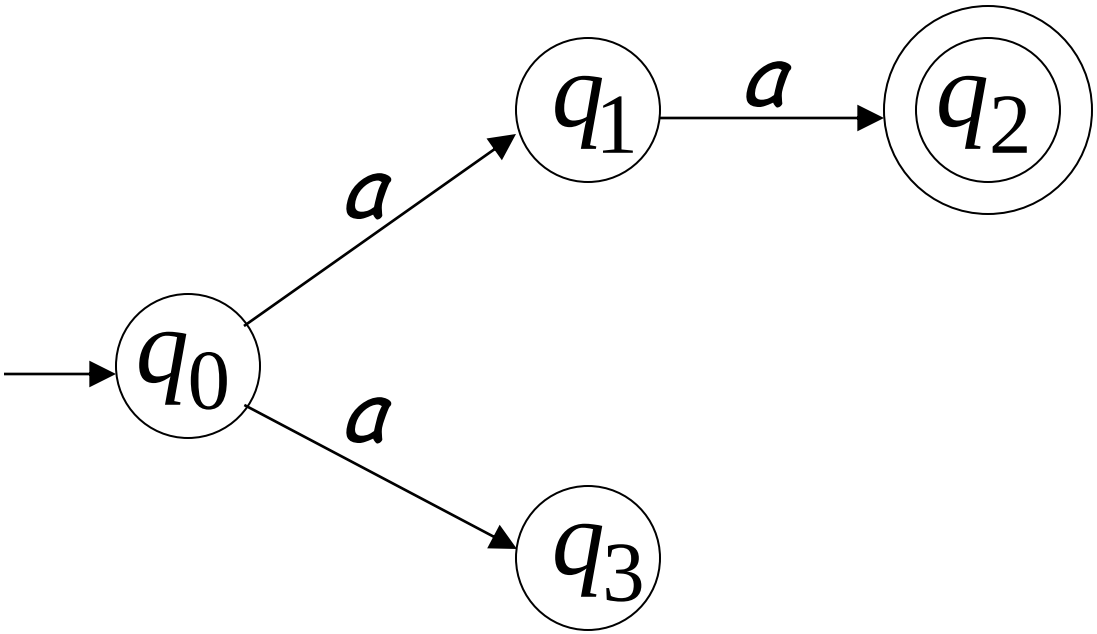
Non Deterministic Automata

Nondeterministic Finite Accepter (NFA)

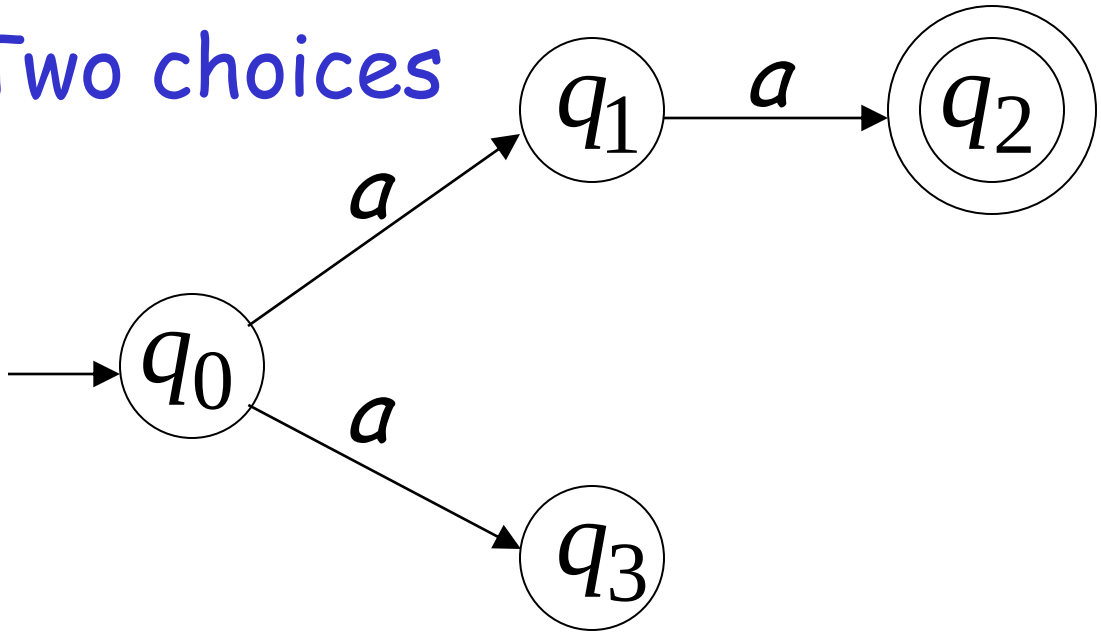
Alphabet = $\{a\}$



Nondeterministic Finite Accepter (NFA)

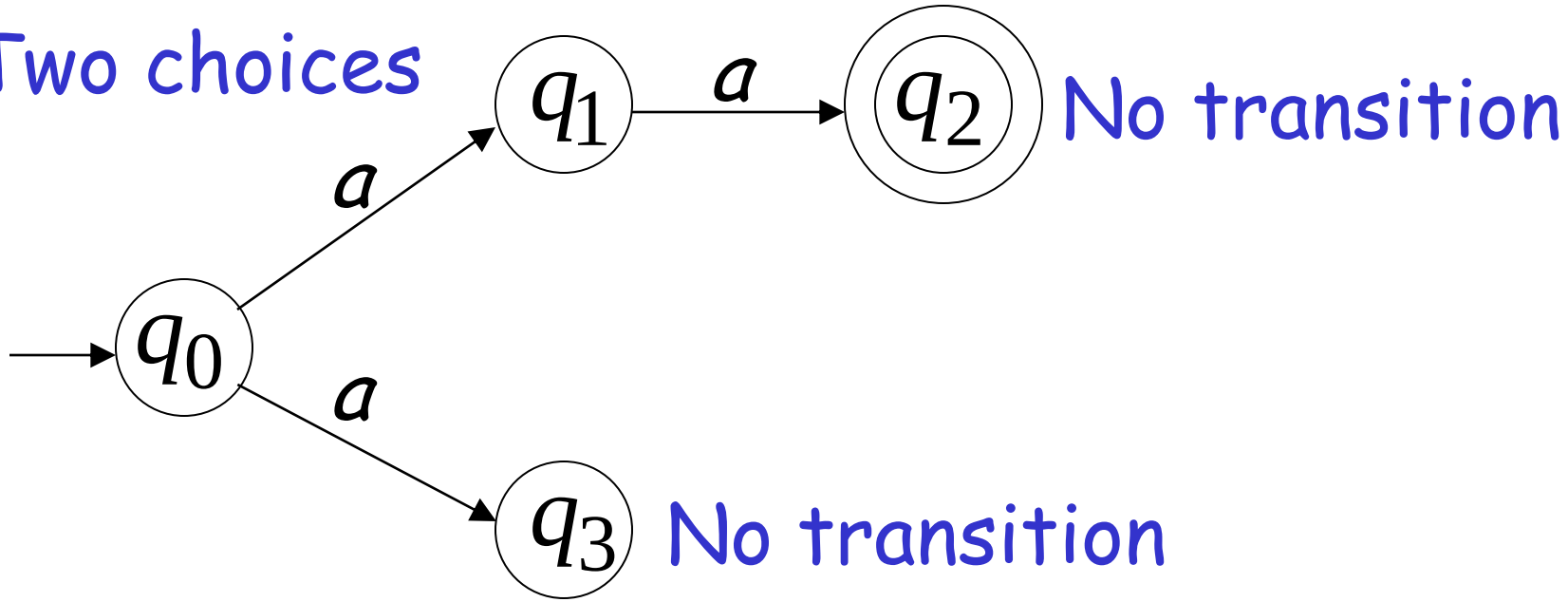
Alphabet = $\{a\}$

Two choices

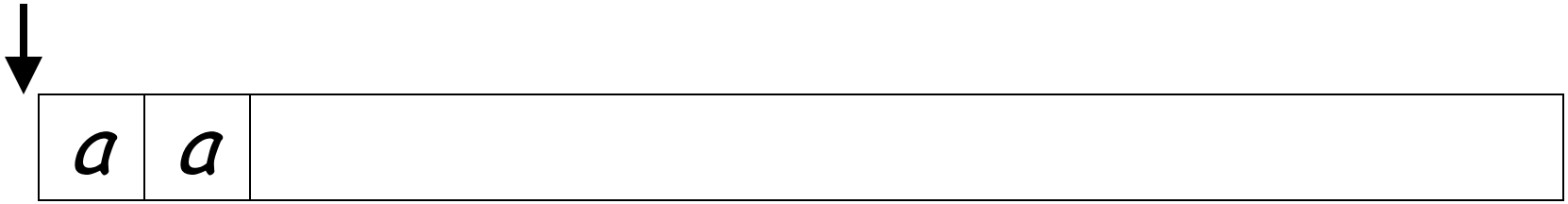


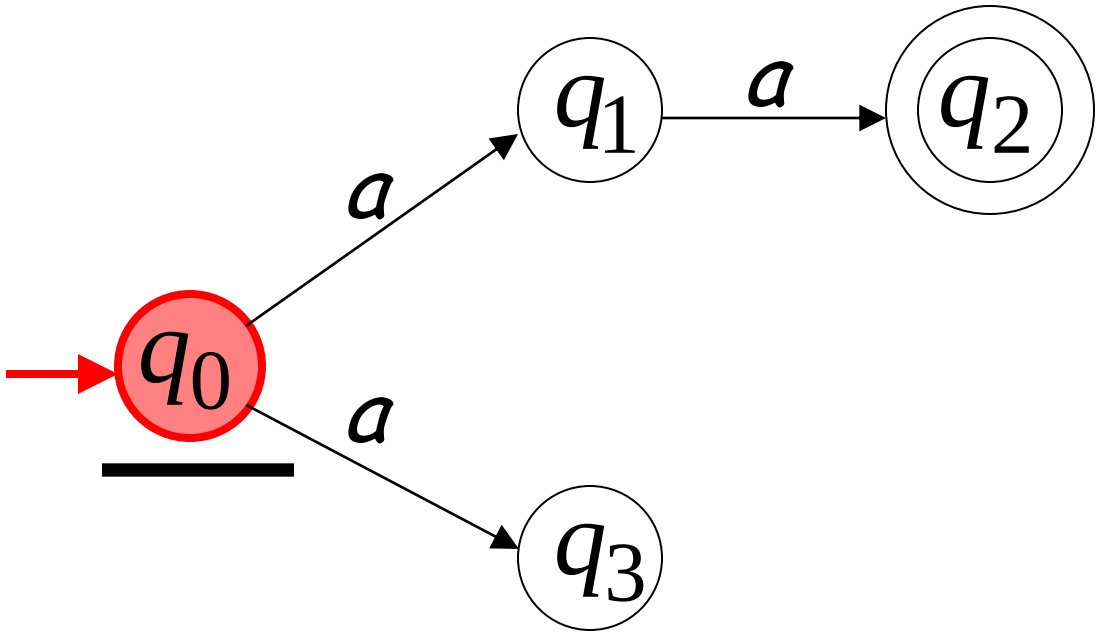
Nondeterministic Finite Accepter (NFA)

