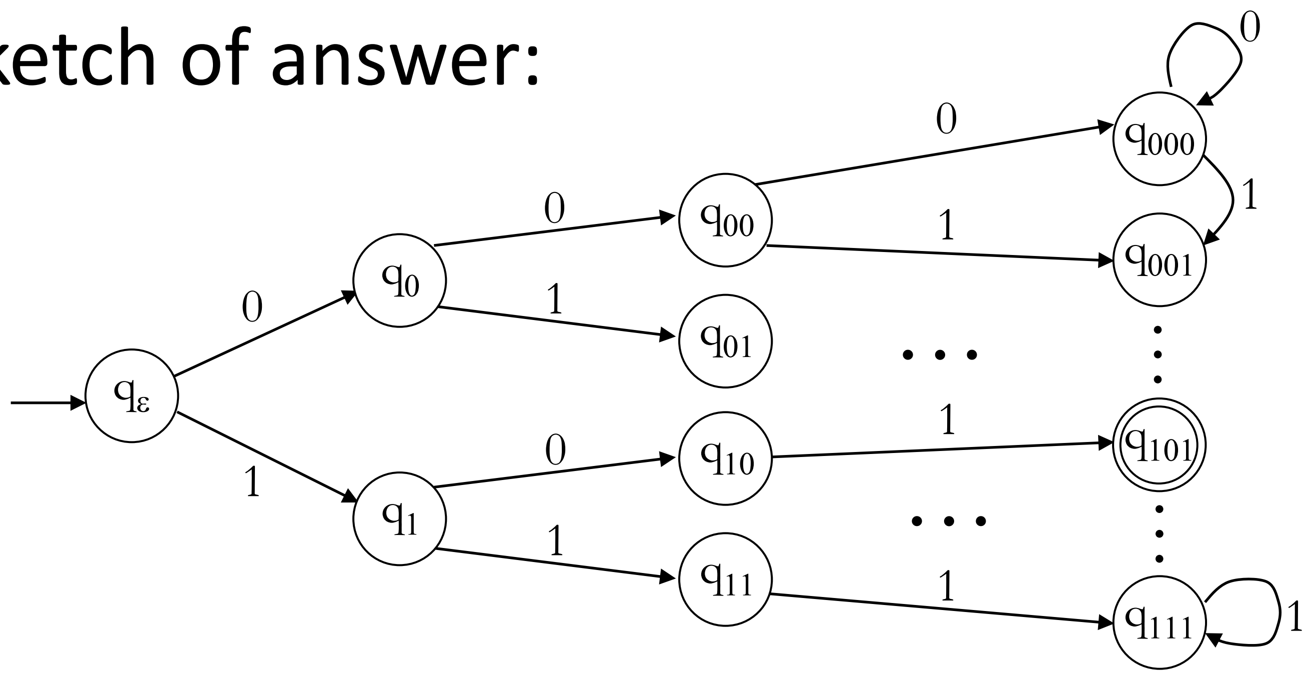
Lesson 4: Nondeterministic Finite Automata

Marc Gaetano

Edition 2018

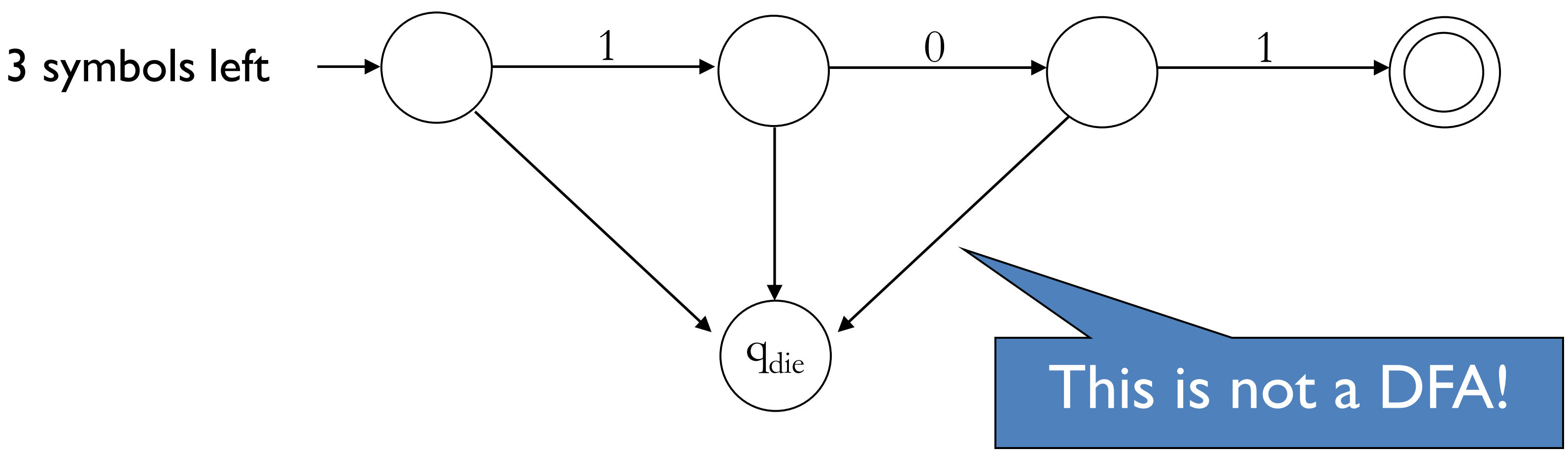
Example from last lesson

- Construct a DFA over alphabet $\{0, 1\}$ that accepts those strings that end in 101
- Sketch of answer:



Would be easier if…

- Suppose we could guess when the string we are reading has only 3 symbols left
- Then we could simply look for the sequence 101 and accept if we see it



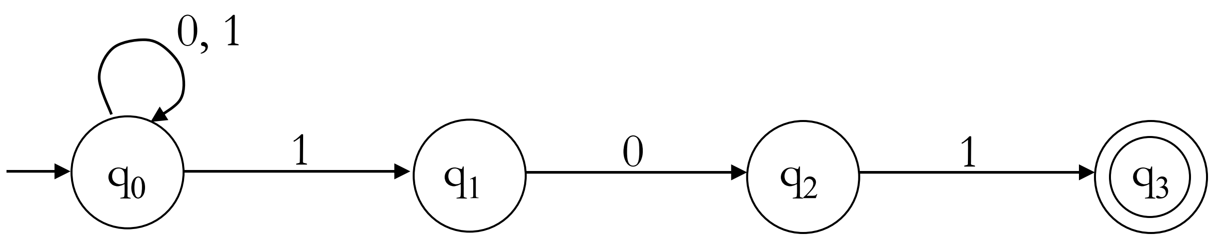
Nondeterminism

- Nondeterminism is the ability to make guesses, which we can later verify
- Informal nondeterministic algorithm for language of strings that end in 101:

1. Guess if you are approaching end of input
2. If guess is yes, look for 101 and accept if you see it
3. If guess is no, read one more symbol and go to step 1

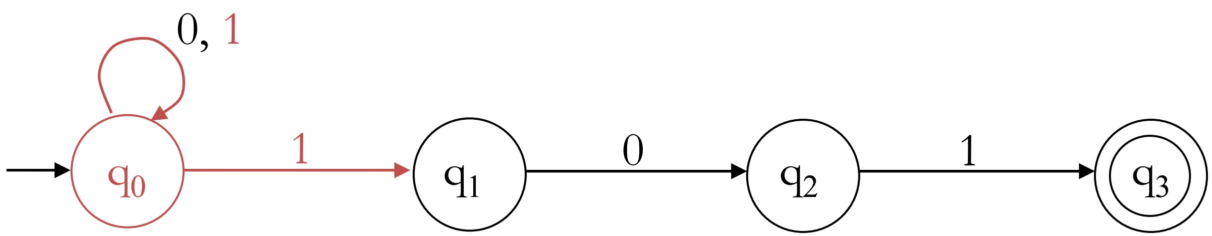
Nondeterministic finite automaton

- This is a kind of automaton that allows you to make guesses



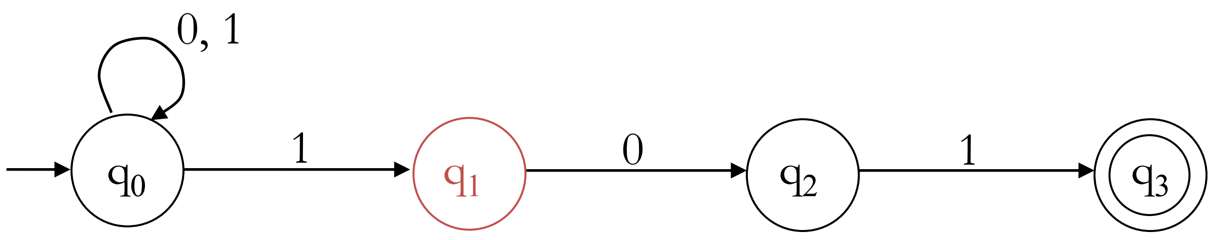
- Each state can have zero, one, or more transitions out labeled by the same symbol

Semantics of guessing



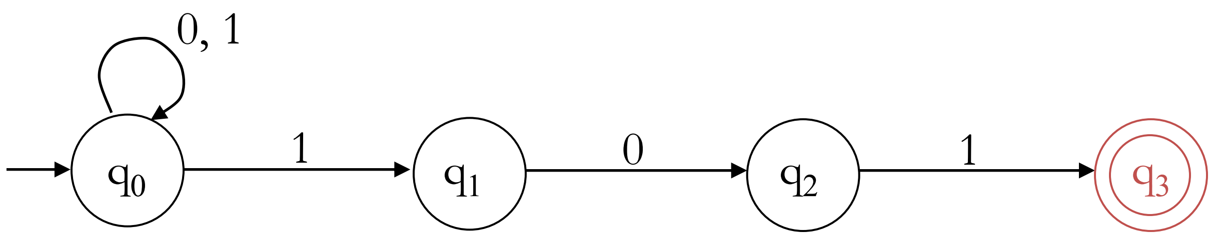
- State q_0 has two transitions labeled 1
- Upon reading 1, we have the choice of staying in q_0 or moving to q_1

Semantics of guessing



- State q_1 has no transition labeled 1
- Upon reading 1 in q_1, we die; upon reading 0, we continue to q_2

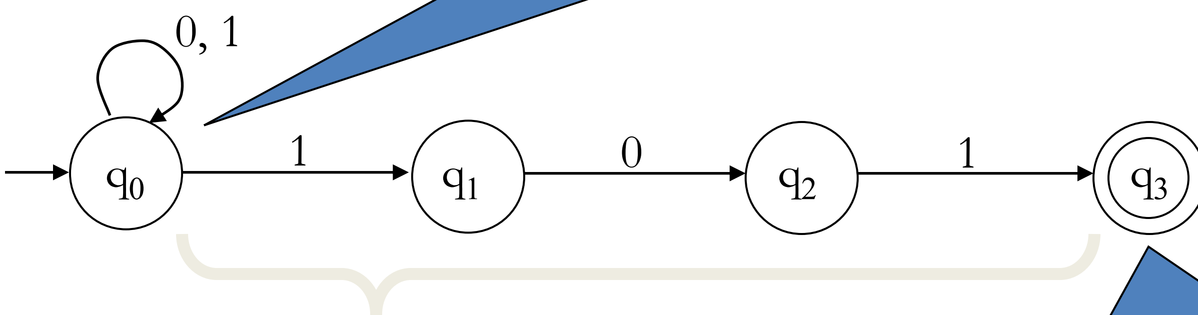
Semantics of guessing



- State q_3 has no transition going out
- Upon reading anything in q_3, we die

Meaning of automaton

Guess if you are 3 symbols away from end of input



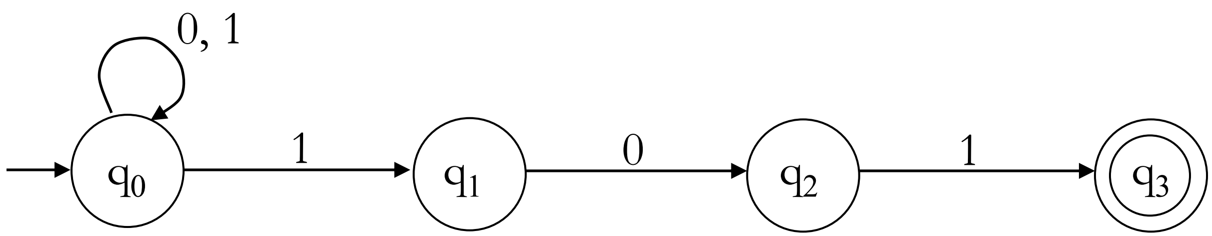
If so, guess you will see the pattern 101

Check that you are at the end of input

Formal definition

- A nondeterministic finite automaton (NFA) is a 5-tuple $(\mathcal{Q}, \Sigma, \delta, q_0, F)$ where
  - $\mathcal{Q}$ is a finite set of states
  - Σ is an alphabet
  - $\delta: \mathcal{Q} \times \Sigma \rightarrow$ subsets of $\mathcal{Q}$ is a transition function
  - $q_0 \in \mathcal{Q}$ is the initial state
  - $F \subseteq \mathcal{Q}$ is a set of accepting states (or final states).
- Only difference from DFA is that output of δ is a set of states

Example



alphabet $\Sigma = \{0, 1\}$
start state $\mathcal{Q} = \{q_0, q_1, q_2, q_3\}$
initial state q_0
accepting states $F = \{q_3\}$

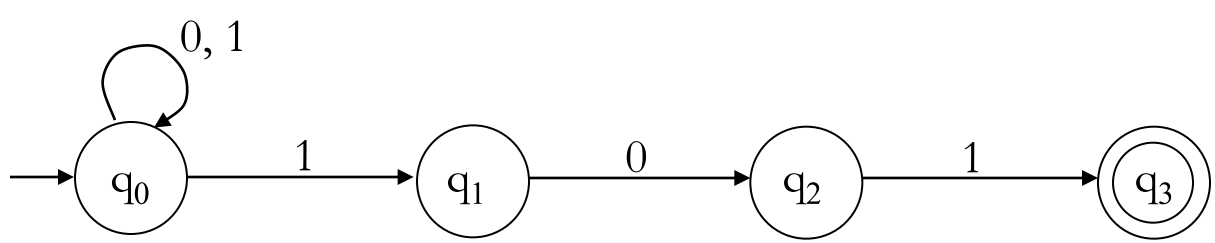
transition function δ:

states \ inputs	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\varnothing$

Language of an NFA

The language of an NFA is the set of all strings for which there is some path that, starting from the initial state, leads to an accepting state as the string is read left to right.

- Example



  - 1101 is accepted, but 0110 is not

NFAs are as powerful as DFAs

- Obviously, an NFA can do everything a DFA can do
- But can it do more?

NFAs are as powerful as DFAs

- Obviously, an NFA can do everything a DFA can do
- But can it do more?

NO!

- Theorem

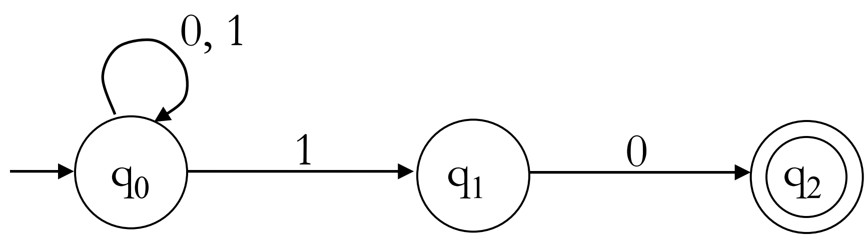
A language L is accepted by some DFA if and only if it is accepted by some NFA.

Proof of theorem

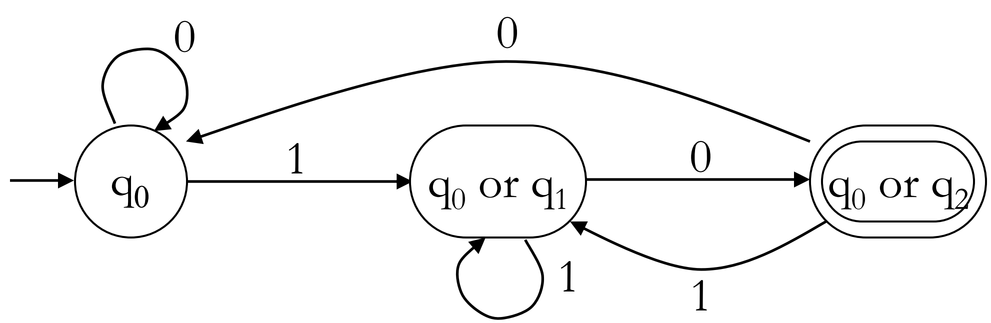
- To prove the theorem, we have to show that for every NFA there is a DFA that accepts the same language
- We will give a general method for simulating any NFA by a DFA
- Let’s do an example first

Simulation example

NFA:



DFA:



General method

	NFA	DFA
states	$q_0, q_1, \ldots, q_n$	$\varnothing, \{q_0\}, \{q_1\}, \{q_0,q_1\}, \ldots, \{q_0,\ldots,q_n\}$ one for each subset of states in the NFA
initial state	q_0	$\{q_0\}$
transitions	δ	$\delta'(\{q_{i1},\ldots,q_{ik}\}, a) =$ $\delta(q_{i1}, a) \cup \ldots \cup \delta(q_{ik}, a)$
accepting states	$F \subseteq Q$	$F' = \{S: S$ contains some state in $F\}$

Proof of correctness

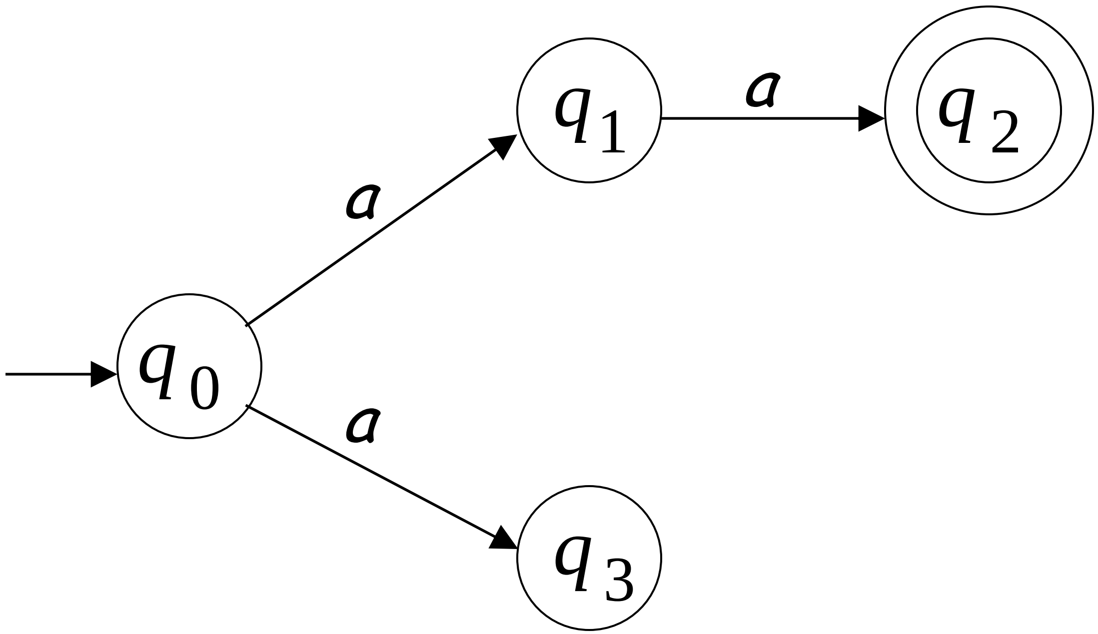
- Lemma

After reading n symbols, the DFA is in state $\{q_{i1},\ldots,q_{ik}\}$ if and only if the NFA is in one of the states $q_{i1},\ldots,q_{ik}$

- Proof by induction on n
- At the end, the DFA accepts iff it is in a state that contains some accepting state of NFA
- By lemma, this is true iff the NFA can reach an accepting state

