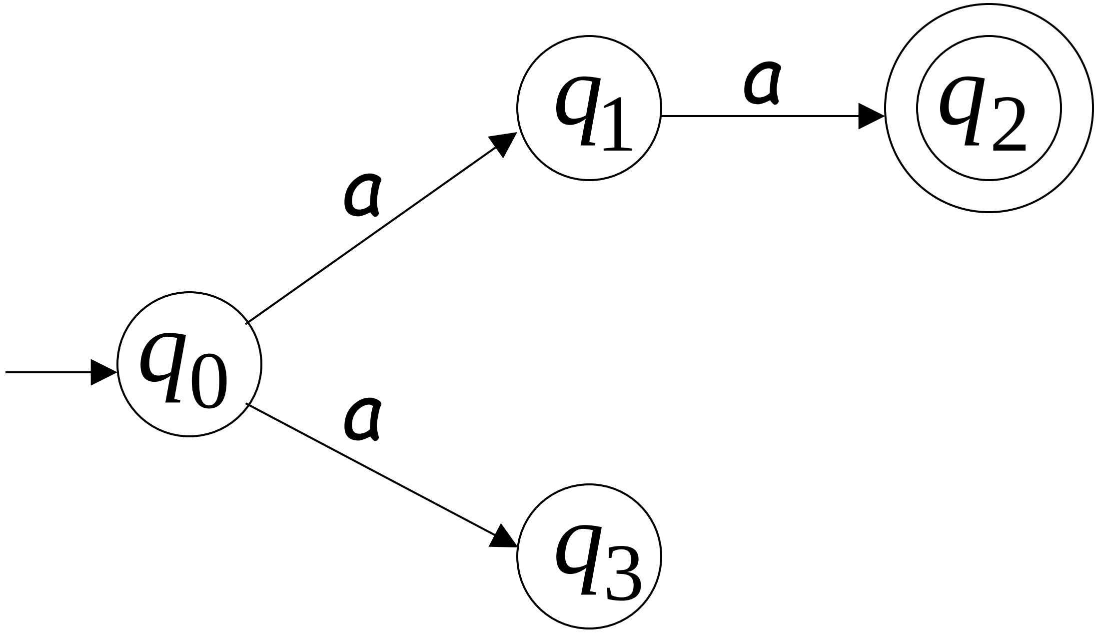
Non-Deterministic Finite Automata

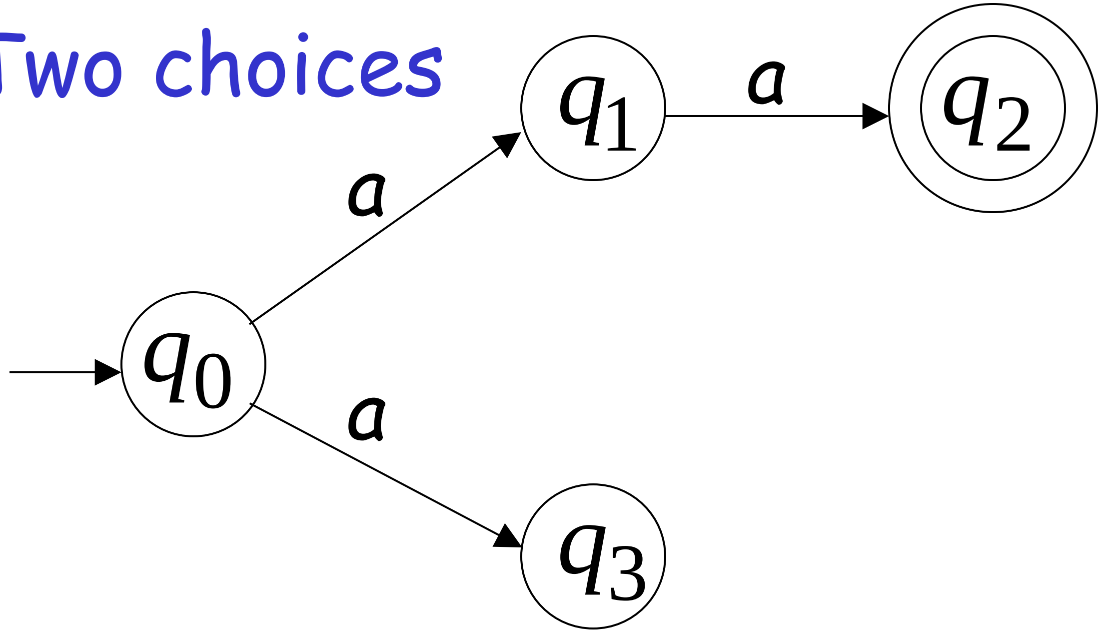
Nondeterministic Finite Automaton (NFA)

Alphabet = $\{a\}$

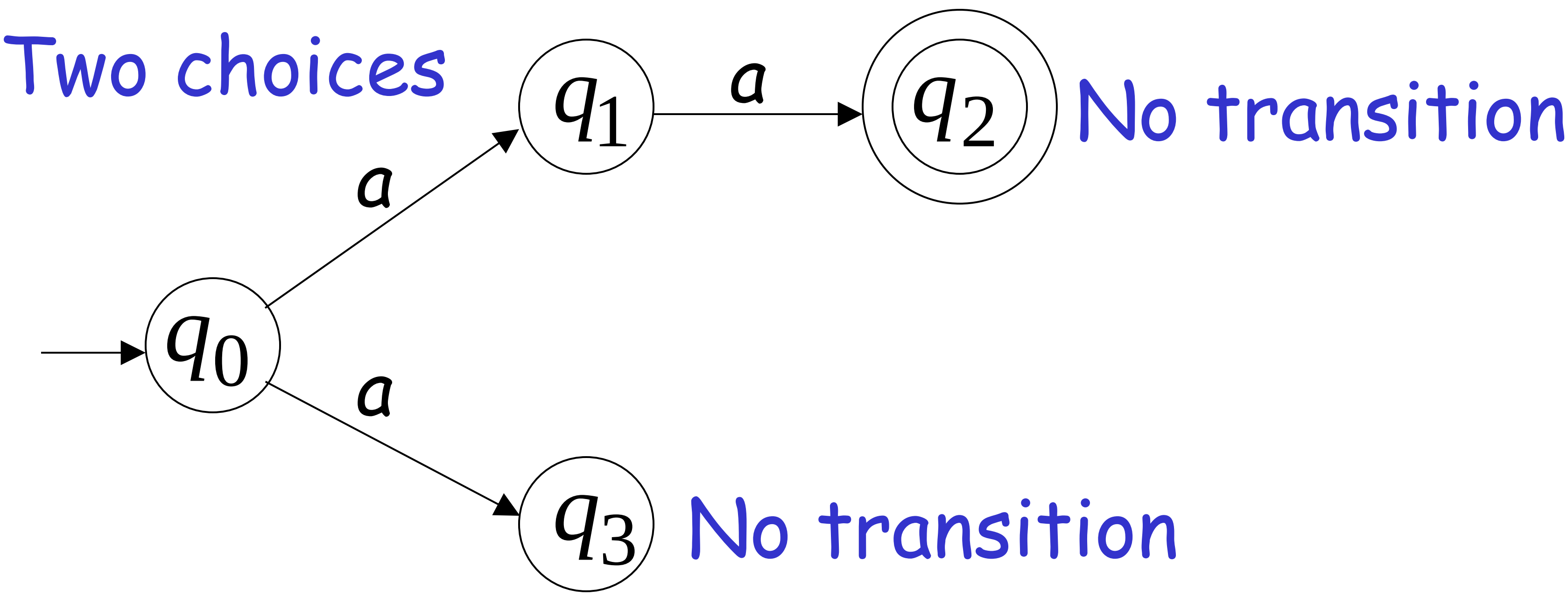


Alphabet = $\{a\}$

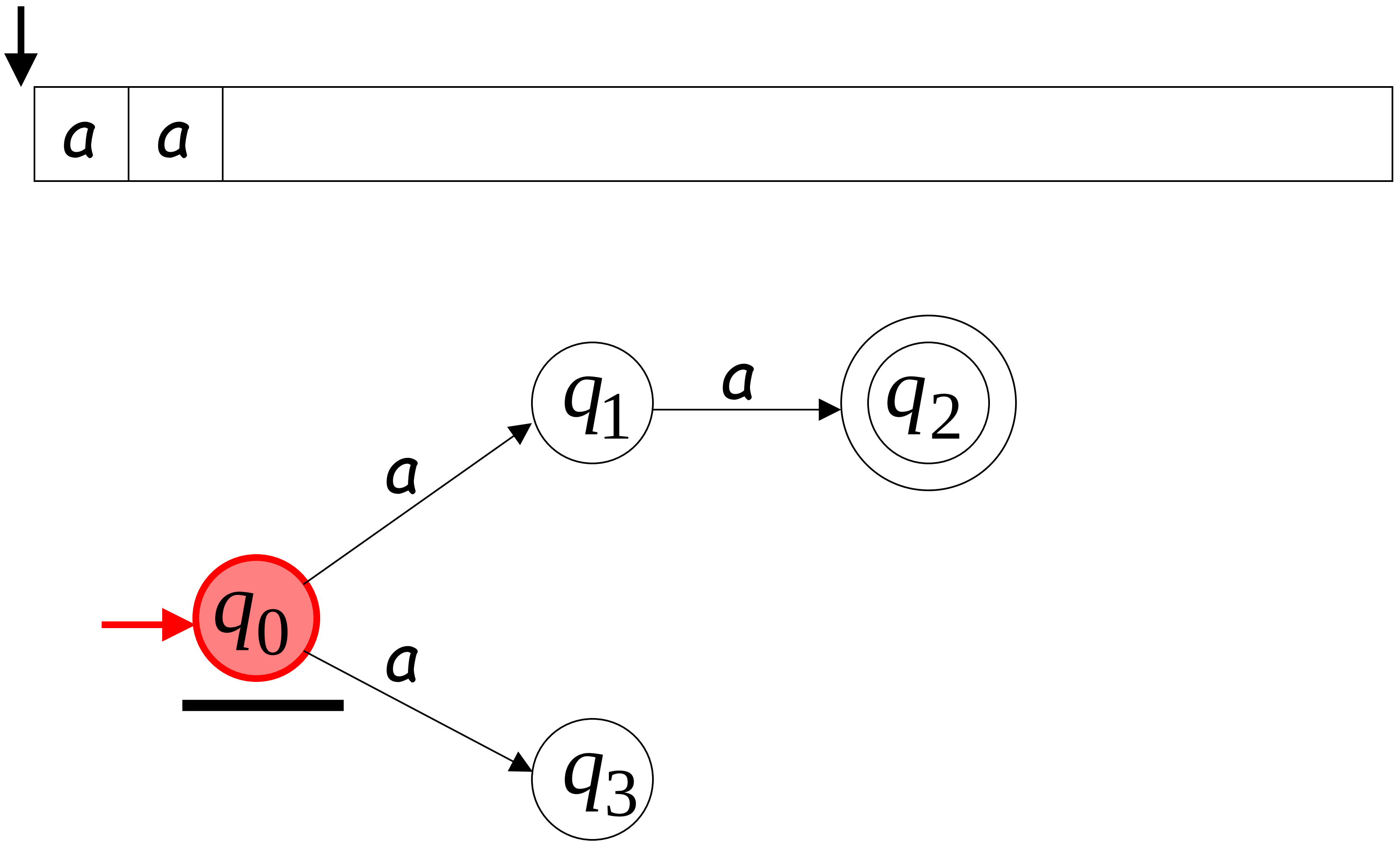
Two choices



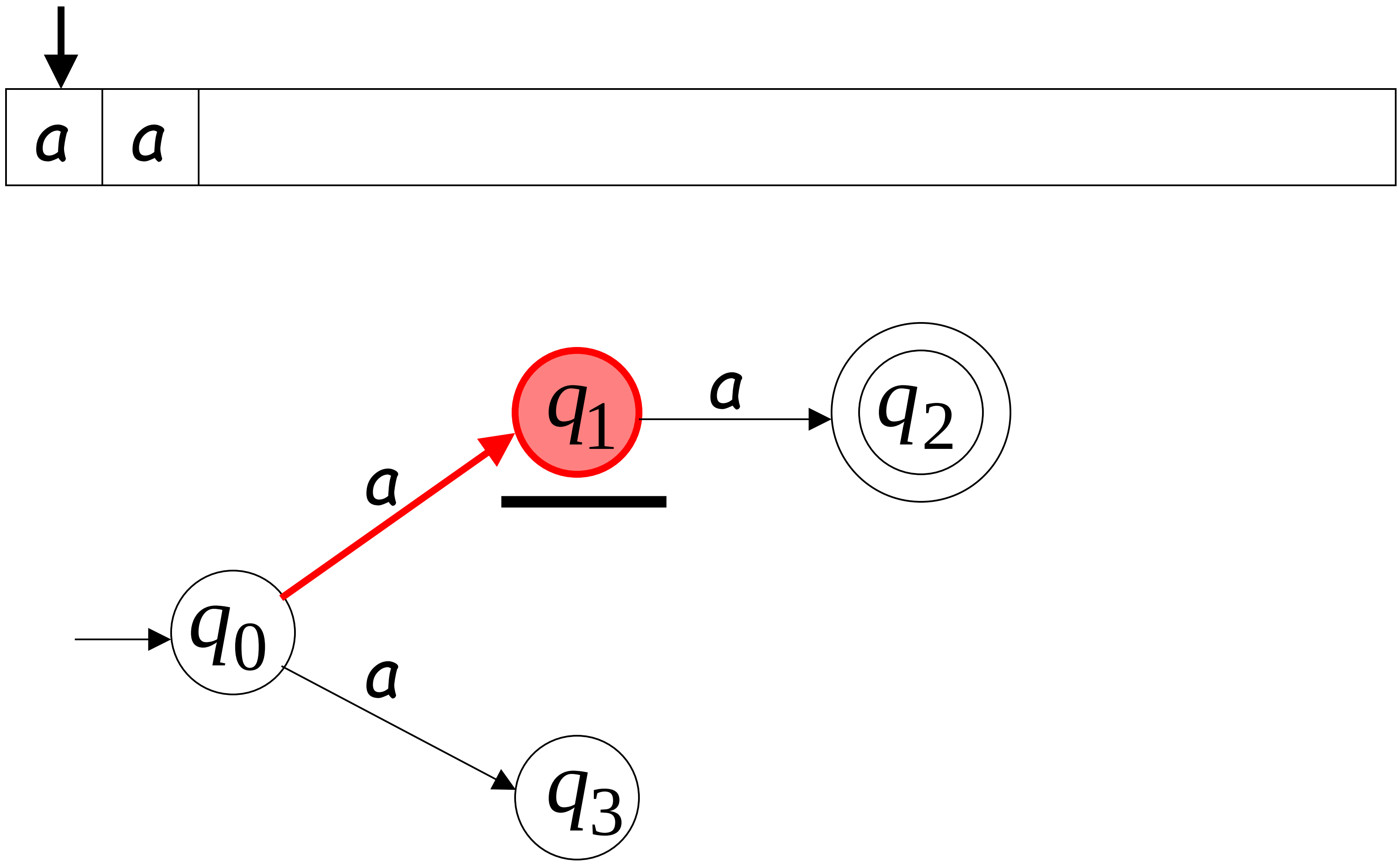
Alphabet = $\{a\}$



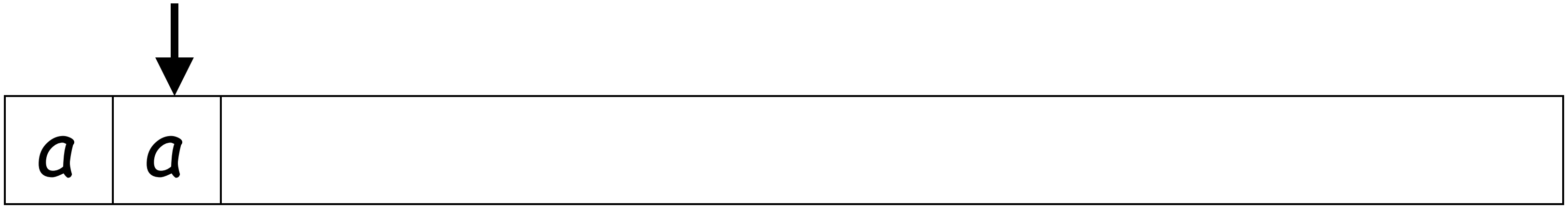
First Choice



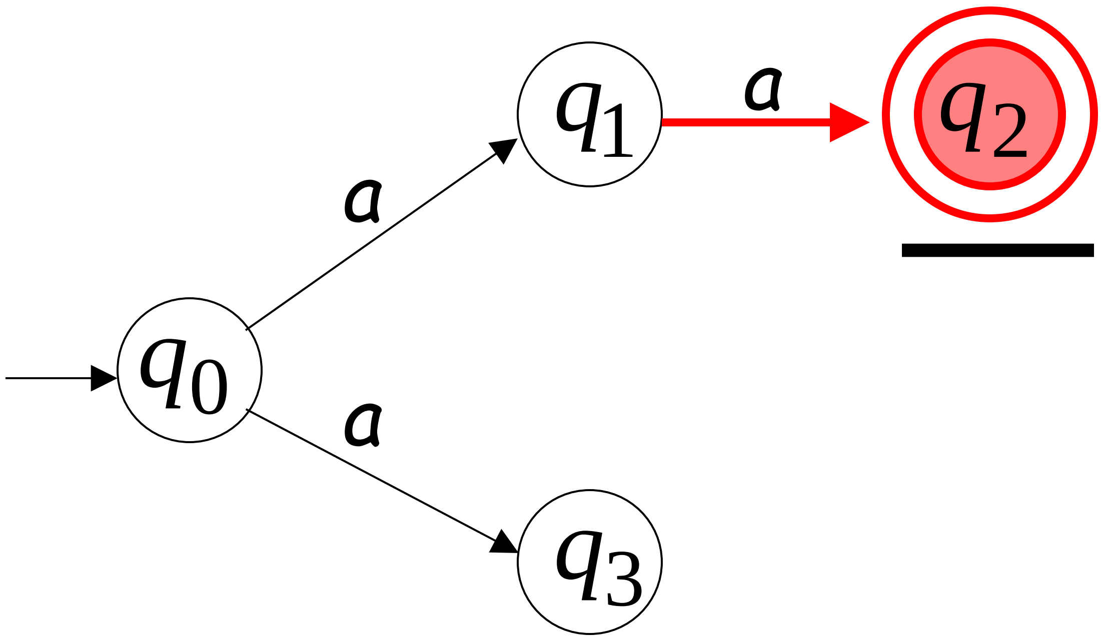
First Choice



First Choice

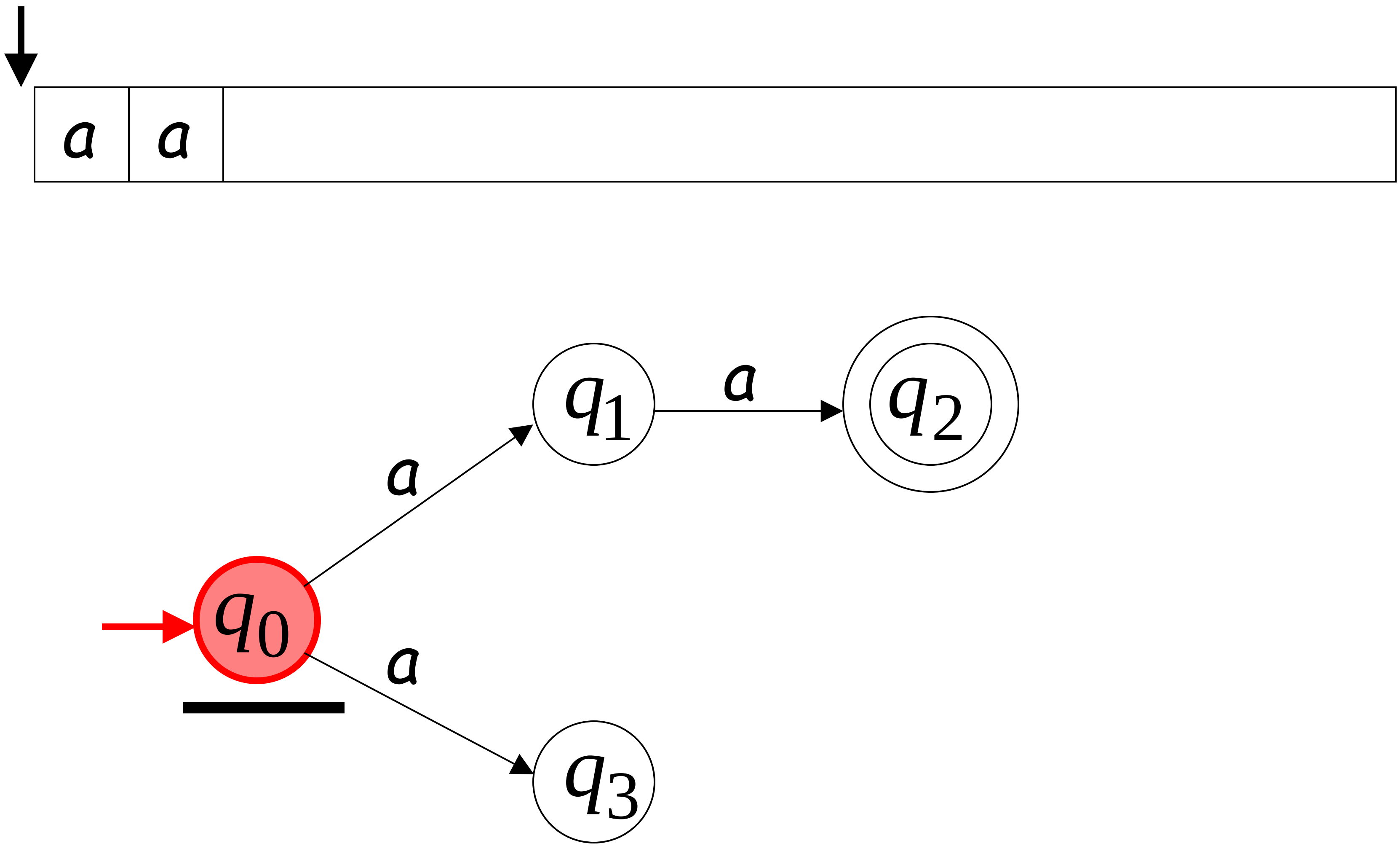


All input is consumed

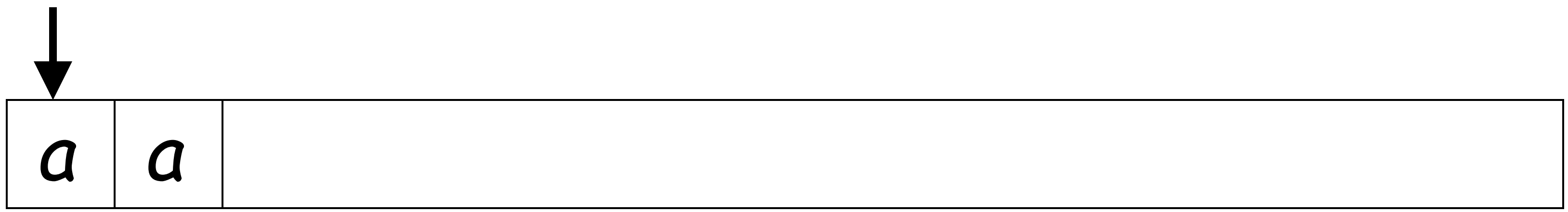


“accept”

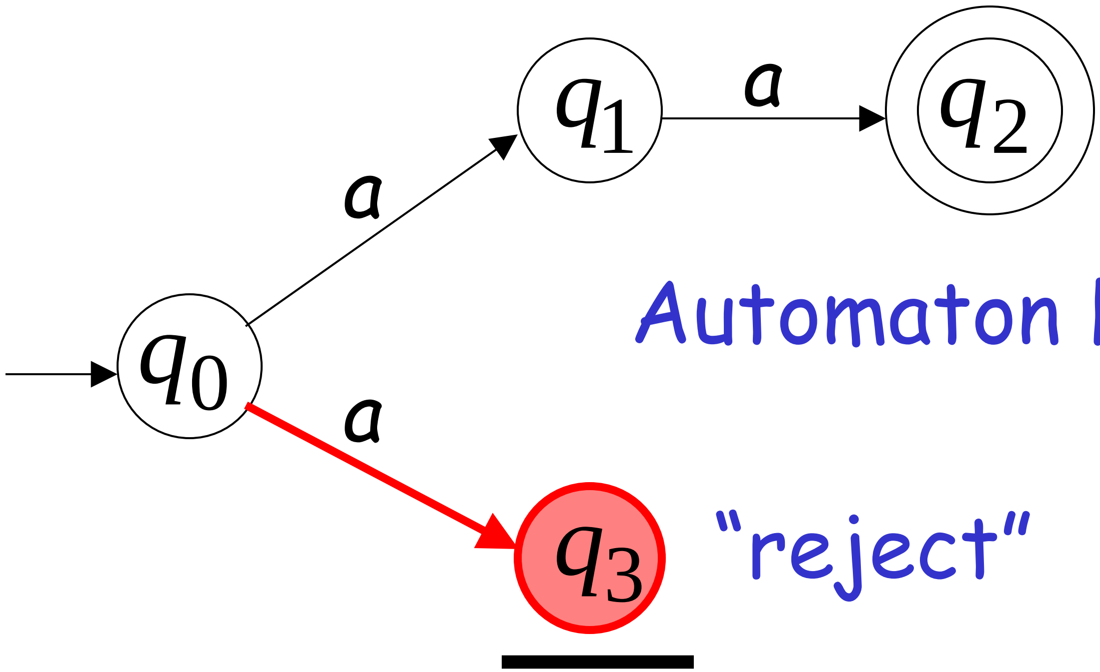
Second Choice



Second Choice



Input cannot be consumed



Automaton Halts

"reject"

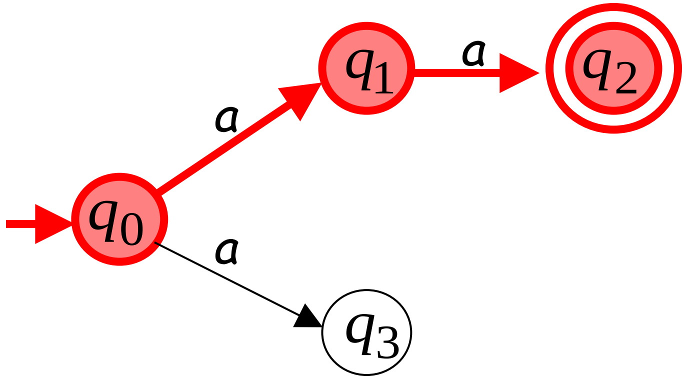
An NFA accepts a string:

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

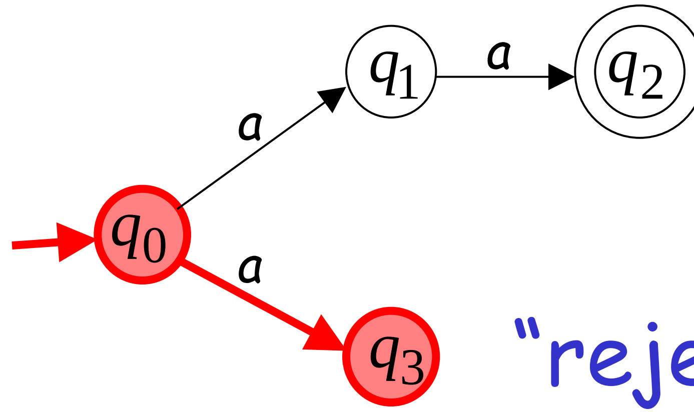
i.e., all the input string is processed and the automaton is in an accepting state

aa is accepted by the NFA:

"accept"