Alphabet = $\{a\}$

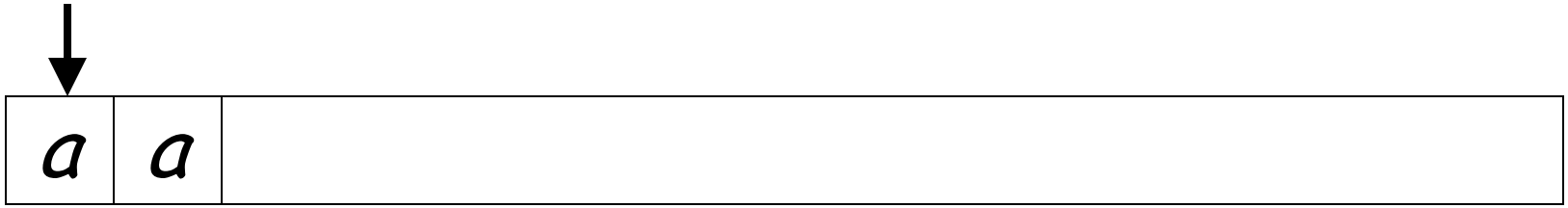


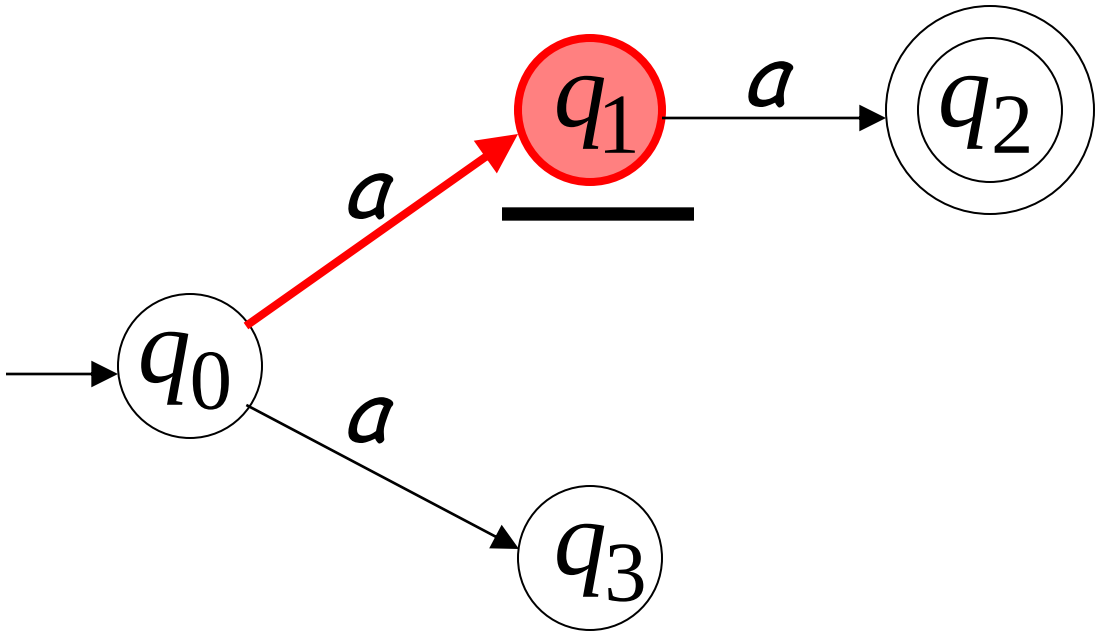
First Choice



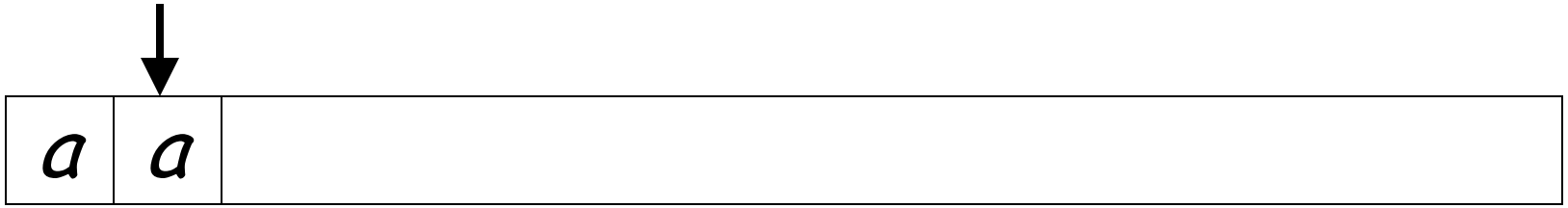


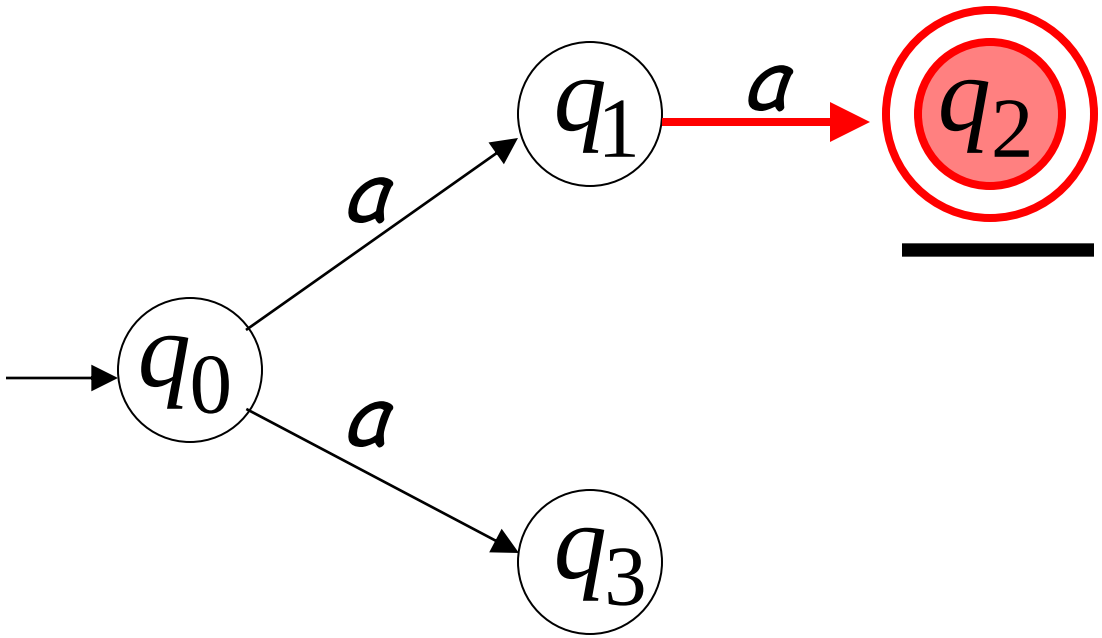
First Choice



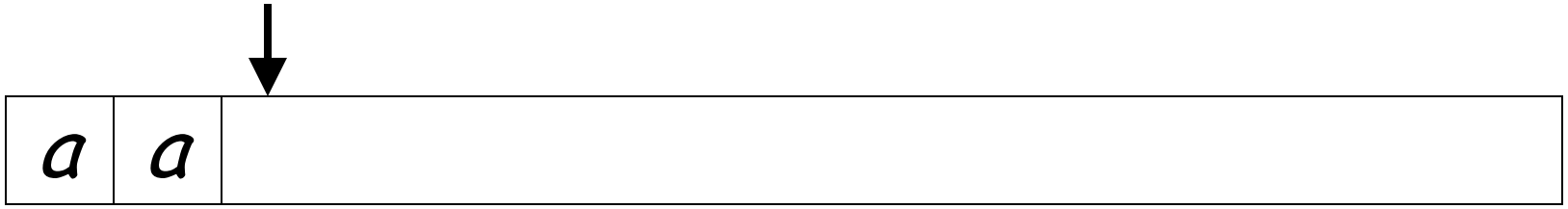


First Choice

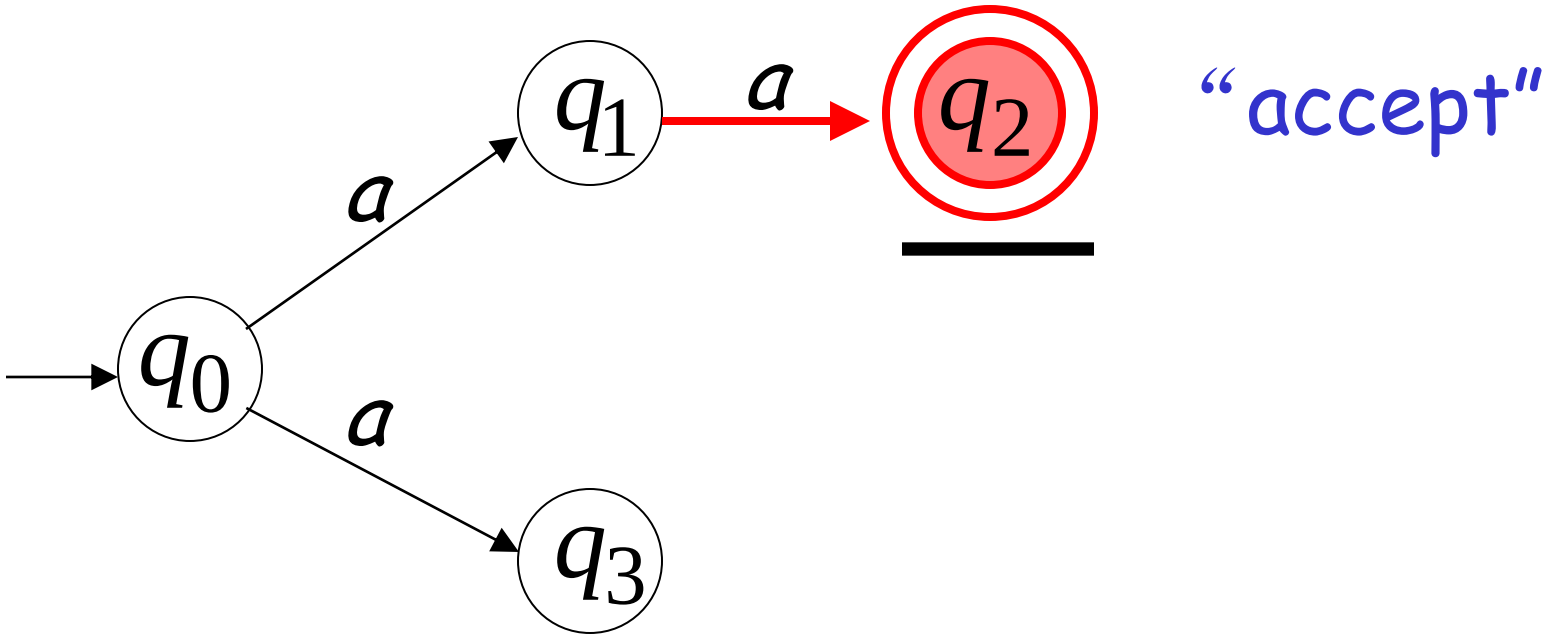




First Choice

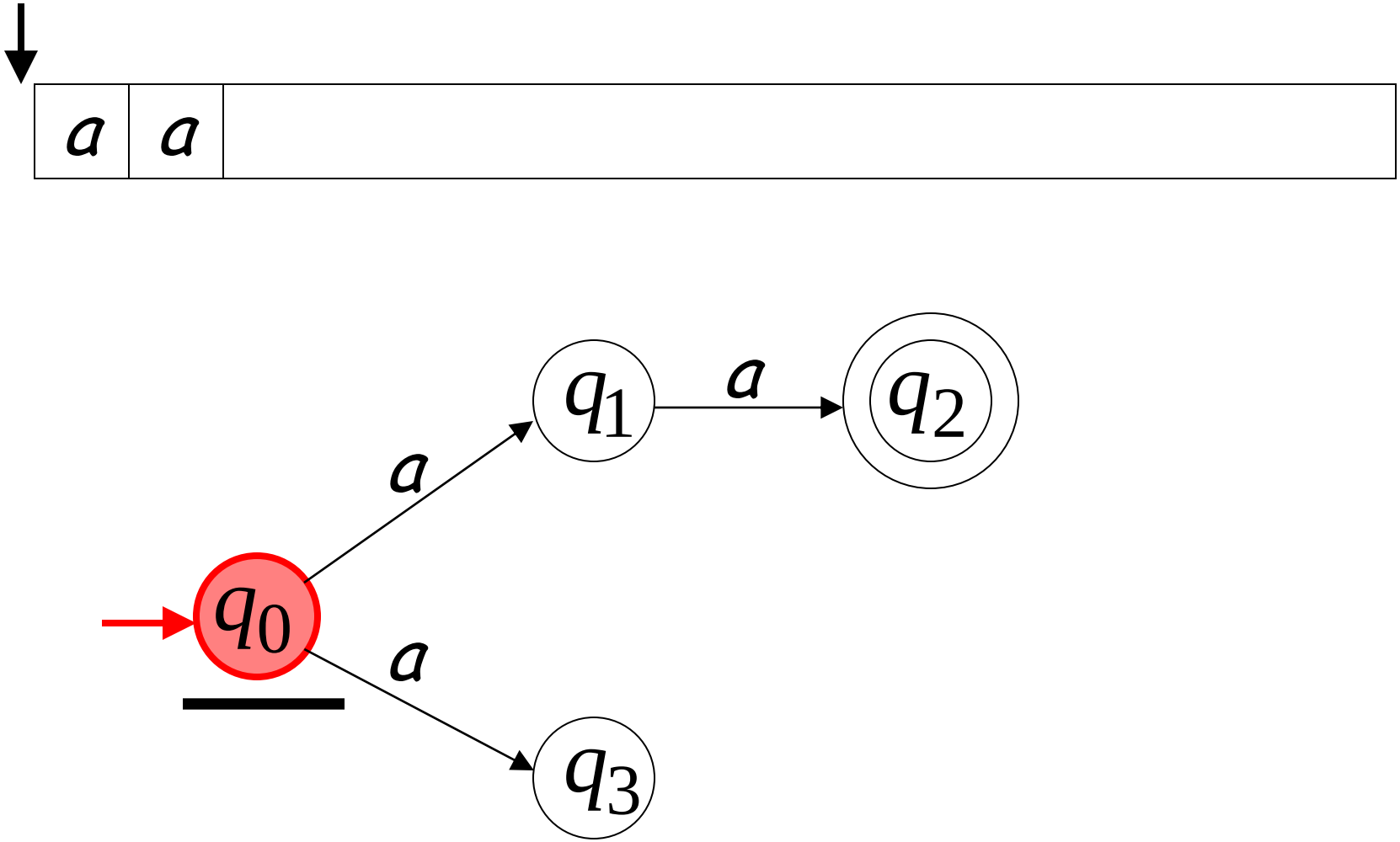


All input is consumed

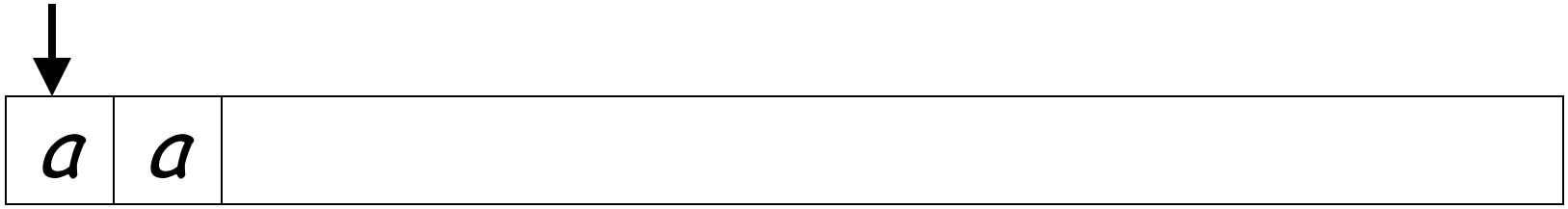


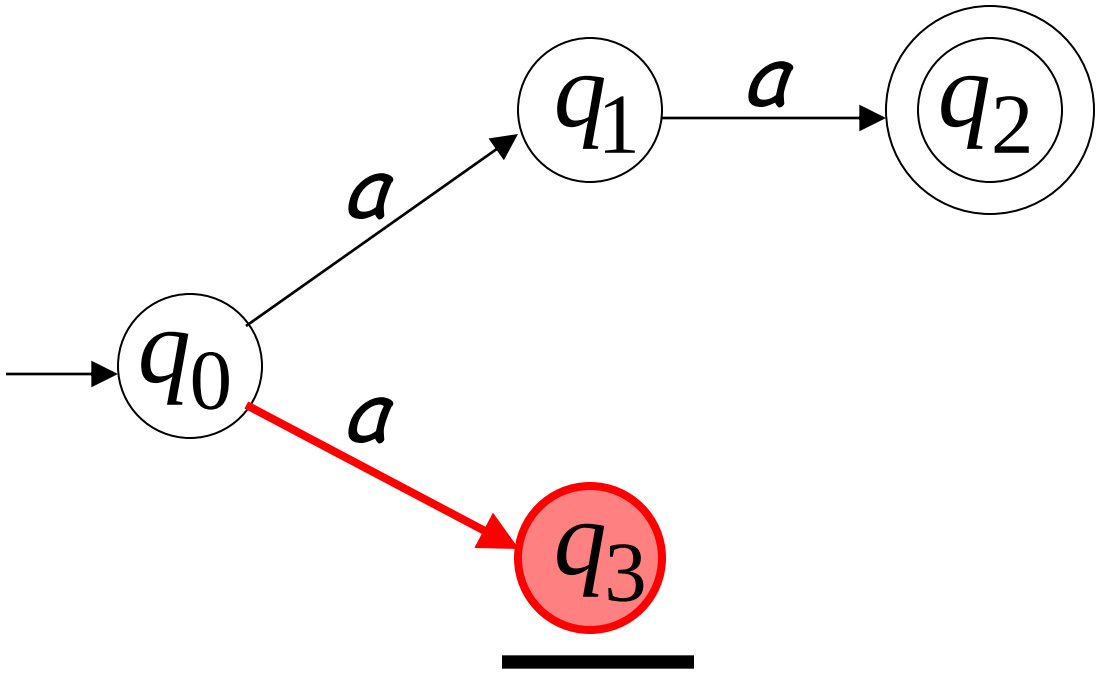
"accept"

Second Choice

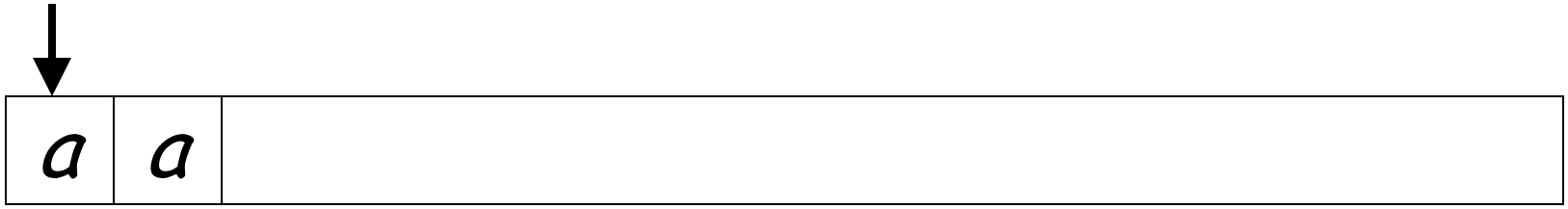


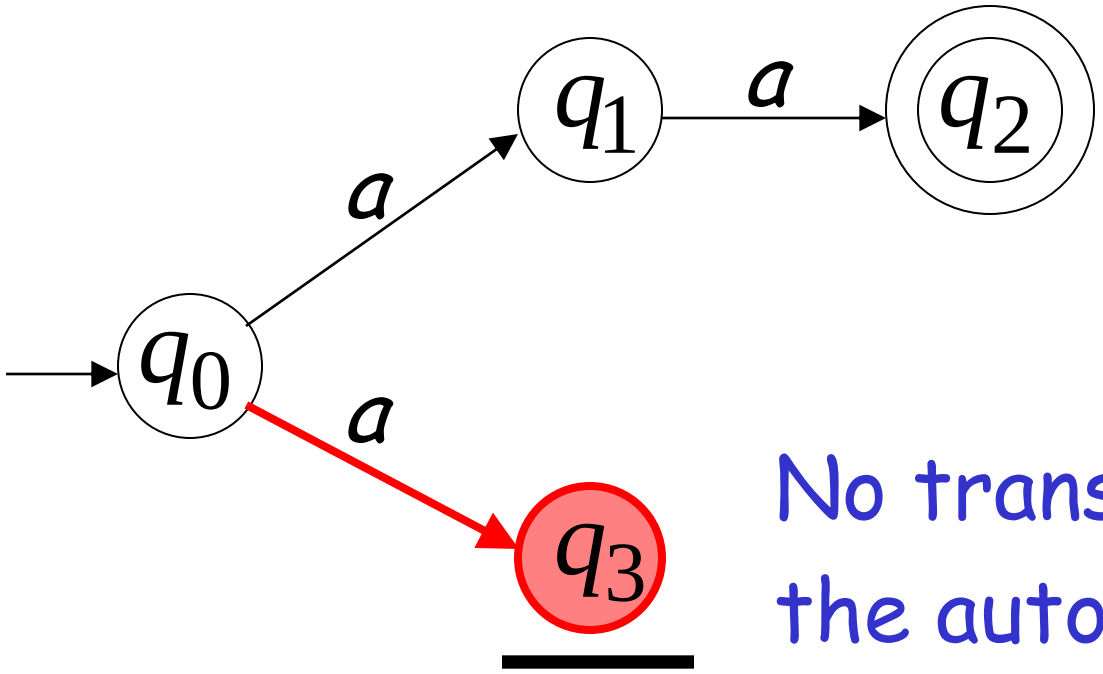
Second Choice





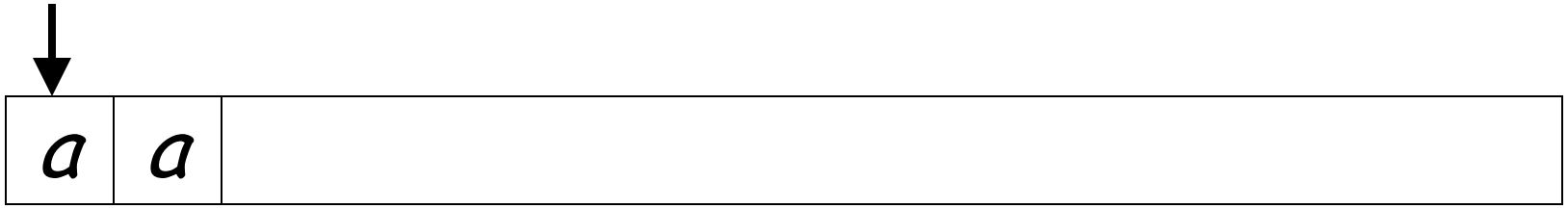
Second Choice



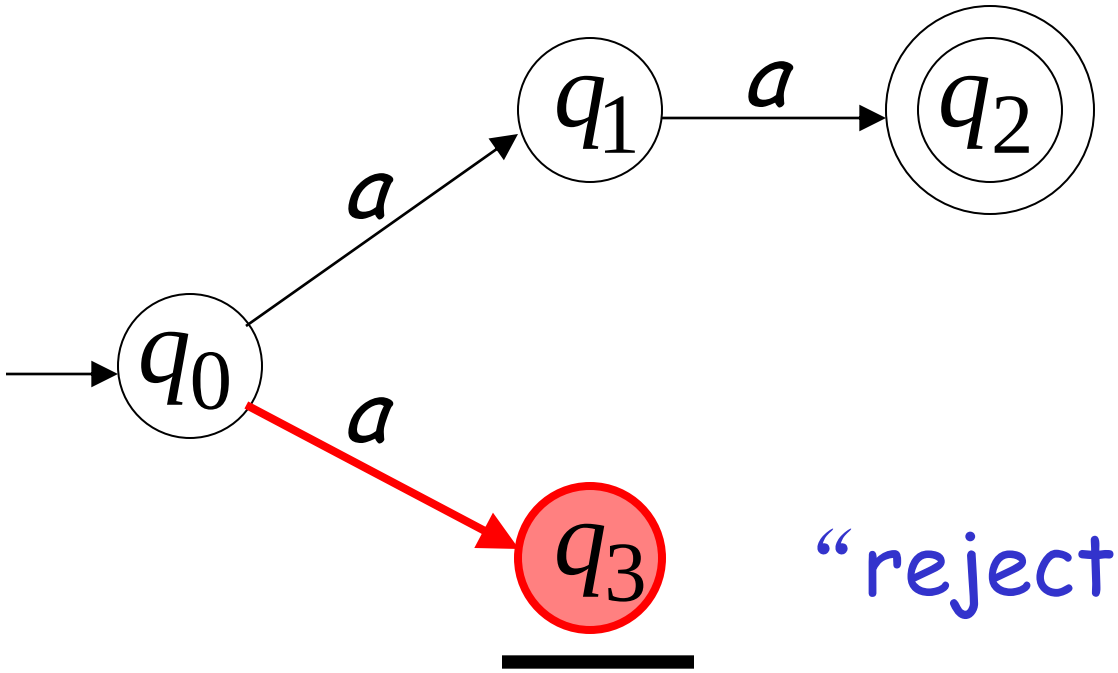


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



"reject"

An NFA accepts a string:

when there is a computation of the NFA

that accepts the string

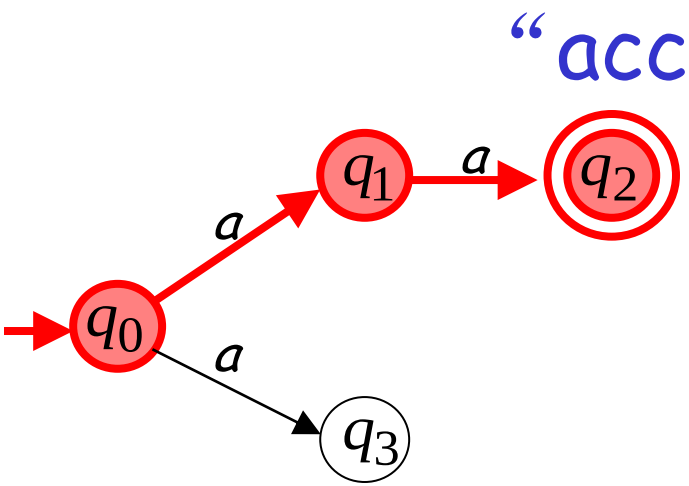
AND

all the input is consumed and the automaton

is in a final state

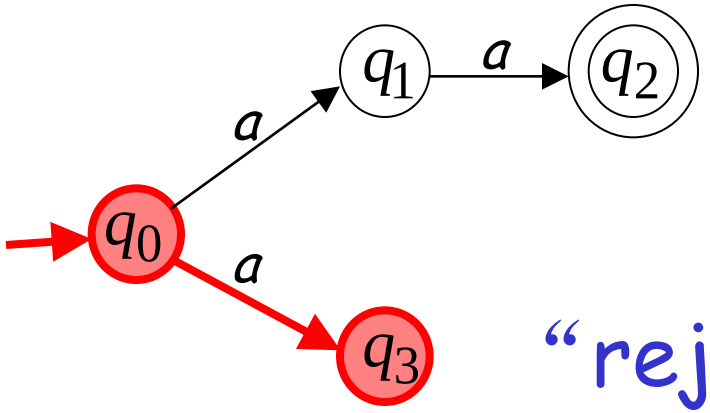
Example

aa is accepted by the NFA:



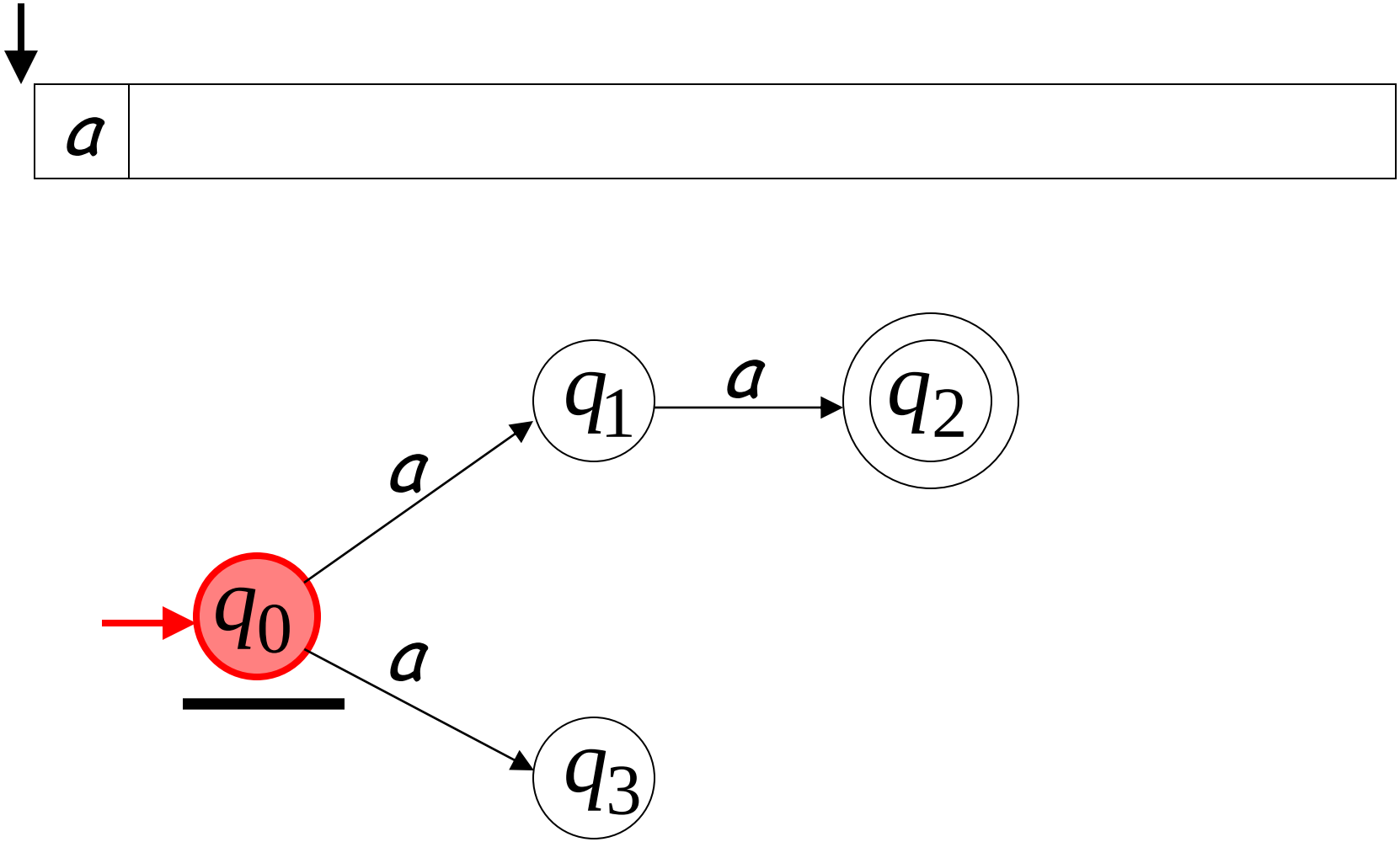
"accept"