Example of NFA

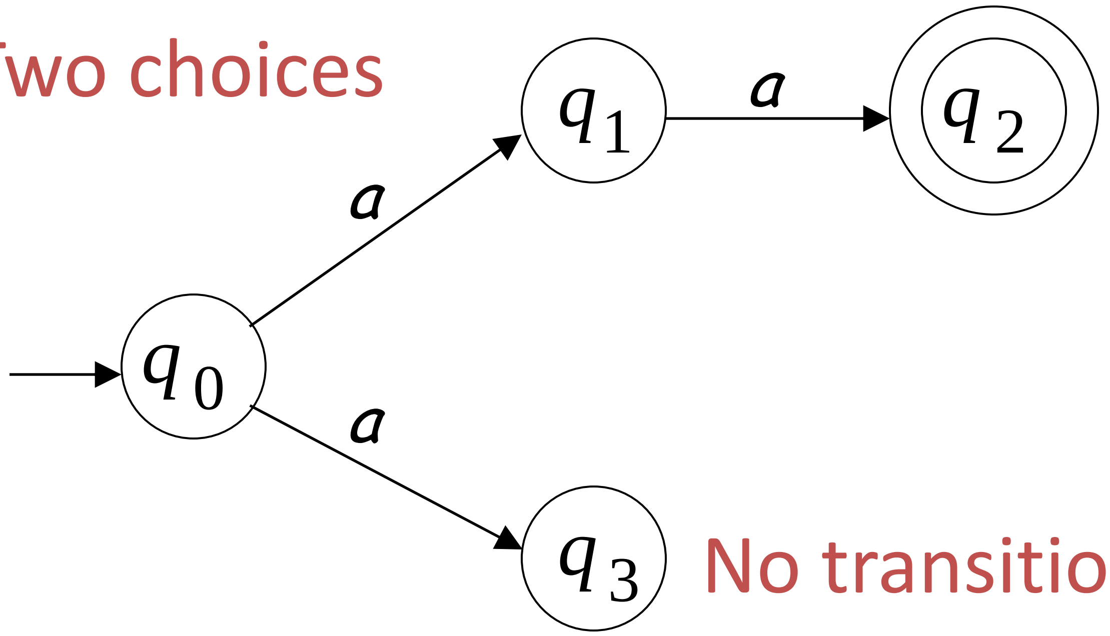
Alphabet = $\{a\}$



Example of NFA

Alphabet = $\{a\}$

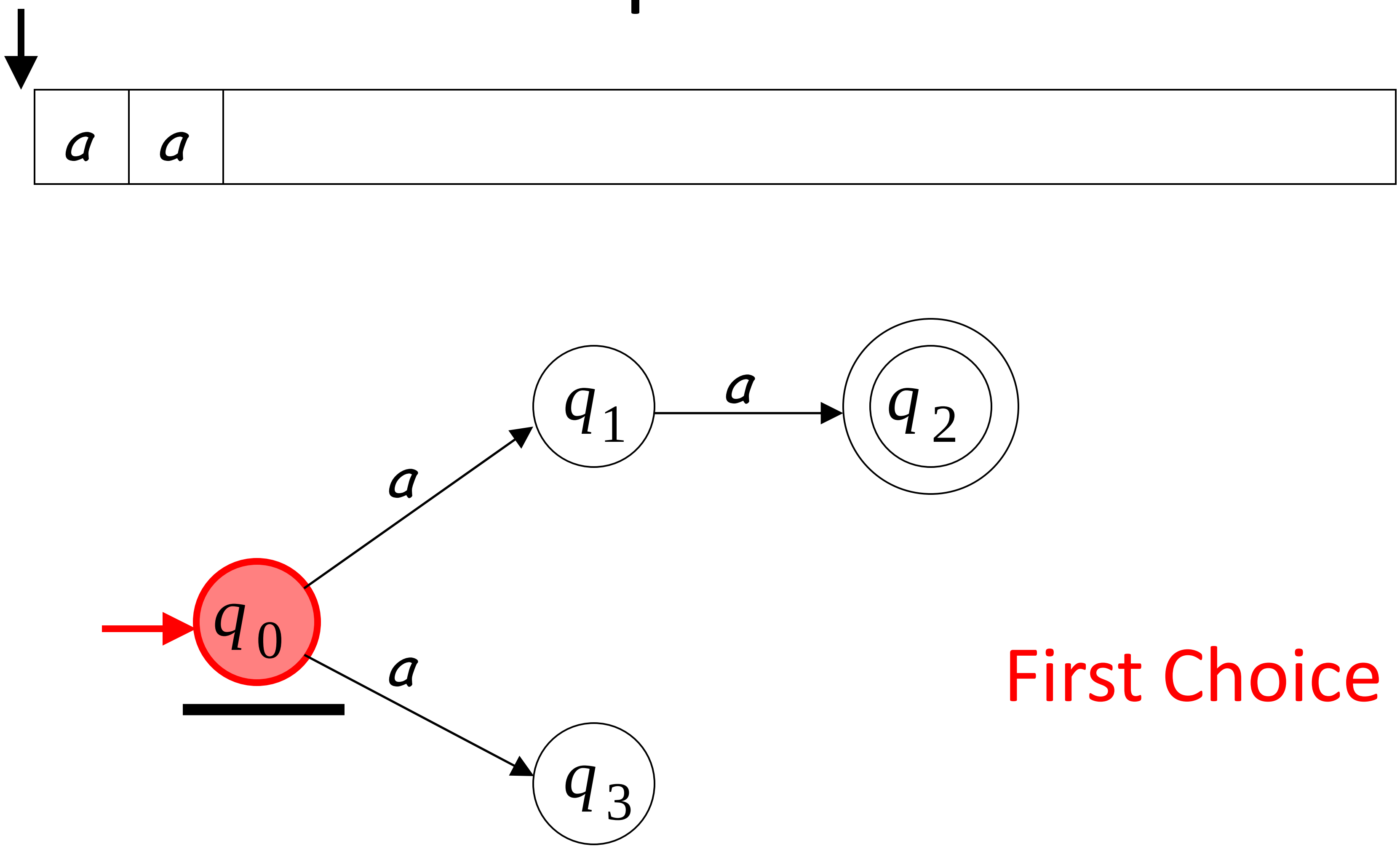
Two choices



No transition

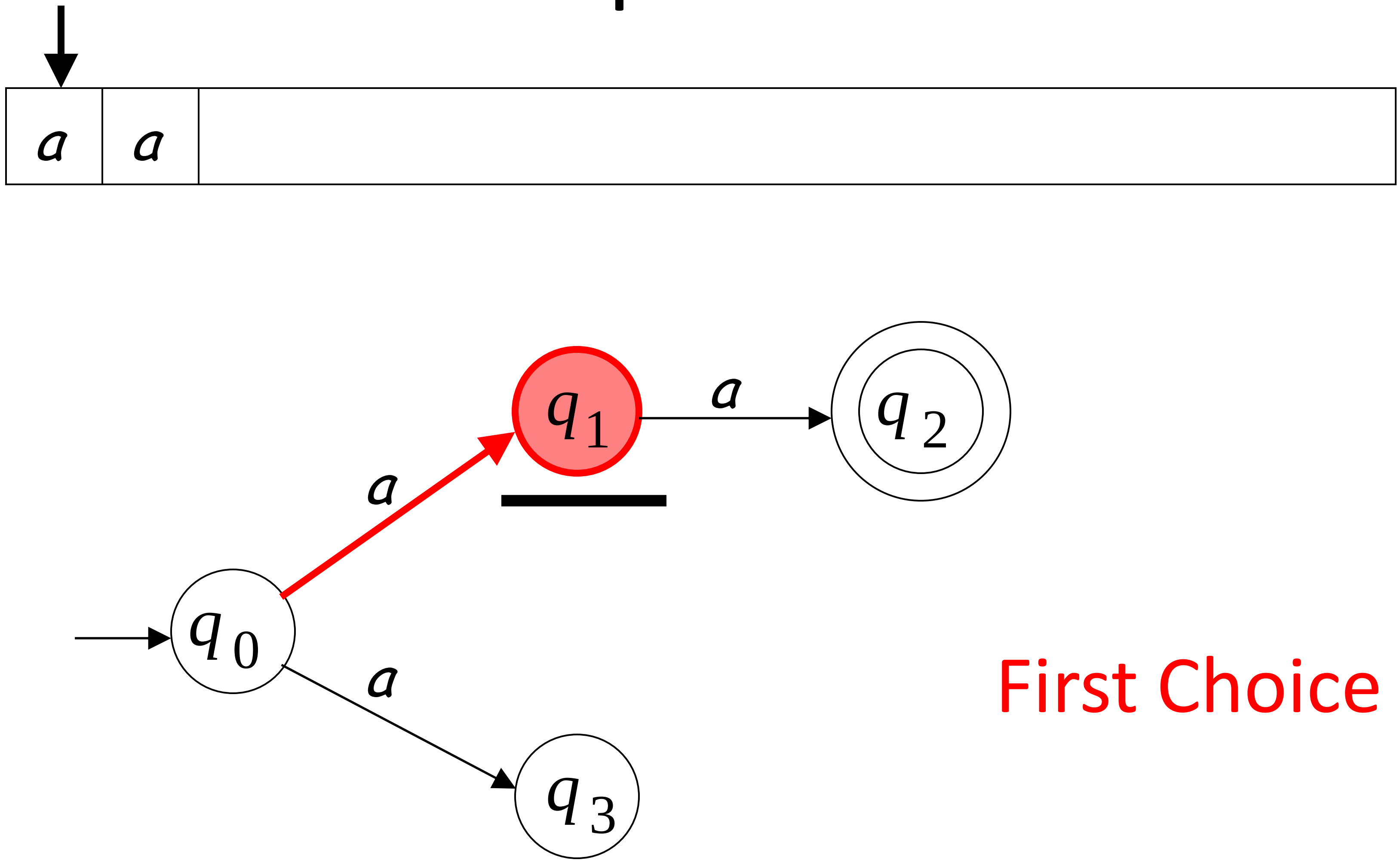
No transition

Example of NFA



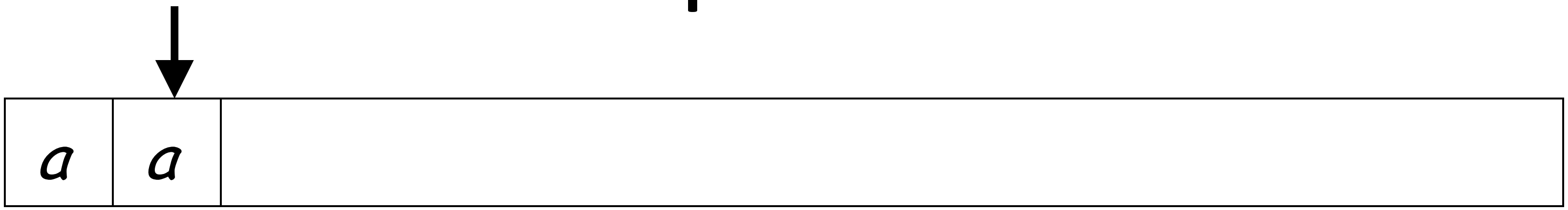
First Choice

Example of NFA

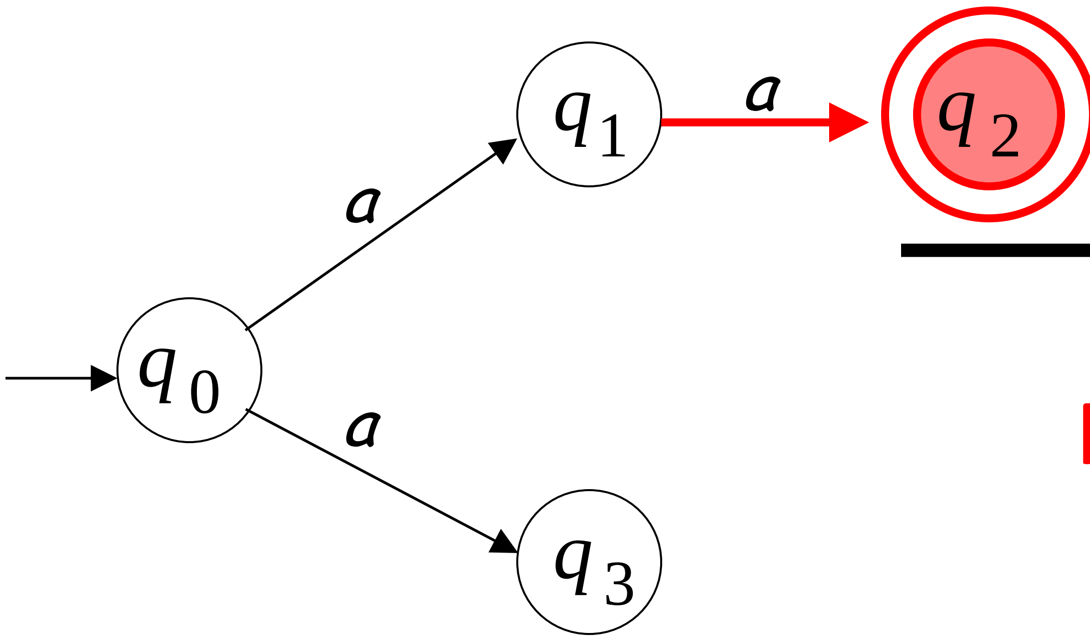


First Choice

Example of NFA



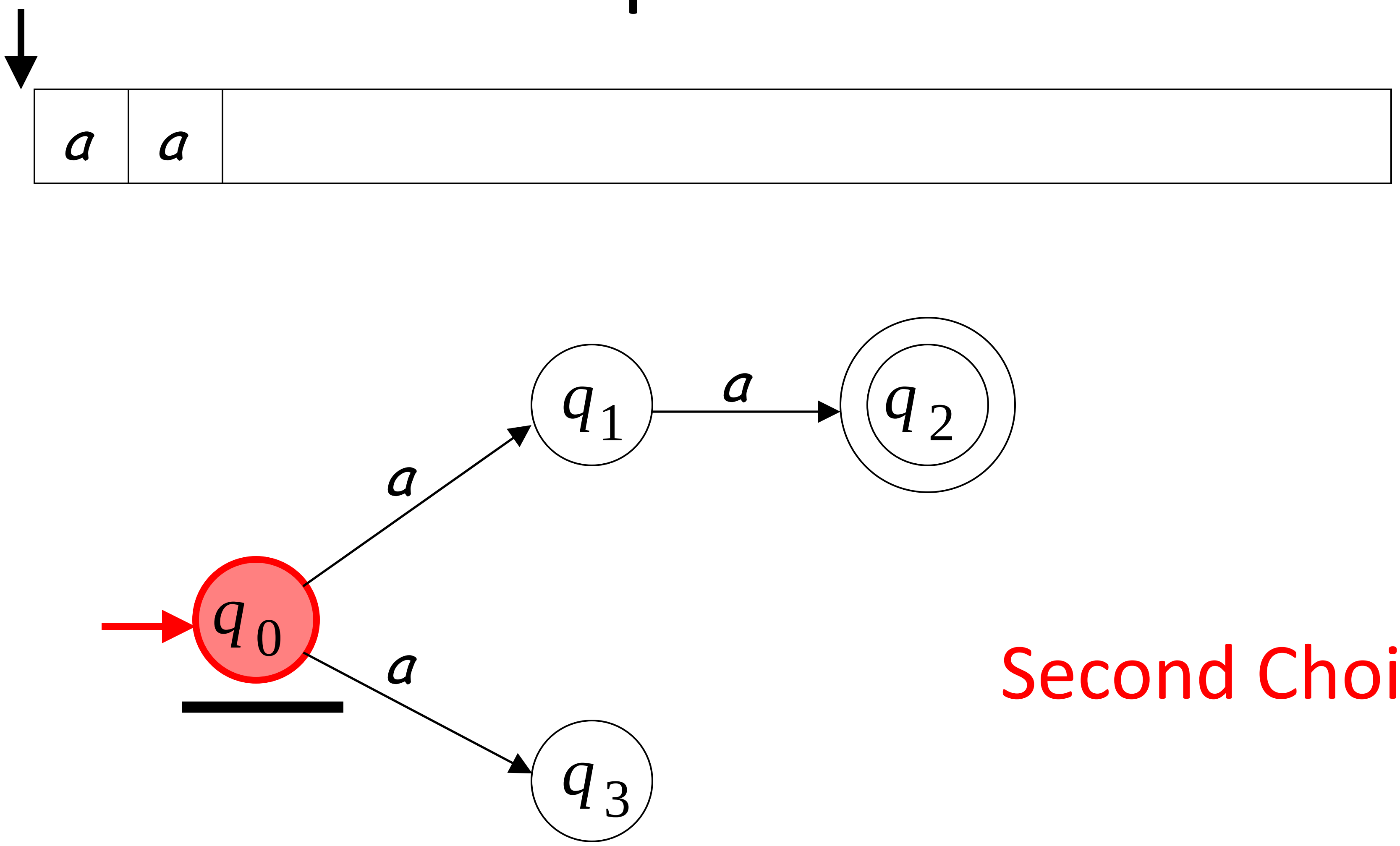
All input is consumed



"accept"

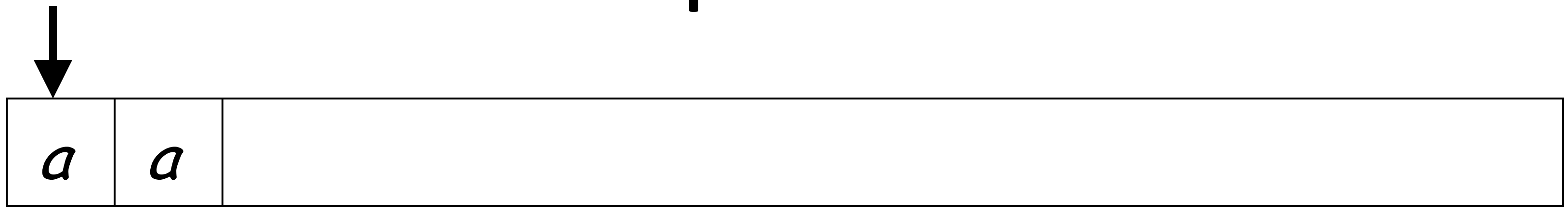
First Choice

Example of NFA

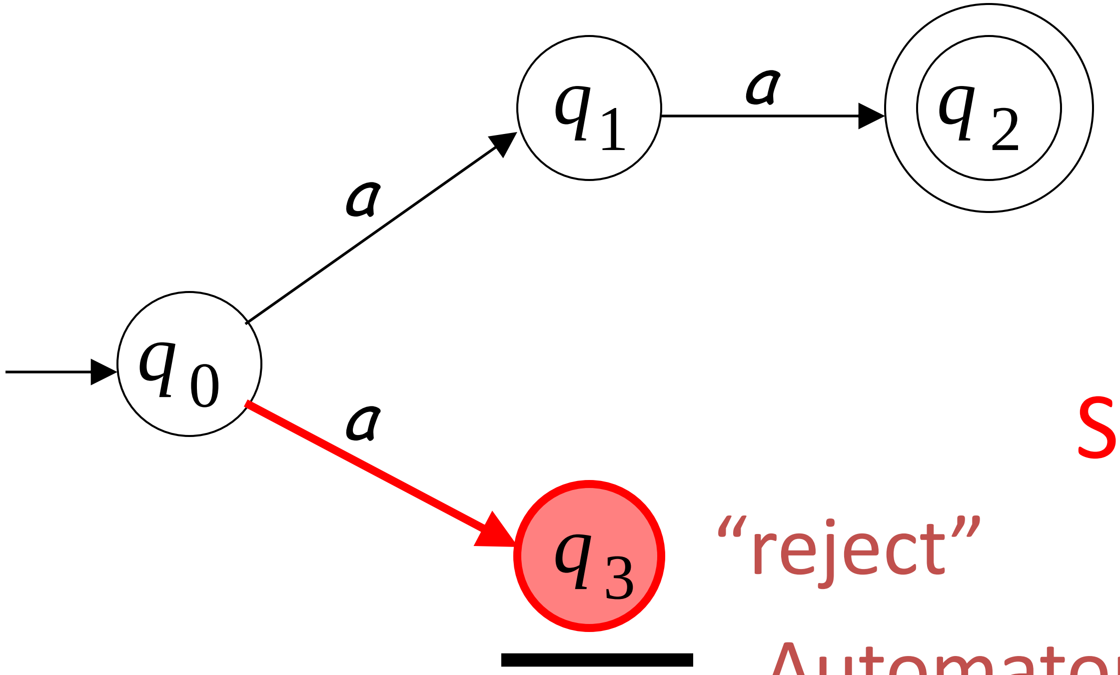


Second Choice

Example of NFA



Input cannot be consumed



Second Choice

"reject"

Automaton Halts

NFA acceptance

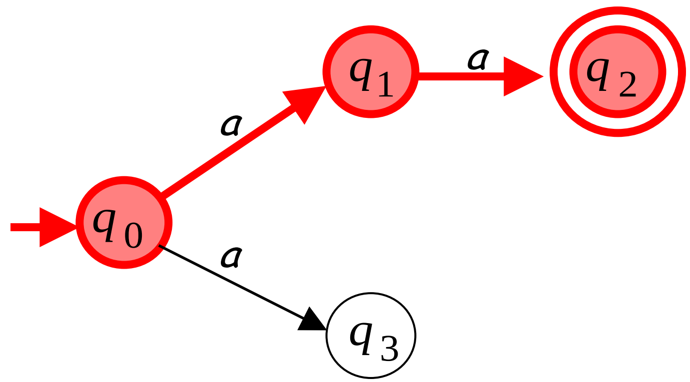
An NFA accepts a string
if there is a computation path of the NFA
that accepts the string

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

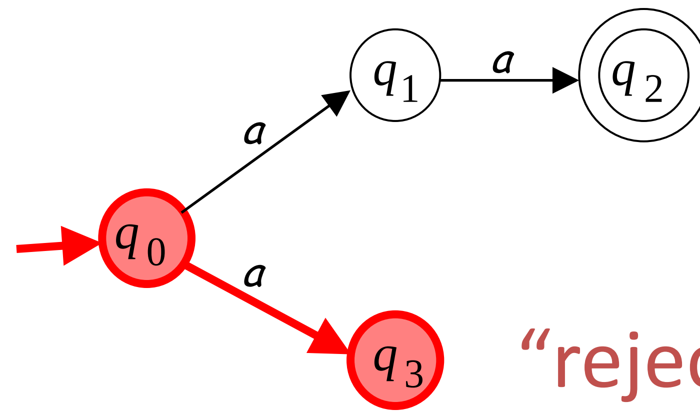
NFA acceptance

aa is accepted by the NFA:

"accept"



