



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



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 alphabet, i.e. $\{a, b\}$ $\varepsilon \notin \Sigma$

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

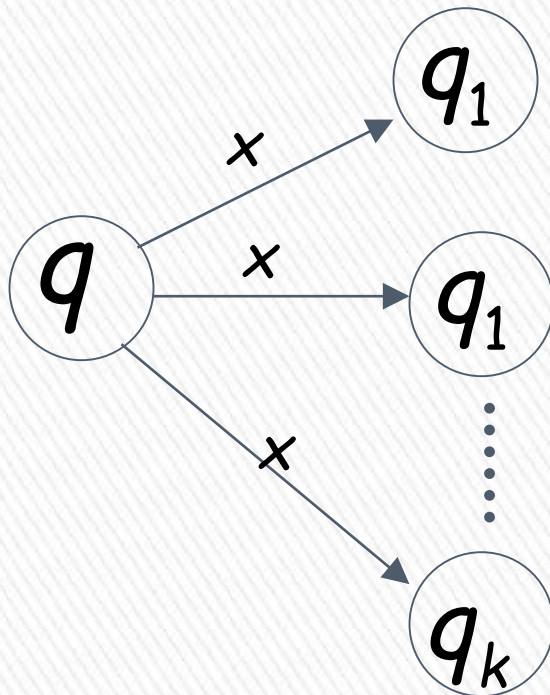
q_0 : Initial state

F : Accepting states



Transition Function δ

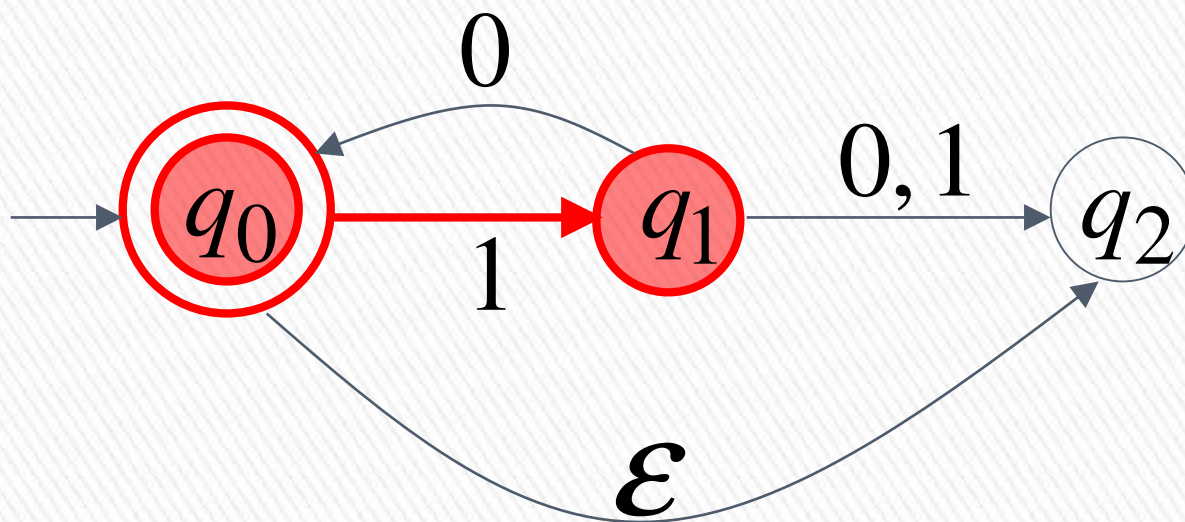
$$\delta(q, x) = \{q_1, q_2, \dots, q_k\}$$



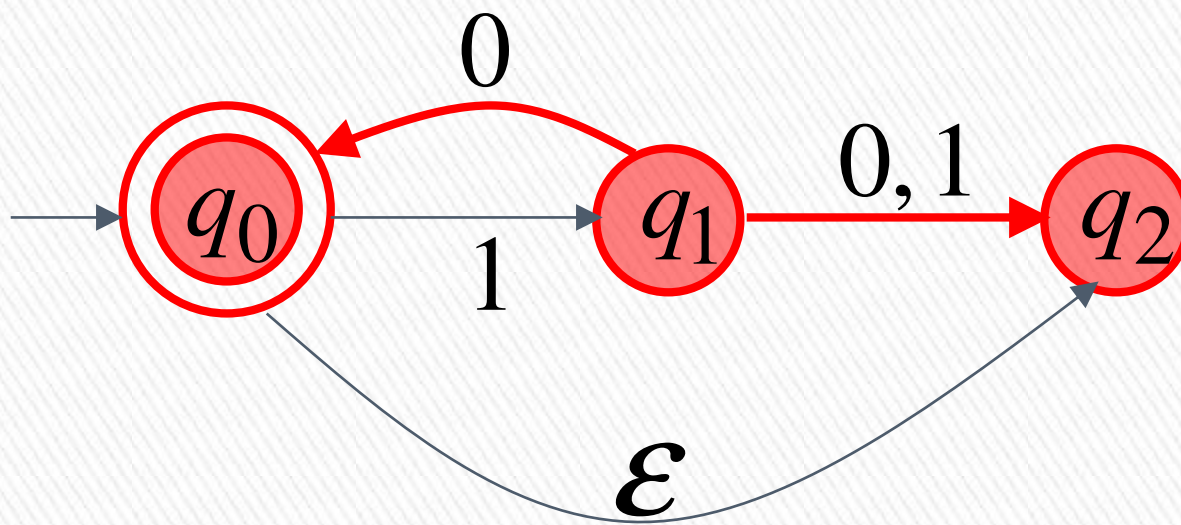
resulting states with
following **one** transition
with symbol x



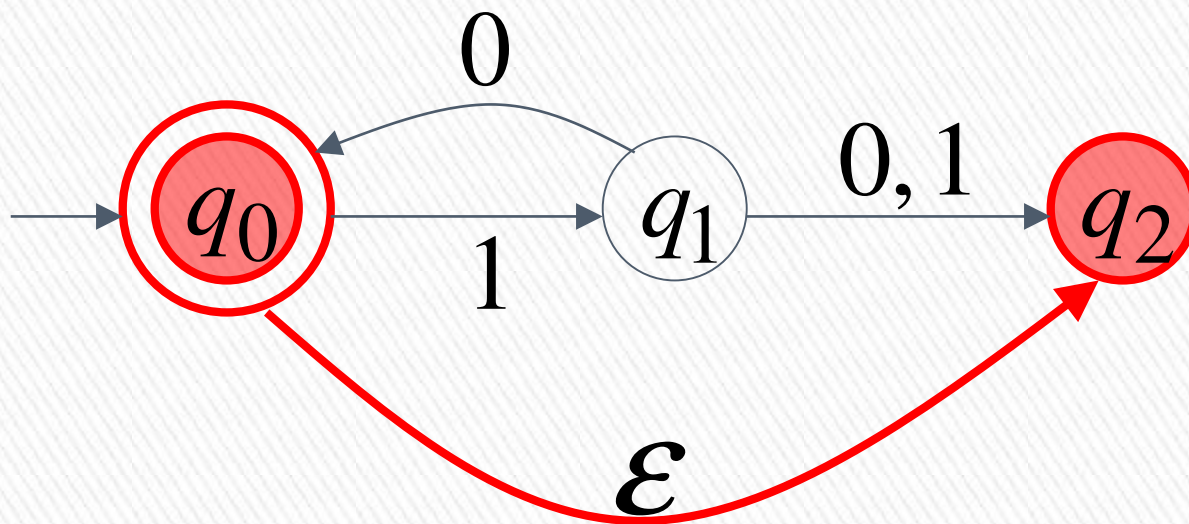
$$\delta(q_0, 1) = \{q_1\}$$



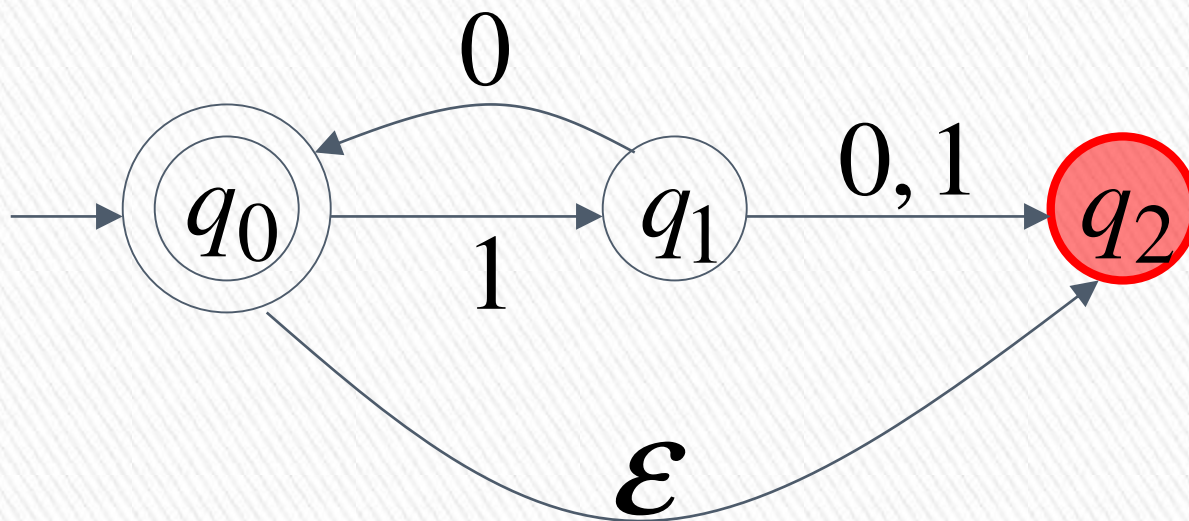
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

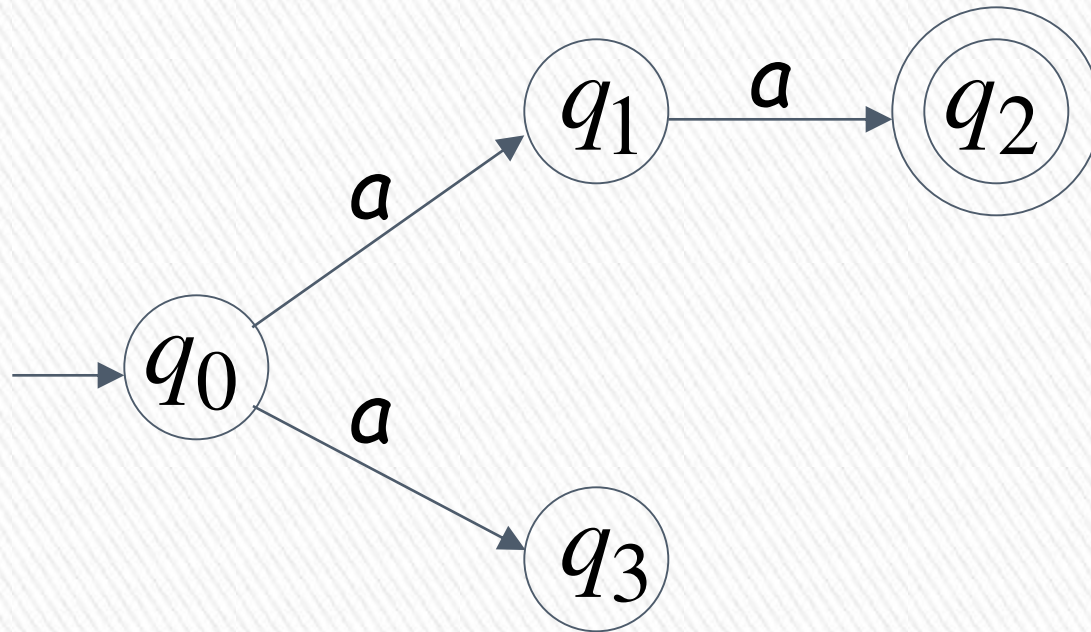


$$\delta(q_2, 1) = \emptyset$$

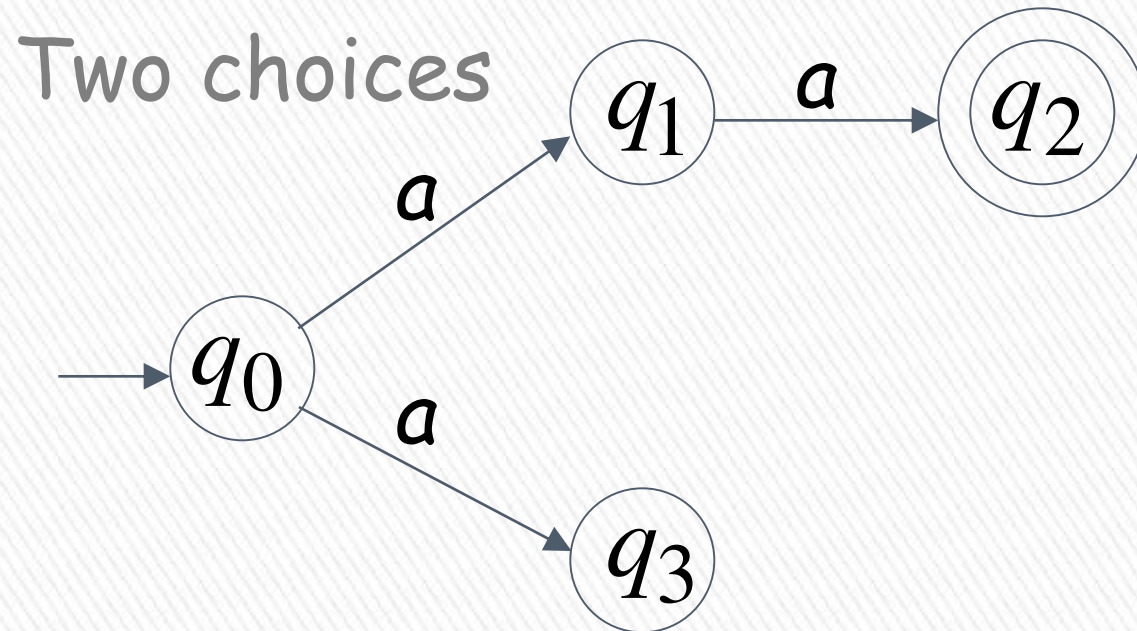


Nondeterministic Finite Automaton (NFA)