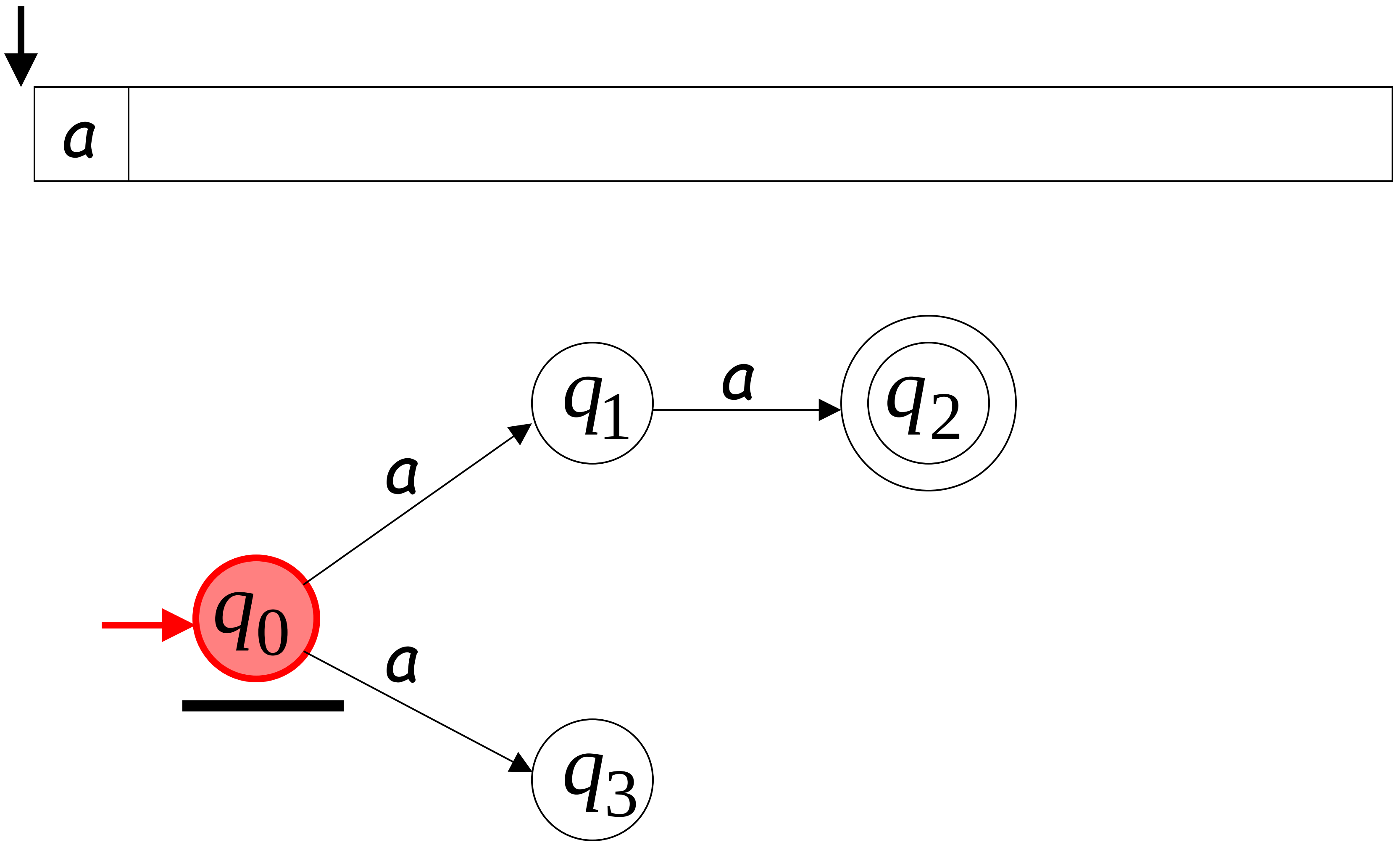
because this computation accepts aa



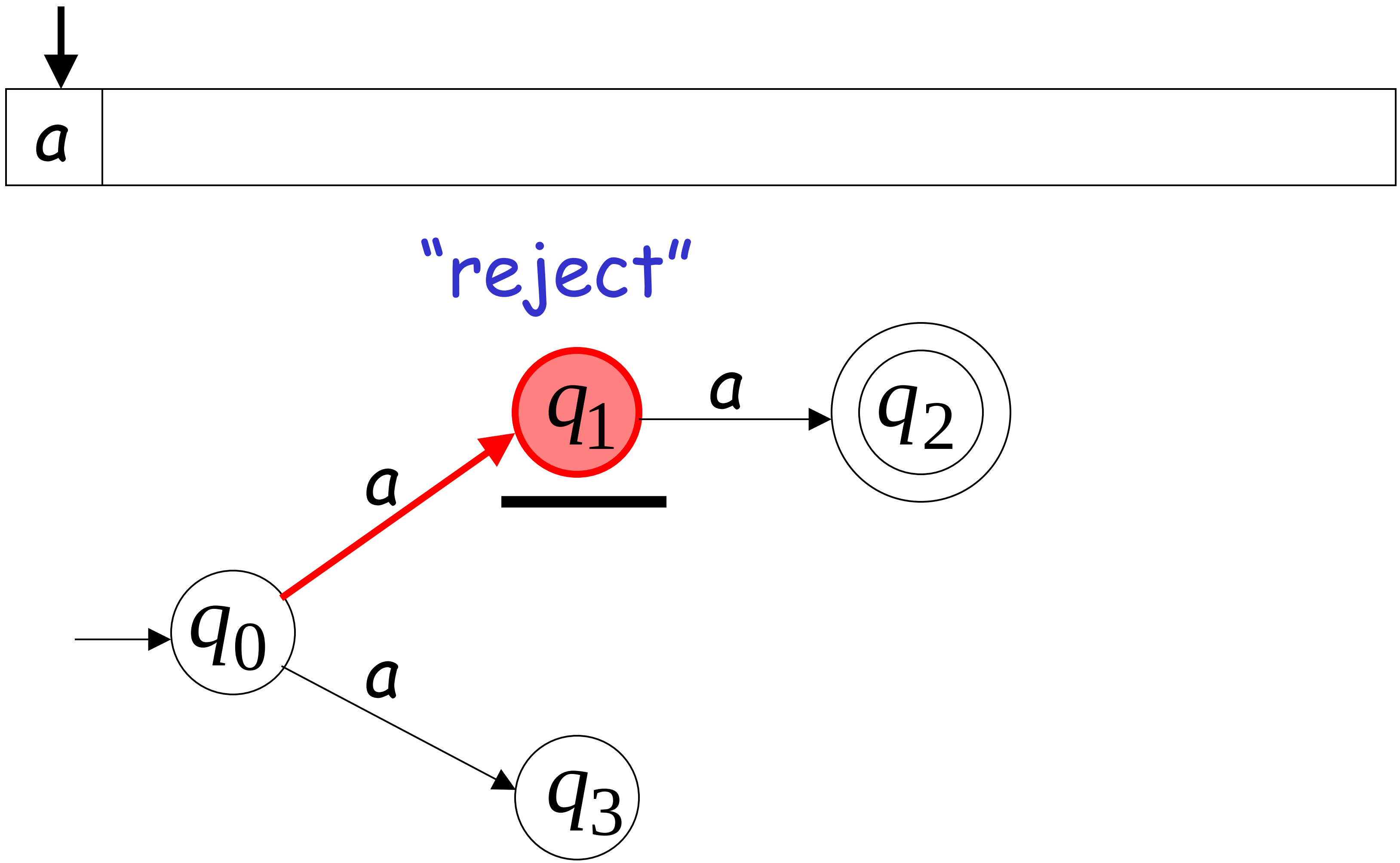
"reject"

this computation is ignored

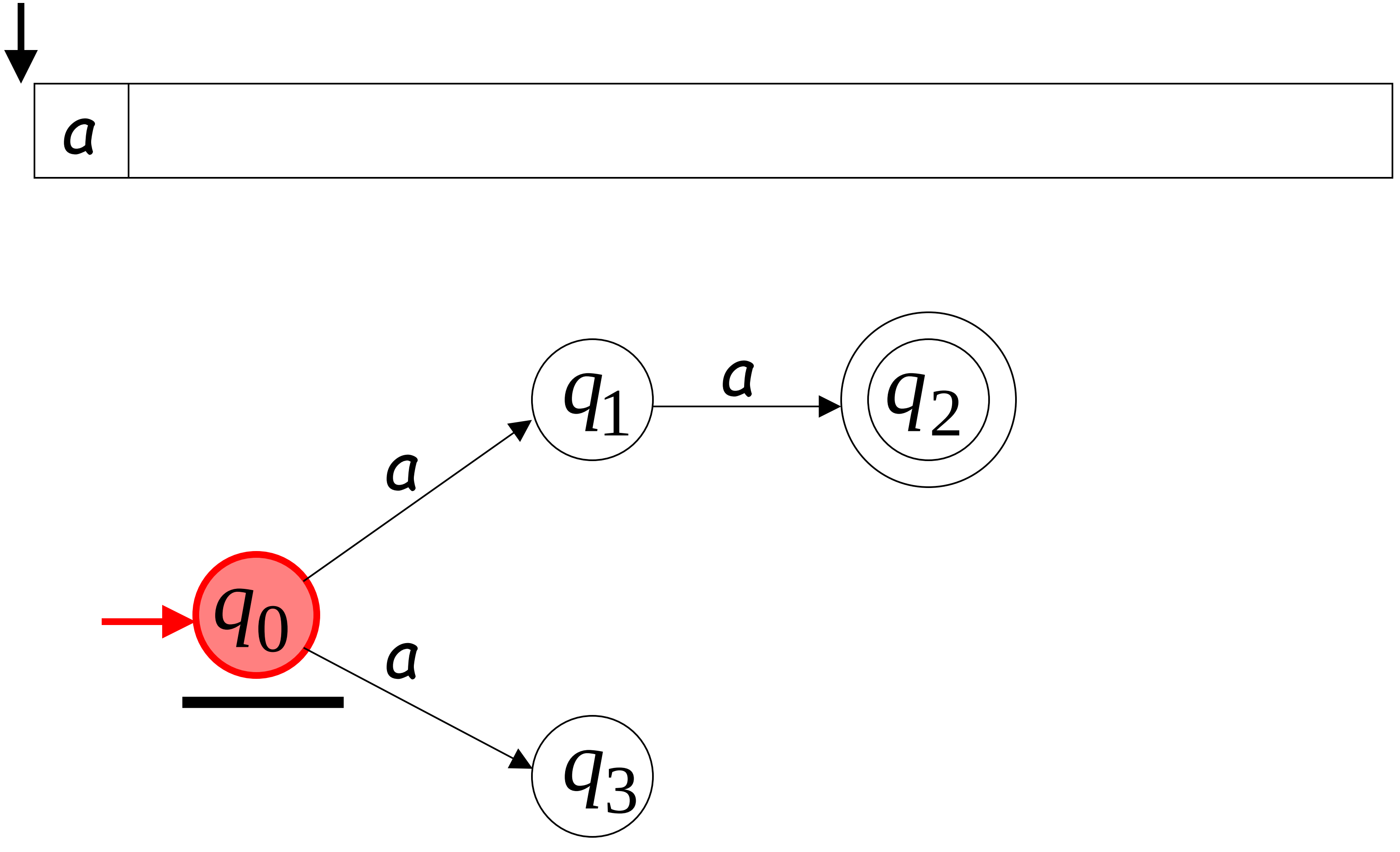
Rejection example



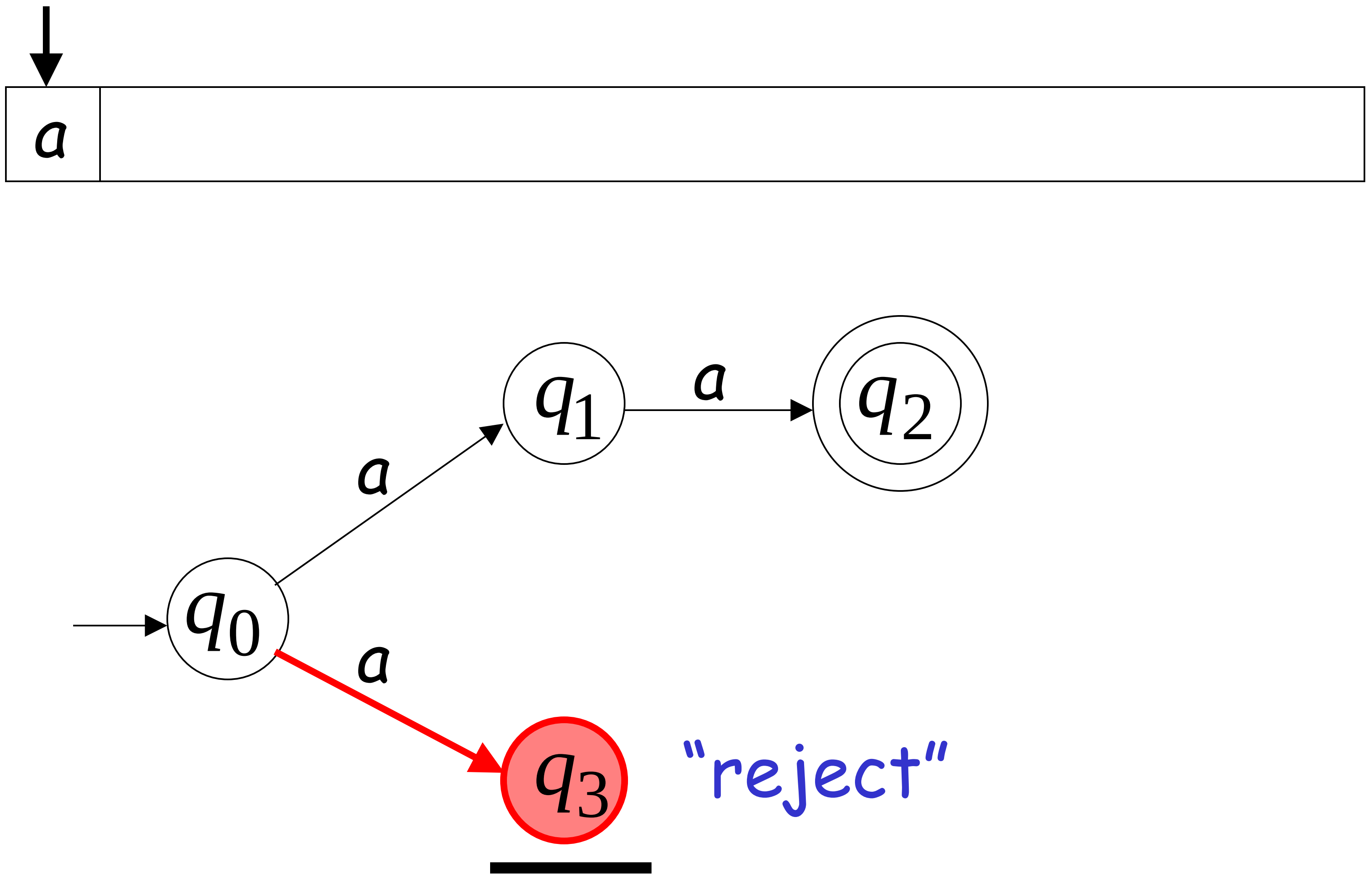
First Choice



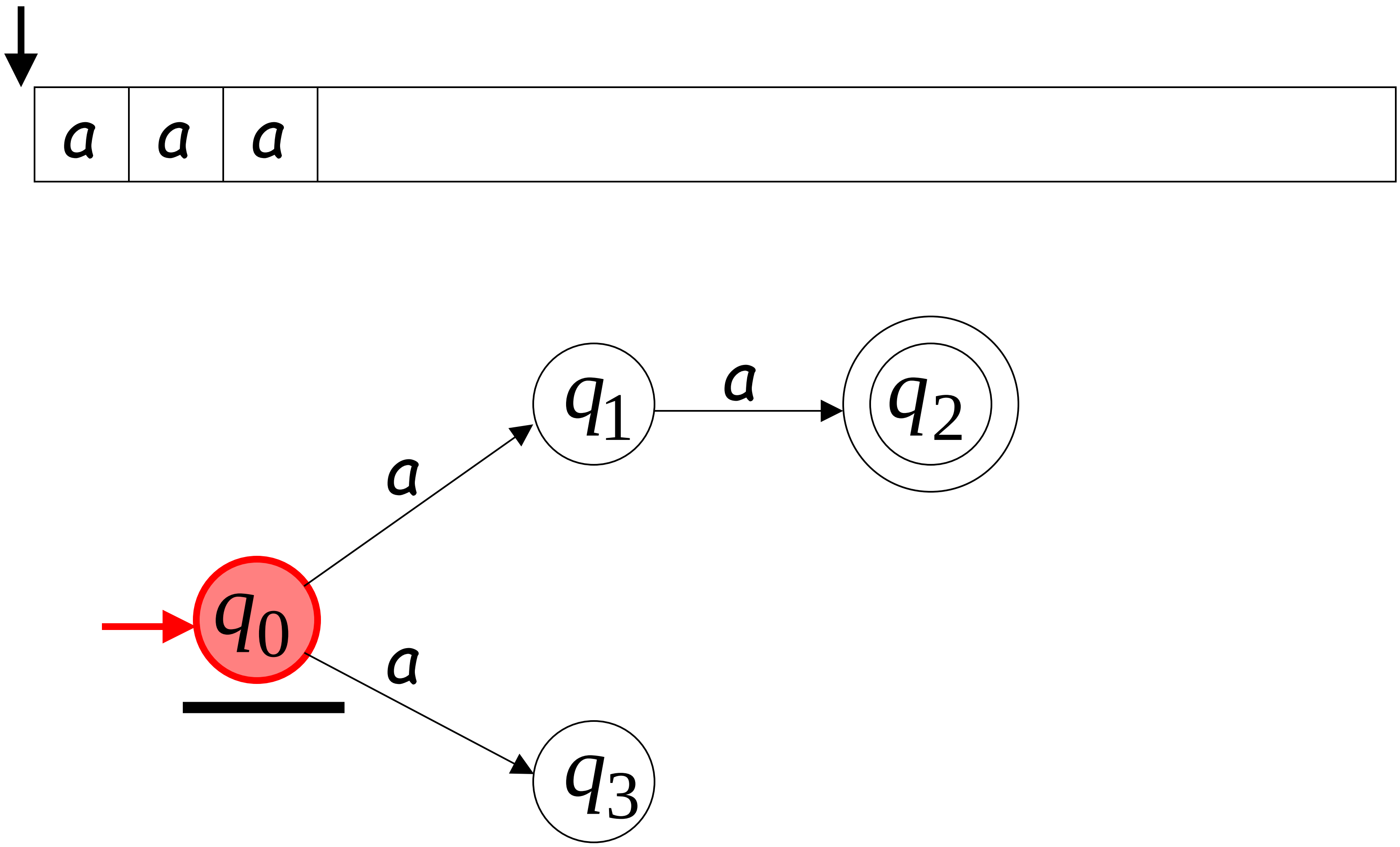
Second Choice



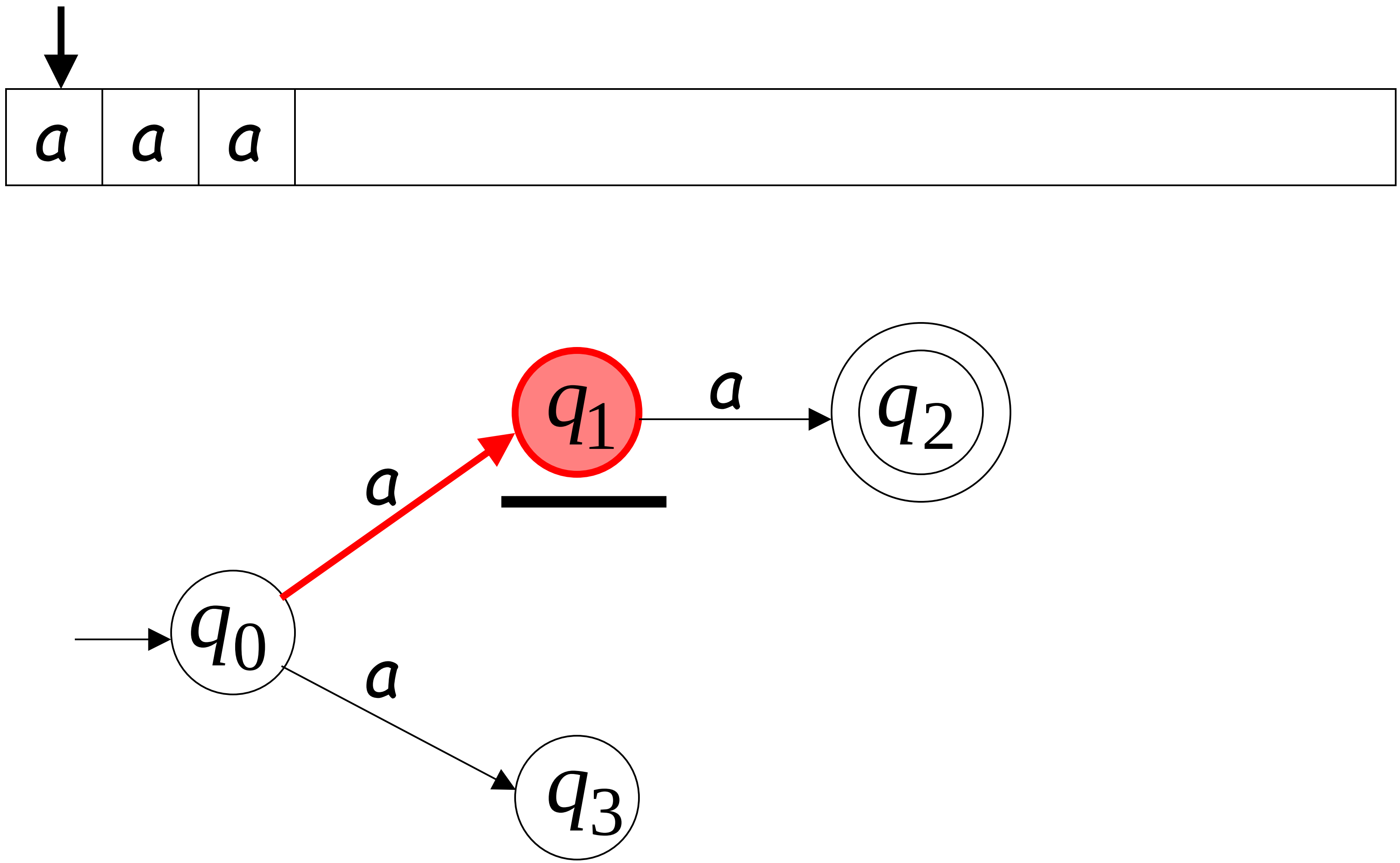
Second Choice



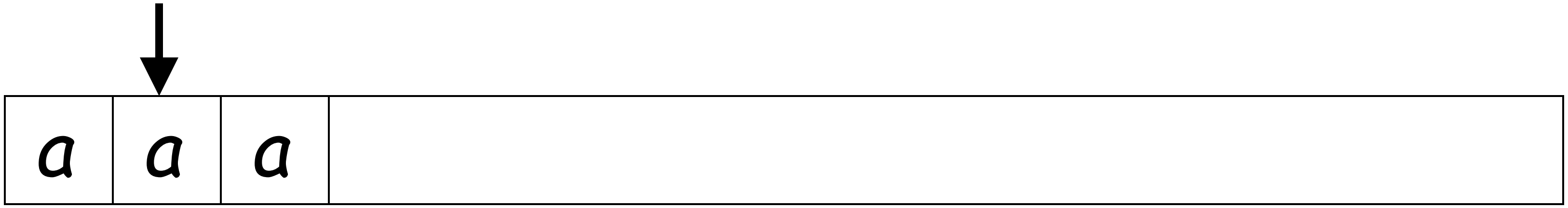
Another Rejection example



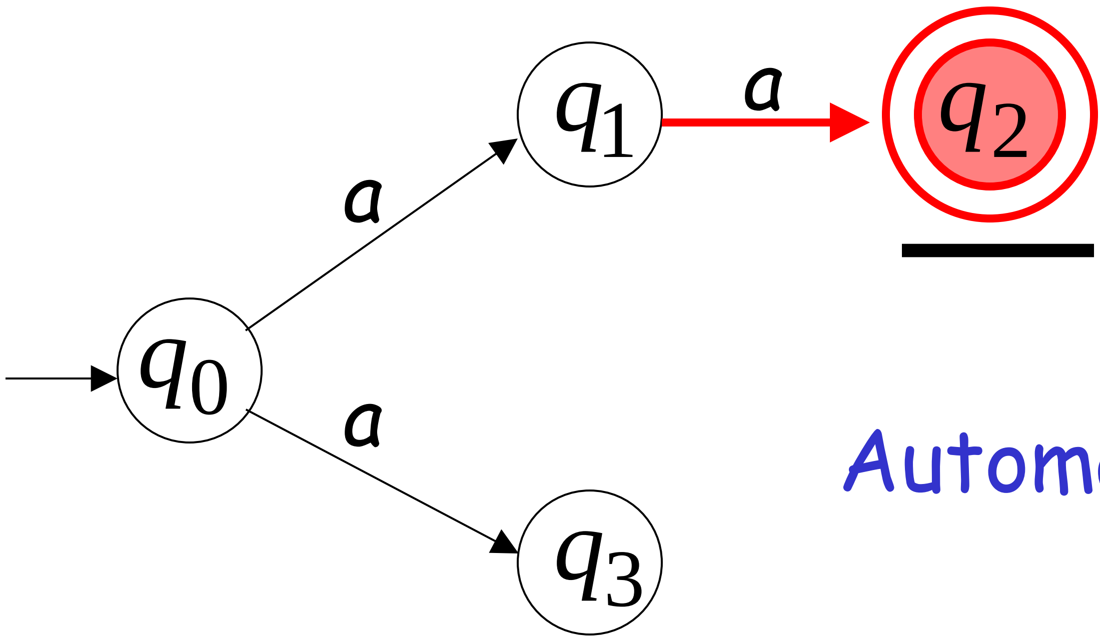
First Choice



First Choice



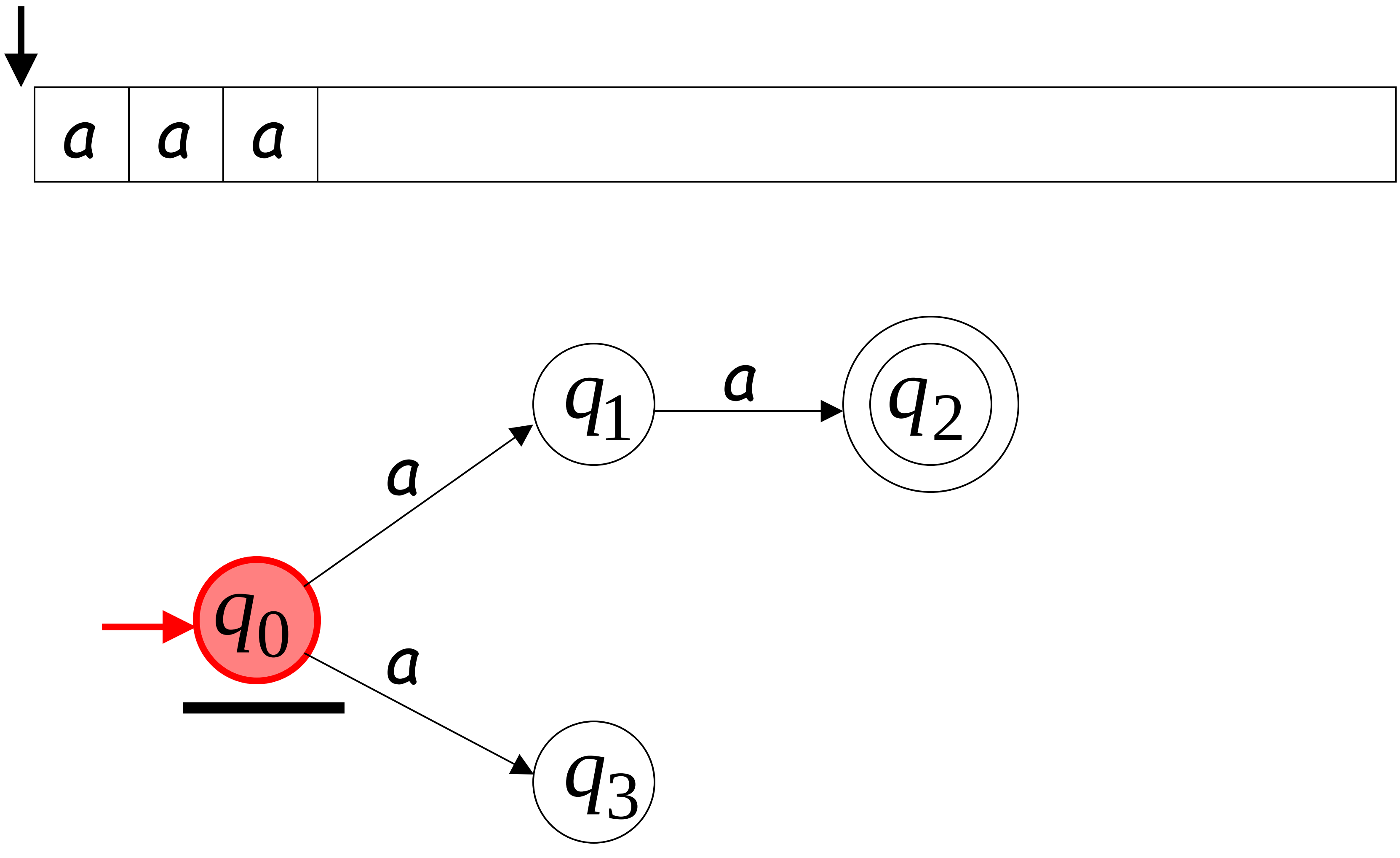
Input cannot be consumed



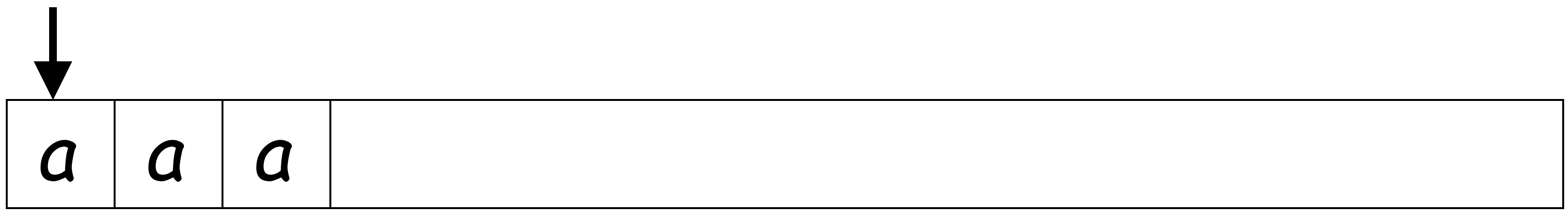
"reject"

Automaton halts

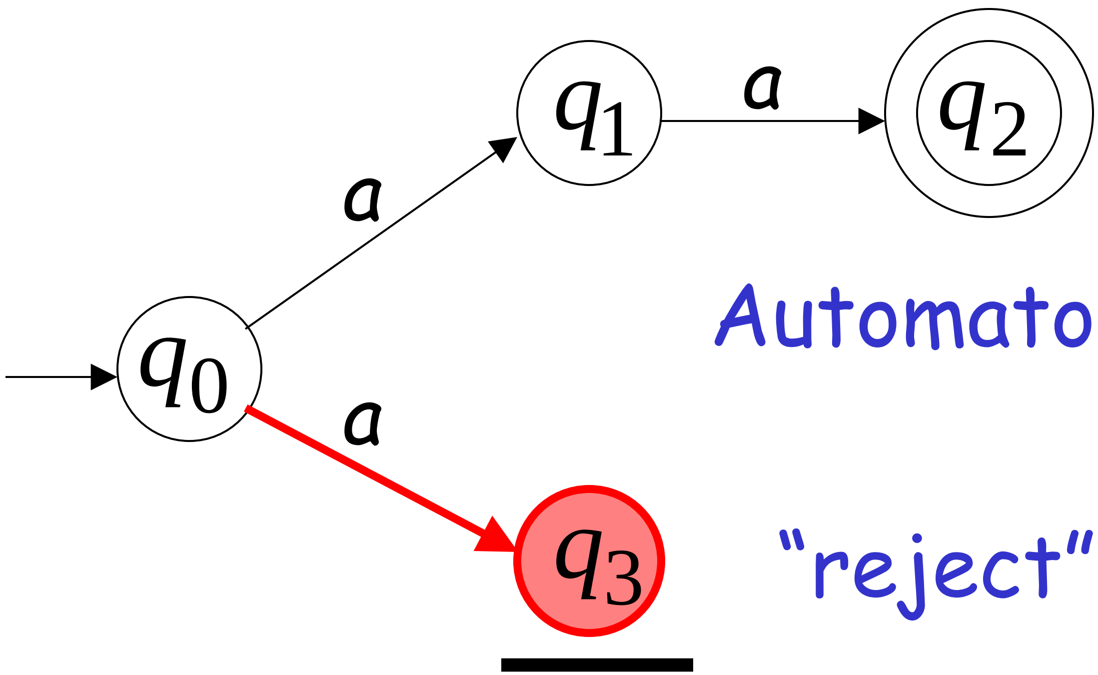
Second Choice



Second Choice



Input cannot be consumed



Automaton halts

“reject”

An NFA rejects a string:

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

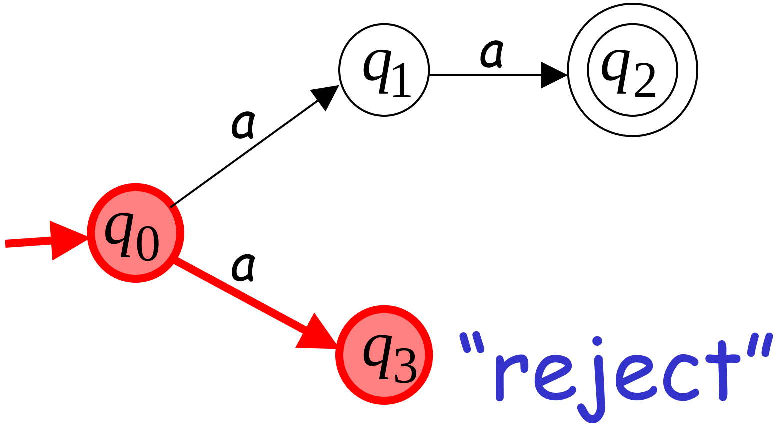
For each computation:

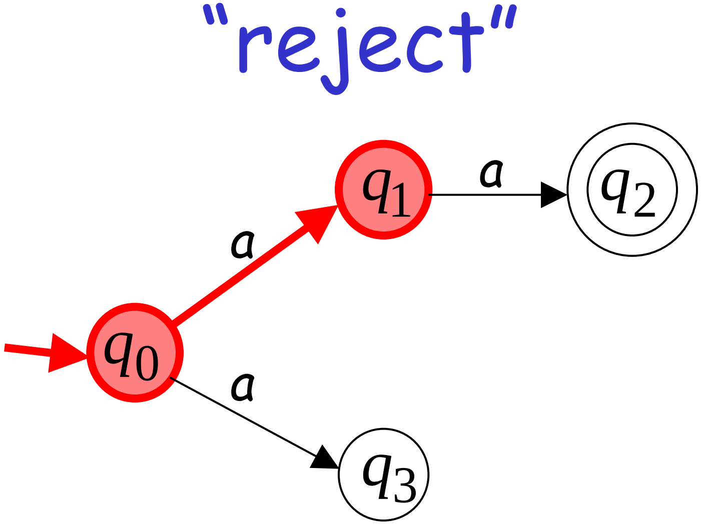
- All the input is consumed and the automaton is in a non final state

OR

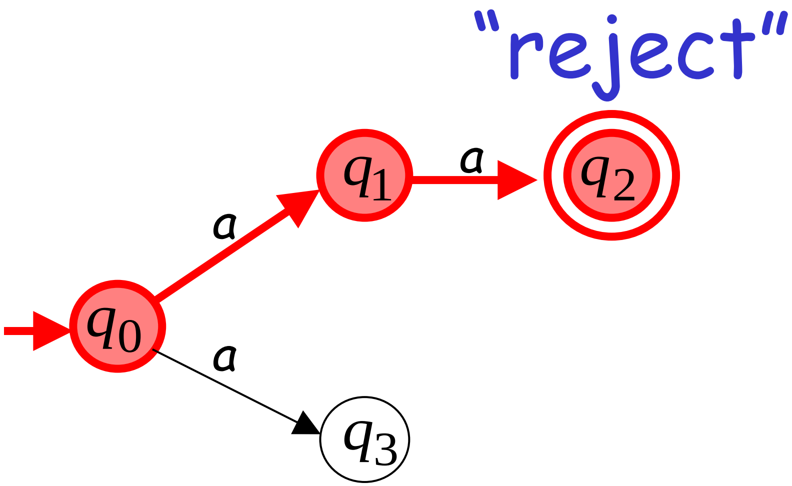
- The input cannot be consumed

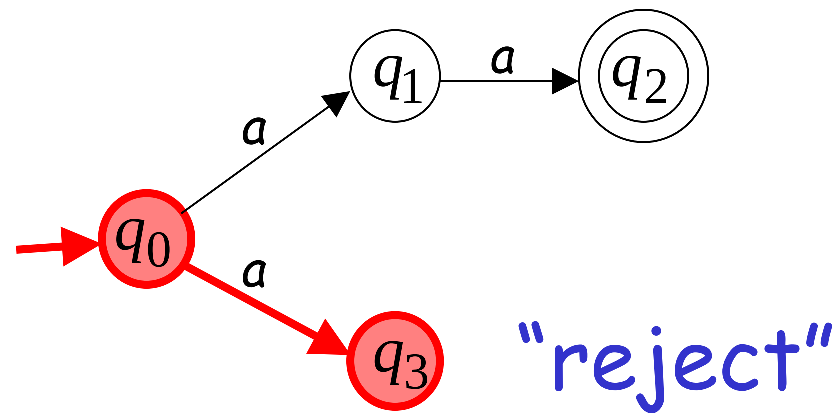
a is rejected by the NFA:





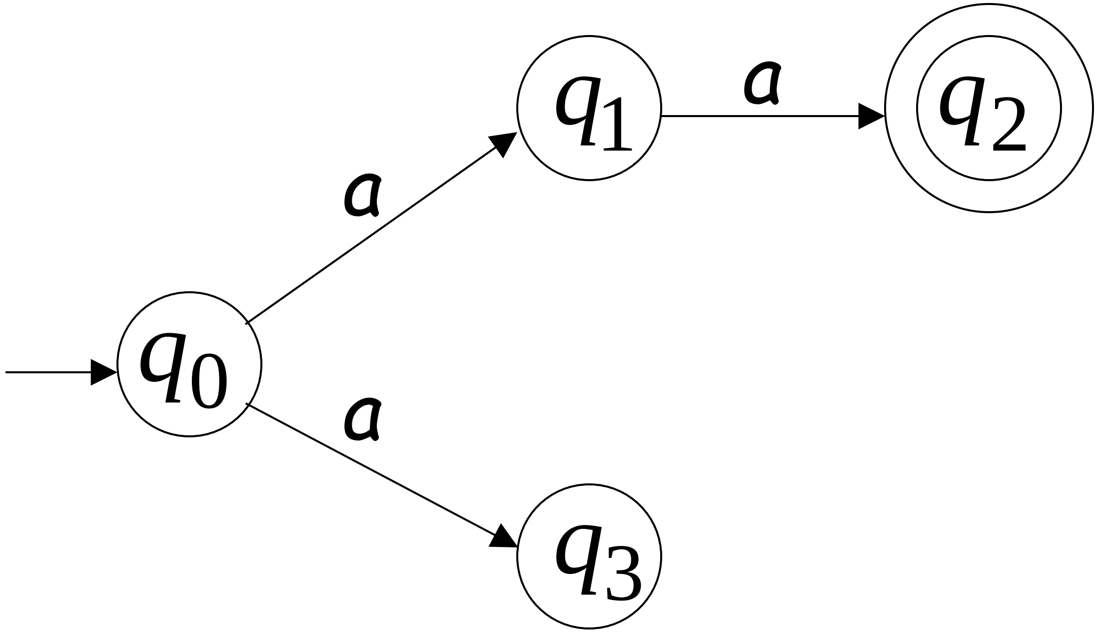
All possible computations lead to rejection aaa is rejected by the NFA:



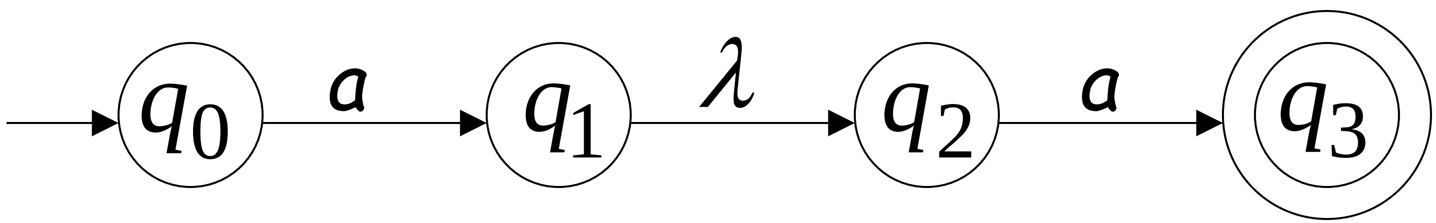


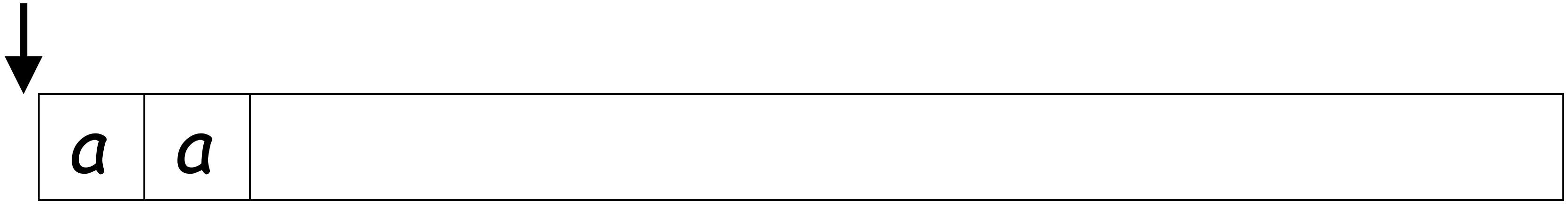
All possible computations lead to rejection

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

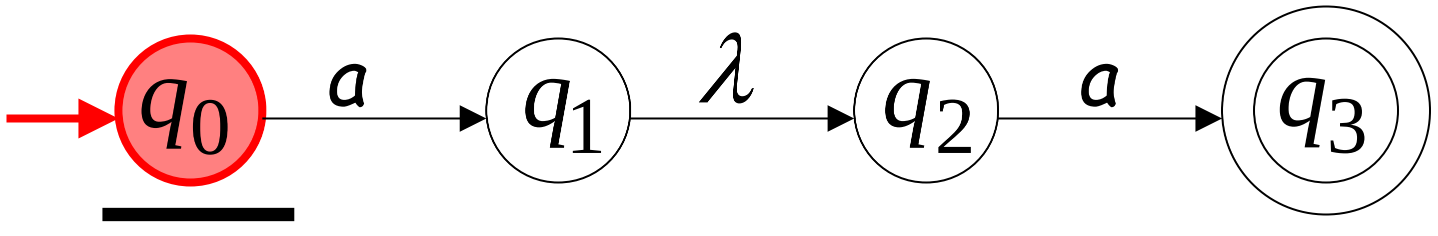


Lambda Transitions

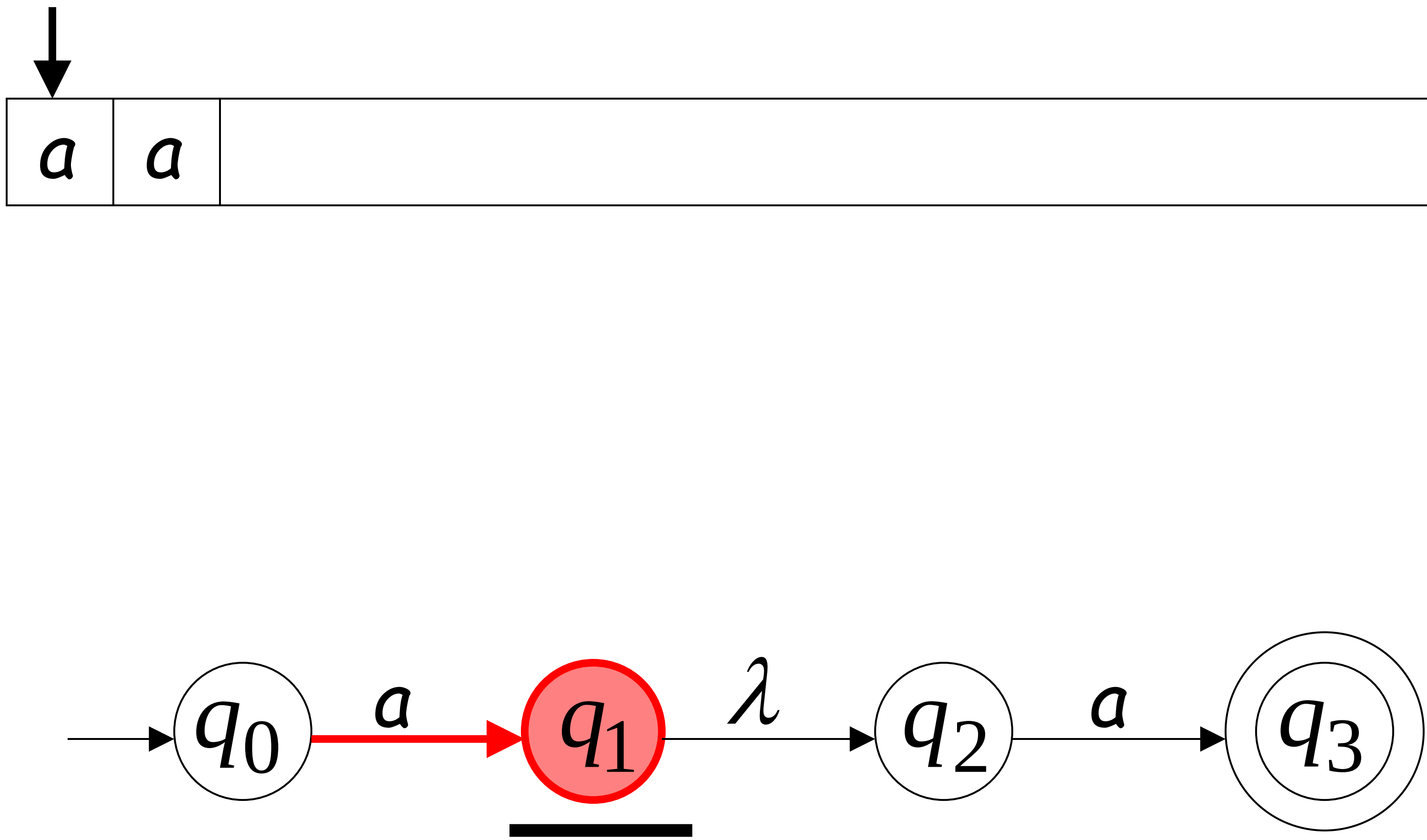




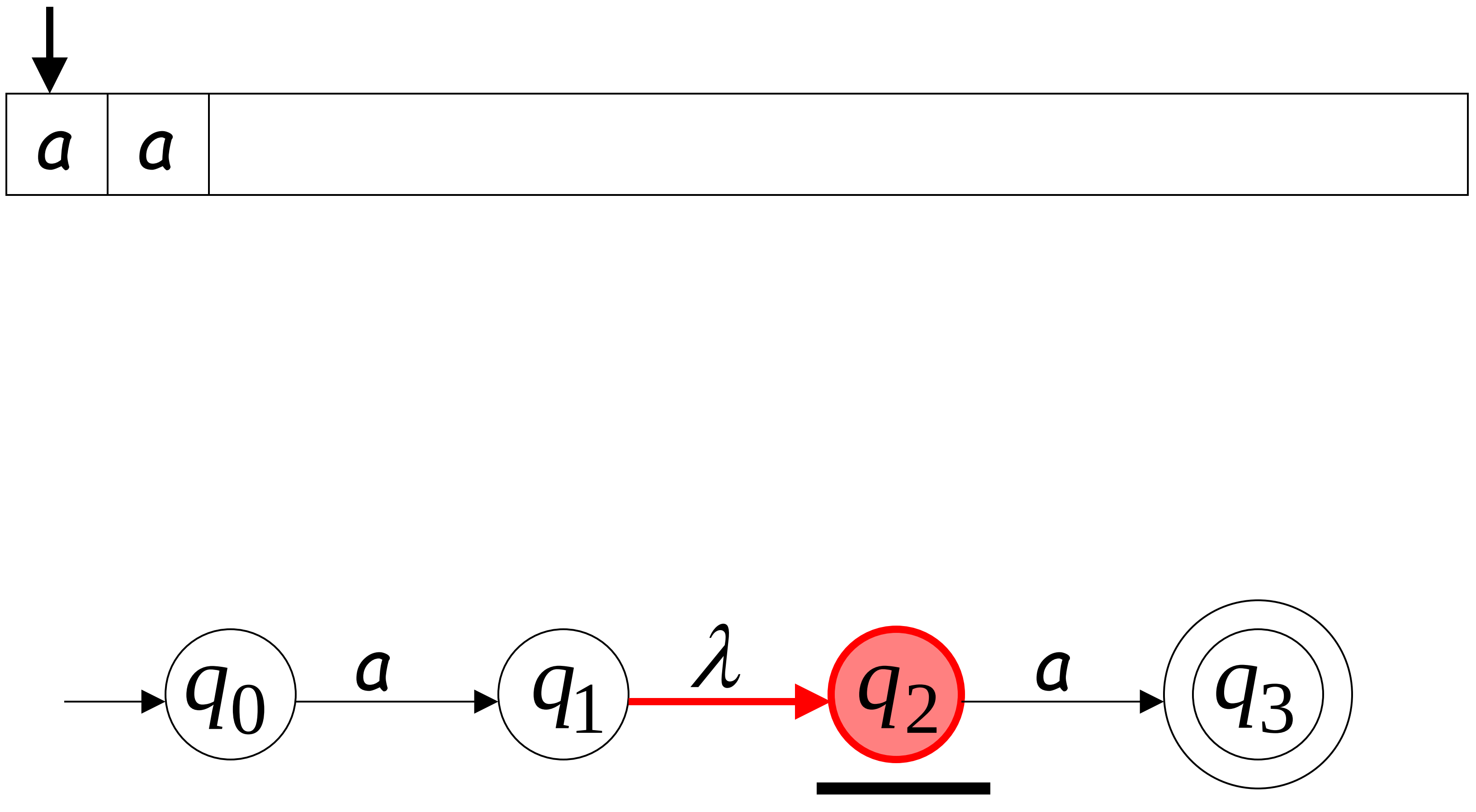
a
a



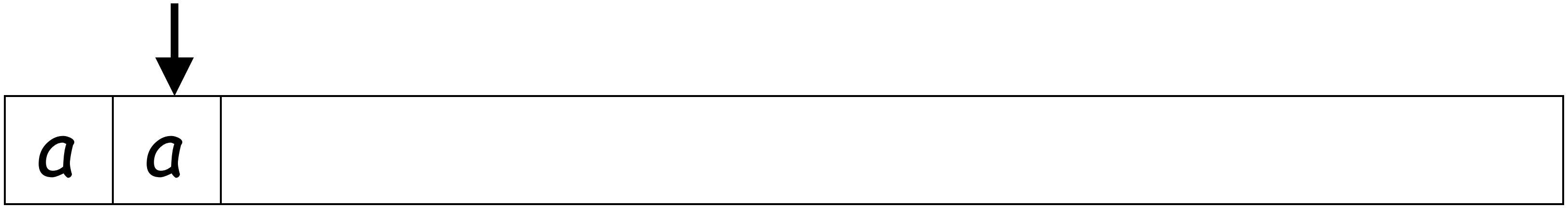
q_0
a
q_1
λ
q_2
a
q_3


a a
q0
a
q1
λ
q2
a
q3

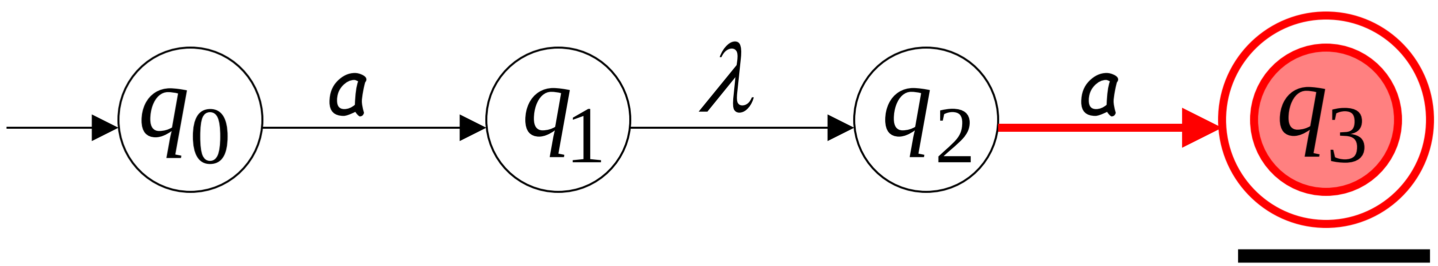
input tape head does not move



all input is consumed

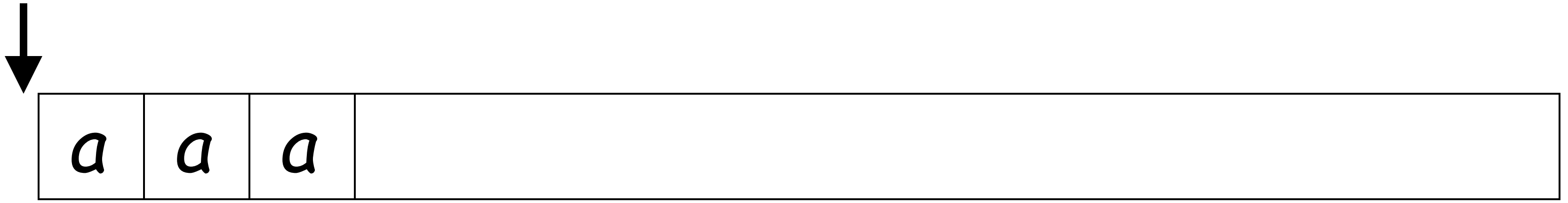


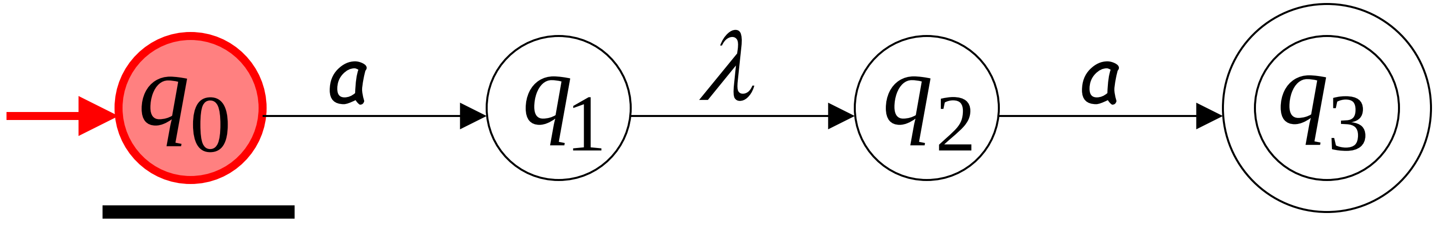
"accept"

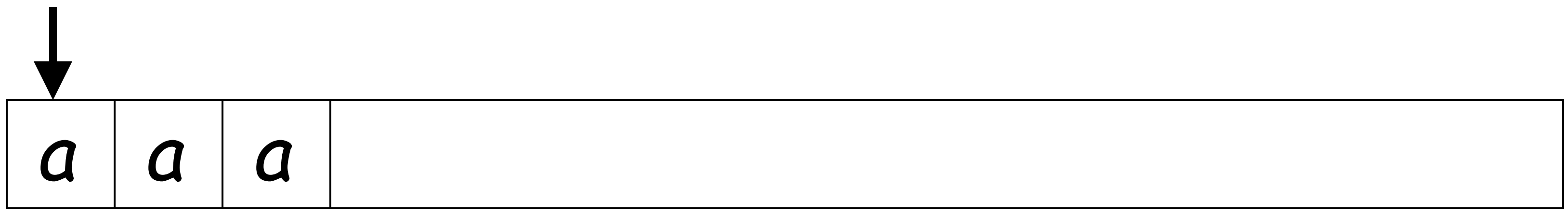


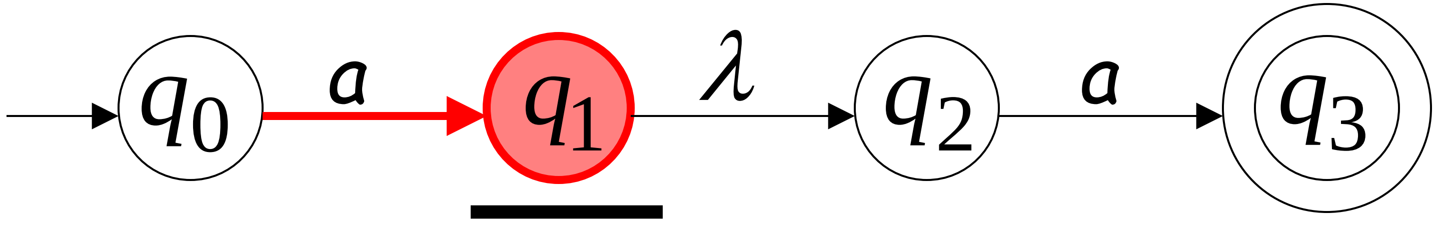
String aa is accepted

Rejection Example

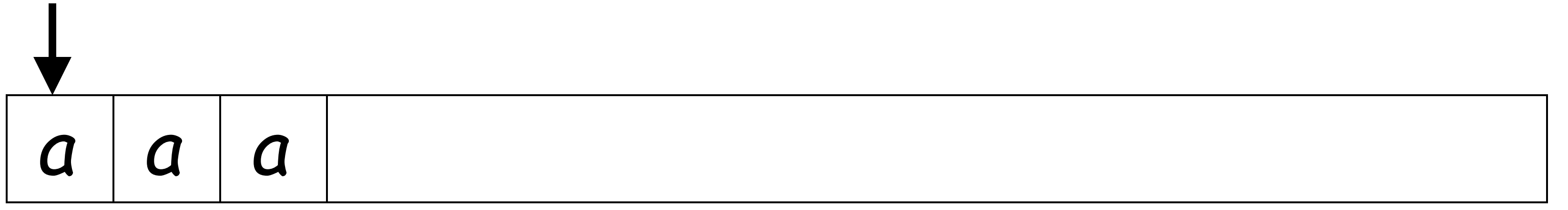


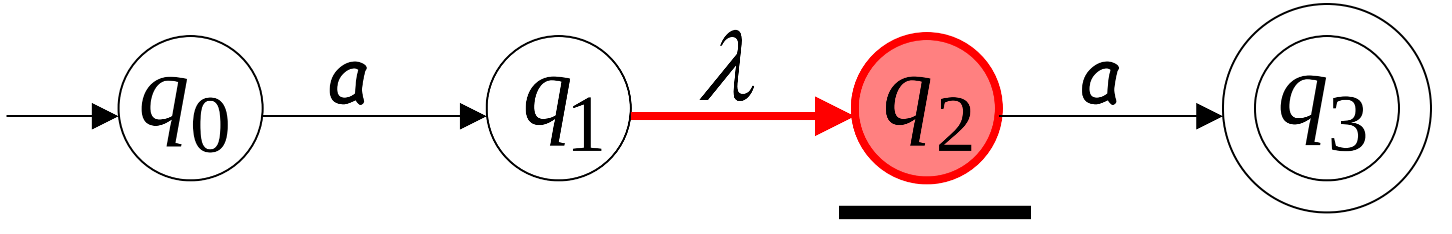



a
a
a

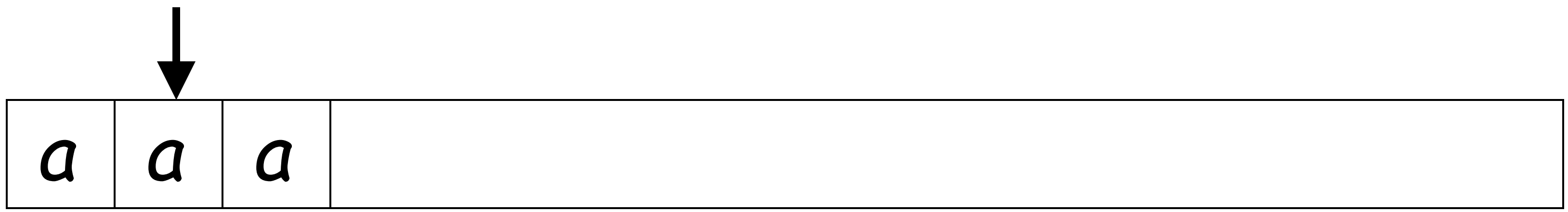

q0
a
q1
λ
q2
a
q3

(read head doesn't move)