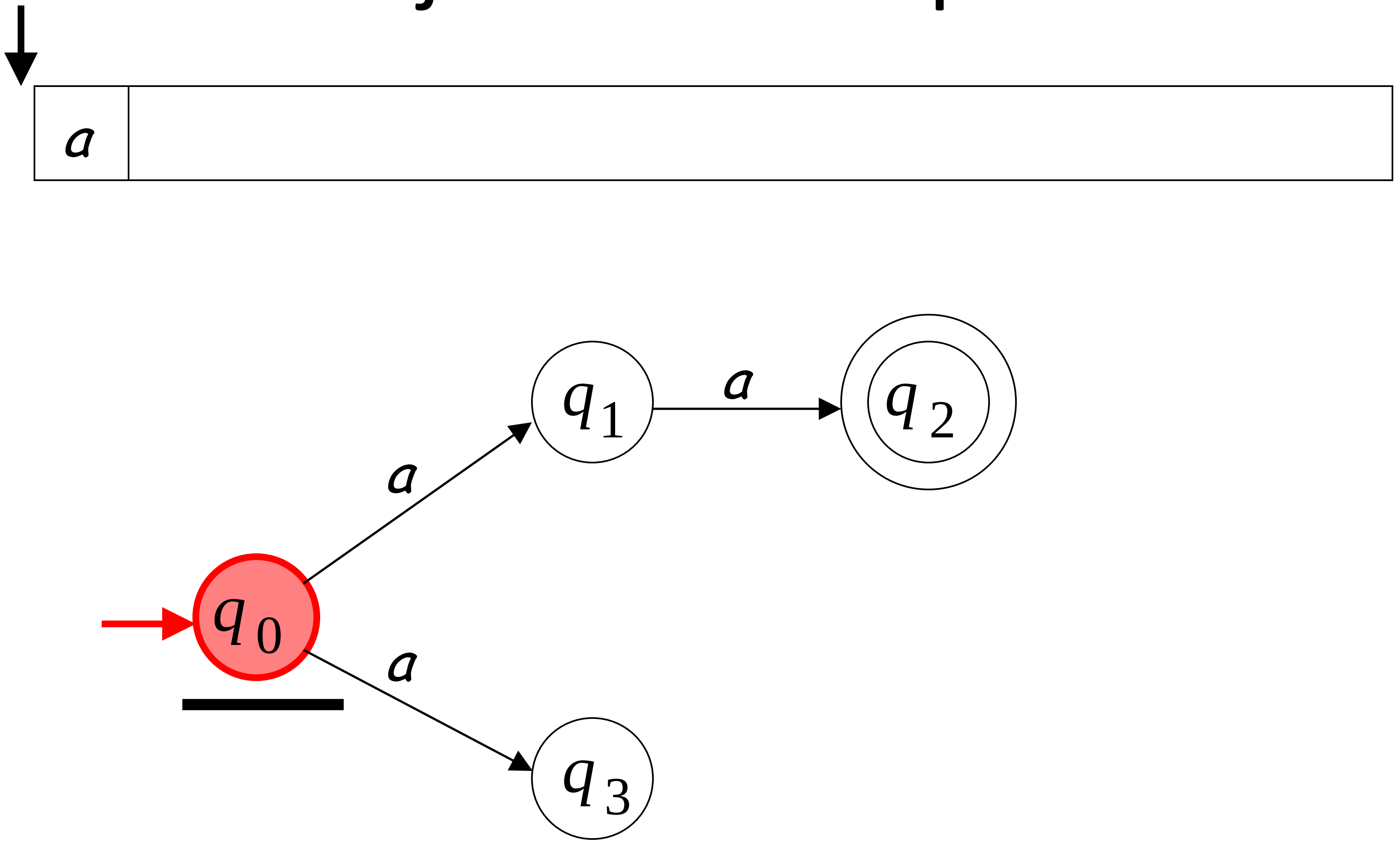
because this computation accepts aa



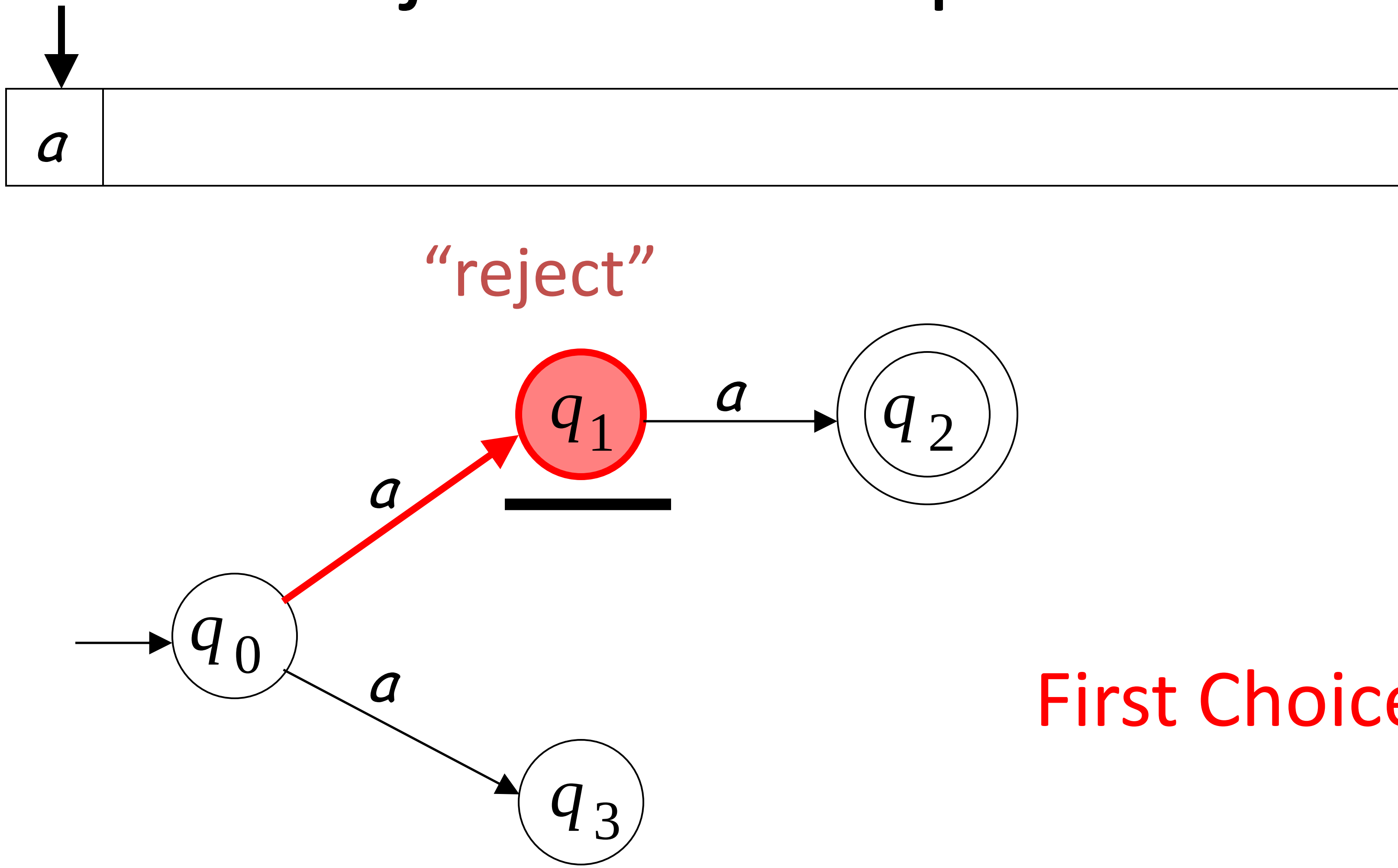
"reject"