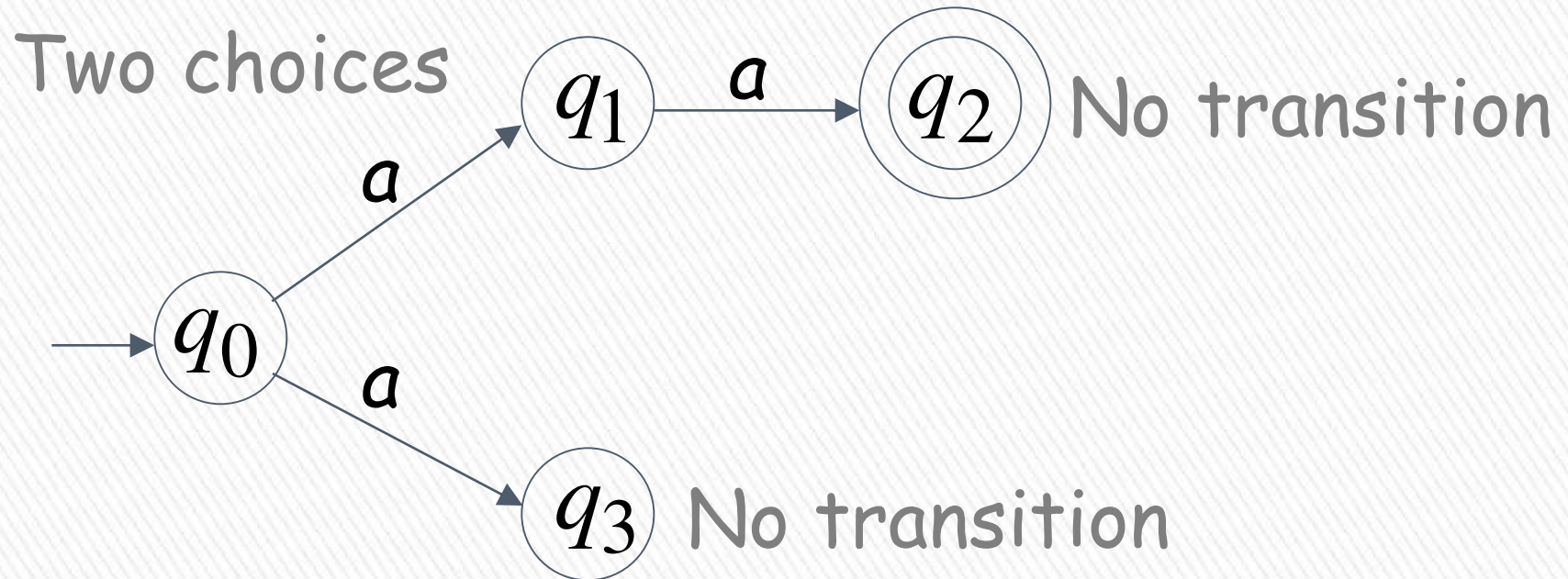
Alphabet = $\{a\}$



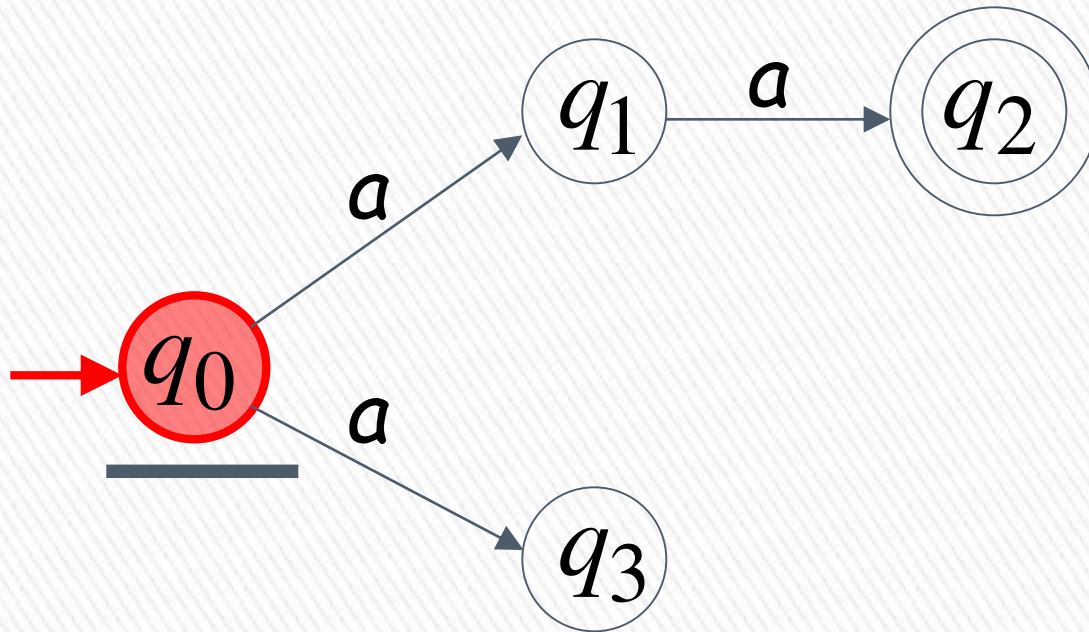
Alphabet = $\{a\}$



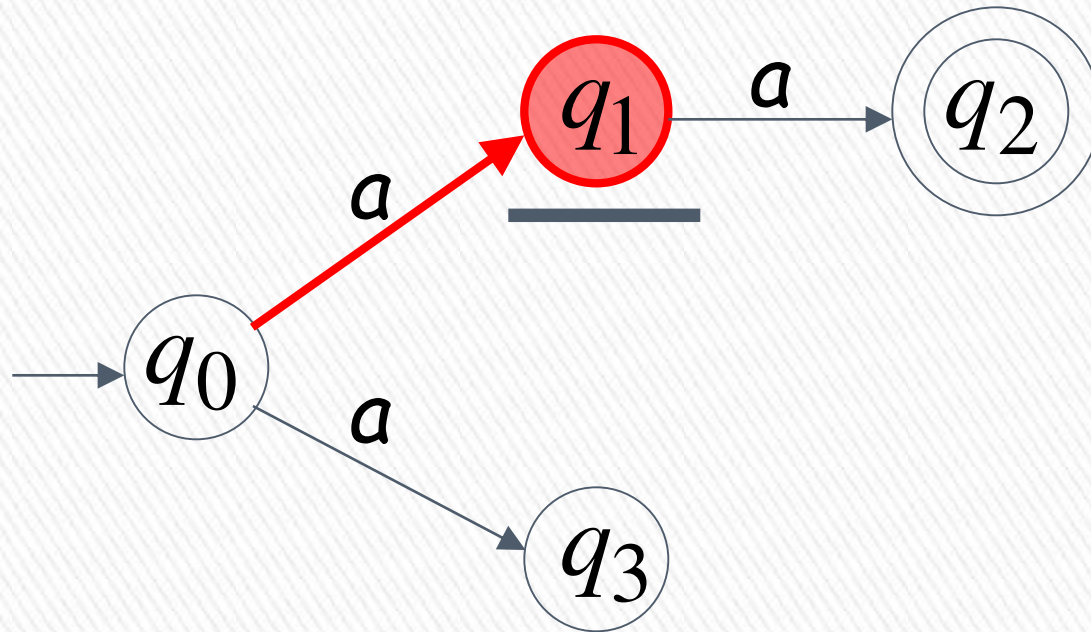
Alphabet = $\{a\}$



First Choice



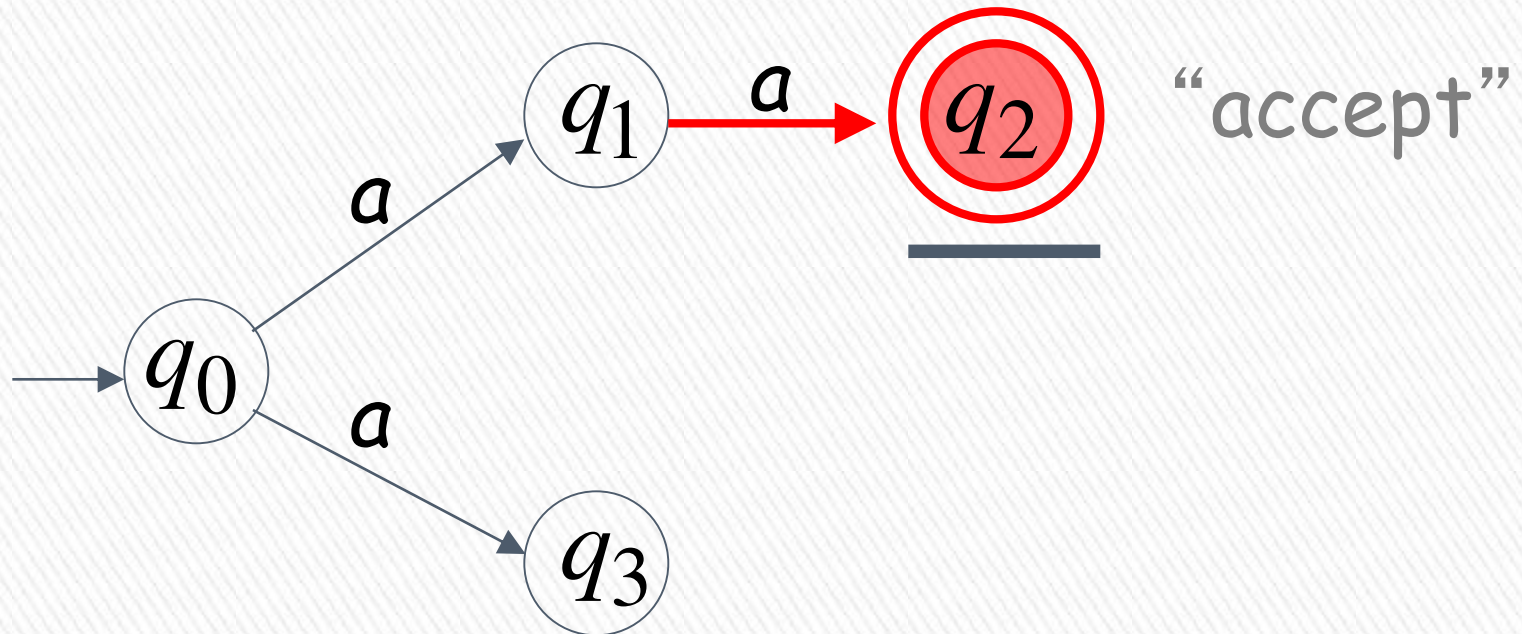
First Choice



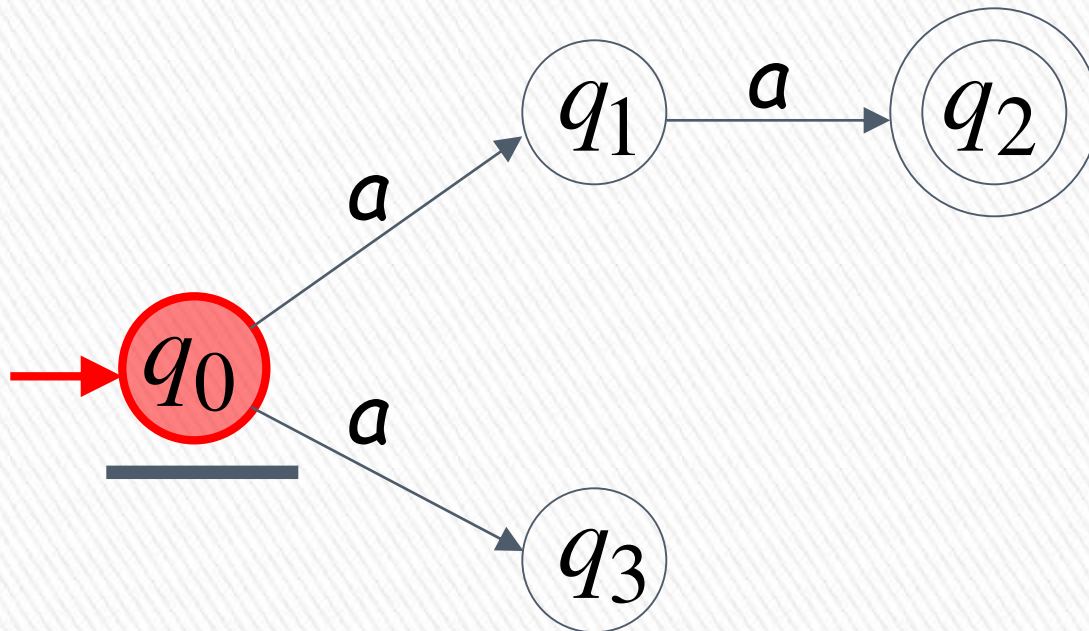
First Choice



All input is consumed



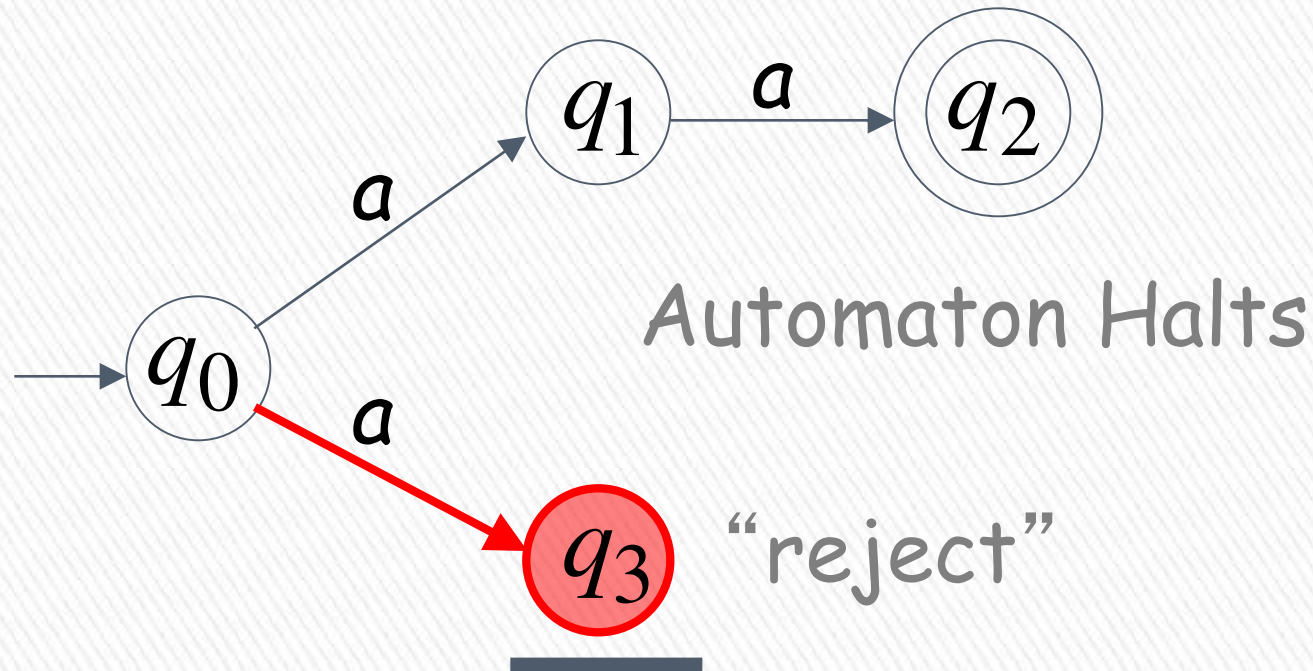
Second Choice



Second Choice



Input cannot be consumed



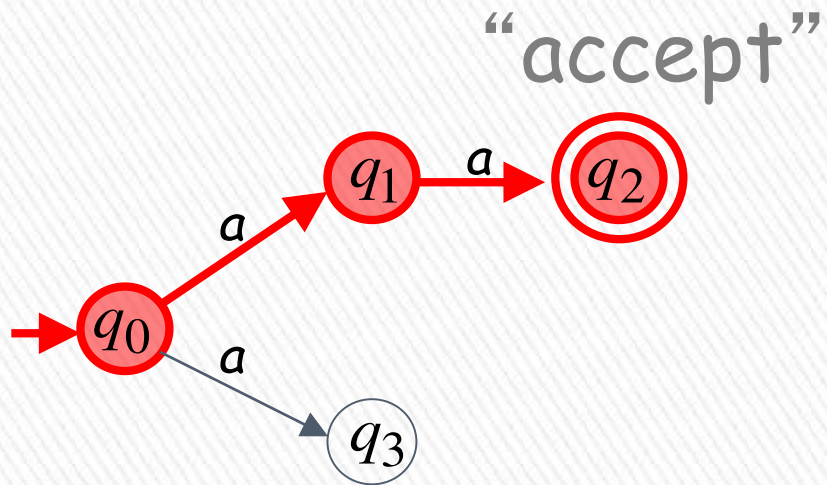
An NFA accepts a string:

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

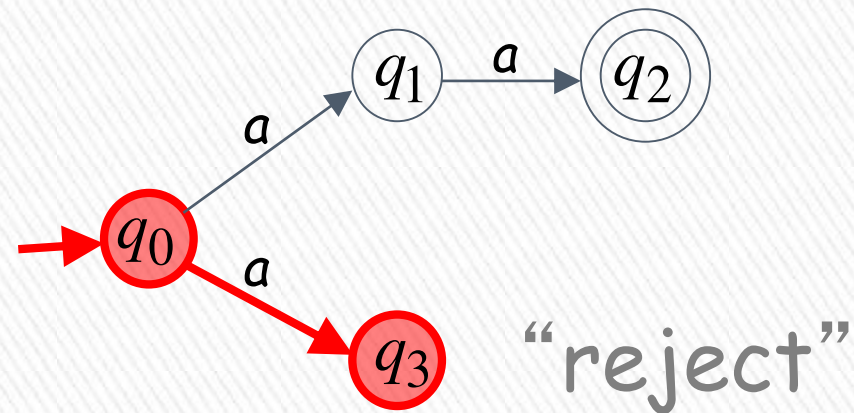
i.e., all the input string is processed and the automaton is in an accepting state



aa is accepted by the NFA:

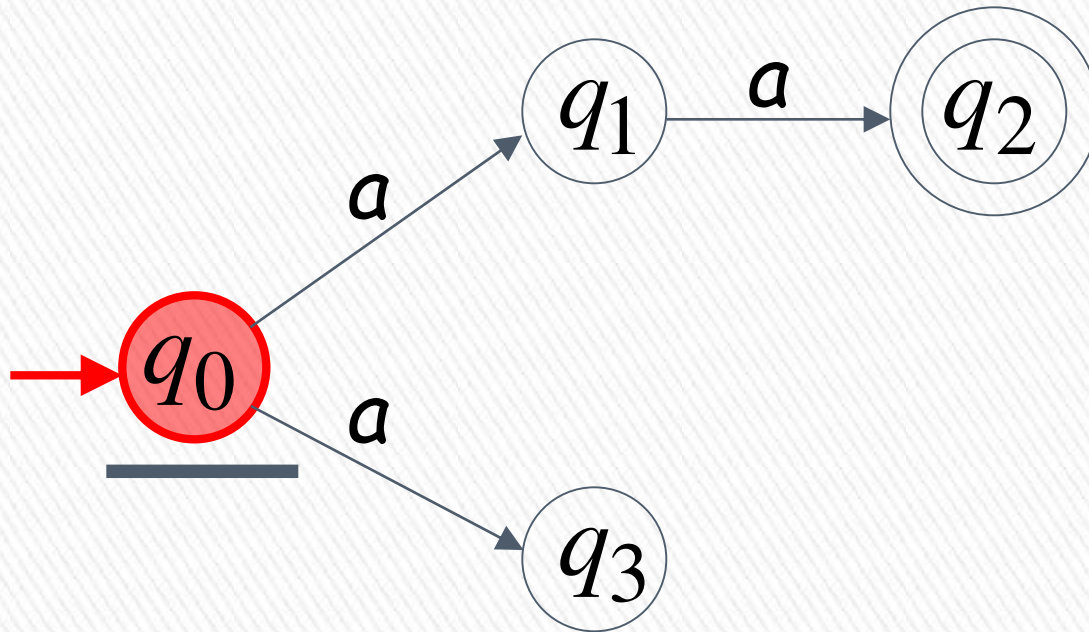
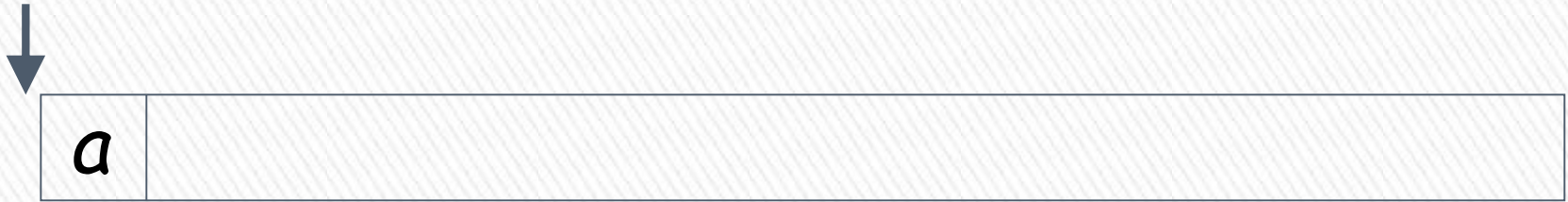


because this
computation
accepts aa



this computation
is ignored

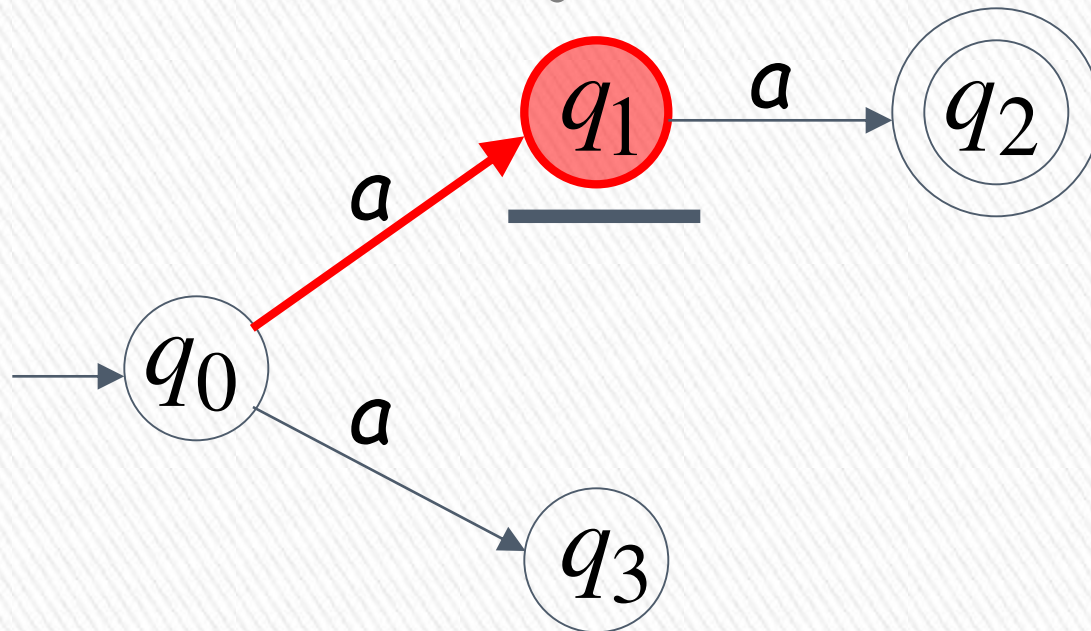
Rejection example



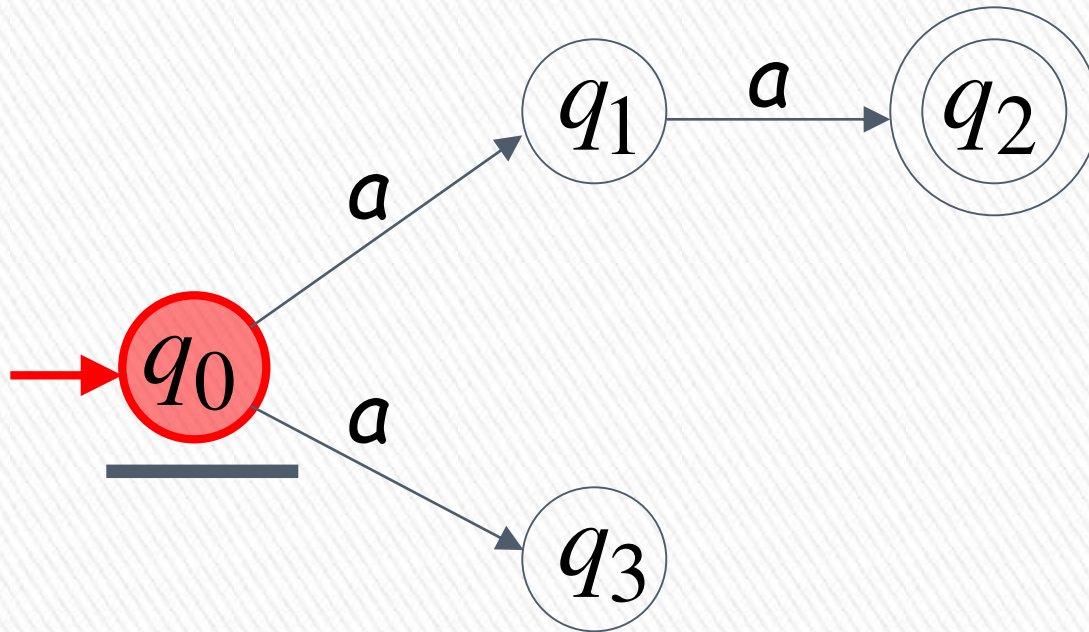
First Choice



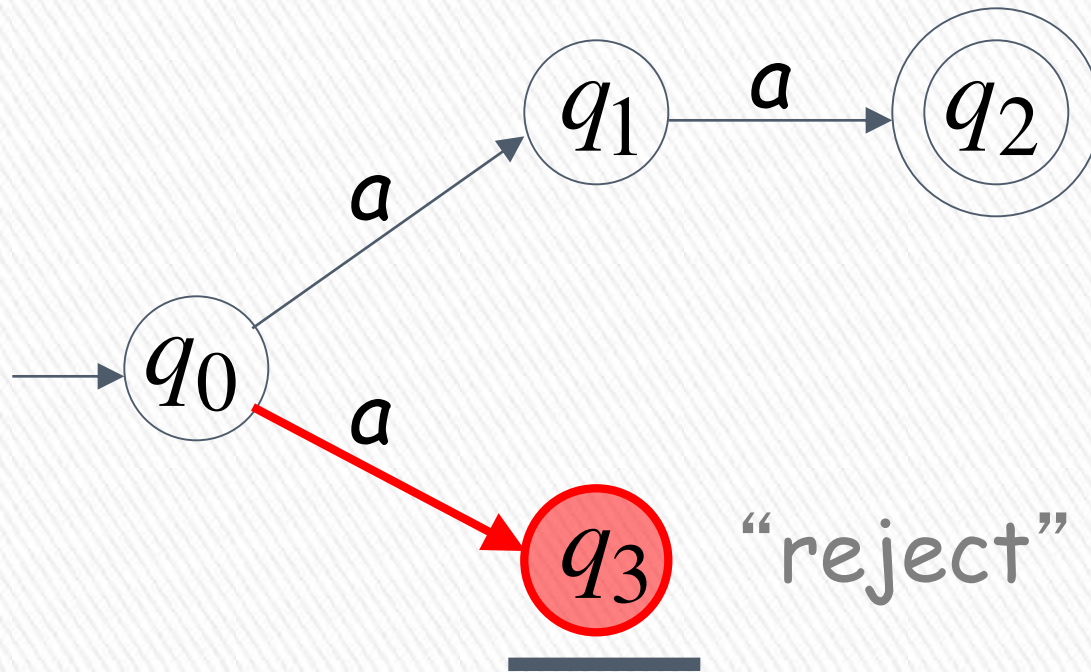
“reject”



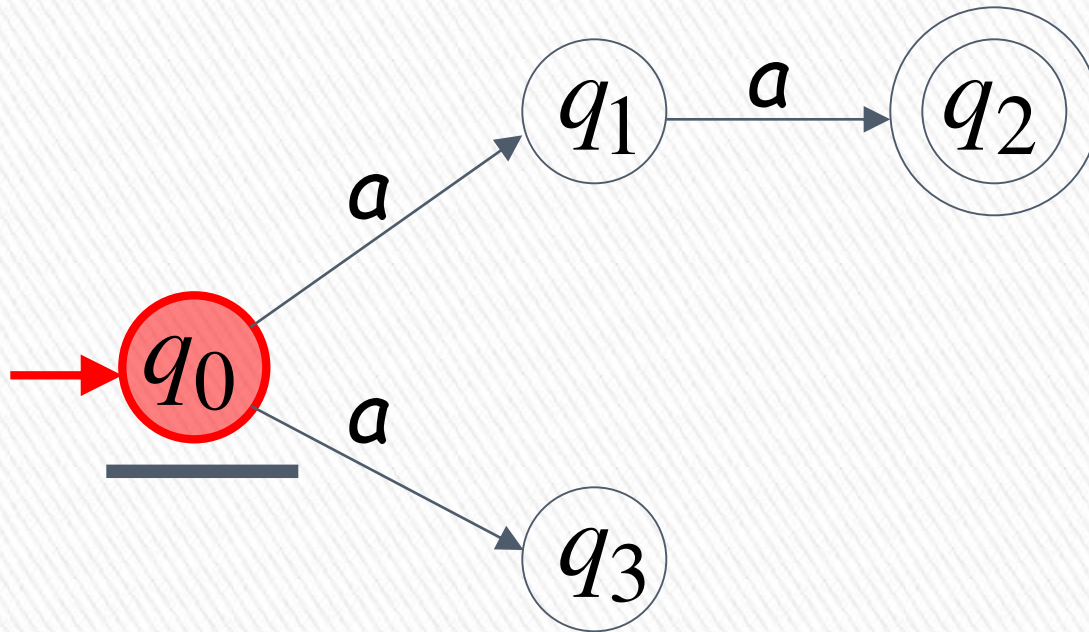
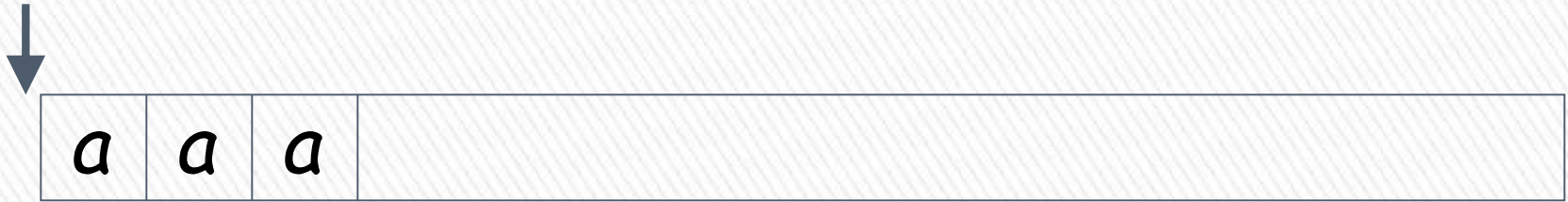
Second Choice



