DFAs

NFA M_1



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

DFA M_2

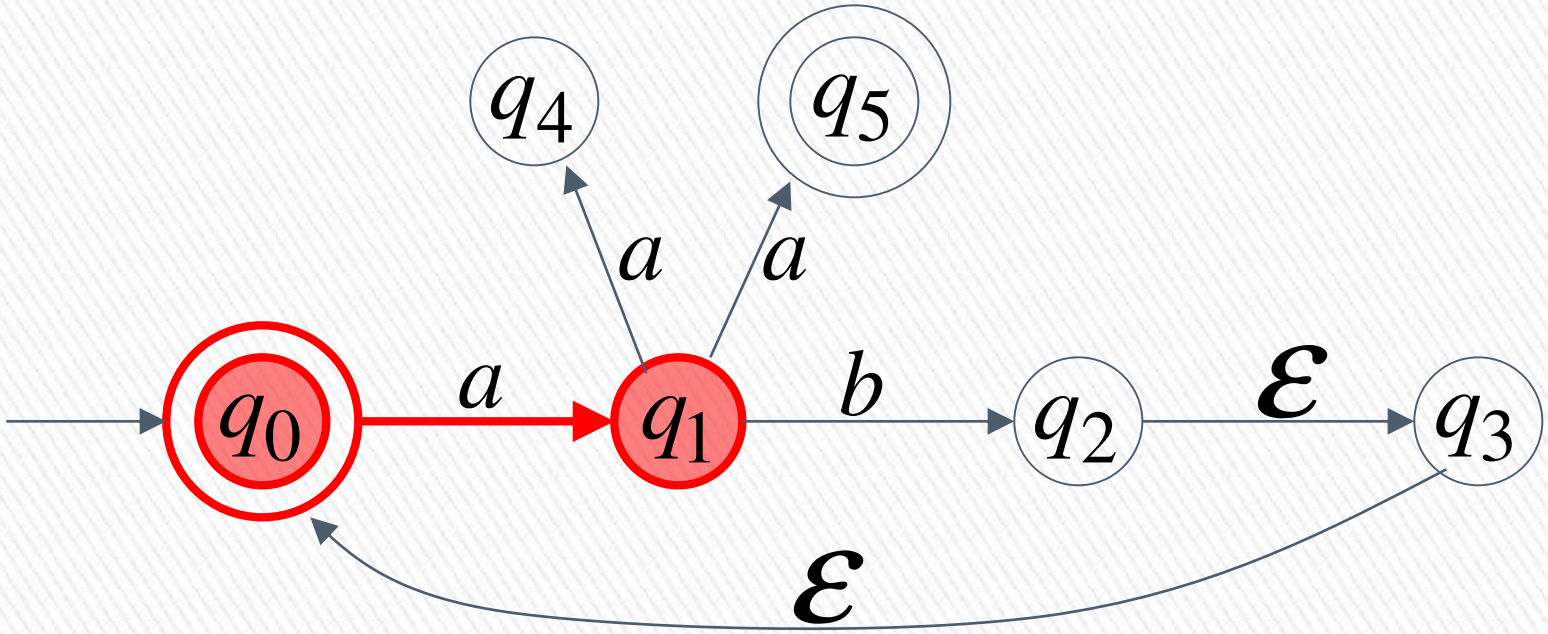


$L(M_2) = \{a\}$

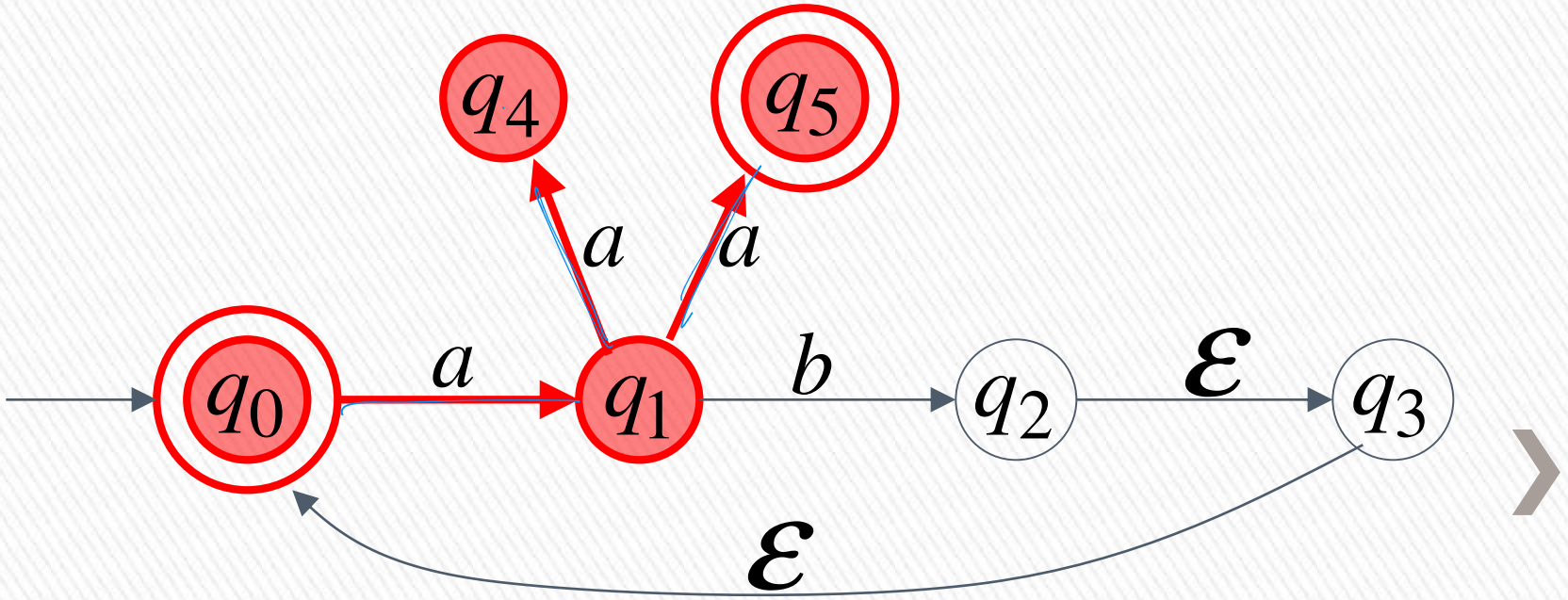
Extended Transition Function δ^*

Same with δ but applied on strings

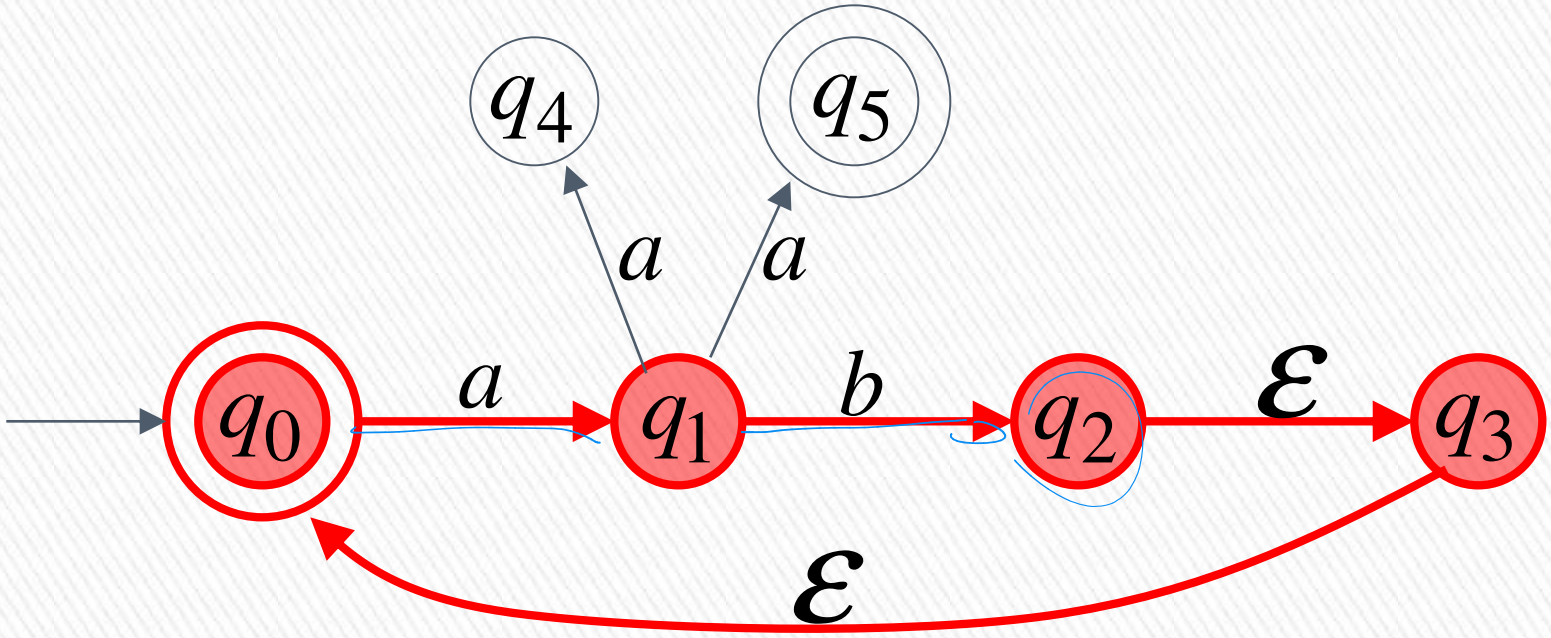
$$\delta^*(q_0, a) = \{q_1\}$$



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



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



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

- The language accepted by M is:

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

- where

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

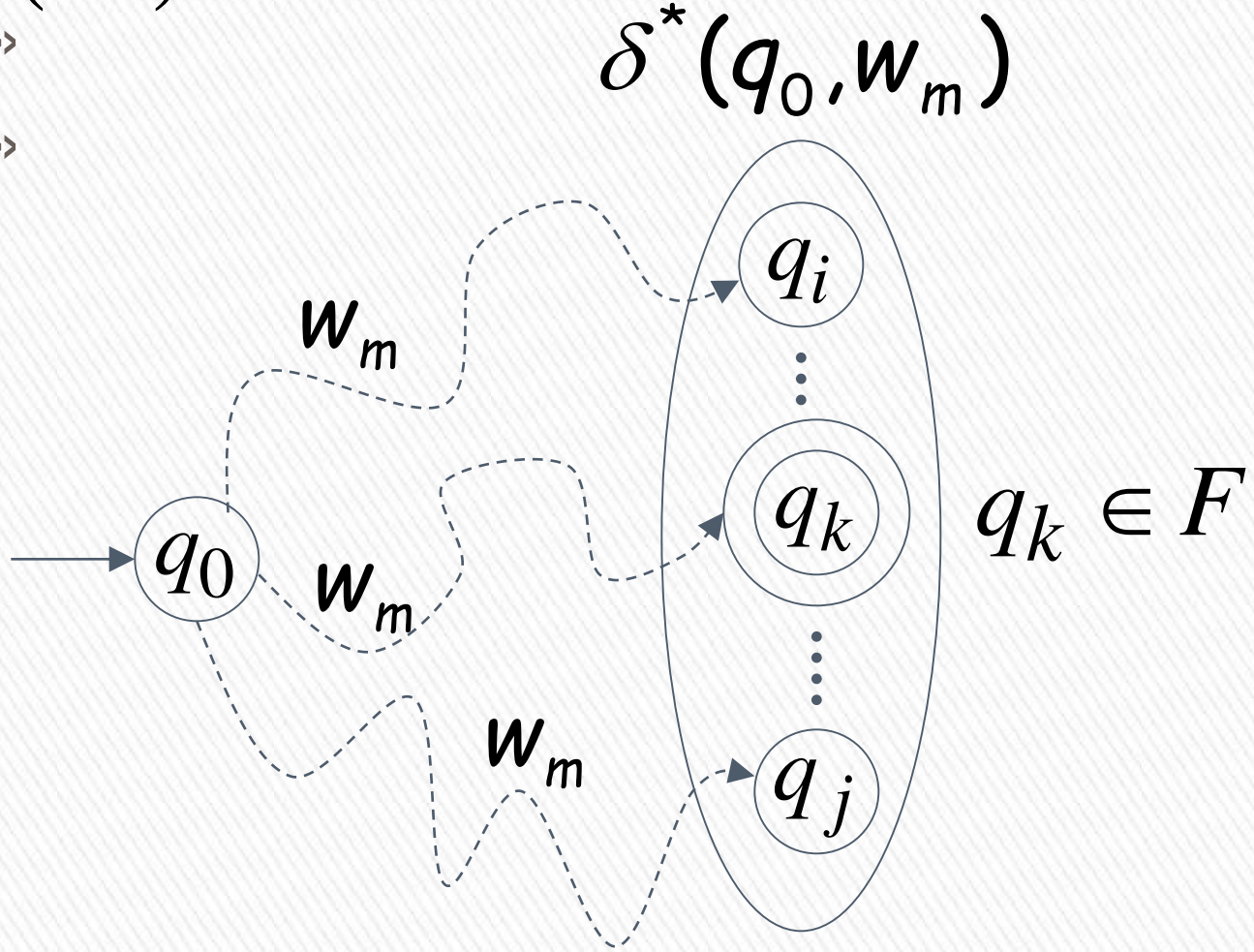
- and there is some $q_k \in F$

(accepting state)