this computation is ignored

Rejection example 1

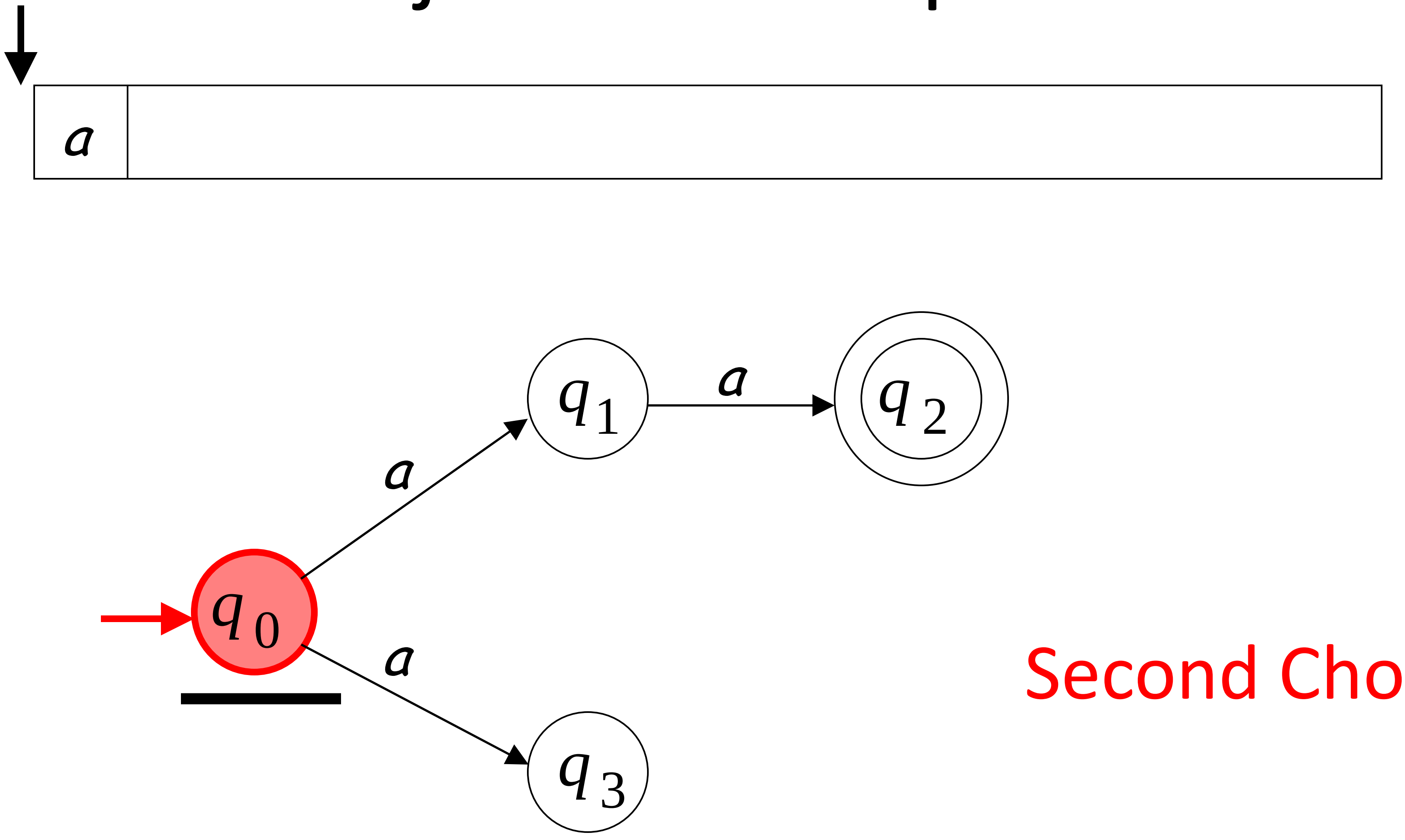


Rejection example 1



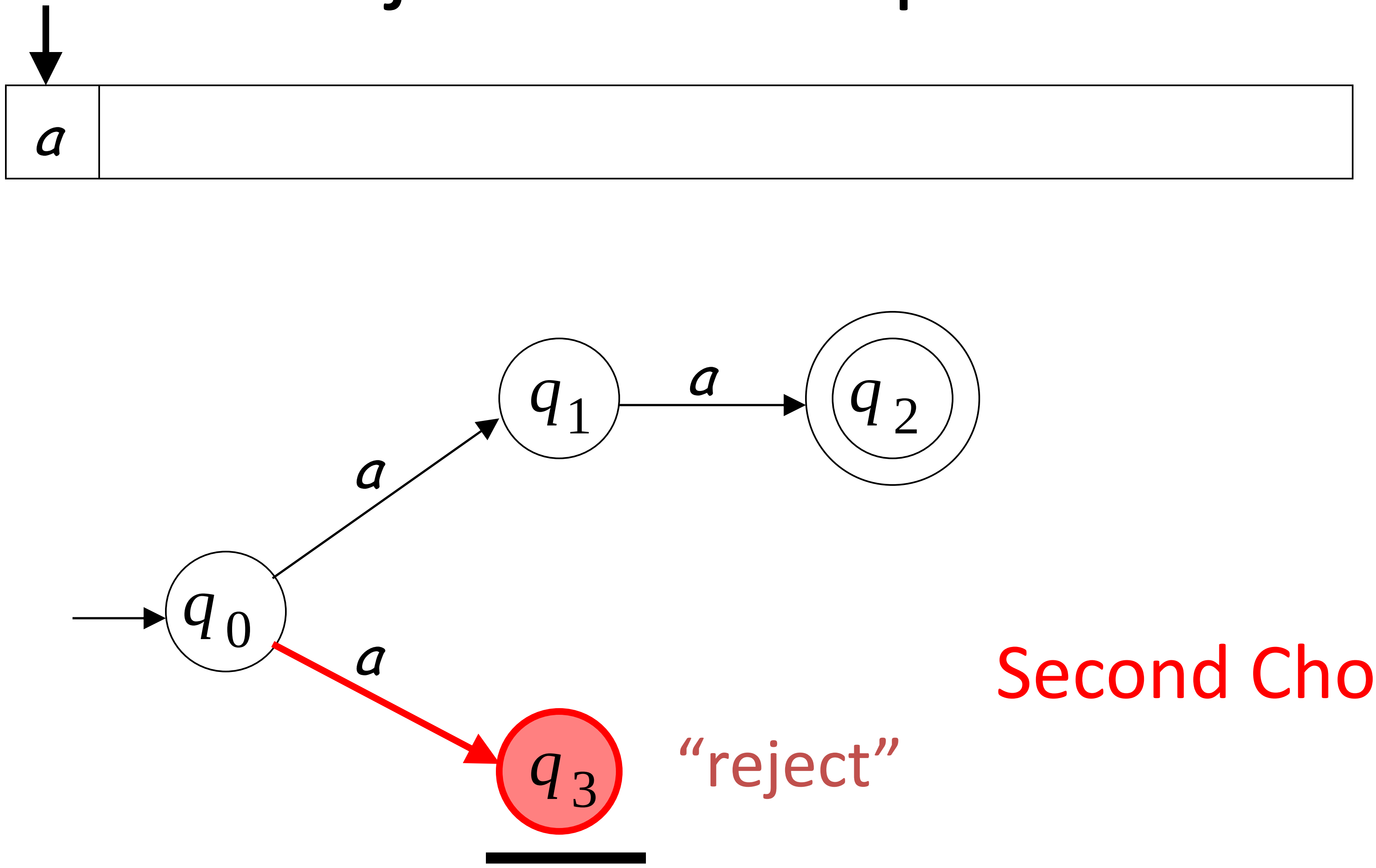
First Choice

Rejection example 1



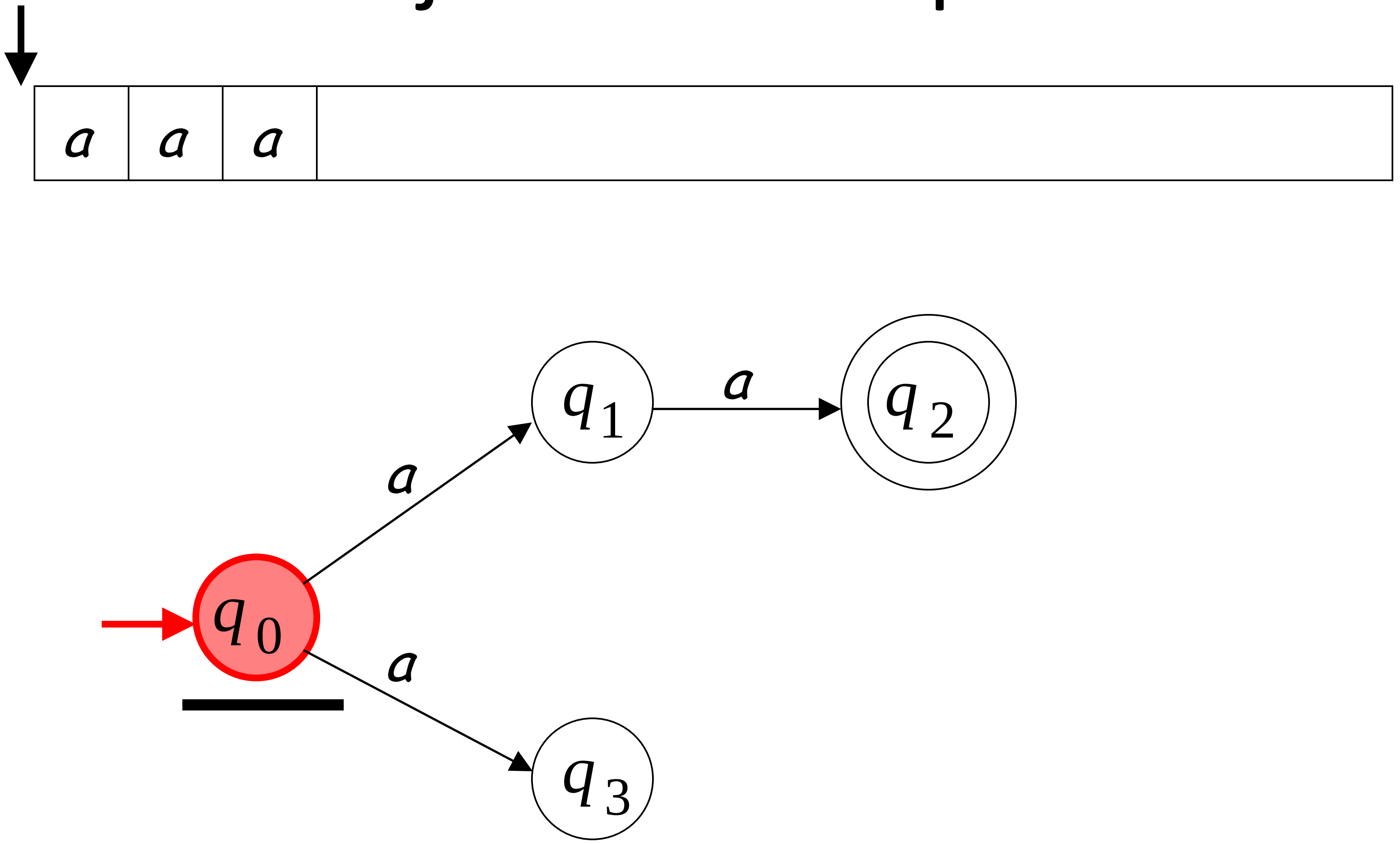
Second Choice

Rejection example 1

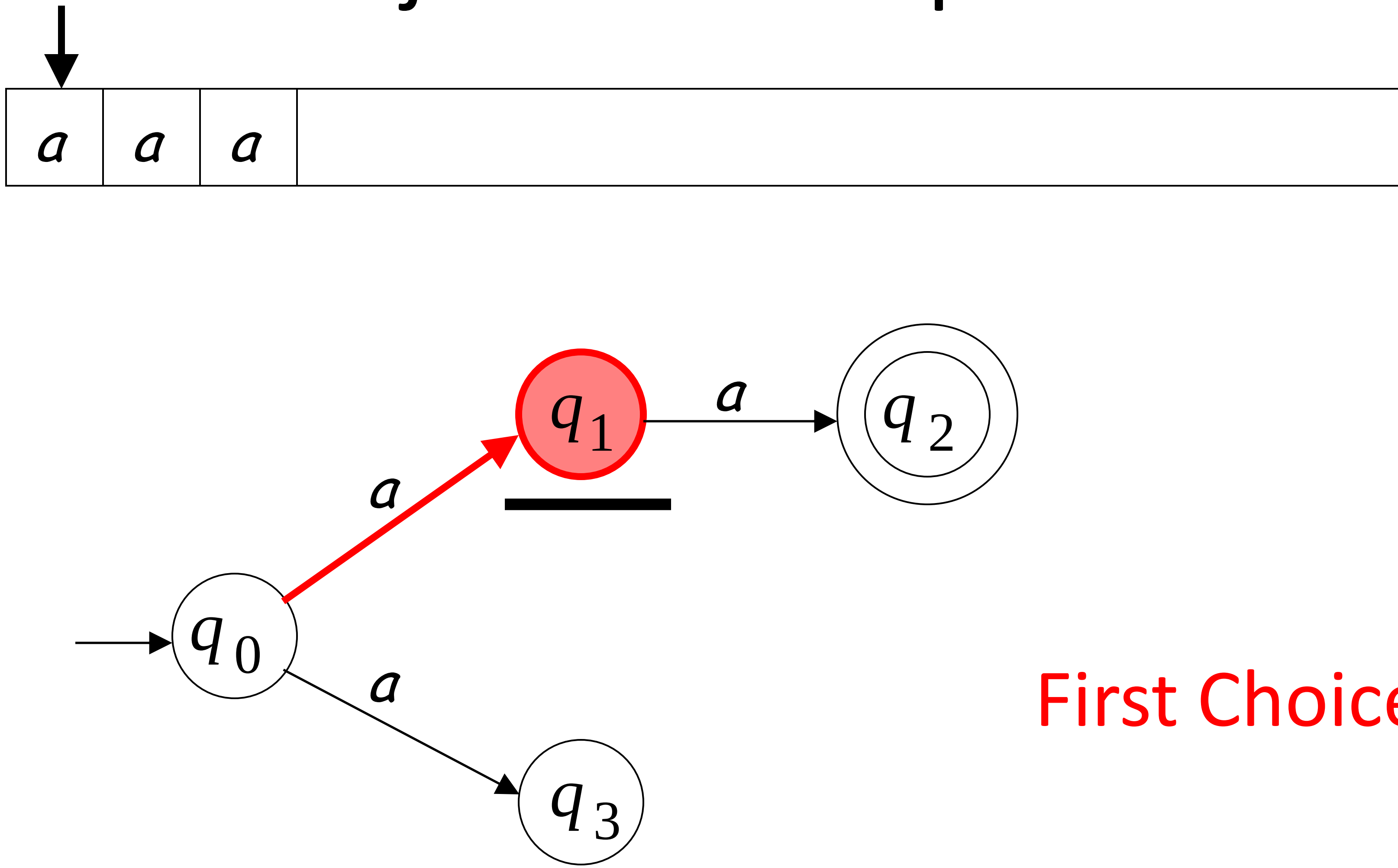


Second Choice

Rejection example 2

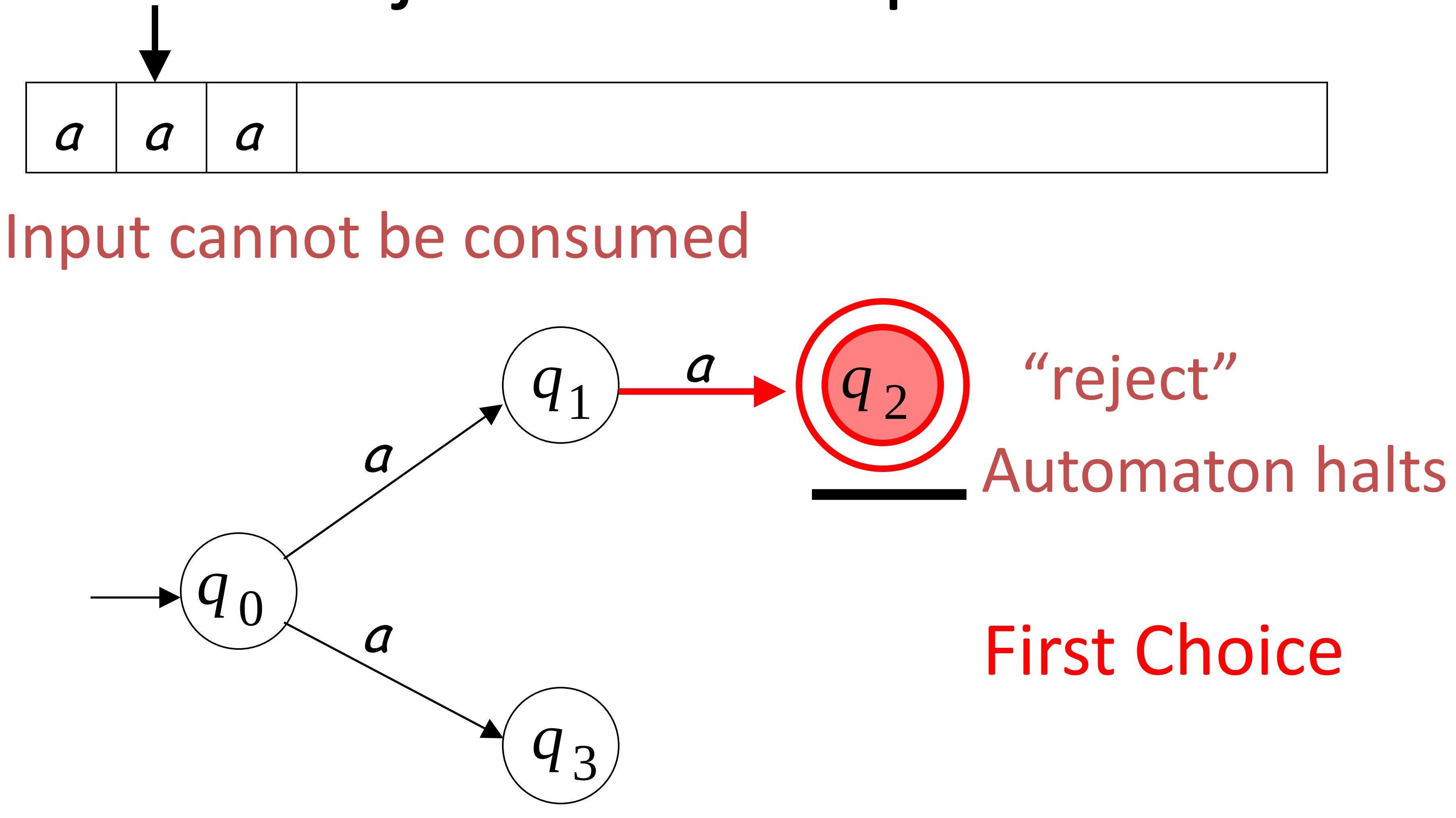


Rejection example 2



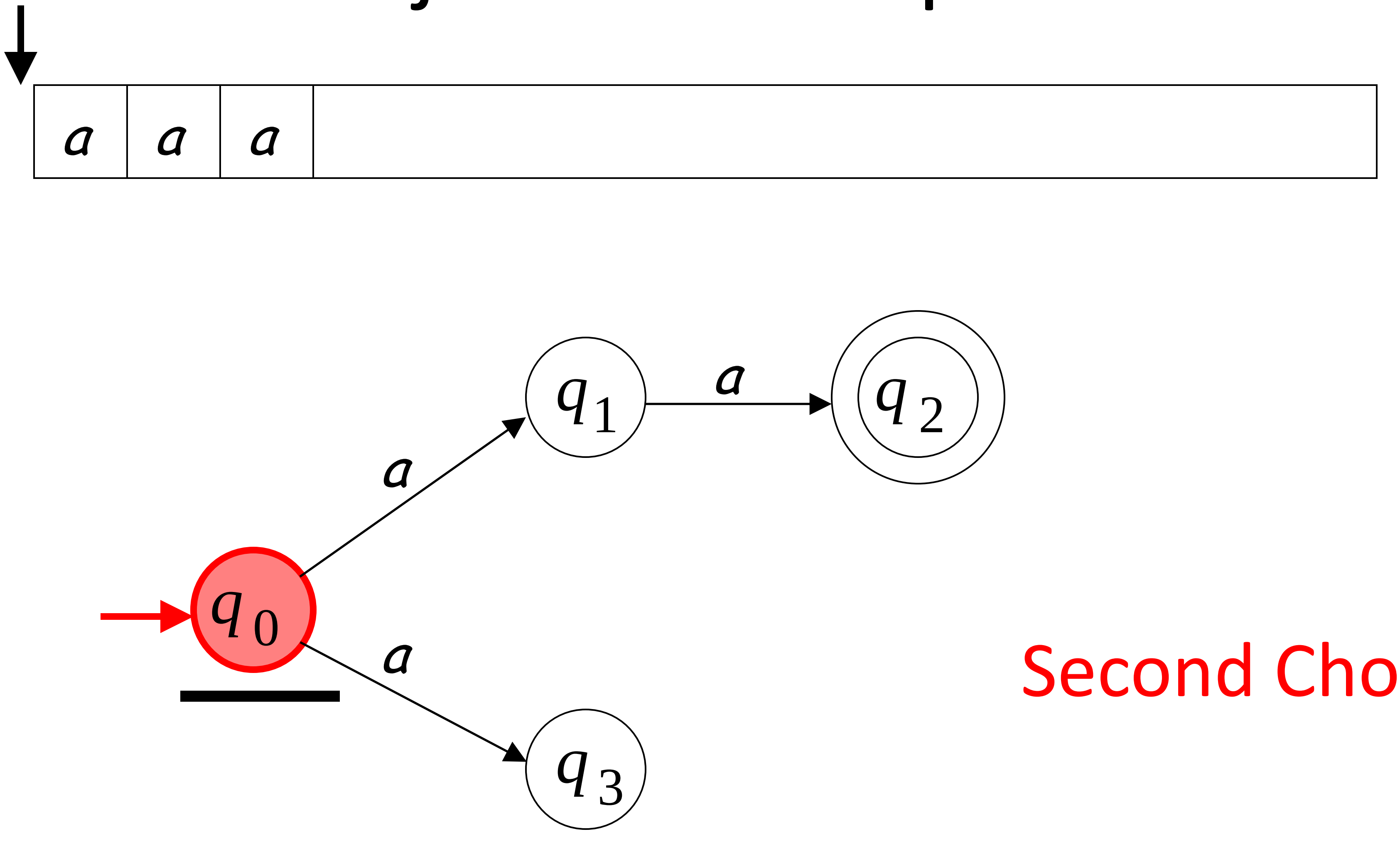
First Choice

Rejection example 2



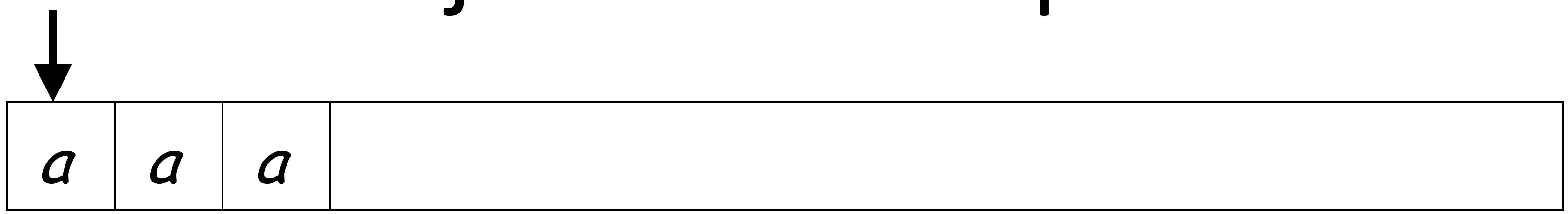
First Choice

Rejection example 2

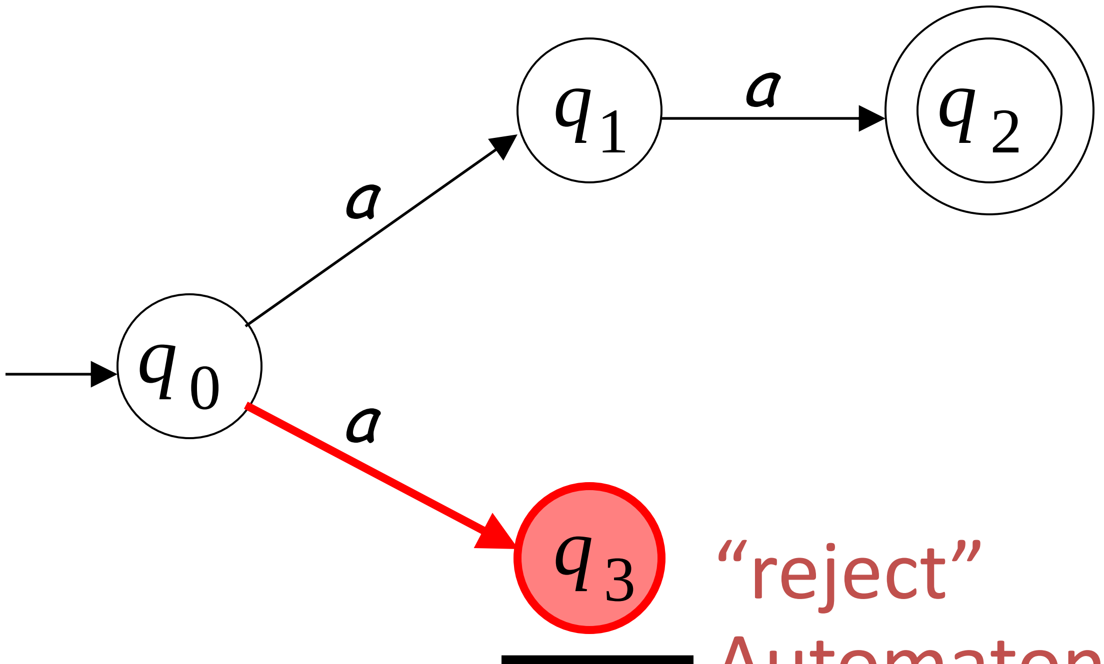


Second Choice

Rejection example 2



Input cannot be consumed



Second Choice

"reject"

Automaton halts

NFA rejection

An NFA rejects a string:

if there is no computation of the NFA

that accepts the string.

For **each** computation path:

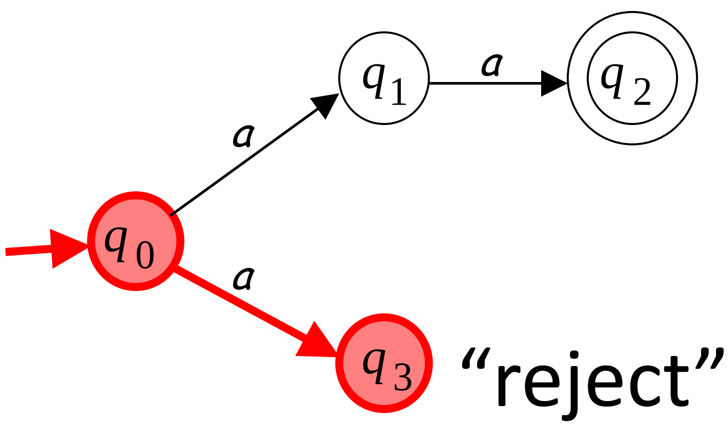
- All the input is consumed and the automaton is in a non accepting state

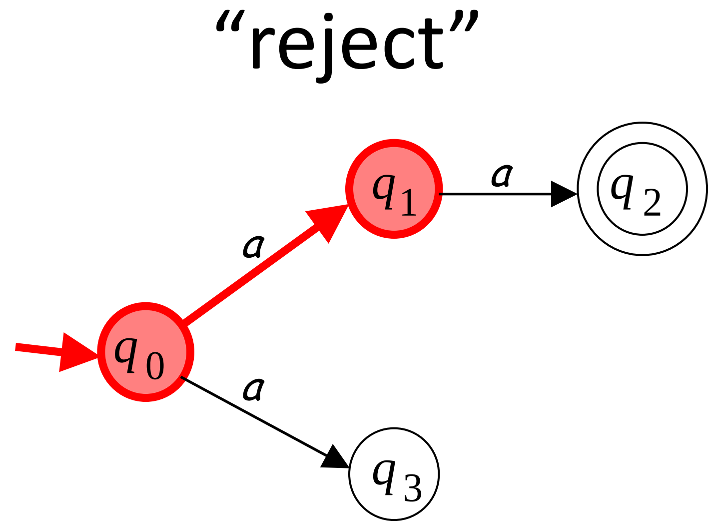
OR

- The input cannot be consumed

NFA rejection

a is rejected by the NFA:

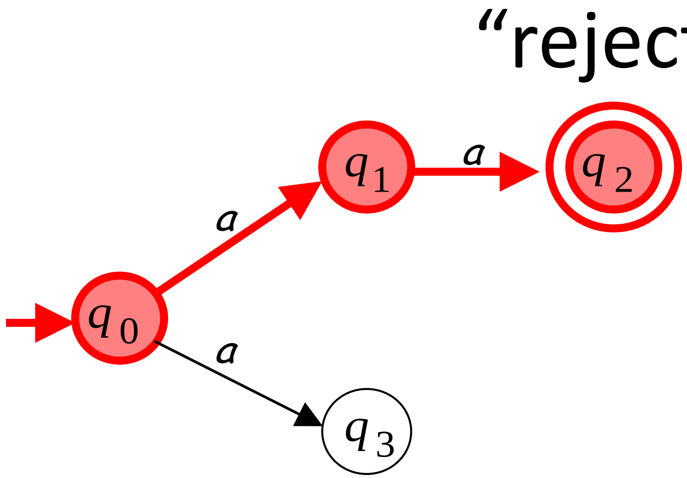


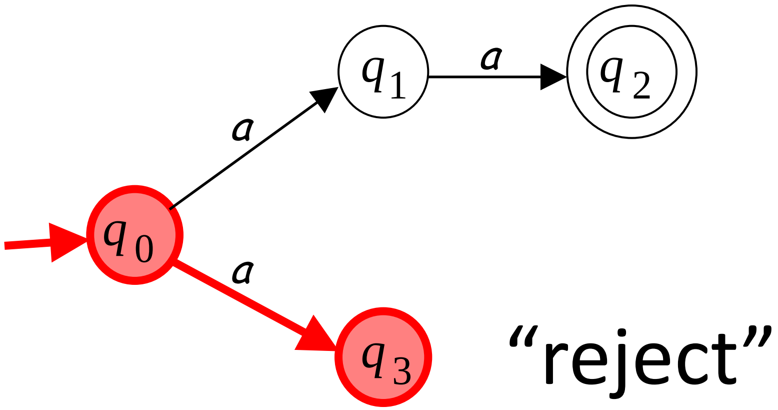


All possible computations lead to rejection

NFA rejection

aaa is rejected by the NFA:

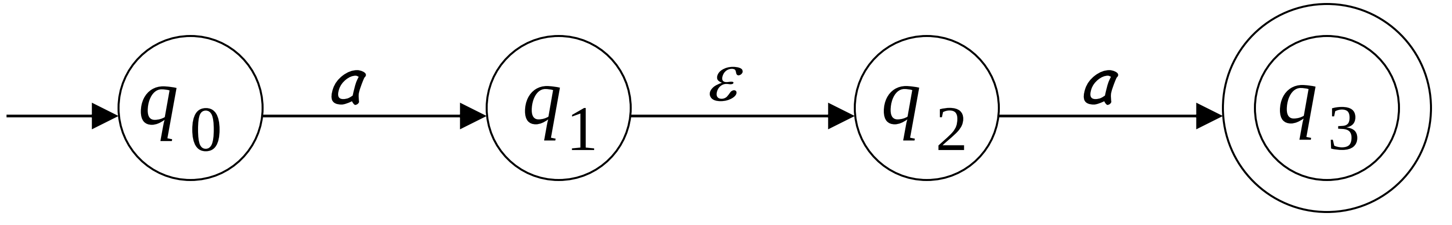




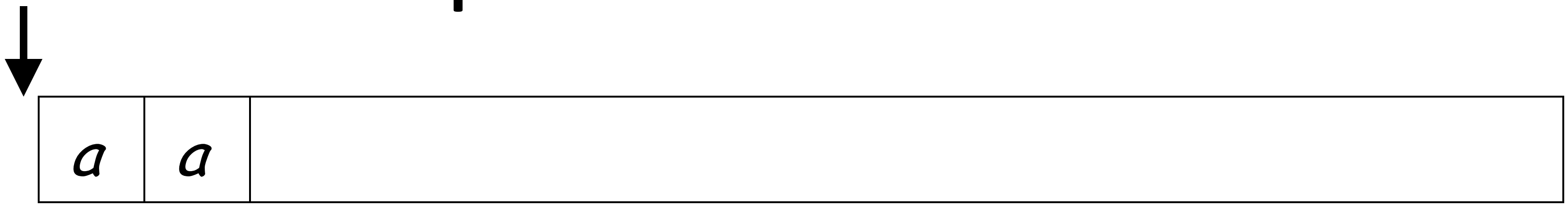
All possible computations lead to rejection

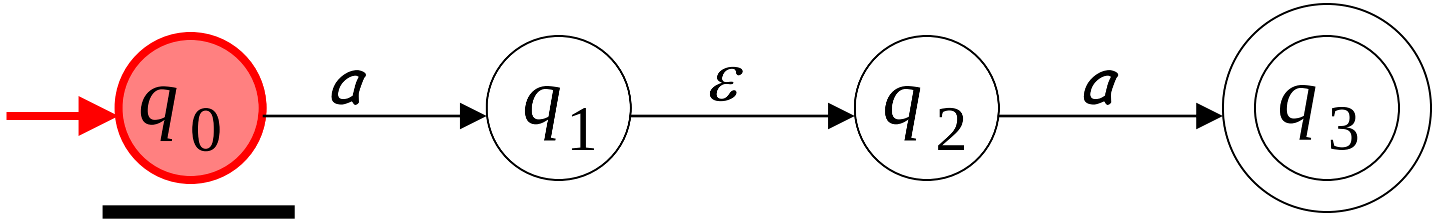
Epsilon Transitions

Spontaneous transition with **NO** input consumed

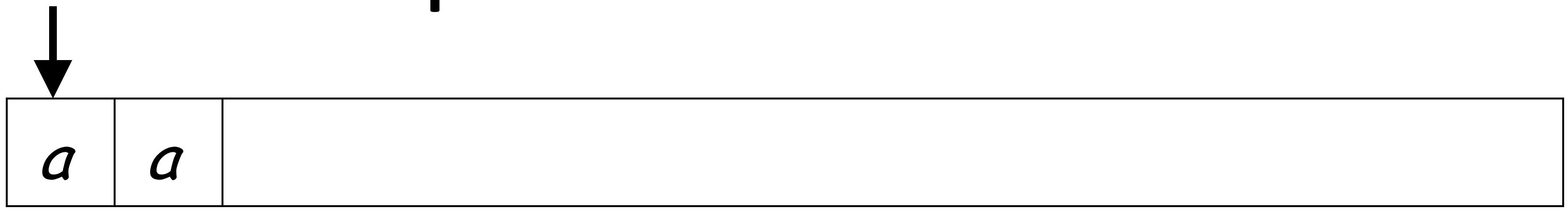


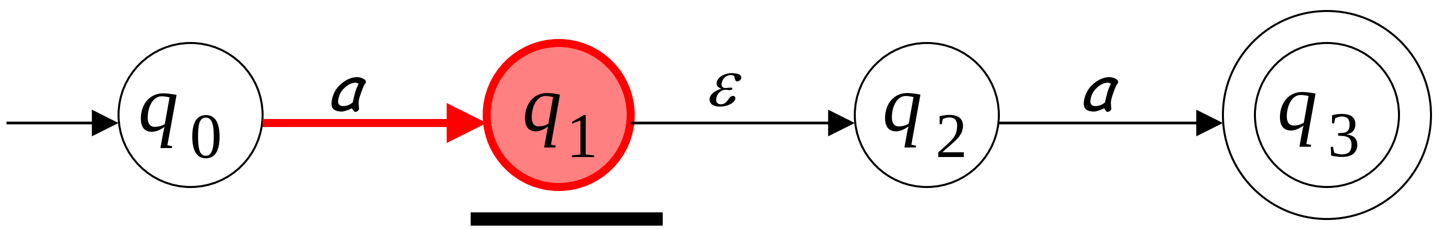
Epsilon Transitions



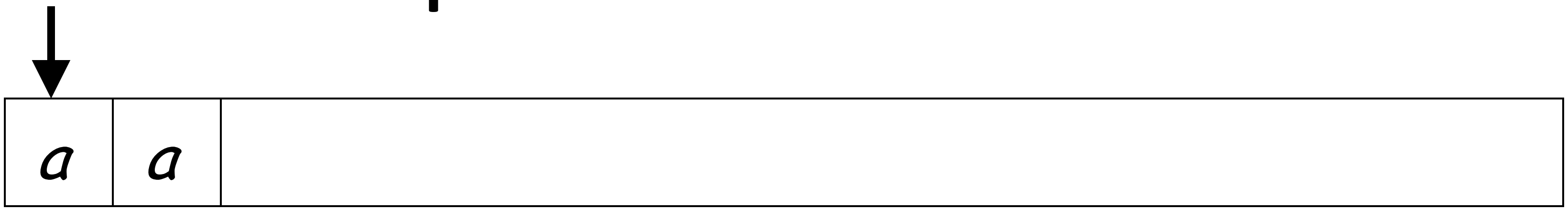


Epsilon Transitions

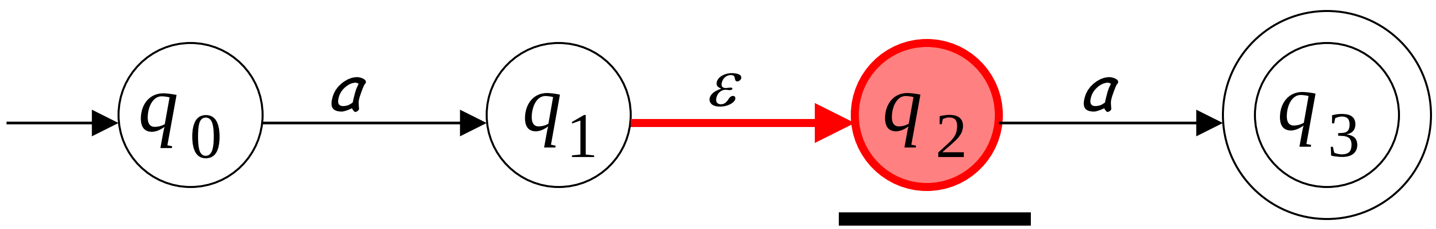




Epsilon Transitions

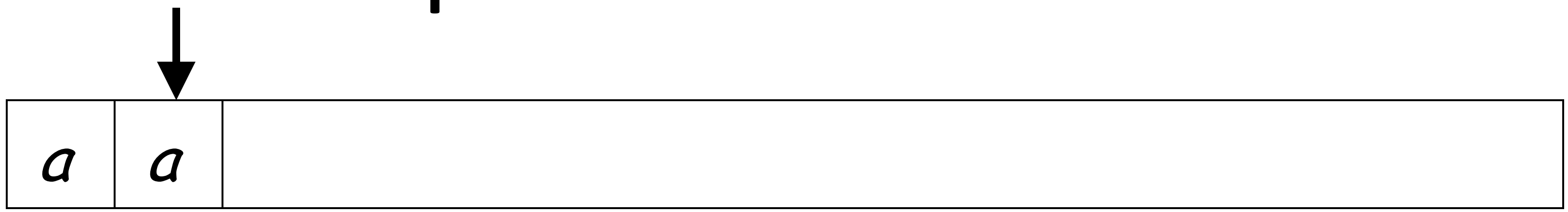


input tape head does not move



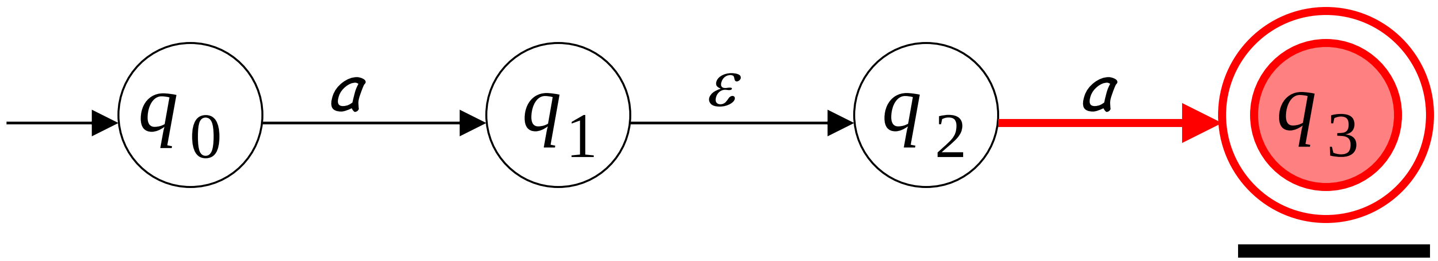
Automaton changes state

Epsilon Transitions



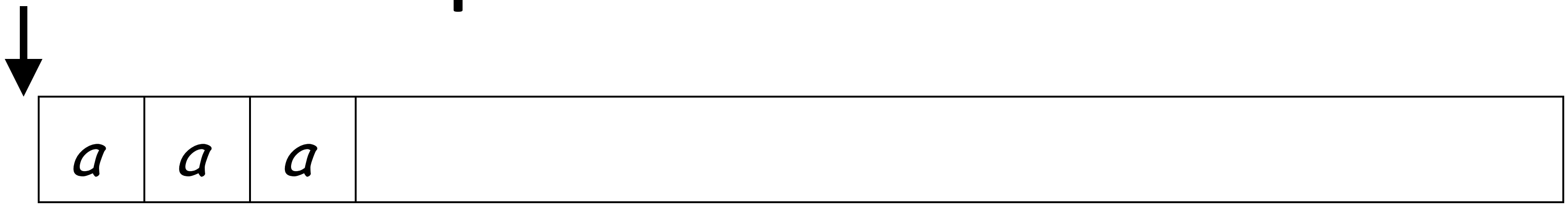
all input is consumed

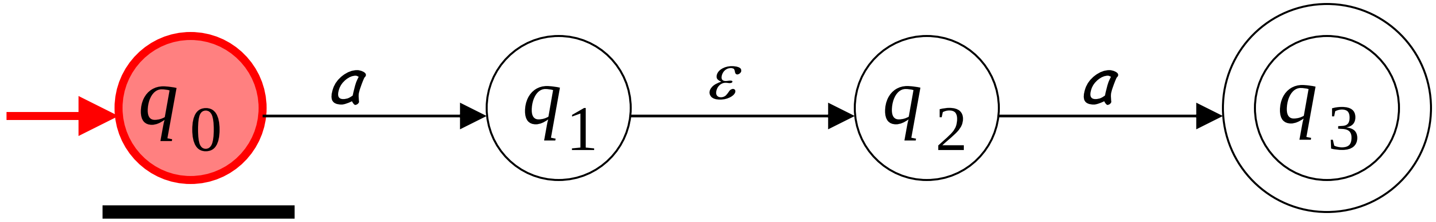
"accept"