because this computation accepts aa

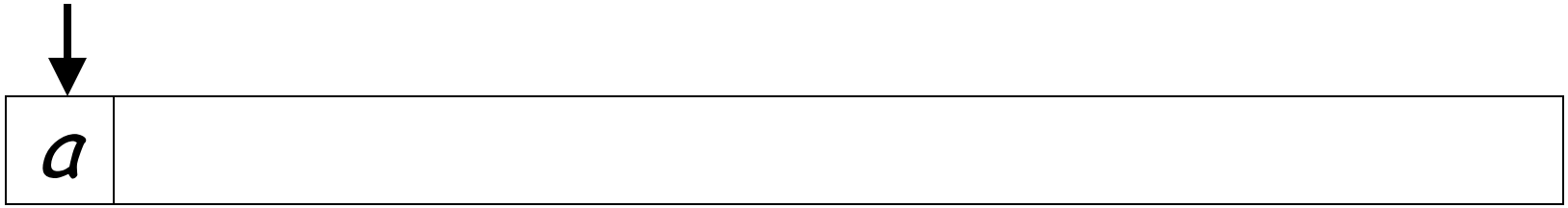


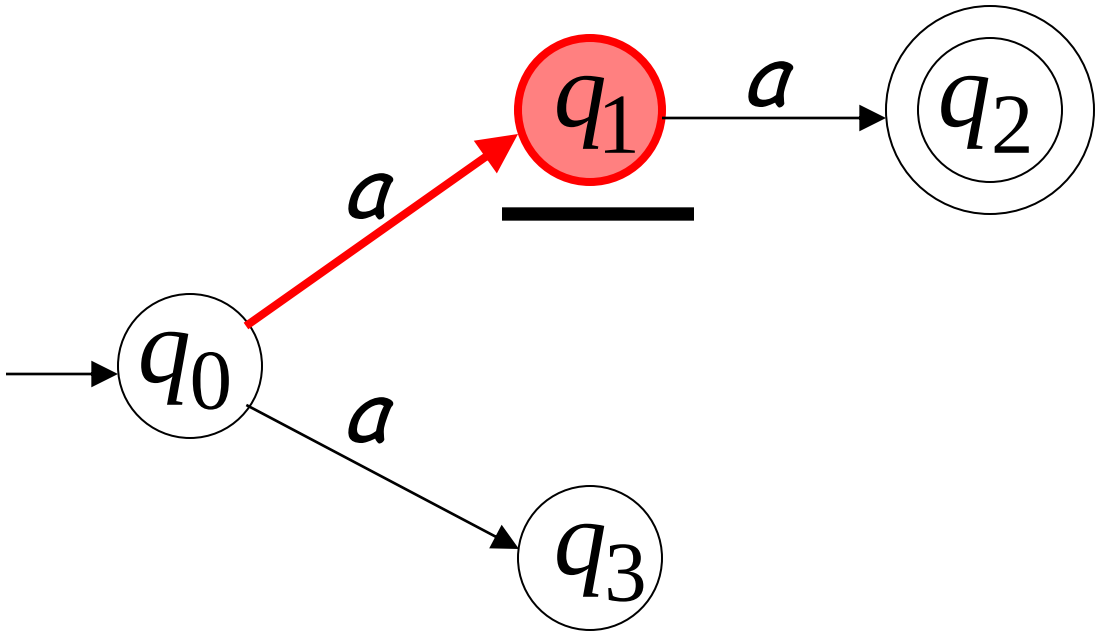
"reject

Rejection example

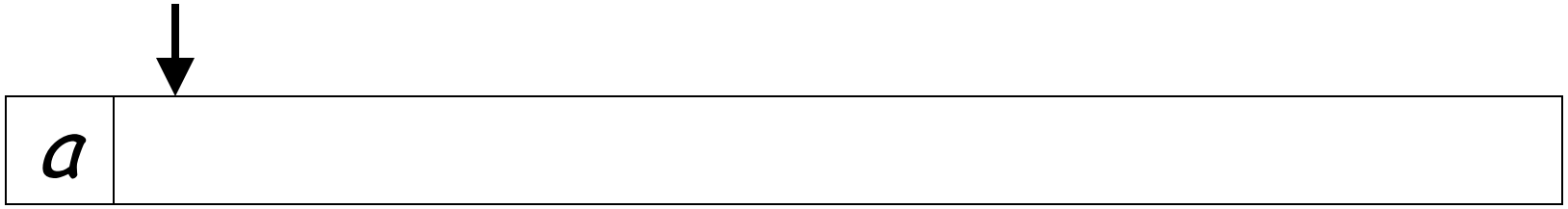


First Choice

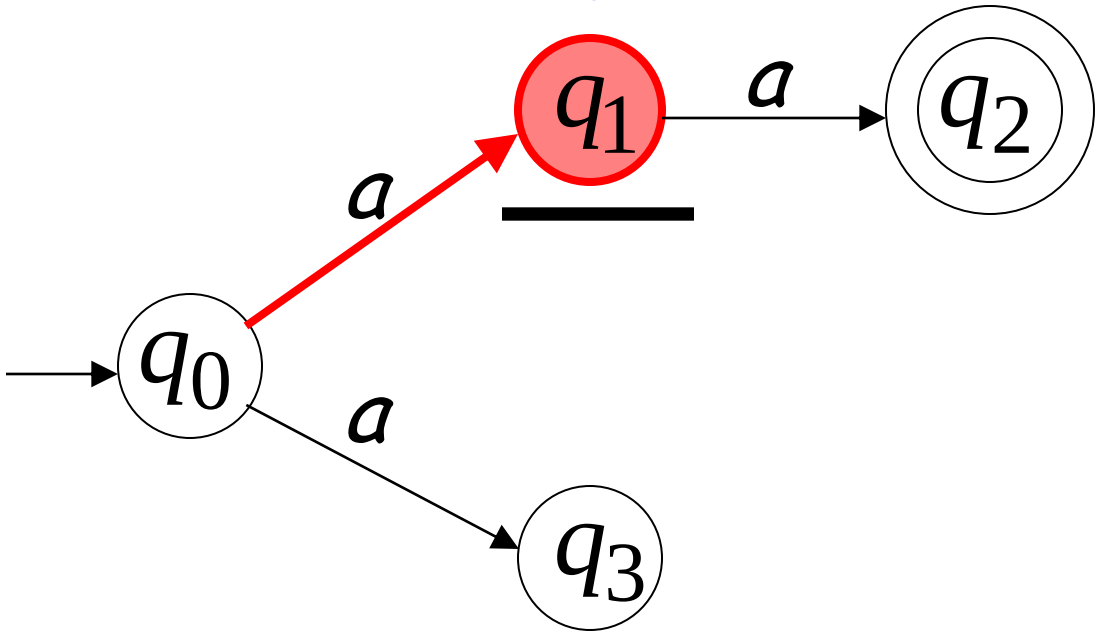




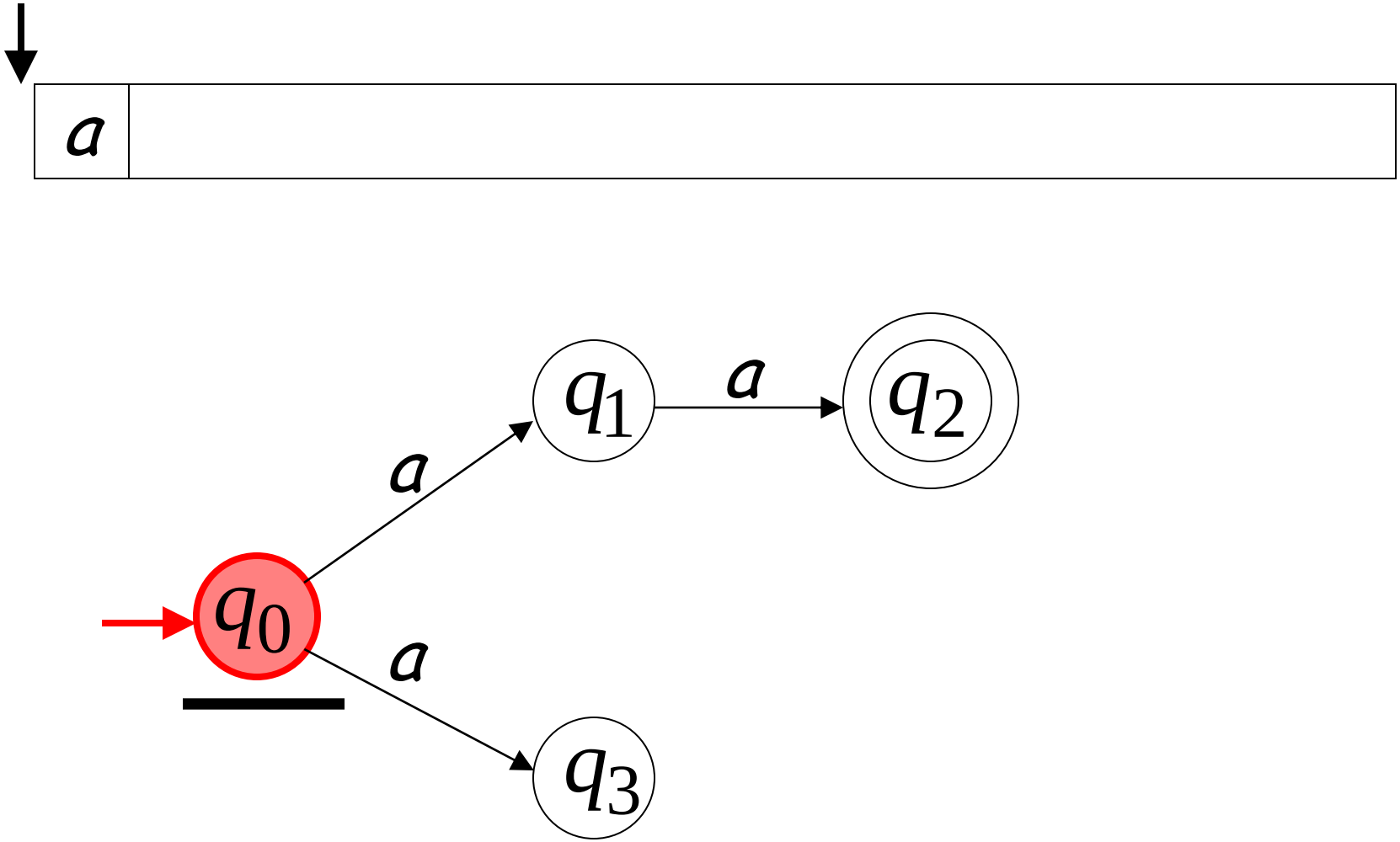
First Choice



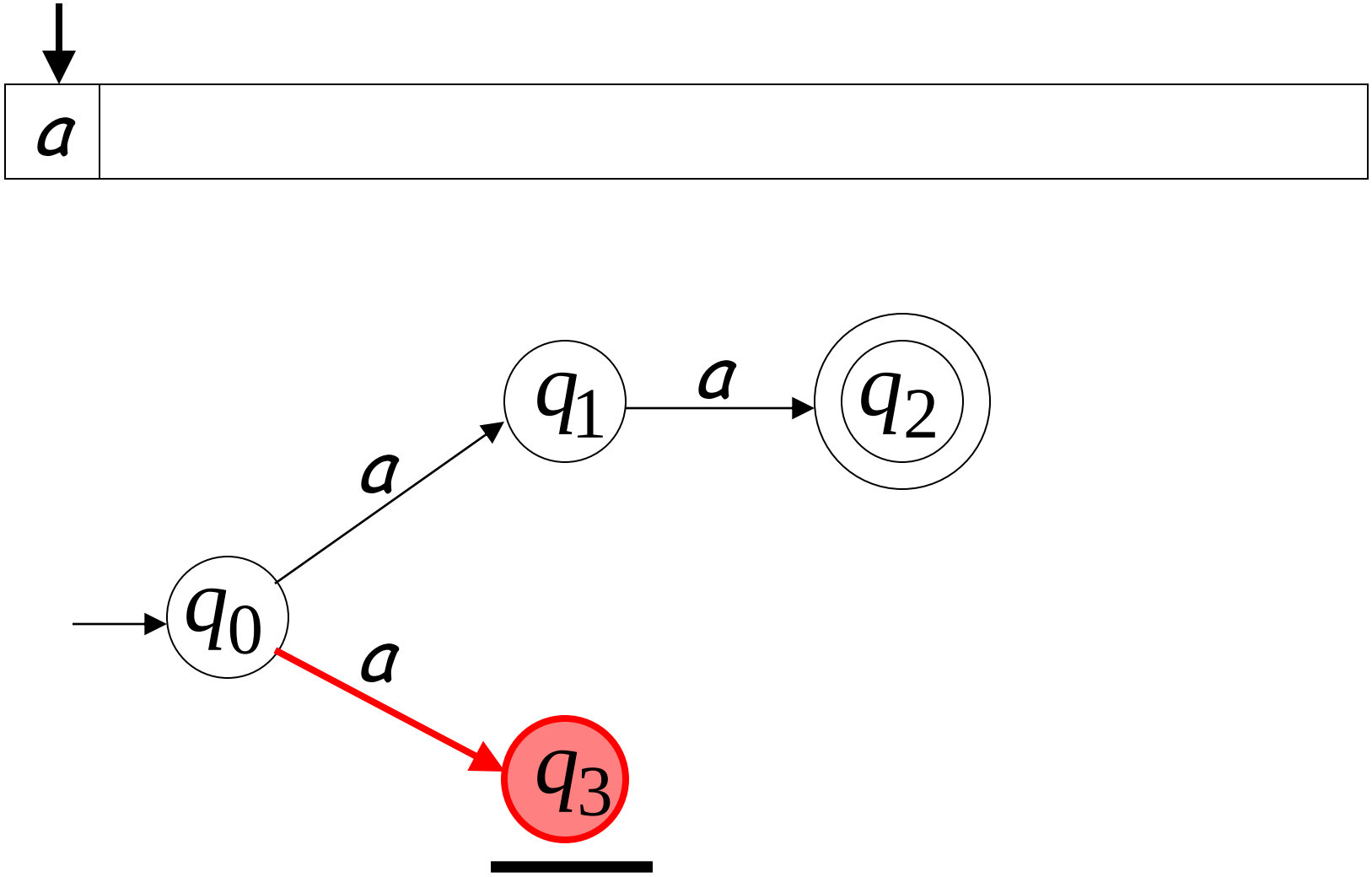
"reject"



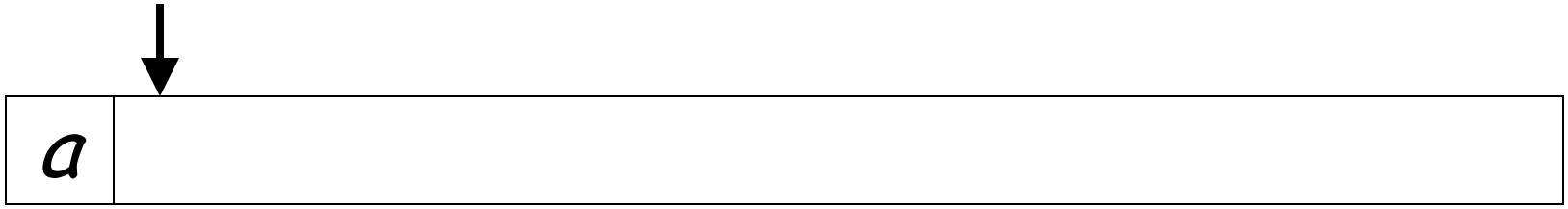
Second Choice

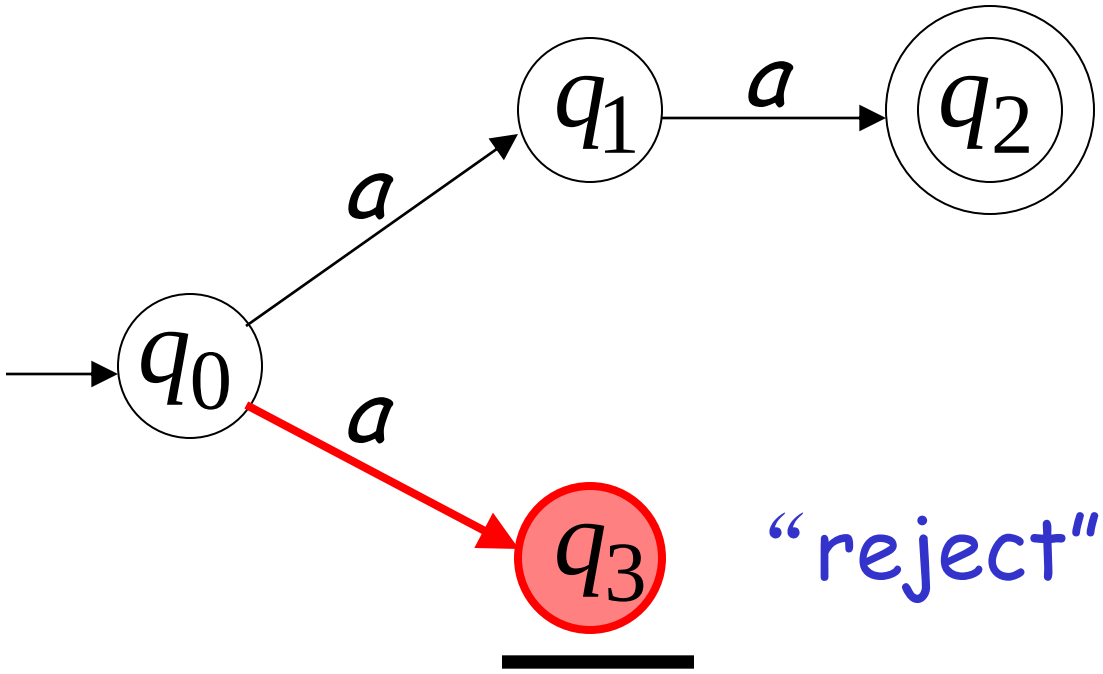


Second Choice



Second Choice





An NFA rejects a string:

when there is no computation of the NFA that accepts the string:

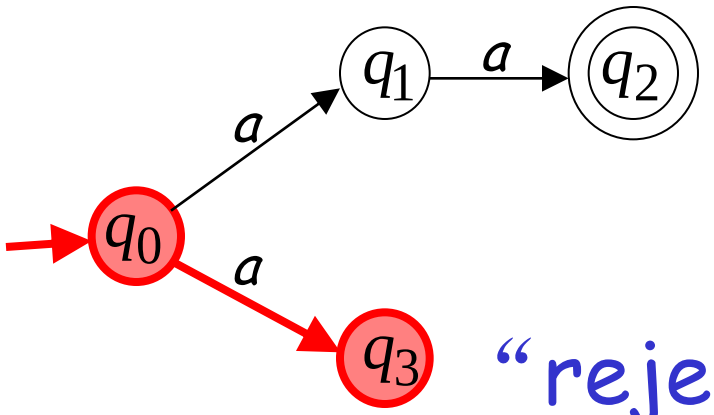
- All the input is consumed and the automaton is in a non final state

OR

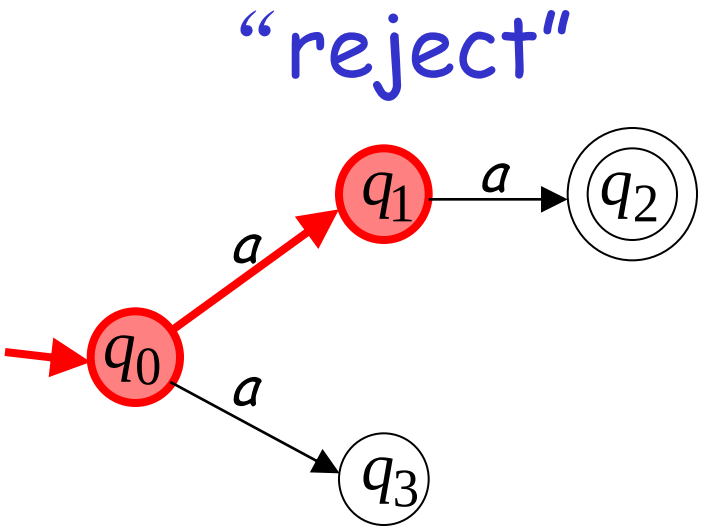
- The input cannot be consumed

Example

a is rejected by the NFA:



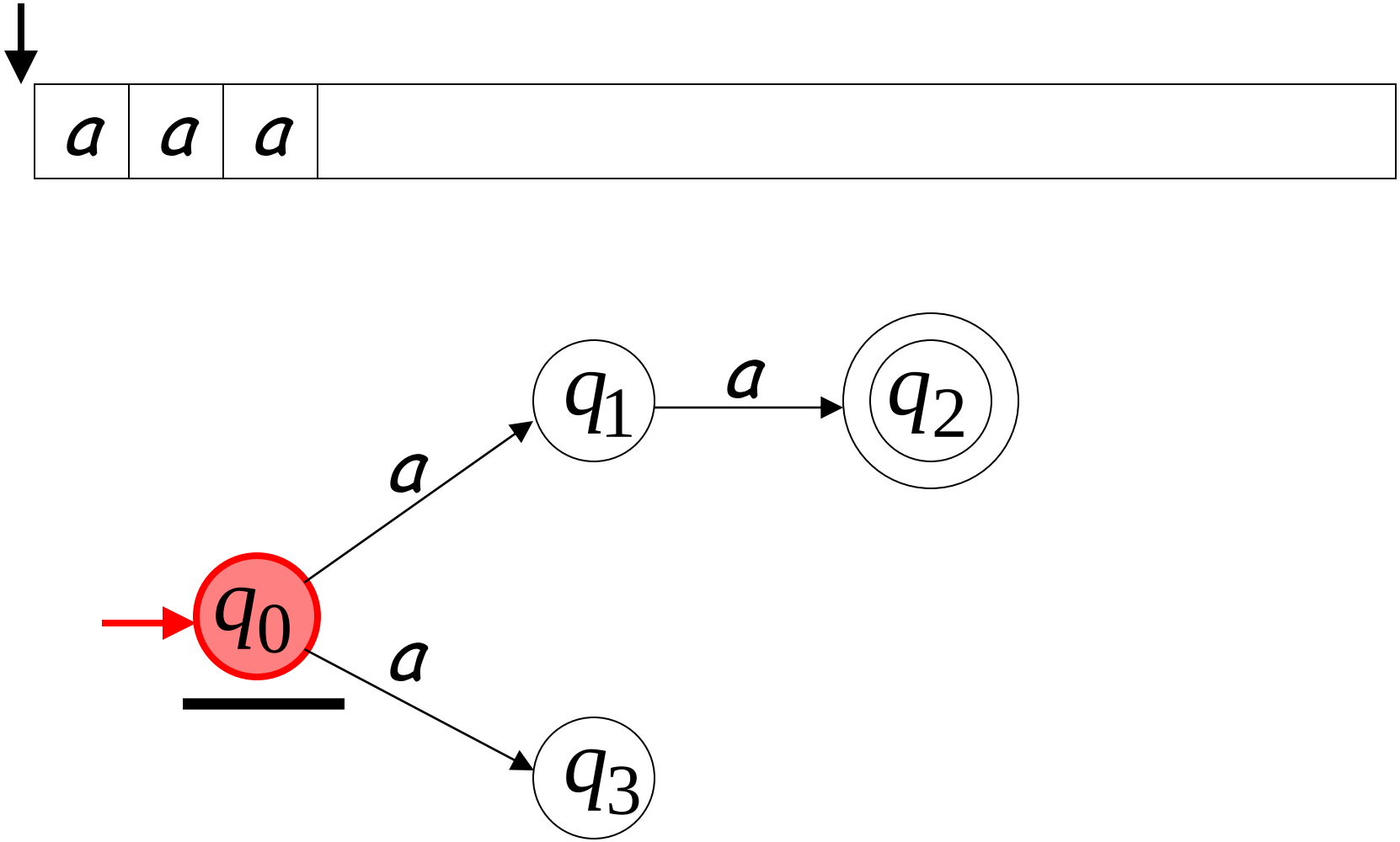
"reject"



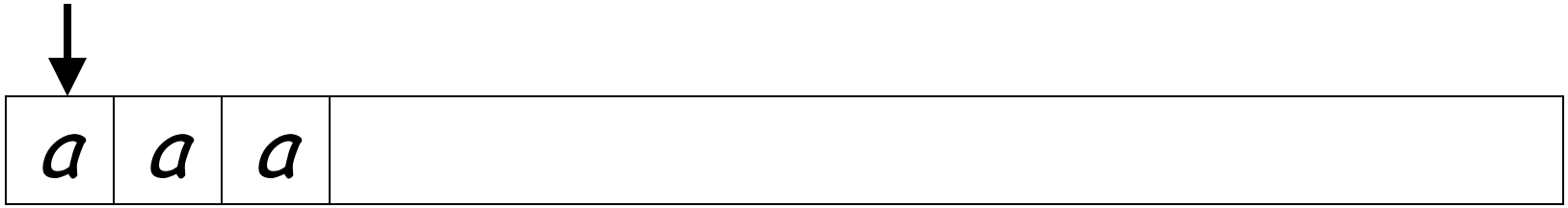
"reject"

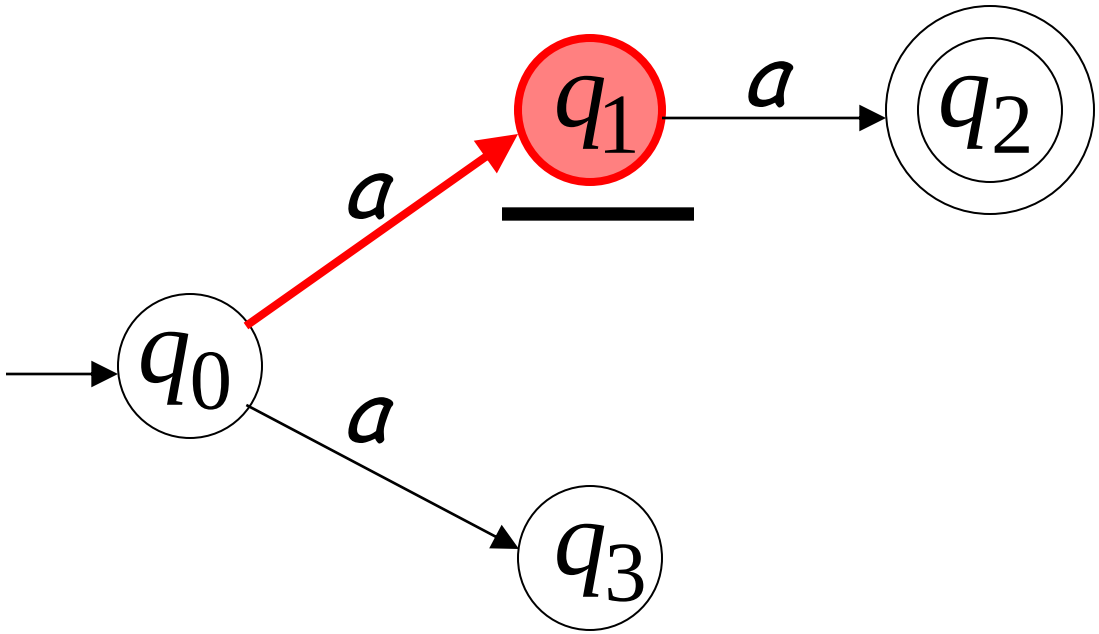
All possible computations lead to rejection