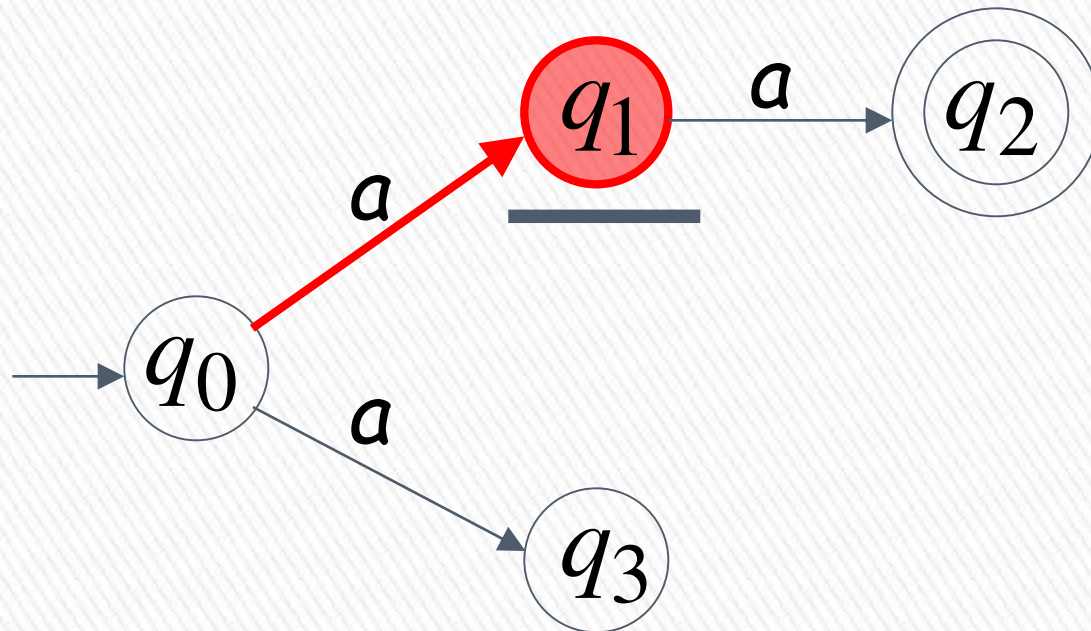
Second Choice



Another Rejection example



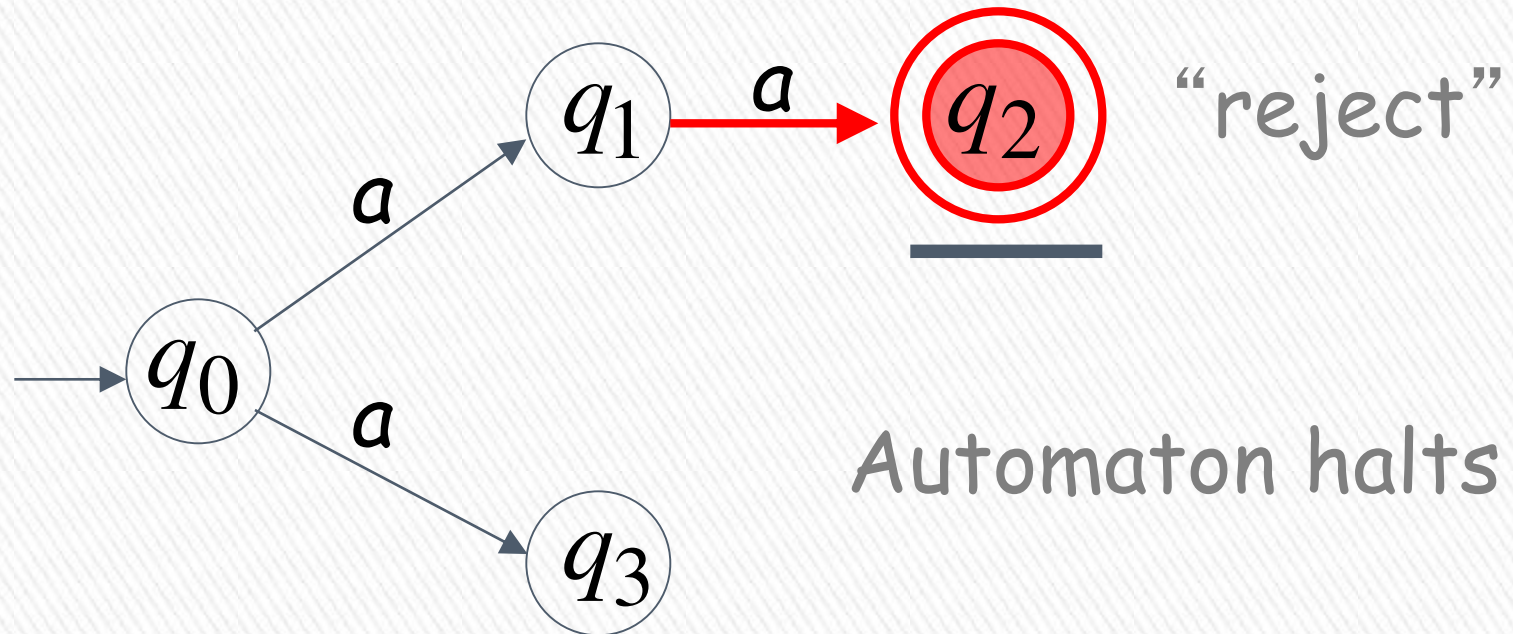
First Choice



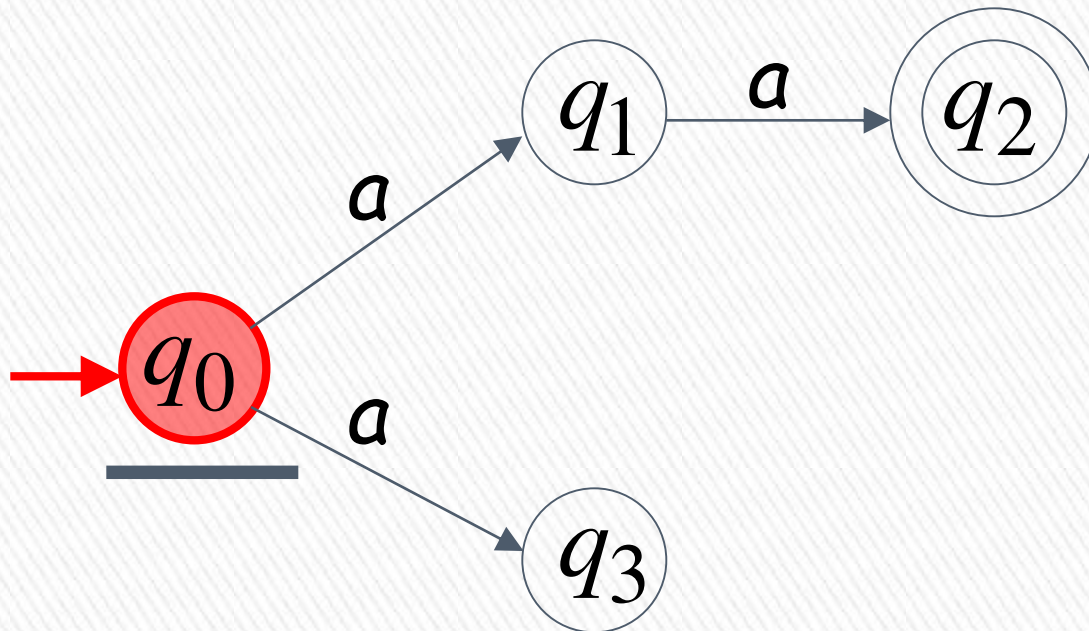
First Choice



Input cannot be consumed



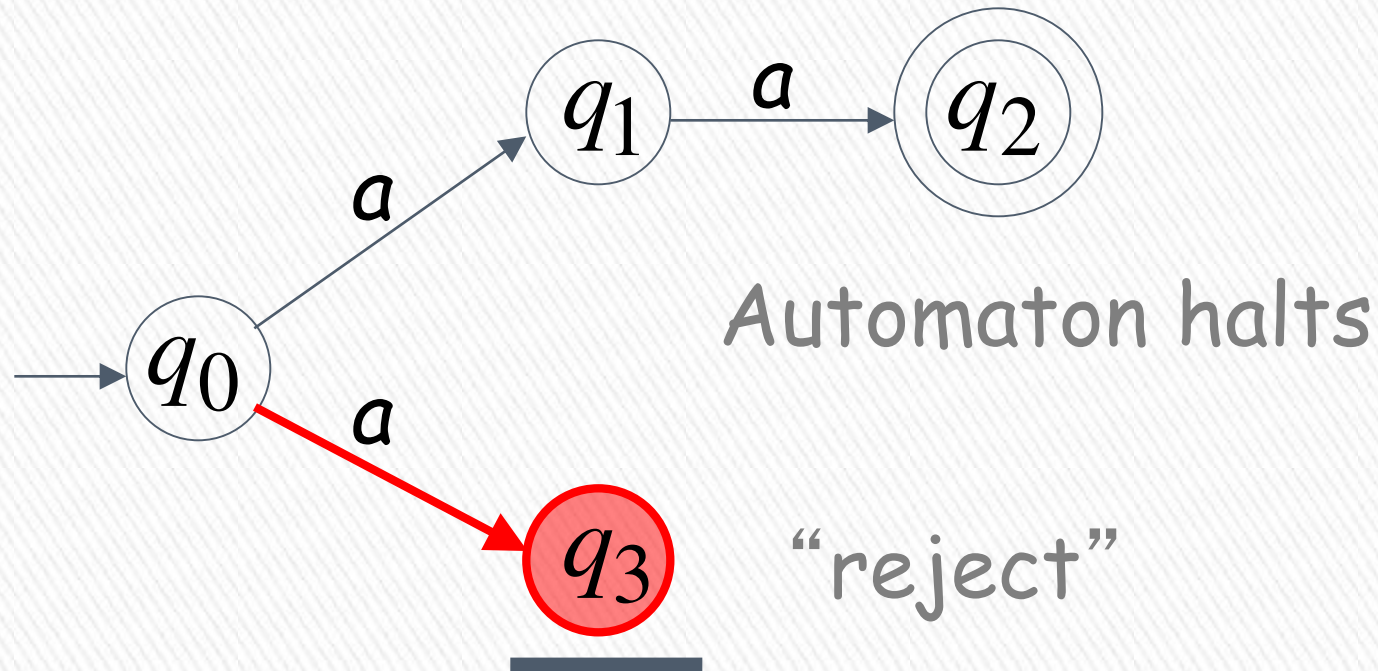
Second Choice



Second Choice



Input cannot be consumed



An NFA rejects a string:

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

For each computation:

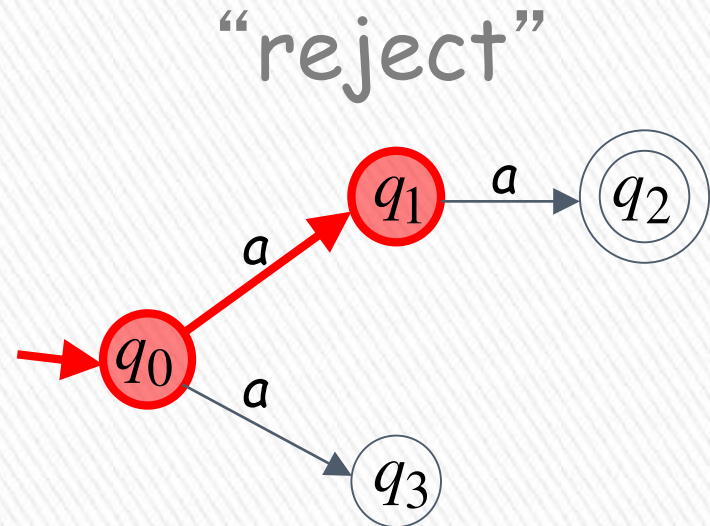
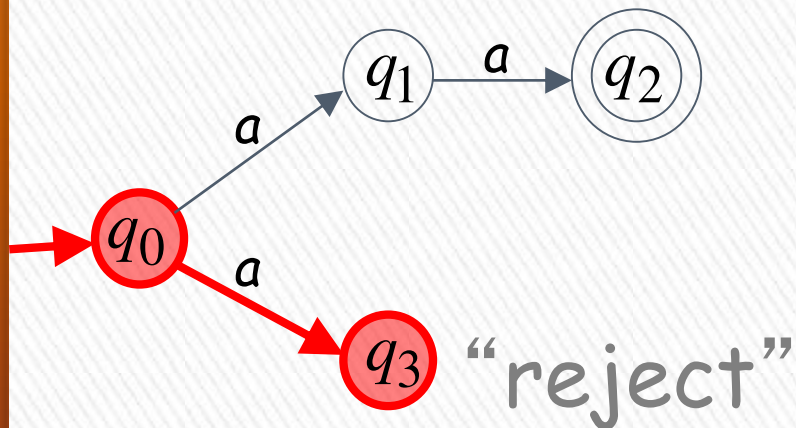
- All the input is consumed and the automaton is in a non final state

OR

- The input cannot be consumed

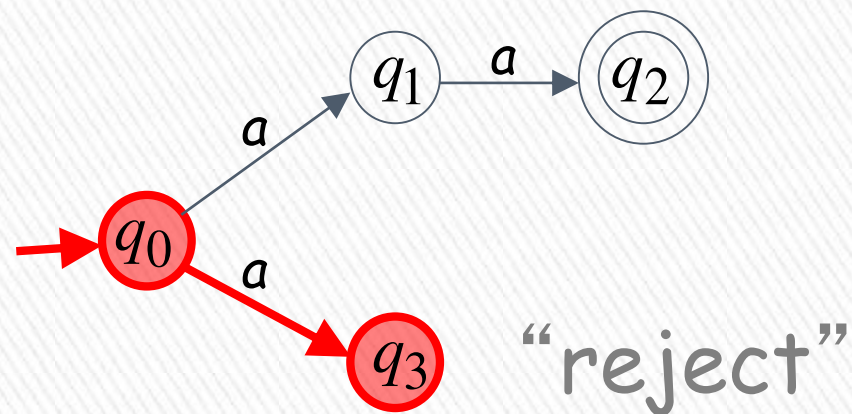
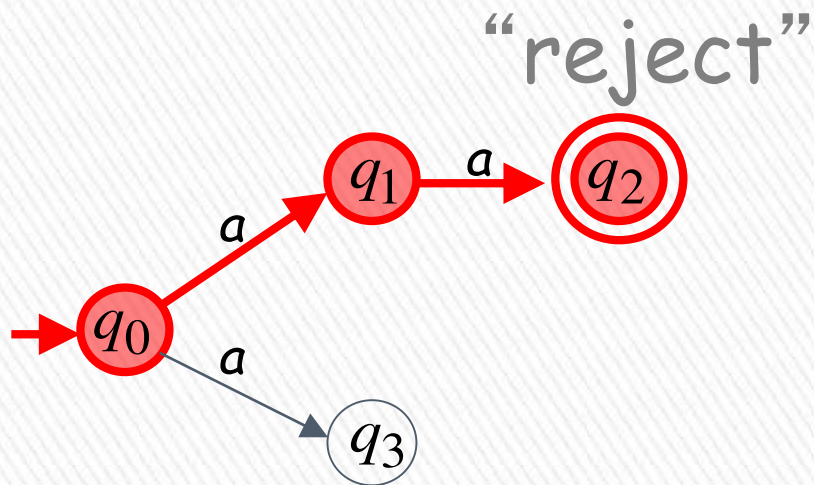


a is rejected by the NFA:



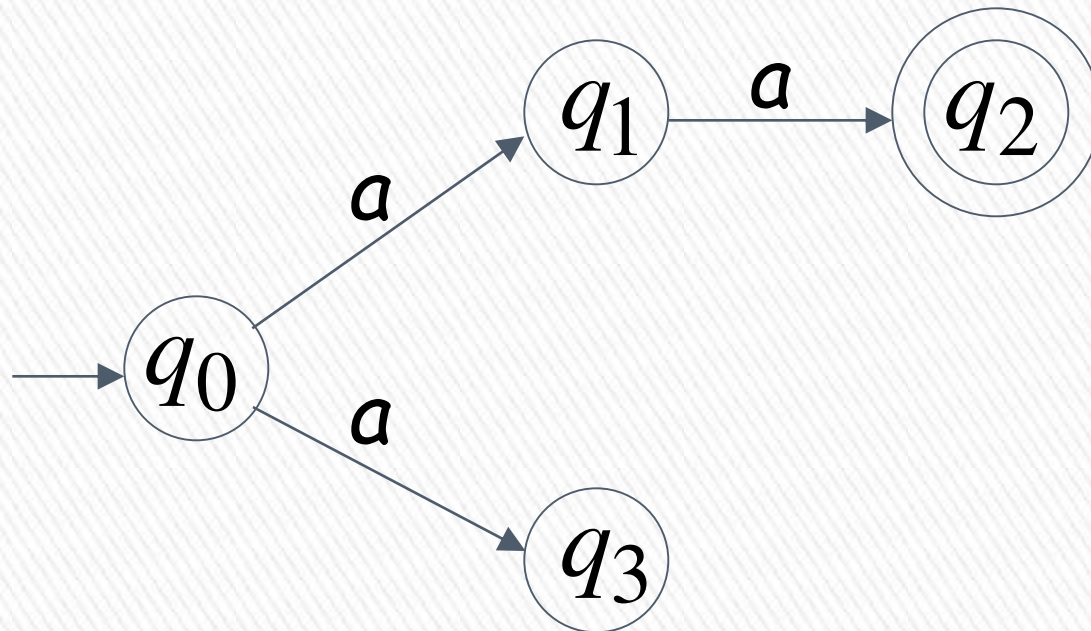
All possible computations lead to rejection >

aaa is rejected by the NFA:



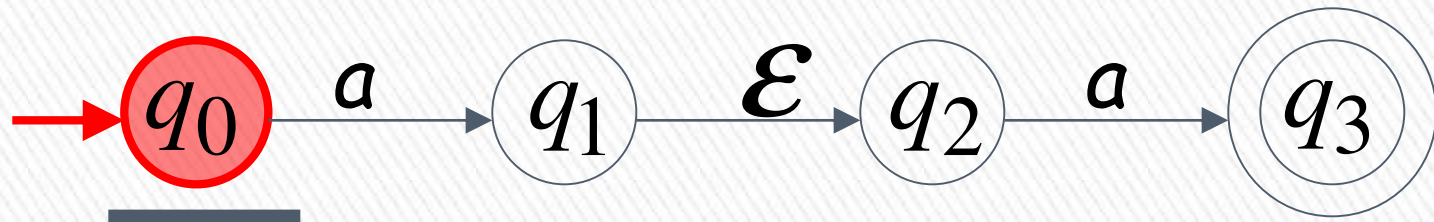
All possible computations lead to rejection

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



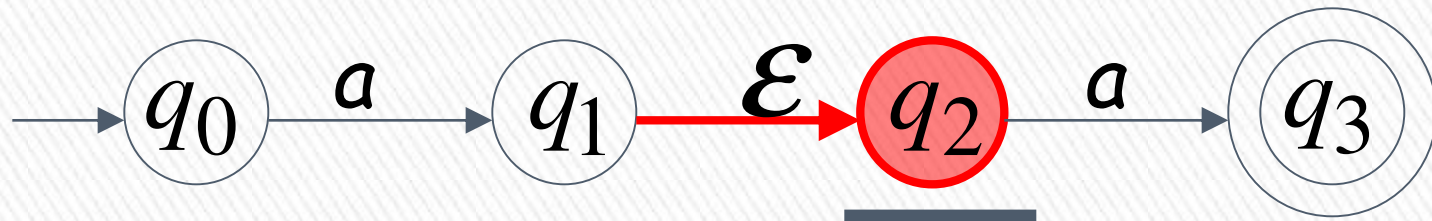
Epsilon Transition



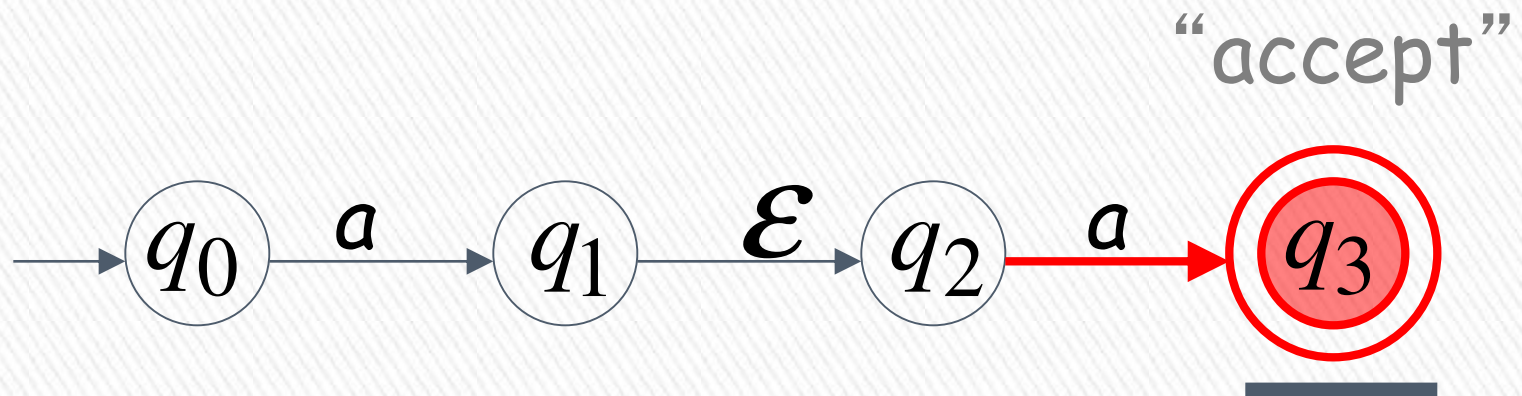




input tape head does not move



all input is consumed



String aa is accepted

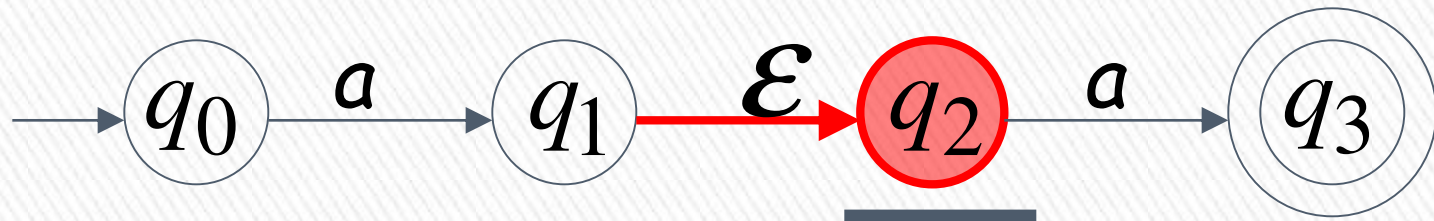


Rejection Example

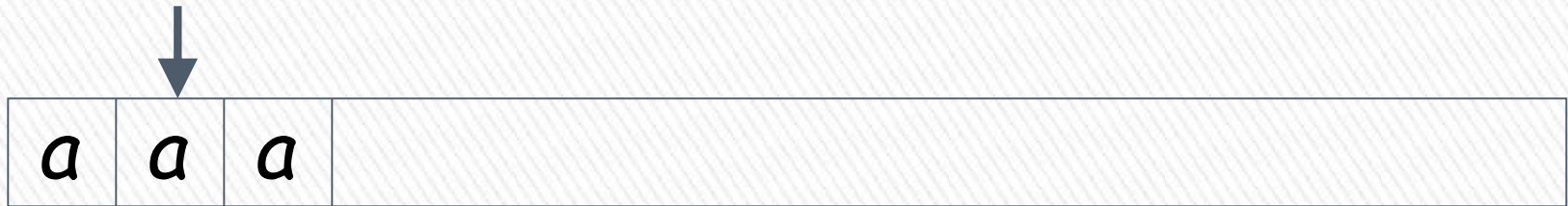




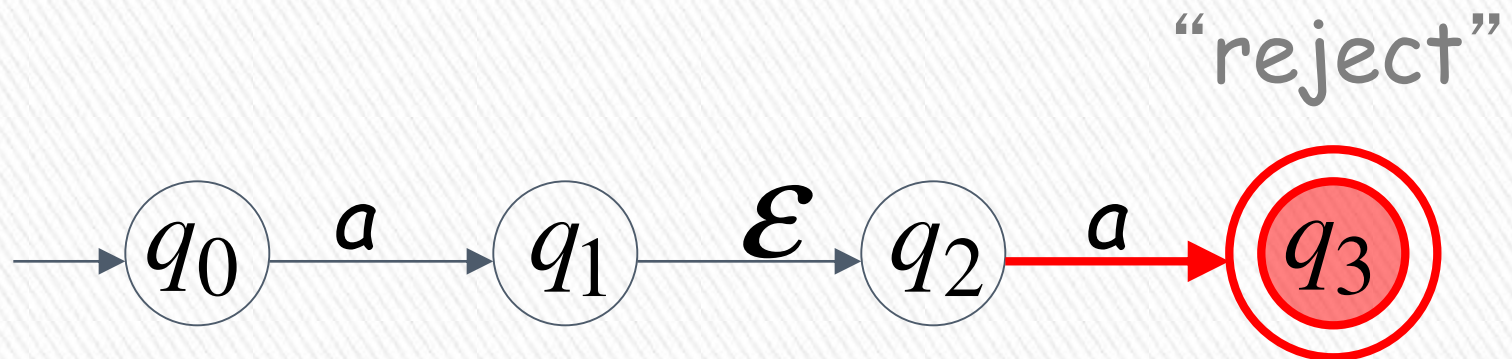
(read head doesn't move)