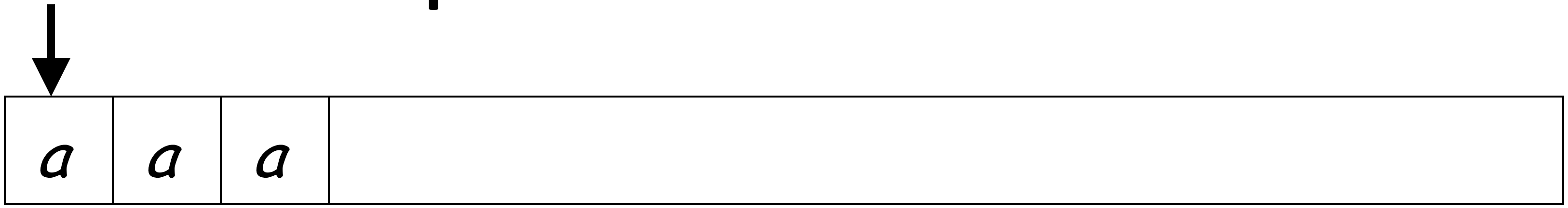
String aa is accepted

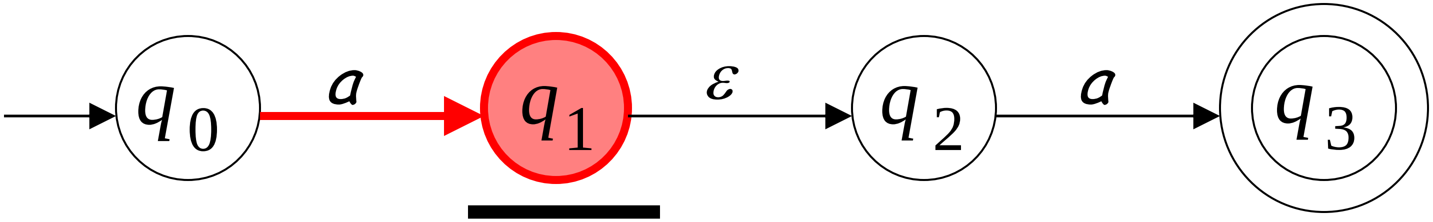
Epsilon Transitions



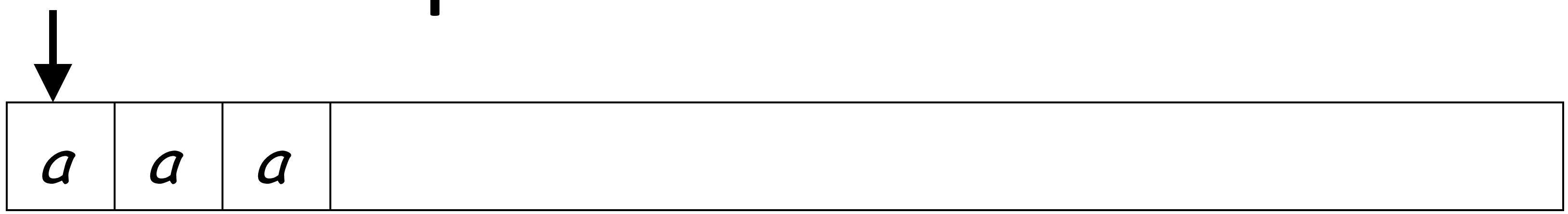


Epsilon Transitions

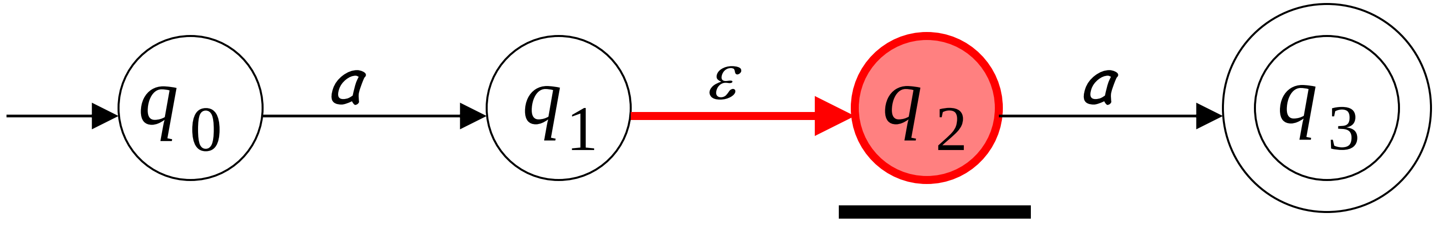




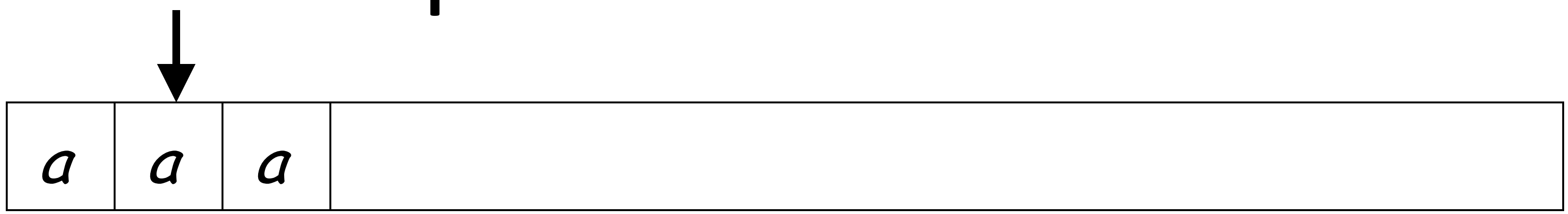
Epsilon Transitions



(read head doesn't move)