Rejection example

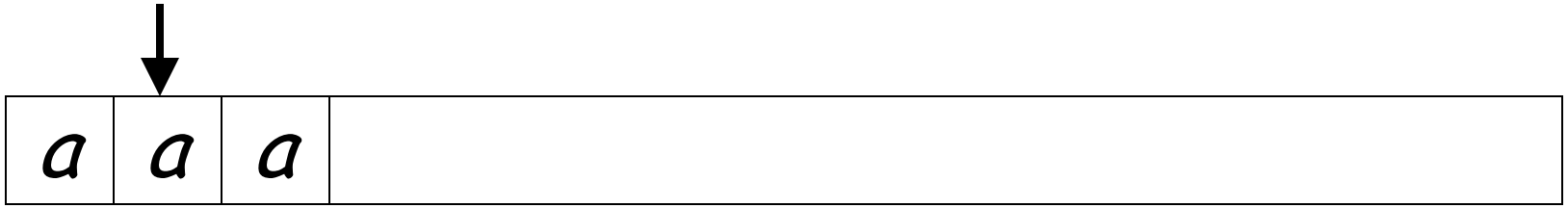


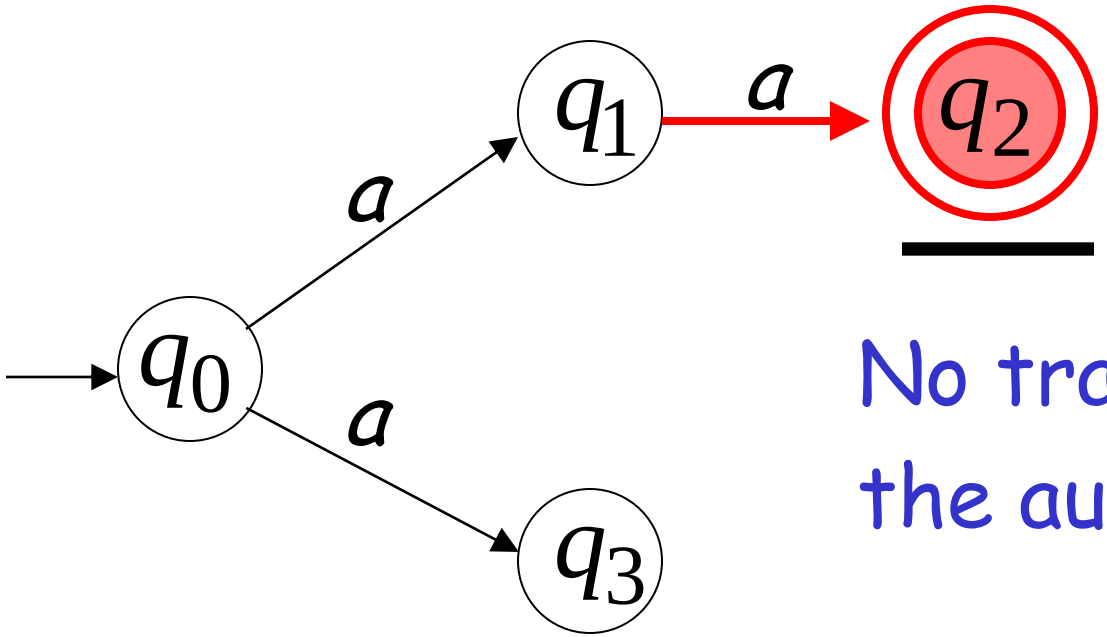
First Choice





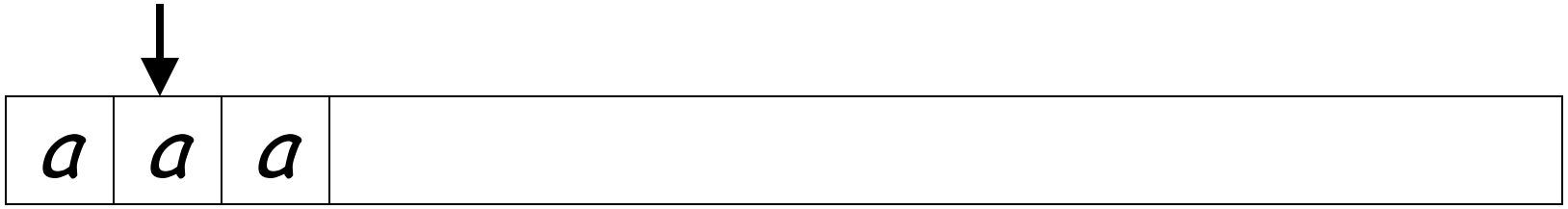
First Choice



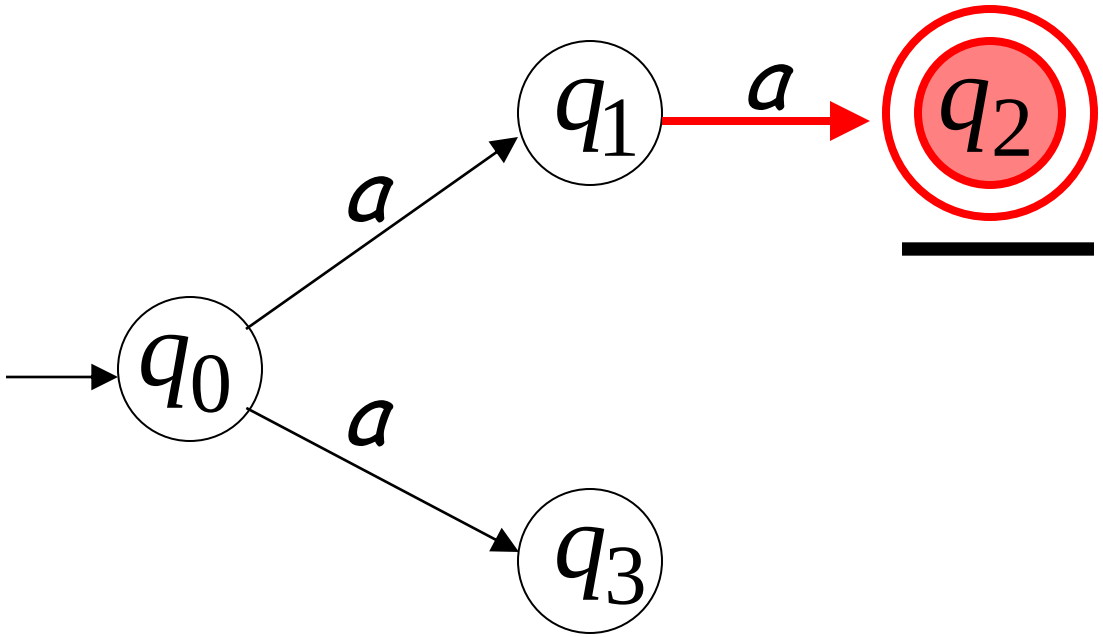


No transition:
the automaton hangs

First Choice

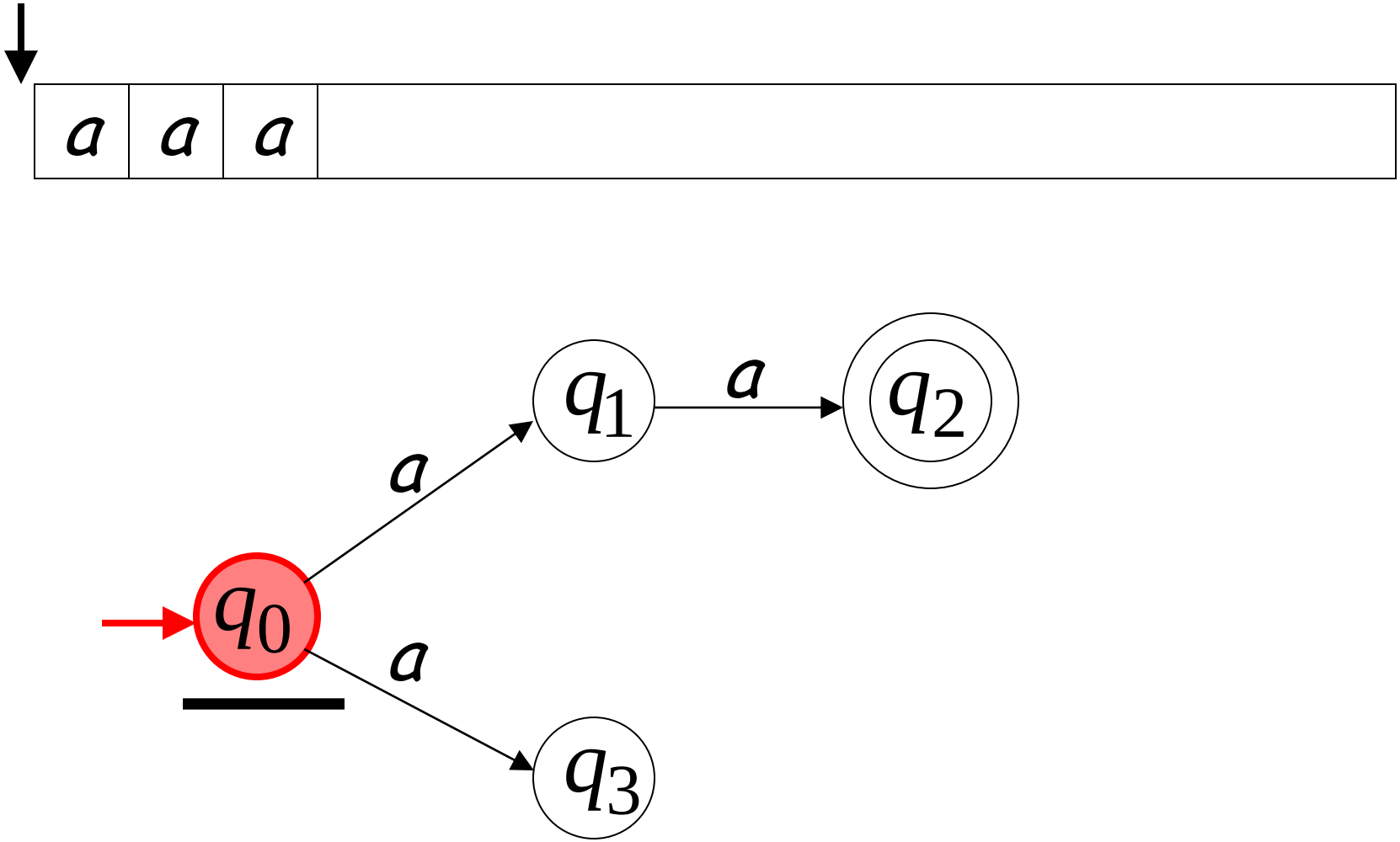


Input cannot be consumed

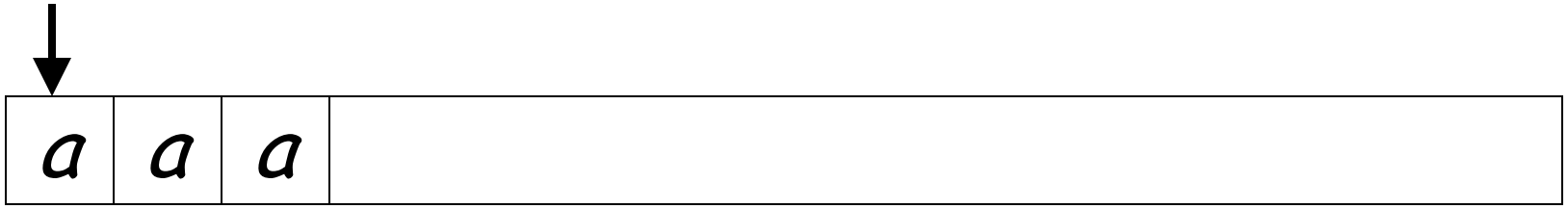


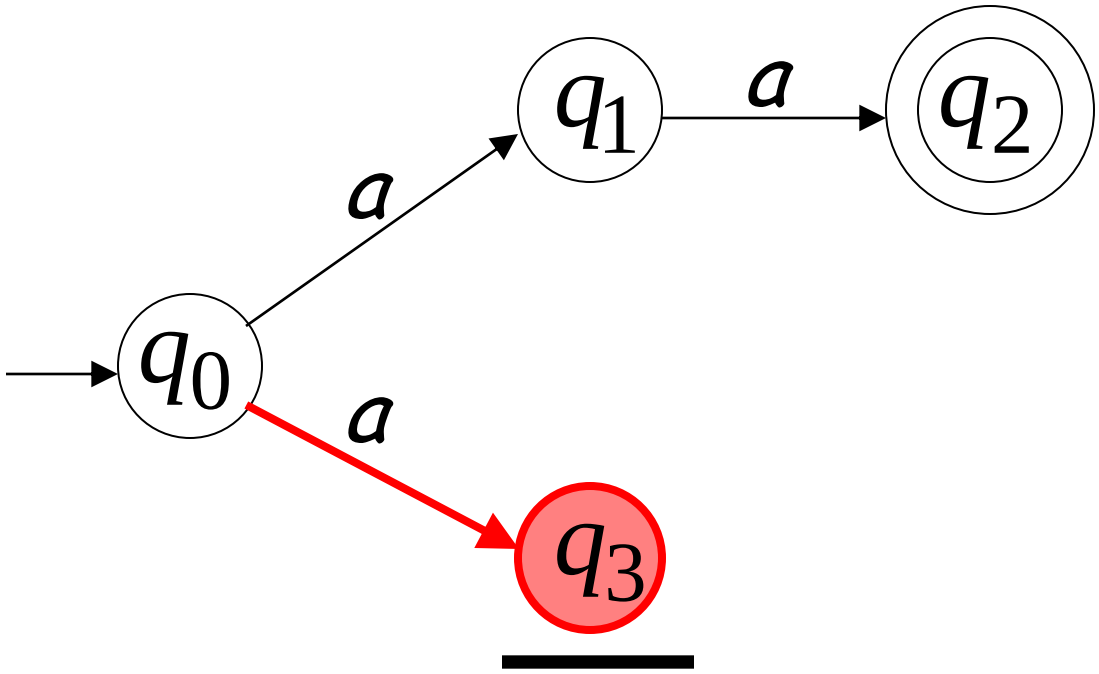
"reject"

Second Choice

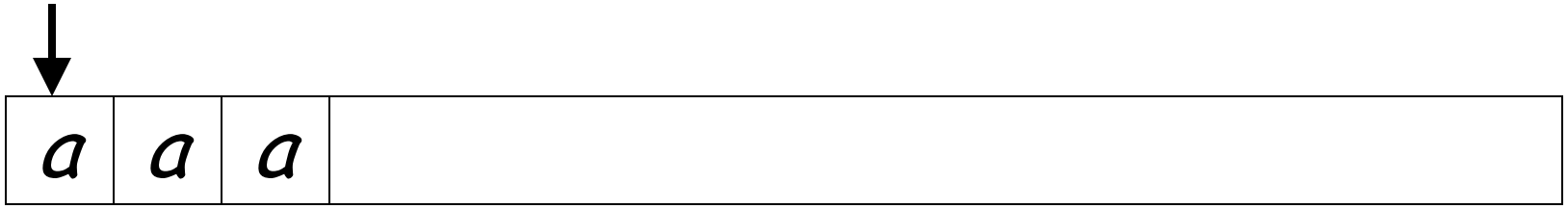


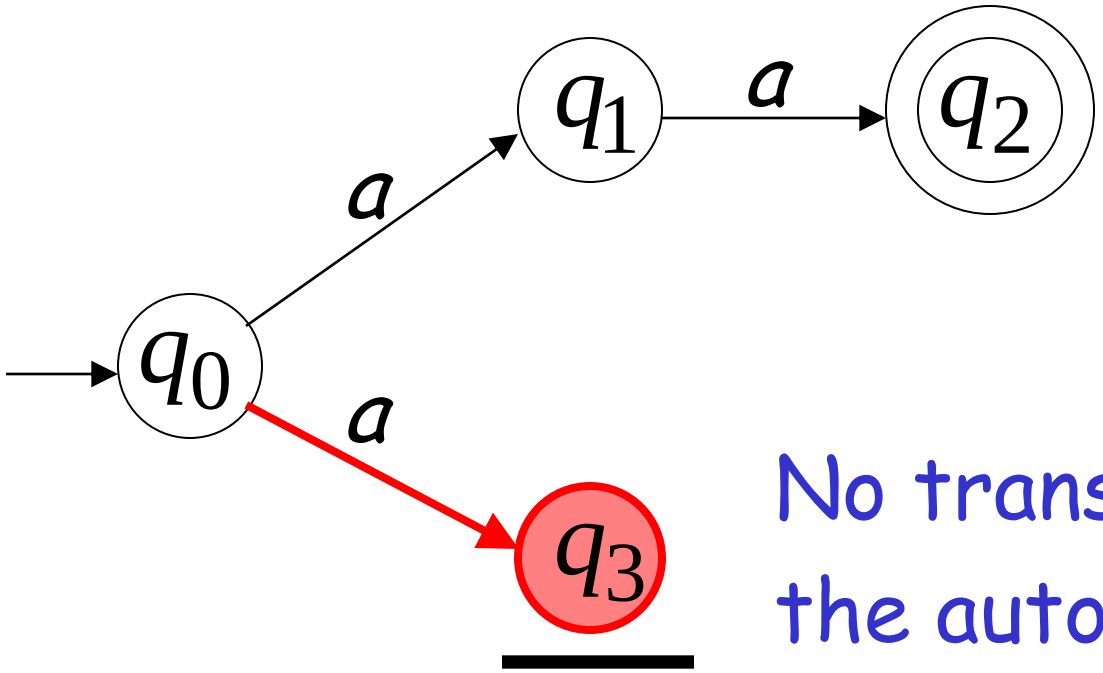
Second Choice





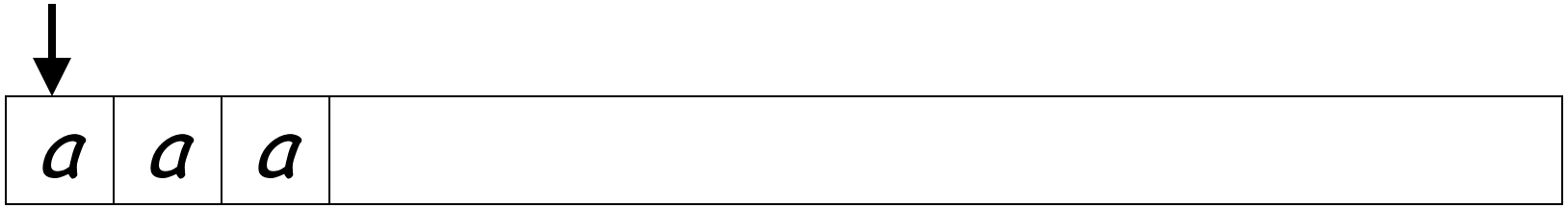
Second Choice



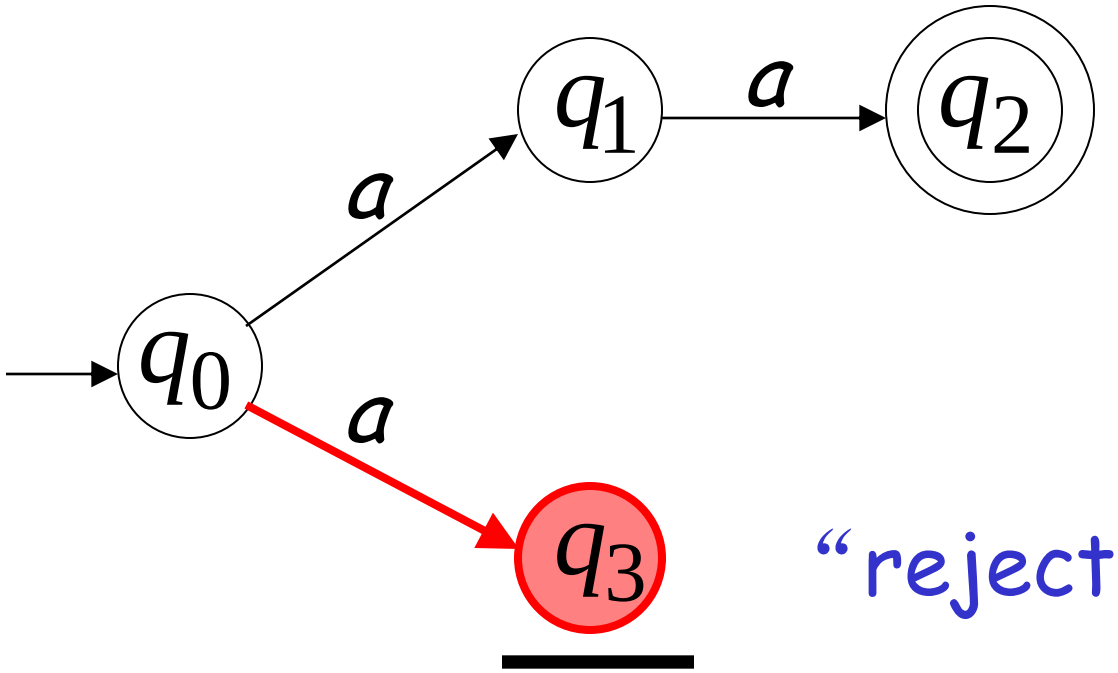


No transition:
the automaton hangs

Second Choice

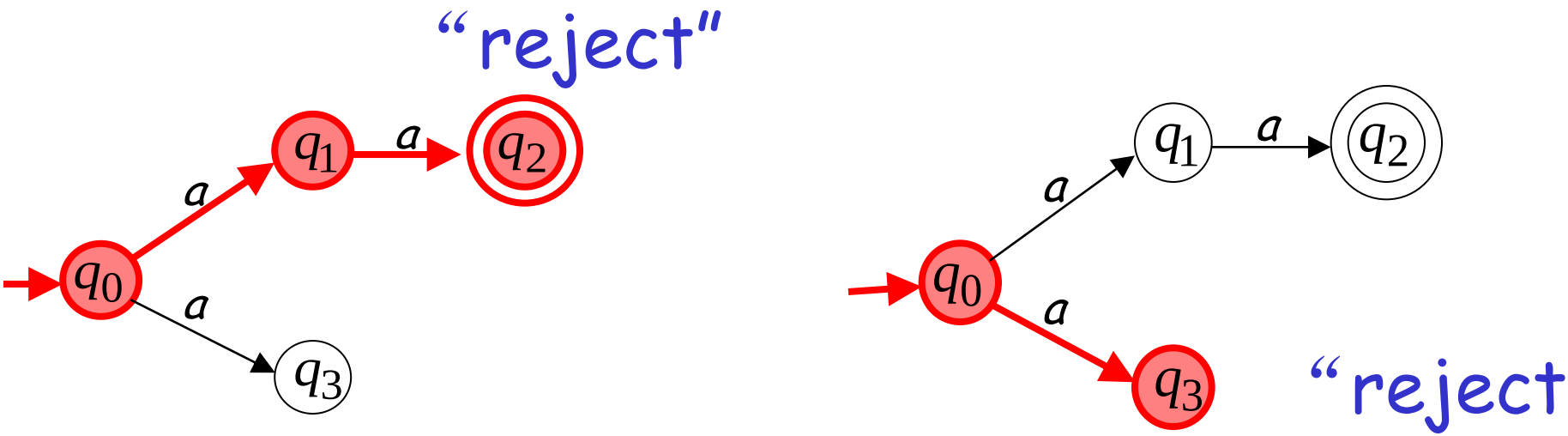


Input cannot be consumed



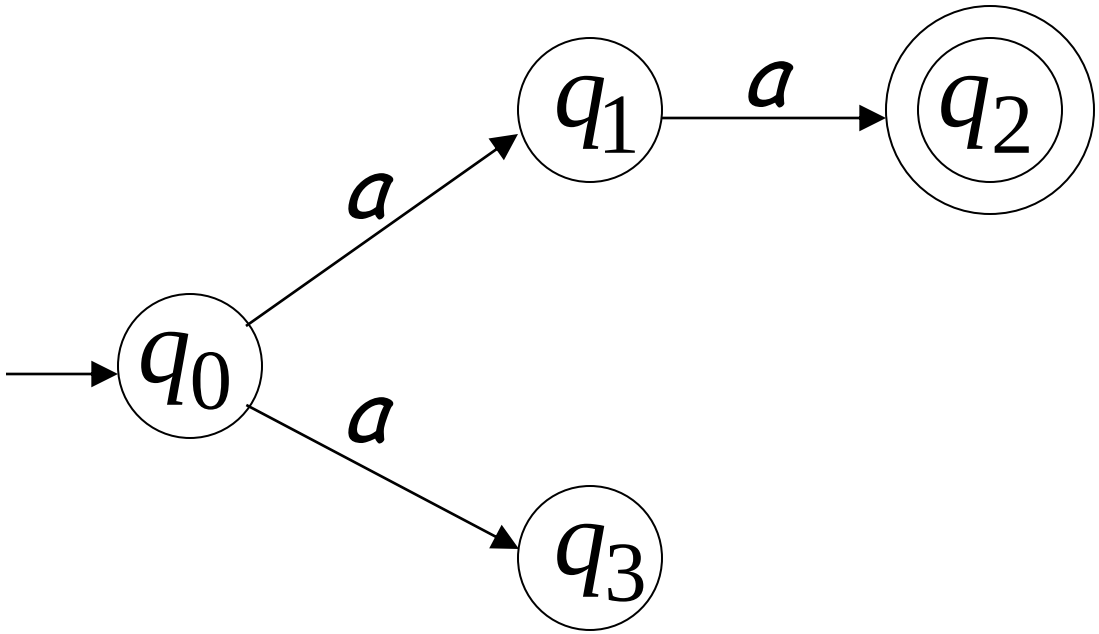
"reject"

aaa is rejected by the NFA:

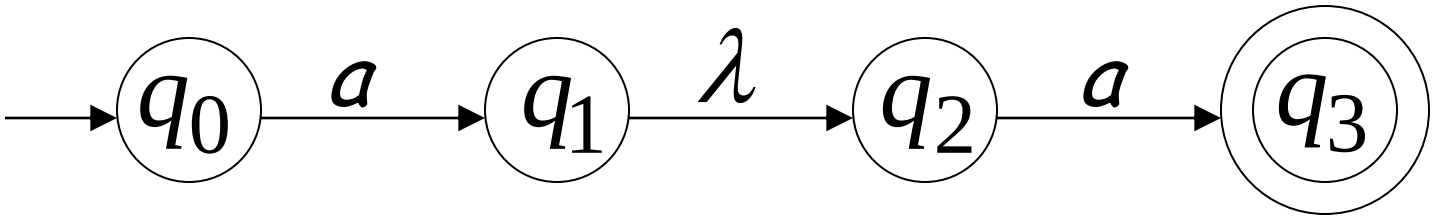


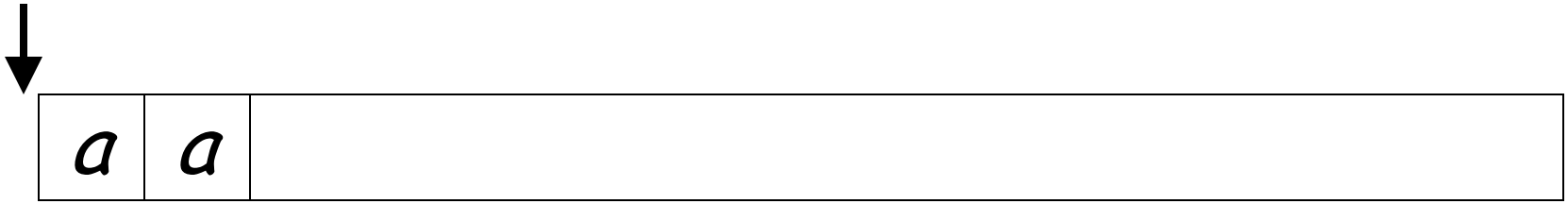
All possible computations lead to rejection