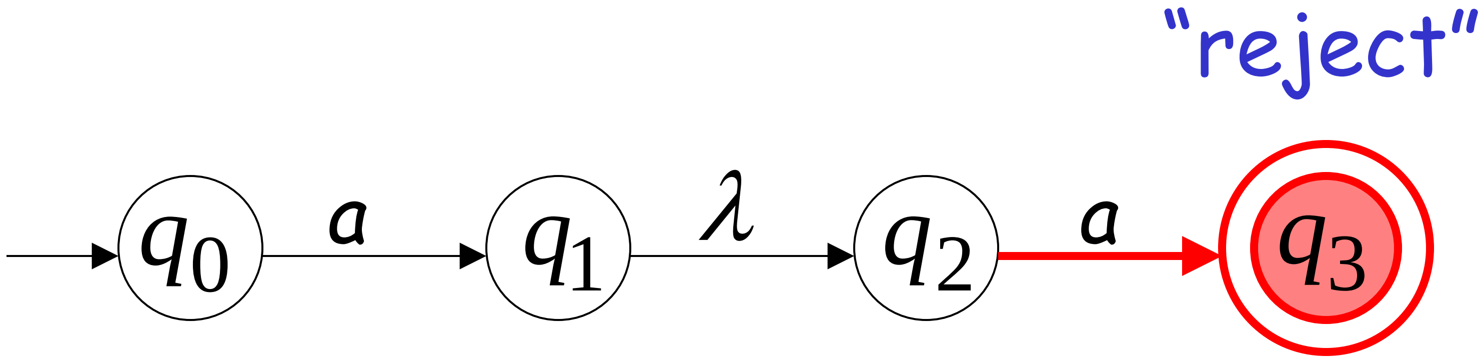




Input cannot be consumed

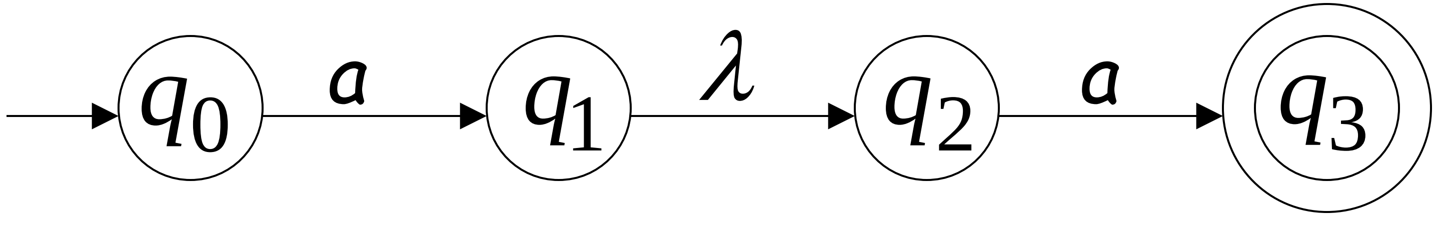


Automaton halts

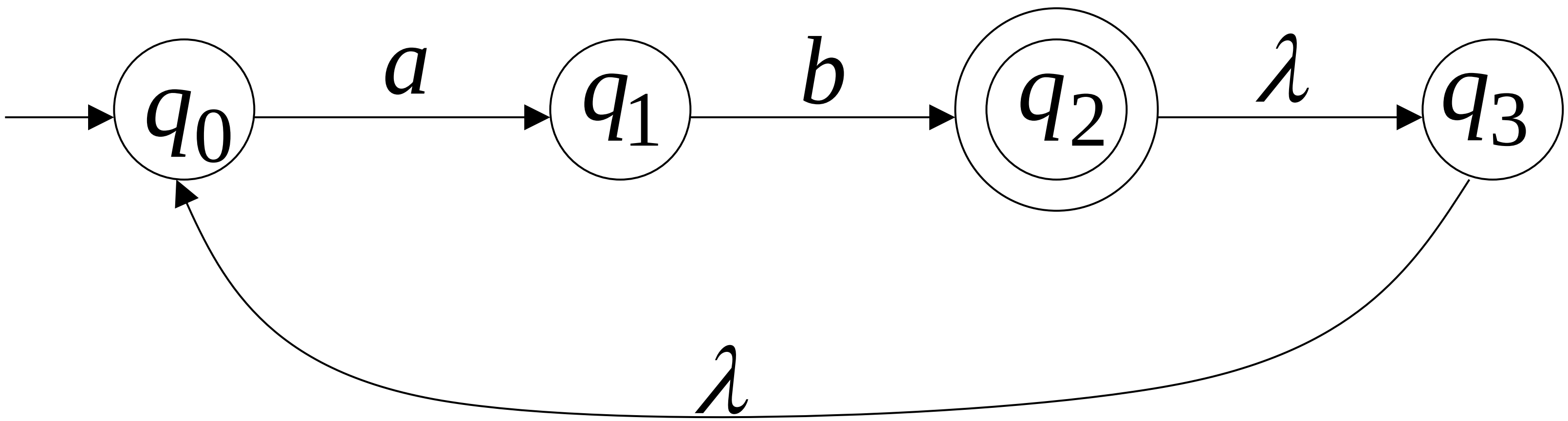


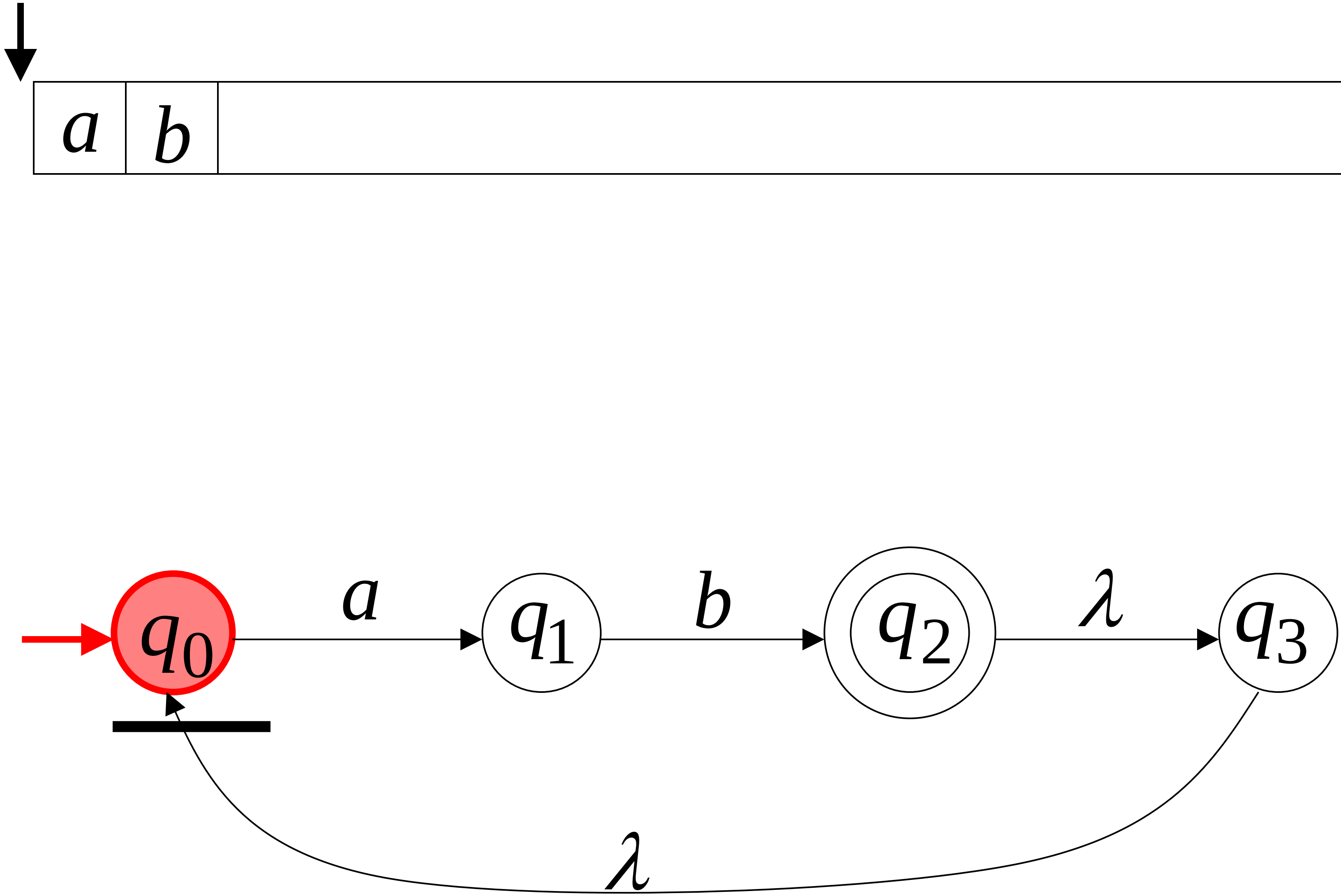
String aaa is rejected

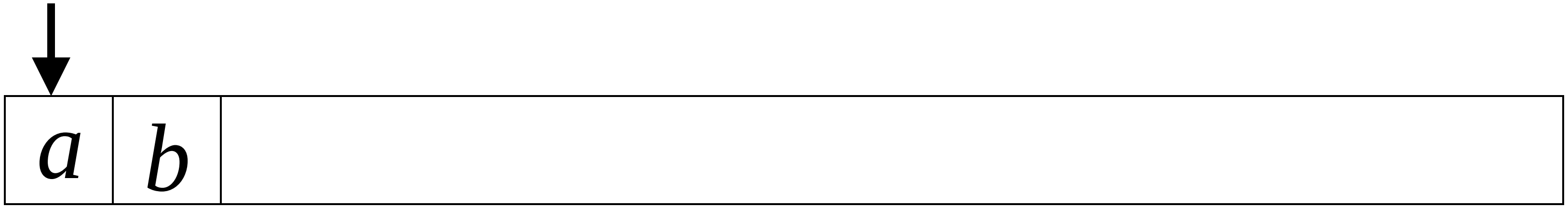
Language accepted: $L = \{aa\}$

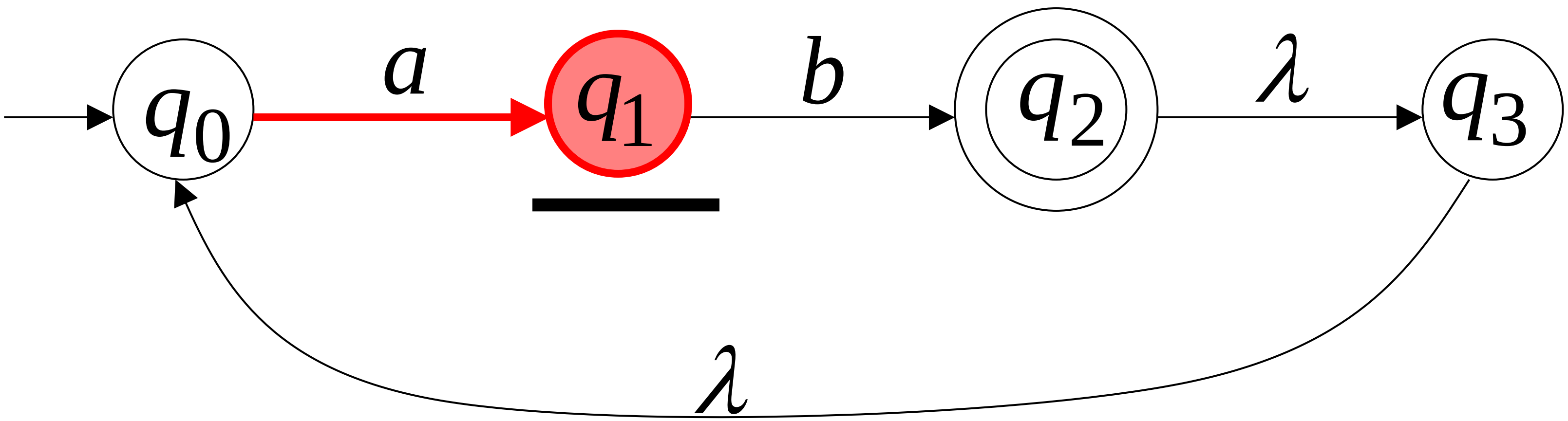


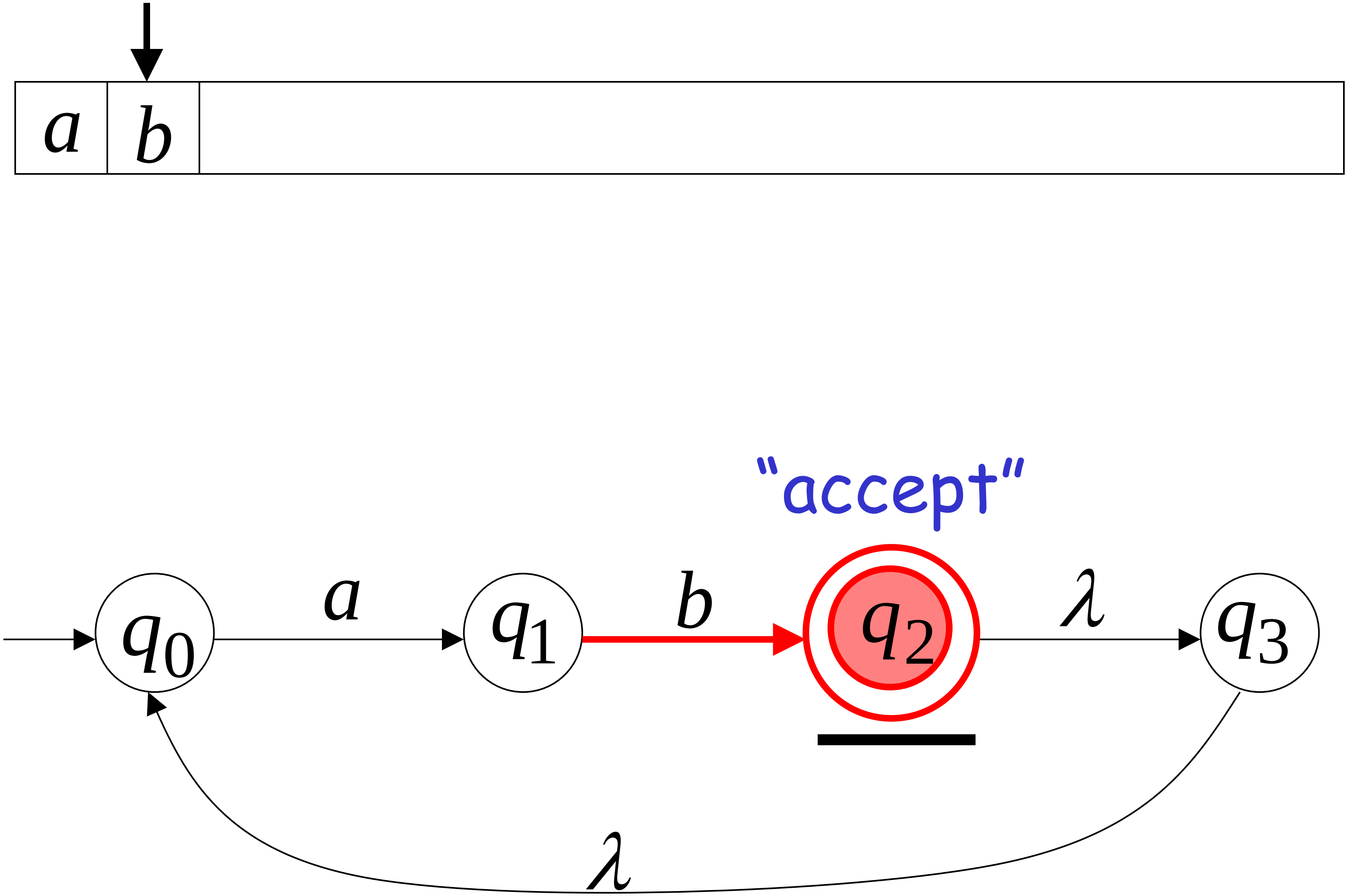
Another NFA Example




a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

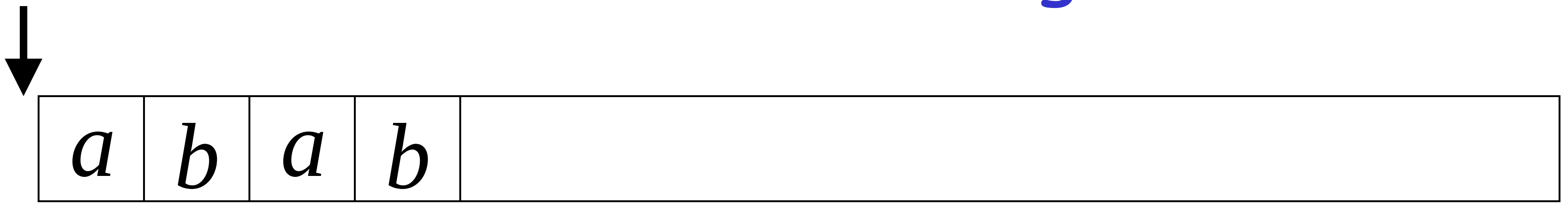

a
b

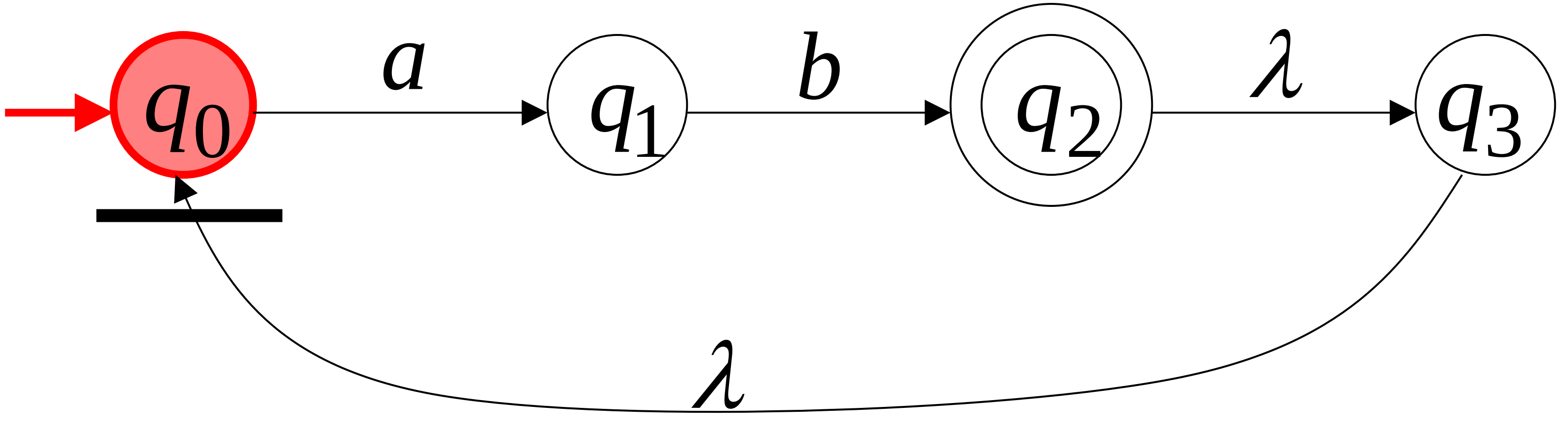

q_0
a
q_1
b
q_2
λ
q_3
λ

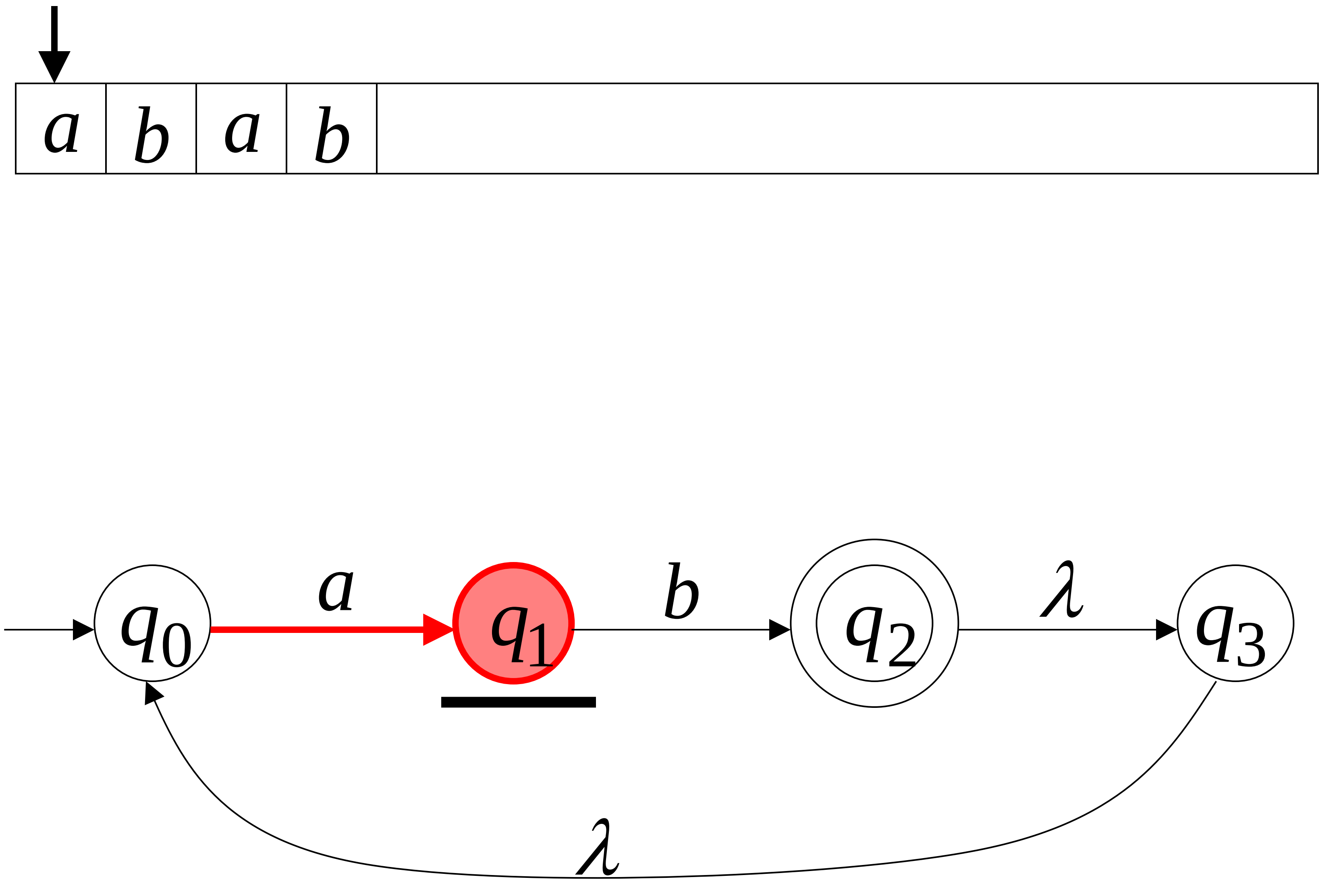


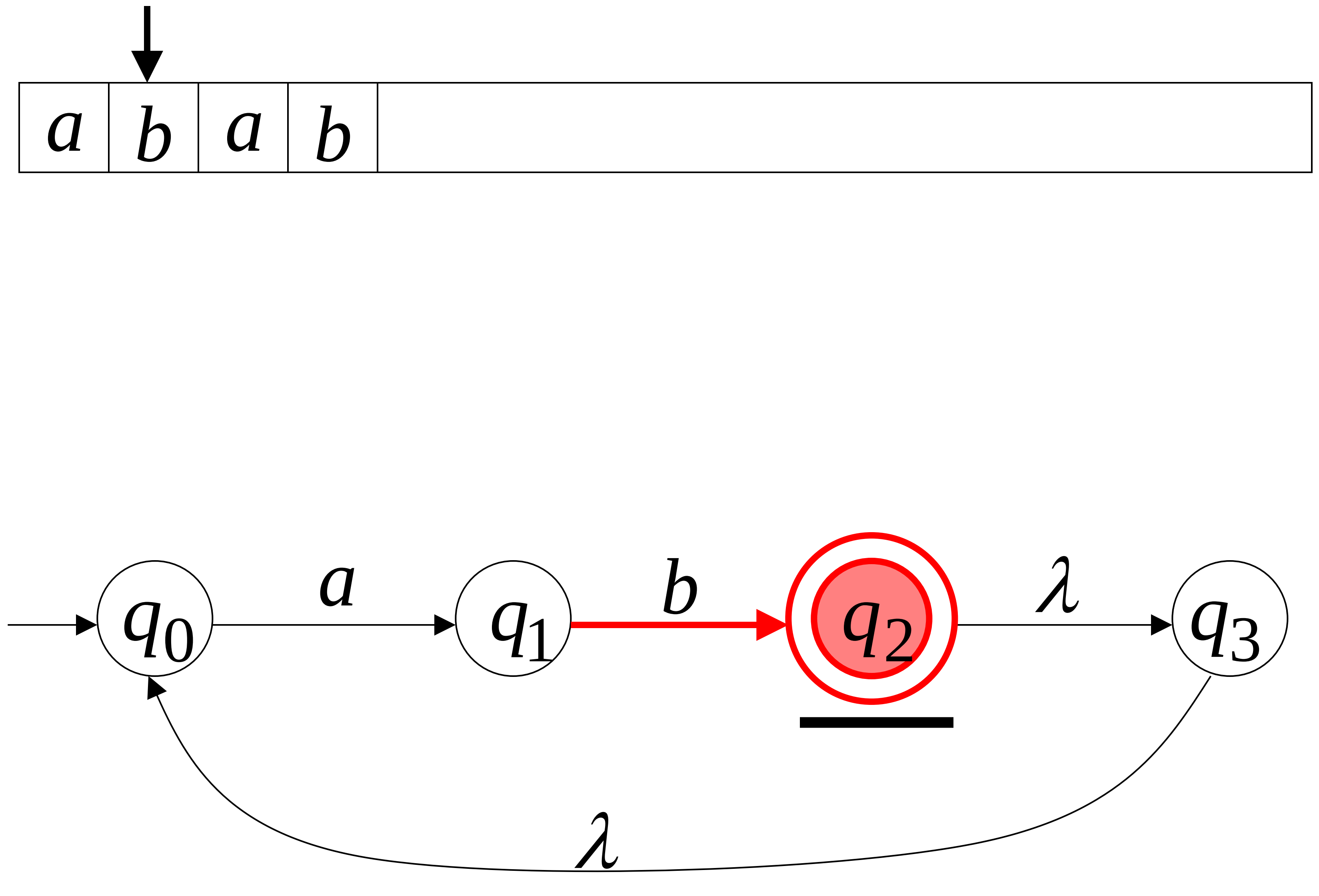
a
b
"accept"
q_0
a
q_1
b
q_2
λ
q_3
λ

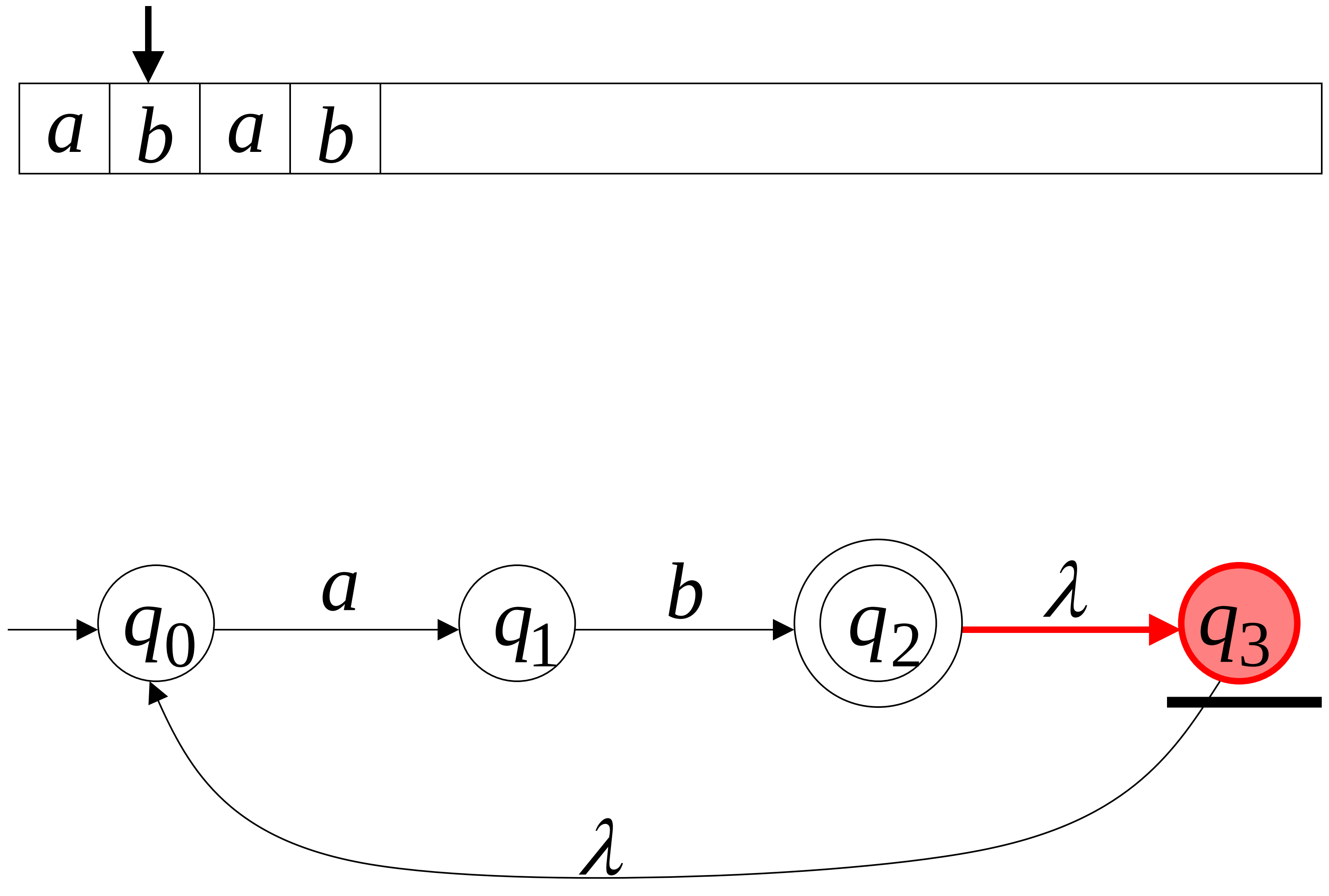
Another String



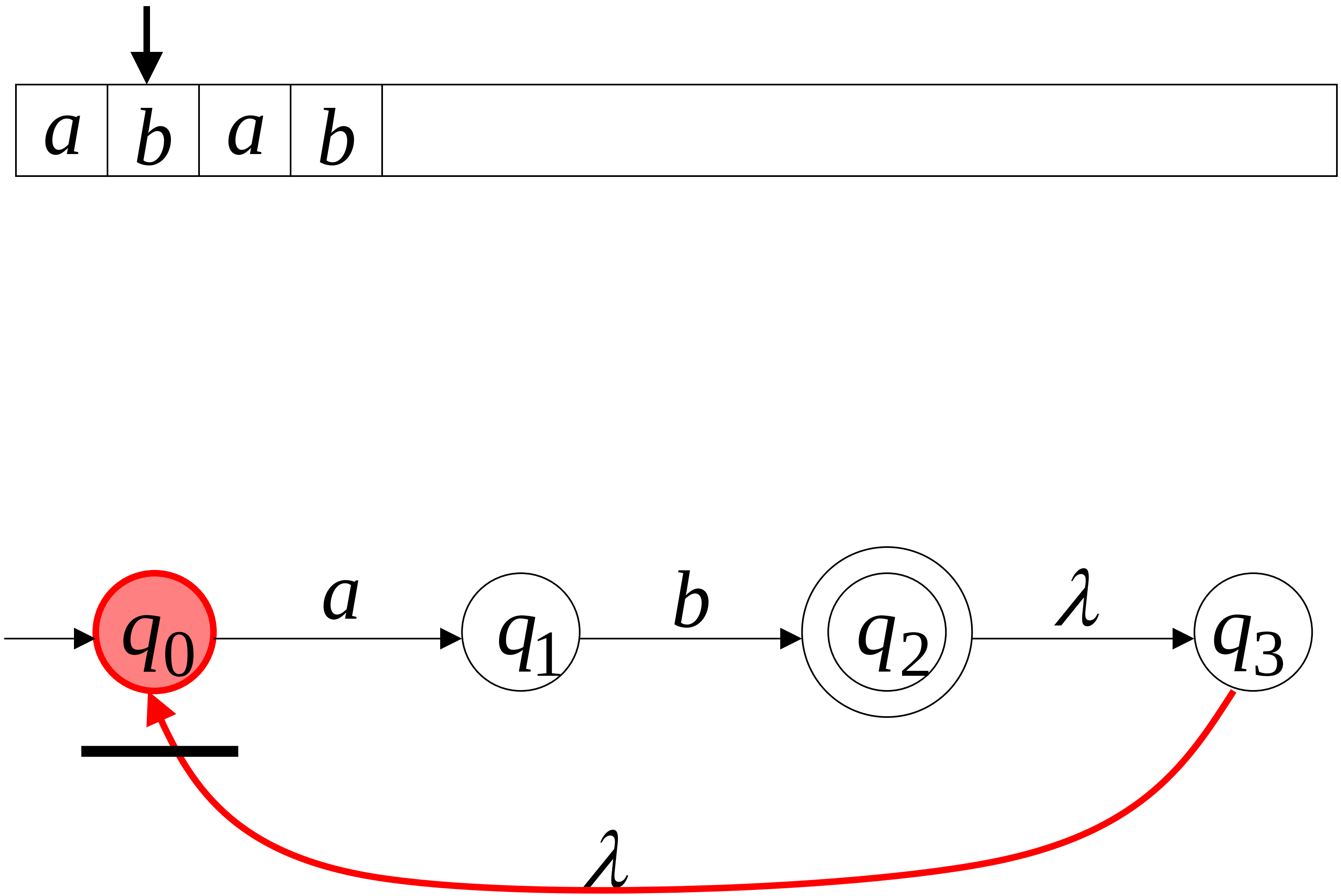



a
b
a
b
a
b
λ
λ
q0
q1
q2
q3

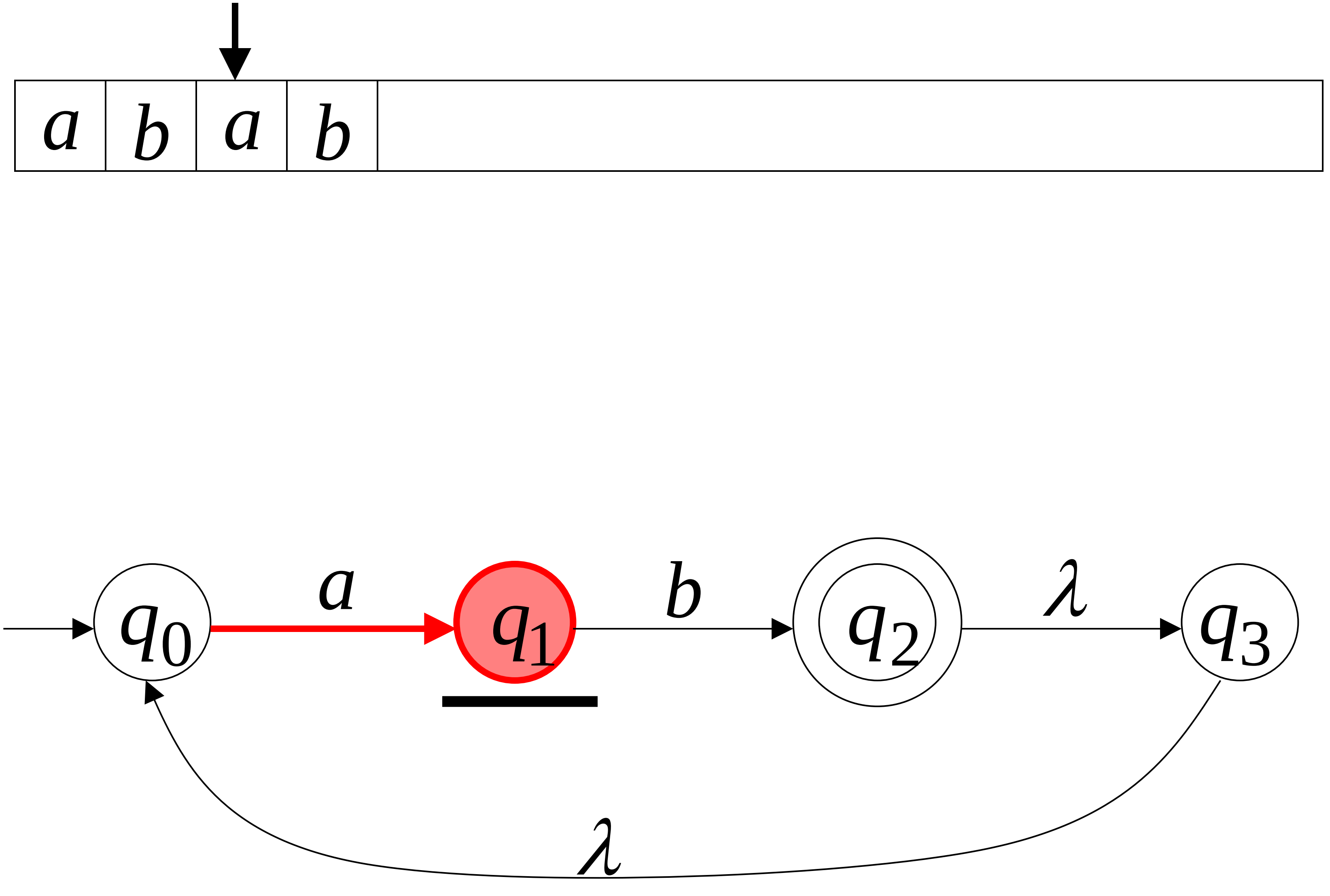

a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3

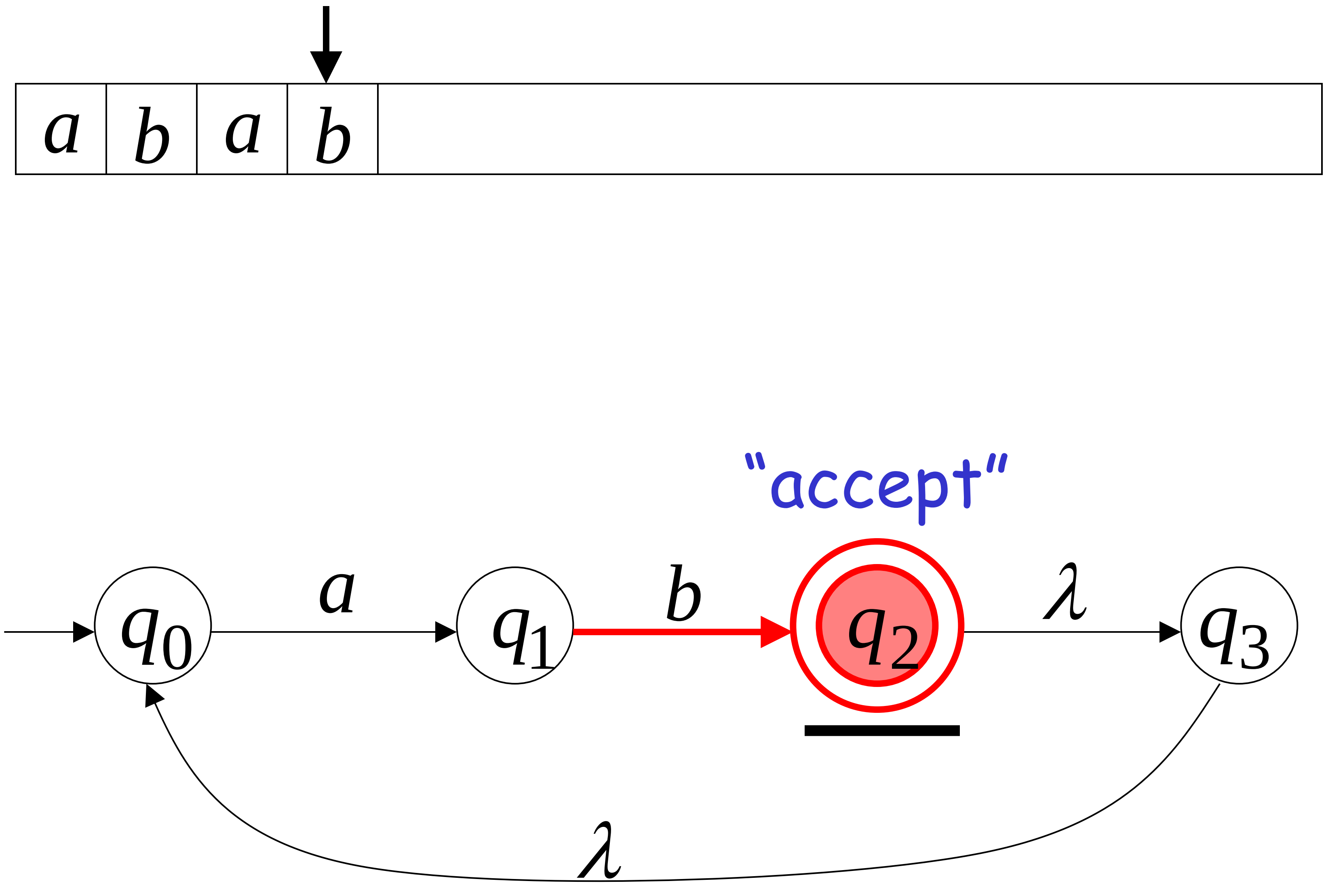


a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ



a
b
a
b
q0
a
q1
b
q2
λ
q3
λ

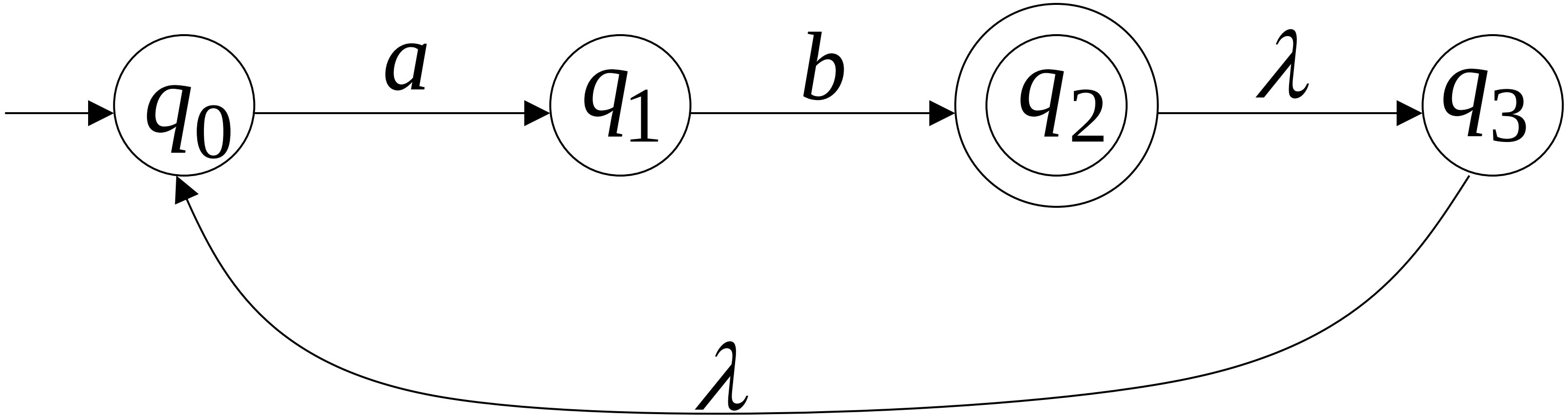

a
b
a
b
a
b
λ
λ
q0
q1
q2
q3


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

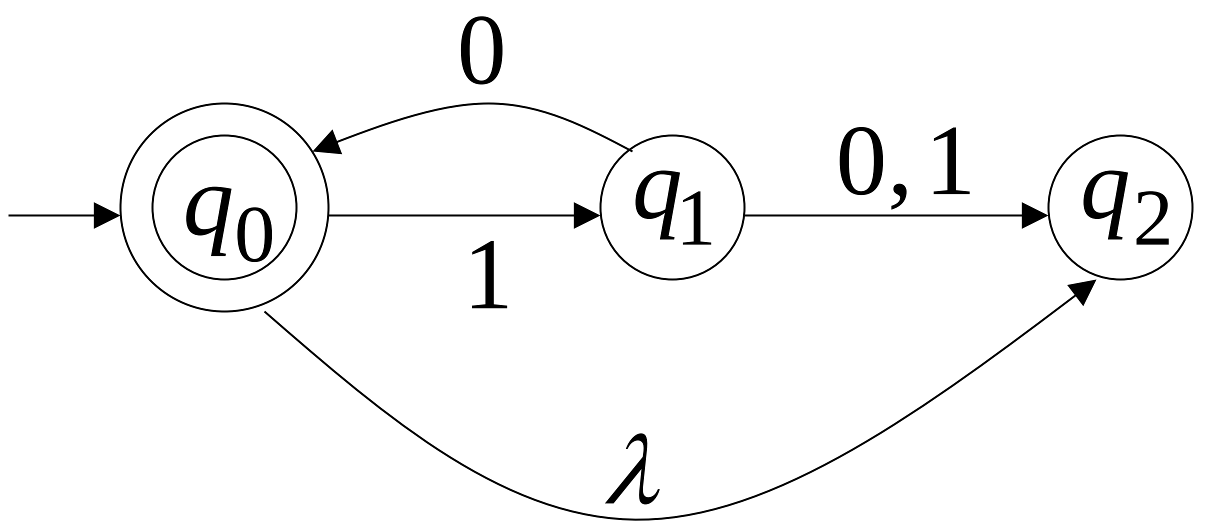
Language accepted

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

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



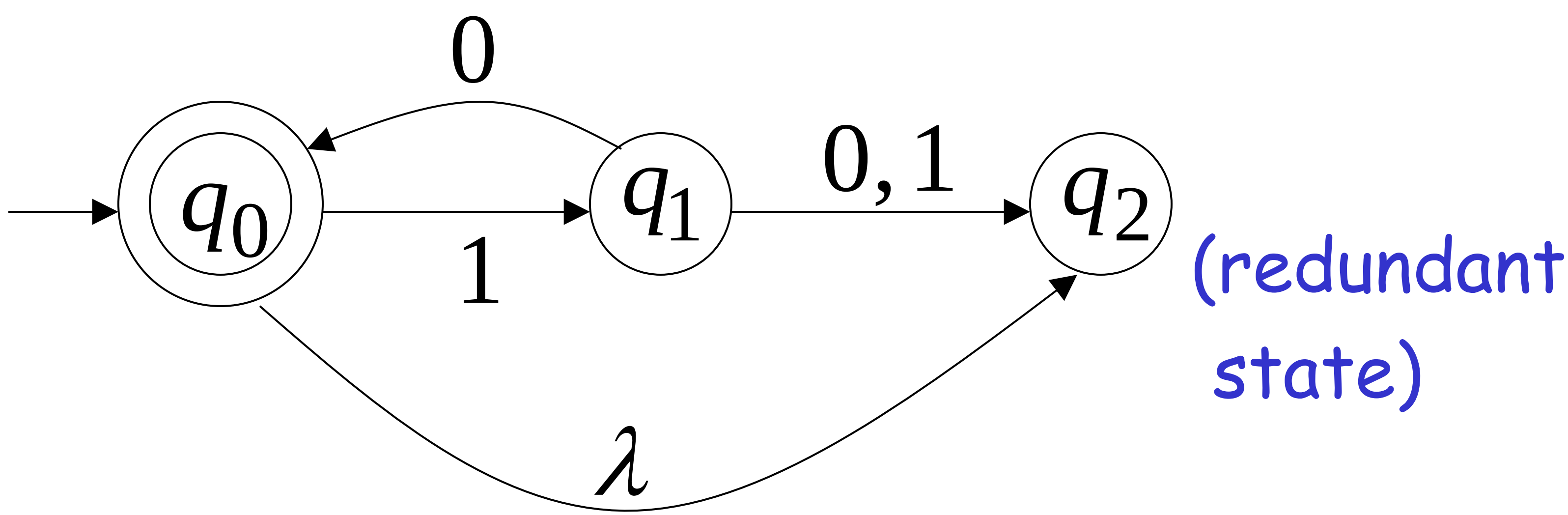
Another NFA Example



Language accepted

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

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

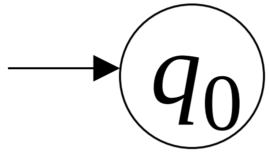


Remarks:

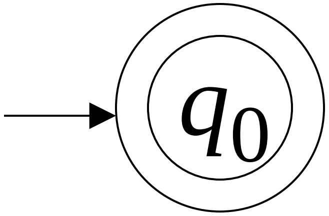