Input cannot be consumed



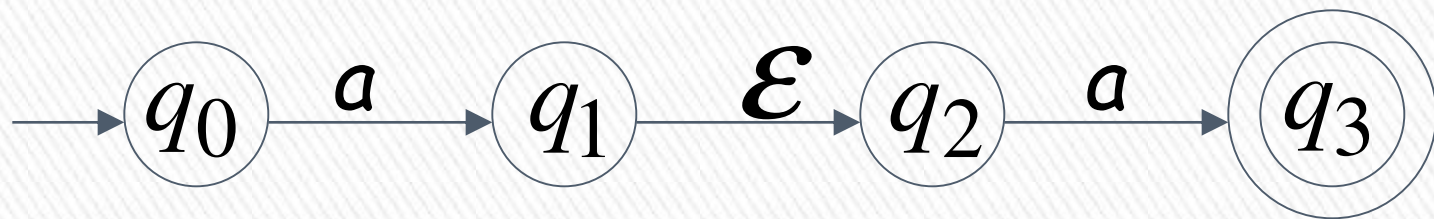
Automaton halts



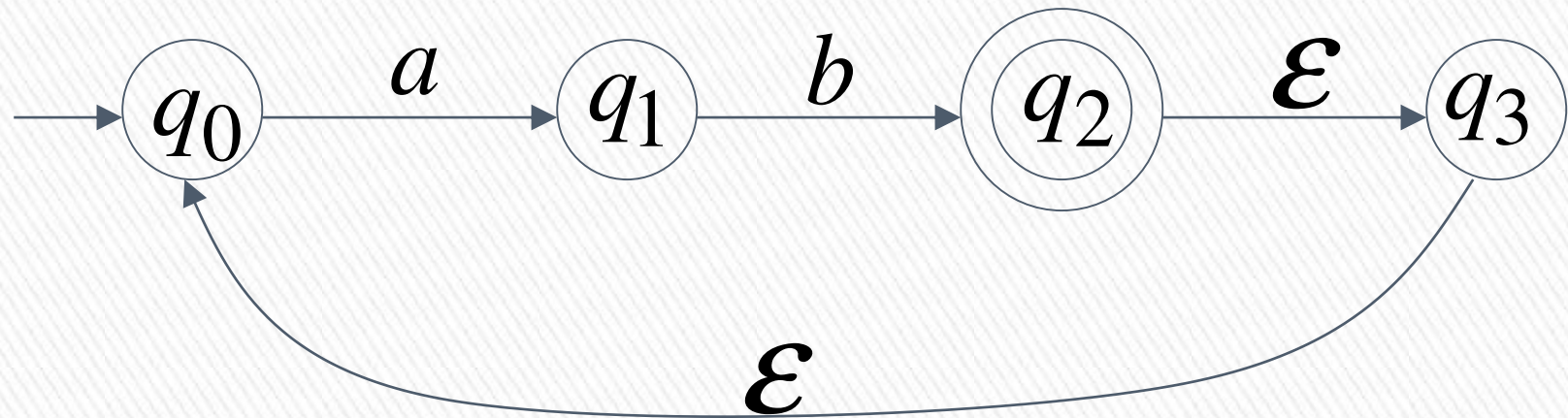
String **aaa** is rejected

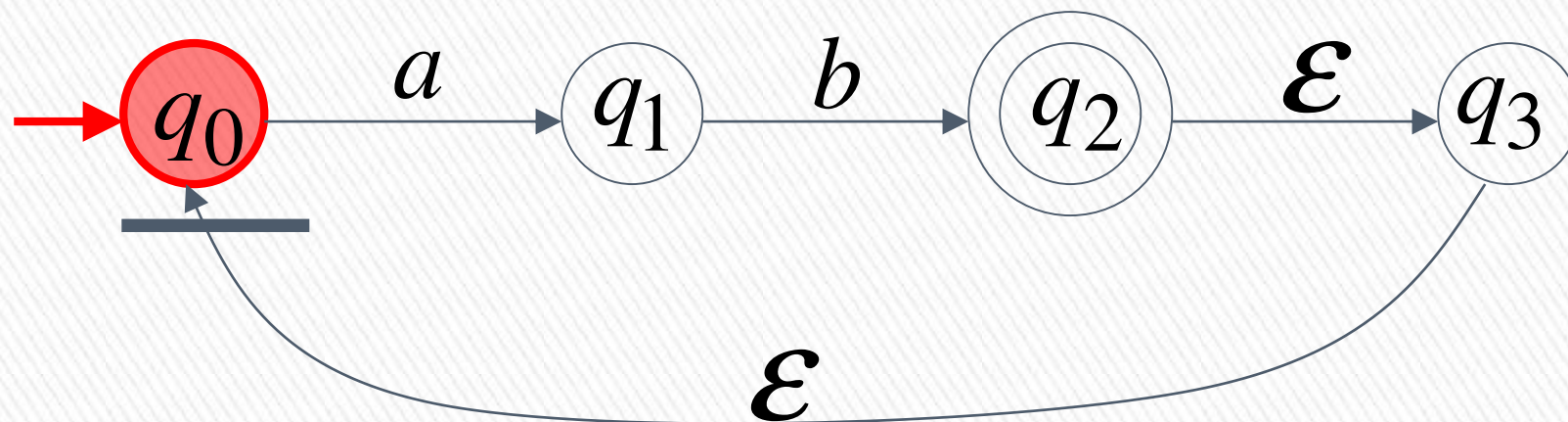


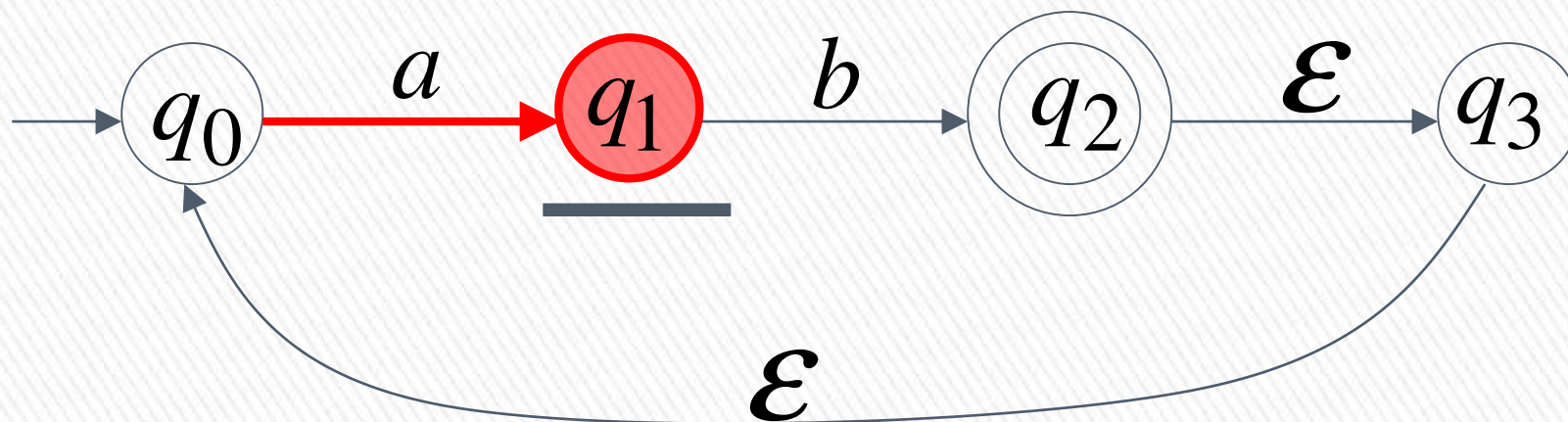
Language accepted: $L = \{aa\}$

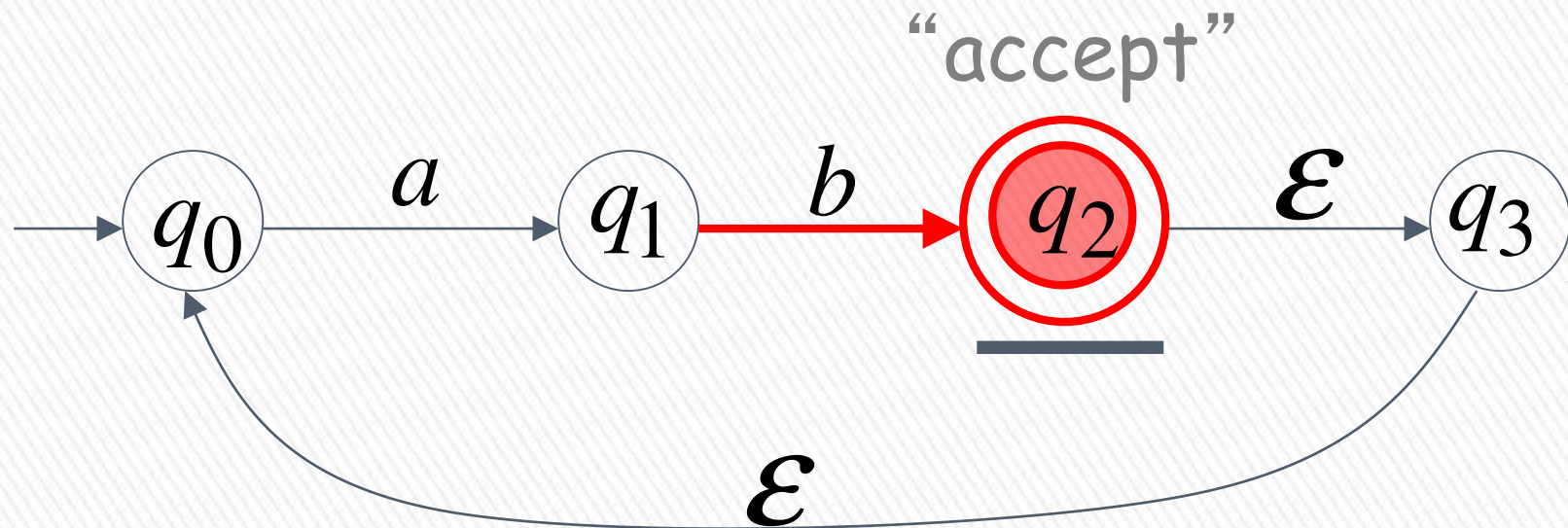


Another NFA Example

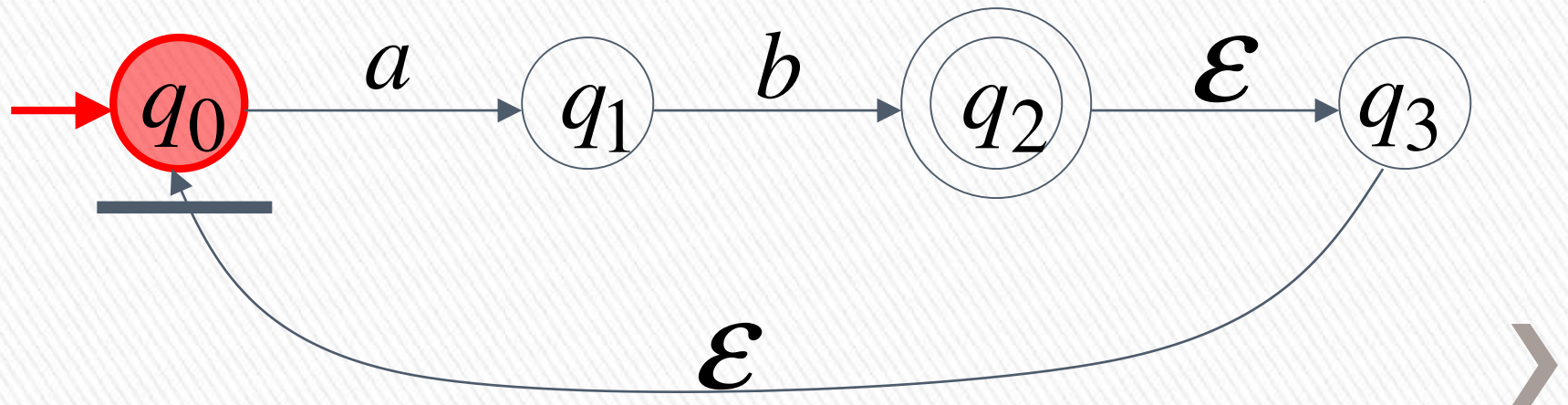


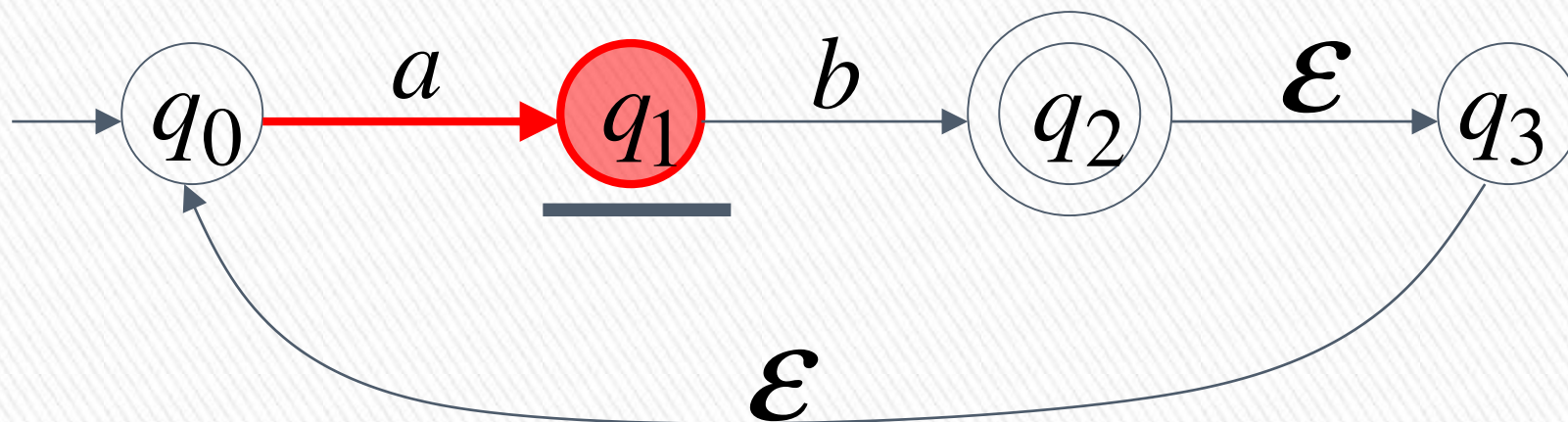


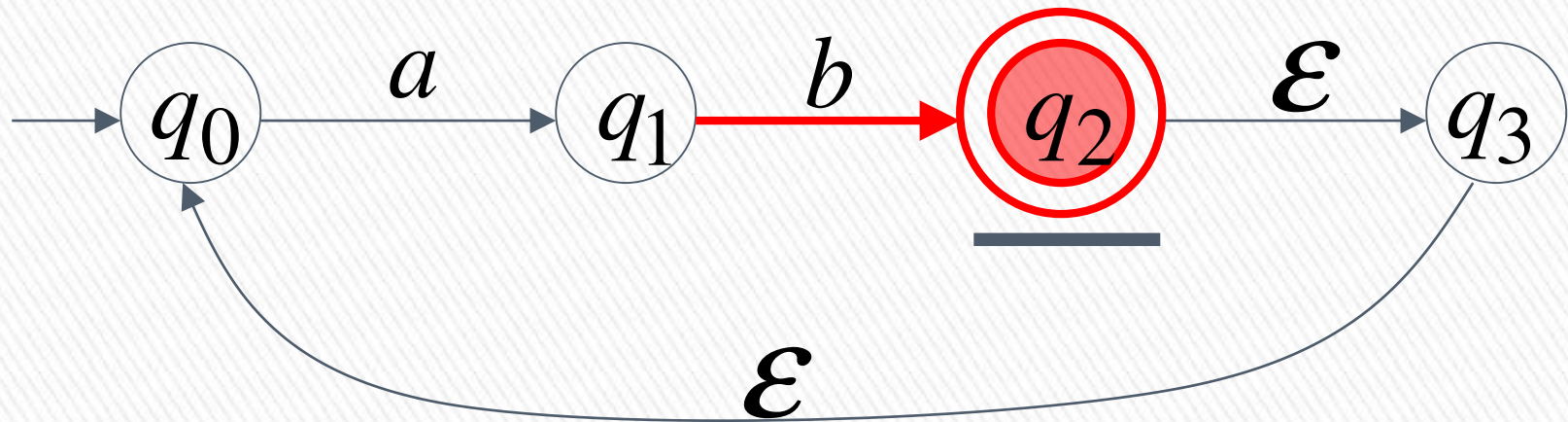


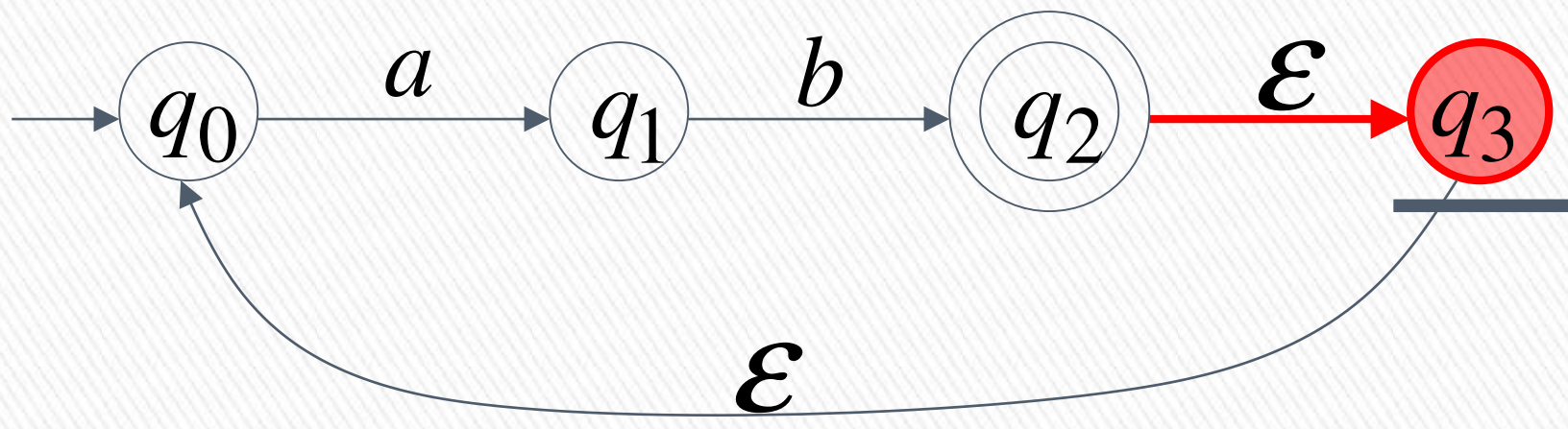


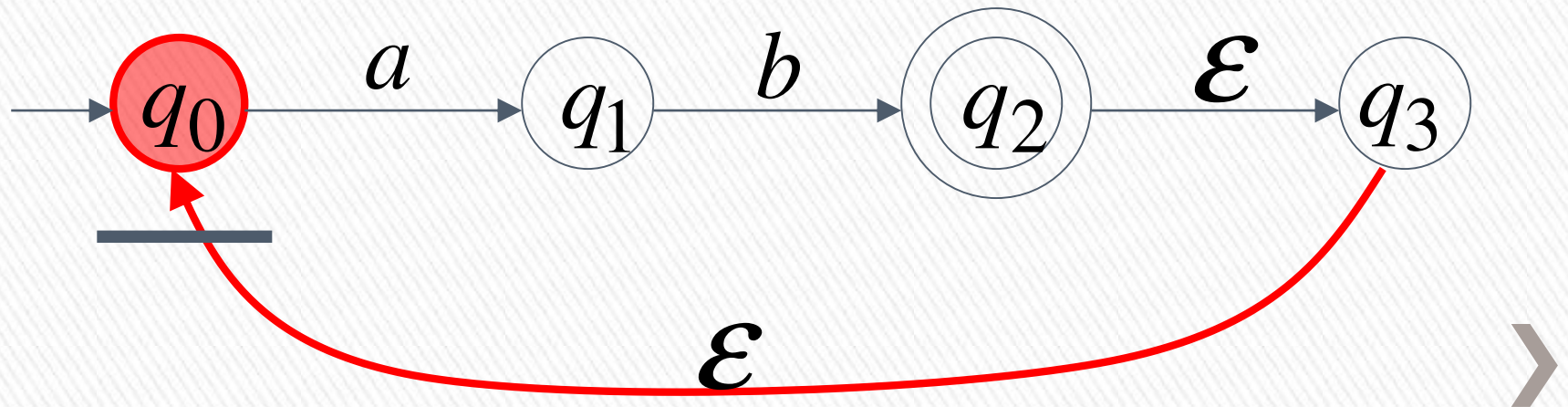
Another String

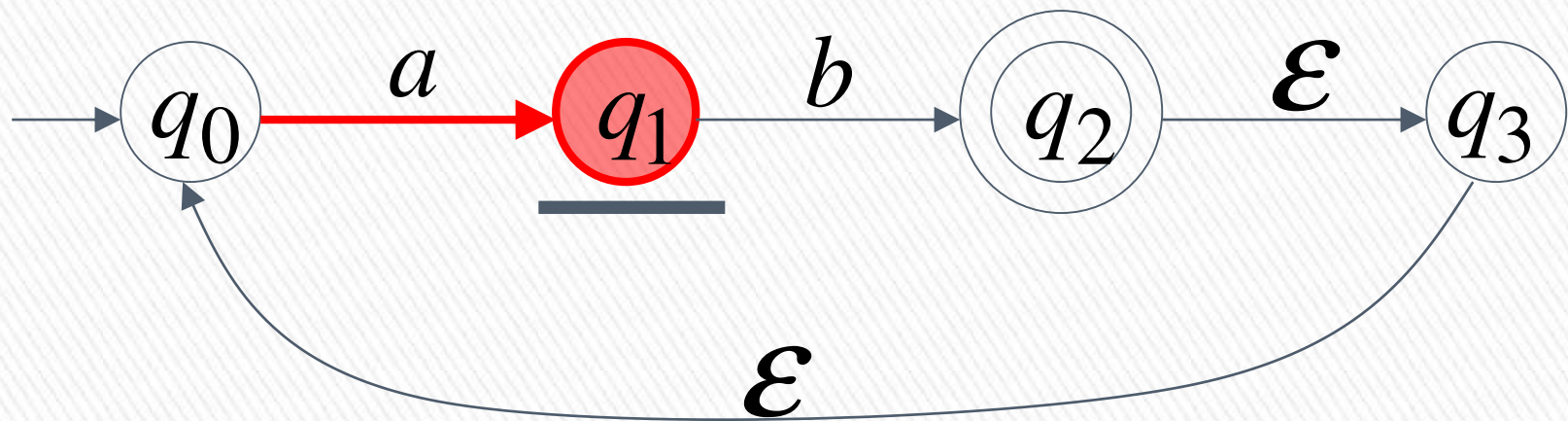


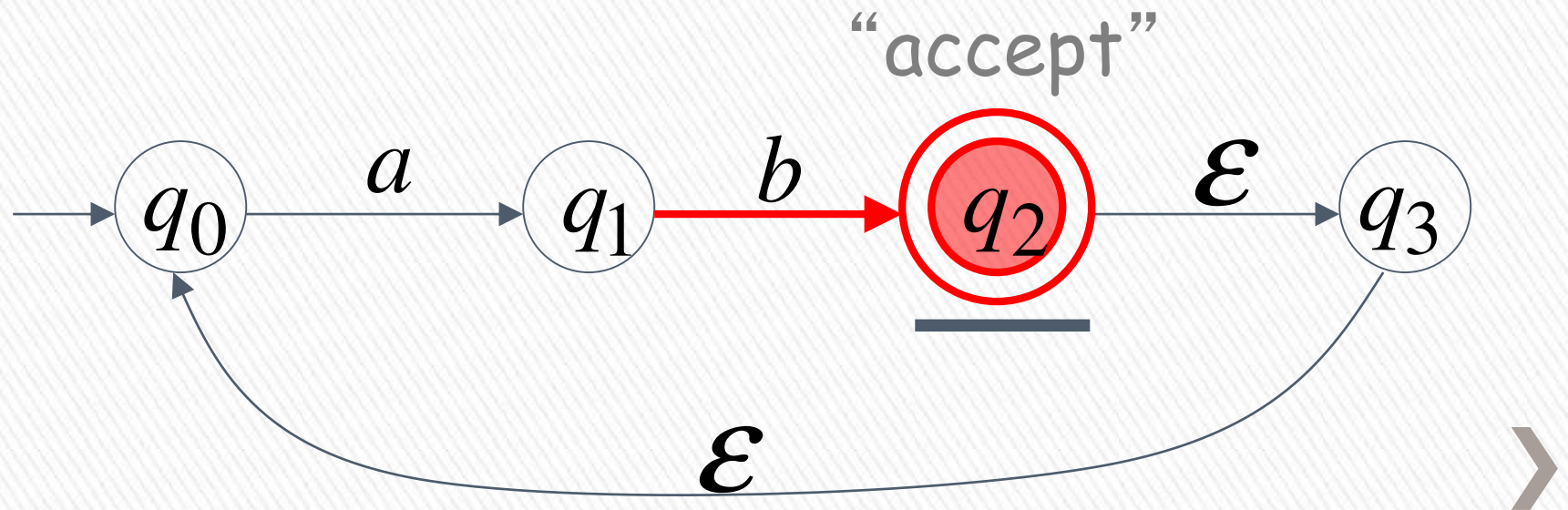






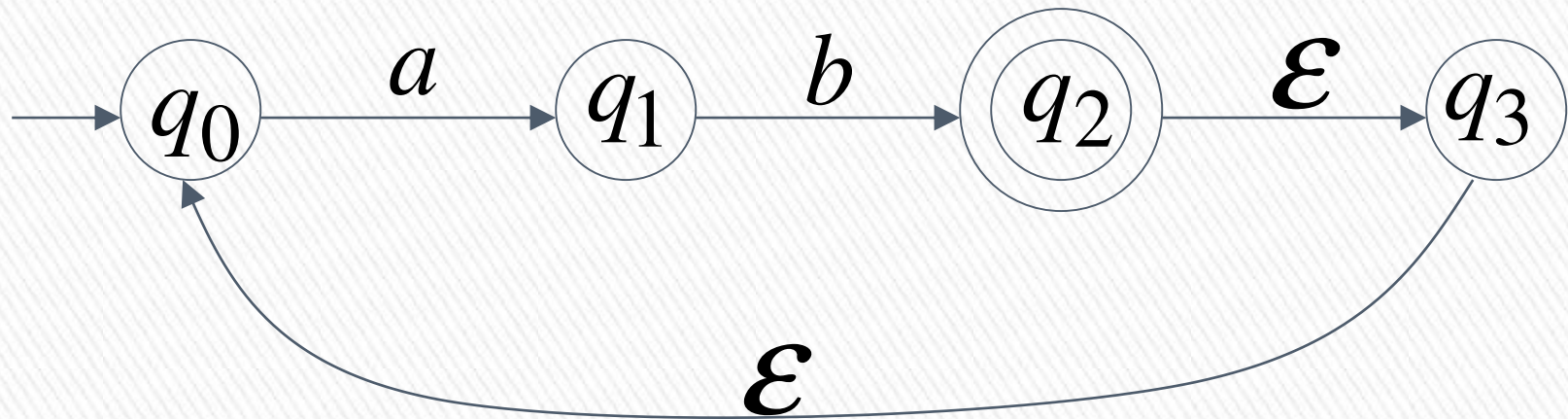




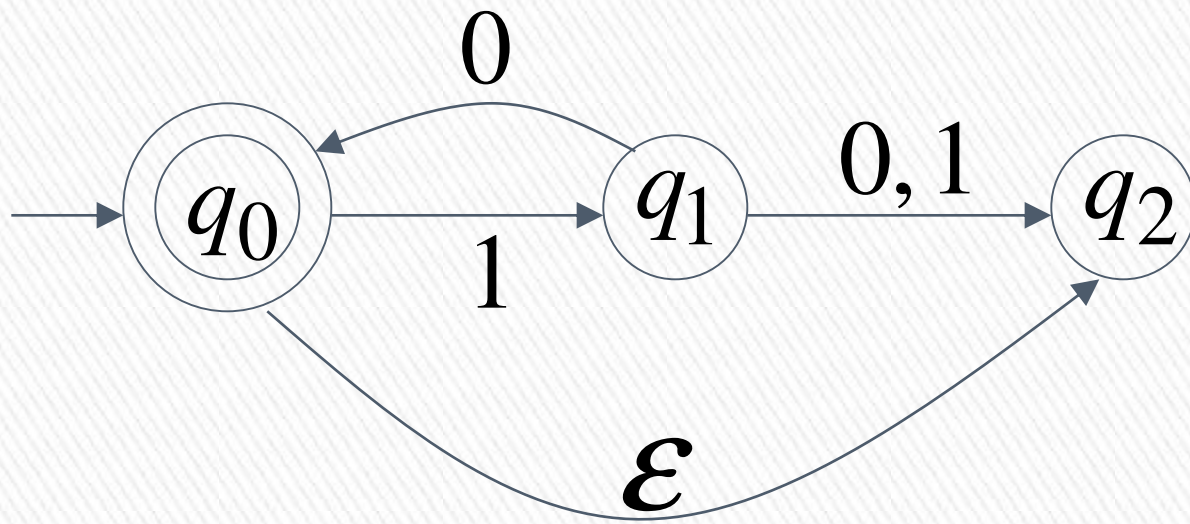


Language accepted

$$L = \{ab, abab, ababab, \dots\}$$
$$= \{ab\}^+$$

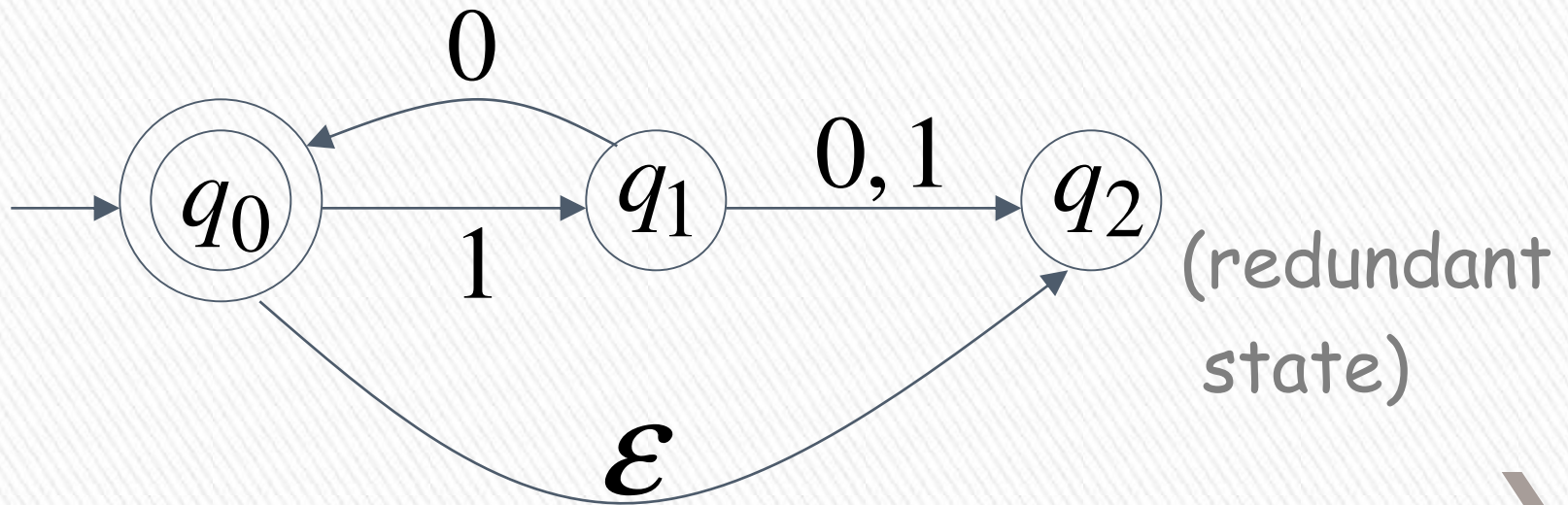


Another NFA Example



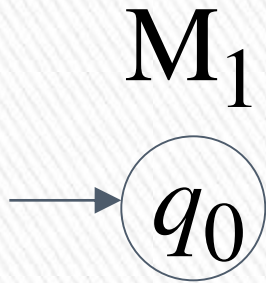
Language accepted

$$\begin{aligned} L(M) &= \{\varepsilon, 10, 1010, 101010, \dots\} \\ &= \{10\}^* \end{aligned}$$