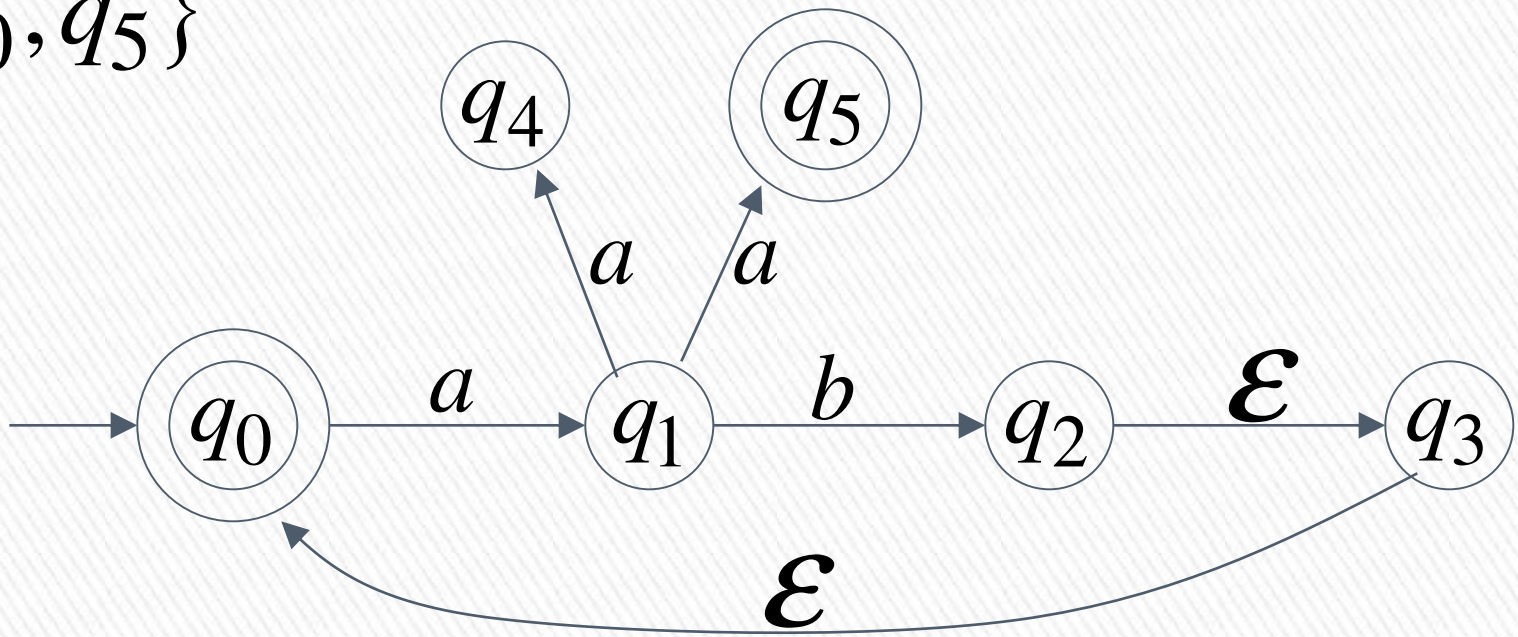
$w_m \in L(M)$

»

»