- The λ symbol never appears on the input tape

- Simple automata:

M_1



M_2

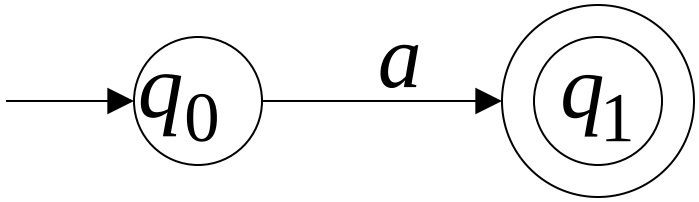


$L(M_1) = \{\}$

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

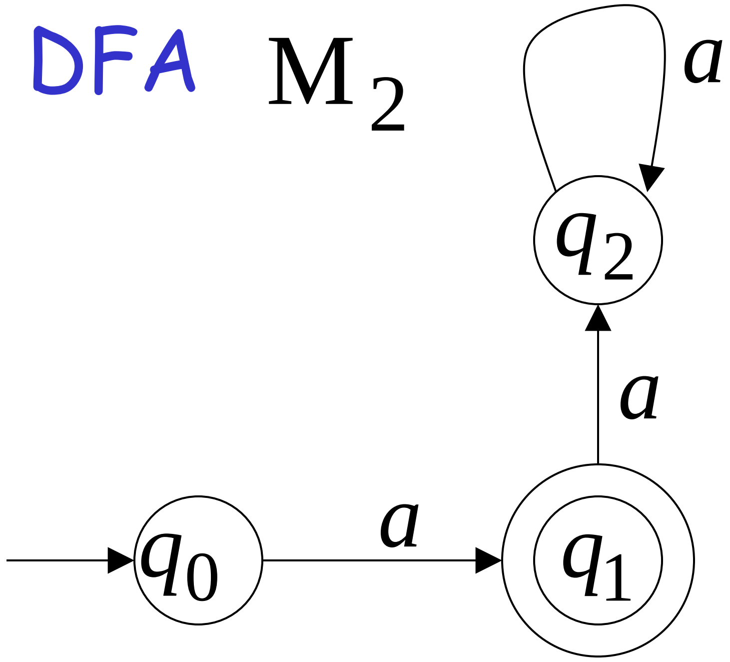
• NFAs are interesting because we can express languages easier than DFAs

NFA M_1



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

DFA M_2



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

Formal Definition of NFAs

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

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

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

$\delta:$ Transition function

$q_0:$ Initial state

$F:$ Accepting states

Definition

Given an alphabet Σ, we define

$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\},$$

which is the set of all strings over Σ of length 0 or 1.

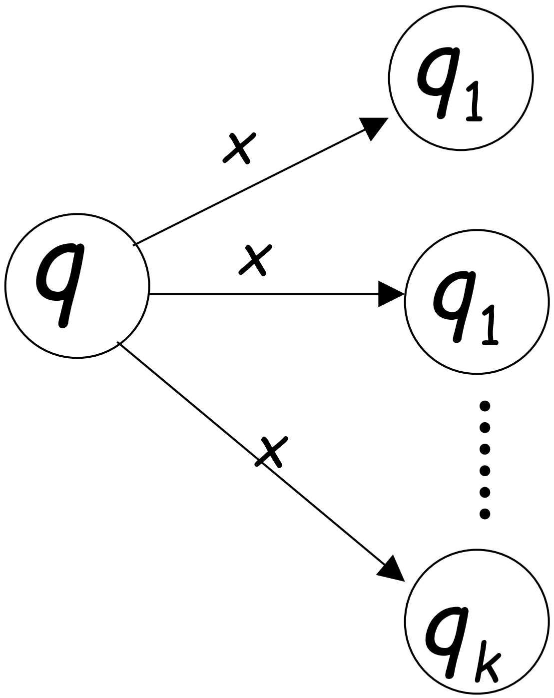
Definition

A *nondeterministic finite automaton with ε-transitions* (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set (the set of states),
- Σ is a finite set (the alphabet),
- $\delta : Q \times \Sigma_\varepsilon \to 2^Q$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accepting states.

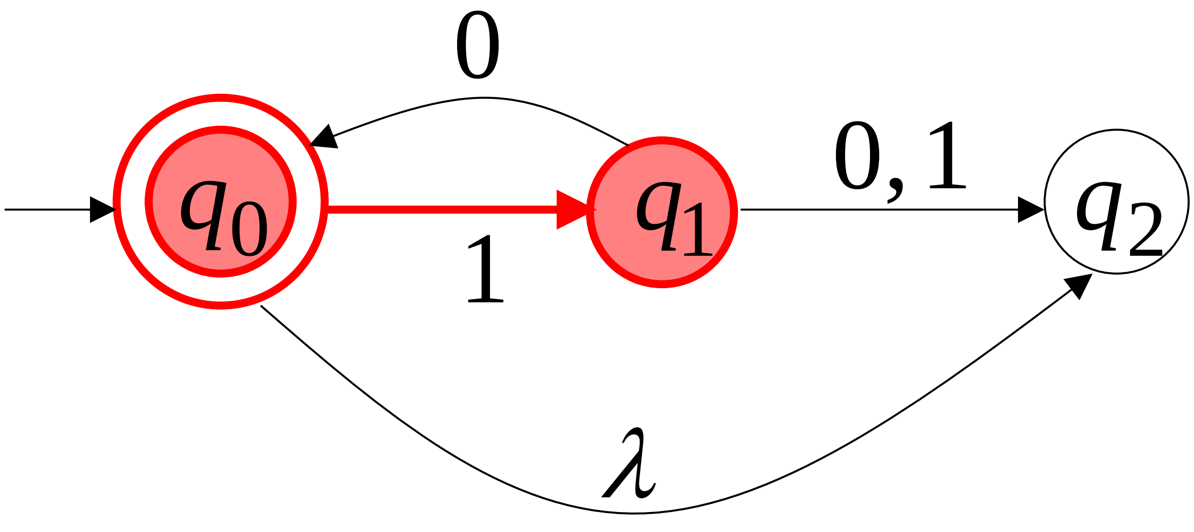
Transition Function δ

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

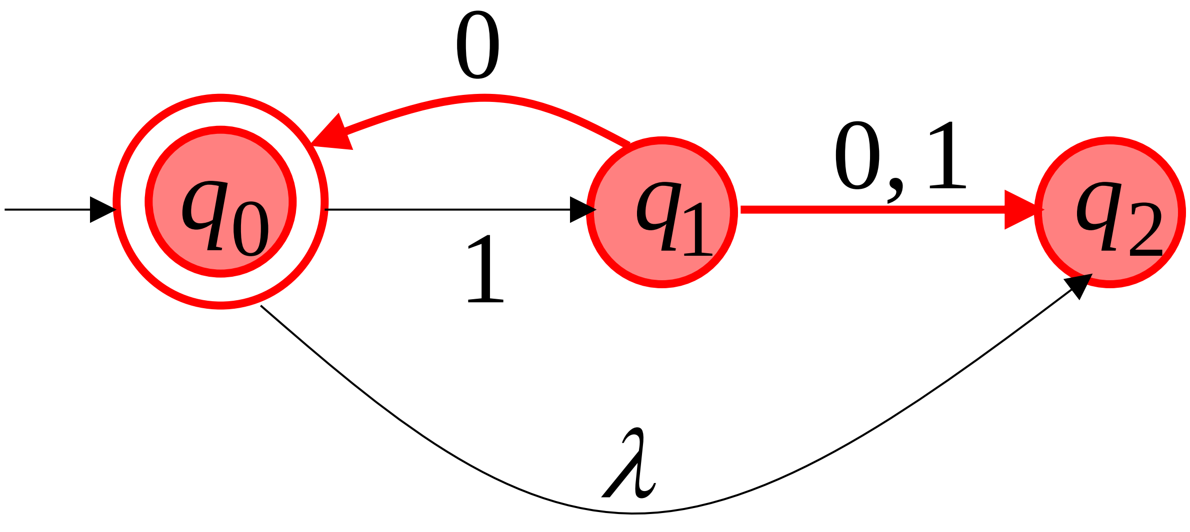


resulting states with following one transition with symbol x

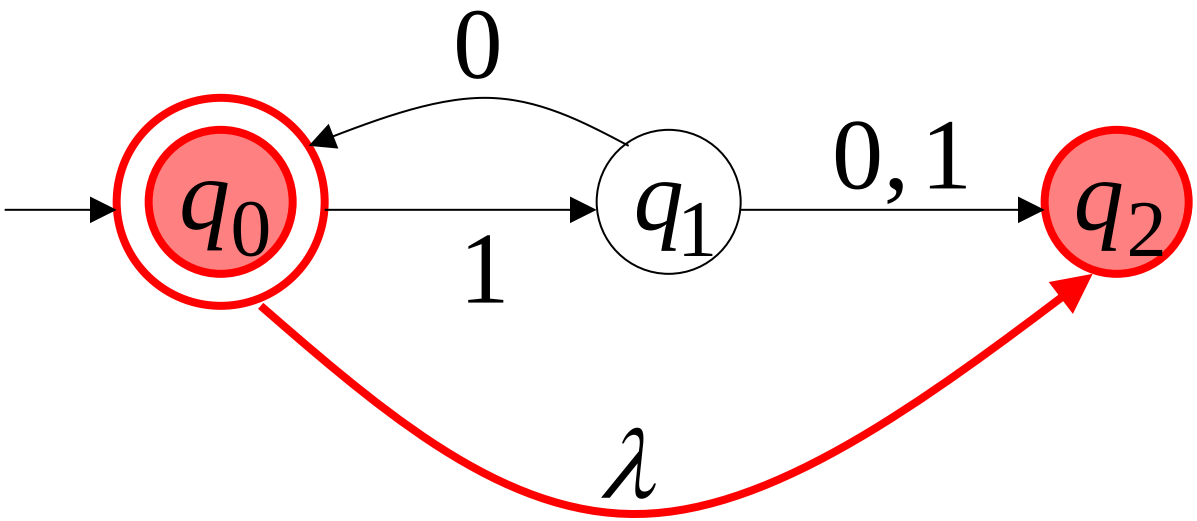
$$\delta(q_0, 1) = \{q_1\}$$



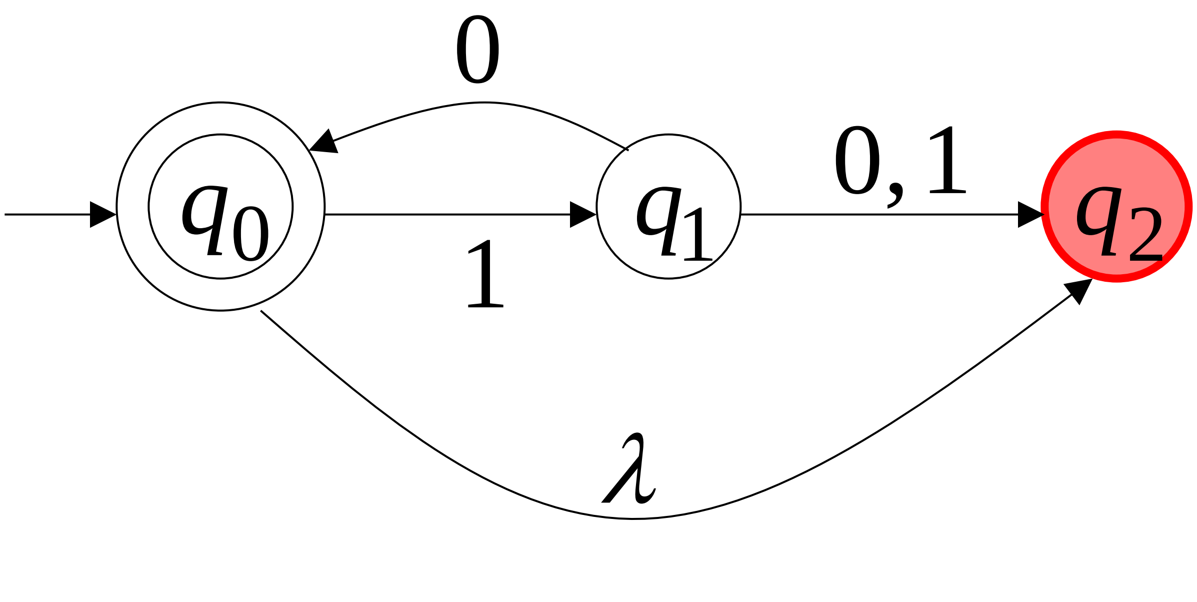
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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