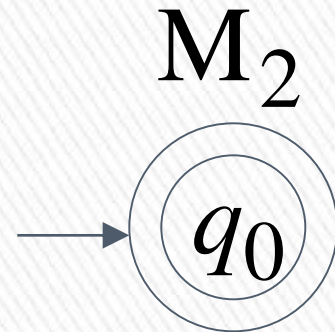


Remarks:

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



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



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

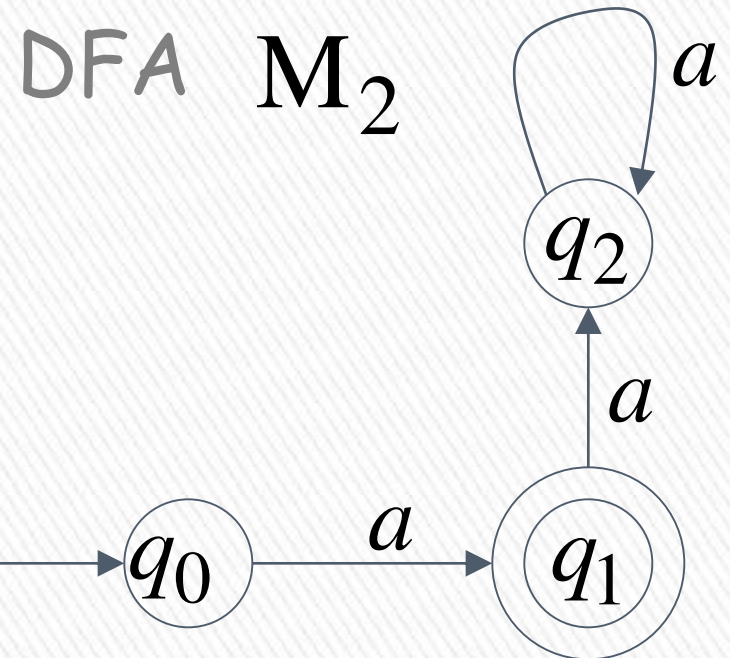


- NFAs are interesting because we can express languages easier than DFAs

NFA M_1



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

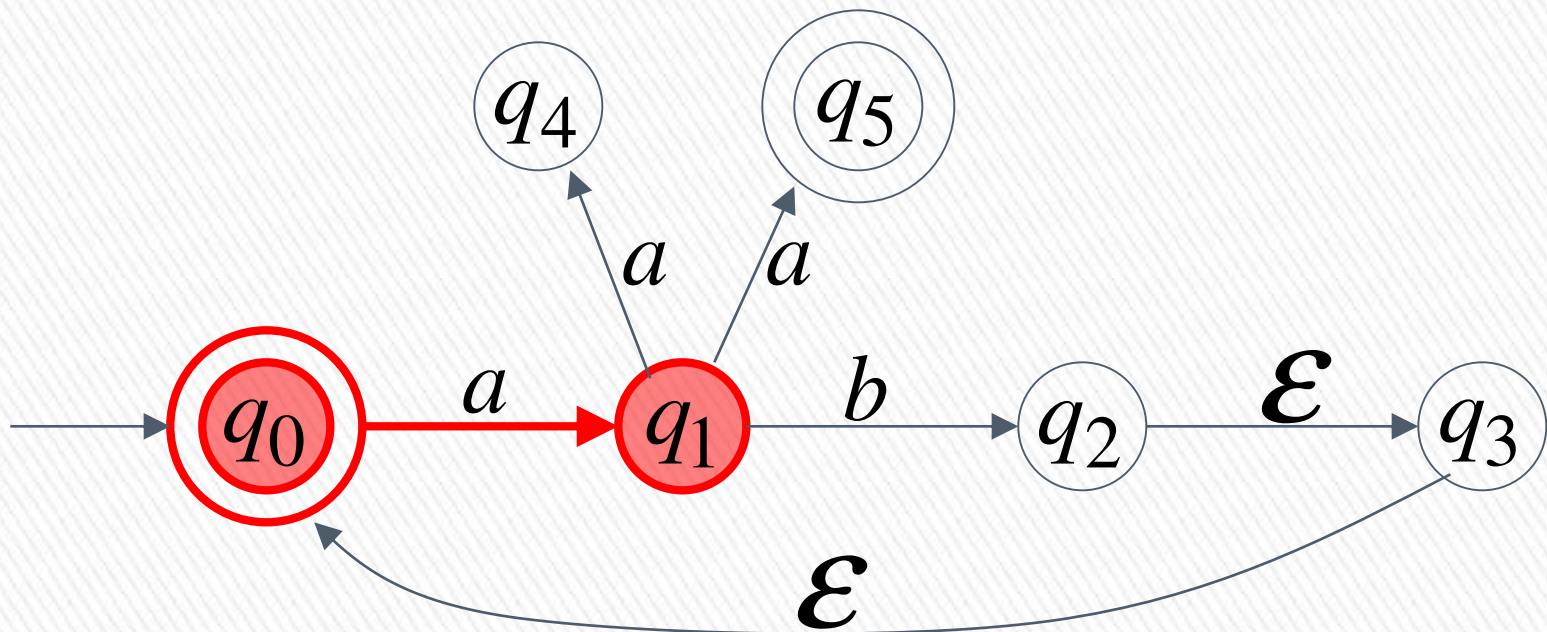


$$L(M_2) = \{a\}$$

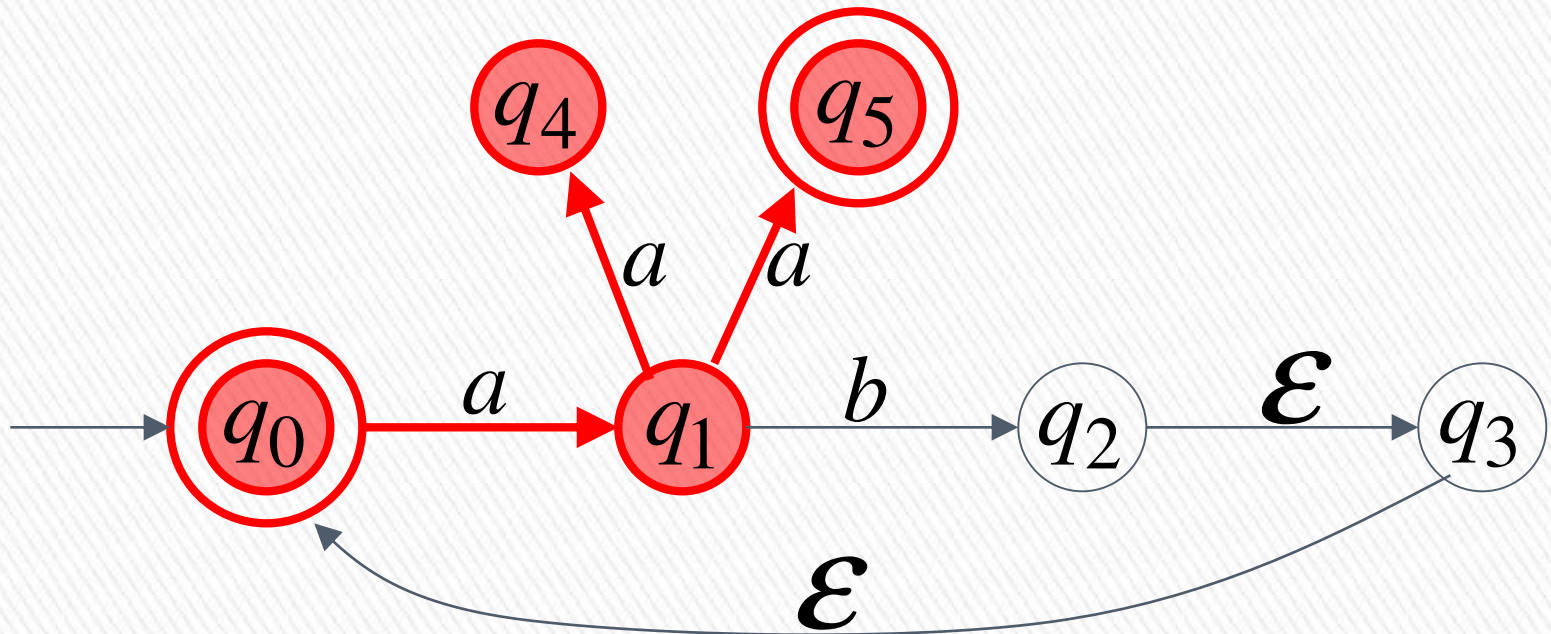
Extended Transition Function δ^*

Same with δ but applied on strings

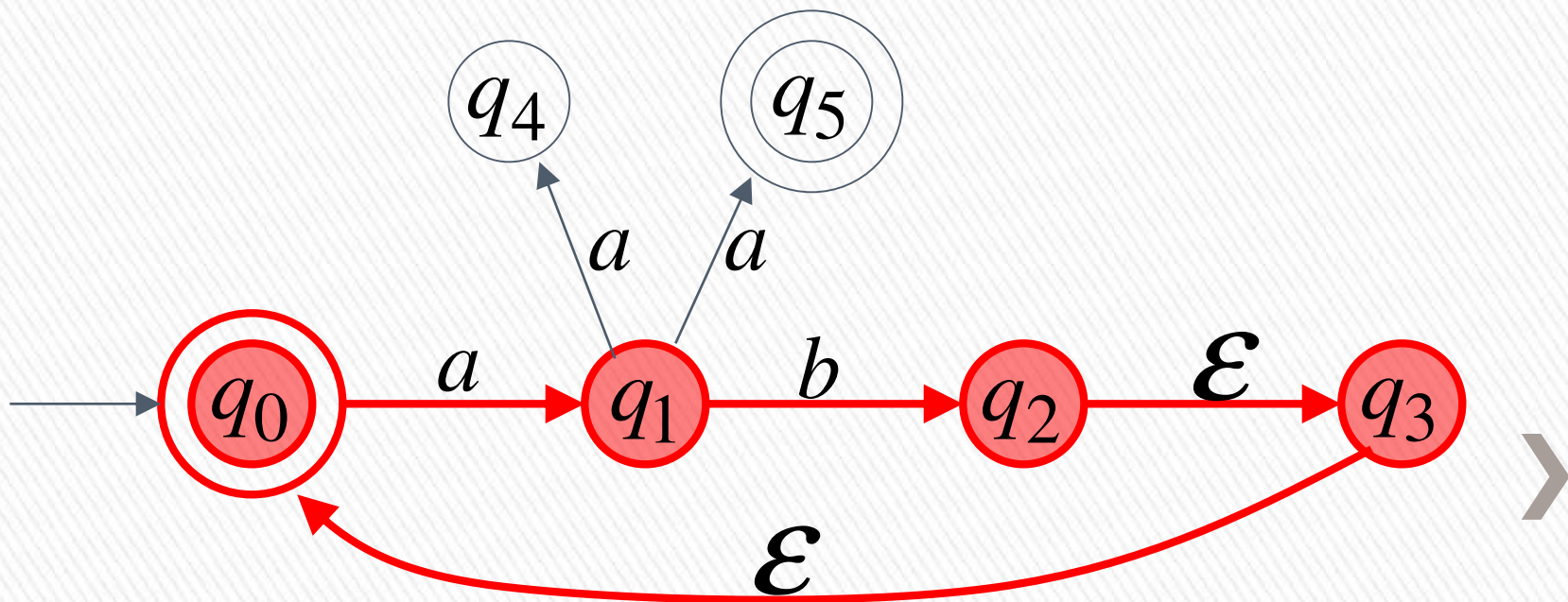
$$\delta^*(q_0, a) = \{q_1\}$$



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



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



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:

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

» where

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

» and there is some $q_k \in F$

(accepting state)