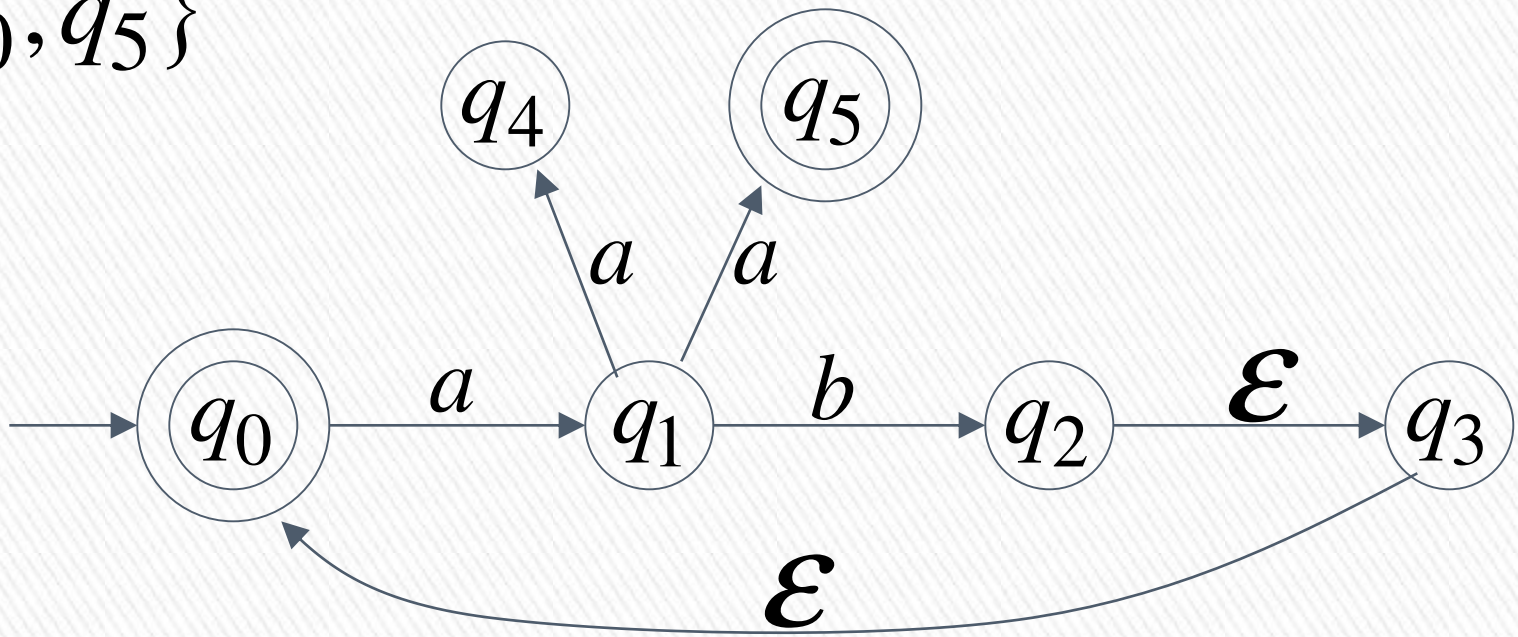


$F = \{q_0, q_5\}$



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

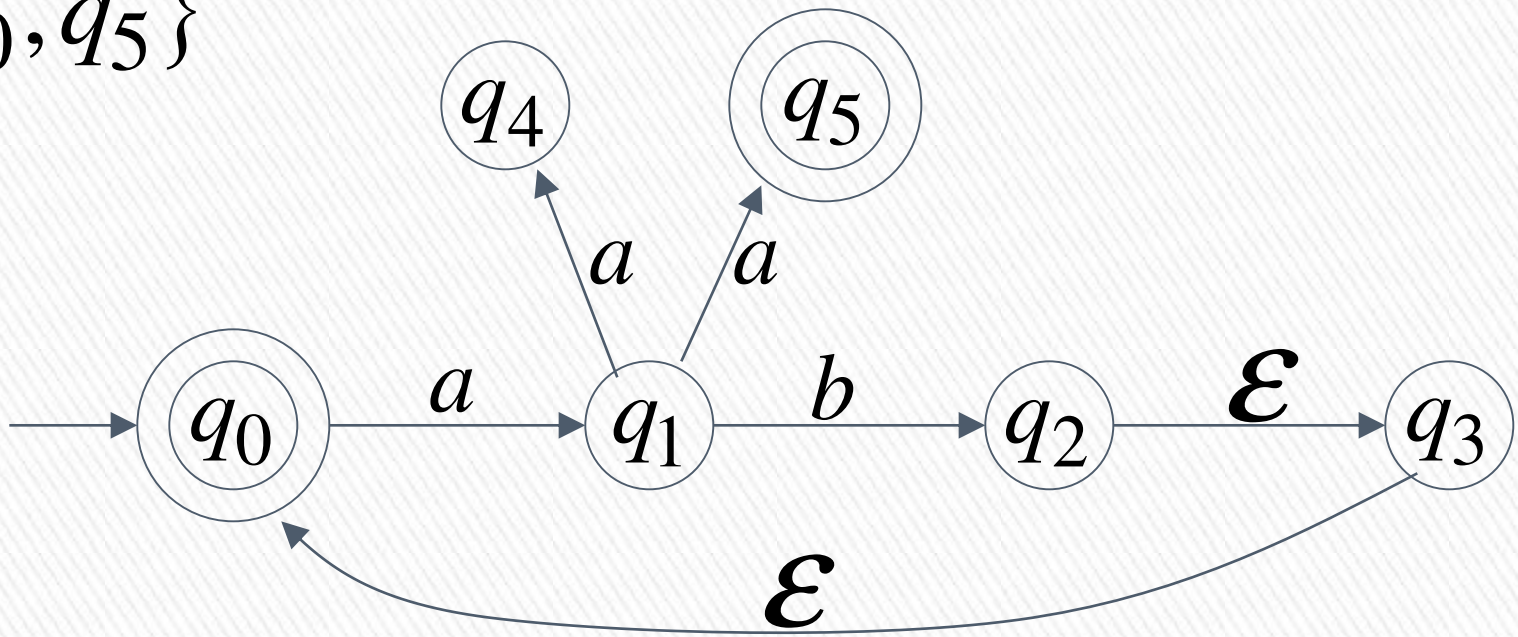
$$F = \{q_0, q_5\}$$



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

$$\underline{q_0} \in F$$

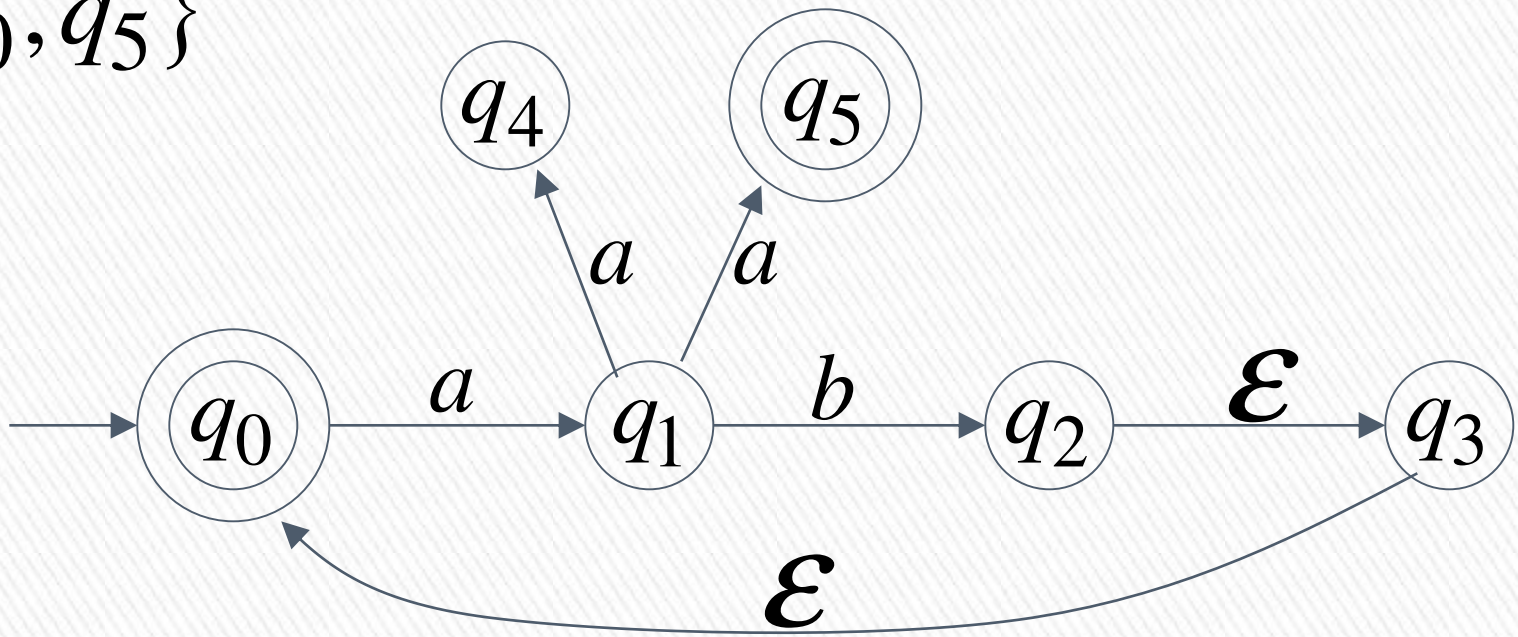
$F = \{q_0, q_5\}$



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

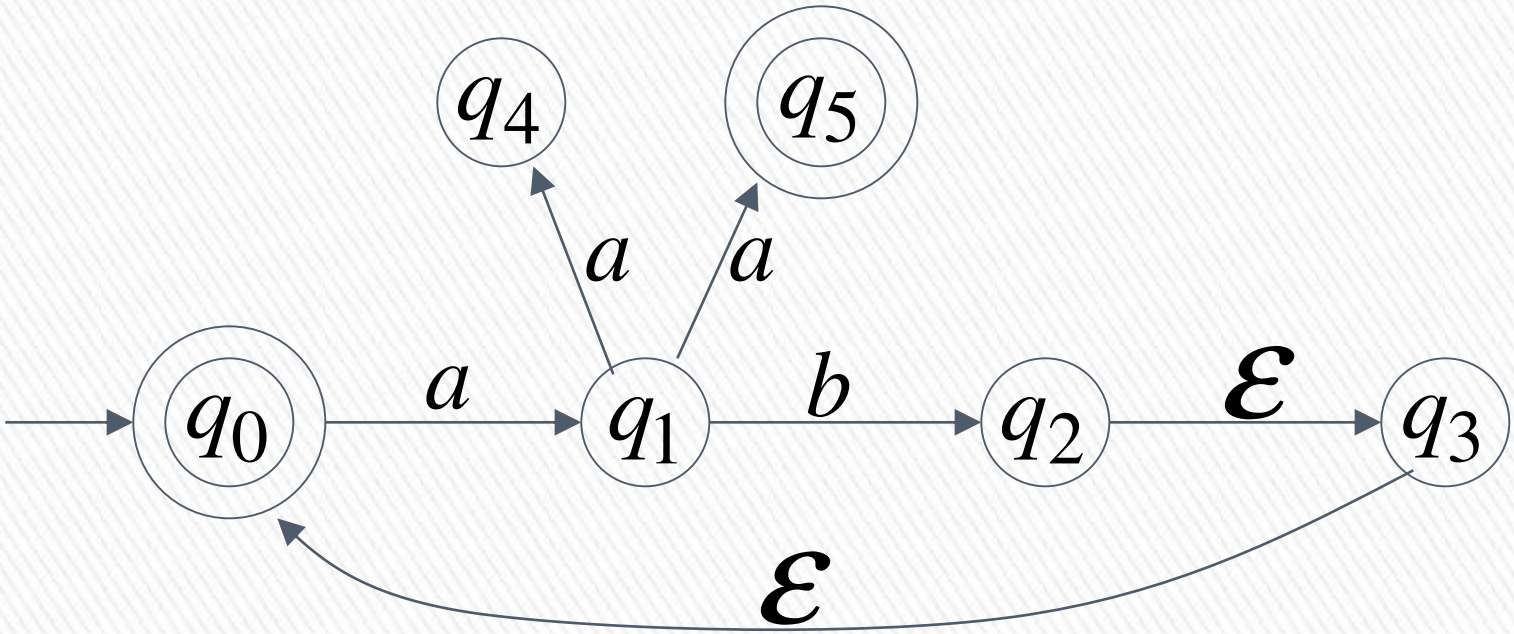
↘ $\in F$

$F = \{q_0, q_5\}$



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

$\notin F$



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

Equivalence of Machines

- Definition:

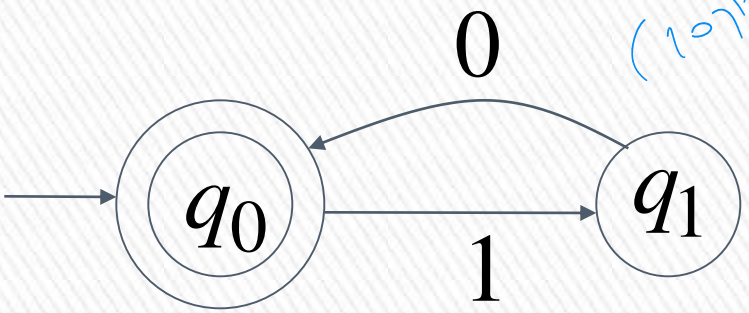
- Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

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

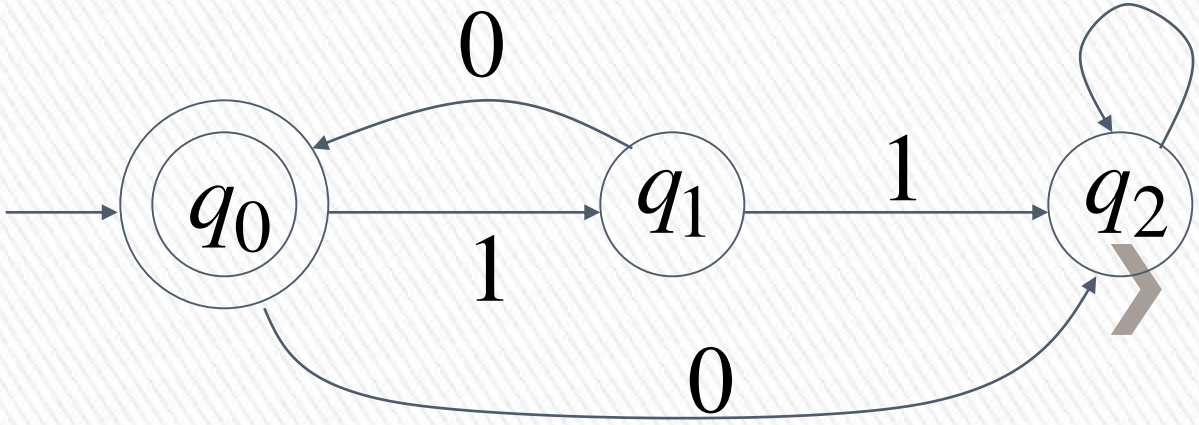
NFA M_1



2 tane verilen ve equivalent mi? sorulur bu da $L(M_1) = \{10\}^*$ verilip makinesi DFA ve NFA olarak istenir.

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

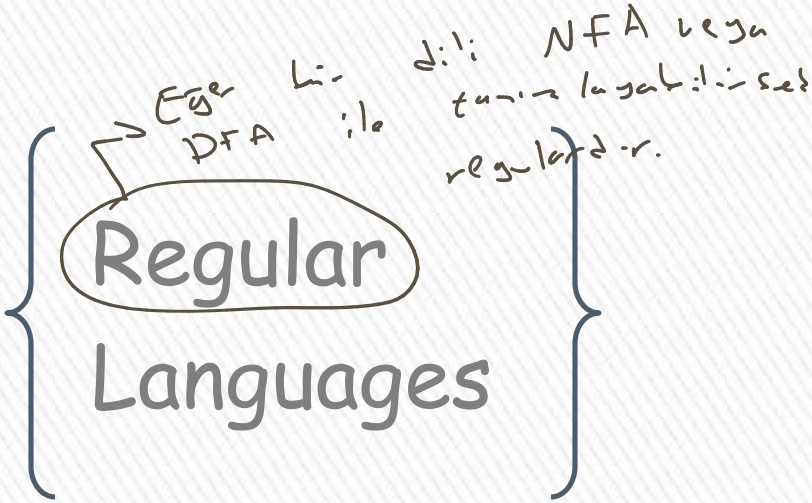
DFA M_2



NFAs accept Regular Languages

Theorem:

{ Languages accepted by NFAs } = { Regular Languages }



Languages Accepted by DFAs

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

Proof: we only need to show

NFA - DFA'e
for yield!..:

DFA de NFA'e

{ Languages accepted by NFAs } $\supseteq$ { Regular Languages }

AND

{ Languages accepted by NFAs } $\subseteq$ { Regular Languages }

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 an NFA

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

Proof-Step 2

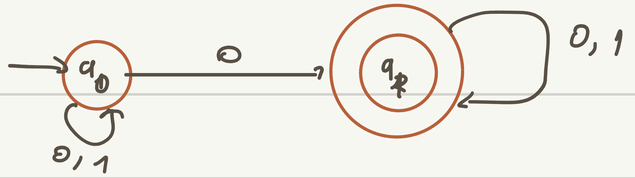
$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Any NFA can be converted to an equivalent DFA

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

›

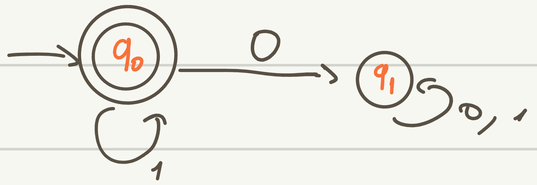
L_1 = { set of all strings that contain 0 }



$\Sigma = \{0, 1\}$ $Q = \{q_0, q_1\}$ q_0 initial state $F = \{q_1\}$

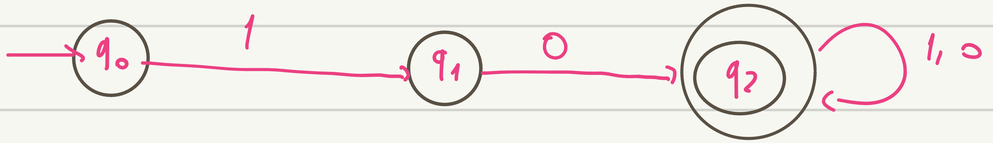
δ	0	1	ε
q_0	q_0, q_1	q_0	$\emptyset$
q_1	q_1	q_1	$\emptyset$

L_2 = { string not contain 0 }

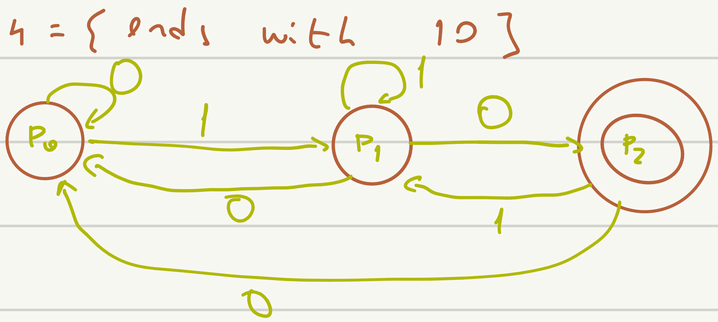


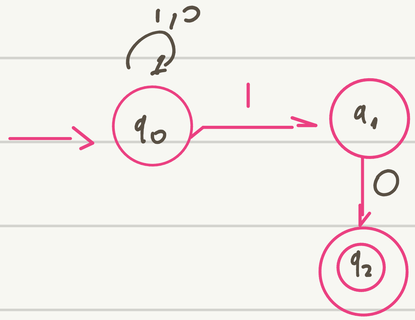
δ	0	1	ε
q_0	q_1	q_0	q_0
q_1	q_1	q_1	$\emptyset$

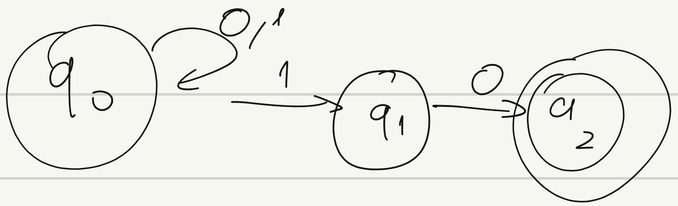
L_3 = { set of strings starts with 10 }



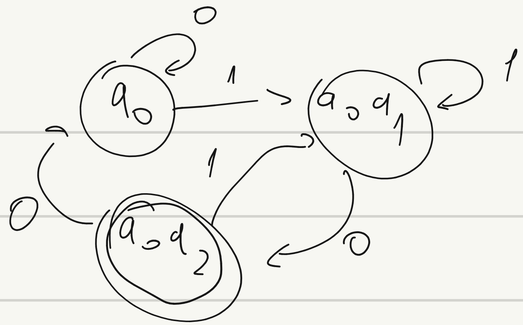
L_4 = { ends with 10 }

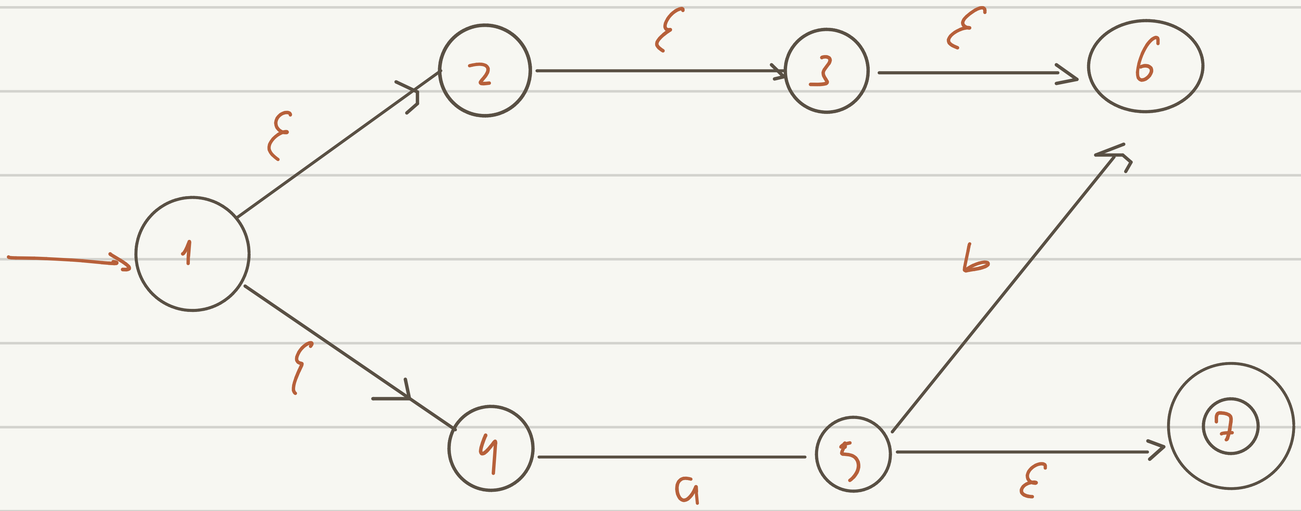






	0	1
q_0	q_0	q_0q_1
q_0q_1	q_0q_2	q_0q_1
q_0q_2	q_0	q_0q_1





$L = \{a, aa\}$

Lemma:

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

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

Proof:

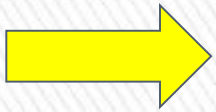
We only need to show: $L(M) \subseteq L(M')$

AND

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

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

We only need to prove:

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

NFA

Consider $w \in L(M)$

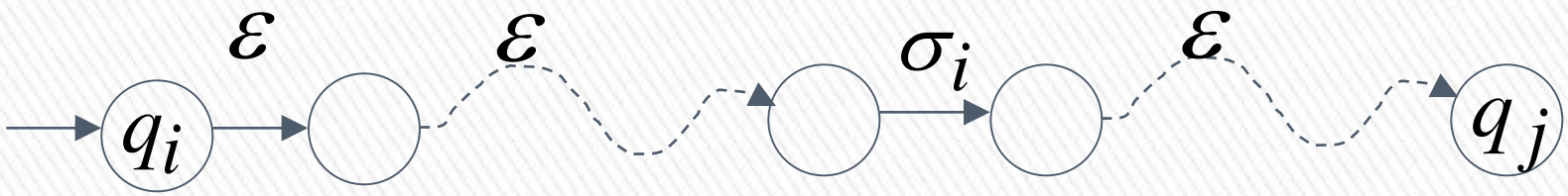
symbols

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

symbol

denotes a possible sub-path like

symbol



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

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

NFA M:

then

DFA M':

state label

$w \in L(M')$

state label

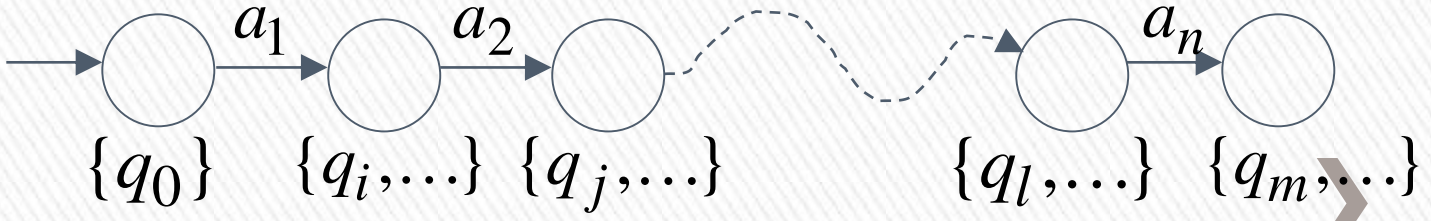
More generally, we will show that if in M:

(arbitrary string) $v = a_1 a_2 \cdots a_n$

NFA M:

then

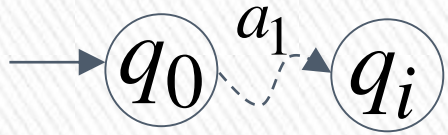
DFA M':



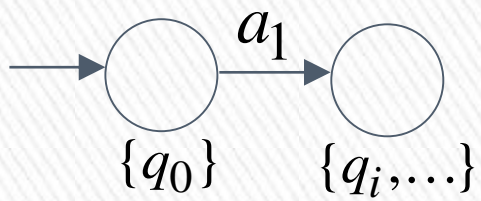
Proof by induction on $|v|$

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

NFA M:



DFA M':



is true by construction of M'

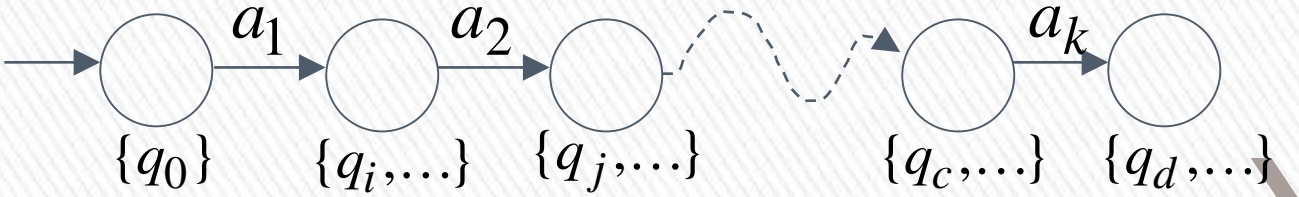
Induction hypothesis: $1 \leq |v| \leq k$

$$v = a_1 a_2 \cdots a_k$$

Suppose that the following hold

NFA M:

DFA M':

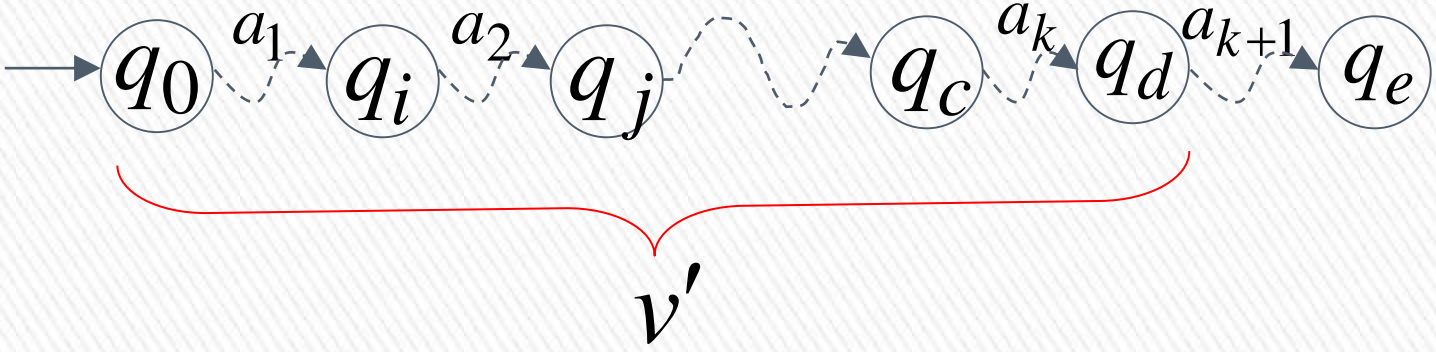


Induction Step: $|v| = k+1$

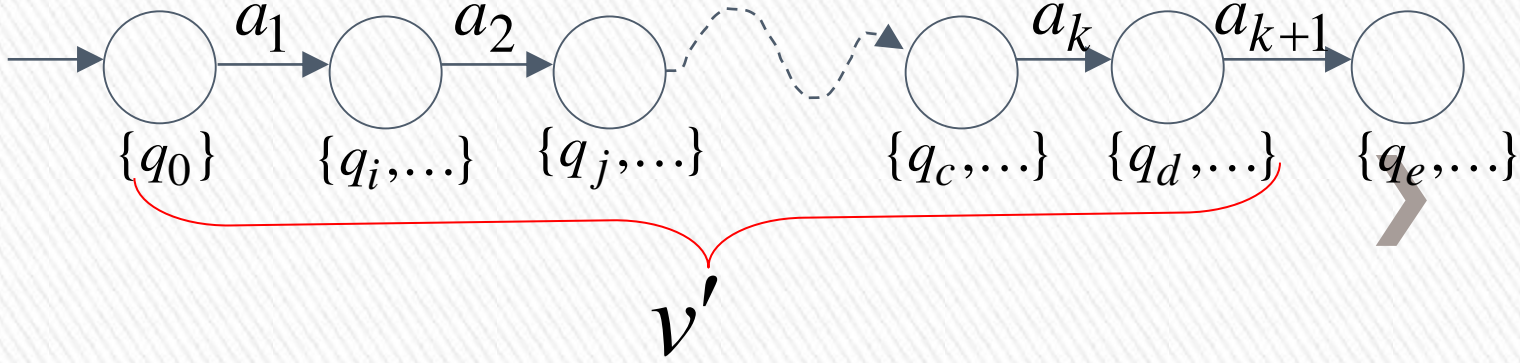
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

Then this is true by construction of M'

NFA M:



DFA M':



Therefore if $w \in L(M)$

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

NFA M:

then

DFA M':

$w \in L(M')$

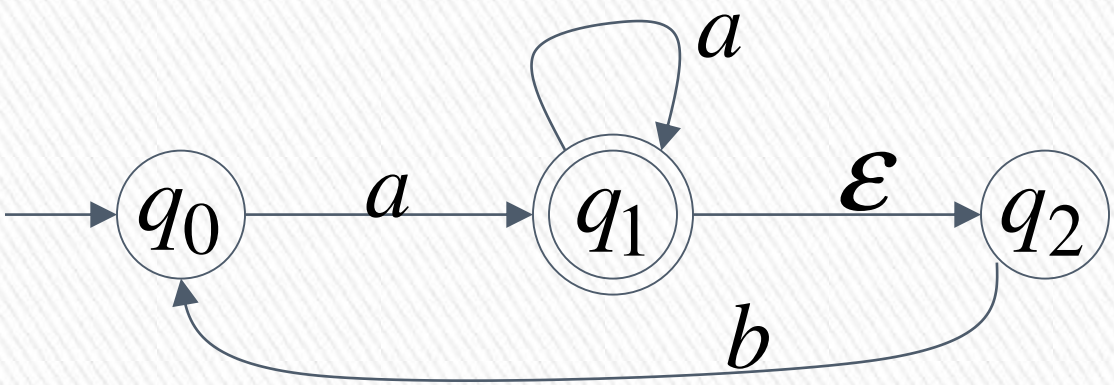
We have shown: $L(M) \subseteq L(M')$

With a similar proof
we can show: $L(M) \supseteq L(M')$

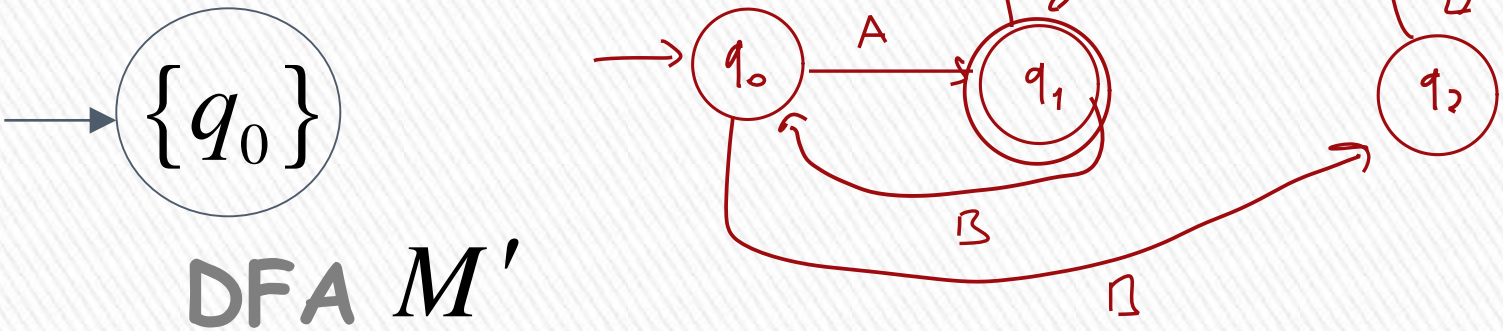
Therefore: $L(M) = L(M')$

END OF LEMMA PROOF

Conversion NFA to DFA

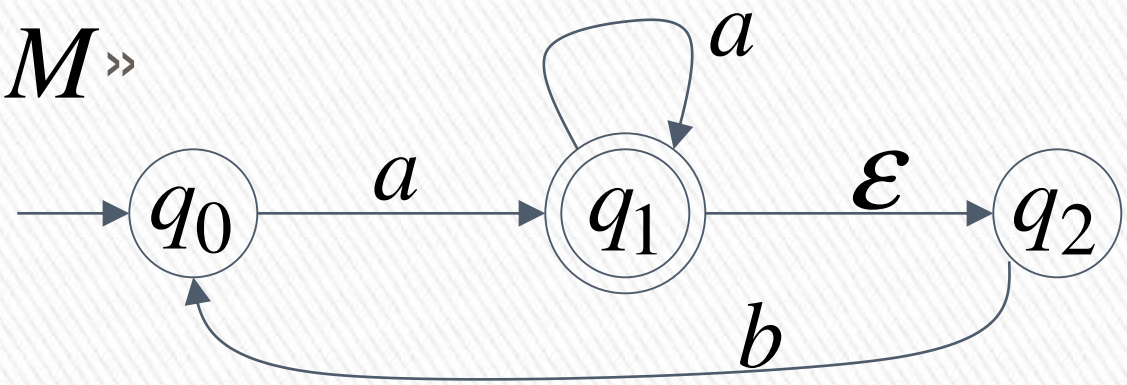


NFA M



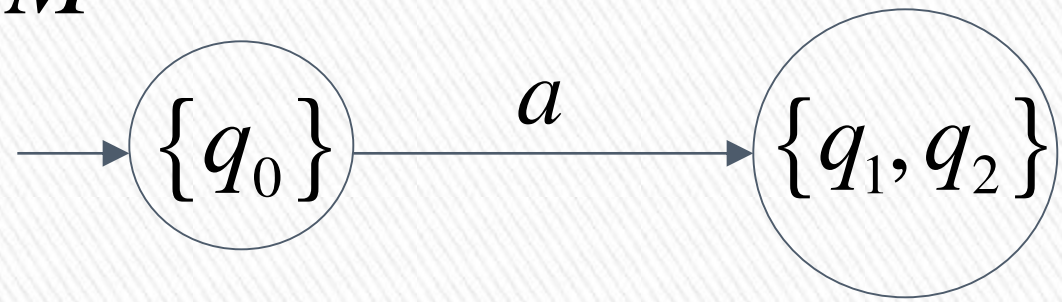
DFA M'