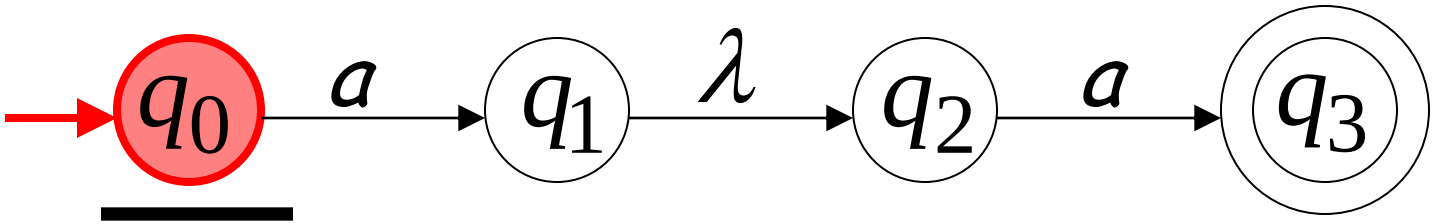
Language accepted: $L = \{aa\}$

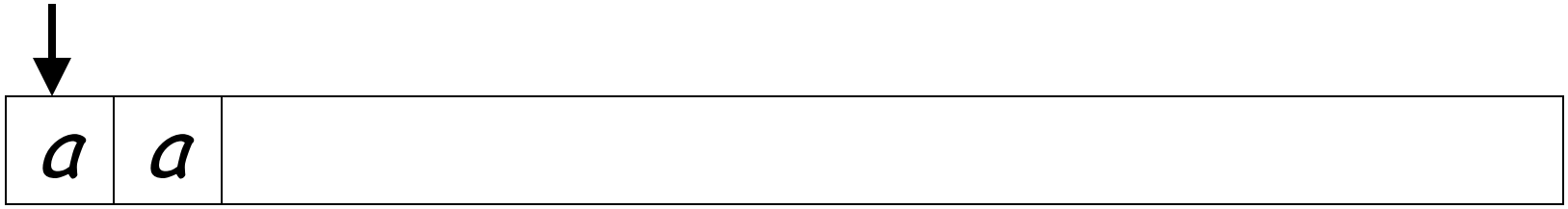


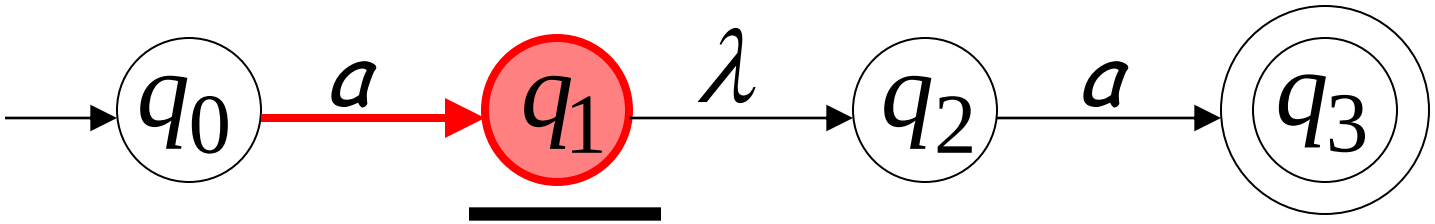
Lambda Transitions



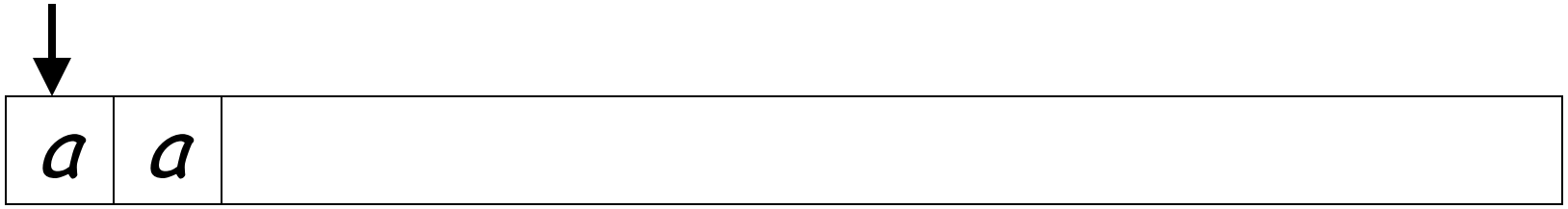

a
a

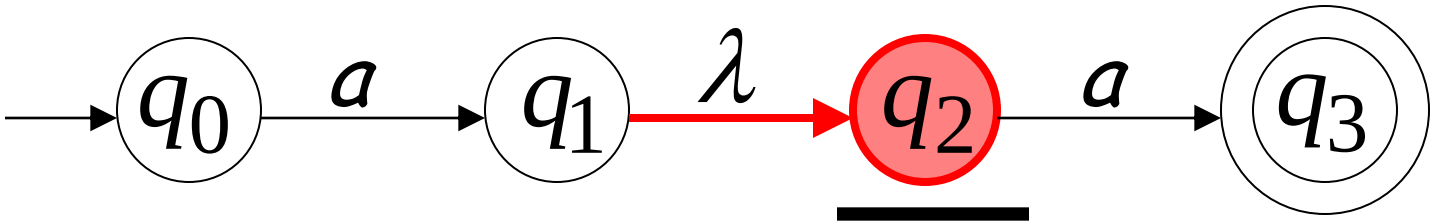

q_0
a
q_1
λ
q_2
a
q_3

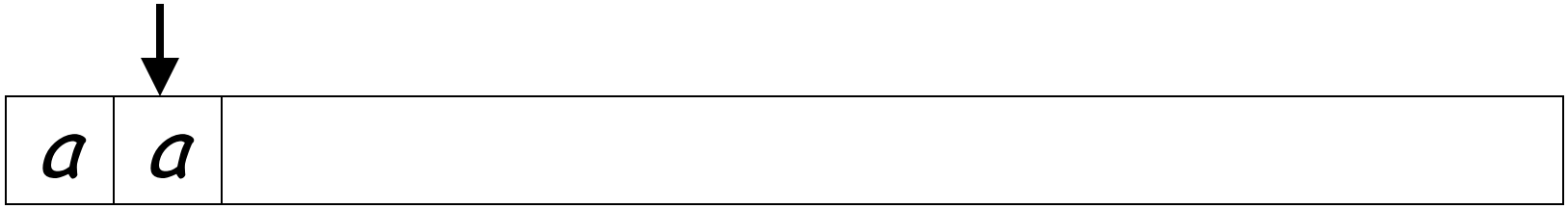

a
a

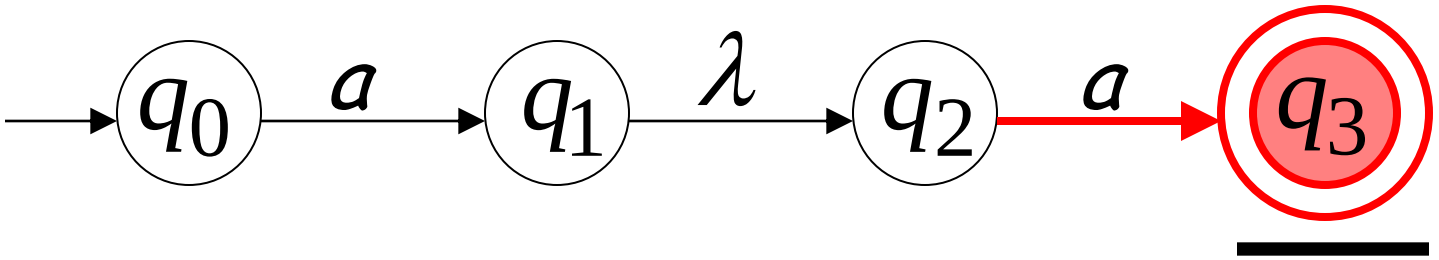

q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

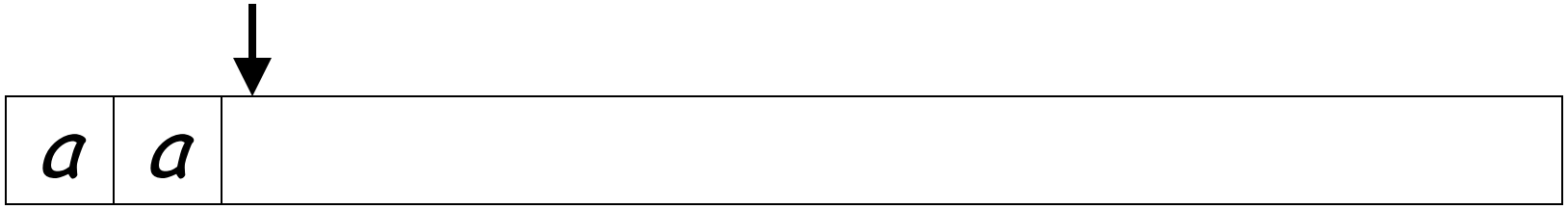




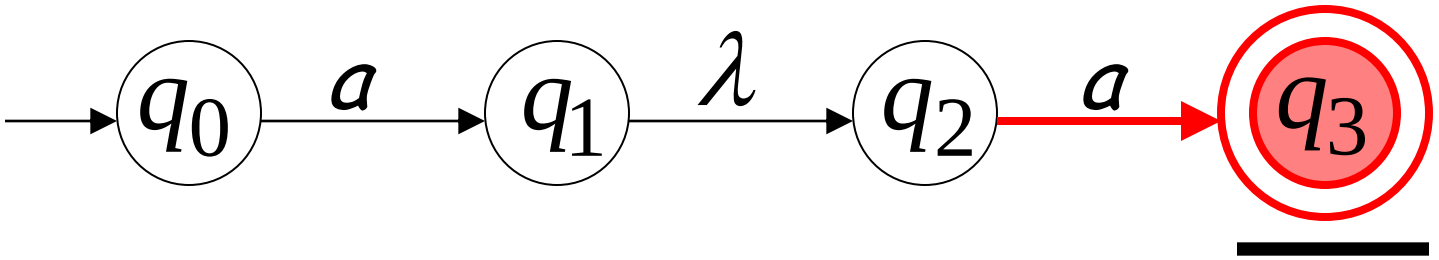

a
a


q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

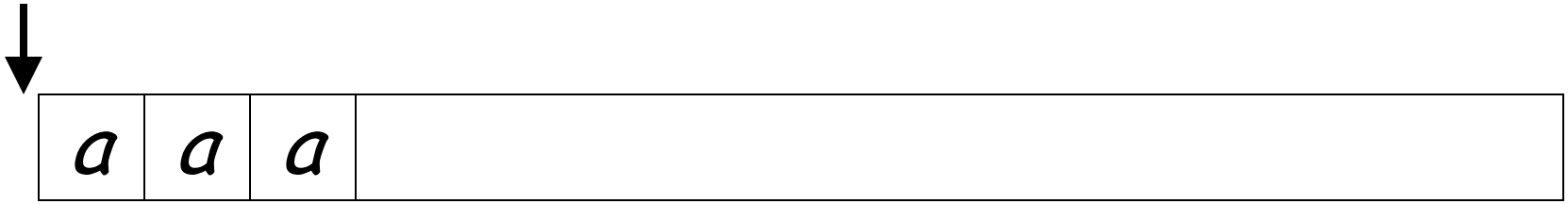


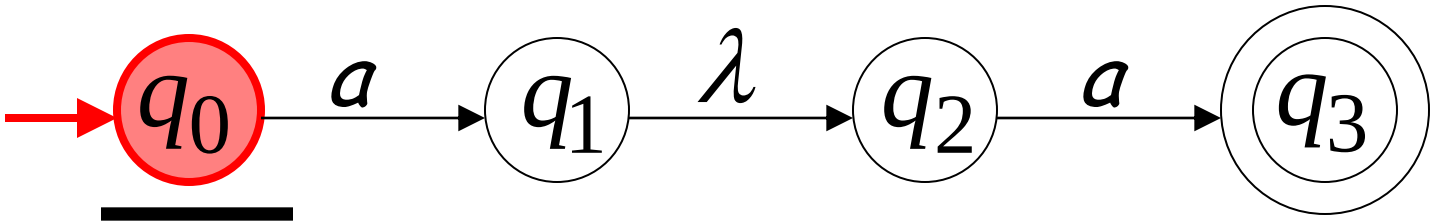
"accept"

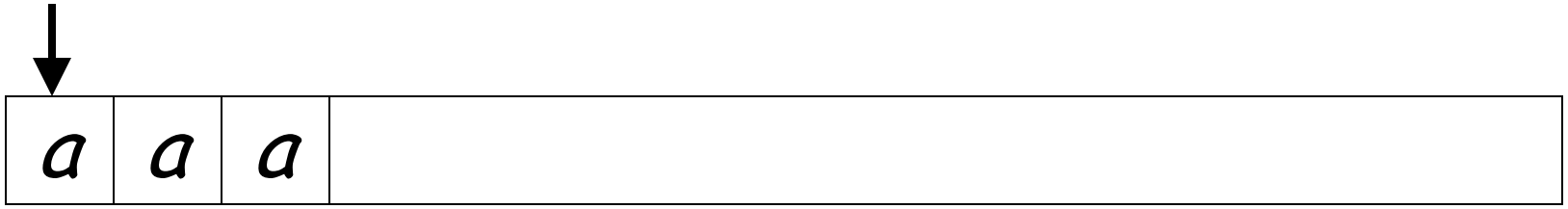


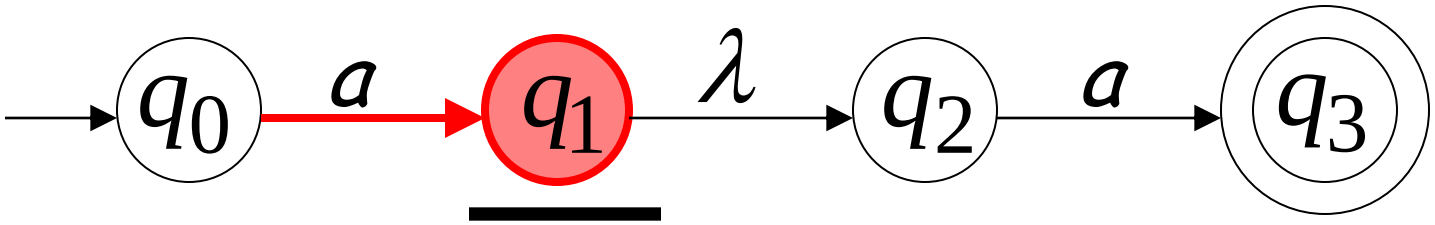
String aa is accepted

Rejection Example

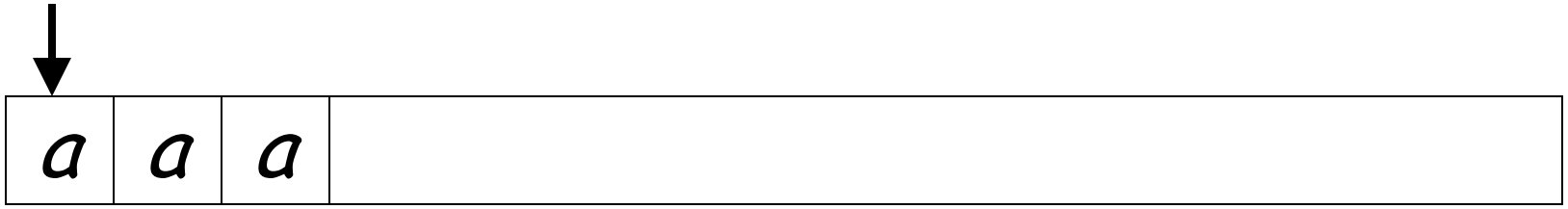


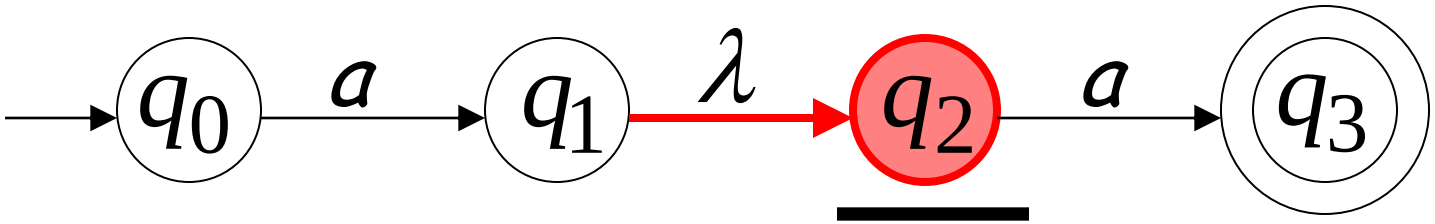


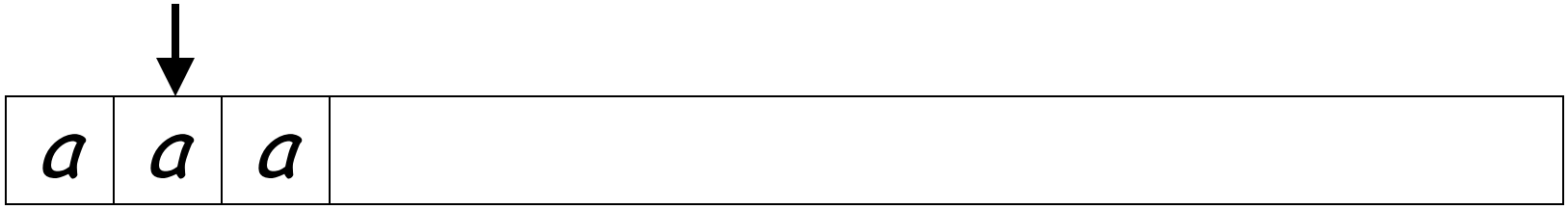

a
a
a

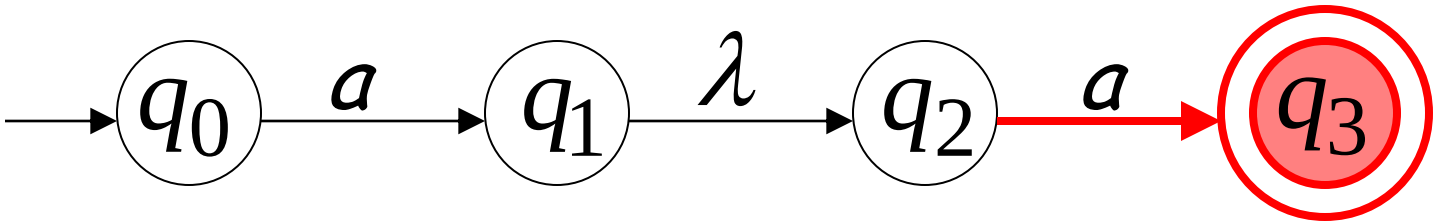

q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)





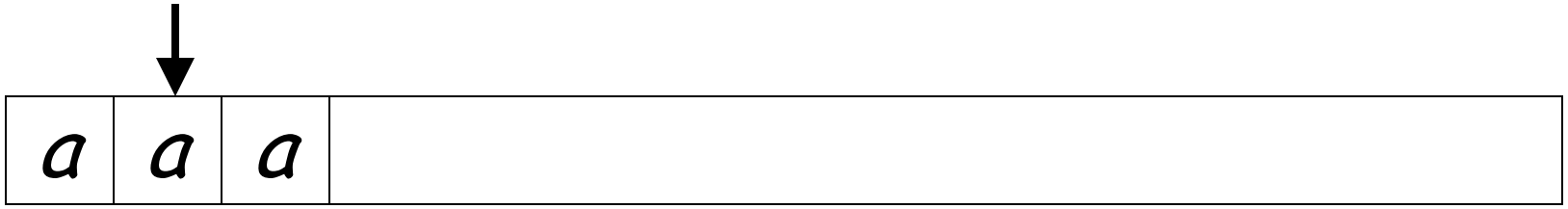




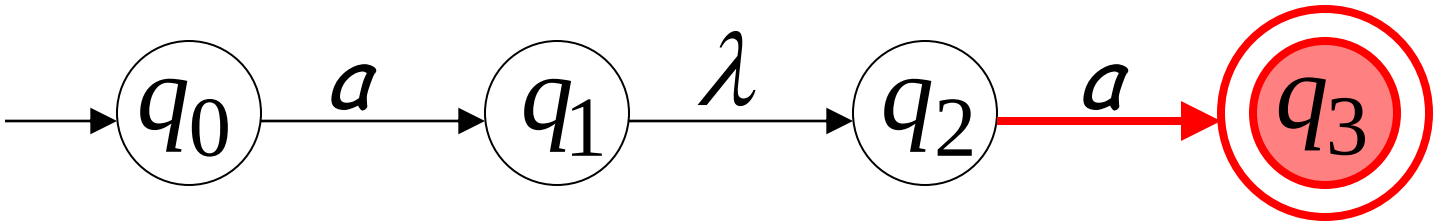
No transition:

the automaton hangs

Input cannot be consumed

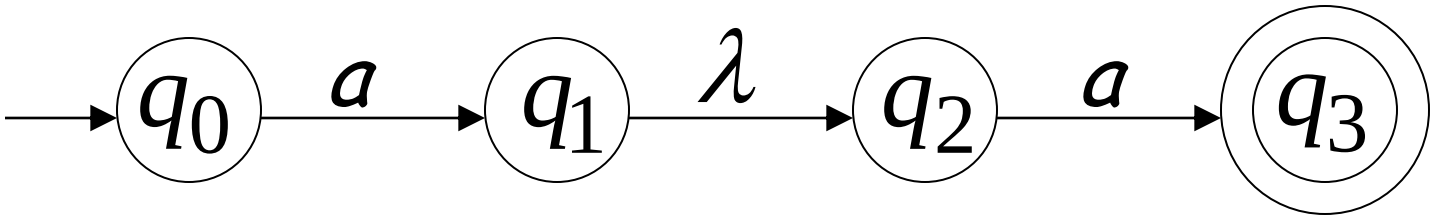


"reject"