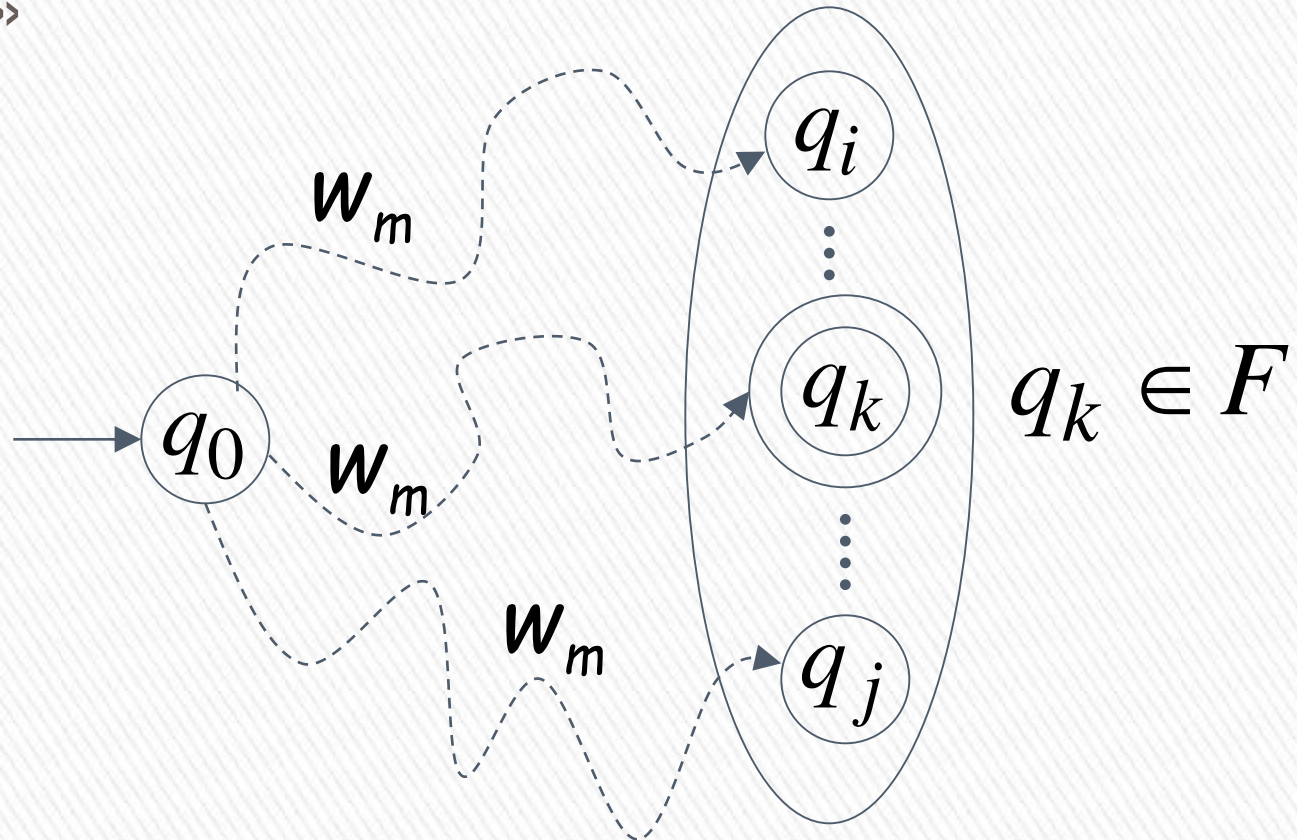


$$w_m \in L(M)$$

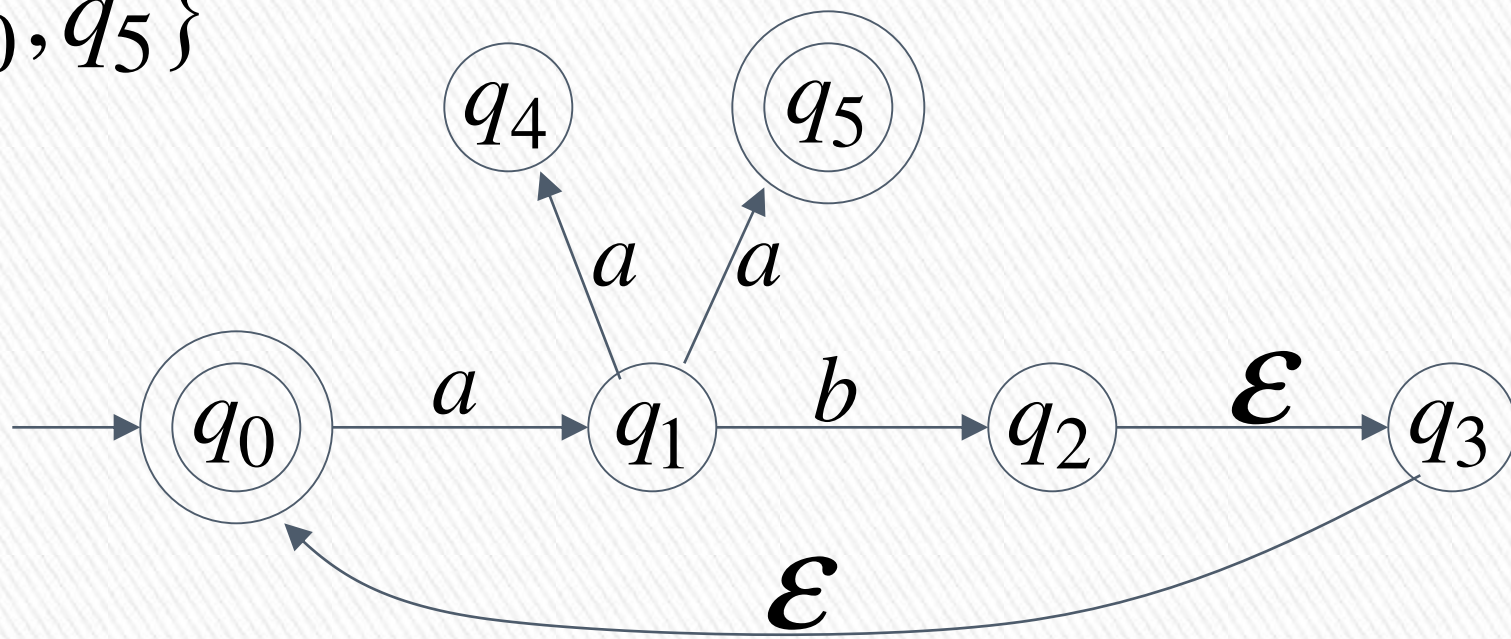
>>

>>

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



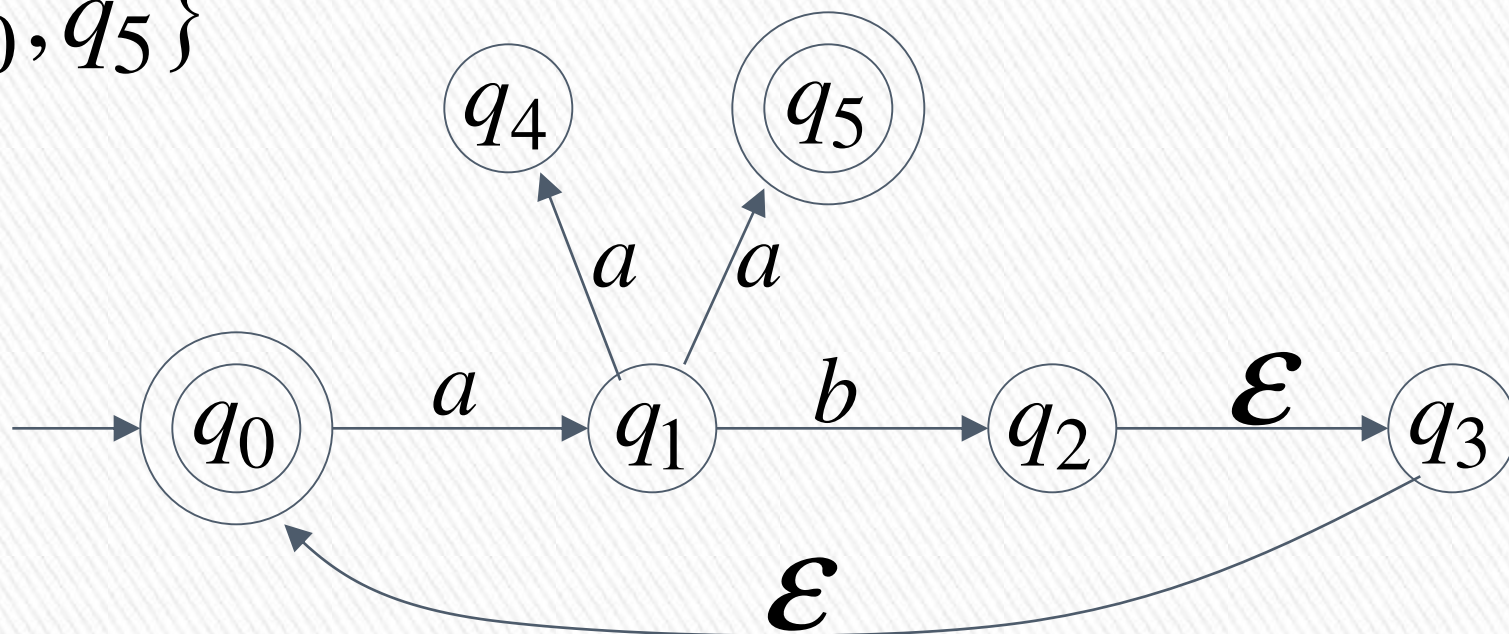
$$F = \{q_0, q_5\}$$



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

$\swarrow \in F$

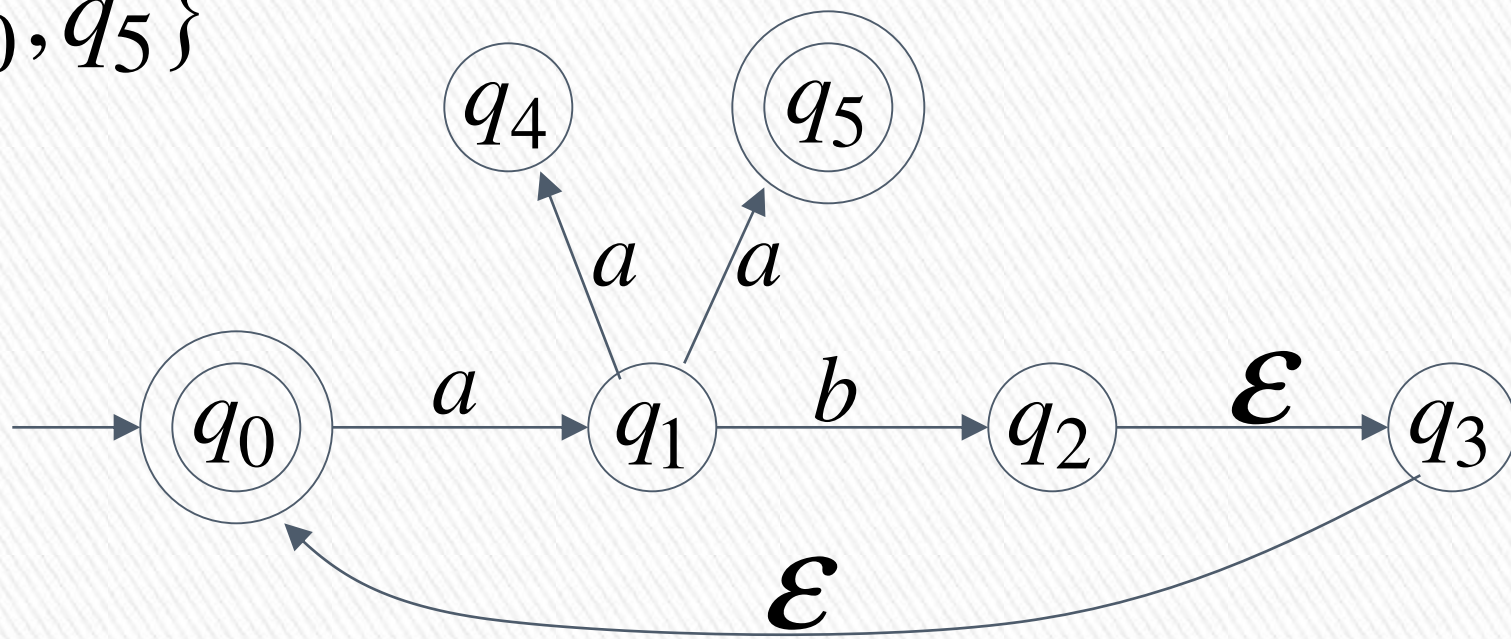
$$F = \{q_0, q_5\}$$



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

$\swarrow \in F$

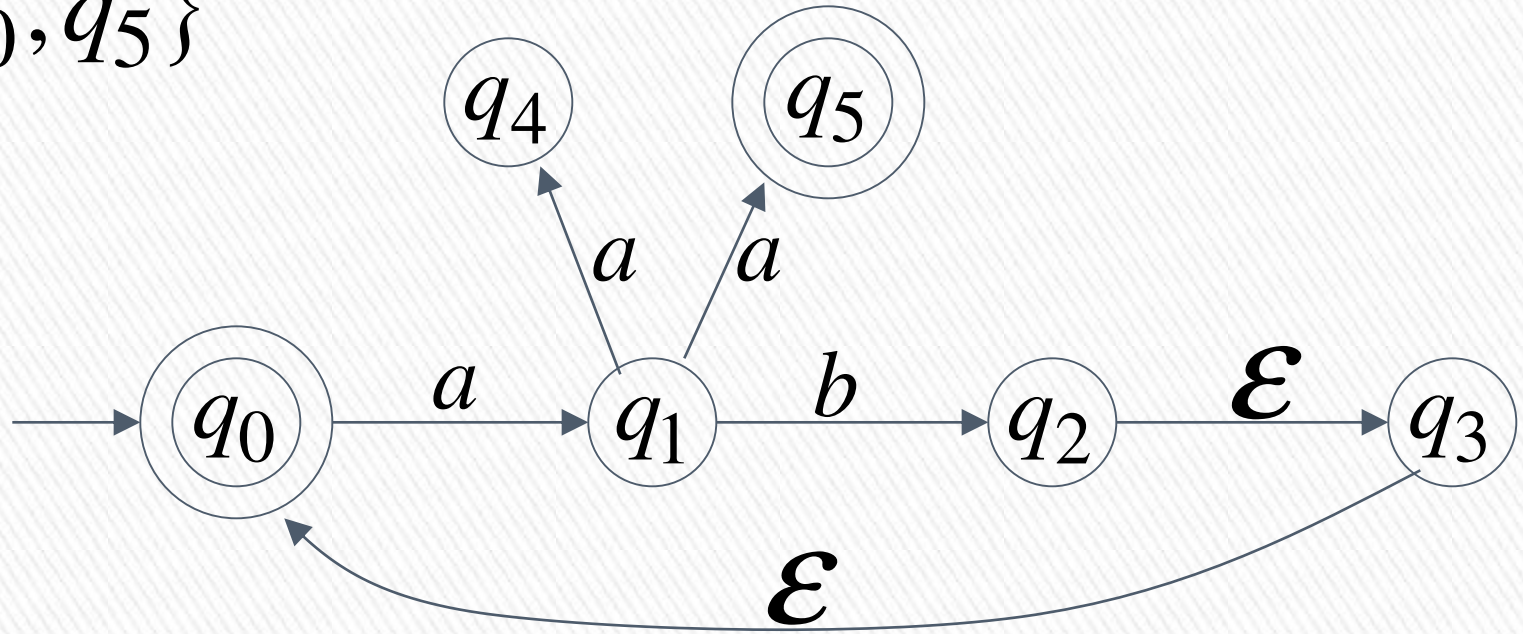
$$F = \{q_0, q_5\}$$



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

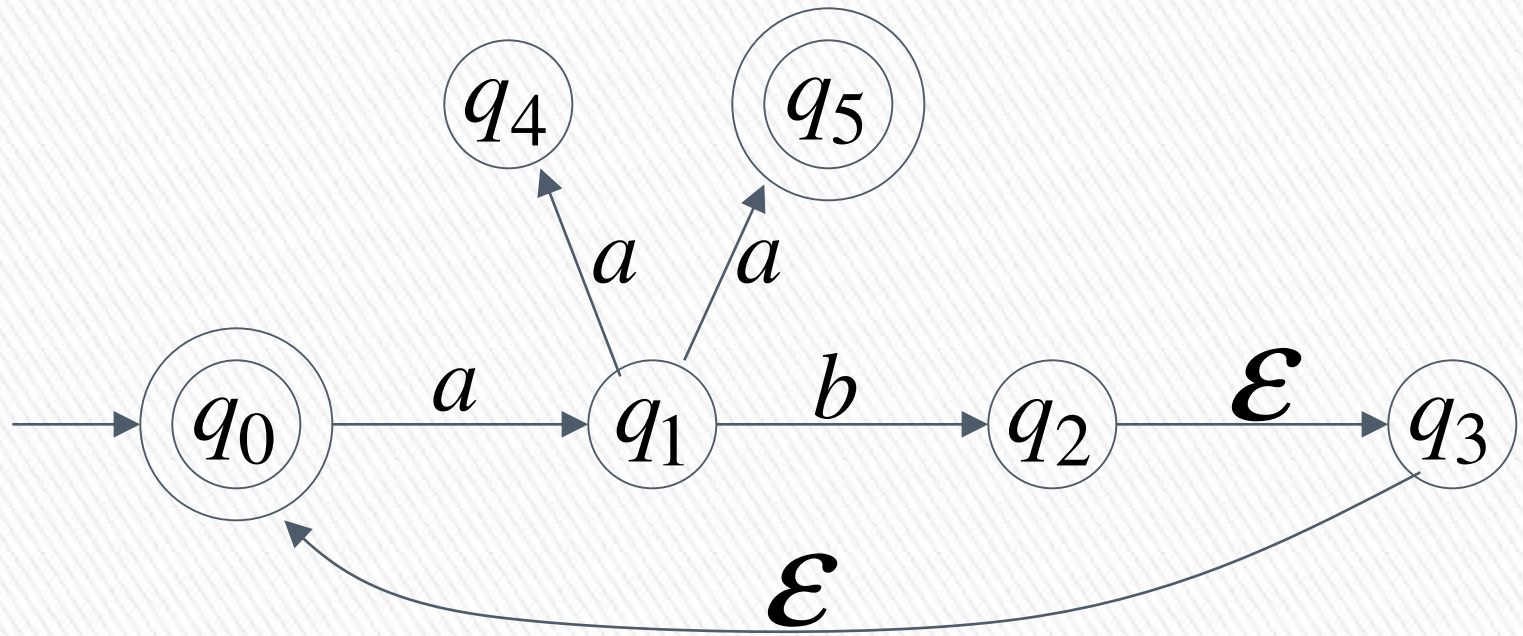
$\nwarrow \in F$
➤

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



$$\delta^*(q_0, aba) = \{q_1\} \xrightarrow{\text{yellow arrow}} aba \notin L(M) \triangleright$$

$\nwarrow \notin F$



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



Equivalence of Machines

» Definition:

» Machine M_1 is equivalent to machine M_2

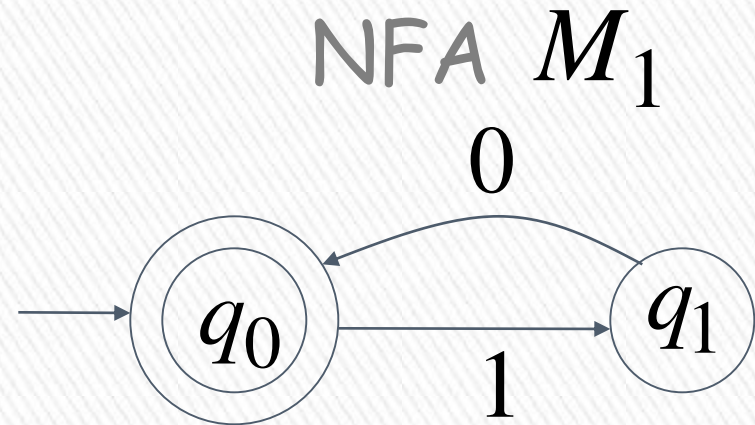
if $L(M_1) = L(M_2)$



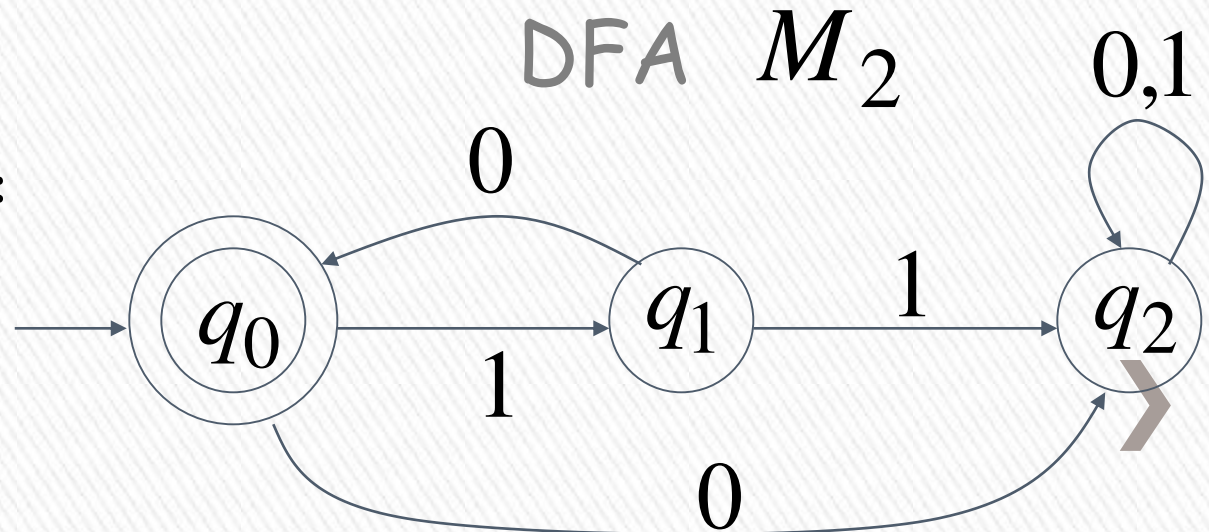
Example of equivalent machines

»

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



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



NFAs accept Regular Languages

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

Proof: we only 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 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 

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

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

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

We only need to prove:

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



NFA

Consider $w \in L(M)$



symbols

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



symbol



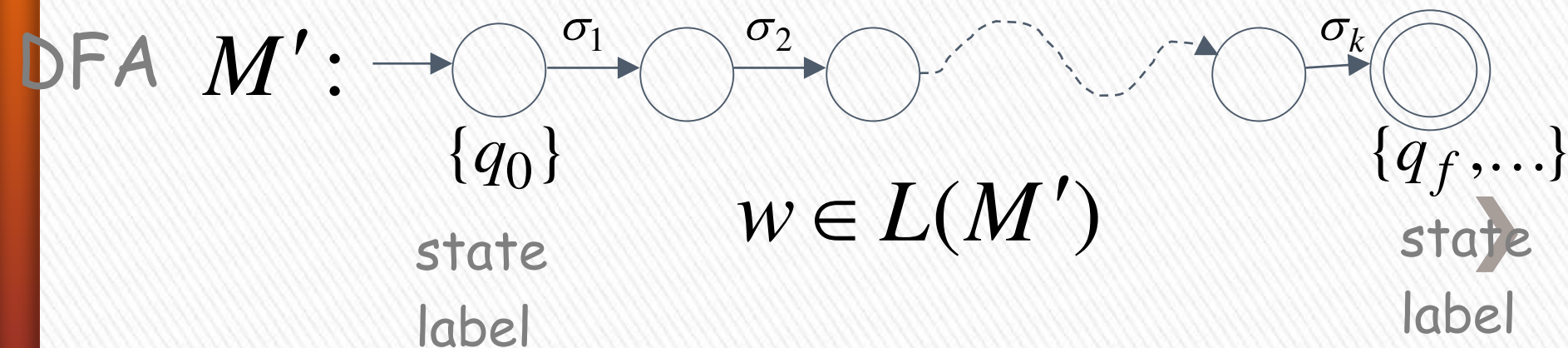
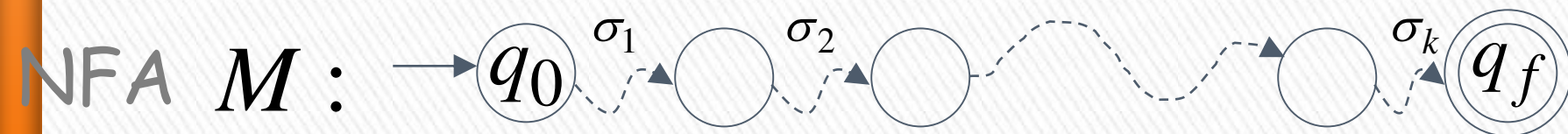
denotes a possible sub-path like

symbol



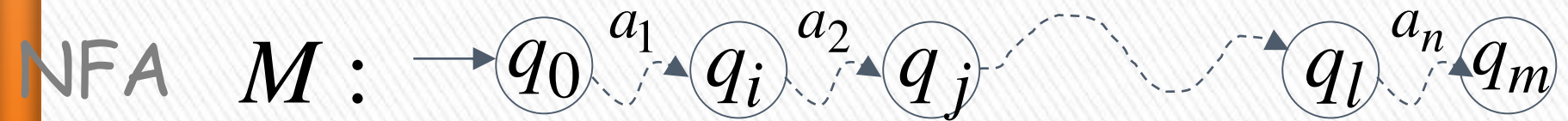
We will show that if $w \in L(M)$

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

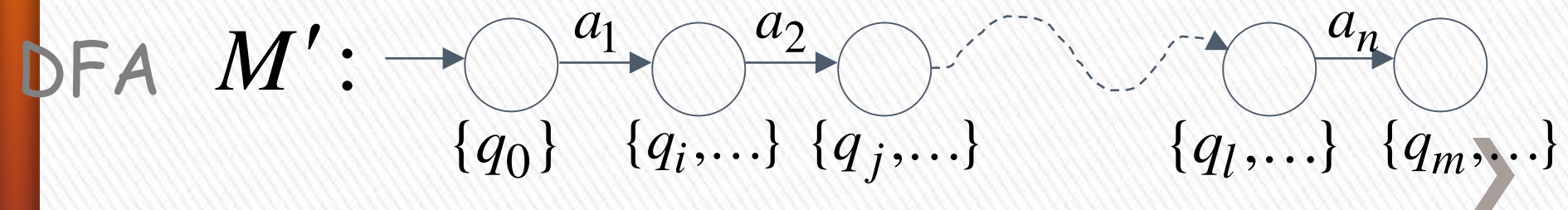


More generally, we will show that if in M :