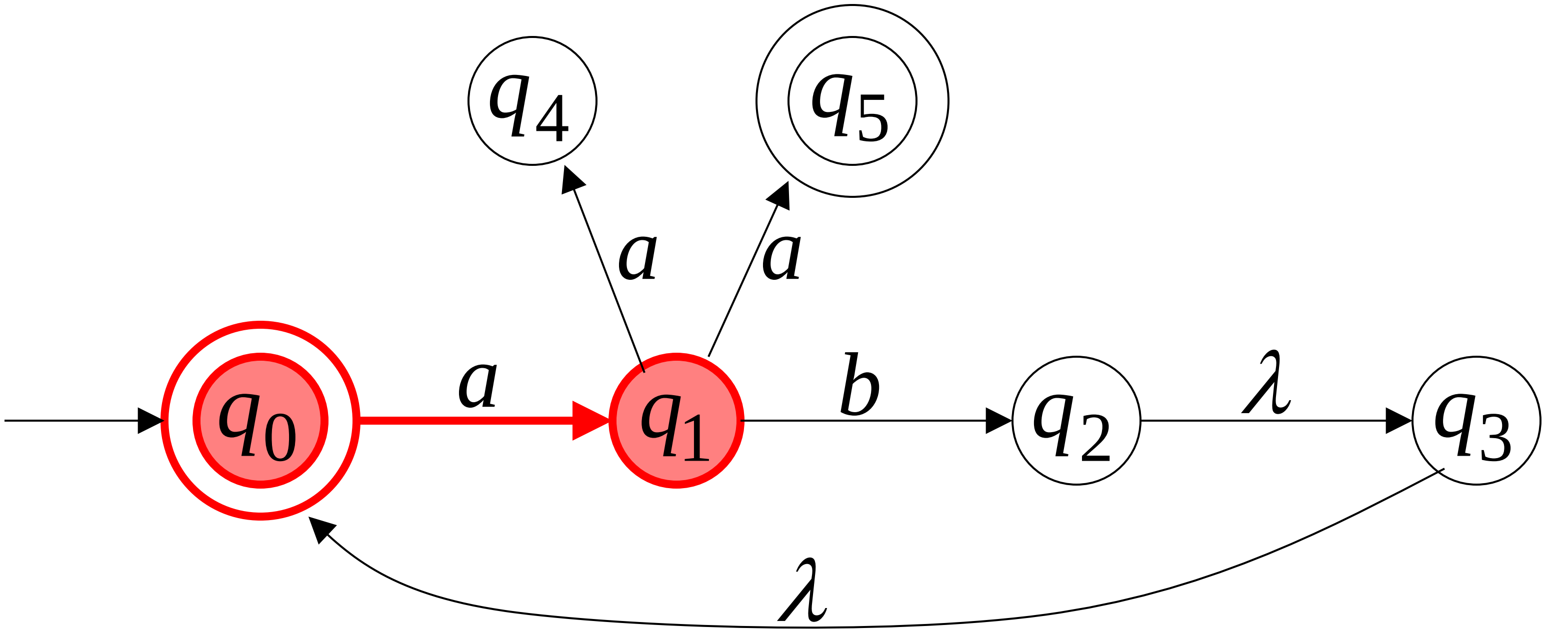
$$\delta(q_2, 1) = \varnothing$$



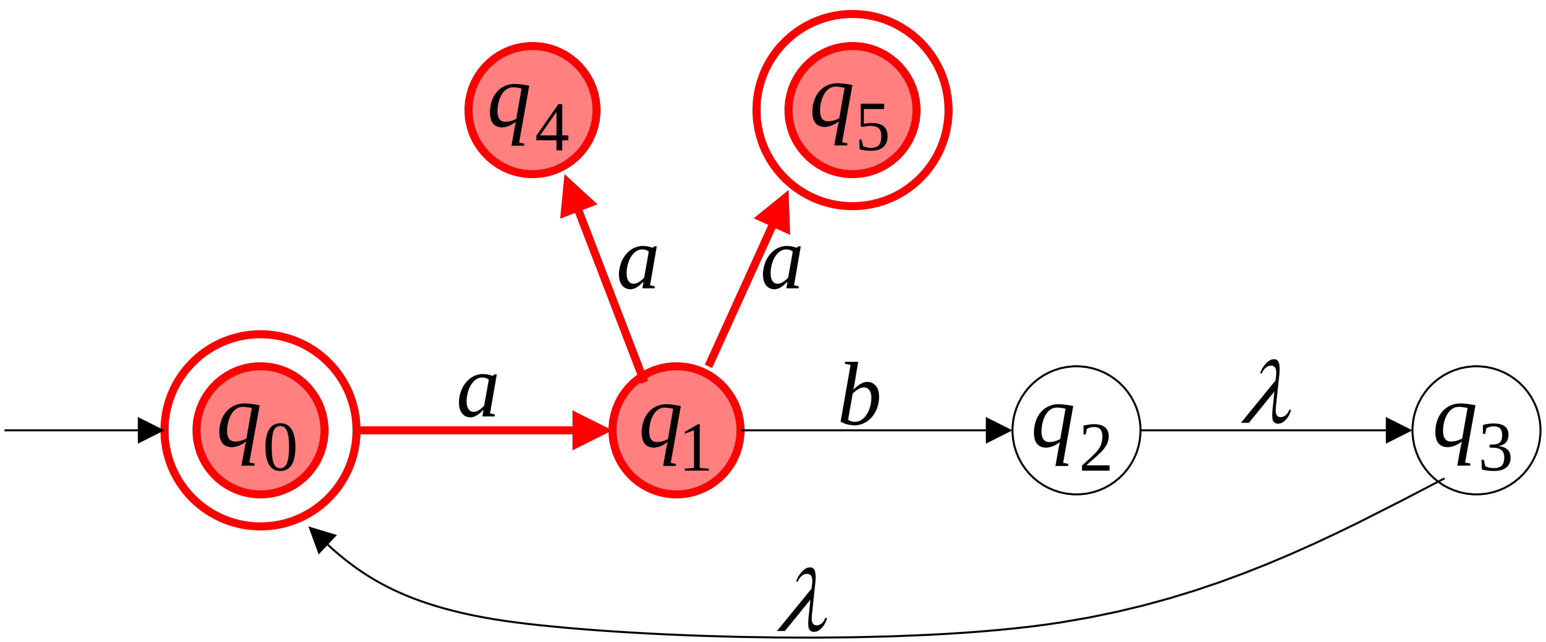
Extended Transition Function δ^*

Same with δ but applied on strings

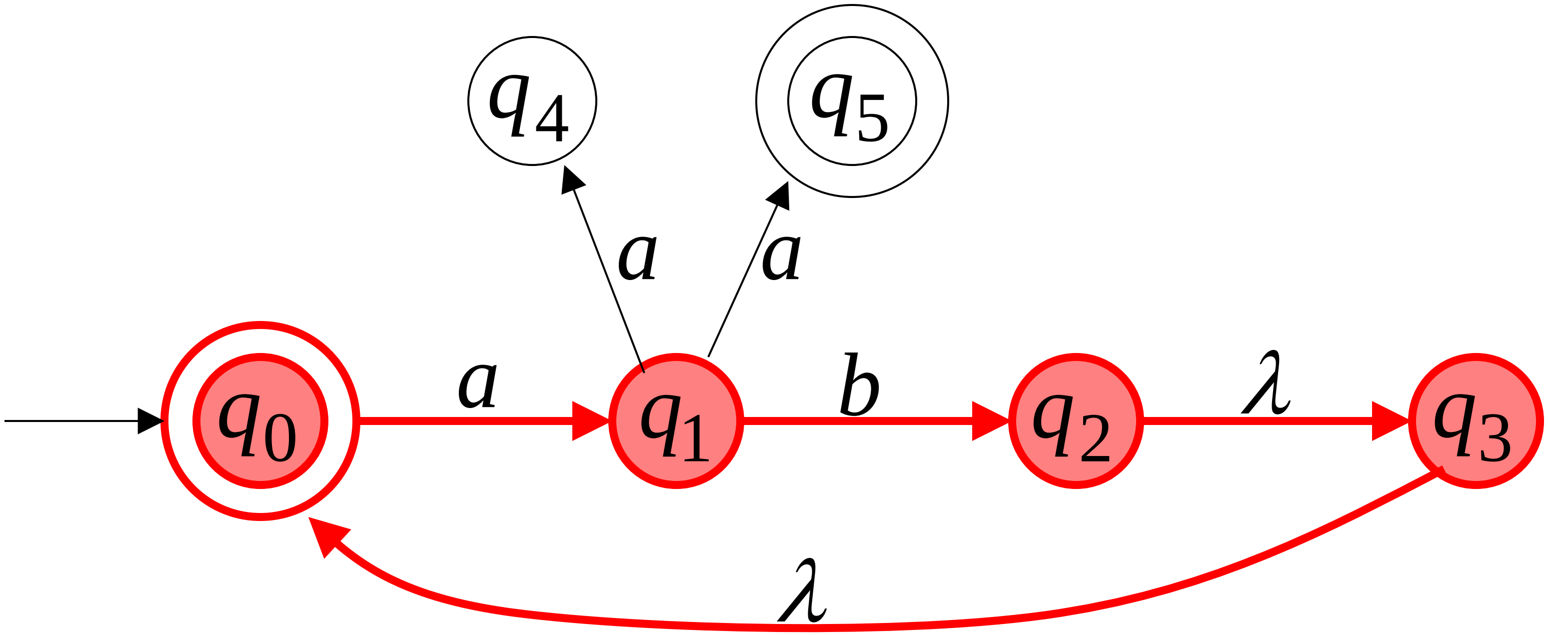
$$\delta^*(q_0, a) = \{q_1\}$$



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



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



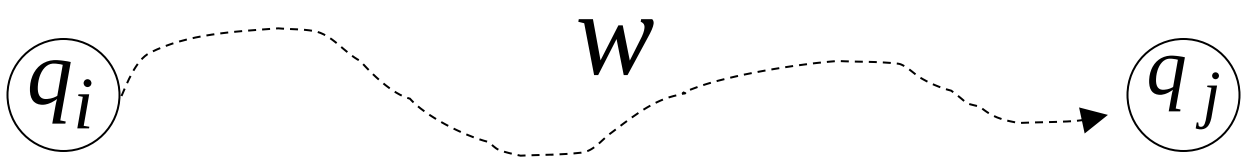
Special case:

for any state q

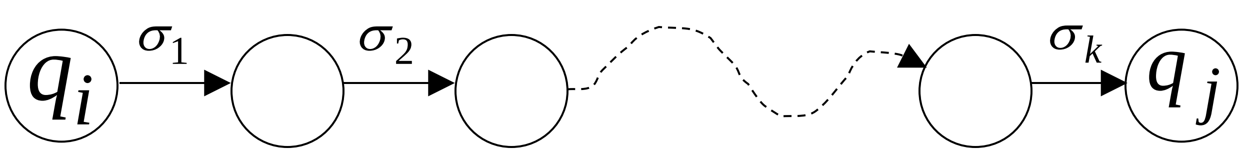
$$q \in \delta^*(q, \lambda)$$

In general

$q_j \in \delta^*(q_i, w)$: there is a walk from q_i to q_j with label w



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



The Language of an NFA M

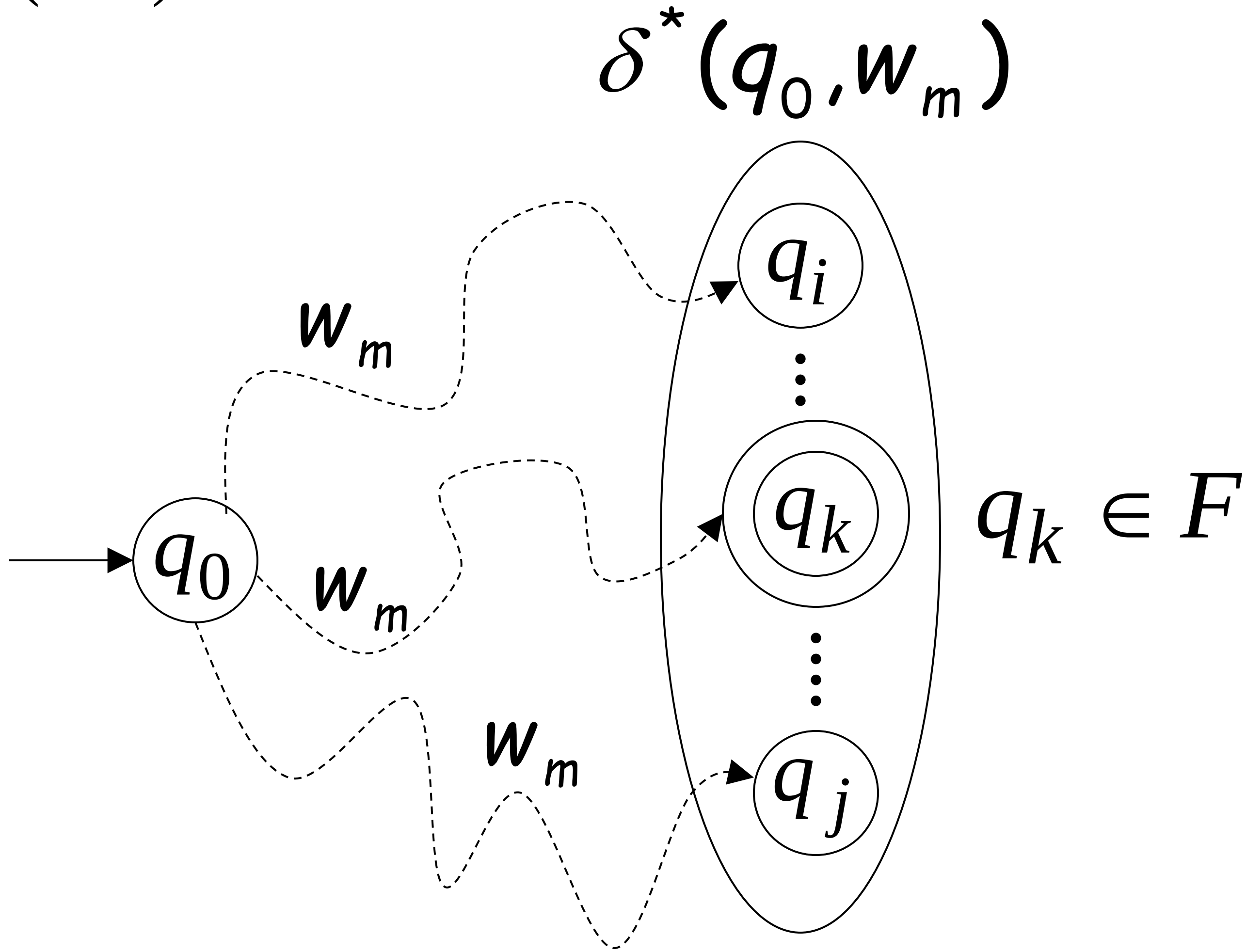
The language accepted by M is:

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

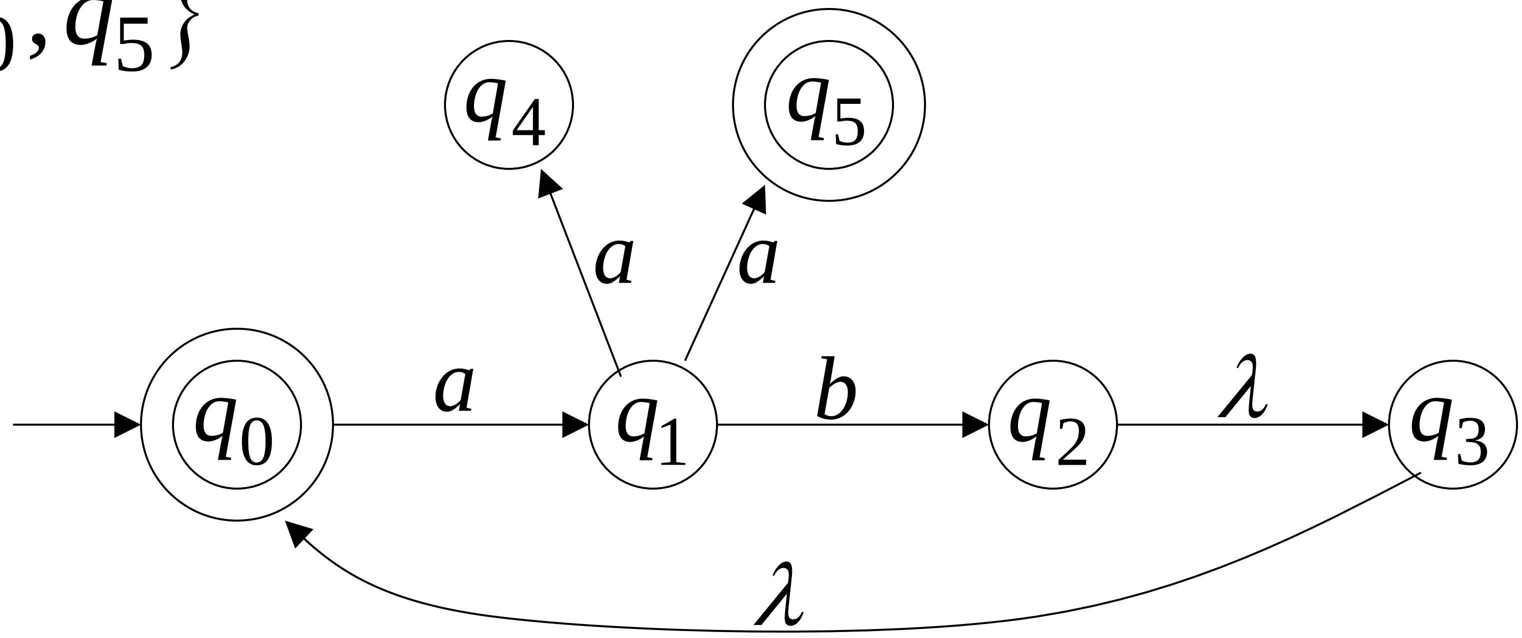
where $\delta^*(q_0, w_m) = \{q_i, \ldots, q_k, \ldots, q_j\}$

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

$w_m \in L(M)$



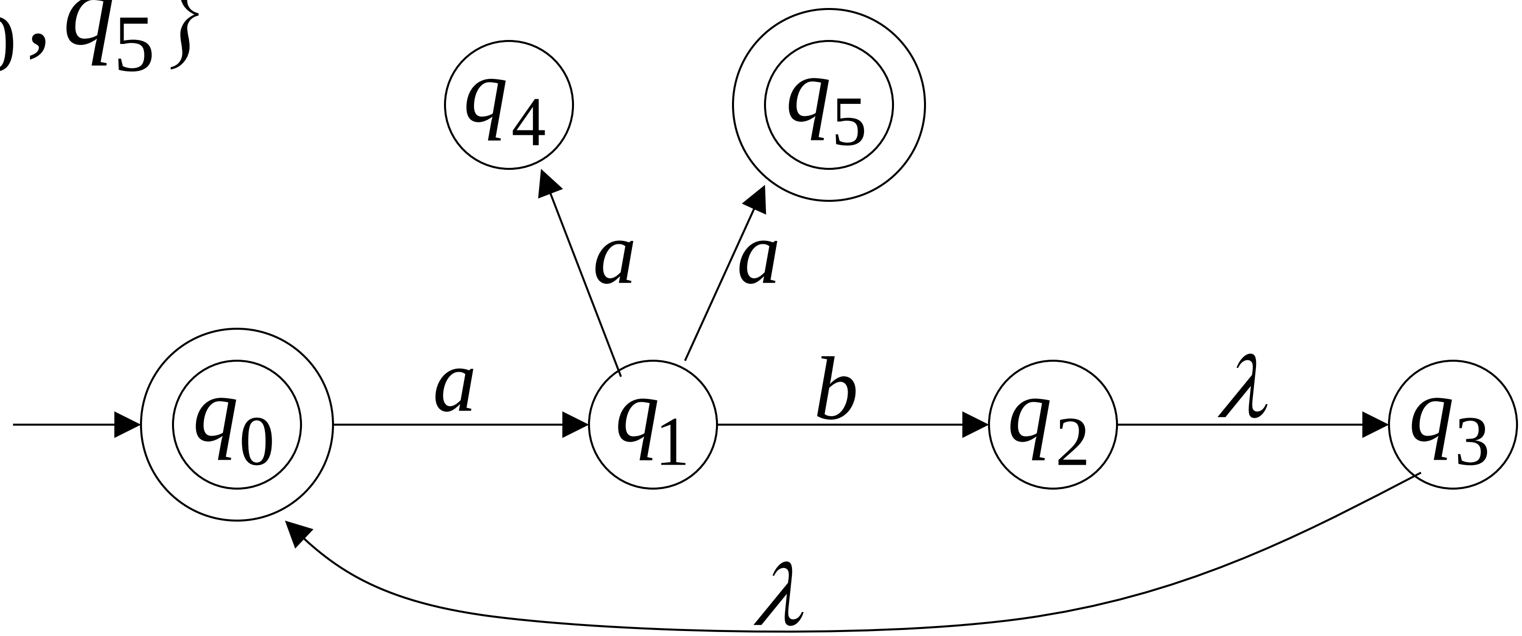
$$F = \{q_0, q_5\}$$



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

$\in F$

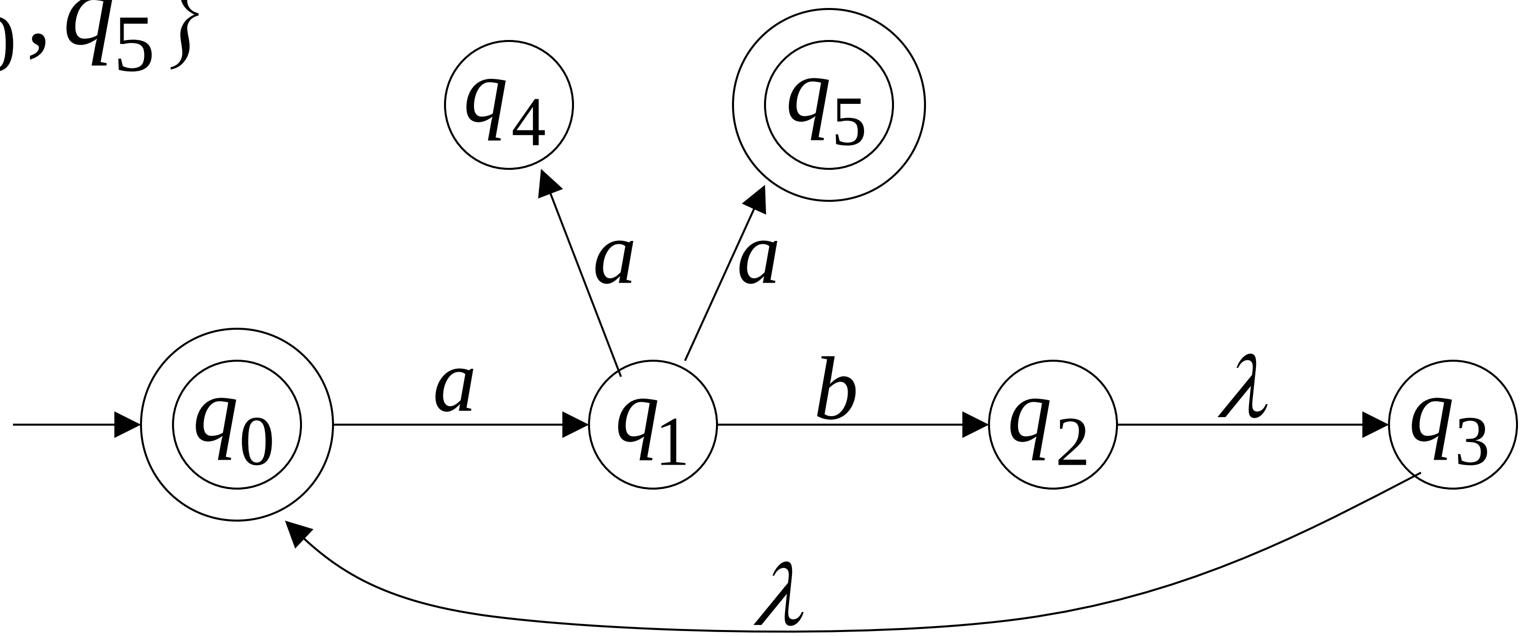
$F = \{q_0, q_5\}$



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

$\in F$

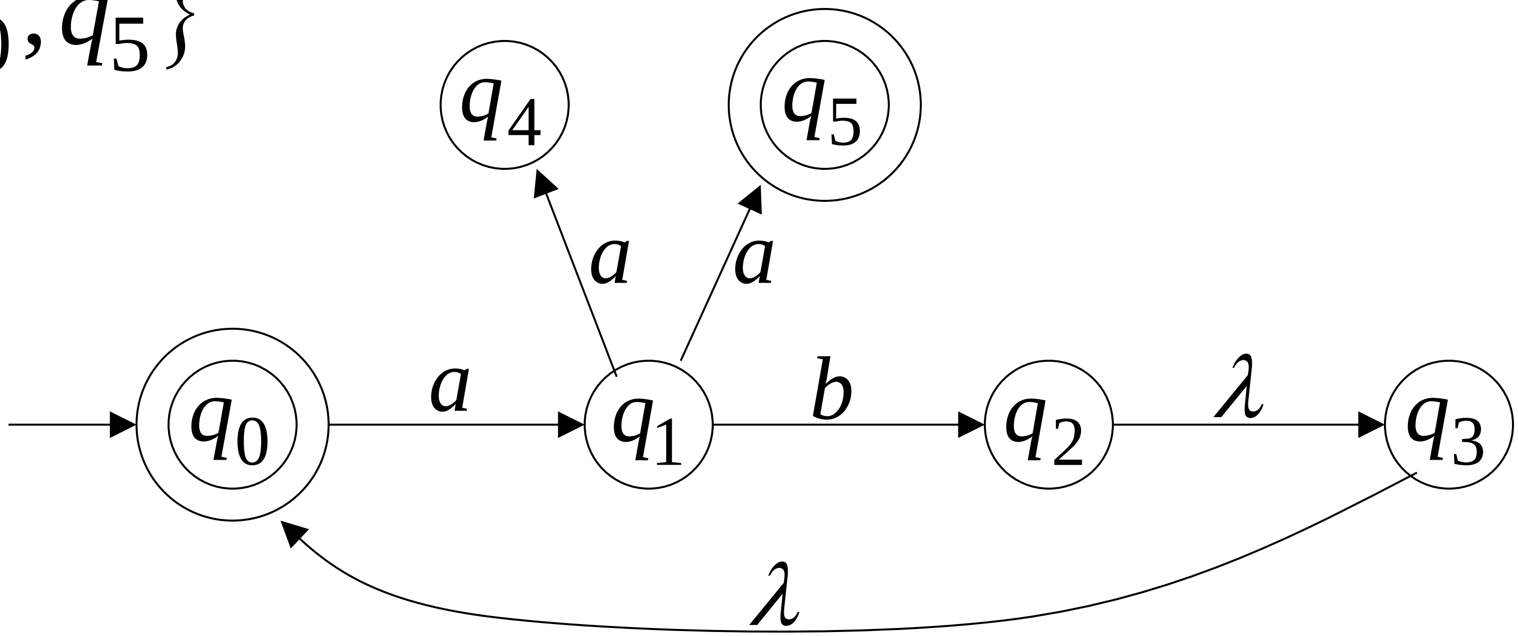
$$F = \{q_0, q_5\}$$



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

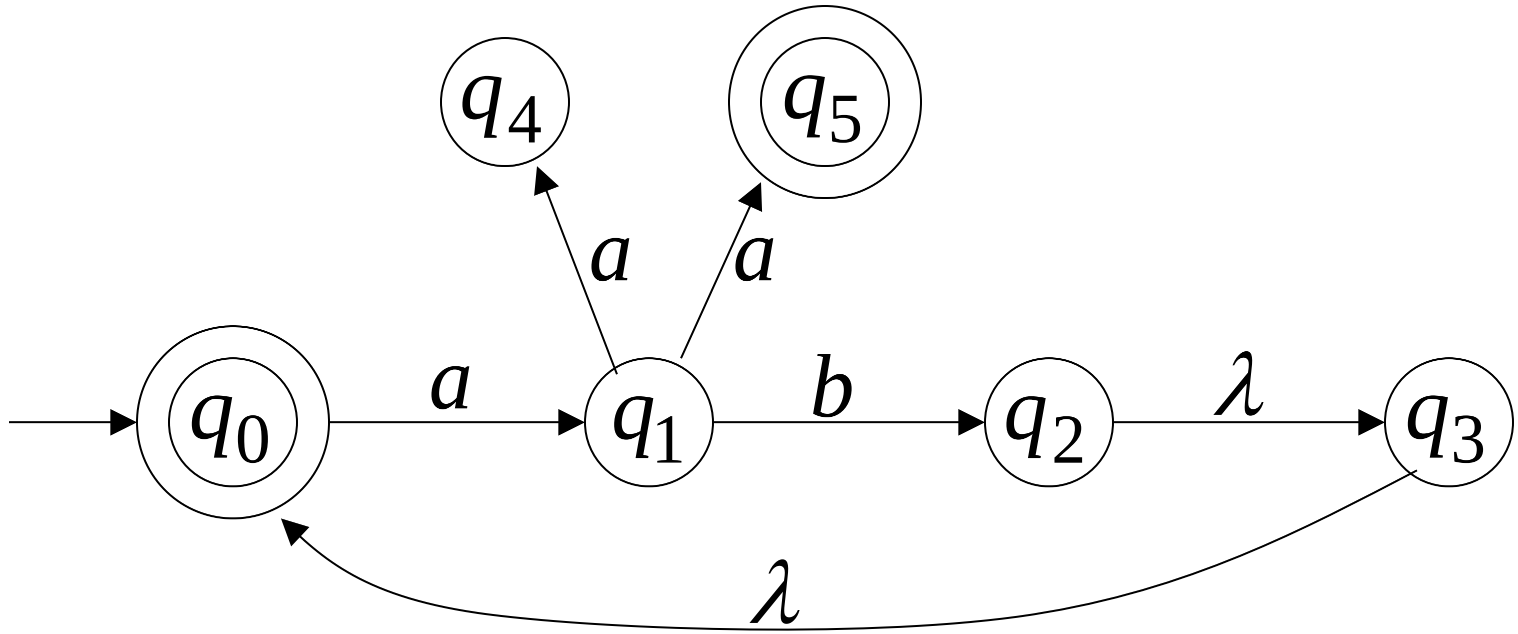
$$\in F$$

$F = \{q_0, q_5\}$



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

$\notin F$



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

NFAs accept the Regular Languages

Equivalence of Machines

Definition:

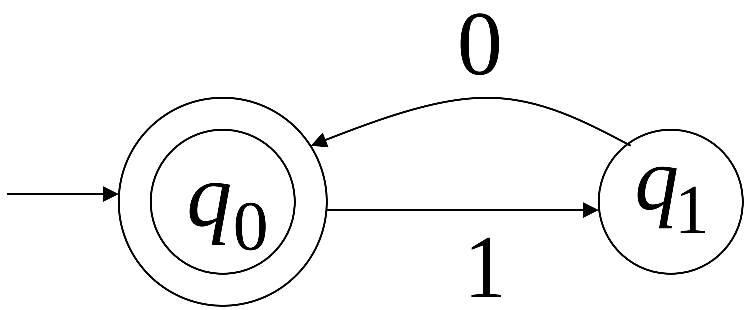
Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

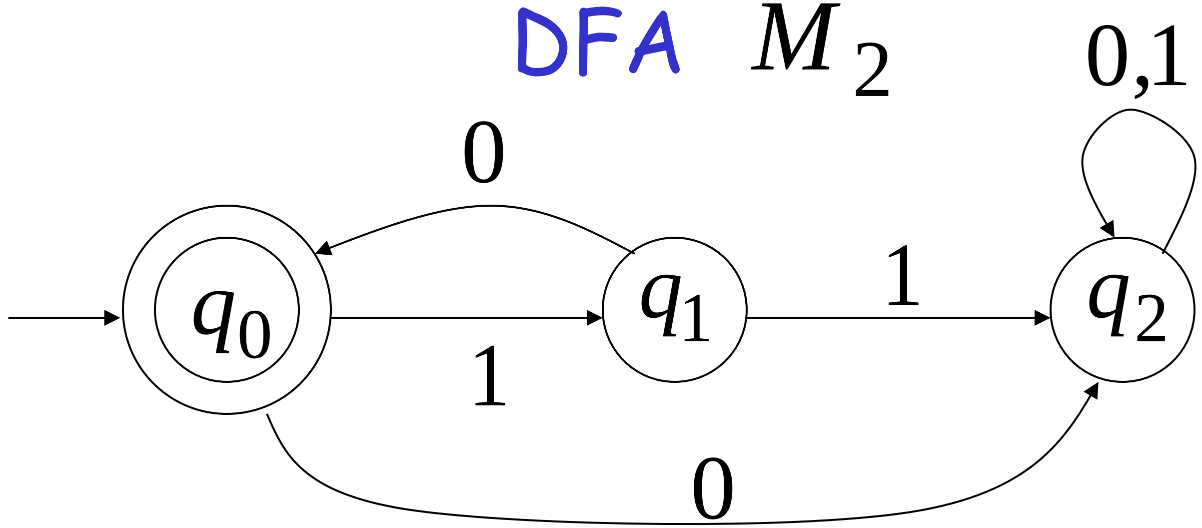
$$L(M_1) = \{10\}^*$$

NFA M_1



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

DFA M_2



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

$\Downarrow$

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

Every DFA is trivially an NFA

For any DFA D, we build an equivalent NFA N with the same transition diagram, i.e., each $\delta_N(q,a)$ is a singleton for $a \in \Sigma$, and $\delta_N(q,\epsilon) = \emptyset$. More formally, if $D = (Q, \Sigma, \delta_D, q_0, F)$ is a DFA, we define the NFA $N = (Q, \Sigma, \delta_N, q_0, F)$, where

$$\delta_N(q,a) = \begin{cases} \{\delta_D(q,a)\} & \text{if } a \in \Sigma, \\ \emptyset & \text{if } a = \epsilon. \end{cases}$$

It is easy to see that $L(N) = L(D)$, i.e., that N is equivalent to D.