NFA M»



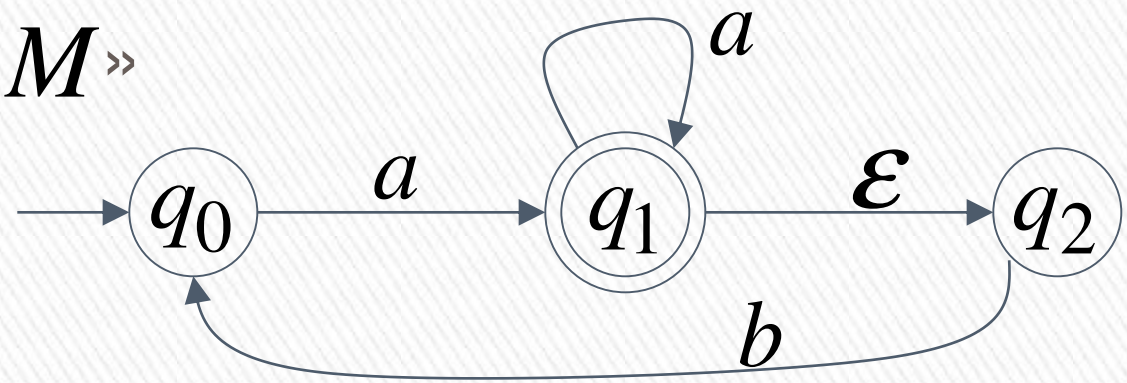
$$\delta^*(q_0, a) = \{q_1, q_2\}$$

DFA M'

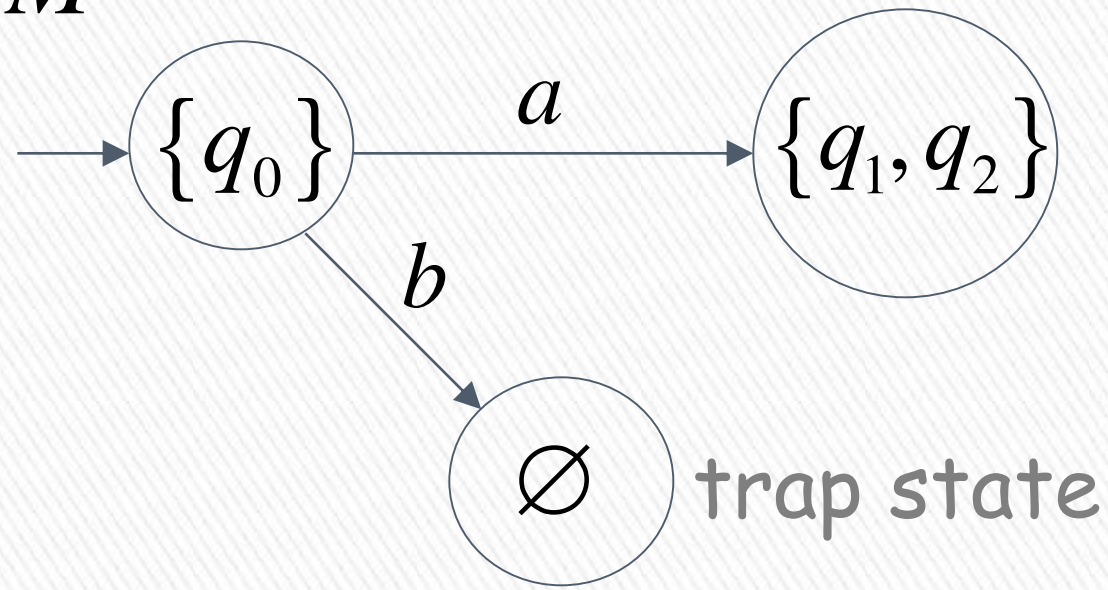


$\delta^*(q_0, b) = \varnothing$ empty set

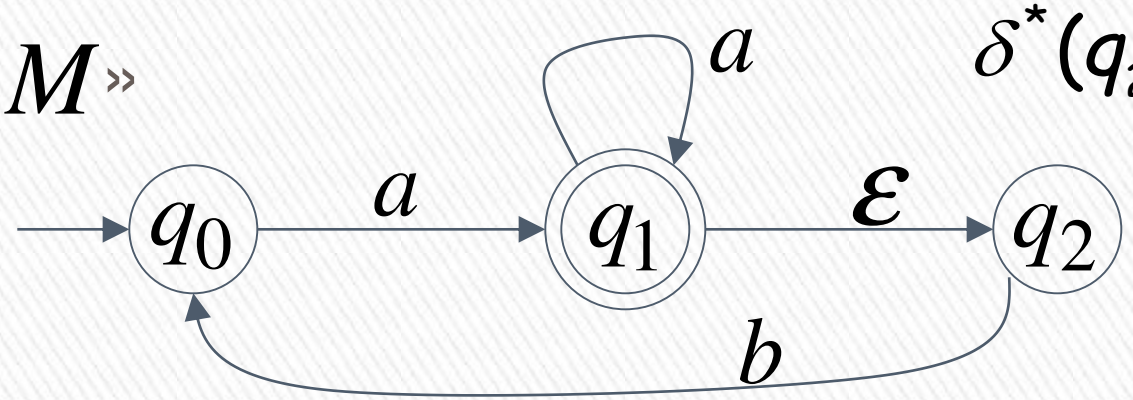
NFA M»



DFA M'



NFA M»



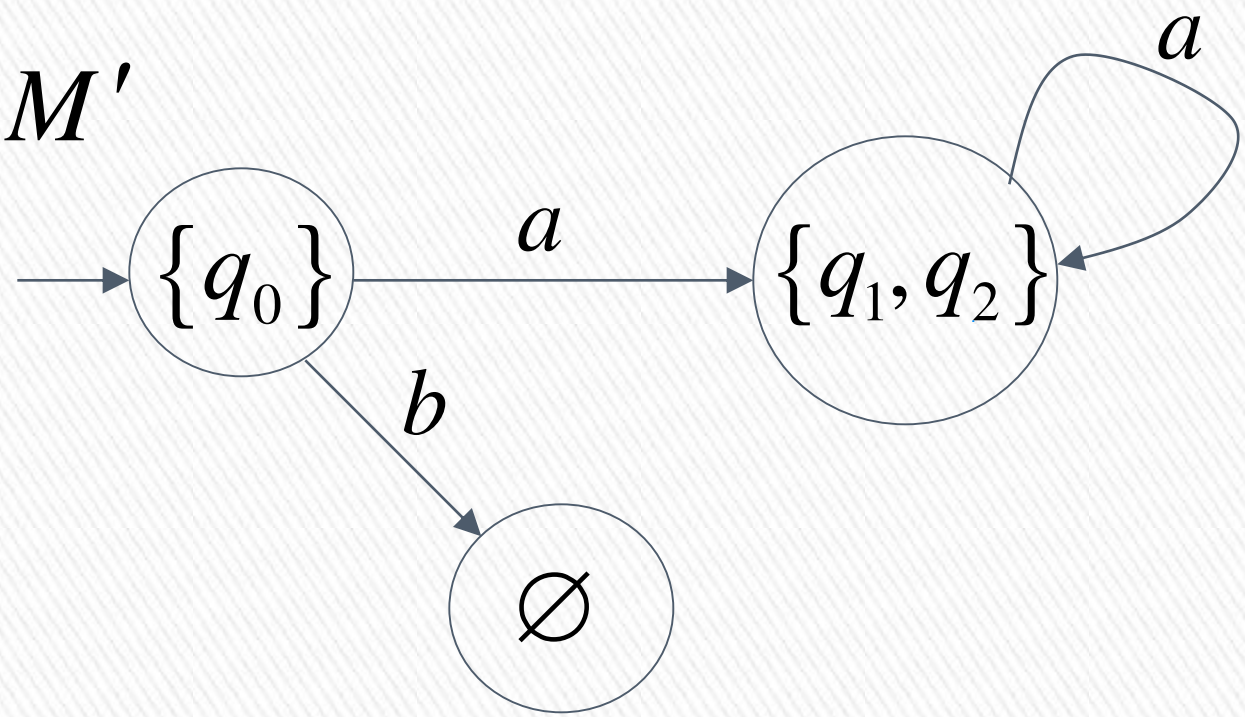
$\delta^*(q_1, a) = \{q_1, q_2\}$

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

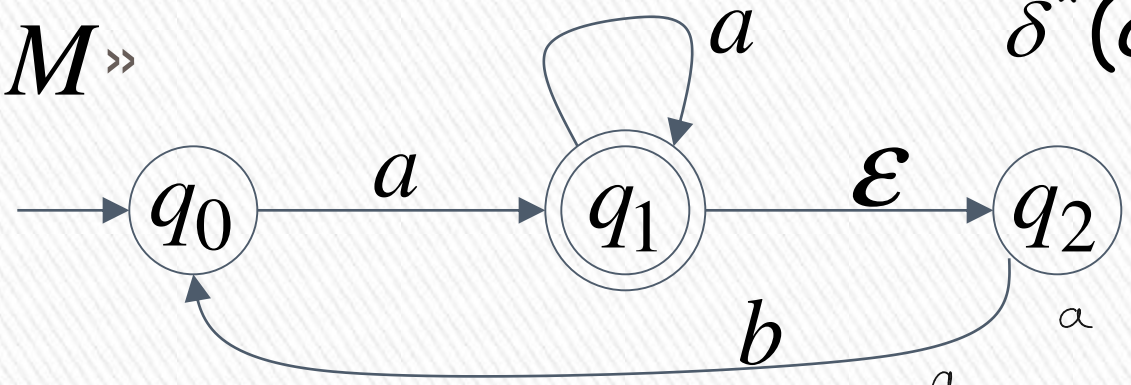
union

$\{q_1, q_2\}$

DFA M'



NFA M»