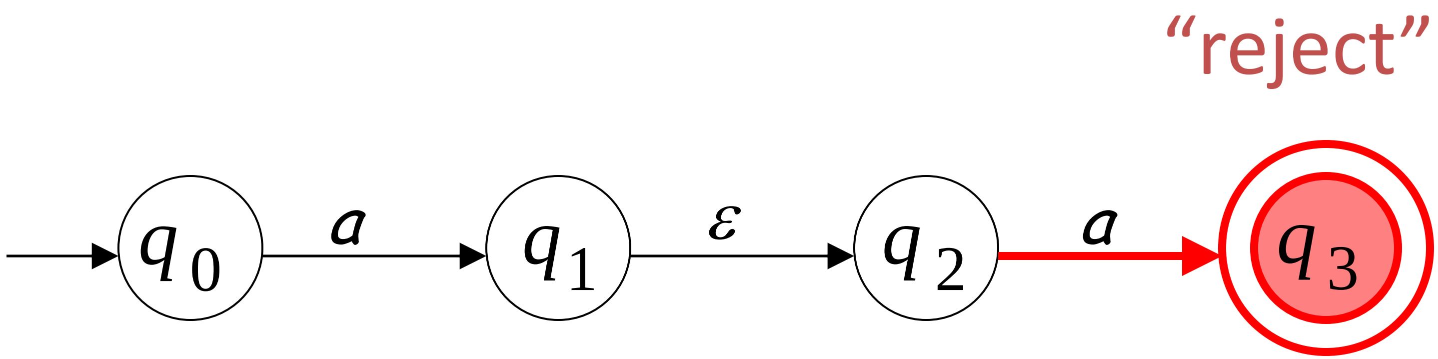


Epsilon Transitions



Input cannot be consumed

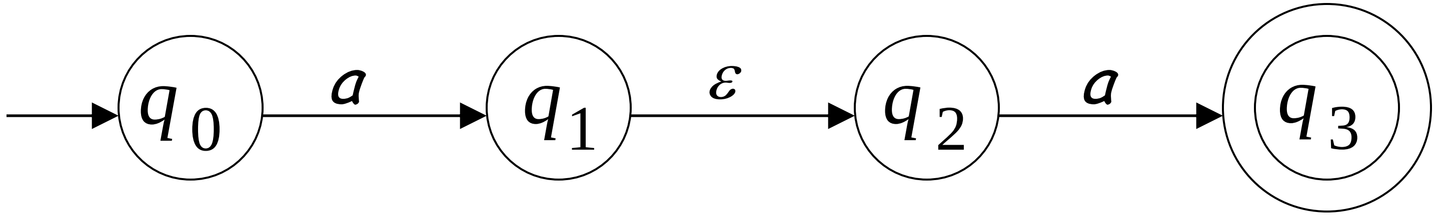
Automaton halts



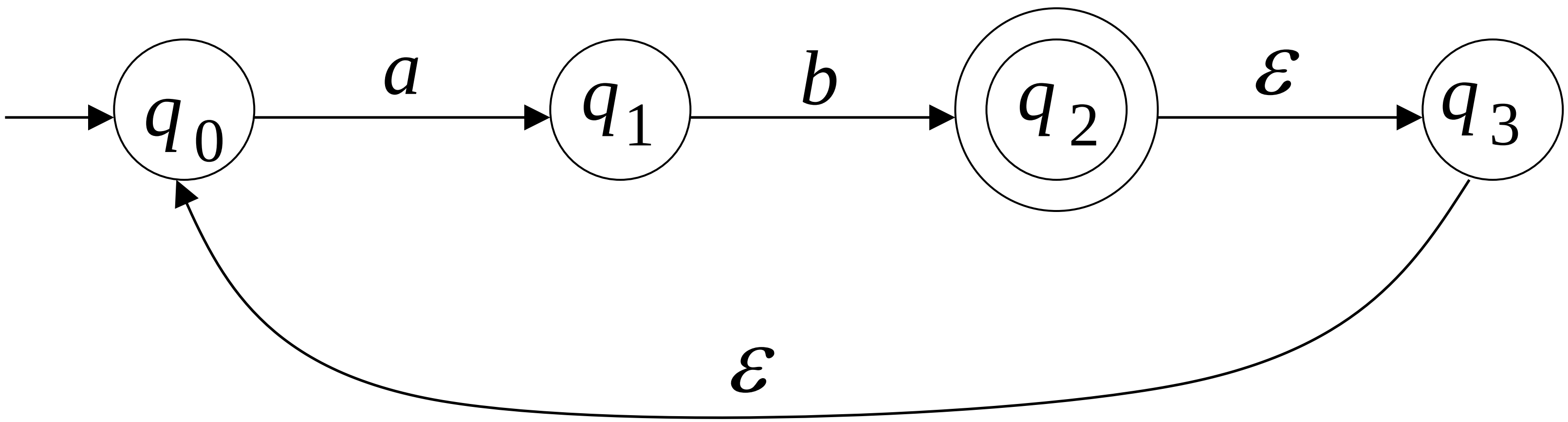
String aaa is rejected

Epsilon Transitions

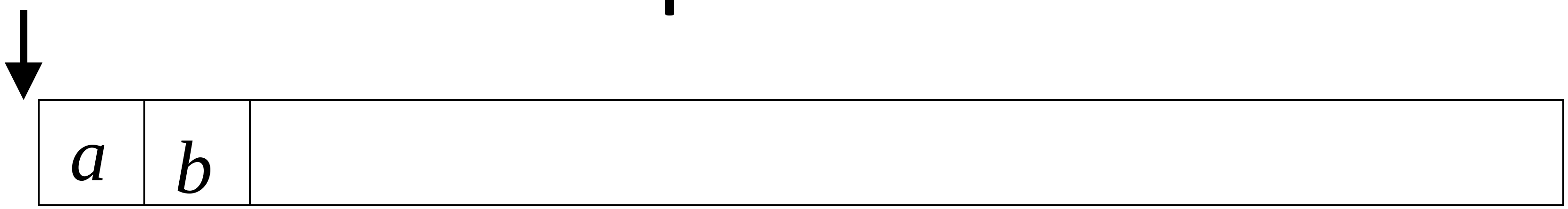
Language accepted: $L = \{aa\}$

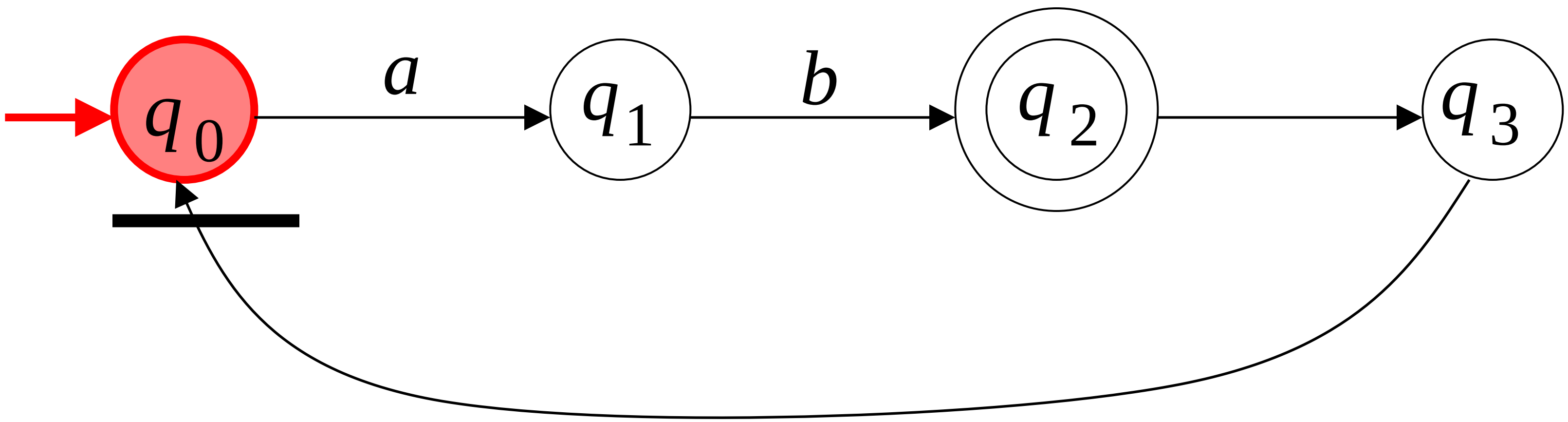


Example of ε-NFA

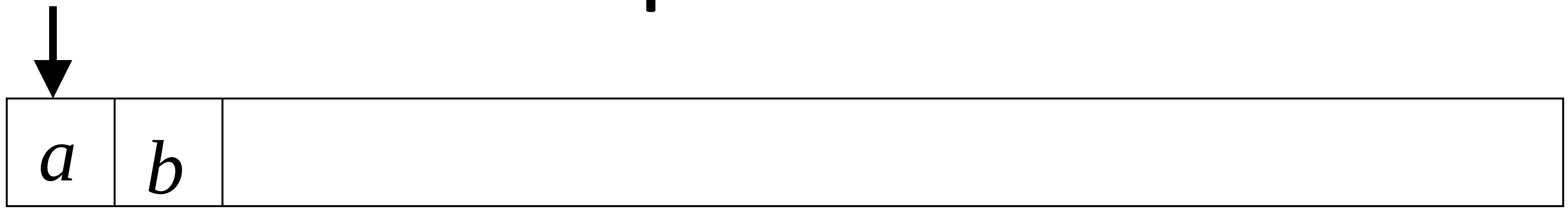


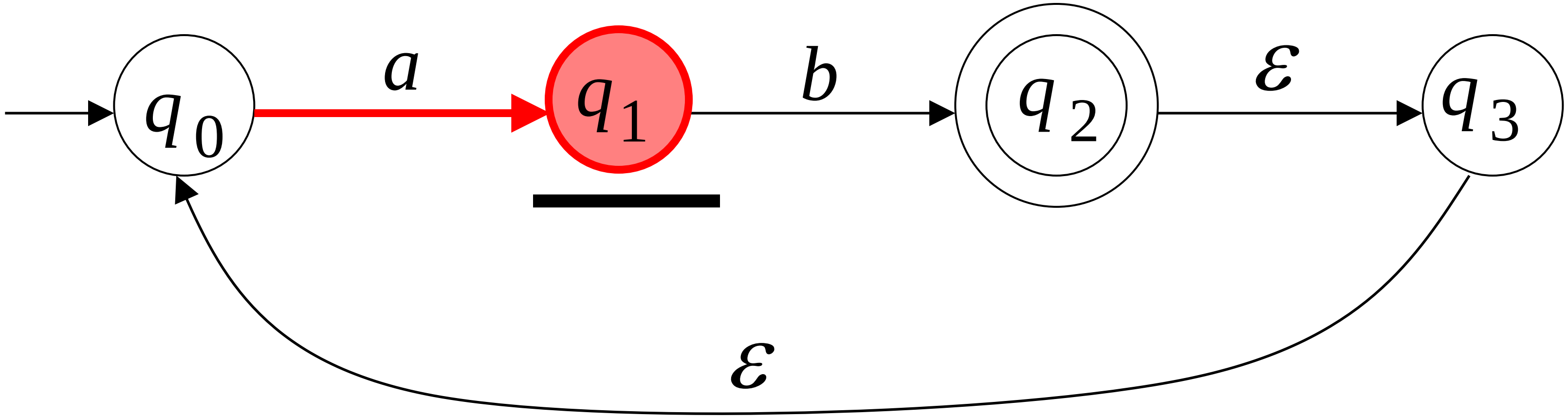
Example of ε-NFA



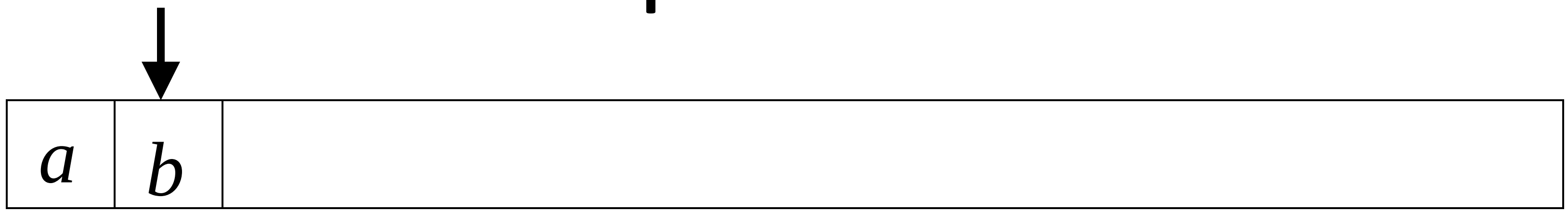


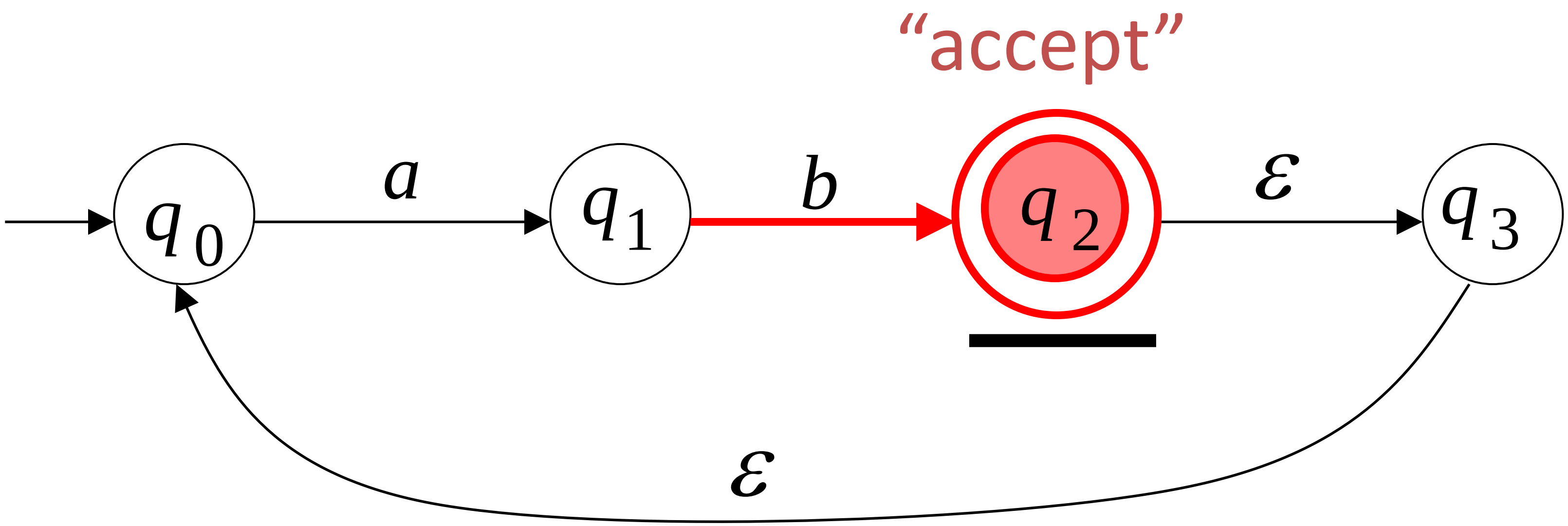
Example of ε-NFA



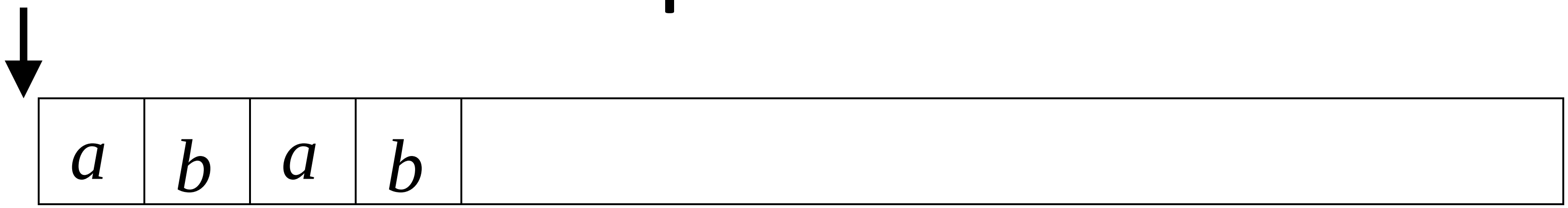


Example of ε-NFA

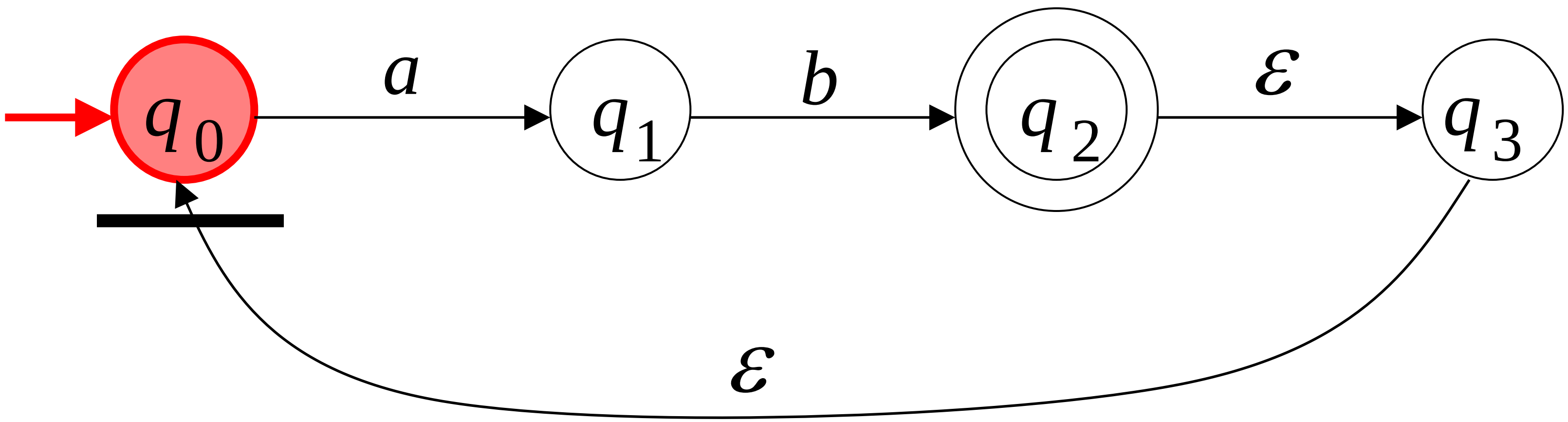




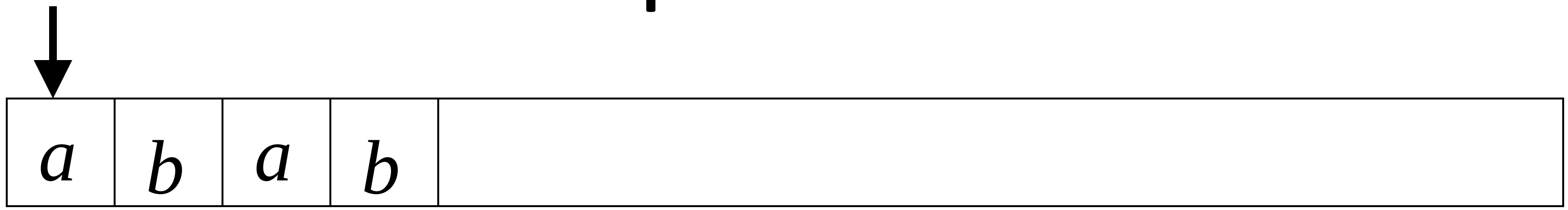
Example of ε-NFA

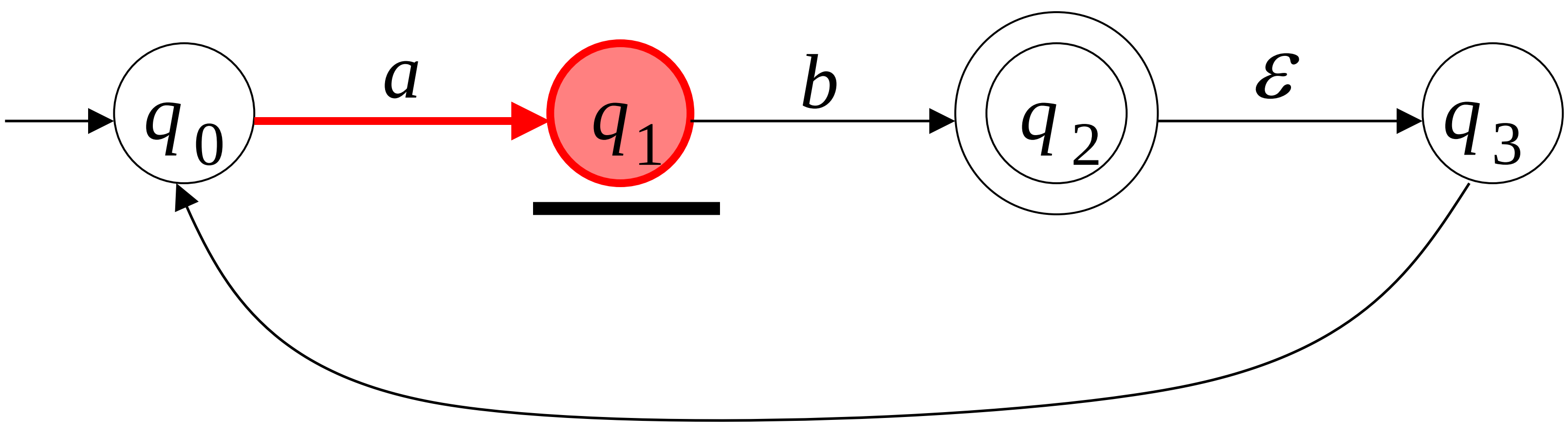


Another String

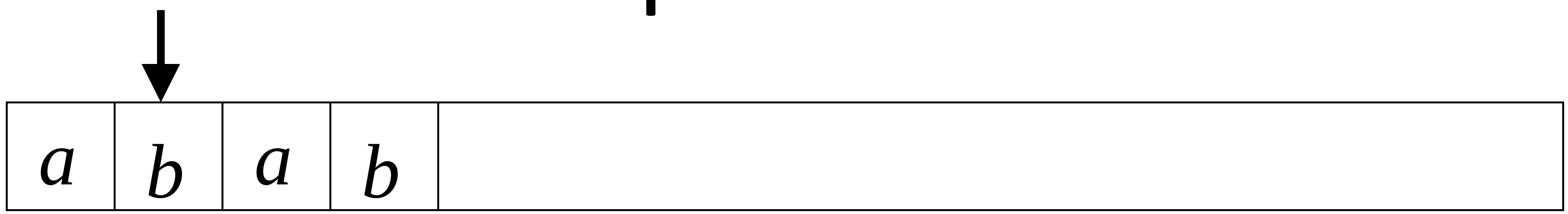


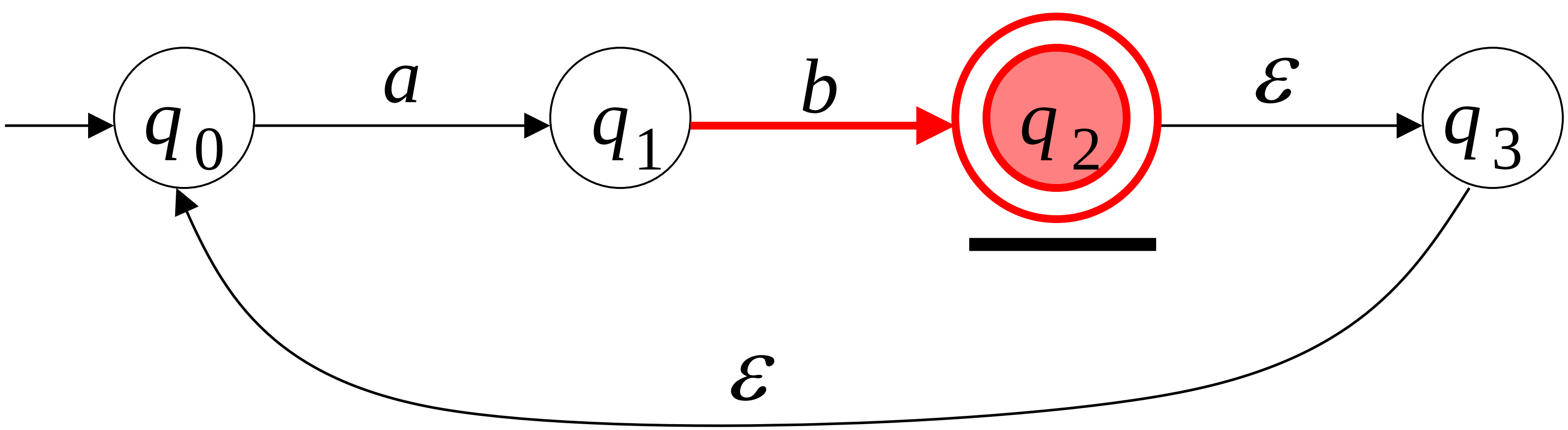
Example of ε-NFA



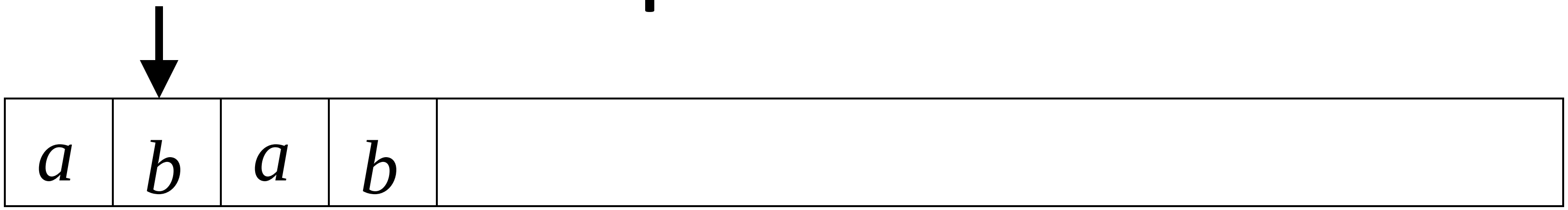


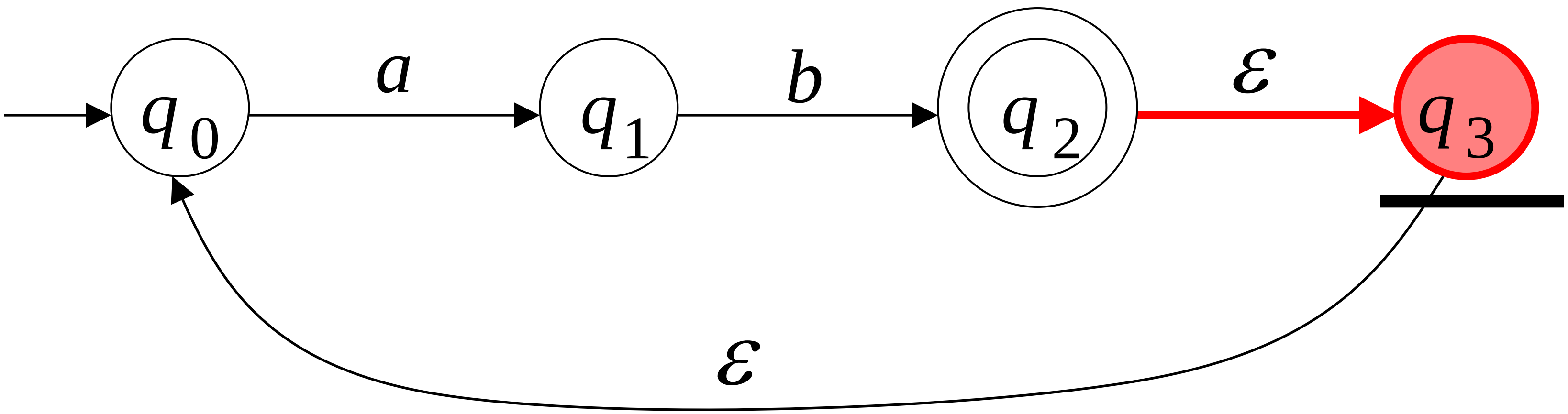
Example of ε-NFA



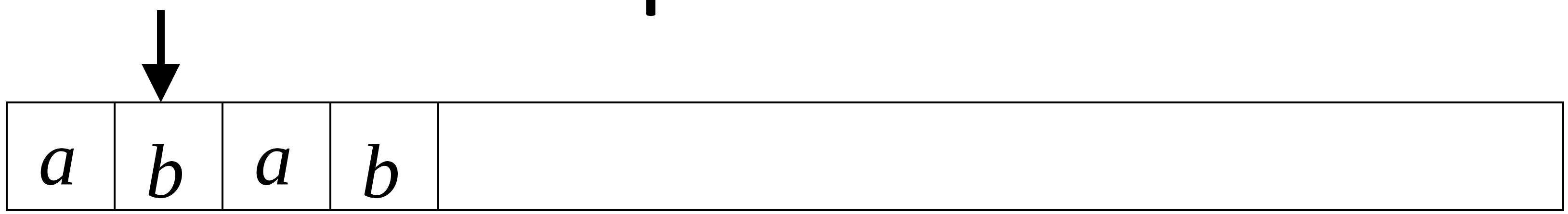


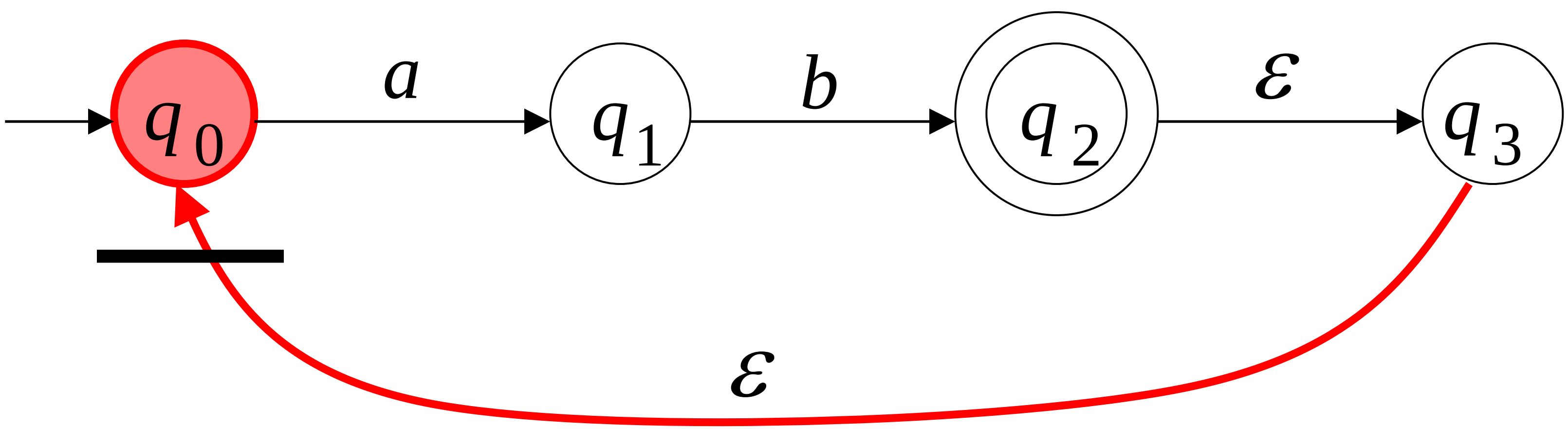
Example of ε-NFA



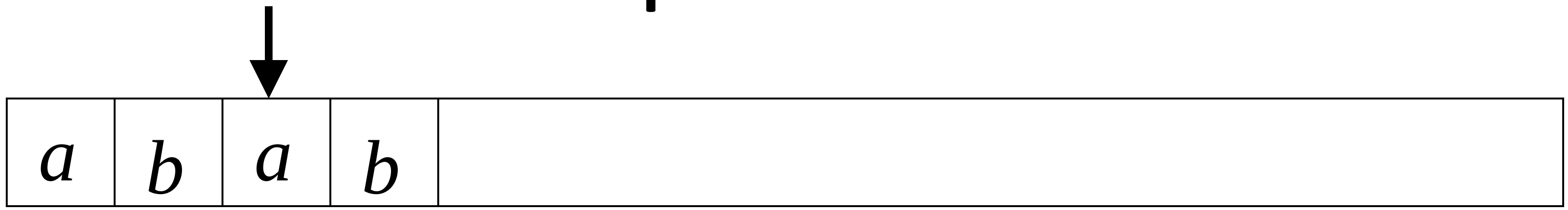


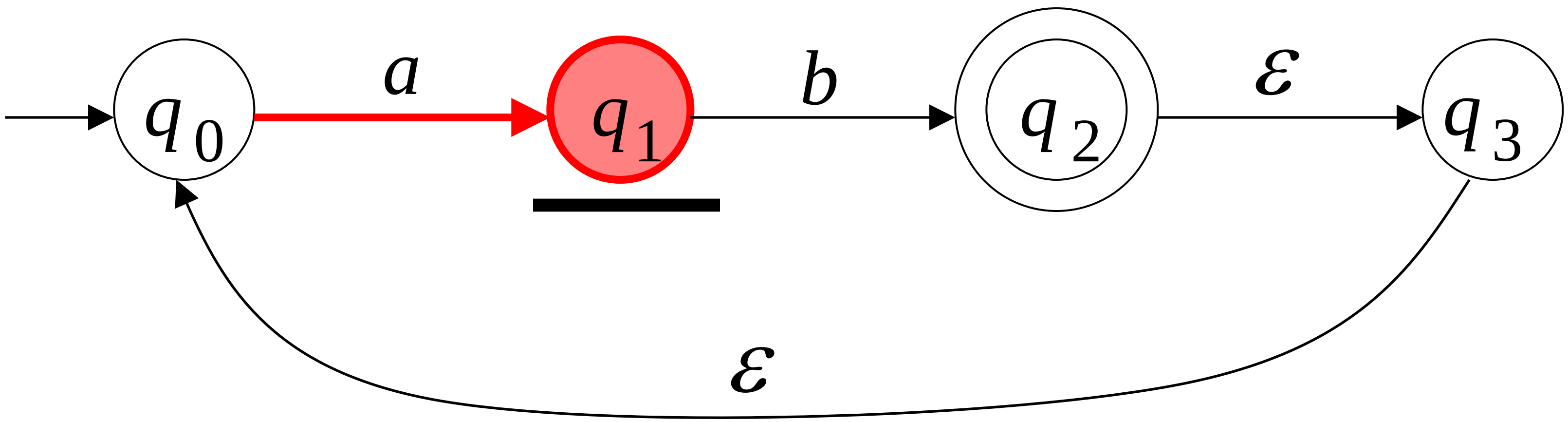
Example of ε-NFA





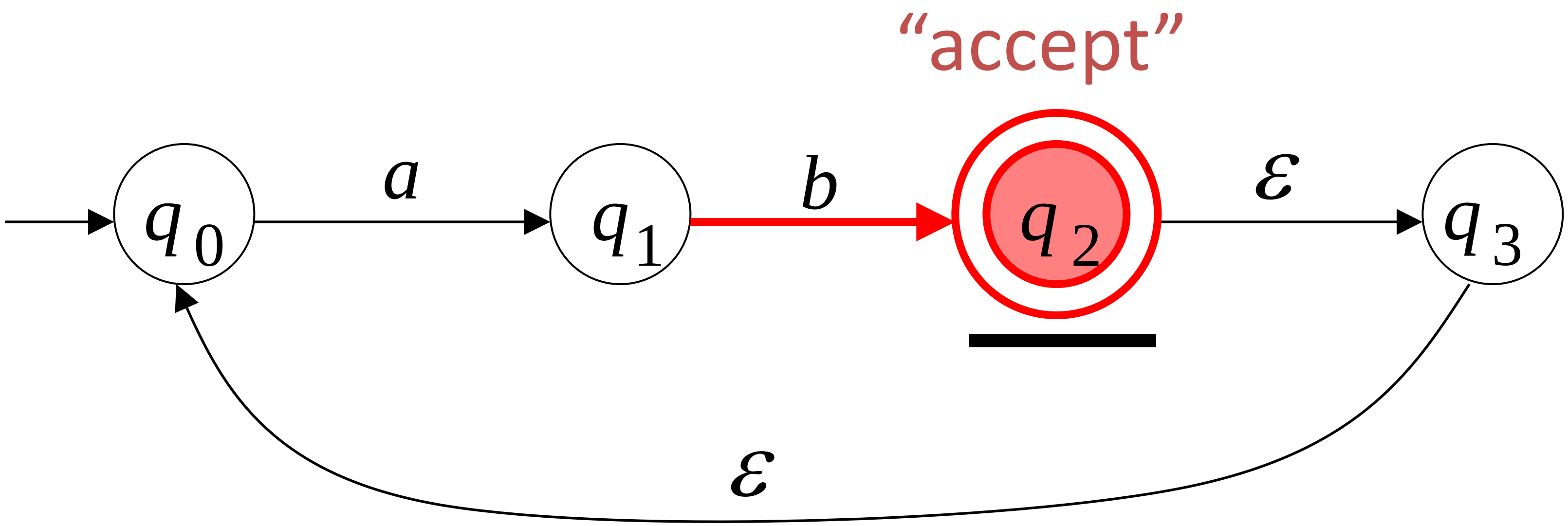
Example of ε-NFA





Example of ε-NFA

a	b	a	b	

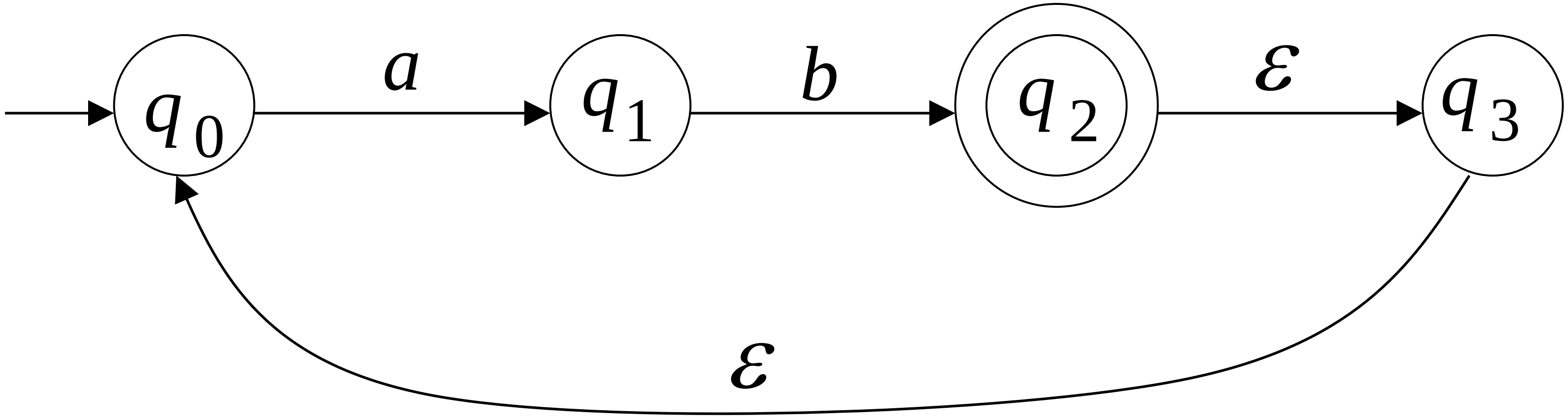


Example of ε-NFA

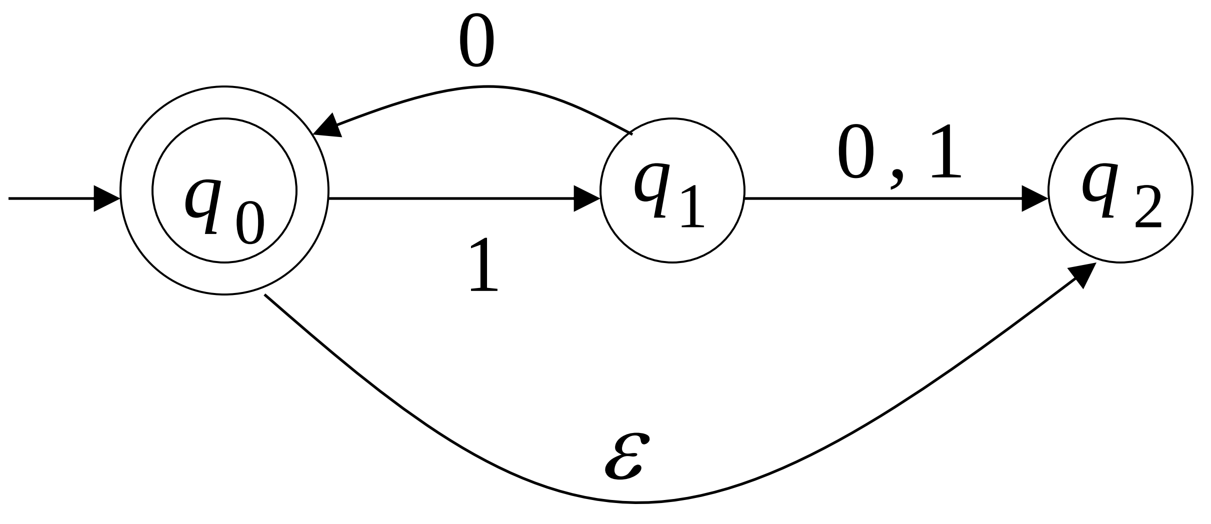
Language accepted

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

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



Another ε-NFA Example

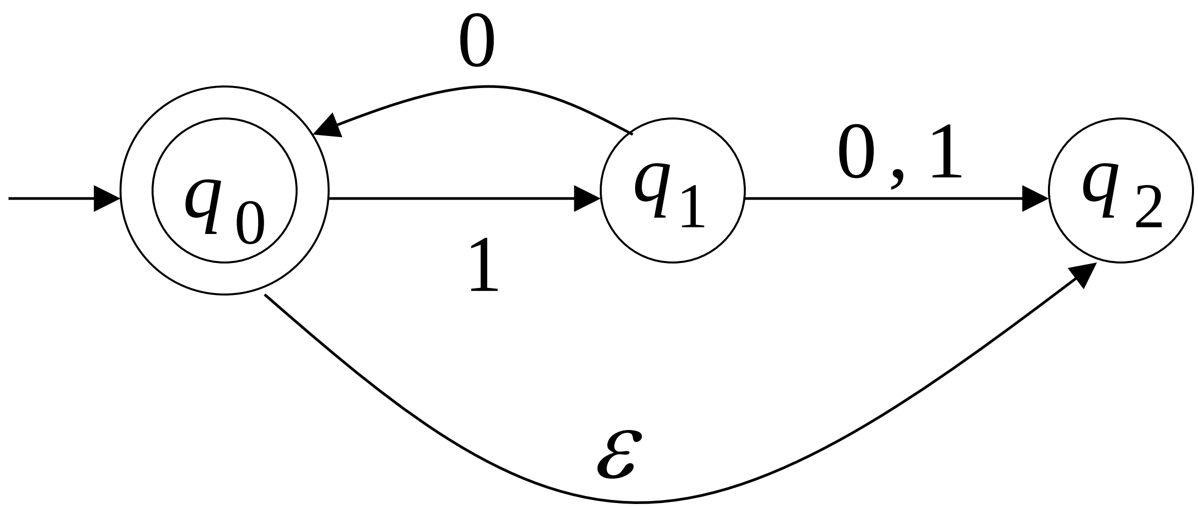


Another ε-NFA Example

Language accepted

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

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