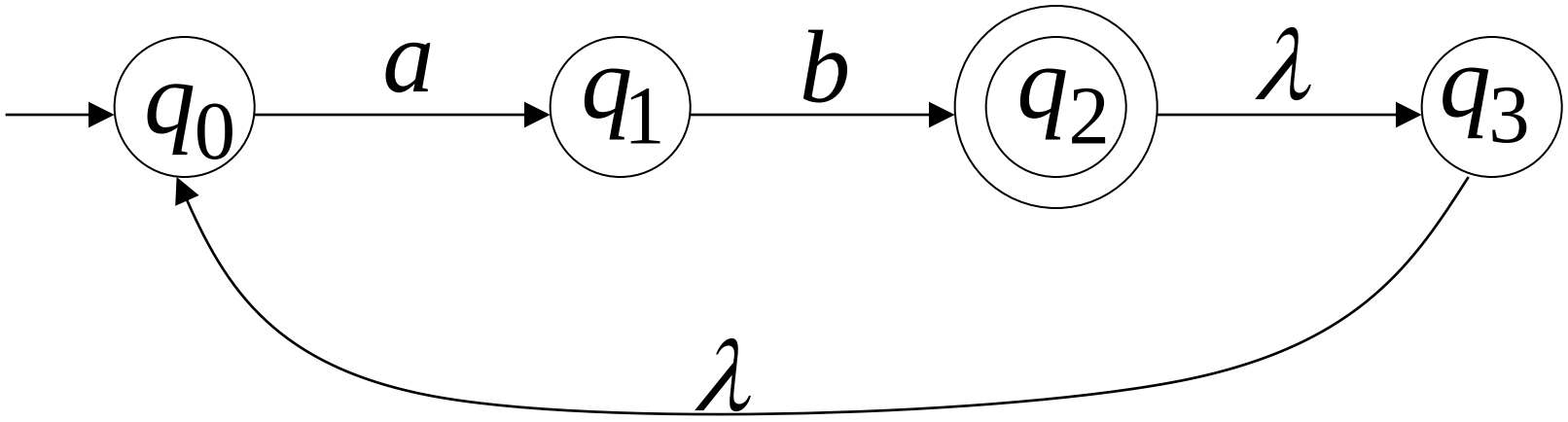


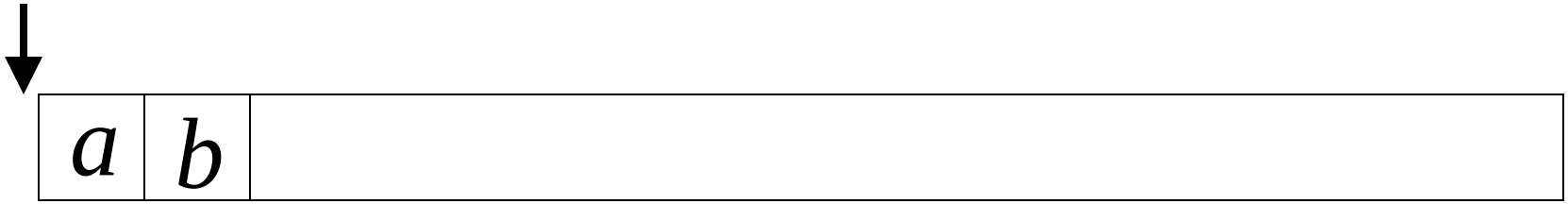
String aaa is rejected

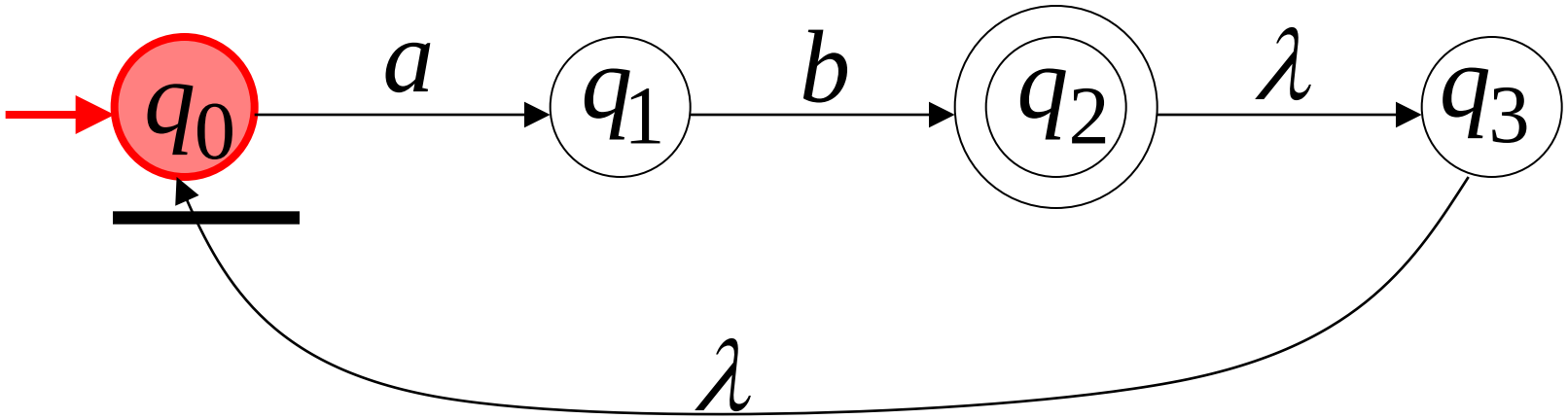
Language accepted: $L = \{aa\}$

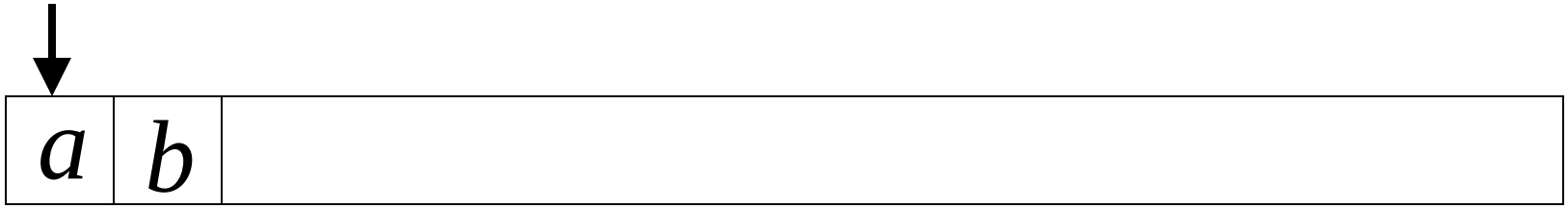


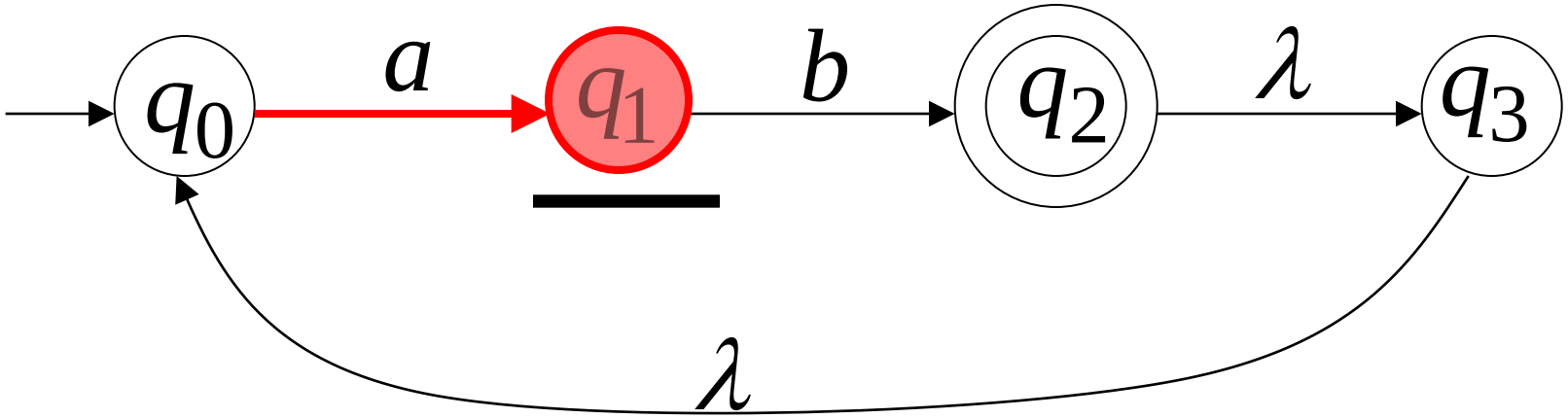
Another NFA Example

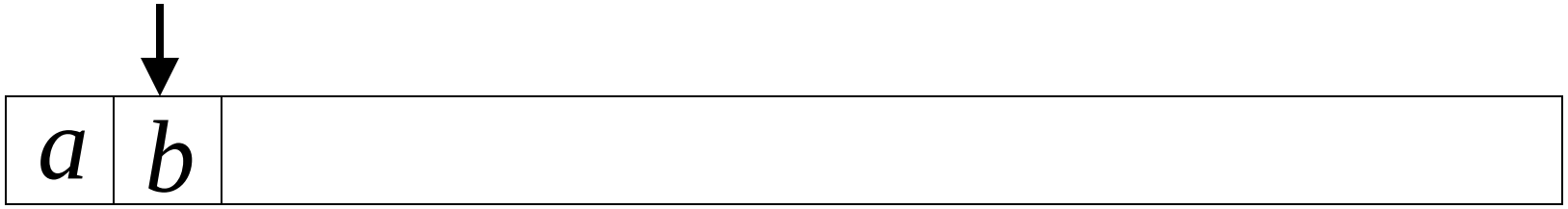


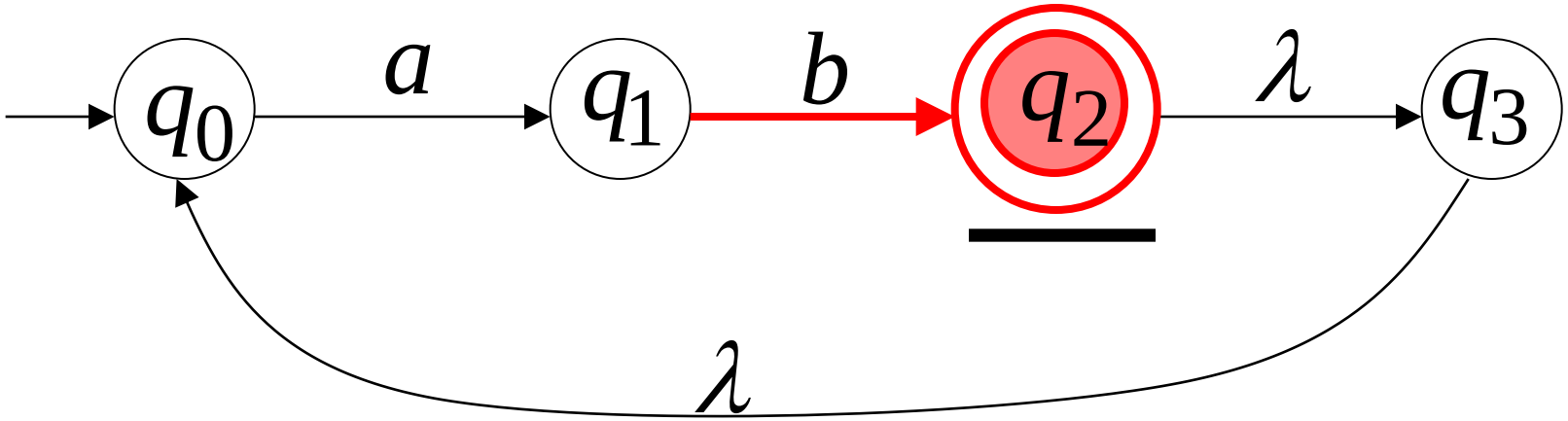

a
b

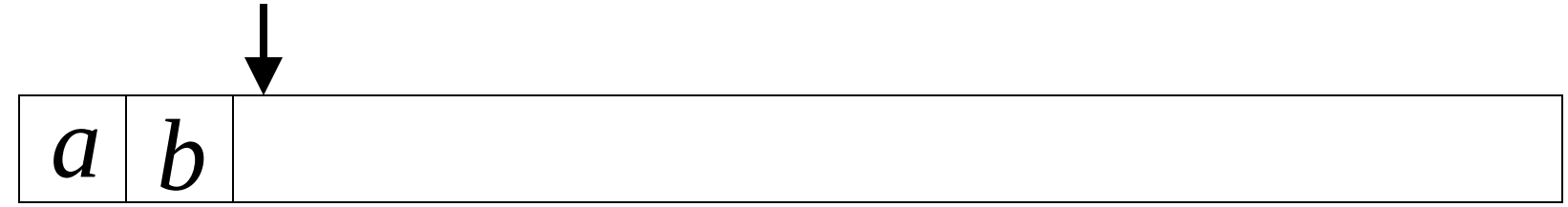

q_0
a
q_1
b
q_2
λ
q_3
λ

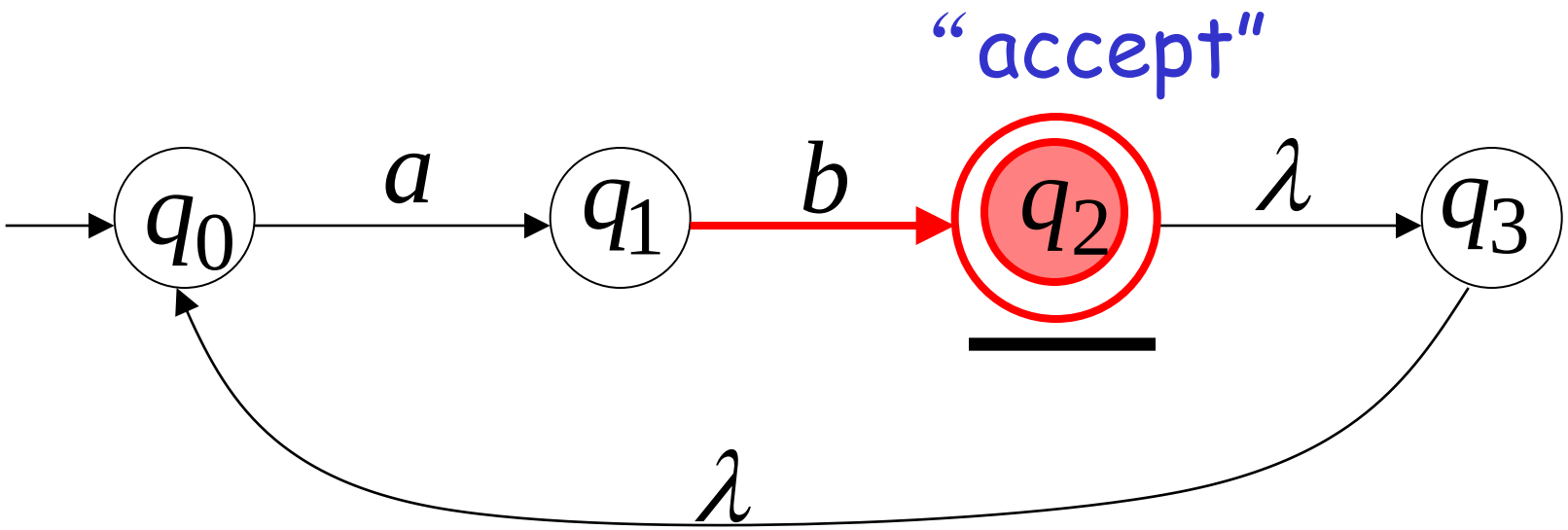

a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

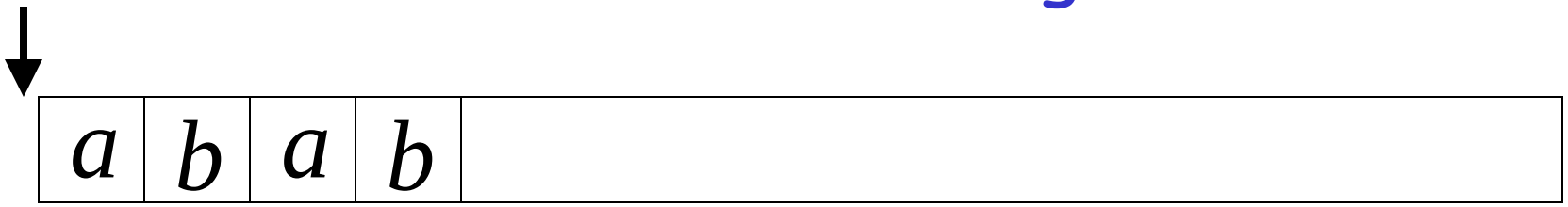

a
b

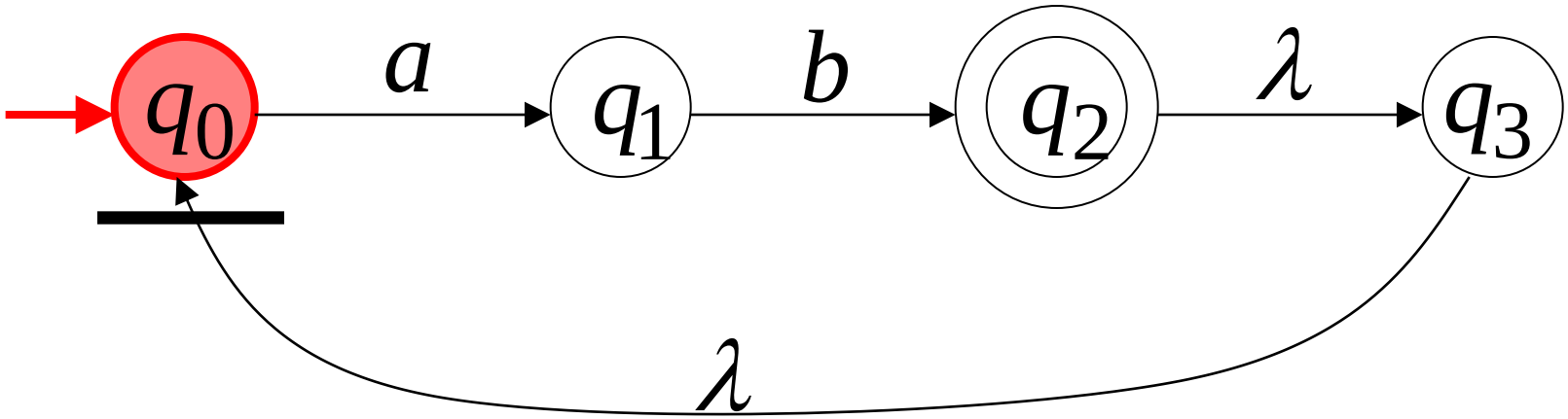

q_0
a
q_1
b
q_2
λ
q_3
λ

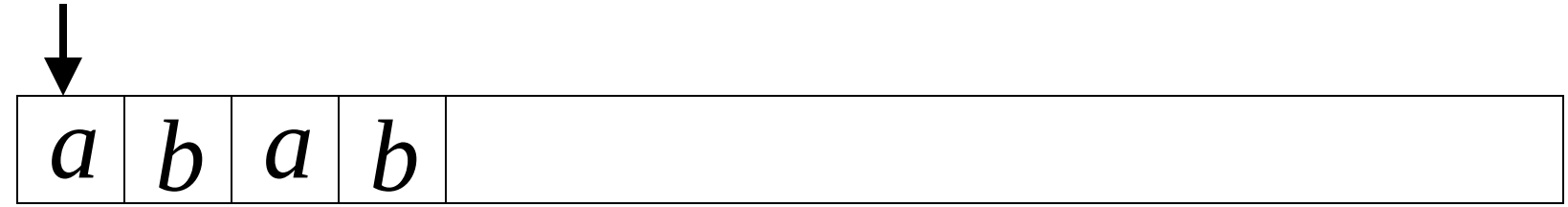

a
b

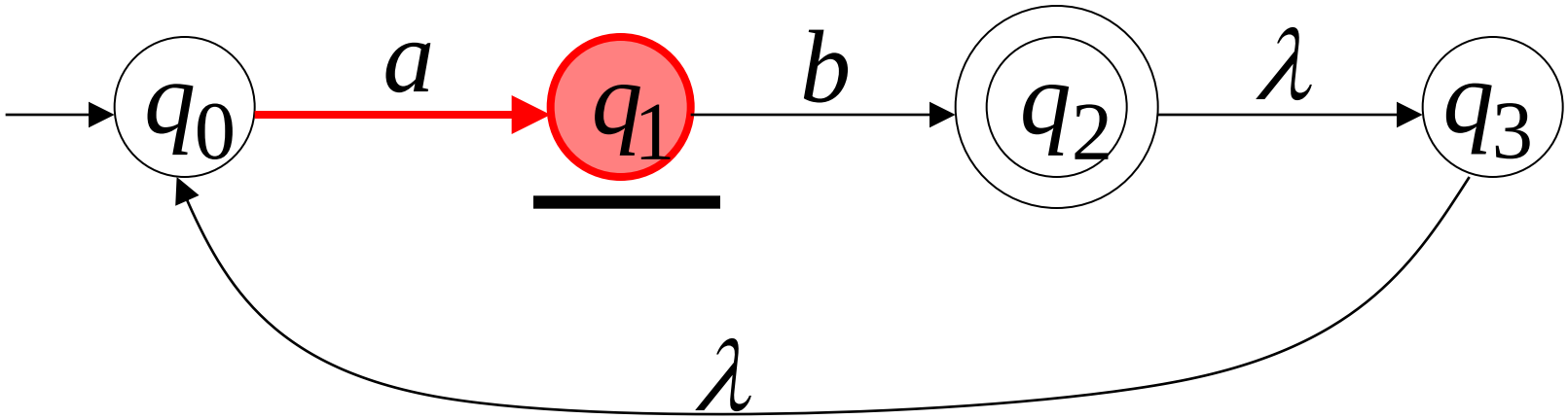

"accept"
q_0
q_1
q_2
q_3
a
b
λ
λ

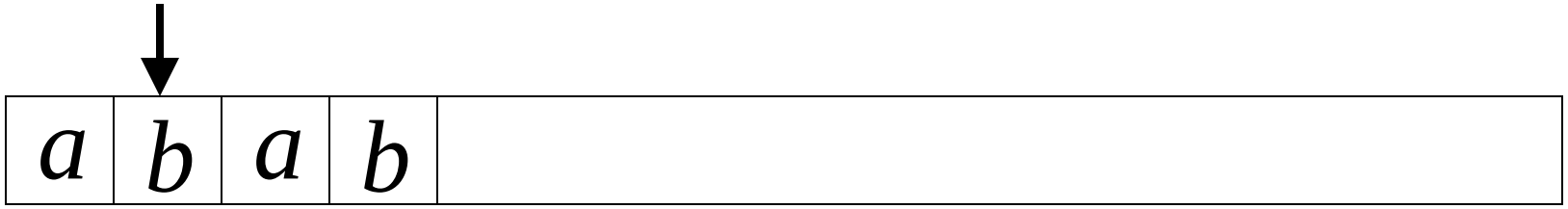
Another String

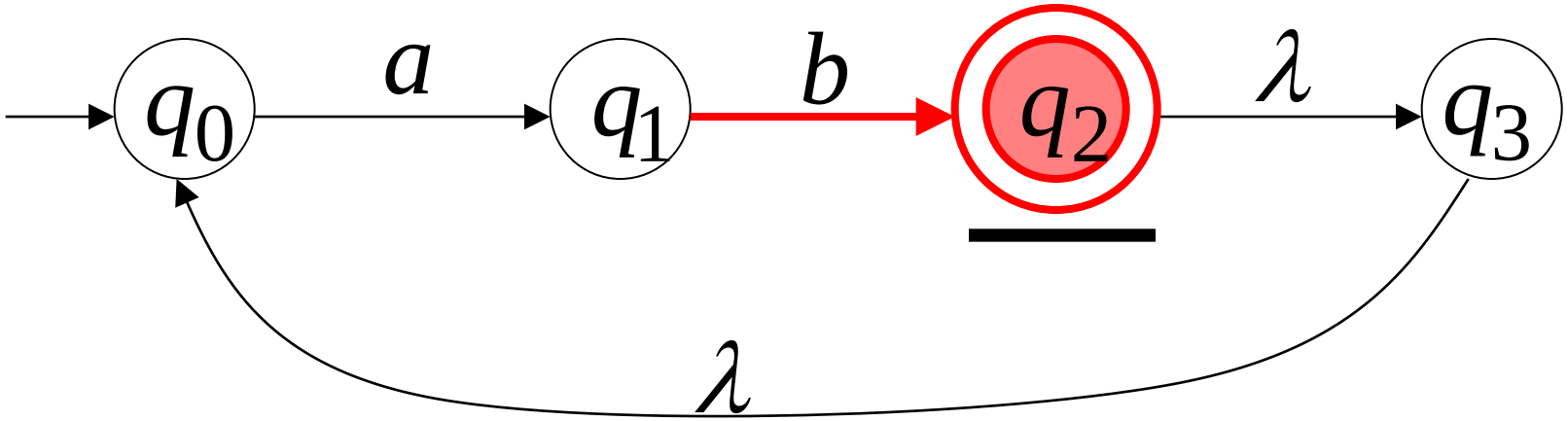


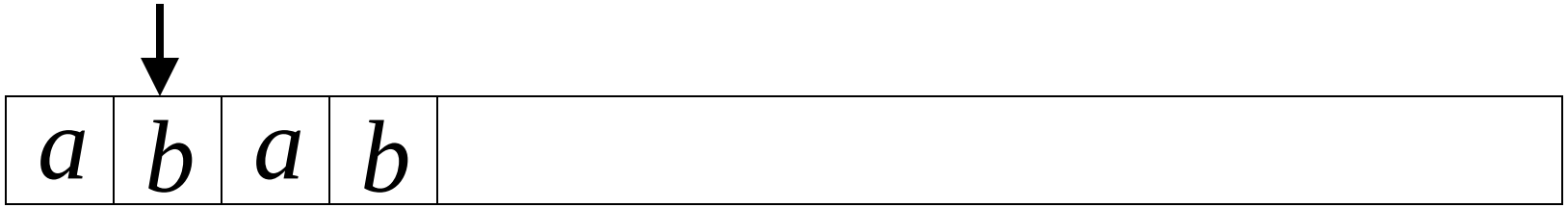


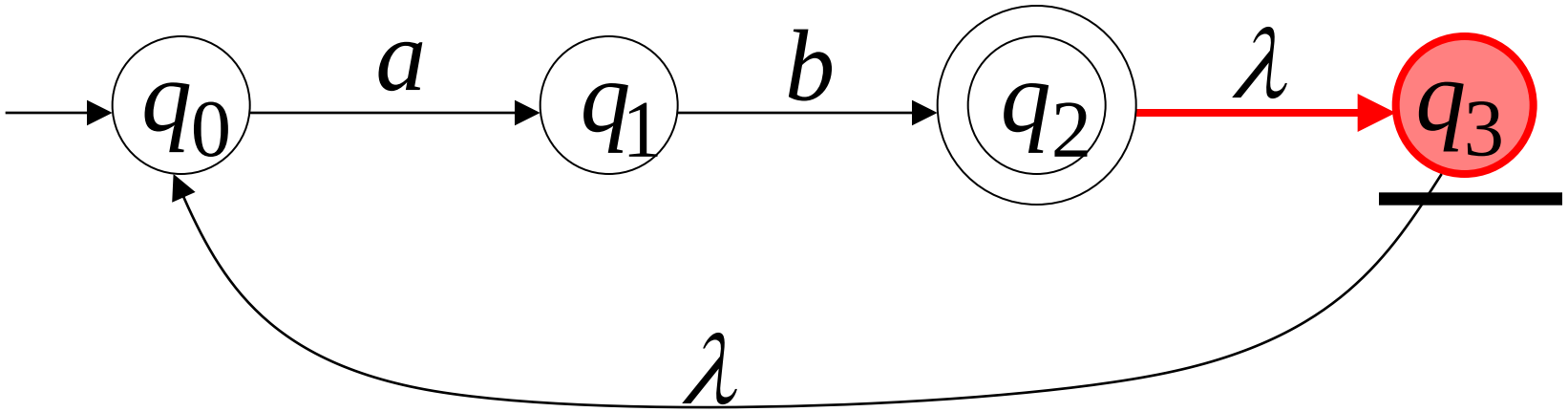

a
b
a
b

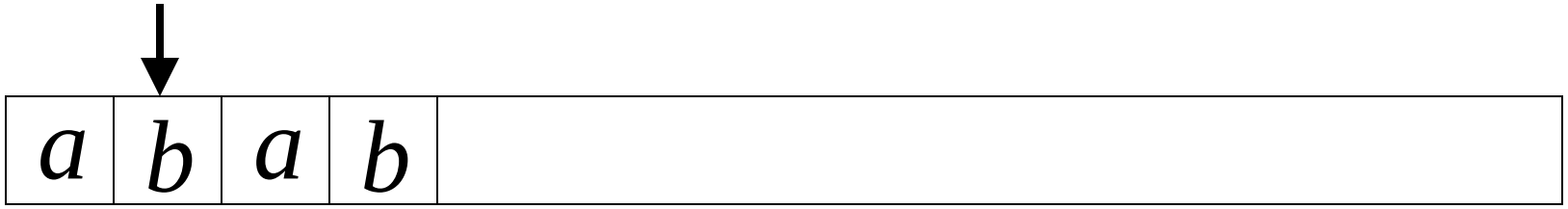

q_0
a
q_1
b
q_2
λ
q_3
λ

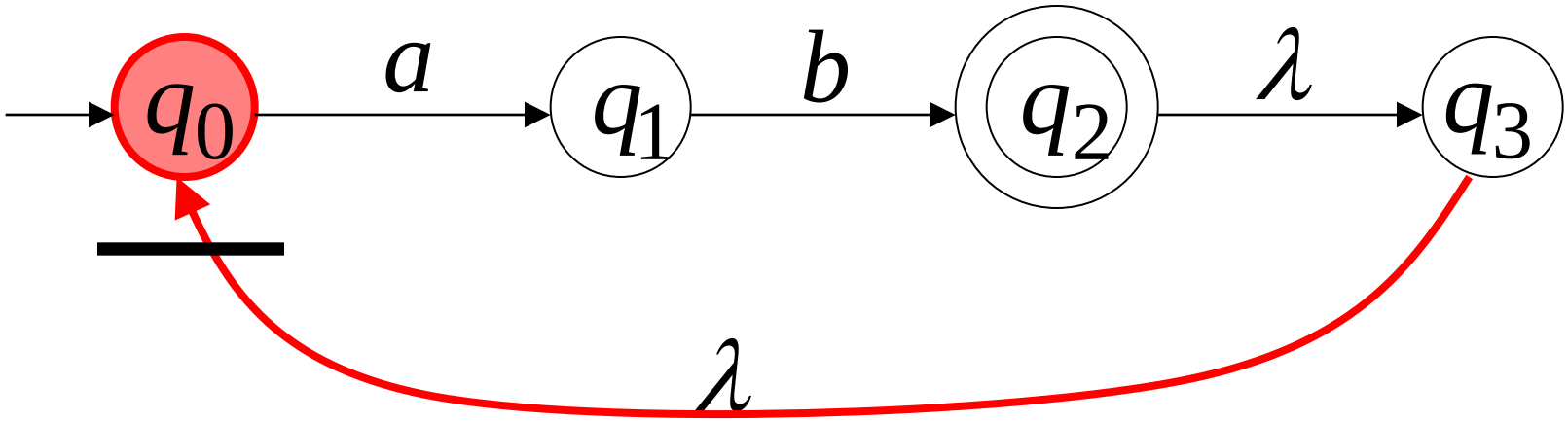