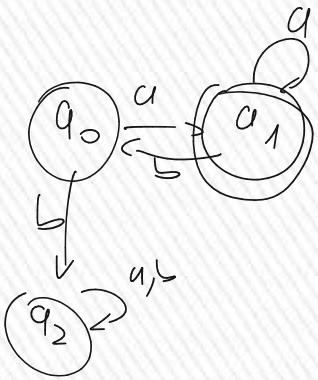
$\delta^*(q_1, b) = \{q_0\}$

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

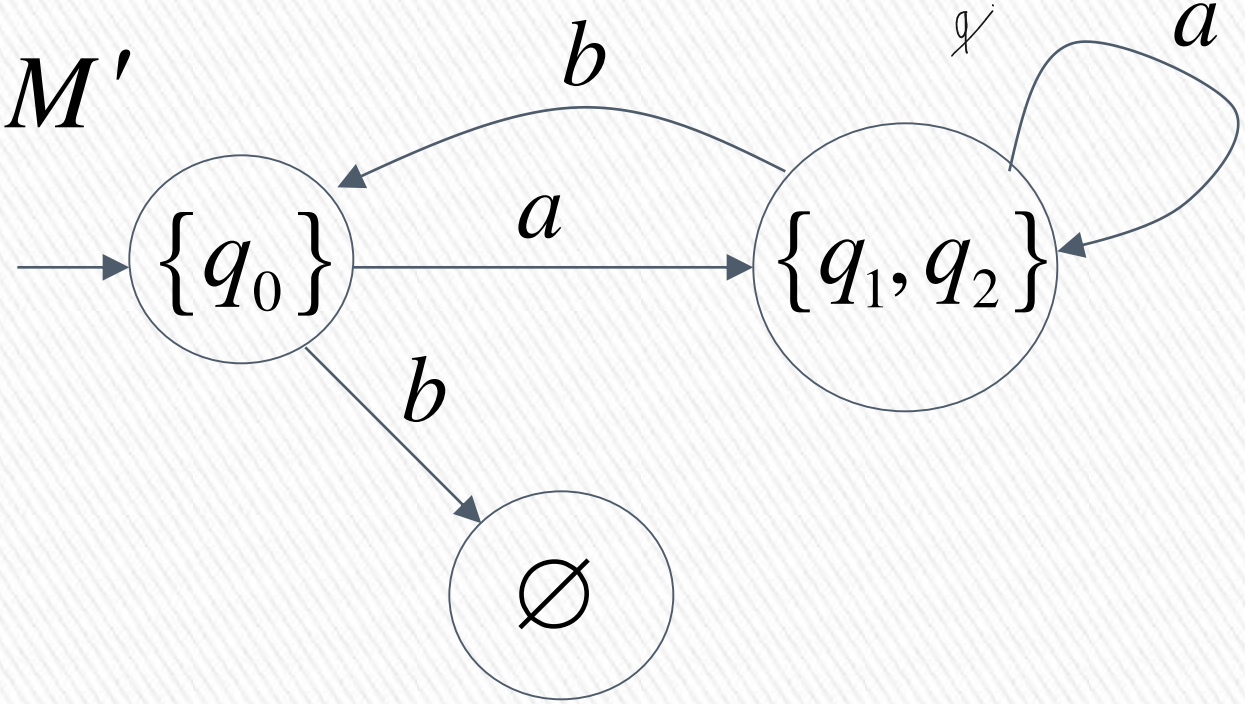
union

$\{q_0\}$

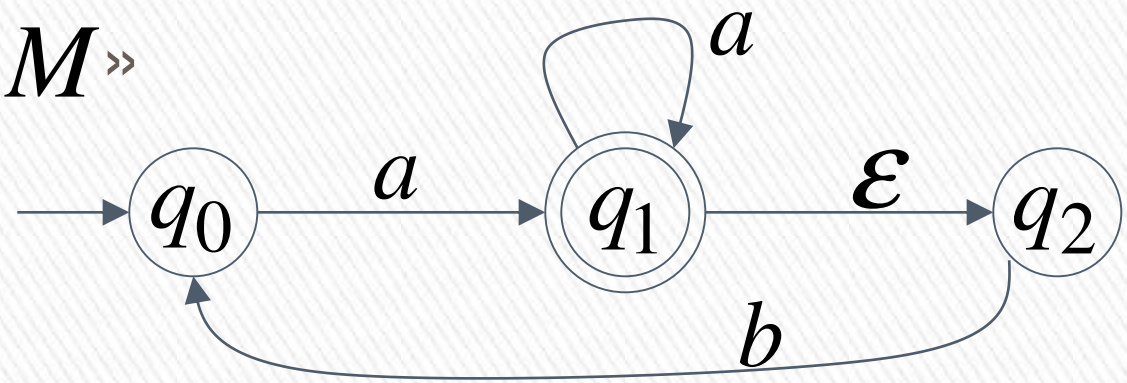
	a	b
q_0	q_1	$\emptyset$
q_1	q_1	q_0
q_2		



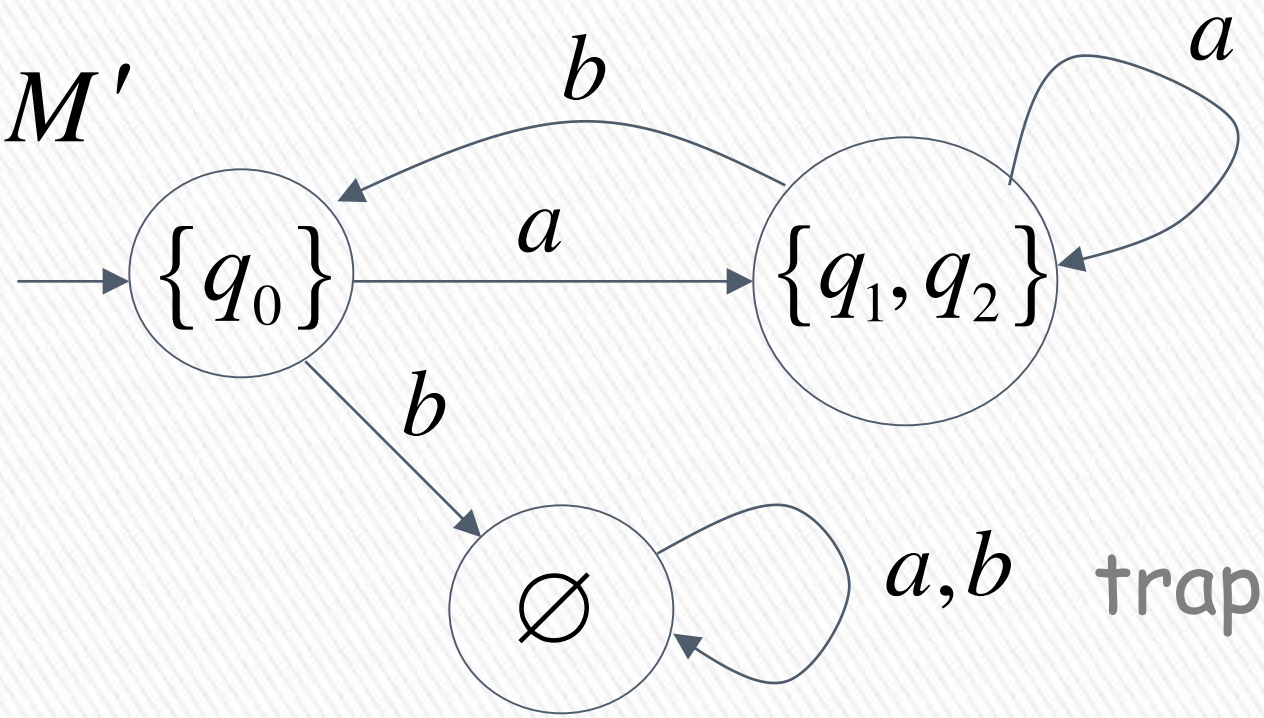
DFA M'



NFA M»



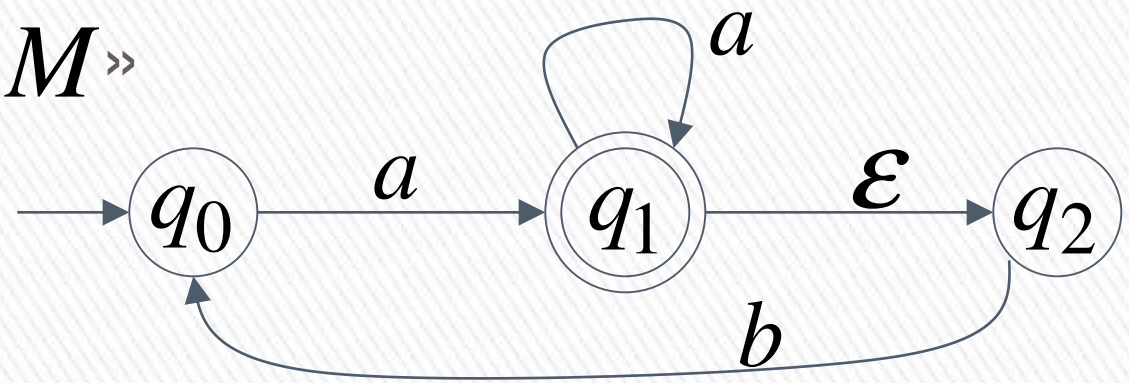
DFA M'



trap state

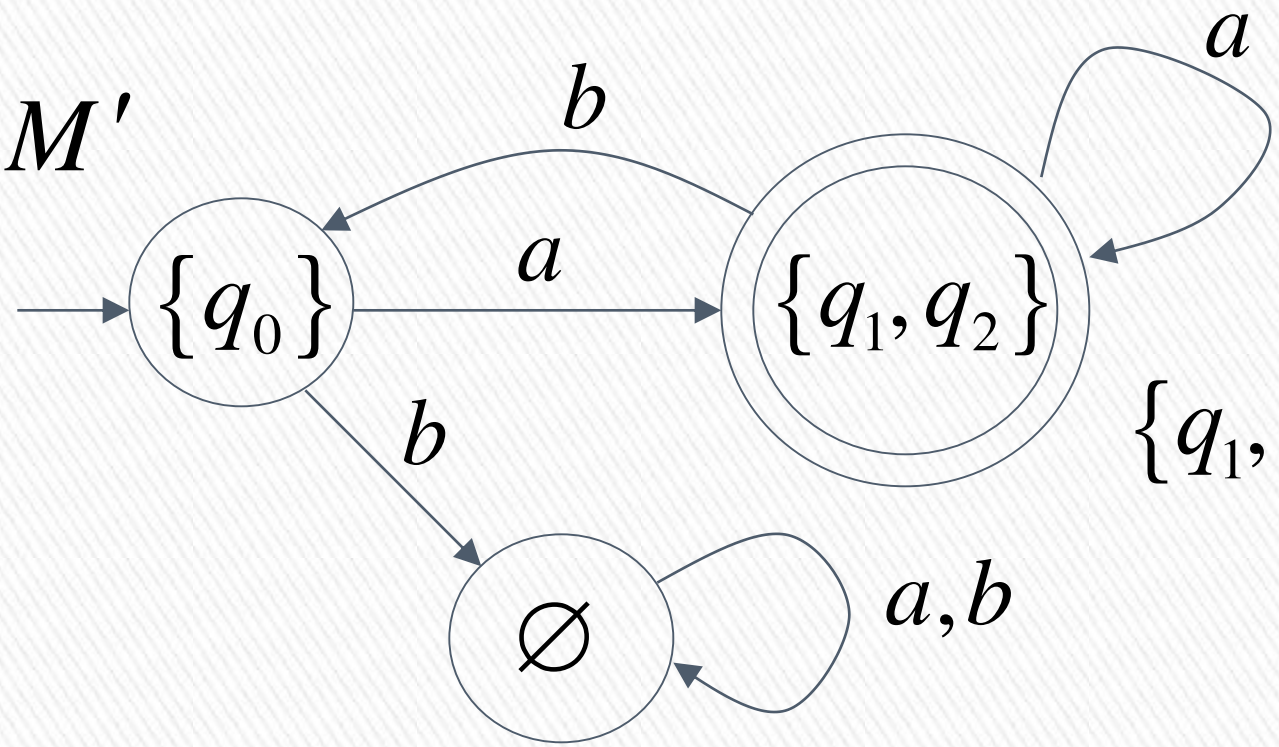
END OF CONSTRUCTION

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

$$\varnothing, \{q_0\}, \{q_1\}, \{q_0, q_1\}, \{q_1, q_2, q_3\}, \ldots.$$

Conversion Procedure Steps

»

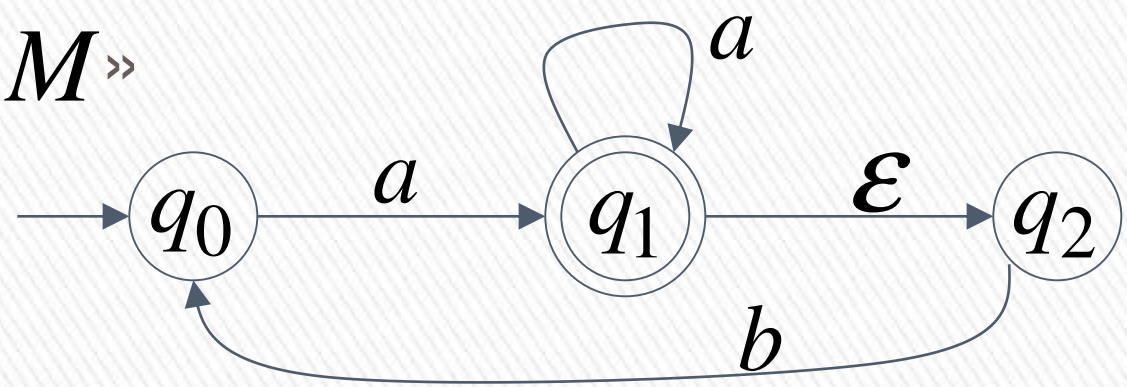
Step 1

» Initial state of NFA: q_0

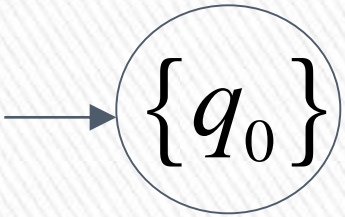
» Initial state of DFA: $\{q_0\}$

Example

NFA M»



DFA M'