(redundant state)

The Language of an NFA

The language accepted by M is:

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

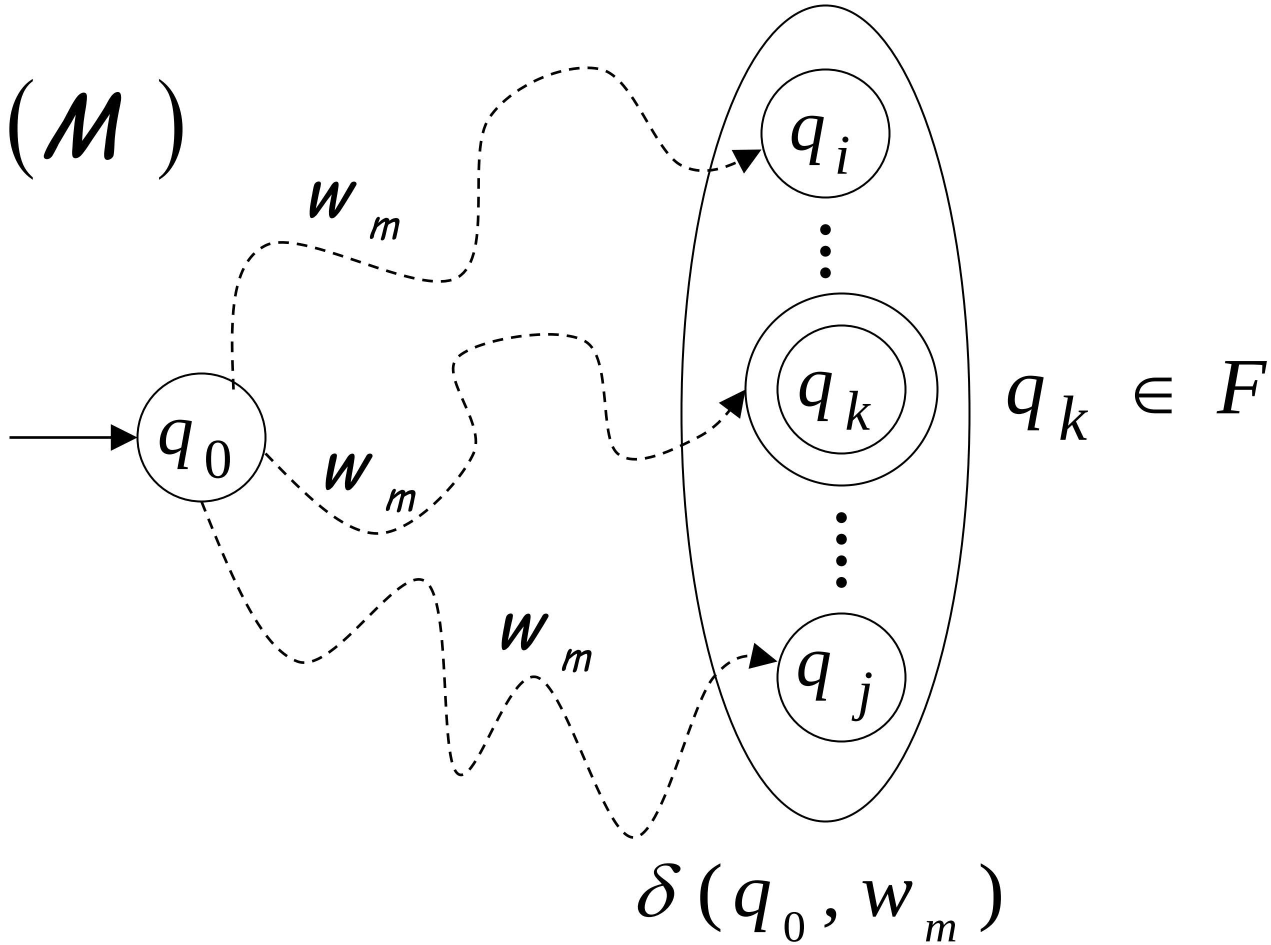
Where for each w_m

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

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

The Language of an NFA

$w_m \in L(M)$



DFA-NFA equivalence

Machine M_1 is equivalent to machine M_2 if and only if

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

DFA-NFA equivalence

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

Languages
accepted
by DFAs

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

DFA-NFA equivalence

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

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

DFA-NFA equivalence

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 accepted by a DFA is also accepted by a NFA

DFA-NFA equivalence

Proof: Step 2

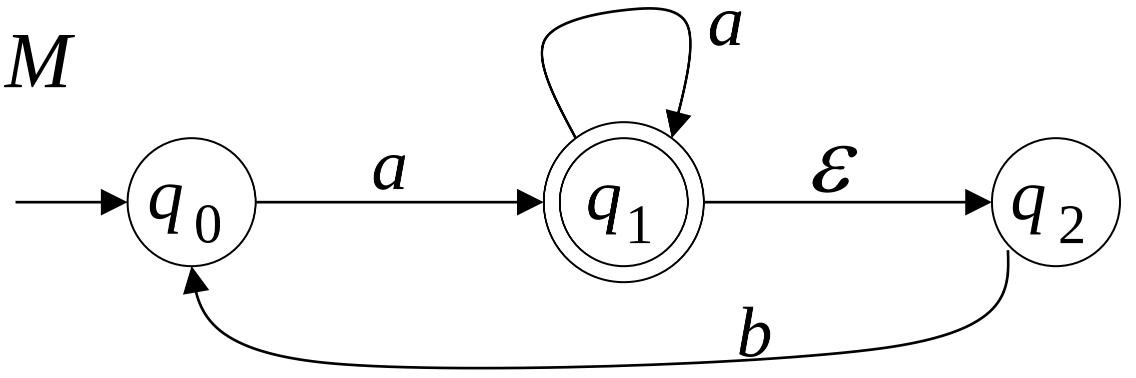
$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Any NFA can be converted to an equivalent DFA

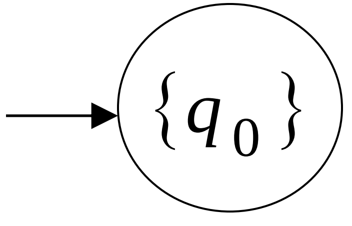
Any language accepted by a NFA is also accepted by a DFA

Conversion of NFA to DFA

NFA M

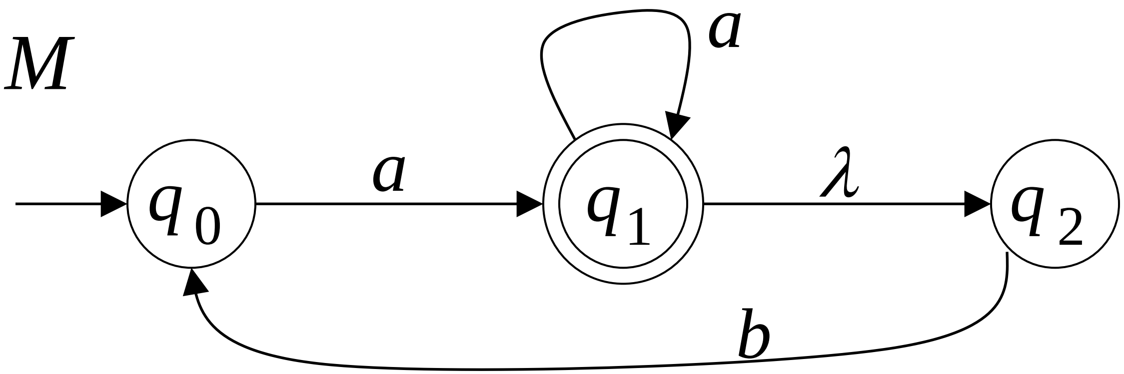


DFA M'



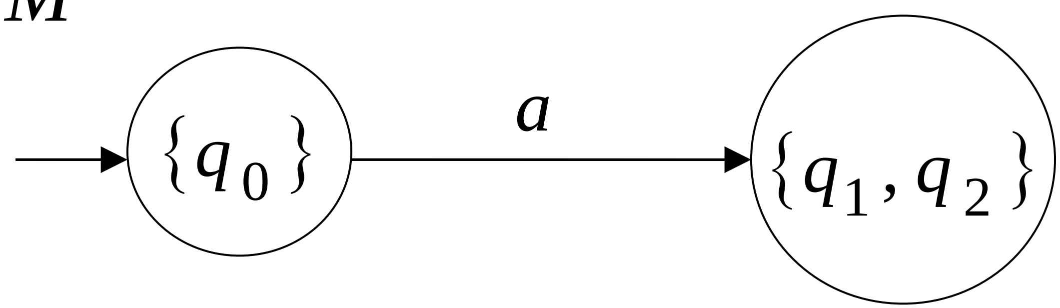
Conversion of NFA to DFA

NFA M



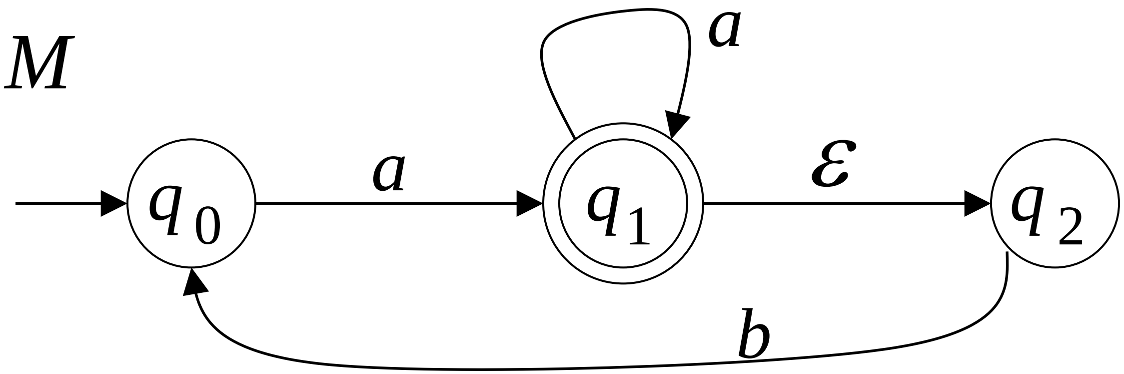
$$\delta(q_0, a) = \{q_1, q_2\}$$

DFA M'



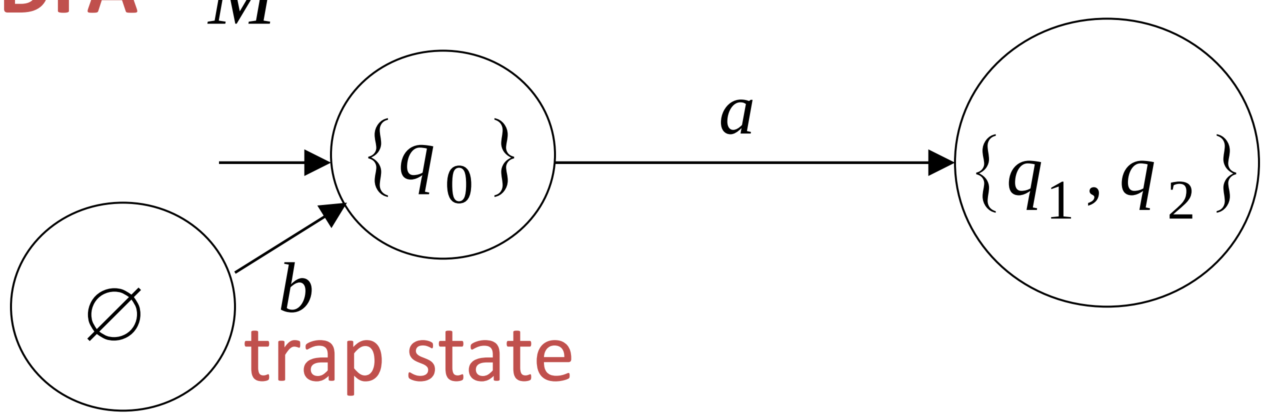
Conversion of NFA to DFA

NFA M

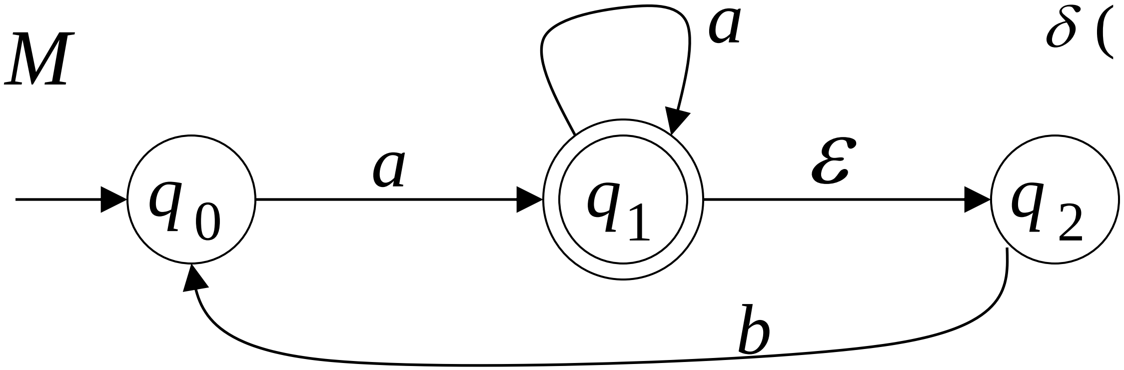


$$\delta(q_0, b) = \varnothing$$ empty set

DFA M'



NFA M



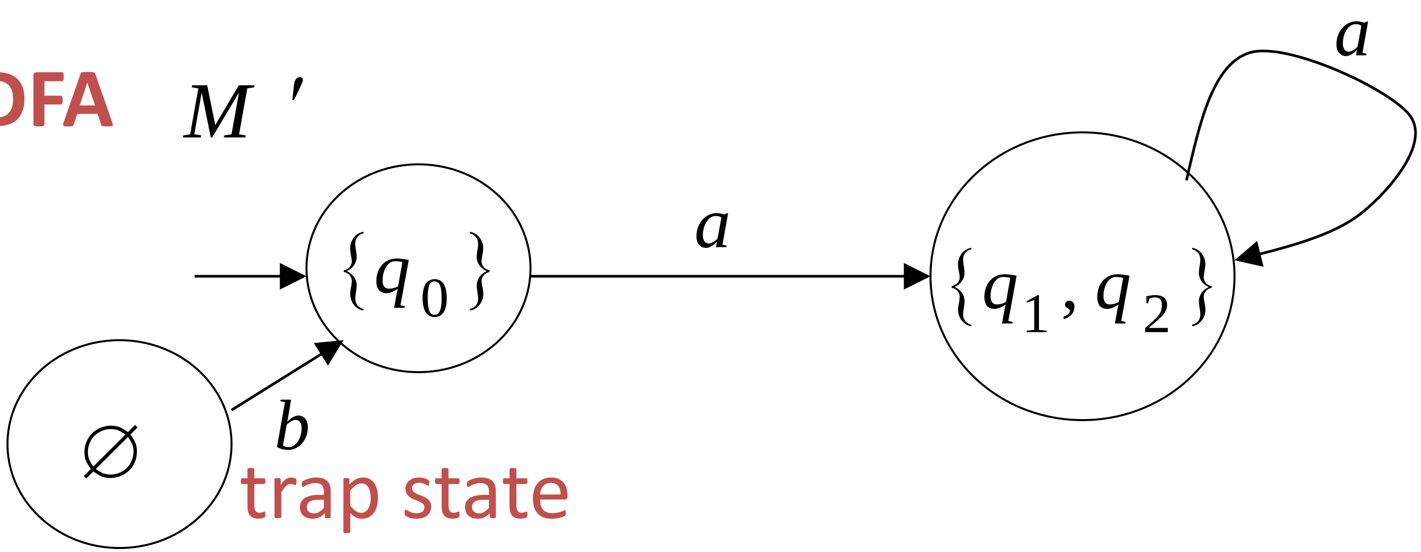
$$\delta(q_1, a) = \{q_1, q_2\}$$

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

union

$$\{q_1, q_2\}$$

DFA M'