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

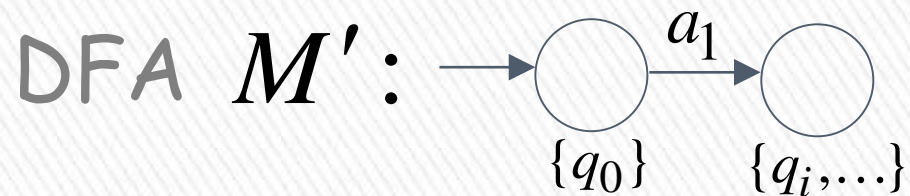


then



Proof by induction on $|v|$

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



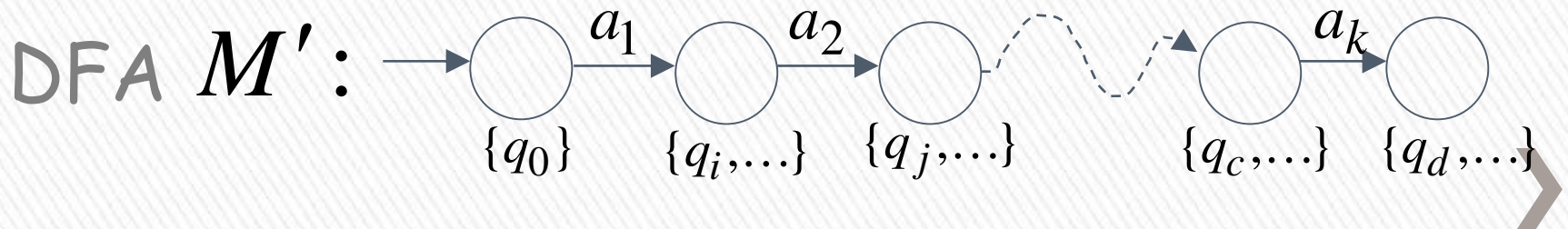
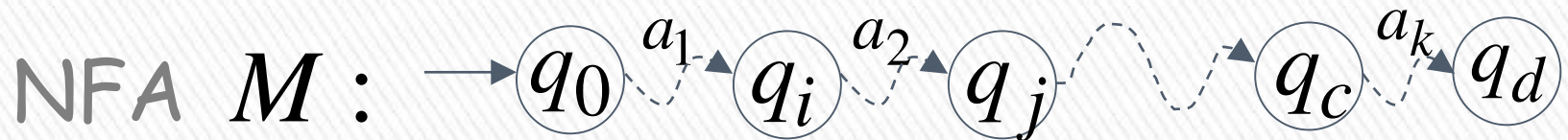
is true by construction of M'



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

$$v = a_1 a_2 \cdots a_k$$

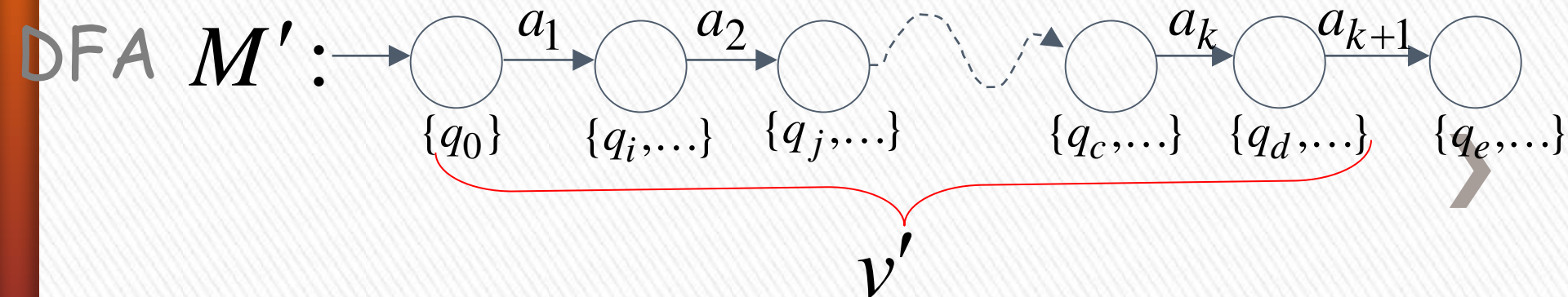
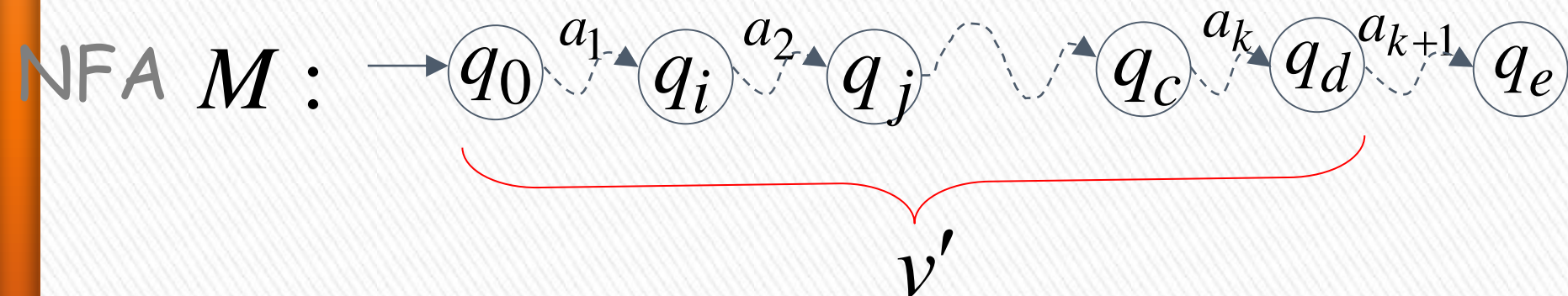
Suppose that the following hold



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

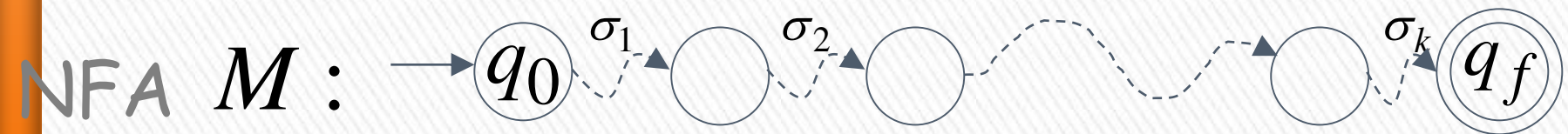
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

Then this is true by construction of M'

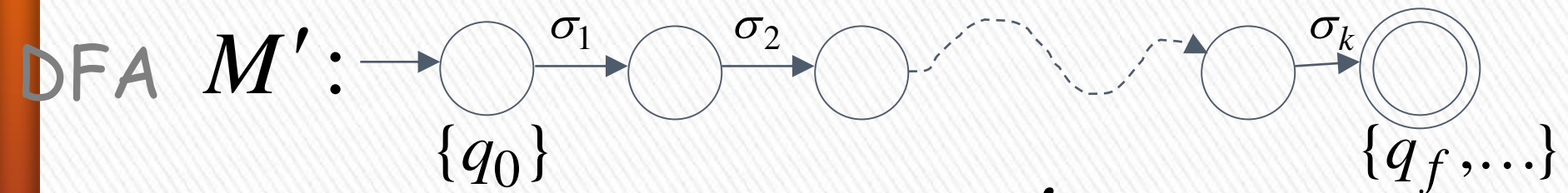


Therefore if $w \in L(M)$

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



then



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



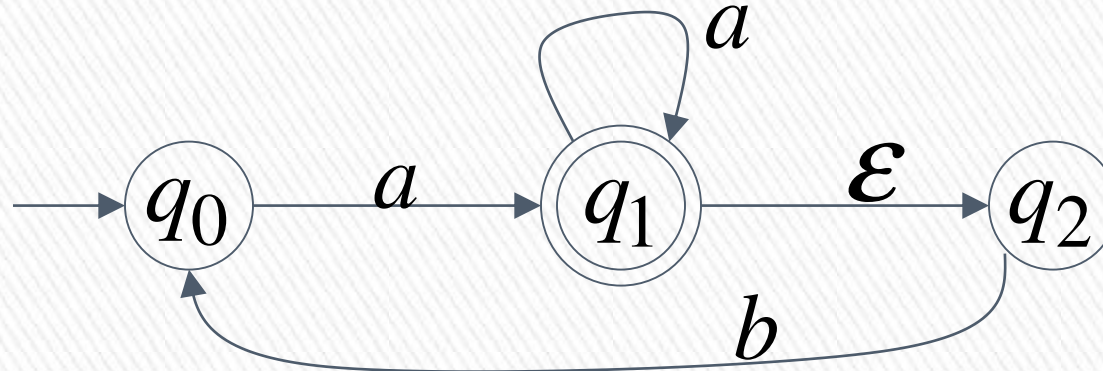
We have shown: $L(M) \subseteq L(M')$

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

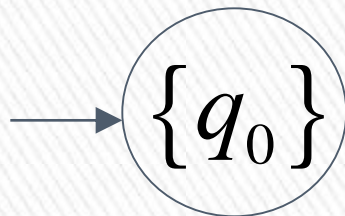
Therefore: $L(M) = L(M')$


END OF LEMMA PROOF

Conversion NFA to DFA



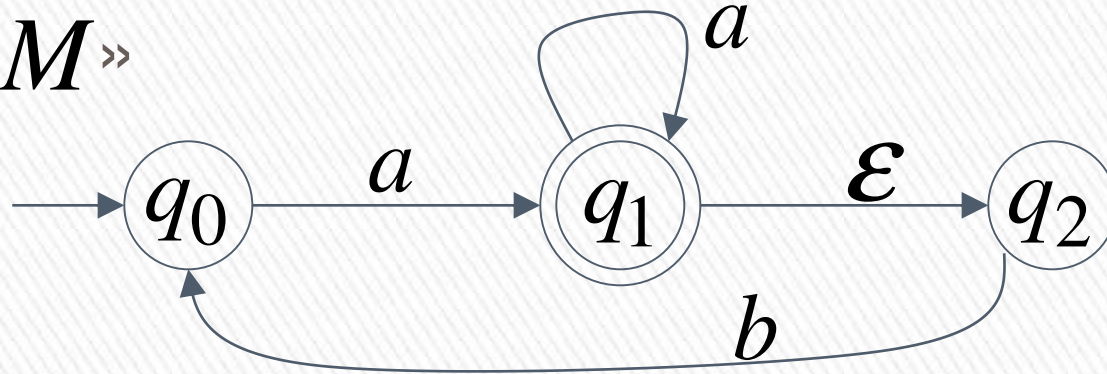
NFA M



DFA M'

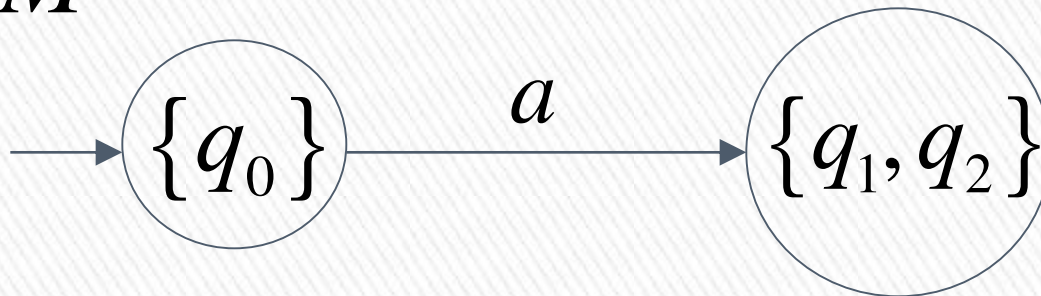


NFA $M \gg$



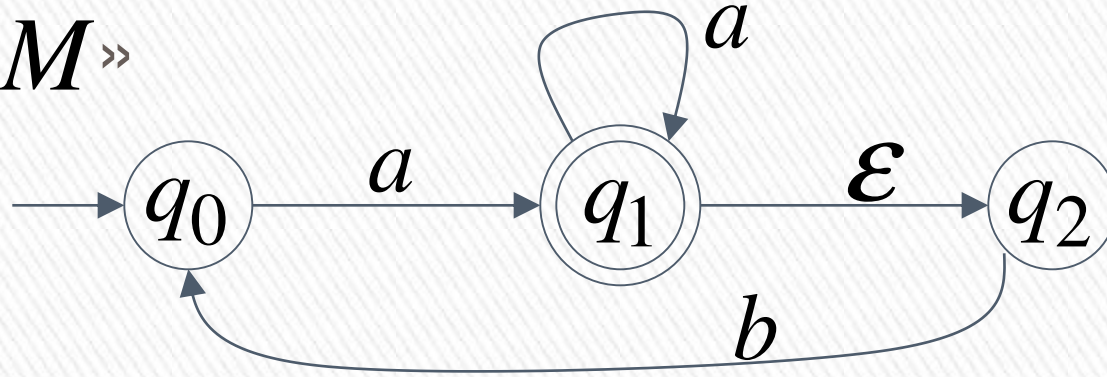
$$\delta^*(q_0, a) = \{q_1, q_2\}$$

DFA M'

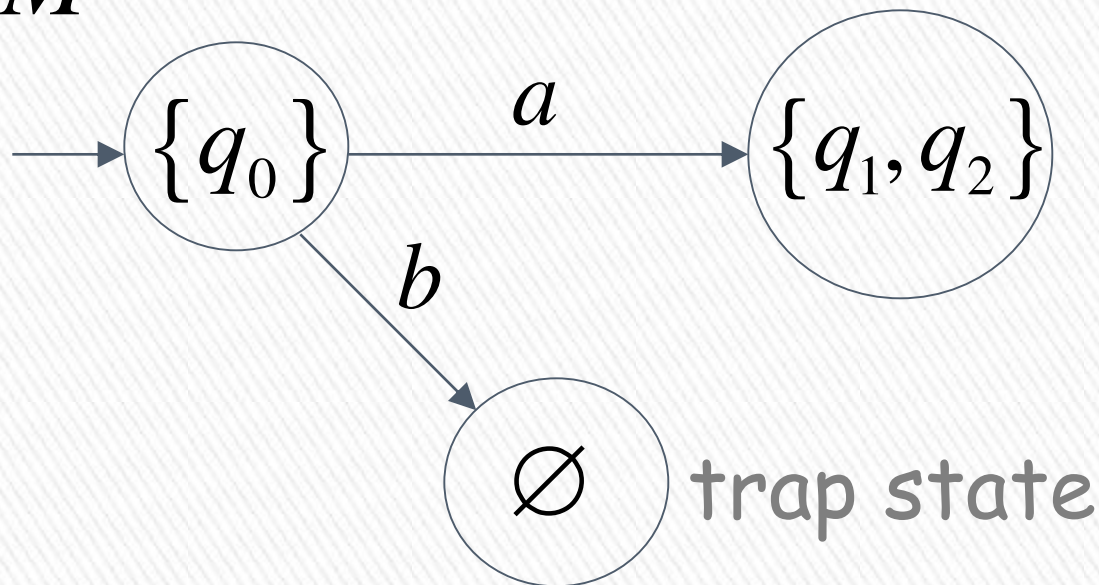


$$\delta^*(q_0, b) = \emptyset \quad \text{empty set}$$

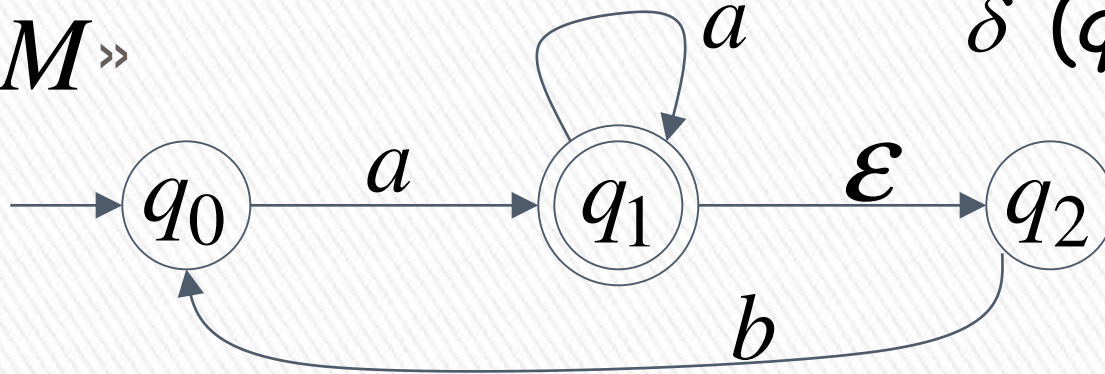
NFA M



DFA M'



NFA $M \gg$



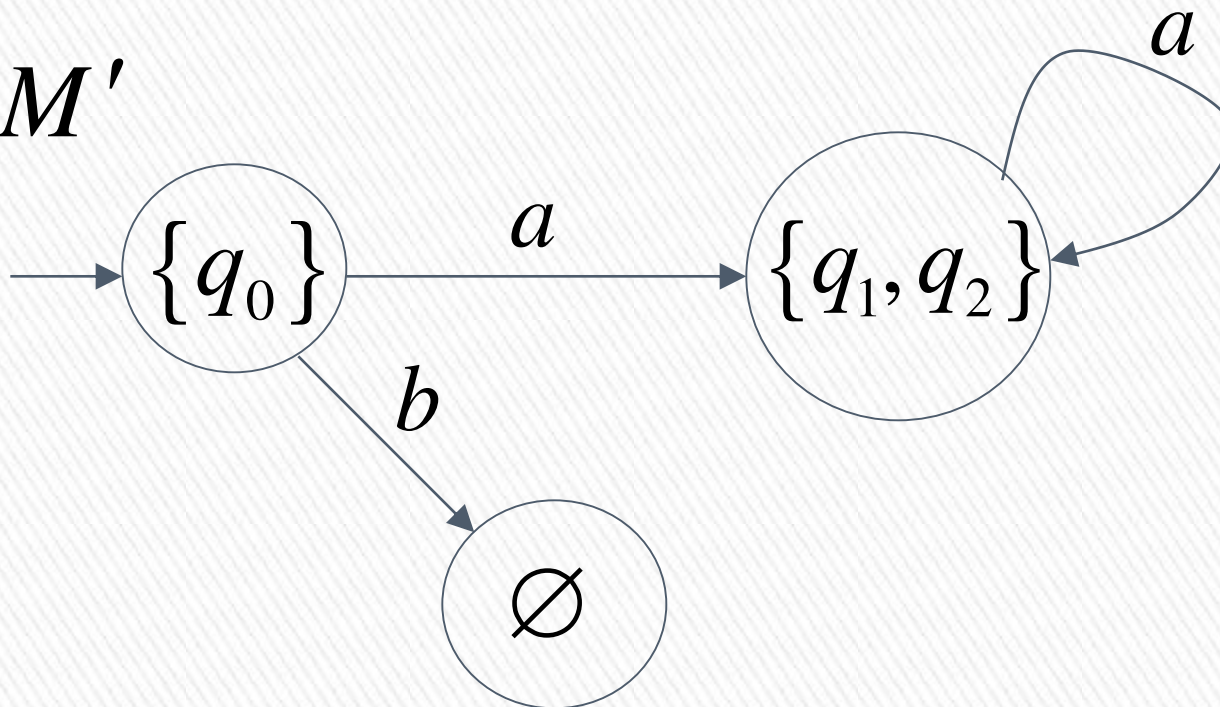
$$\delta^*(q_1, a) = \{q_1, q_2\}$$

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

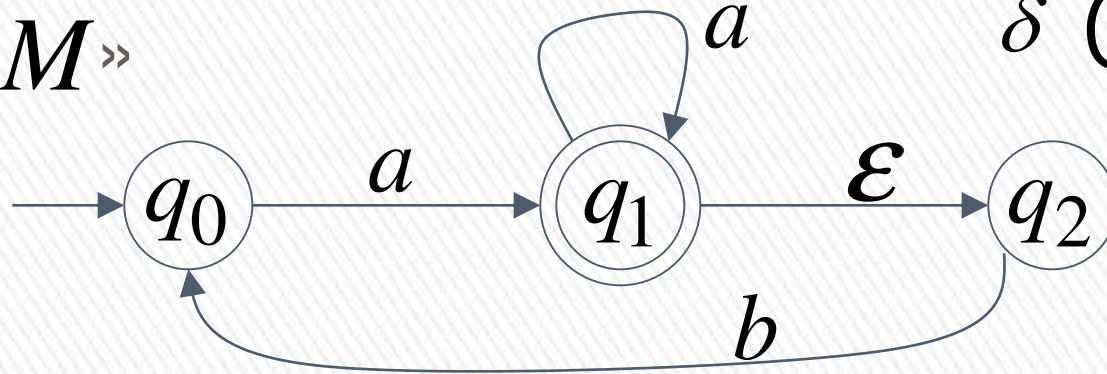
union

$$\{q_1, q_2\}$$

DFA M'