Step 2

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

compute in the NFA

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

Union

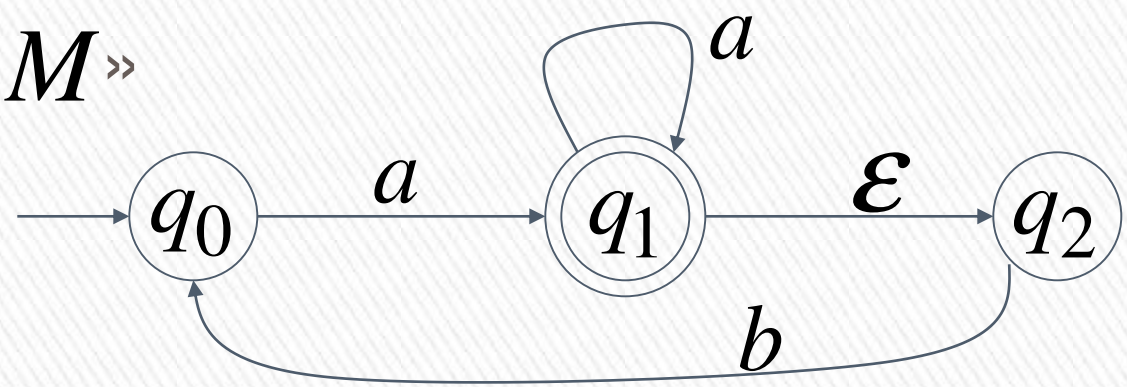
add transition to DFA

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

Example

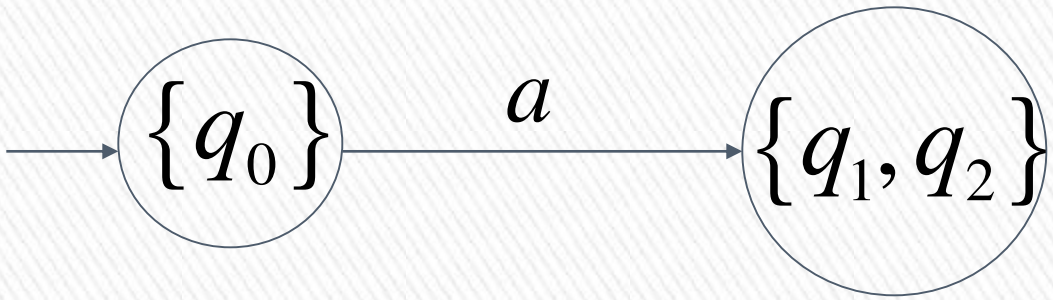
$\delta^*(q_0, a) = \{q_1, q_2\}$

NFA M»



DFA M'

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

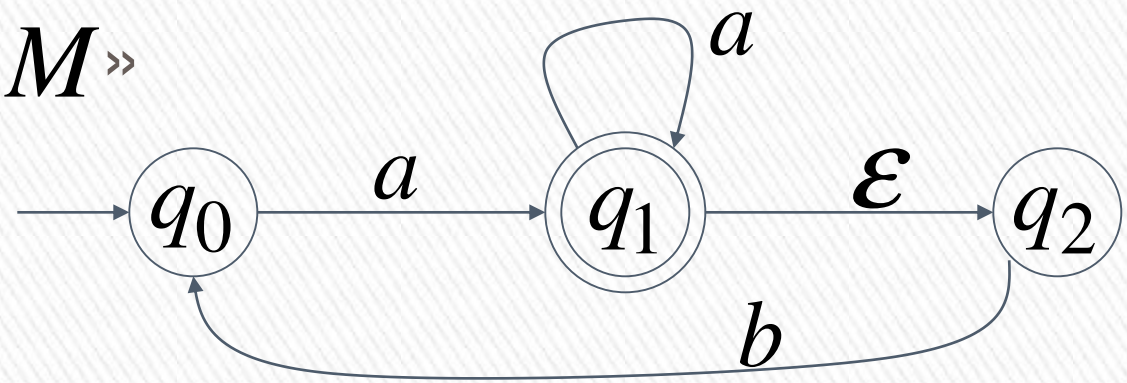


Step 3

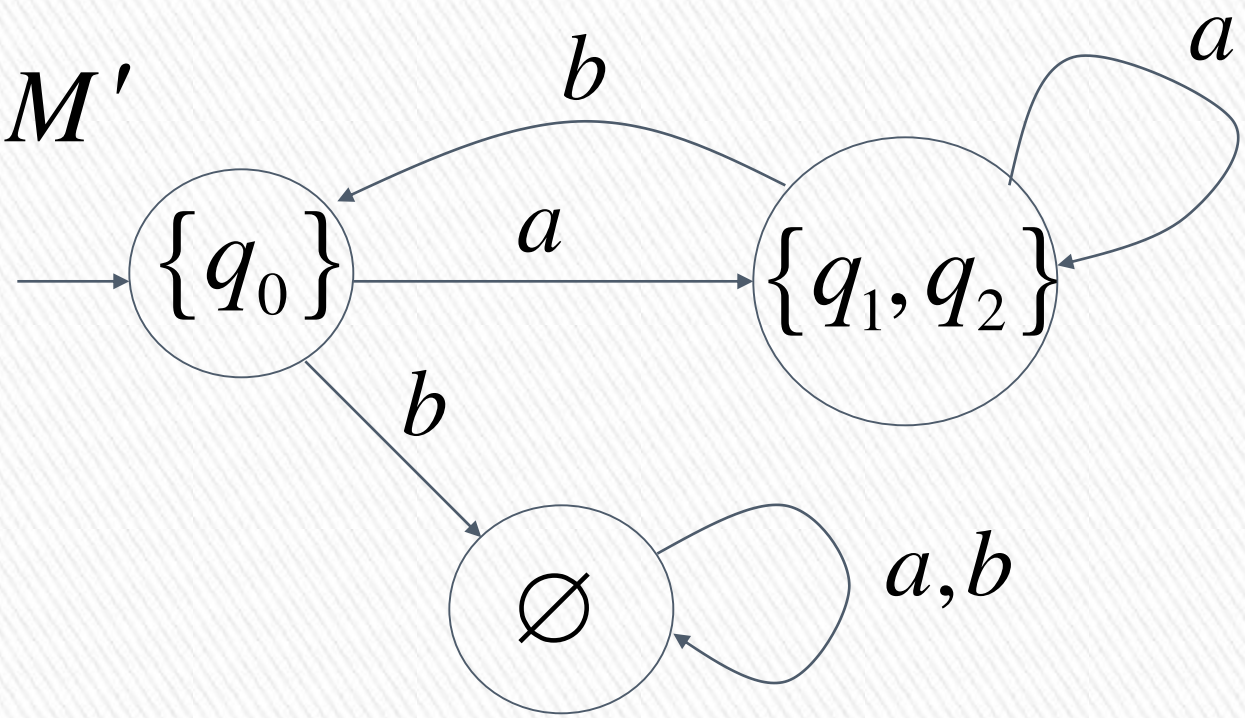
- Repeat Step 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA

Example

NFA M»



DFA M'



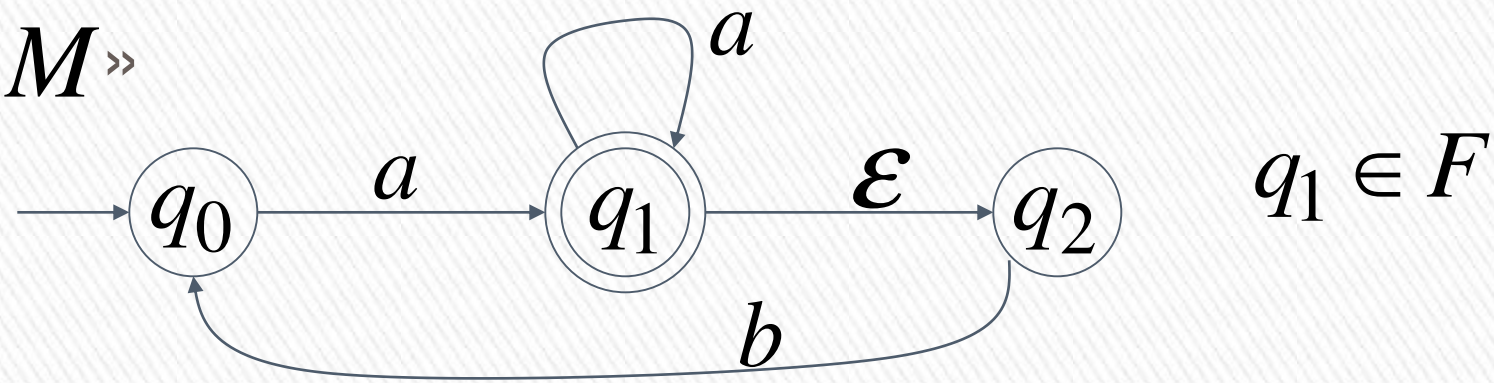
Step 4

- 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

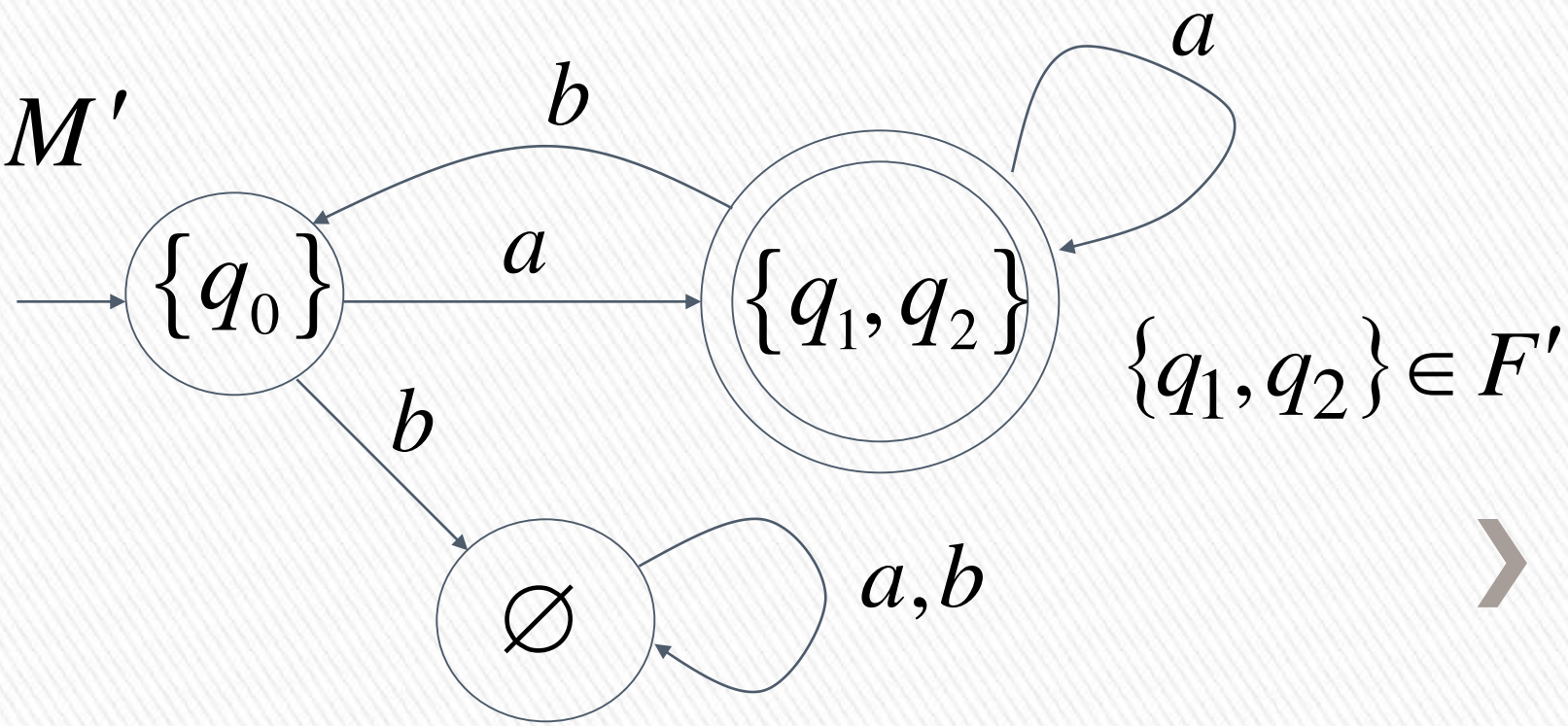
↳ içinde accept state olanların hepsini final state yap ★★★

Example

NFA M»

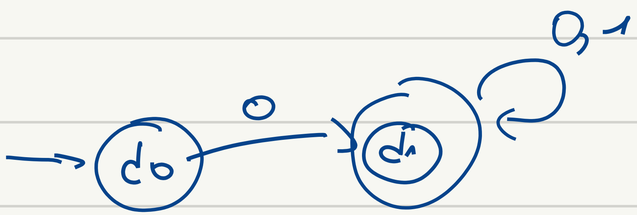


DFA M'



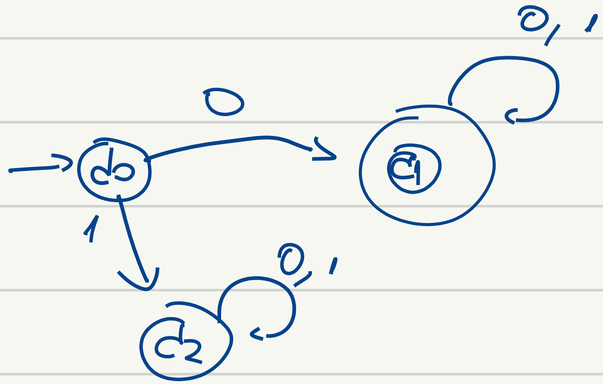
-An NFA accepts all strs over the alphabet $\Sigma = \{0, 1\}$ that begins with 0. Draw nfa, dfa

NFA



	0	1
d_0	d_1	$\emptyset$
d_1	d_1	d_1

DFA

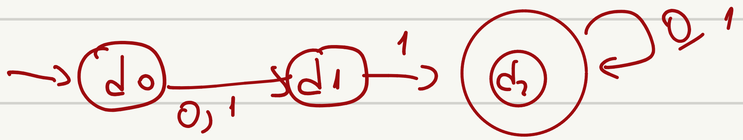


	0	1
d_0	d_1	d_2
d_1	d_1	d_1
d_2	d_2	d_2

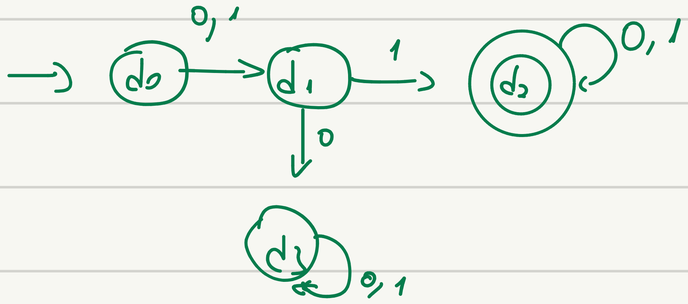
	0	1
d_0	d_1	d_2
d_2	d_2	d_2
d_1	d_1	d_1

- Design an NFA for the language that accepts strings over {0,1} second symbol 2.