Proof-Step 2

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

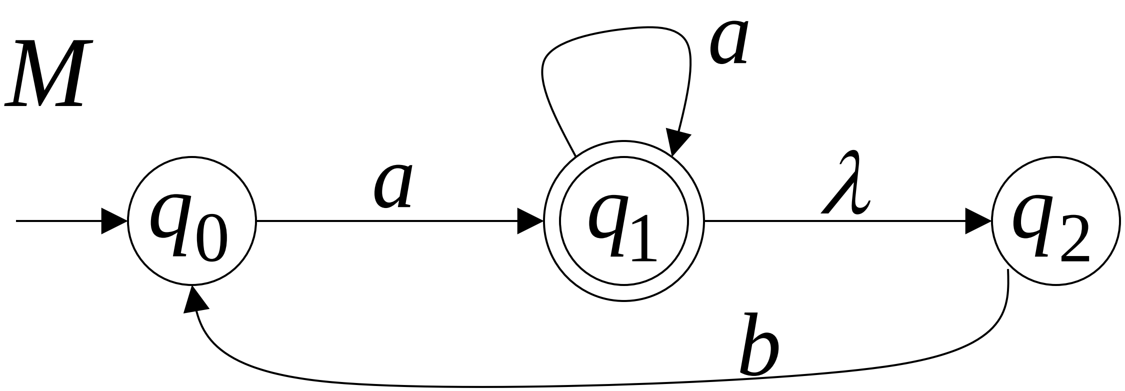
Any NFA can be converted to an equivalent DFA

$$\Downarrow$$

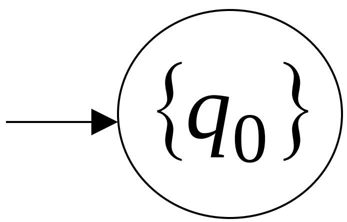
Any language L accepted by an NFA is also accepted by a DFA

Conversion NFA to DFA

NFA M

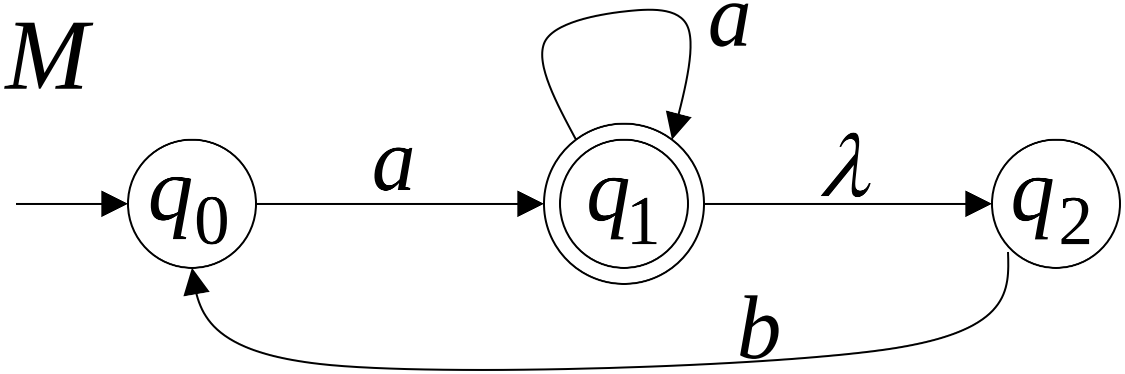


DFA M'

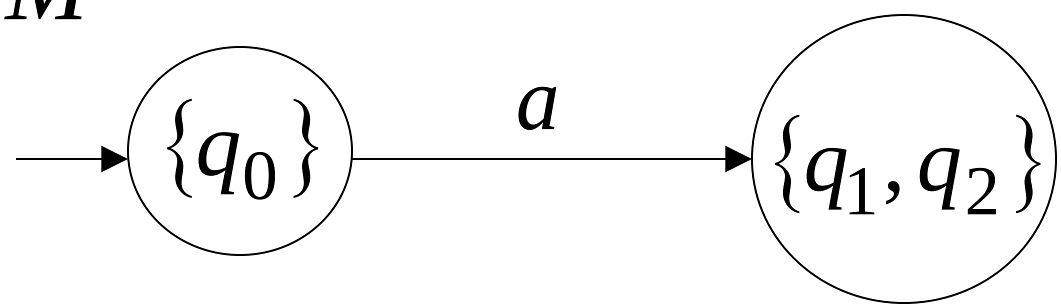


$$\delta^*(q_0, a) = \{q_1, q_2\}$$

NFA M

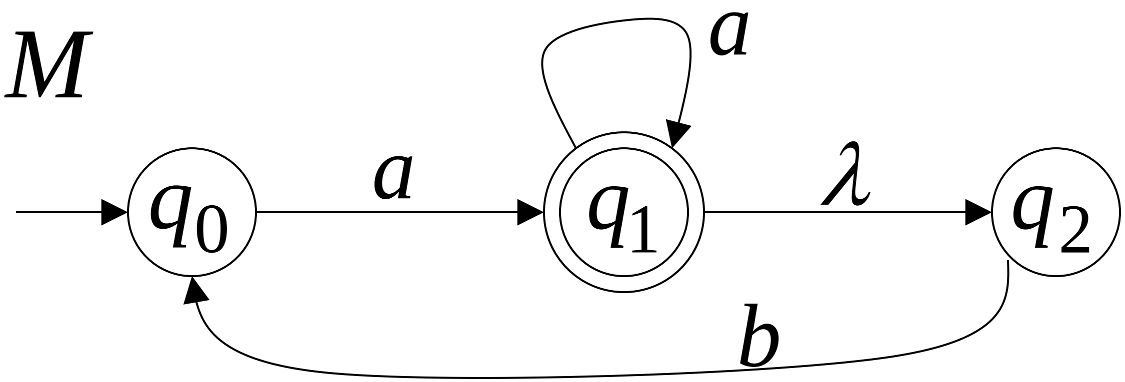


DFA M'

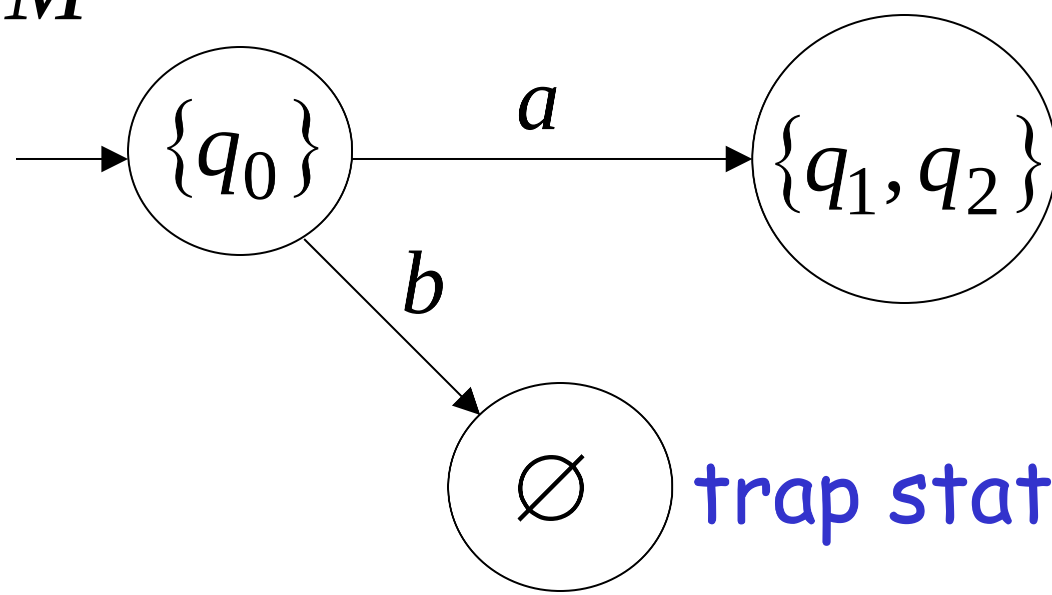


$\delta^*(q_0, b) = \varnothing$ empty set

NFA M

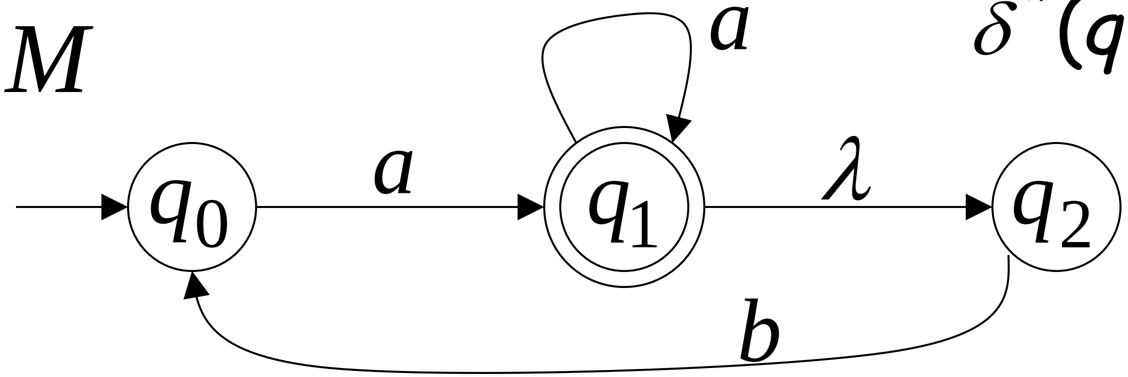


DFA M'



trap state

NFA M

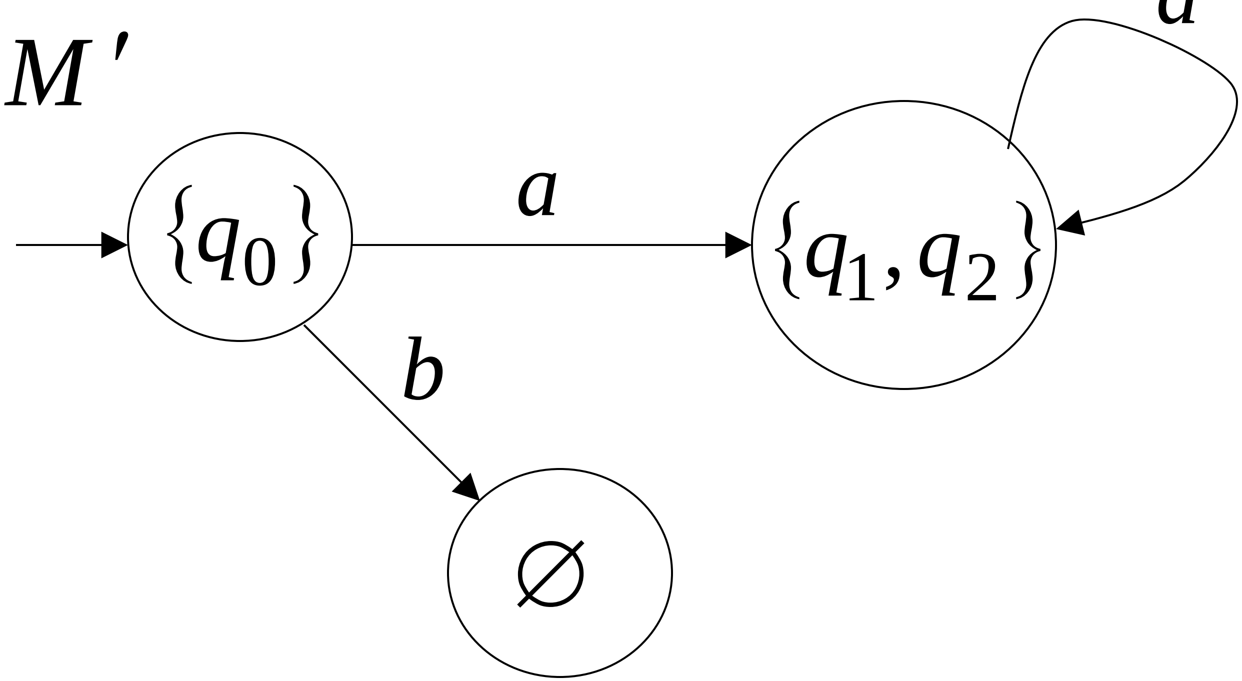


$$\delta^*(q_1, a) = \{q_1, q_2\}$$
$$\delta^*(q_2, a) = \varnothing$$

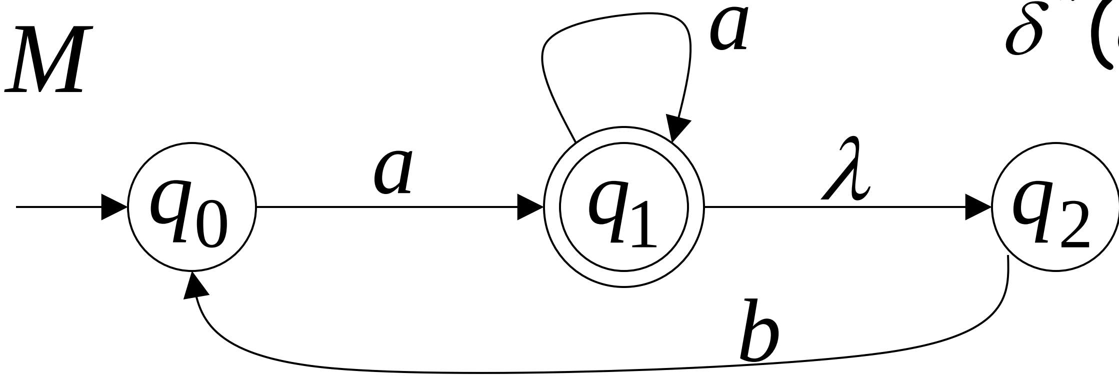
union

$$\{q_1, q_2\}$$

DFA M'



NFA M

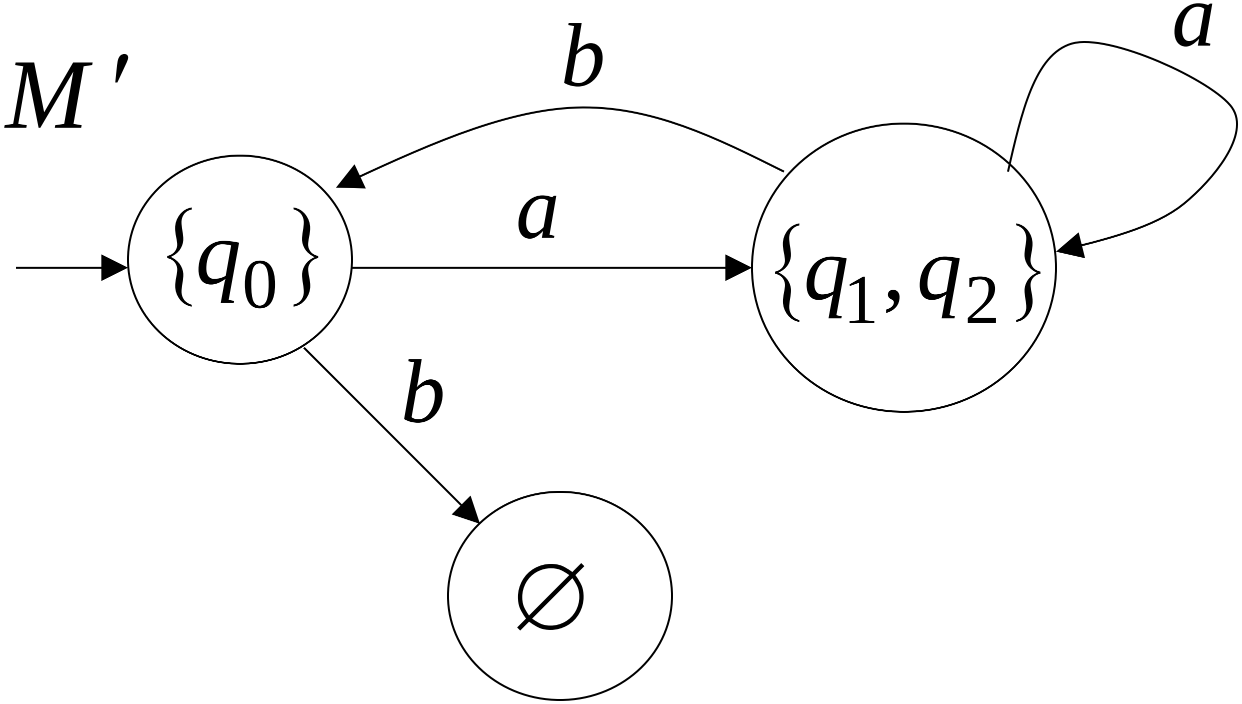


$$\delta^*(q_1, b) = \{q_0\}$$
$$\delta^*(q_2, b) = \{q_0\}$$

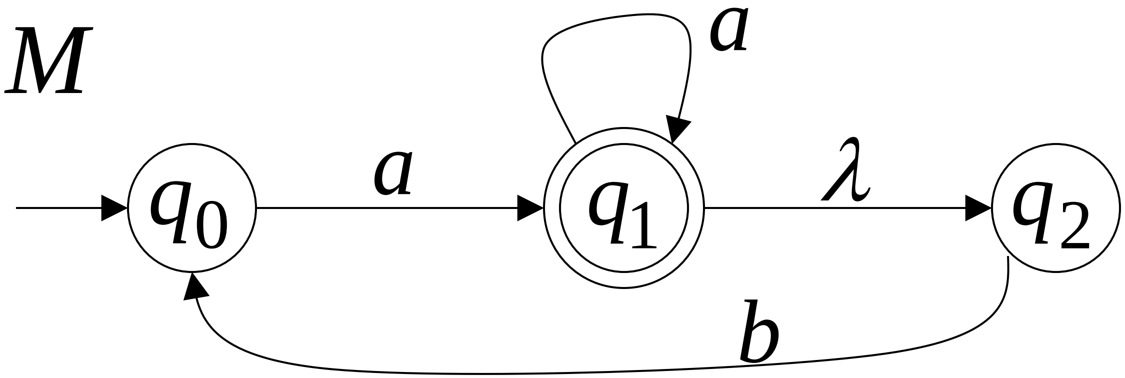
union

$$\{q_0\}$$

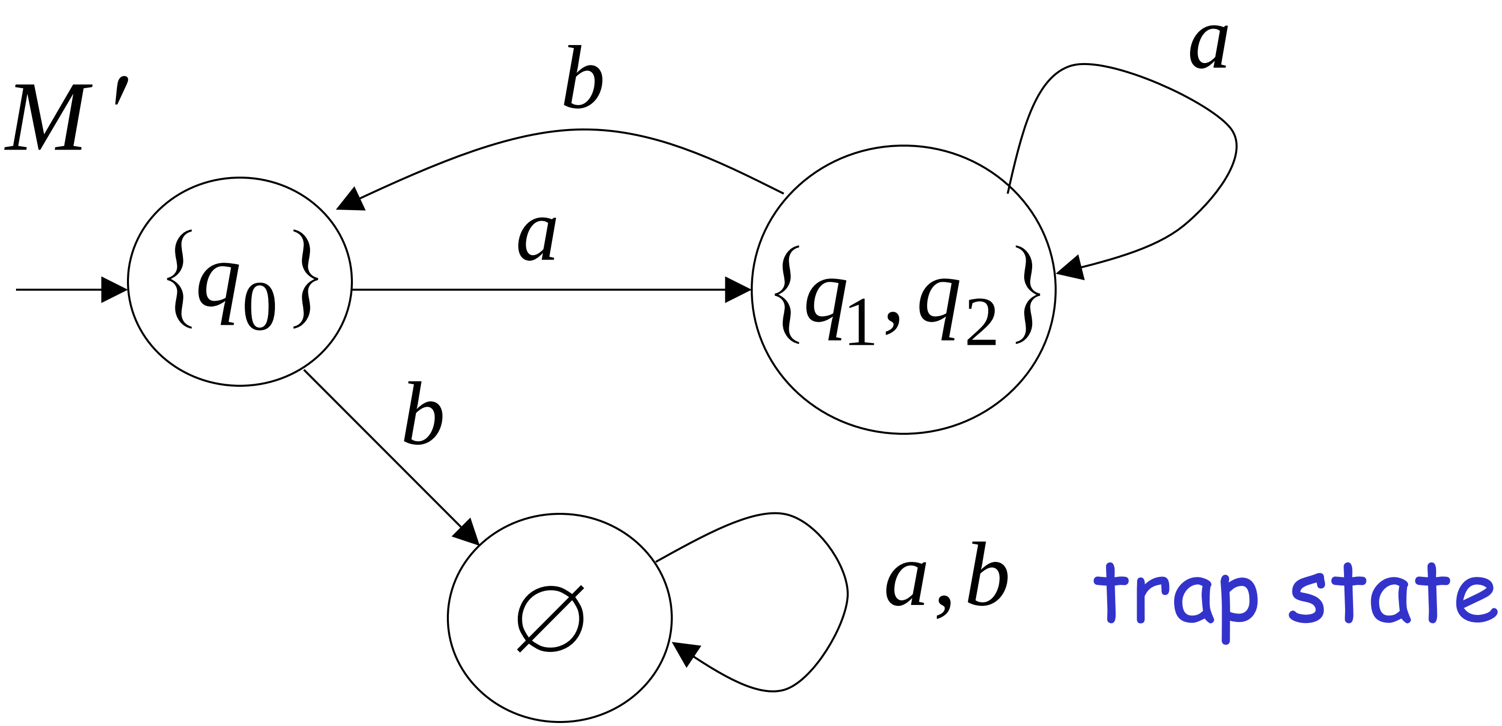
DFA M'



NFA M

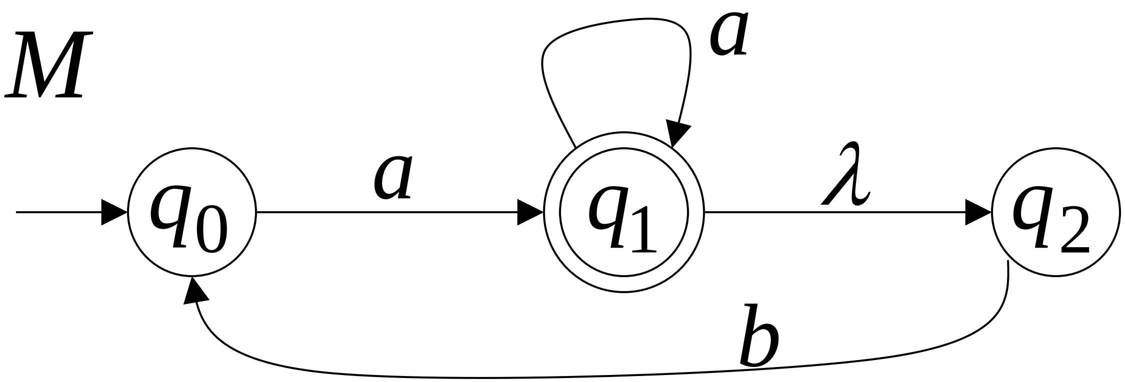


DFA M'



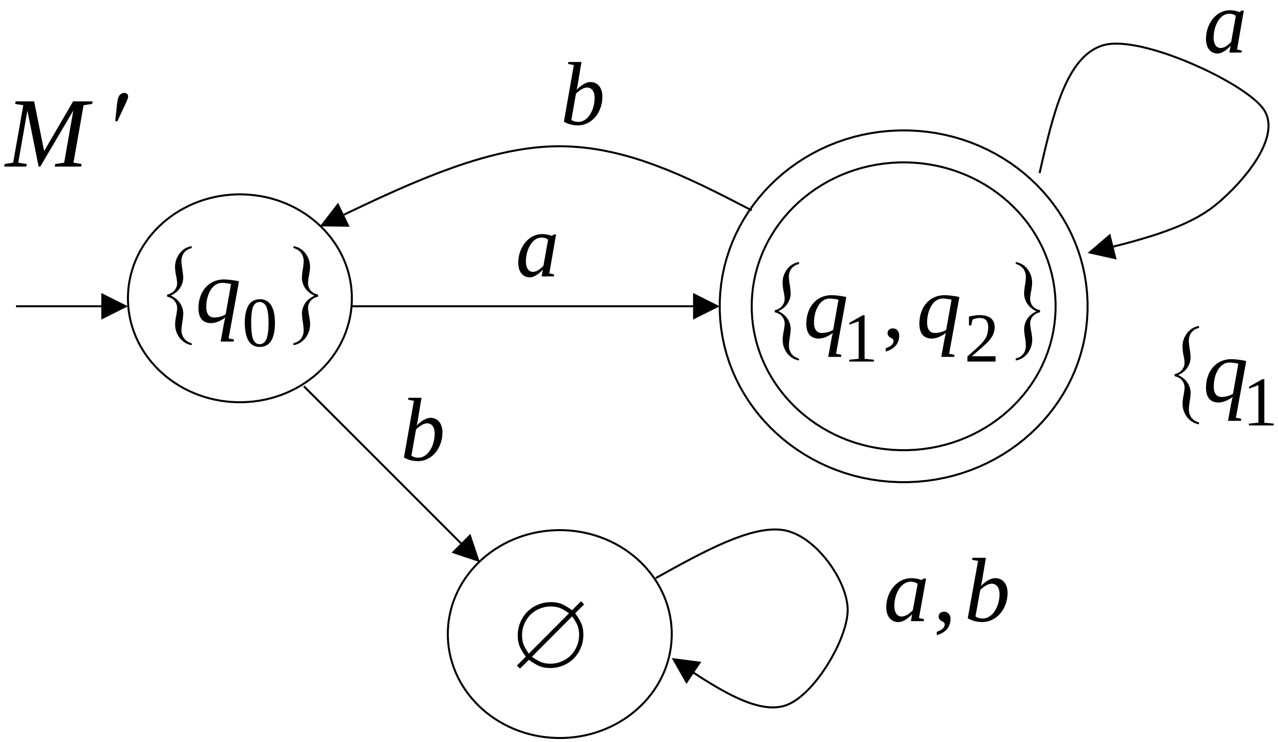
END OF CONSTRUCTION

NFA M



$q_1 \in F$

DFA M'



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

General Conversion Procedure

Input: an NFA M

Output: an equivalent DFA M'

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

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

The DFA has states from the power set

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

Conversion Procedure Steps

step

1. Initial state of NFA: q_0

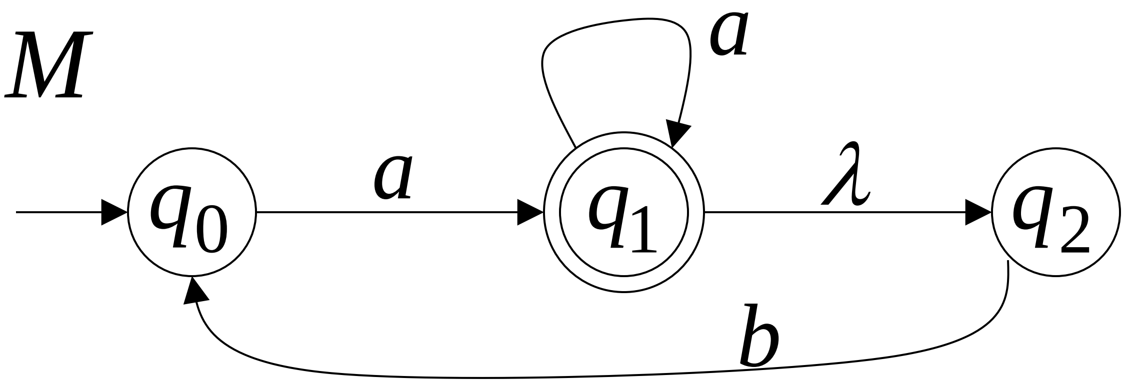
$\Downarrow$

Initial state of DFA (lambda-closure):

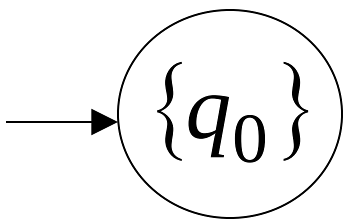
$$\{q_0\}$$

Example

NFA M



DFA M'



step

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

compute in the NFA

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

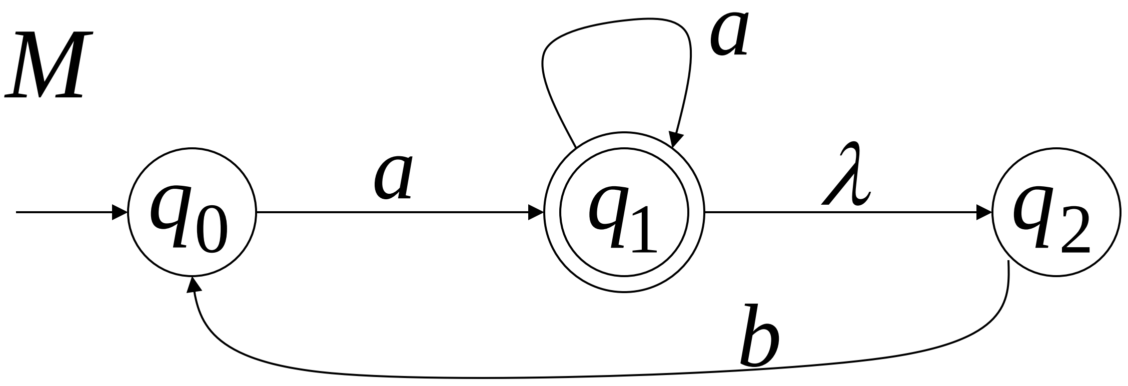
Union

add transition to DFA

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

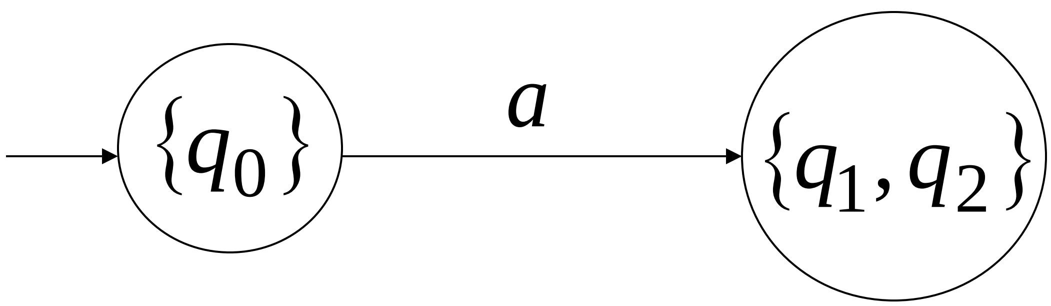
Example $\delta^*(q_0, a) = \{q_1, q_2\}$