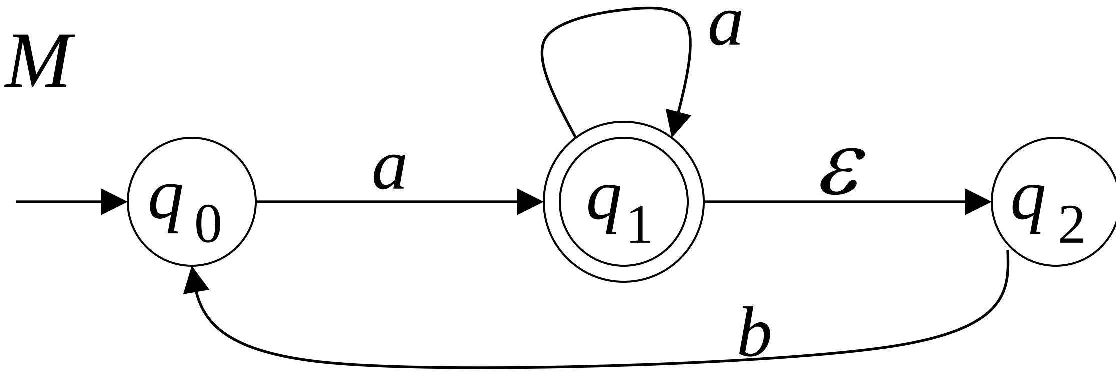
NFA M

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

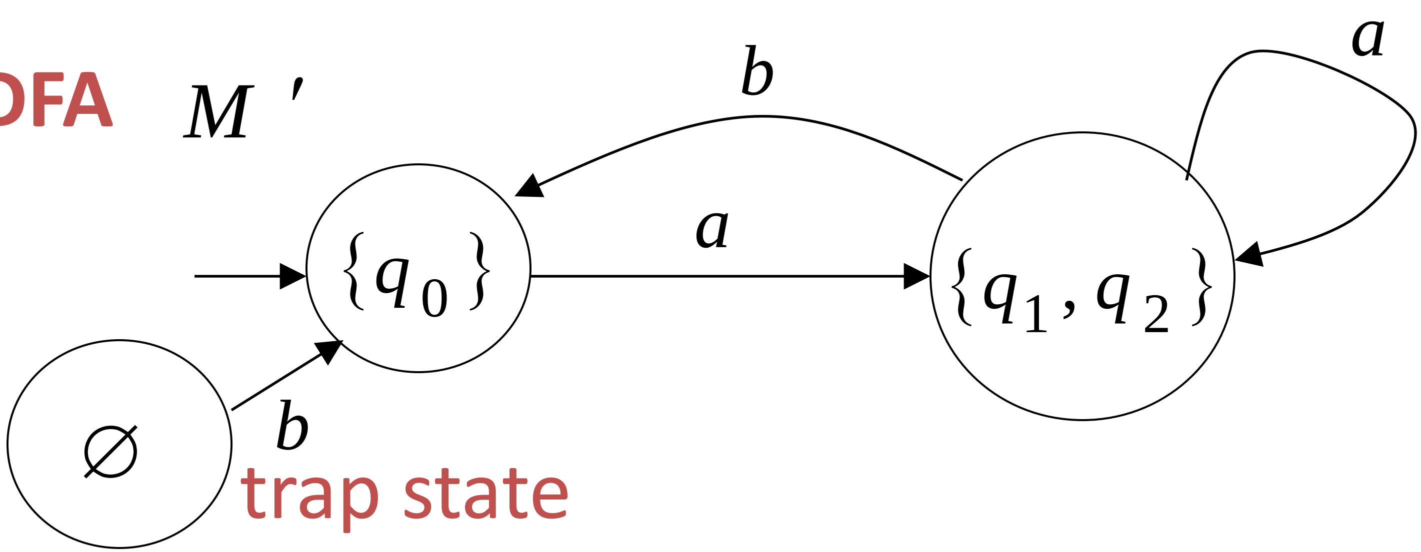
$$\delta(q_2, b) = \{q_0\}$$

union

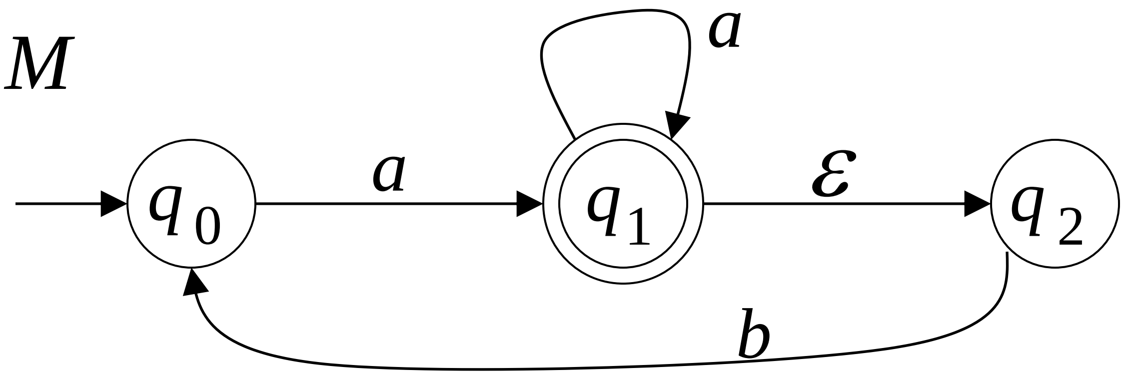
$$\{q_0\}$$



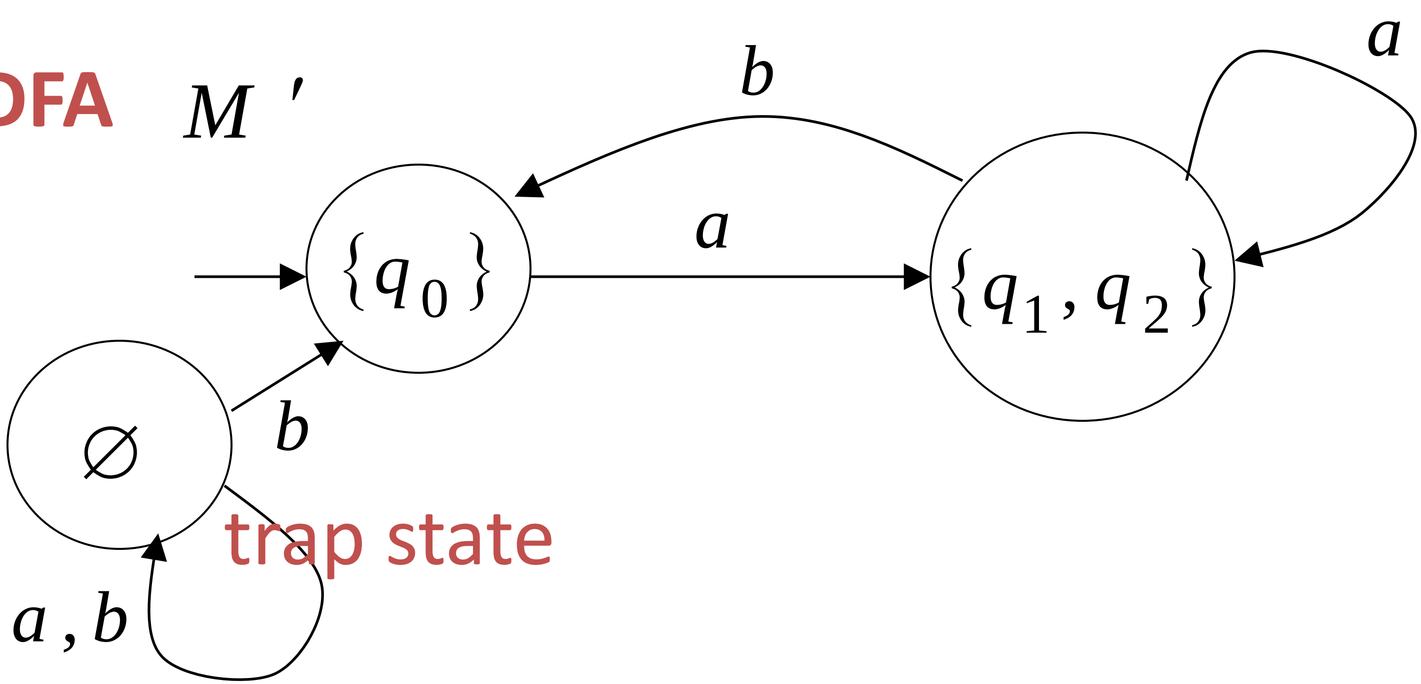
DFA M'



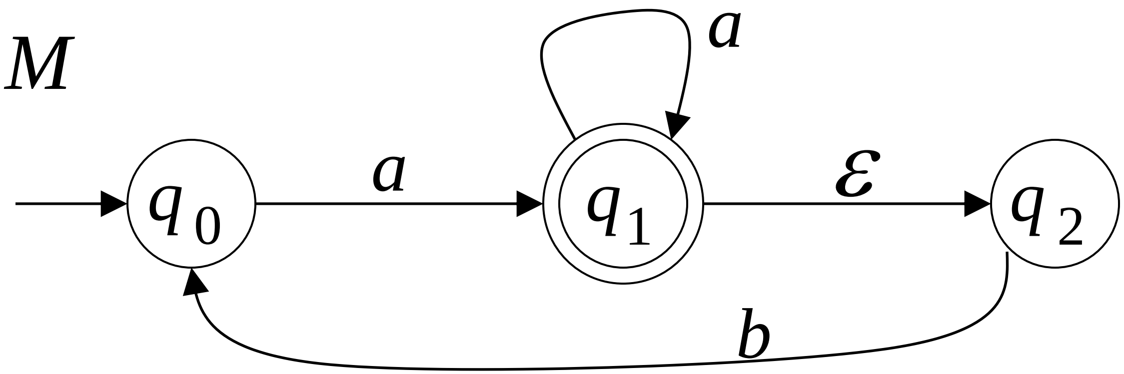
NFA M



DFA M'

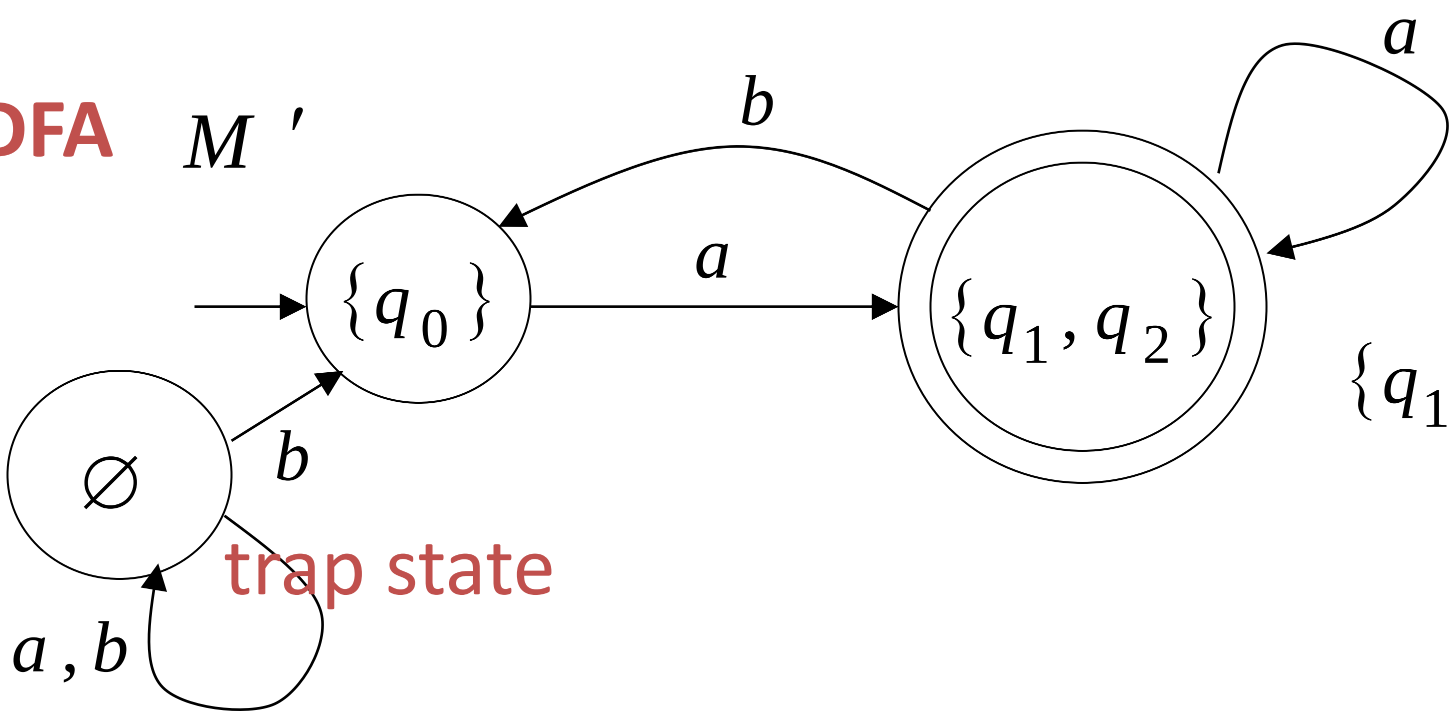


NFA M



$q_1 \in F$

DFA M'



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

Conversion NFA to DFA

Input: an NFA M

Output: an equivalent DFA M'

The NFA has states

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

The DFA has states from the power set

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

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

Conversion NFA to DFA

<u>STEP 1</u> Initial state of NFA: q_0

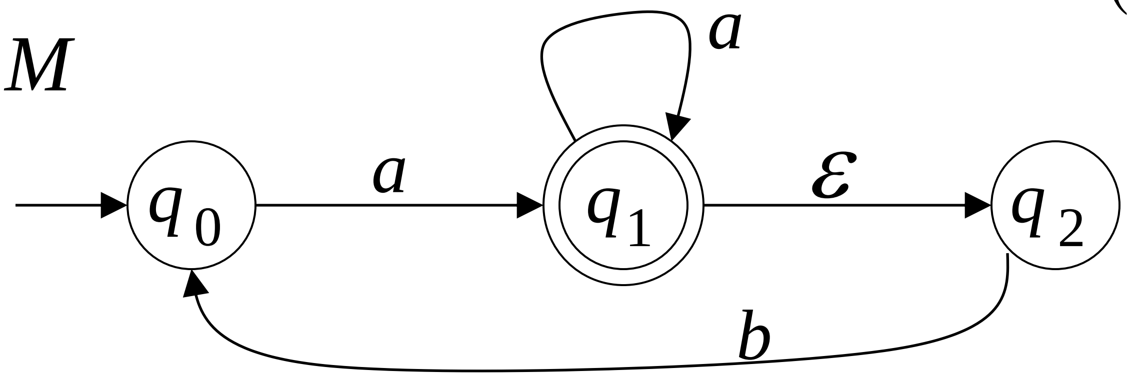
$$\delta(q_0, \varepsilon) = \{q_0, \ldots\}$$

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

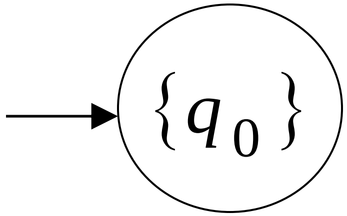
Conversion NFA to DFA

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

NFA M



DFA M'



Conversion NFA to DFA

<u>STEP 2</u> For every DFA's state $\{q_i, q_j, \ldots, q_m\}$

compute in the NFA

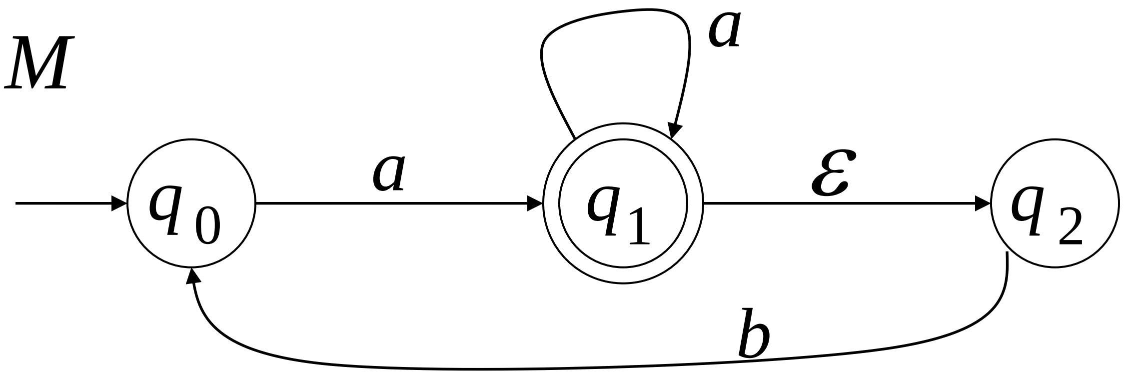
$$\left.\begin{aligned} &\delta(q_i, a) \\ \cup\ &\delta(q_j, a) \\ &\ldots \\ \cup\ &\delta(q_m, a) \end{aligned}\right\} \overset{\text{Union}}{=} \{q'_k, q'_l, \ldots, q'_n\}$$

add transition to DFA

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

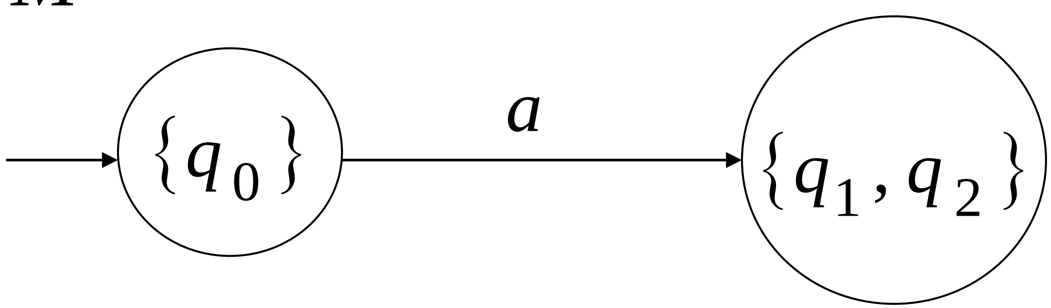
Conversion NFA to DFA

NFA M



DFA M'

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



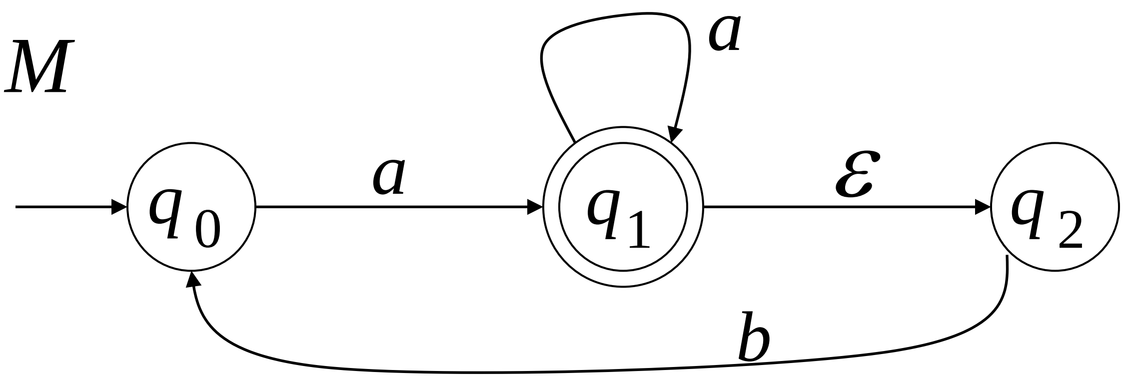
$$\delta'(\{q_0\}, a) = \{q_1, q_2\}$$

Conversion NFA to DFA

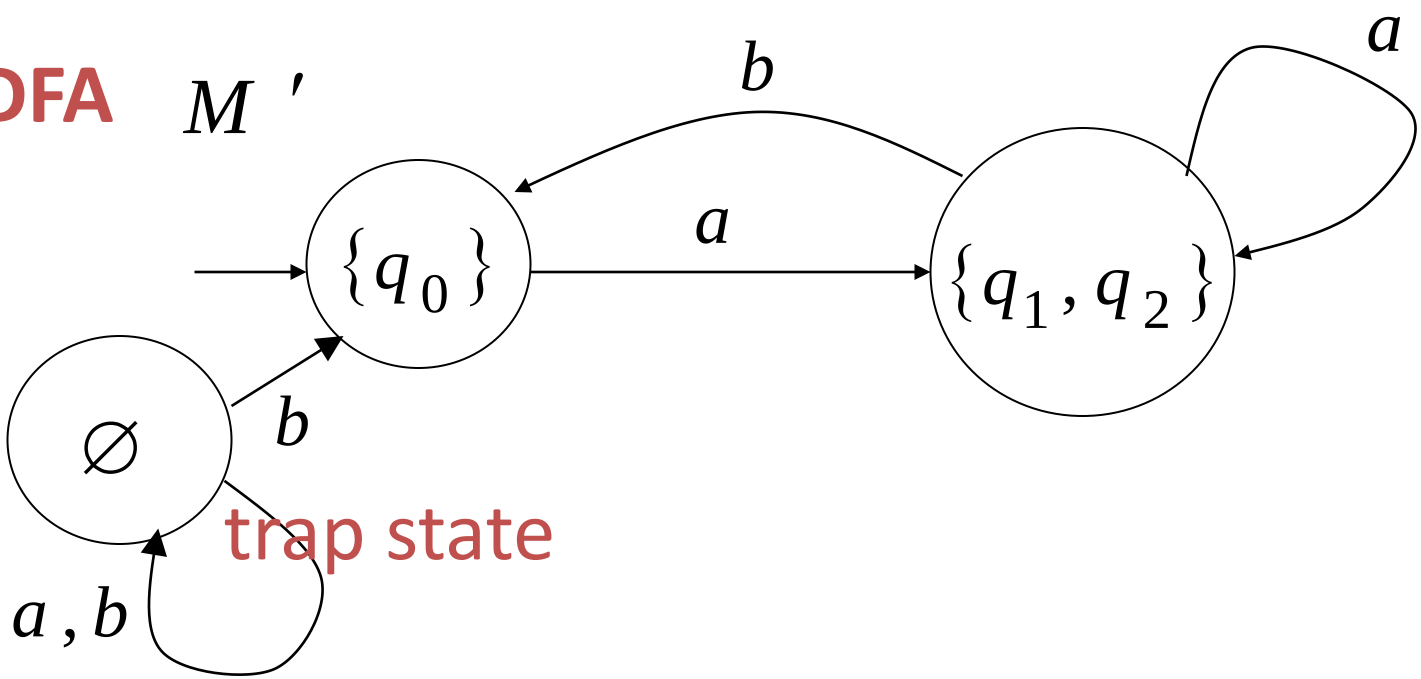
STEP 3 Repeat STEP 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA

Conversion NFA to DFA

NFA M



DFA M'



Conversion NFA to DFA

<u>STEP 4</u> 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

Conversion NFA to DFA