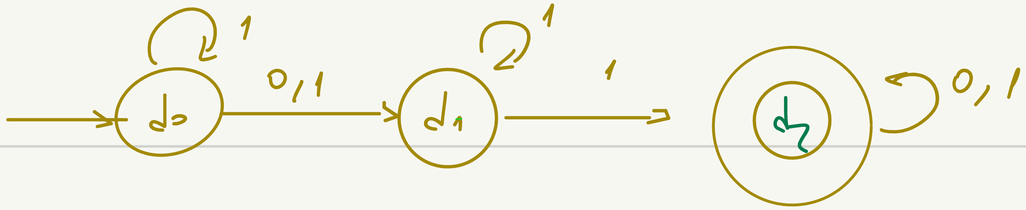
NFA



	0	1
d_0	d_1	d_1
d_1	$\emptyset$	d_2
d_2	d_2	d_2

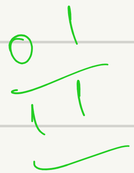


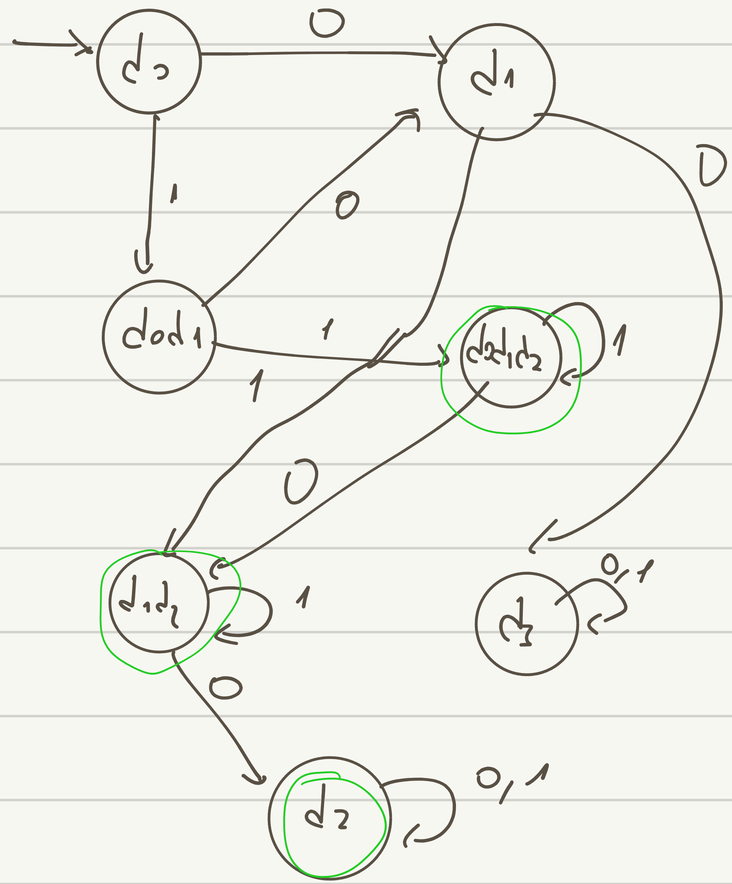
	0	1
d_0	d_1	d_1
d_1	d_3	d_2
d_2	d_2	d_2
d_3	d_3	d_3



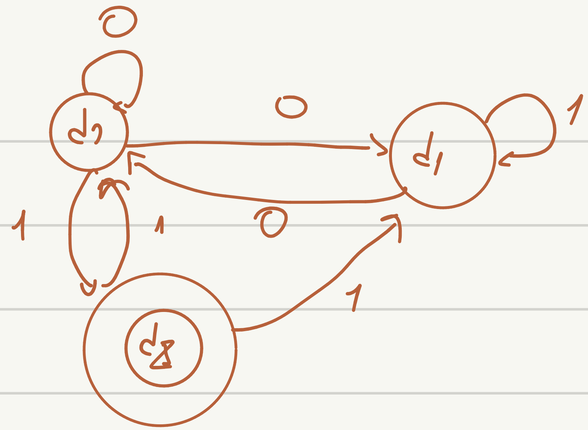
	0	1
d_0	d_1	$d_0 d_1$
d_1	$\emptyset$	$d_1 d_2$
d_2	d_2	d_2

	0	1
d_0	d_1	$d_0 d_1$
$d_0 d_1$	d_1	$d_0 d_1 d_2$
d_1	d_3	$d_1 d_2$
$d_1 d_2$	d_2	$d_1 d_2$
$d_0 d_1 d_2$	$d_1 d_2$	$d_0 d_1 d_2$
d_2	d_2	d_2
d_3	d_3	d_3





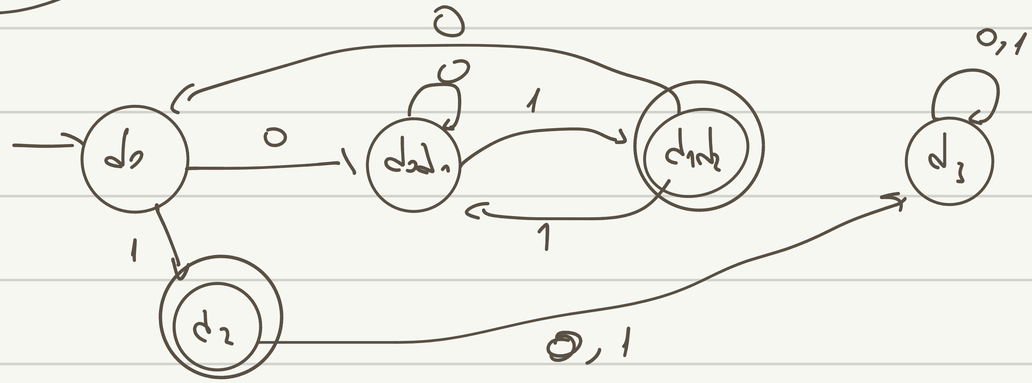
	0	1
q_0	q_0q_1	q_2
q_1	q_2	q_1
q_2	$\emptyset$	q_0q_1



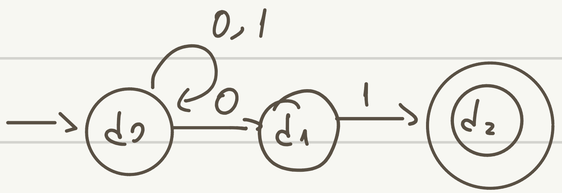
	0	1
q_0	q_0q_1	q_2
q_0q_1	q_0q_1	q_1q_2
q_1q_2	q_0	q_0q_1
q_2	q_3	q_3
q_3	q_3	q_3

$\{q_1, 1\} \cup \{q_2, 1\}$

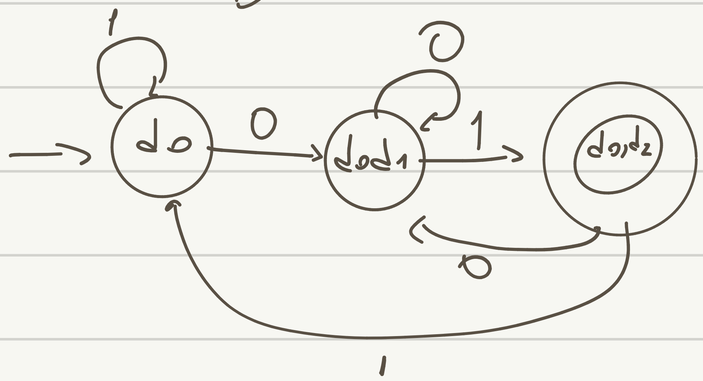
$\{q_0, 0\} \cup \{q_1, 0\}$



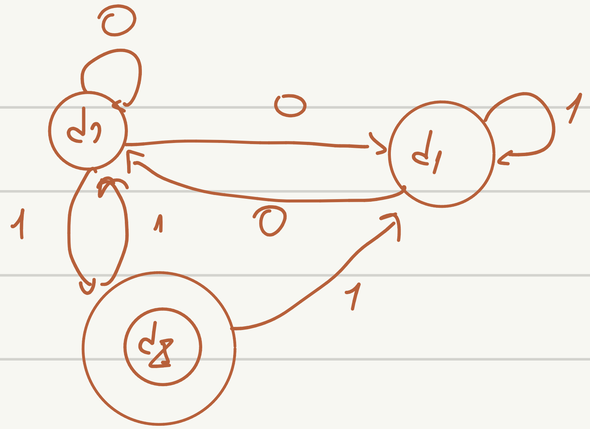
NFA ends with '01'



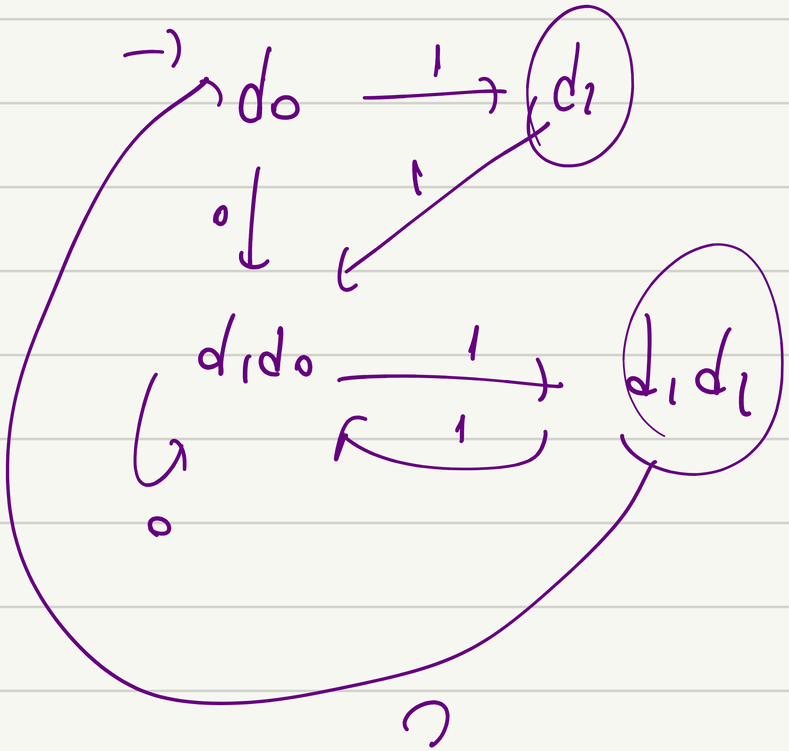
DFA



	0	1
q_0	q_0q_1	q_0
q_0q_1	q_0q_1	q_0q_2
q_0q_2	q_1	q_0



	0	1
$\rightarrow q_0$	$q_1 q_0$	q_2
q_1	$\emptyset$	$q_0 q_1$
$q_0 q_1$	$q_0 q_1$	$q_1 q_2$
$q_1 q_2$	q_0	$q_0 q_1$
q_2	q_2	q_2



NFA ends with '01'

	e	r
q_0	$q_0 q_1$	q_0
q_1	$\emptyset$	q_2
q_2	$\emptyset$	$\emptyset$

	0	1
q_0	$q_0 q_1$	q_0
$q_0 q_1$	$q_0 q_1$	$q_0 q_2$
$q_0 q_2$	q_0	q_0