NFA $M \gg$



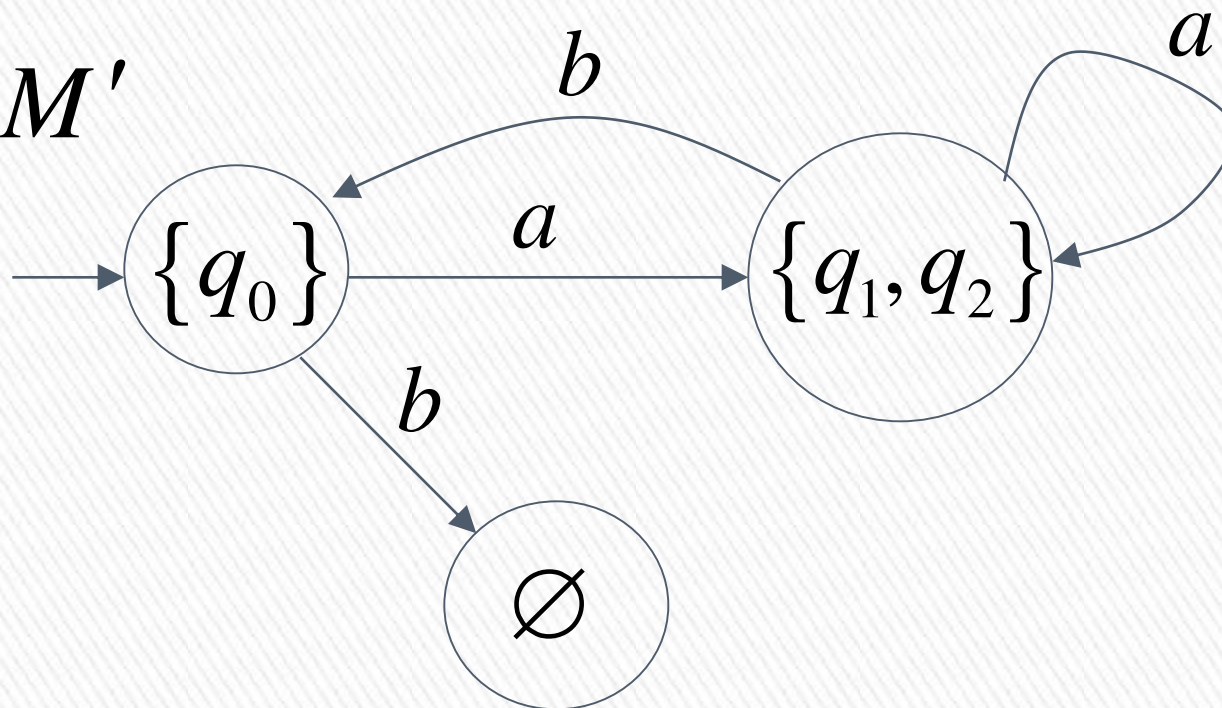
$$\delta^*(q_1, b) = \{q_0\}$$

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

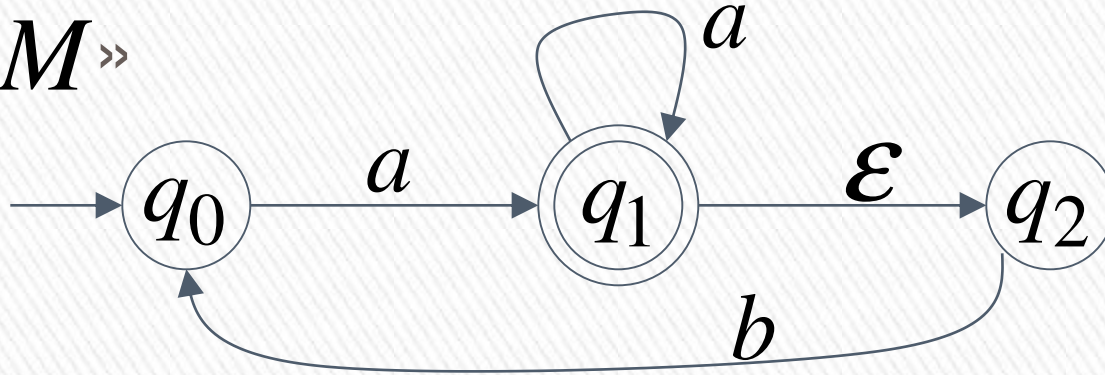
union

$\{q_0\}$

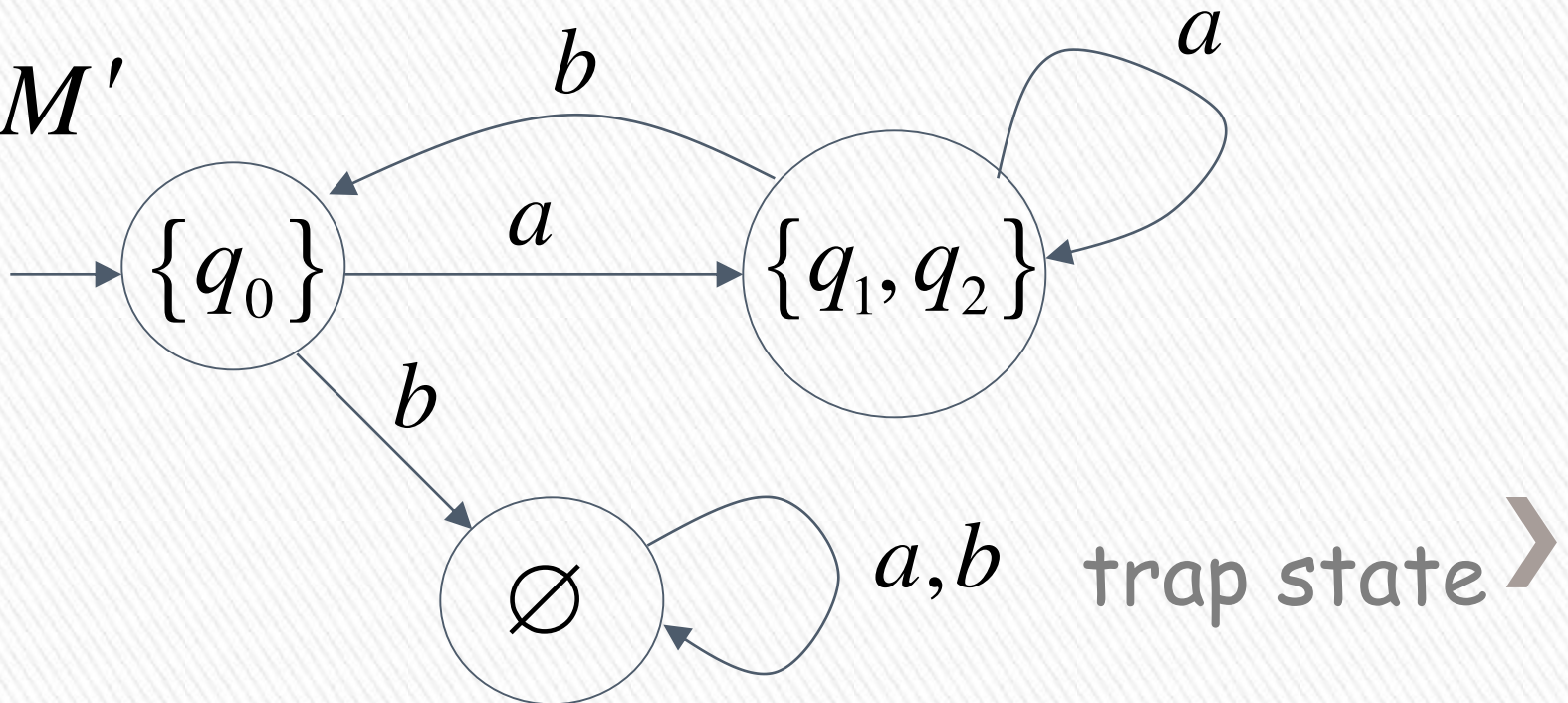
DFA M'



NFA M_{\gg}

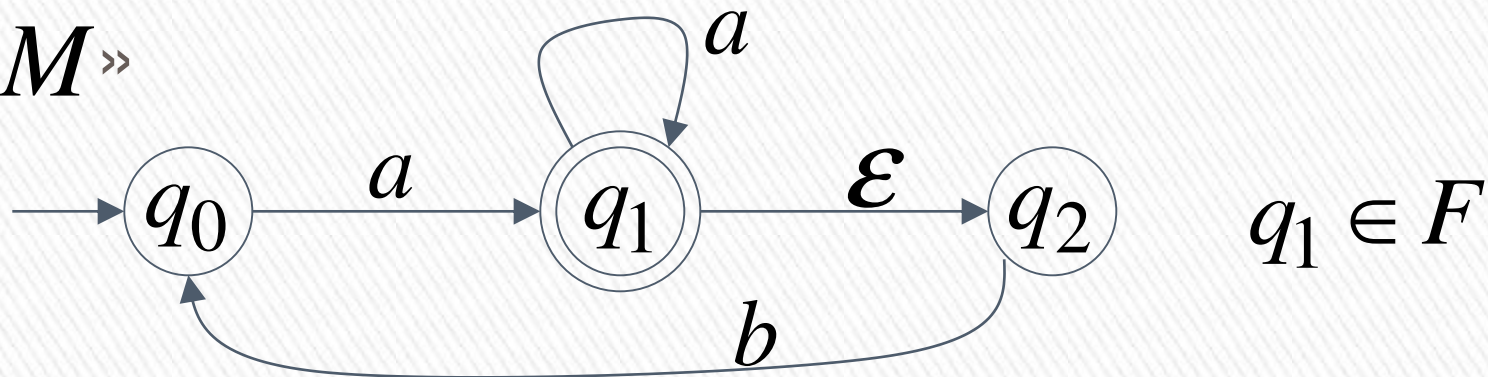


DFA M'

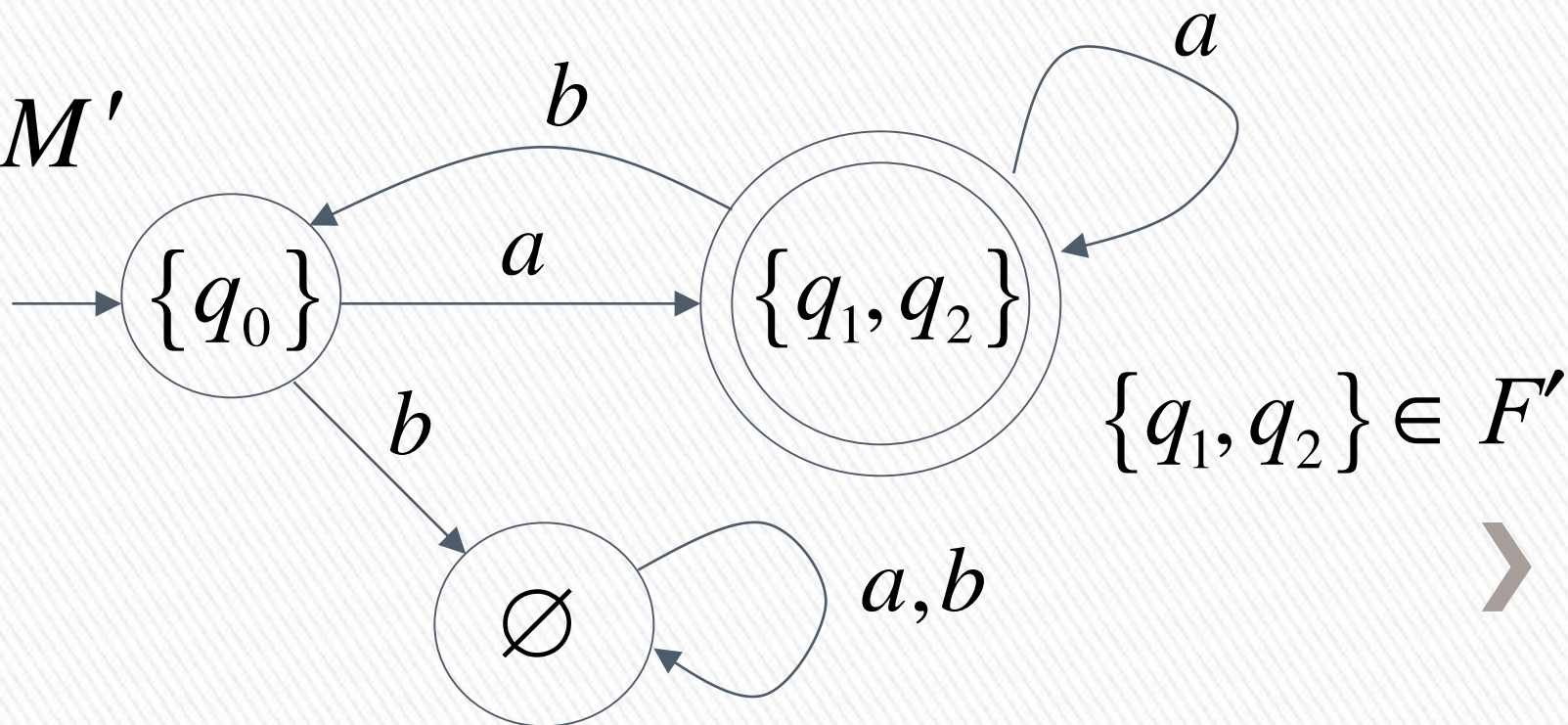


END OF CONSTRUCTION

NFA M_{\gg}



DFA M'



General Conversion Procedure

» **Input:** an NFA M

» **Output:** an equivalent DFA M'
with $L(M) = L(M')$



» The NFA has states

$$q_0, q_1, q_2, \dots$$

» The DFA has states from the power set

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



Conversion Procedure Steps

»

Step 1

» Initial state of NFA: q_0

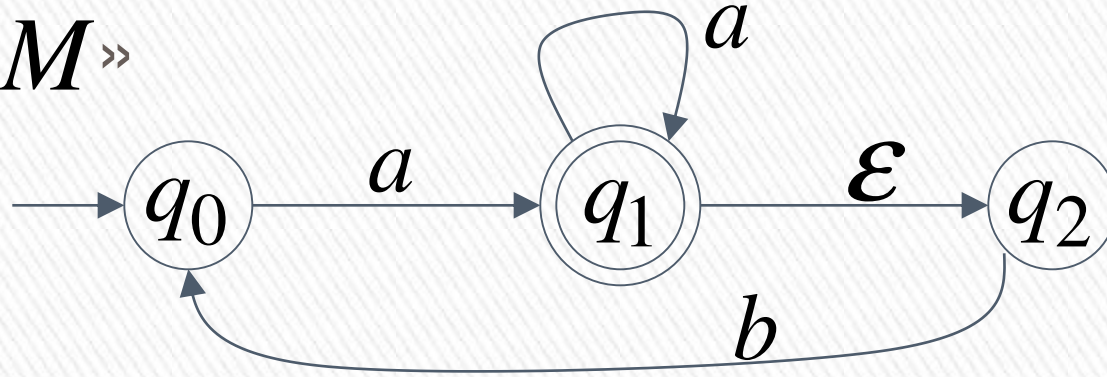


» Initial state of DFA: $\{q_0\}$

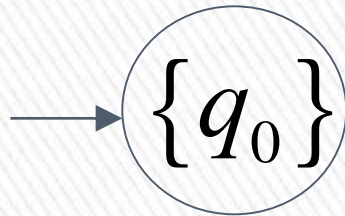


Example

NFA M



DFA M'



Step 2

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

compute in the NFA

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

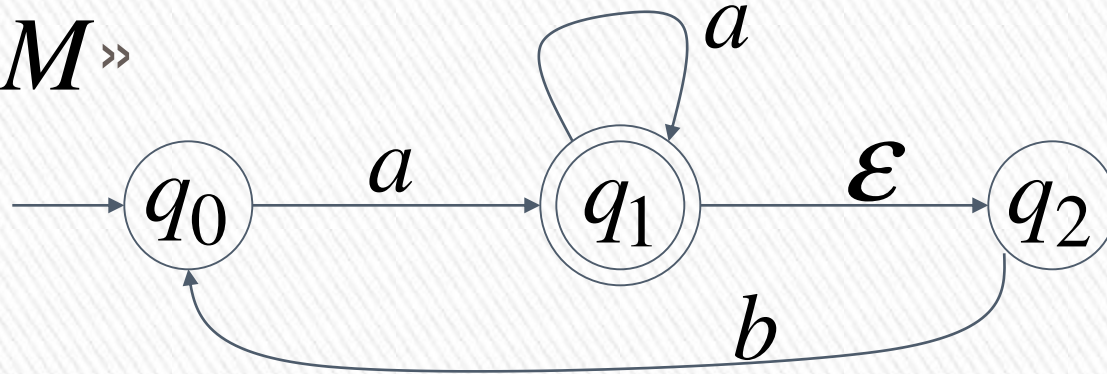
add transition to DFA

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

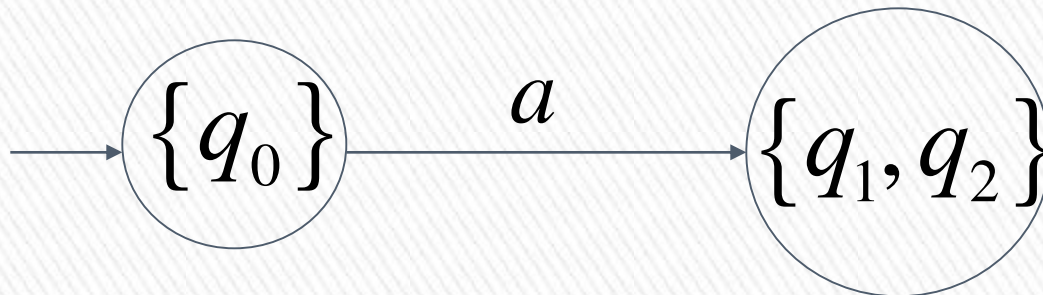


Example $\delta^*(q_0, a) = \{q_1, q_2\}$

NFA $M \gg$



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



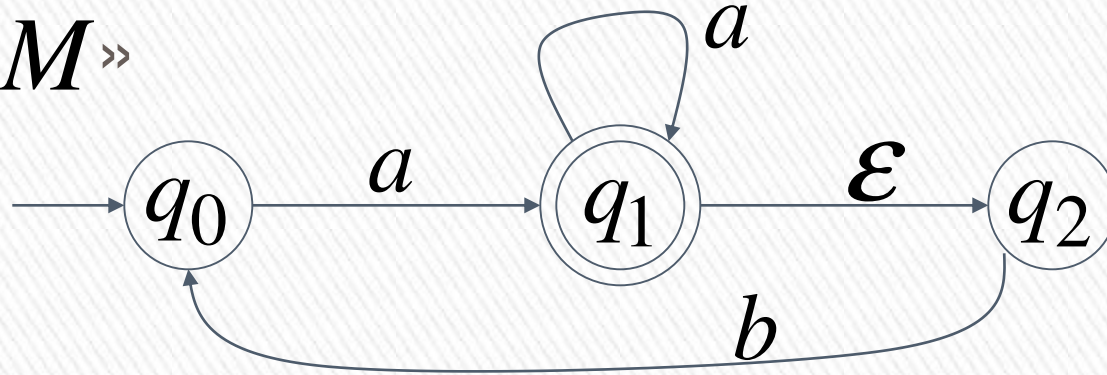
Step 3

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

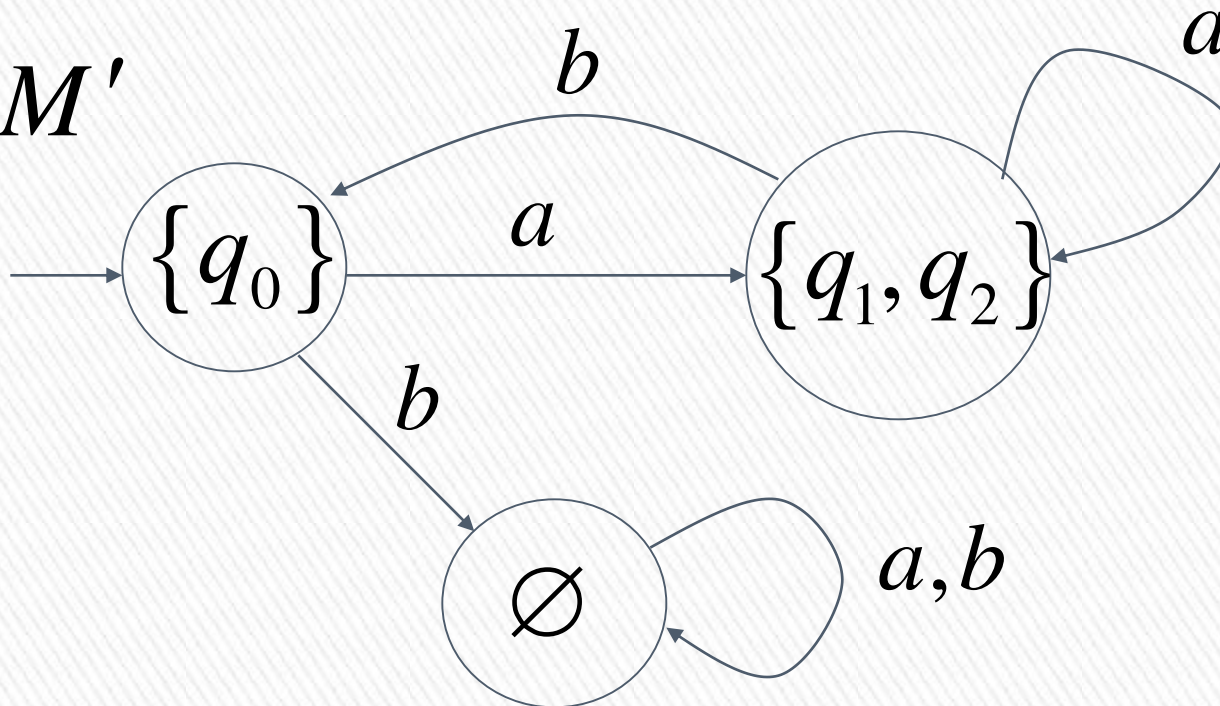


Example

NFA M



DFA M'



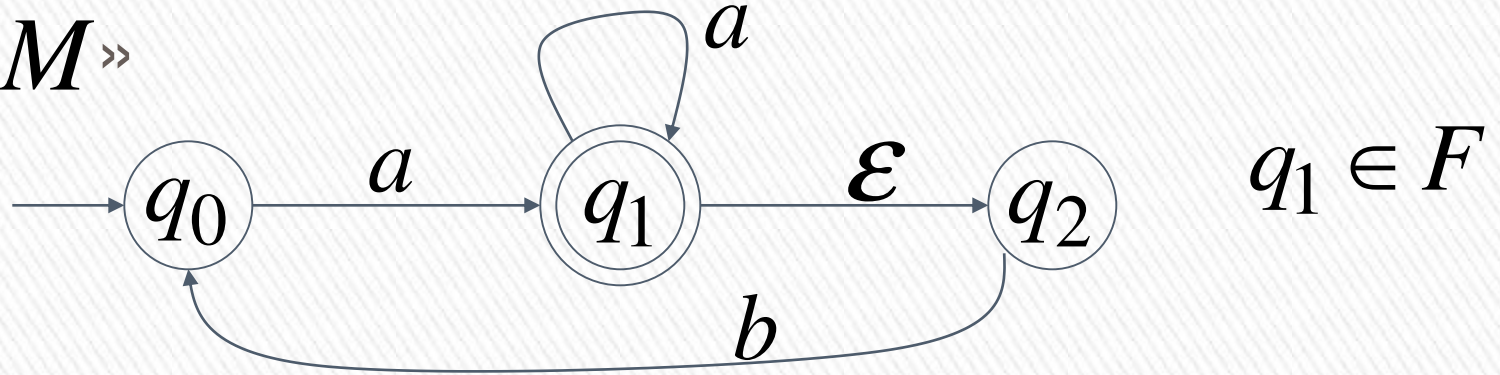
Step 4

- » For any DFA state $\{q_i, q_j, \dots, q_m\}$
- » if some q_j is accepting state in NFA
- » Then, $\{q_i, q_j, \dots, q_m\}$
is accepting state in DFA



Example

NFA $M \gg$



DFA M'