NFA M



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

DFA M'

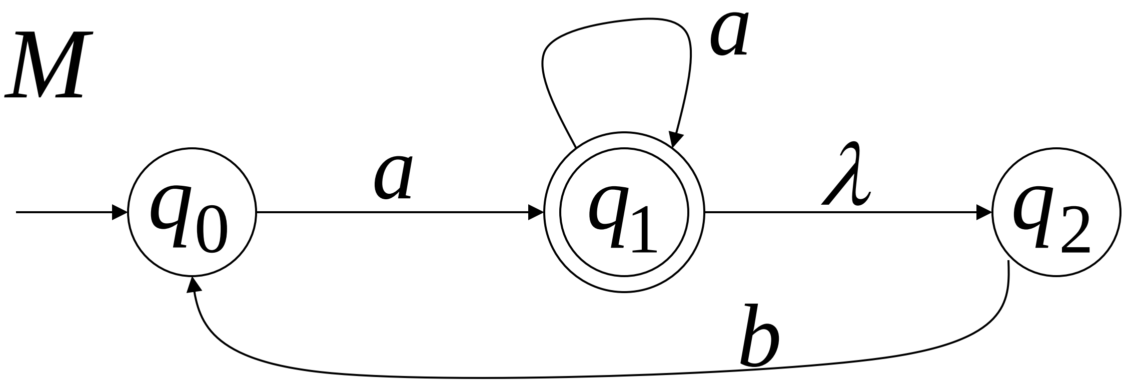


step

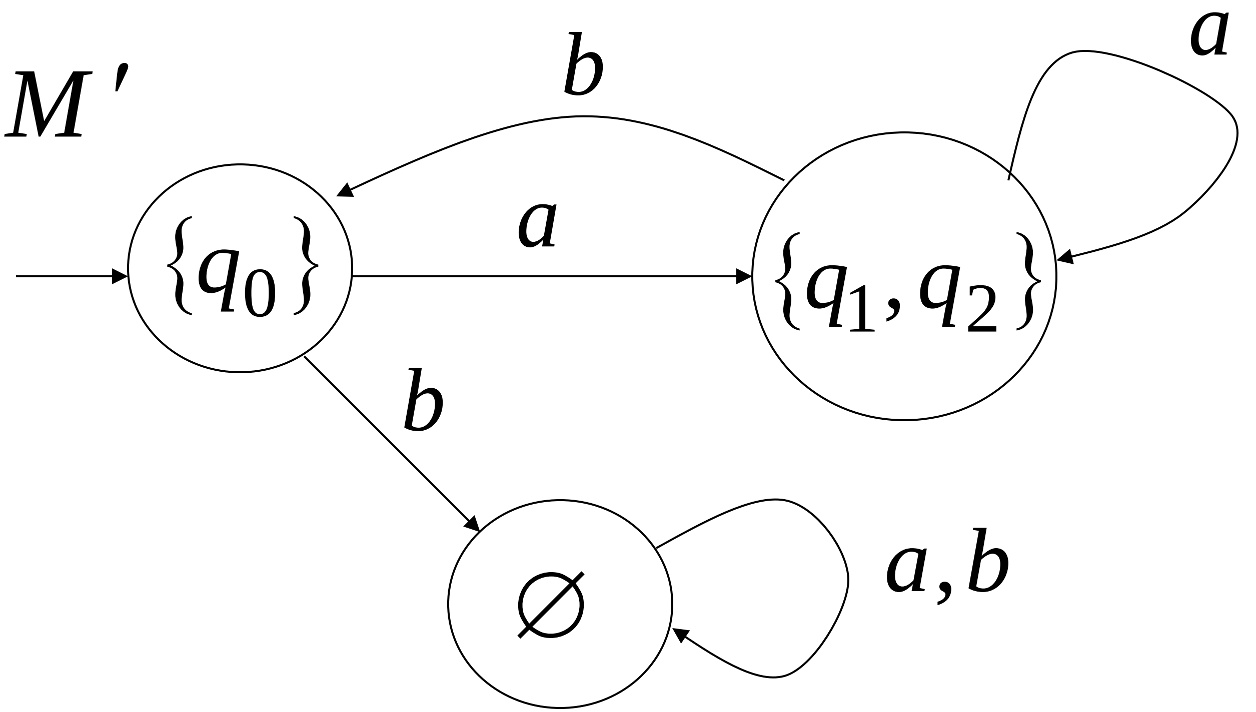
3. Repeat Step 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA

Example

NFA M



DFA M'



step

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

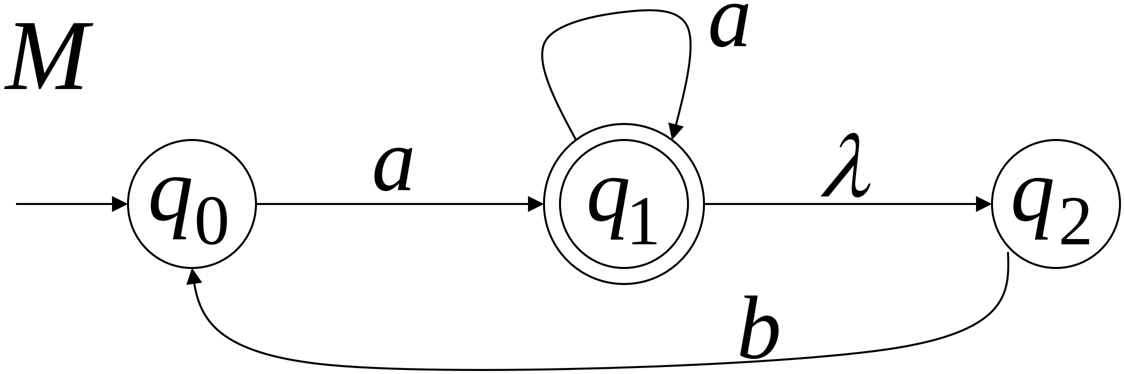
if some q_j is accepting state in NFA

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

is accepting state in DFA

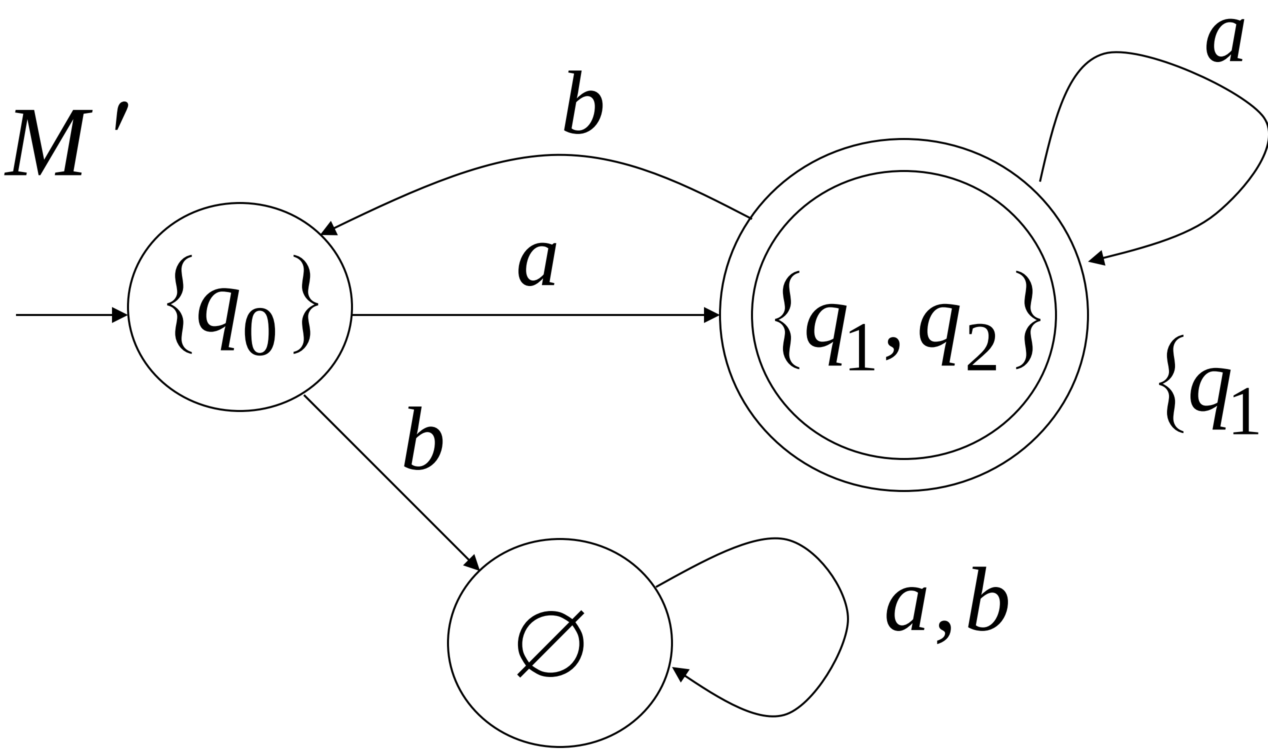
Example

NFA M



$q_1 \in F$

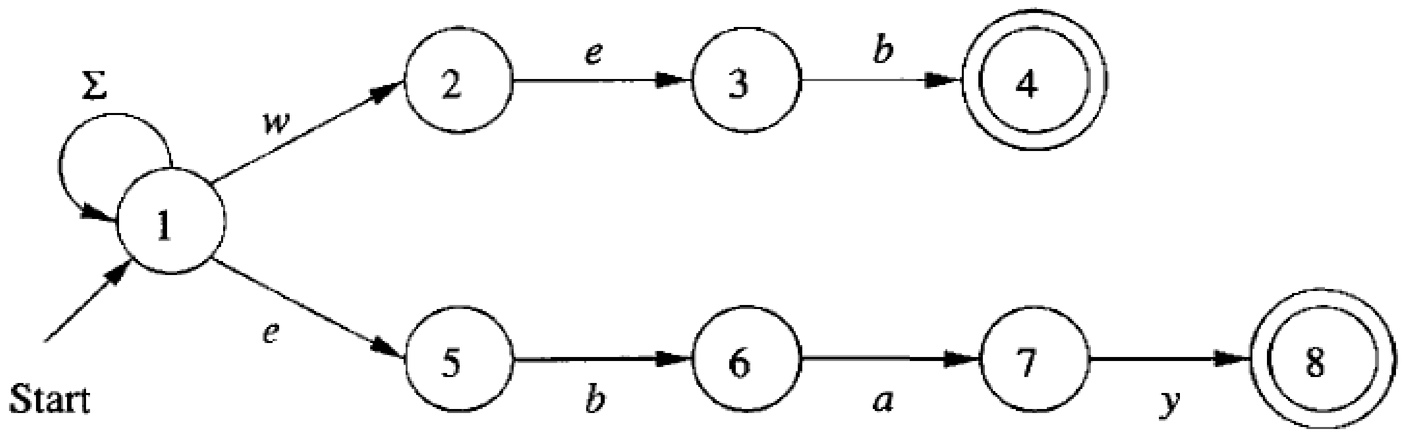
DFA M'



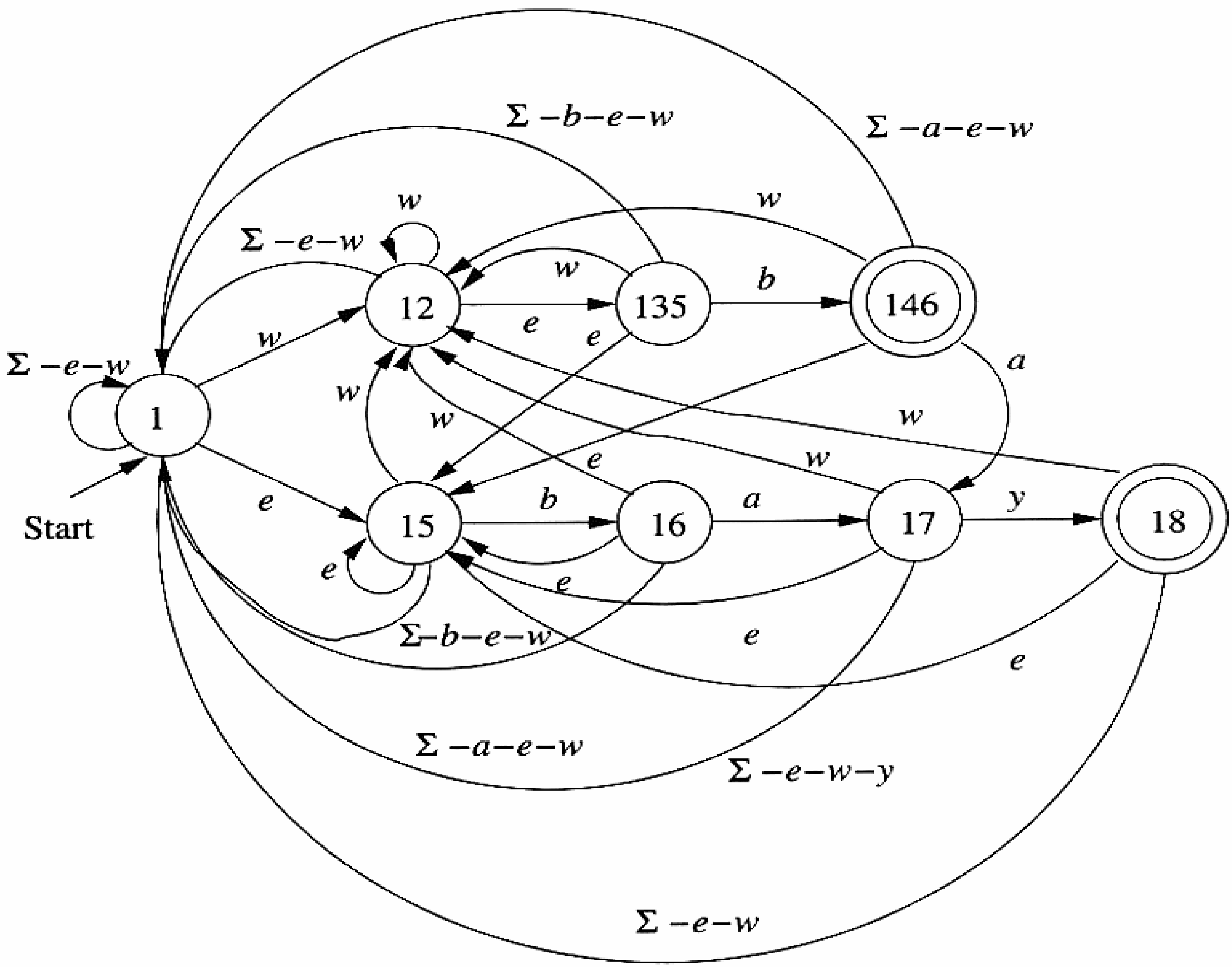
$\{q_1, q_2\} \in F'$

Keyword search: example

The following NFA recognizes occurrences of the keywords WEB and EBAY:



Corresponding DFA for keywords search



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

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

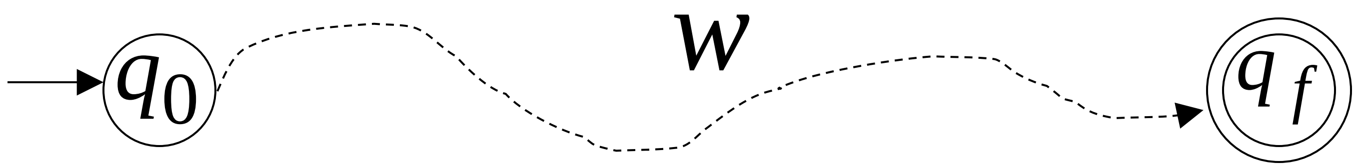
First we show: $L(M) \subseteq L(M')$

We only need to prove:

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

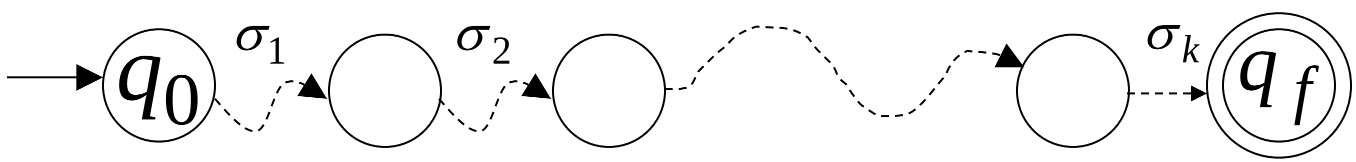
NFA

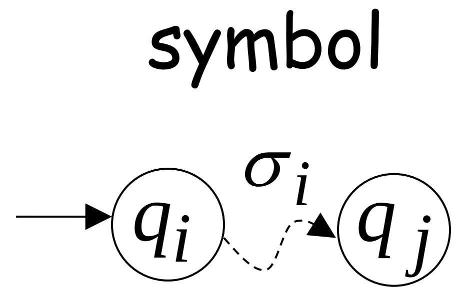
Consider $w \in L(M)$



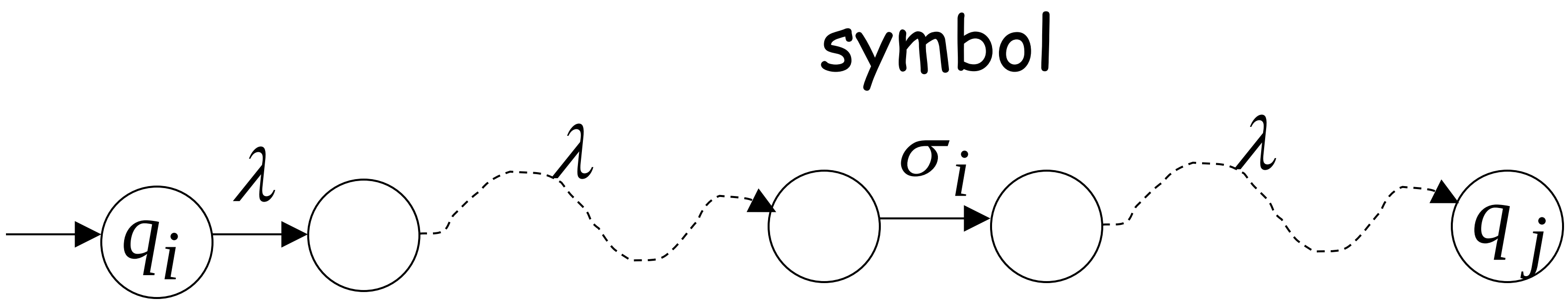
symbols

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





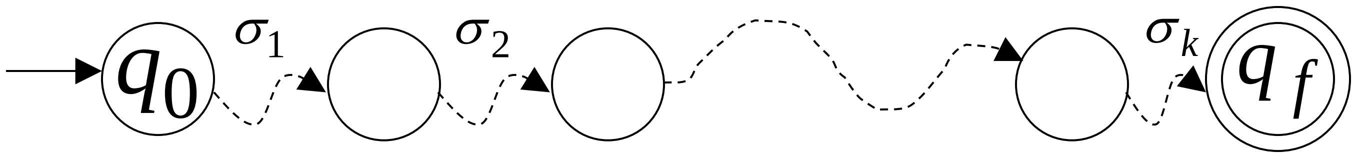
denotes a possible sub-path like



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

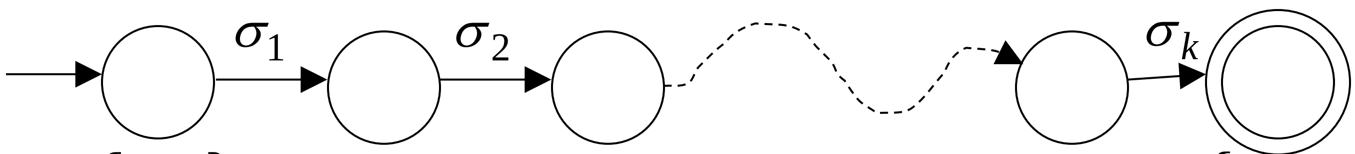
$w = \sigma_1 \sigma_2 \cdots \sigma_k$

NFA $M:$



then

DFA $M':$



$\{q_0\}$ state label

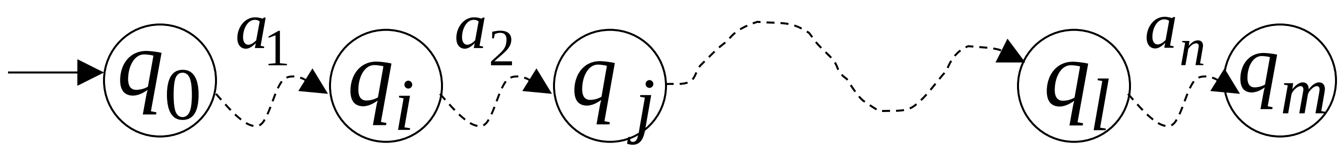
$\{q_f, \ldots\}$ state label

$w \in L(M')$

More generally, we will show that if in M:

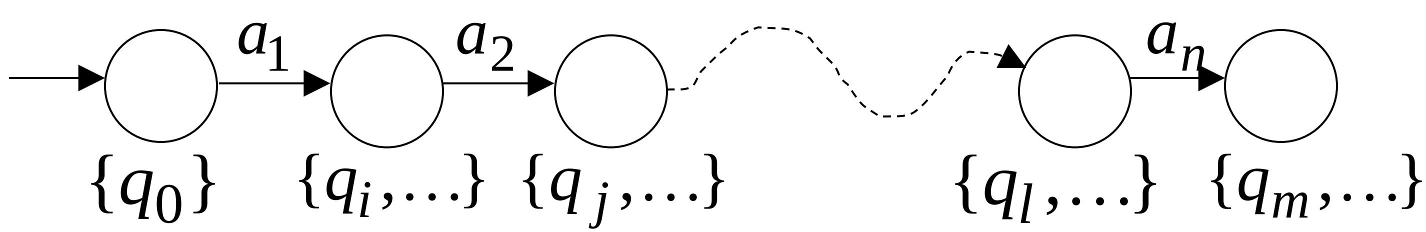
(arbitrary string) $v = a_1 a_2 \cdots a_n$

NFA M :



then

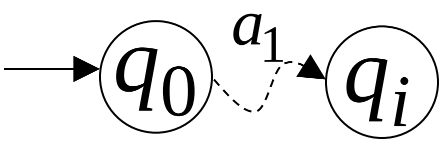
DFA M' :



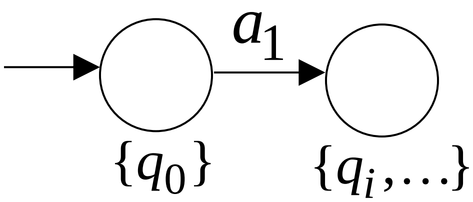
Proof by induction on $|v|$

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

NFA M:



DFA M':



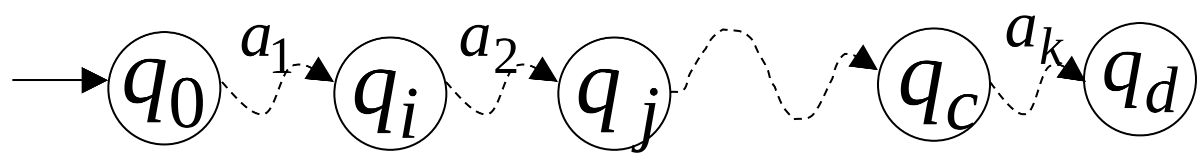
is true by construction of M'

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

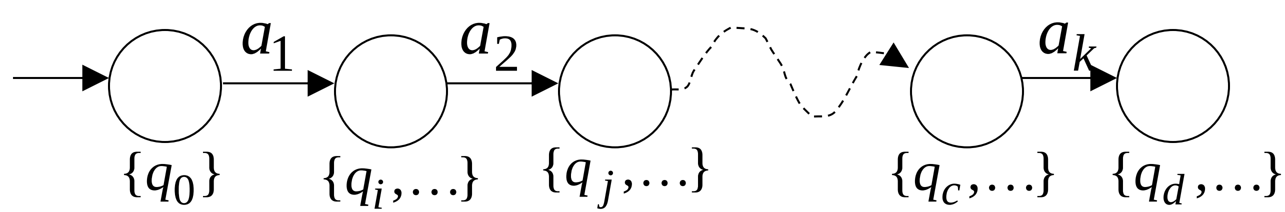
$$v = a_1 a_2 \cdots a_k$$

Suppose that the following hold

NFA M :



DFA M' :

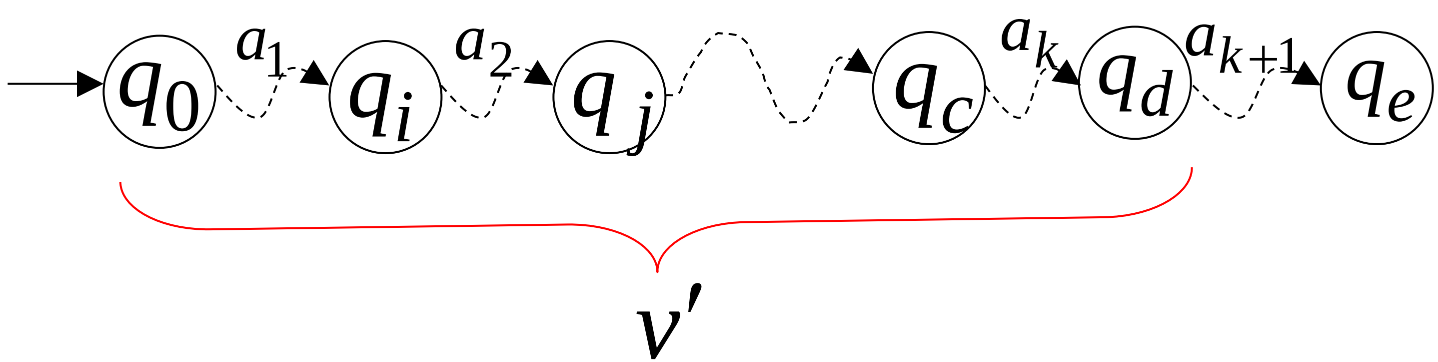


Induction Step: $|v| = k+1$

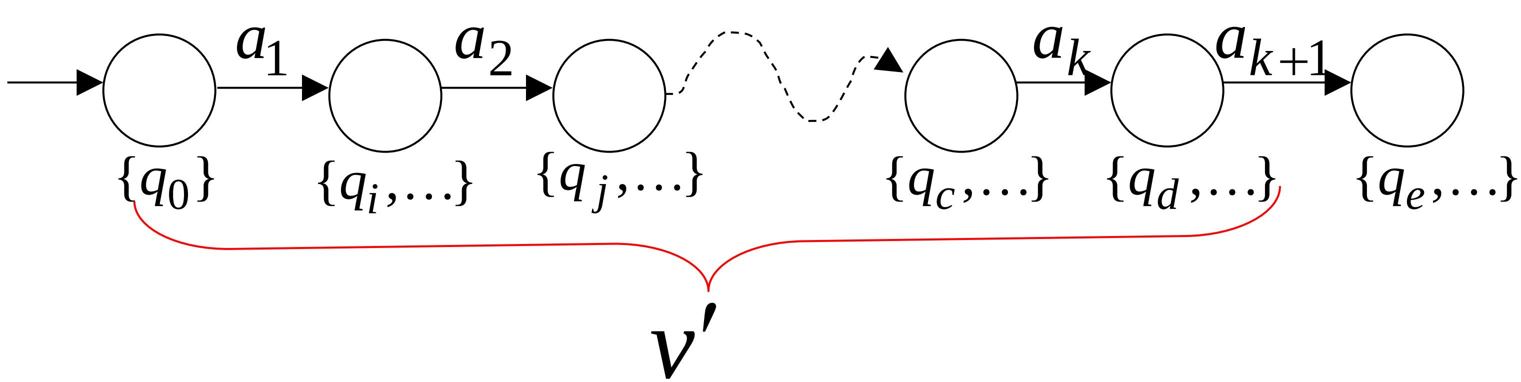
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

Then this is true by construction of M'

NFA $M:$



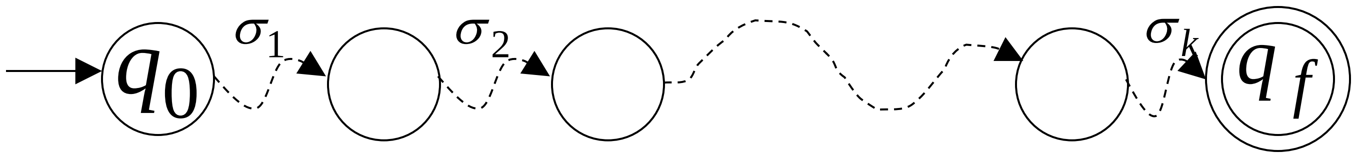
DFA $M':$



Therefore if $w \in L(M)$

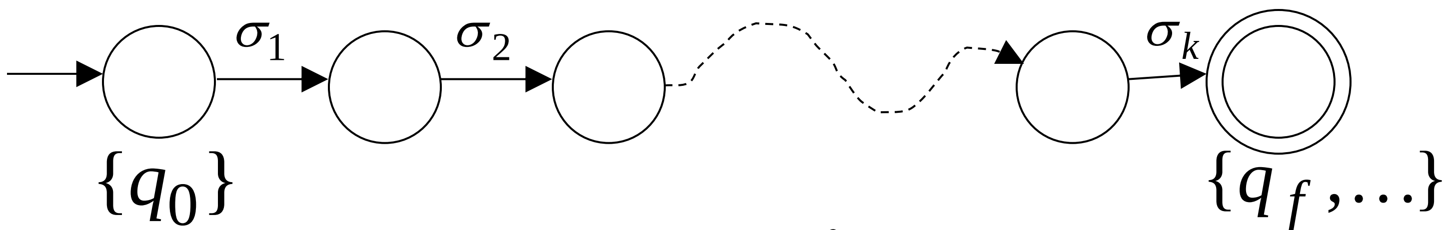
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$

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

Clean NFA

The notion of a clean NFA pops up on several occasions.

Definition

Let M be an NFA. We say that M is *clean* iff

- M has exactly one final state, which is distinct from the start state,
- M has no transitions into its start state (even self-loops), and
- M has no transitions out of its final state (even self-loops).

Existence of a Clean NFA

Proposition

For any NFA M, there is an equivalent clean NFA N.

Proof.

If M is not clean, then we can "clean it up" by adding two additional states:

- a new start state with a single ε-transition to M's original start state (which is no longer the start state), and
- a new final state with ε-transitions from all of M's original final states (which are no longer final states) to the new final state.

The new NFA is obviously clean, and a simple, informal argument shows that it is equivalent to the original M. □

Q: Give clean NFAs for concatenation and union (series and parallel connections, respectively) as well as for the Kleene closure.