a
b
a
b

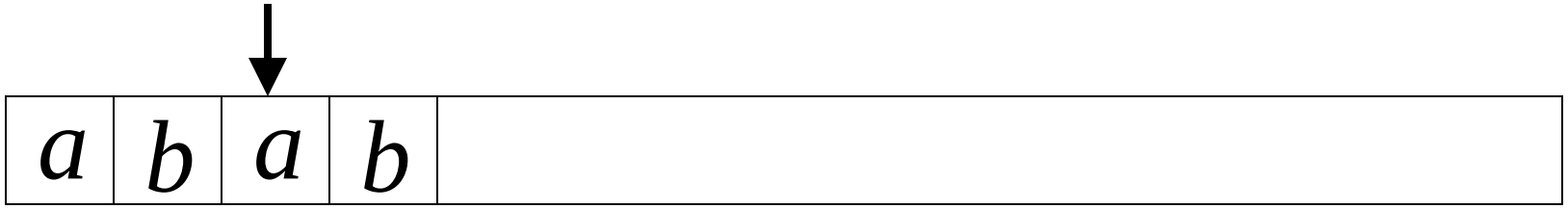

q_0
a
q_1
b
q_2
λ
q_3
λ

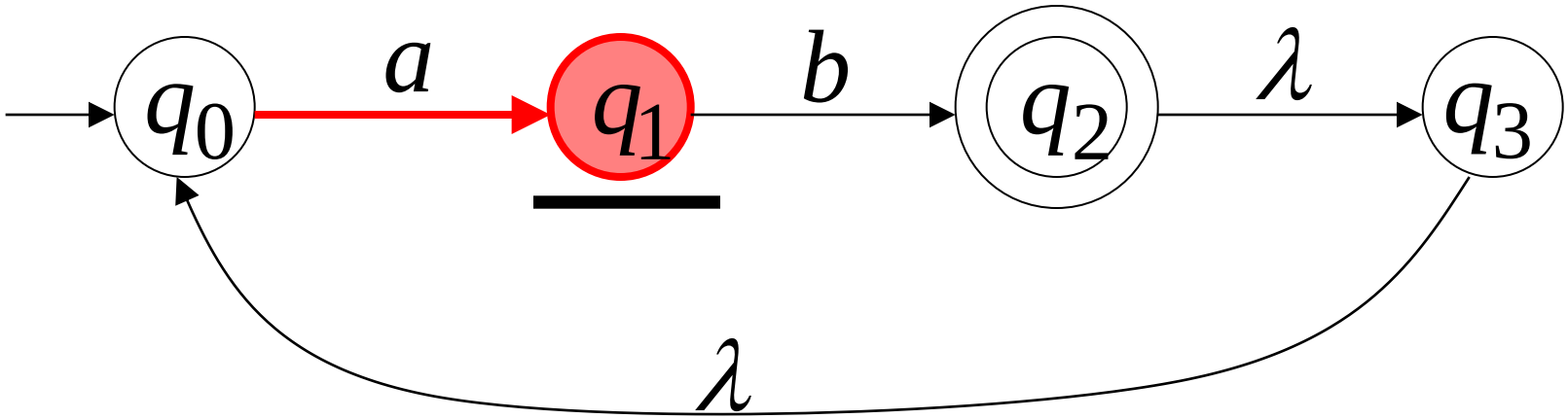

a
b
a
b

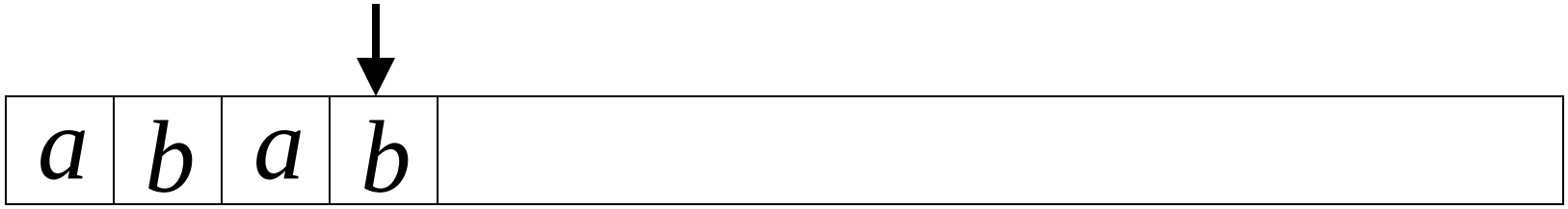

q_0
a
q_1
b
q_2
λ
q_3
λ

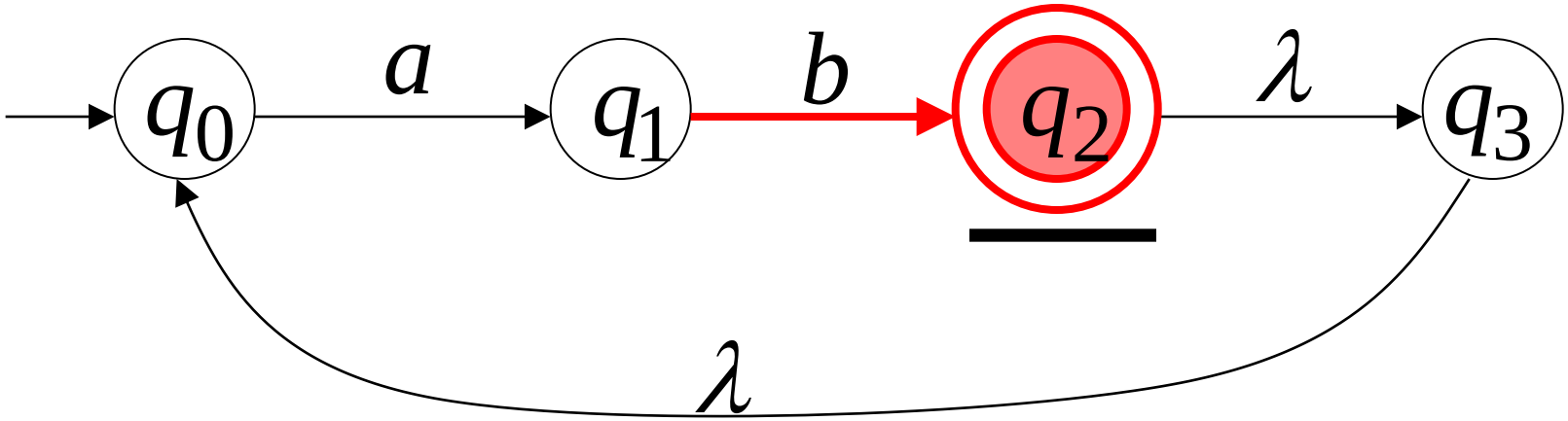

a
b
a
b

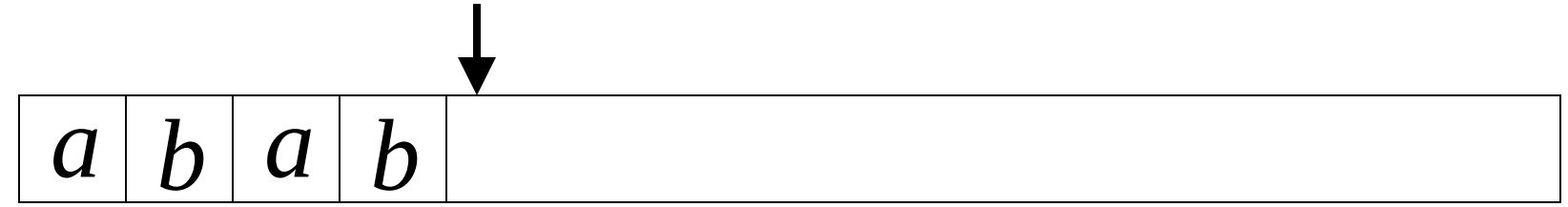

q_0
a
q_1
b
q_2
λ
q_3
λ

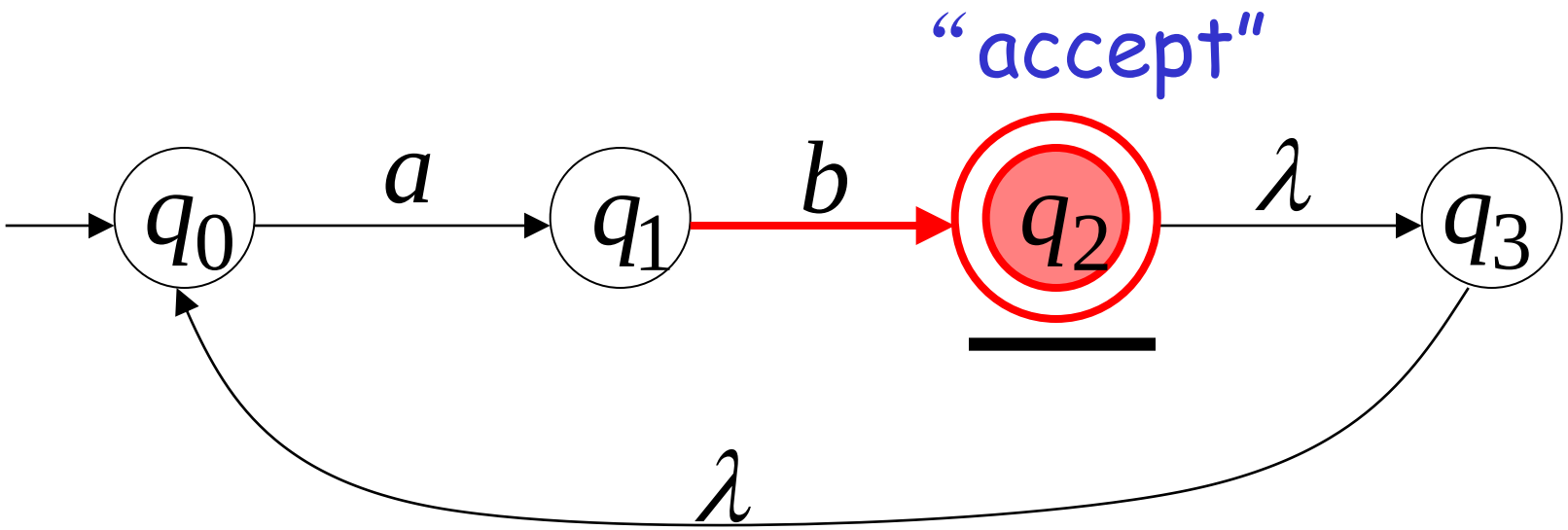

a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

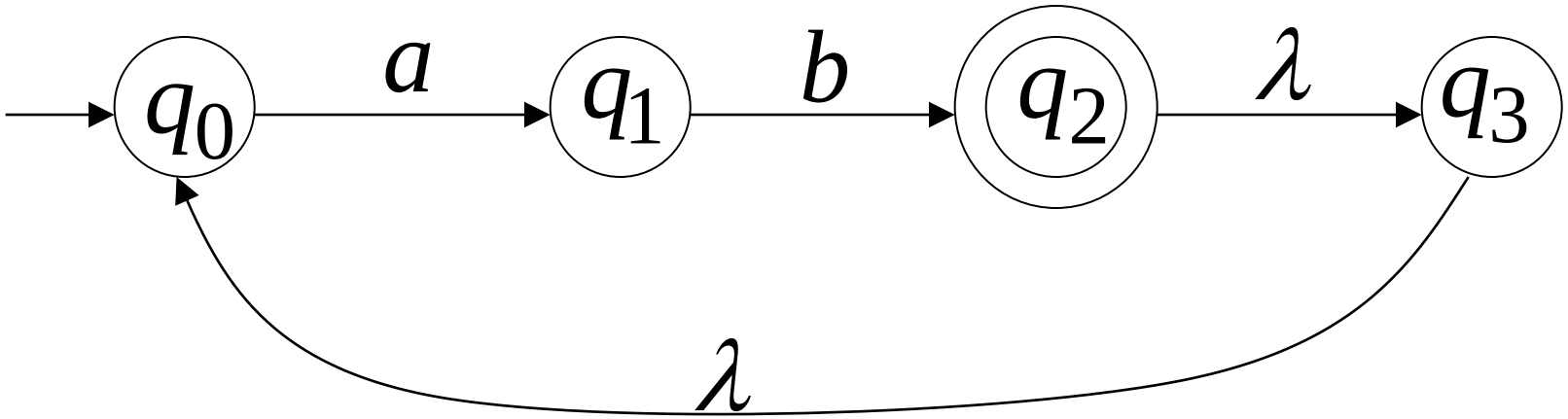

a
b
a
b


"accept"
q_0
q_1
q_2
q_3
a
b
λ
λ

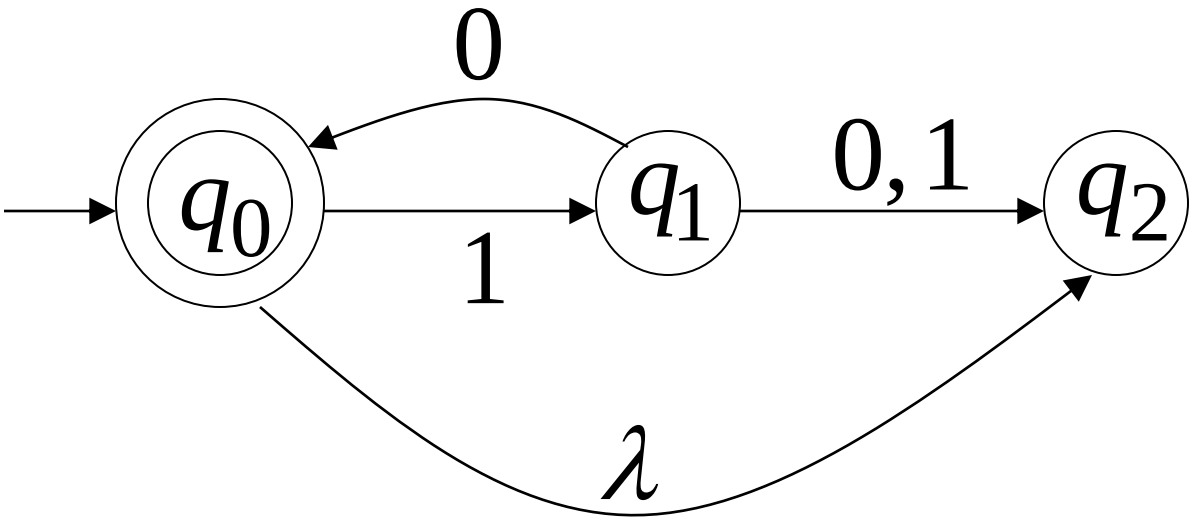
Language accepted

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

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



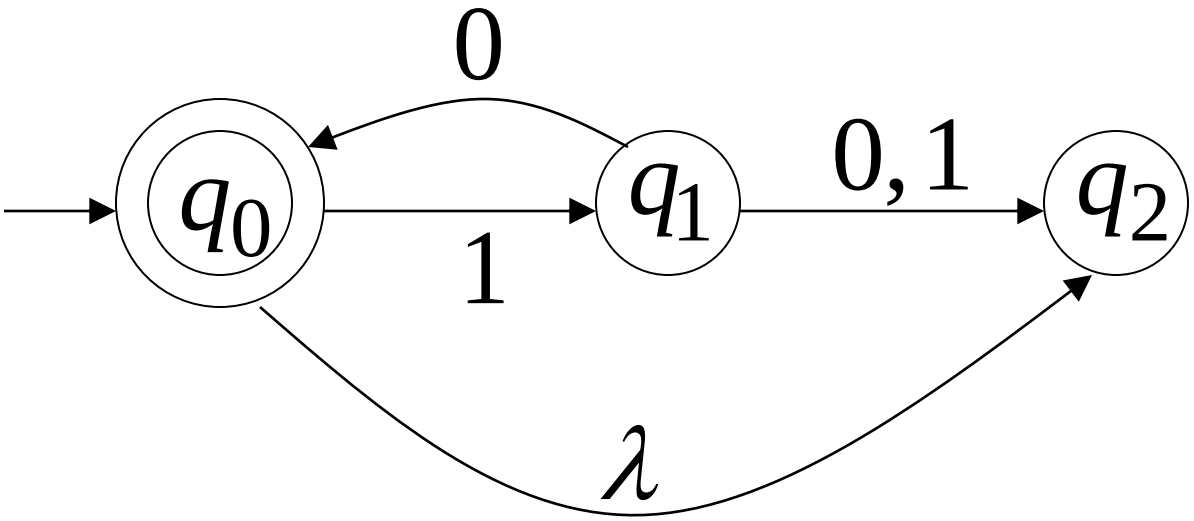
Another NFA Example



Language accepted

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

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

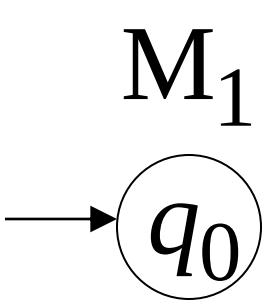


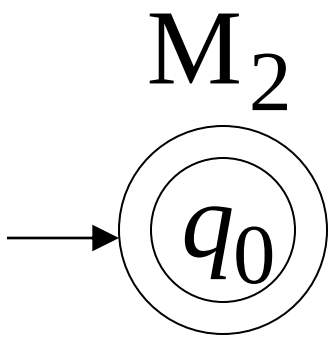
(redundant state)

Remarks:

- The λ symbol never appears on the input tape

- Simple automata:



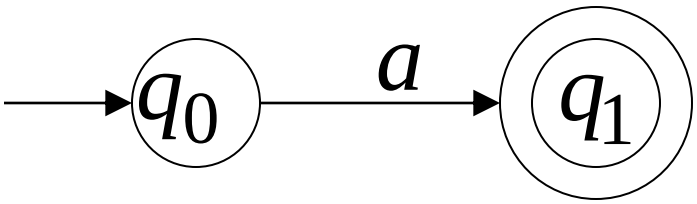


$L(M_1) = \{\}$

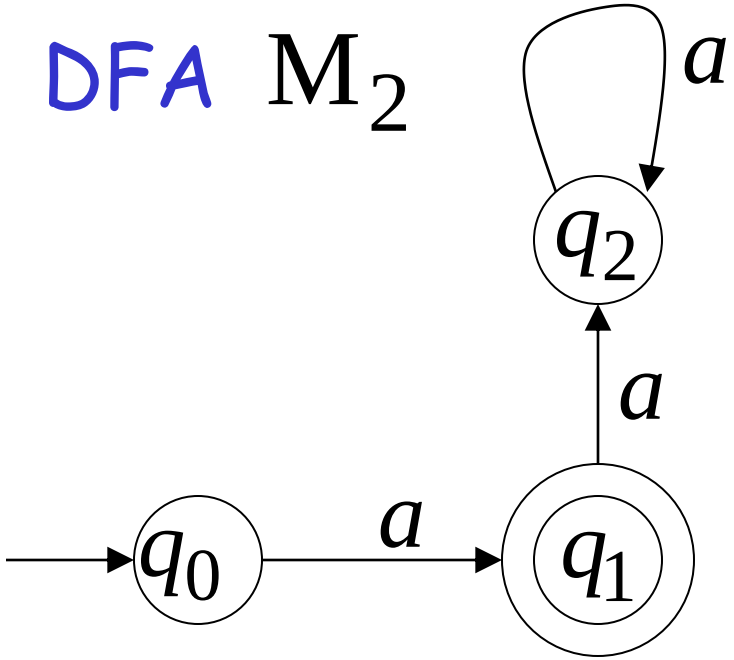
$L(M_2) = \{\lambda\}$

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

NFA M_1



DFA M_2



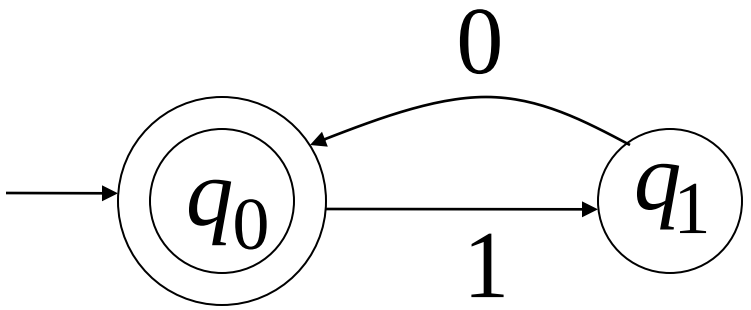
$$L(M_1) = \{a\}$$

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

Example

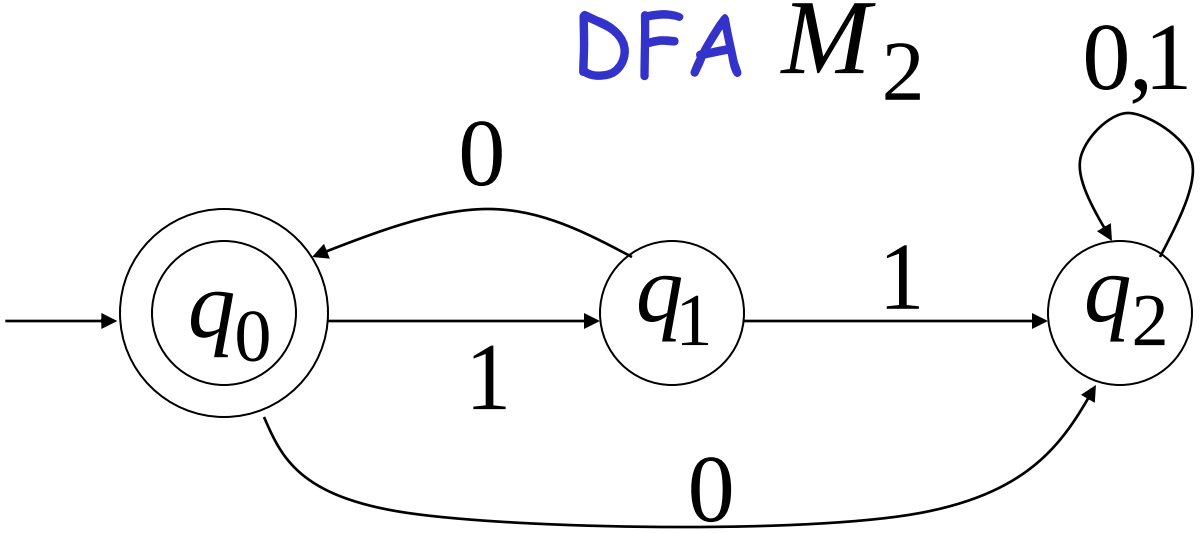
$L(M_1) = \{10\}^*$

NFA M_1



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

DFA M_2



Formal Definition of NFAs

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

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

Σ: Input aplhabet, i.e. $\{a, b\}$

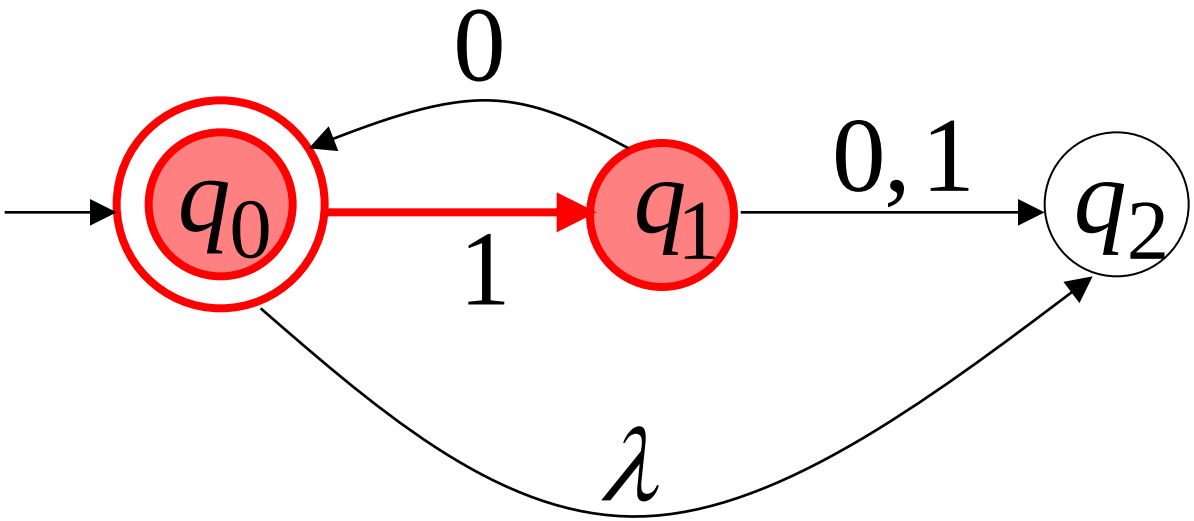
δ: Transition function

q_0: Initial state

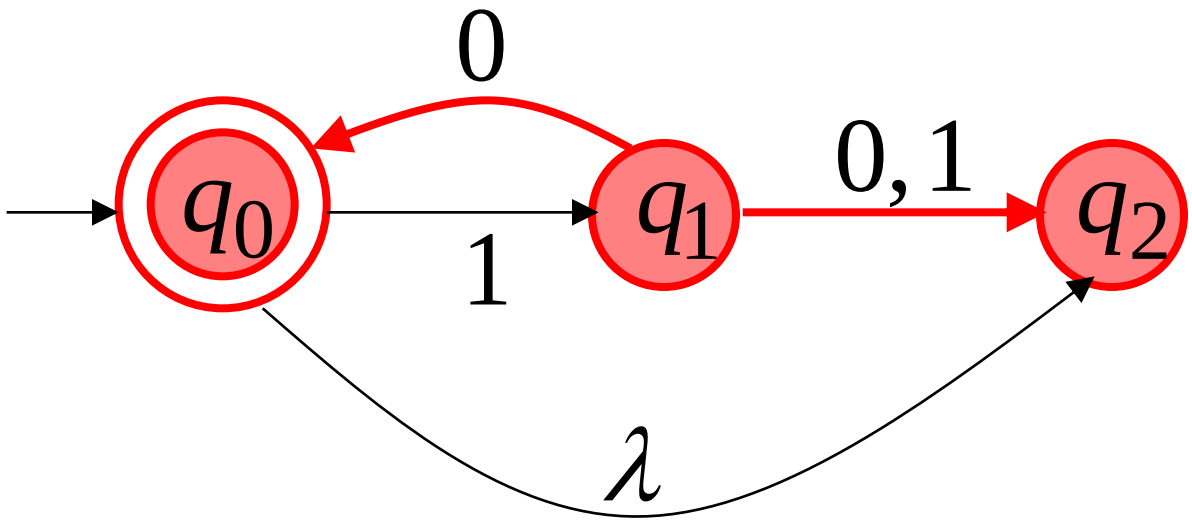
F: Final states

Transition Function δ

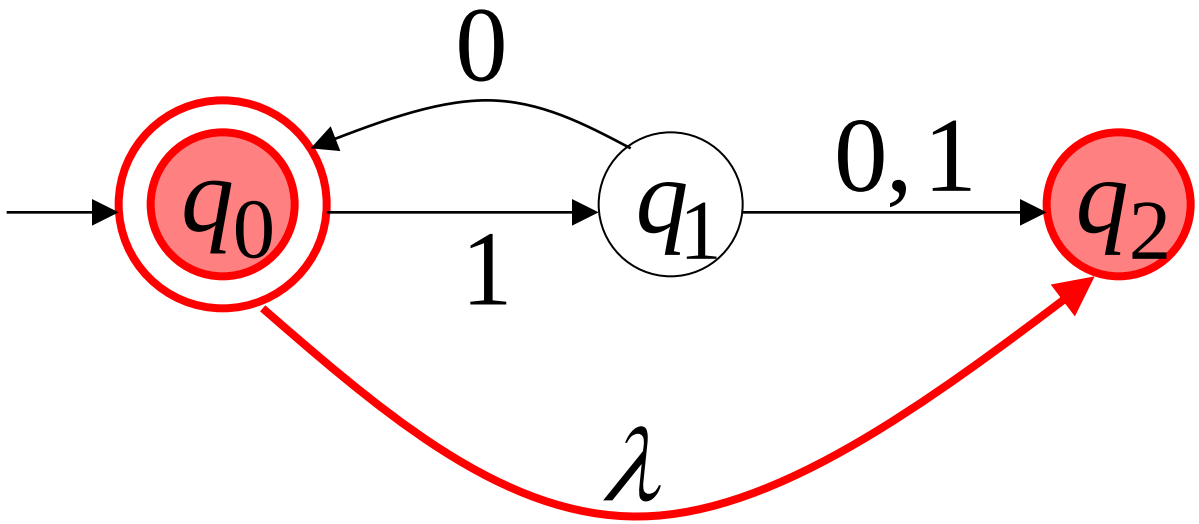
$$\delta(q_0, 1) = \{q_1\}$$



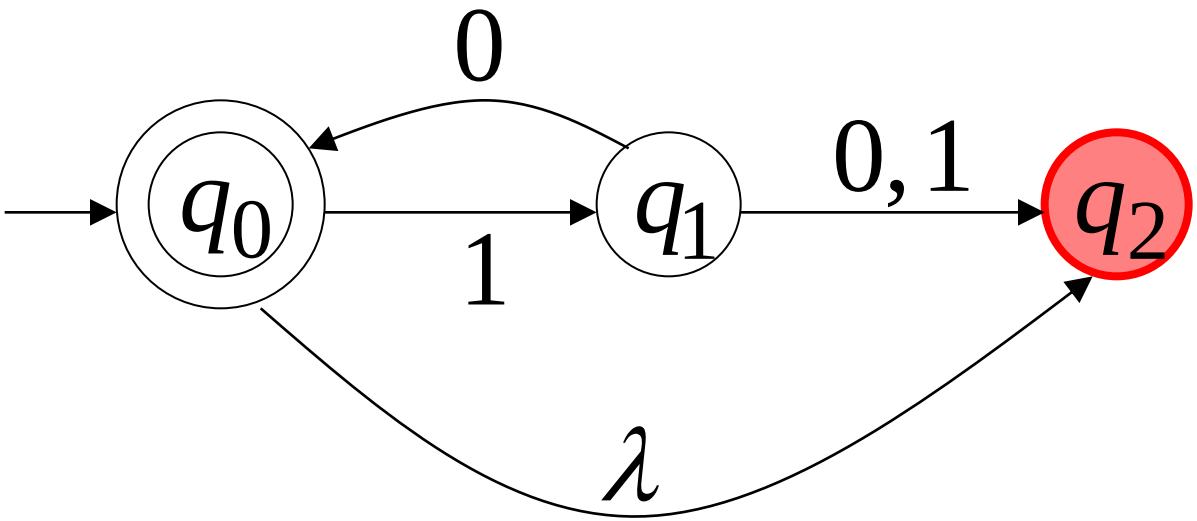
$$\delta(q_1,0)=\{q_0,q_2\}$$



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

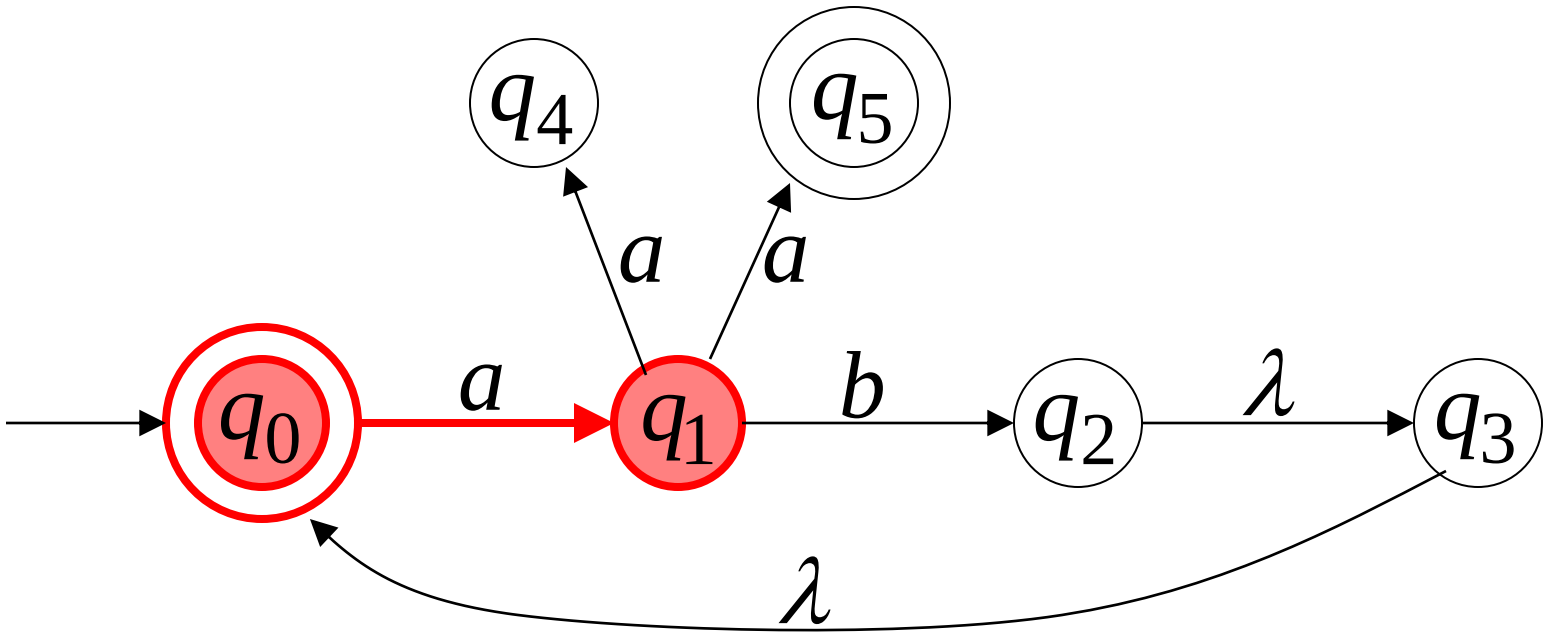


$$\delta(q_2,1)=\varnothing$$

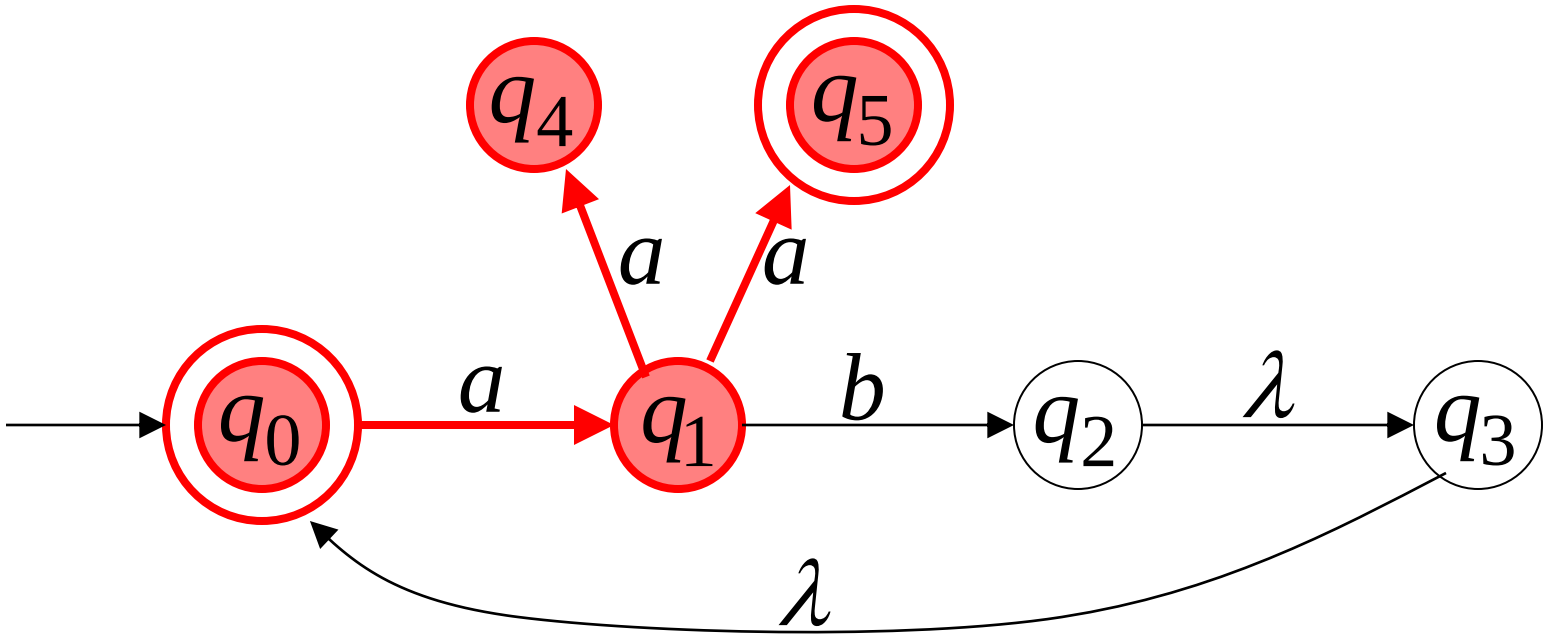


Extended Transition Function δ^*

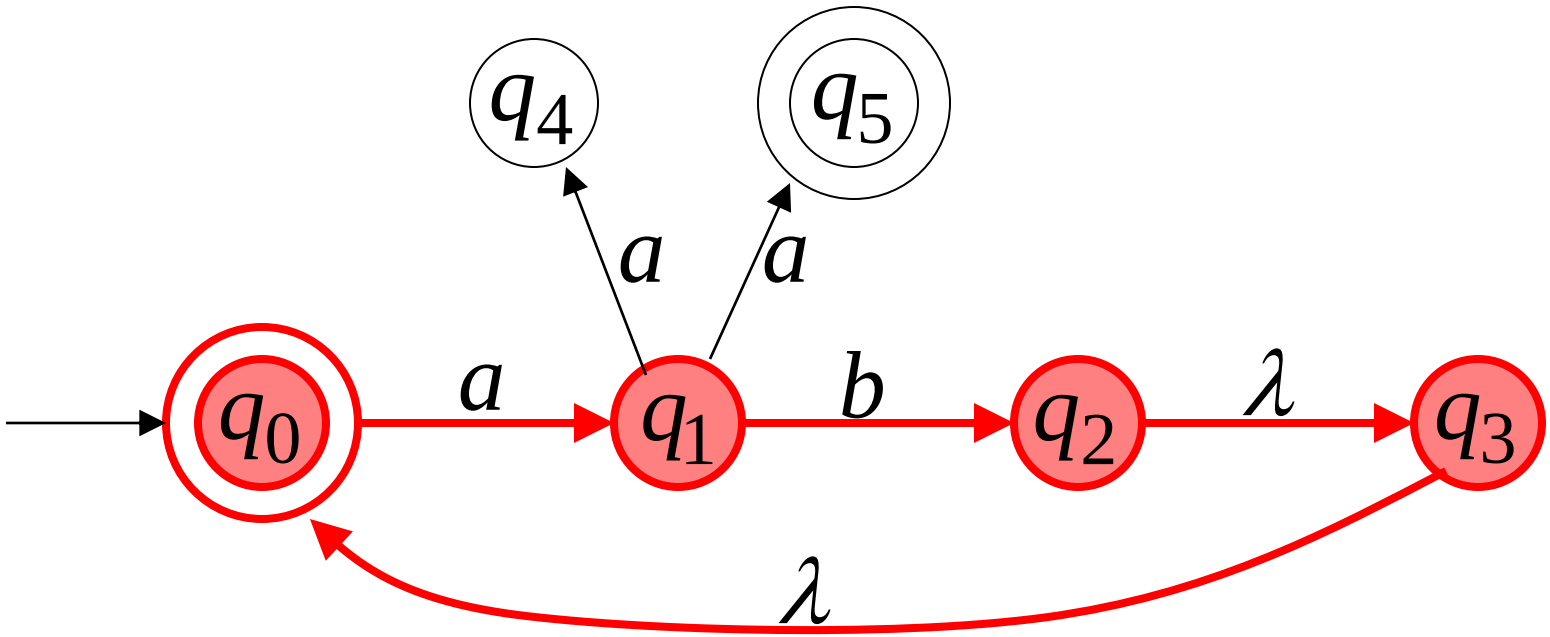
$$\delta^*(q_0, a) = \{q_1\}$$



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

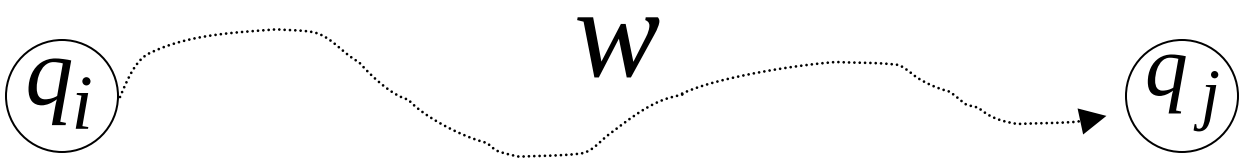


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

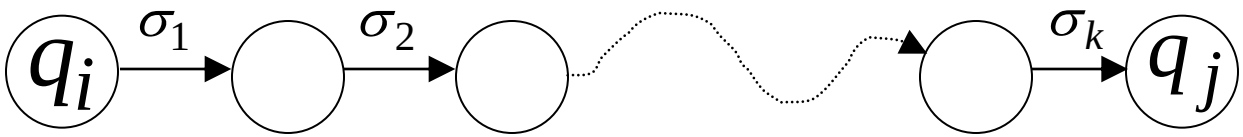


Formally

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

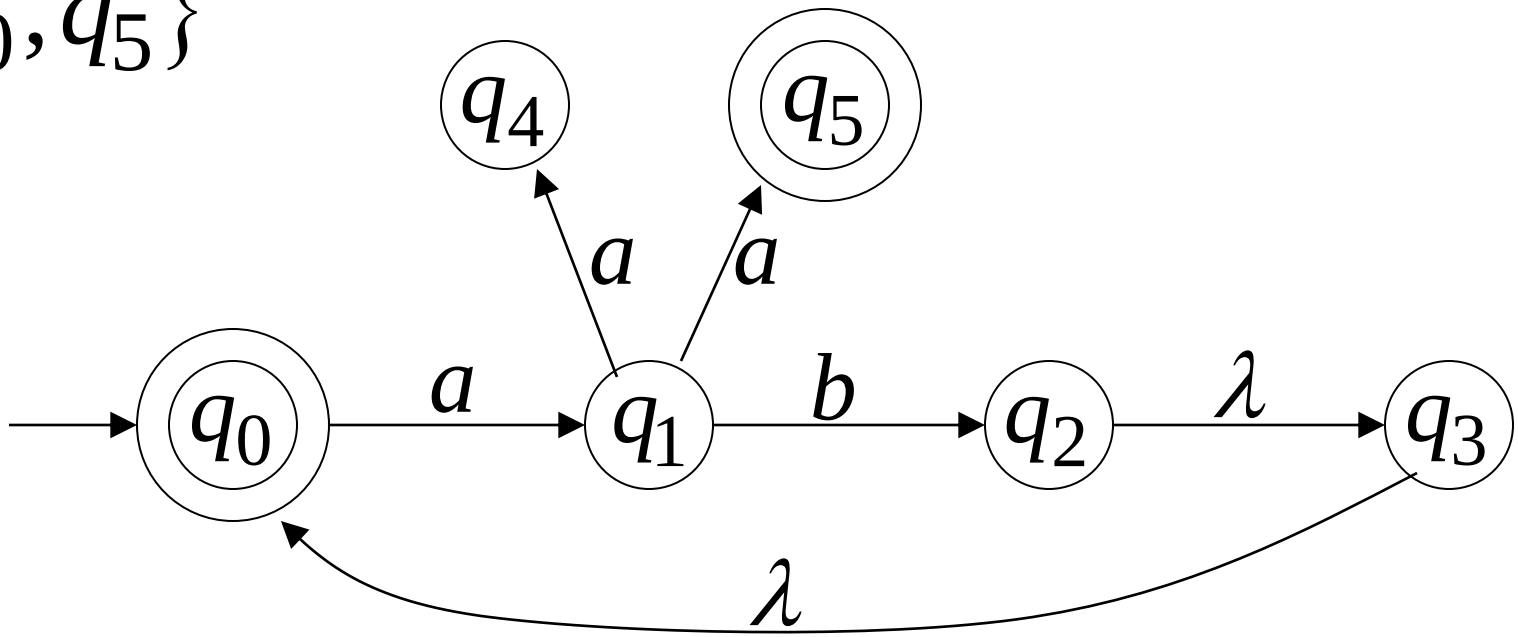


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



The Language of an NFA M

$F = \{q_0, q_5\}$

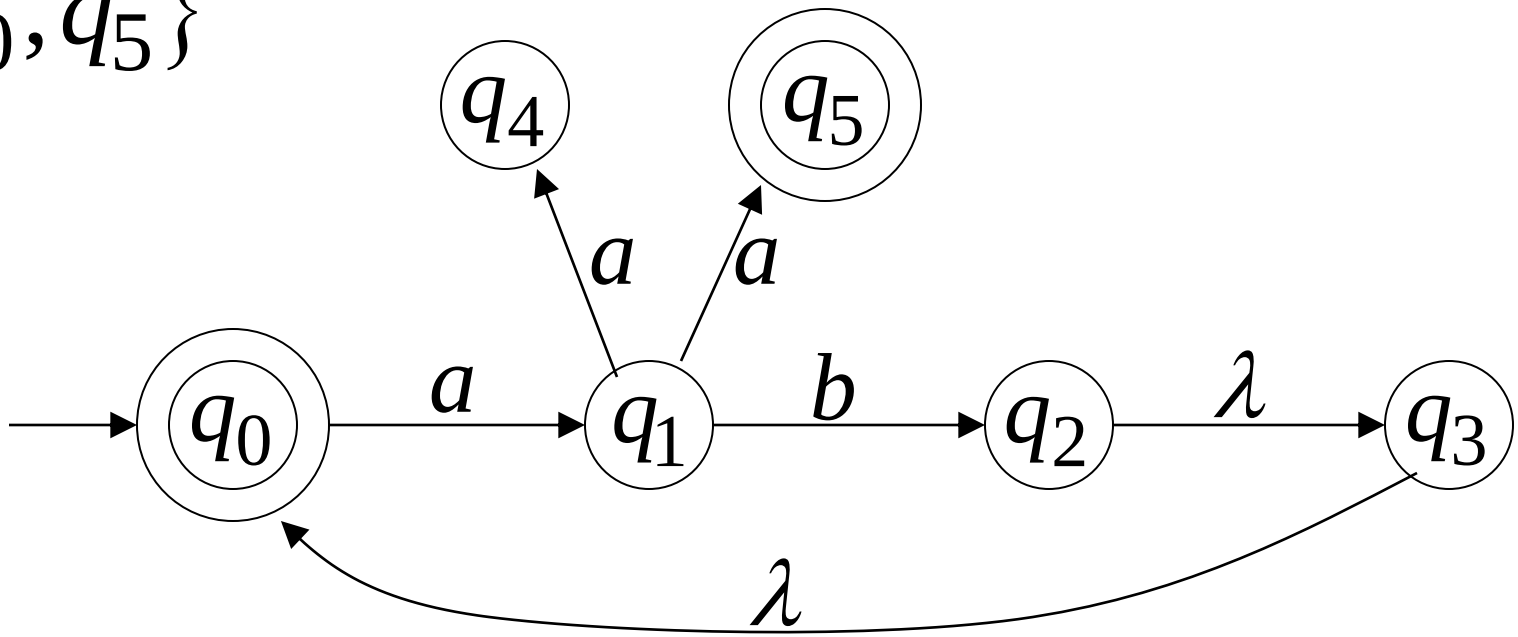


$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\}$

$\searrow \in F$

$aa \in L(M)$

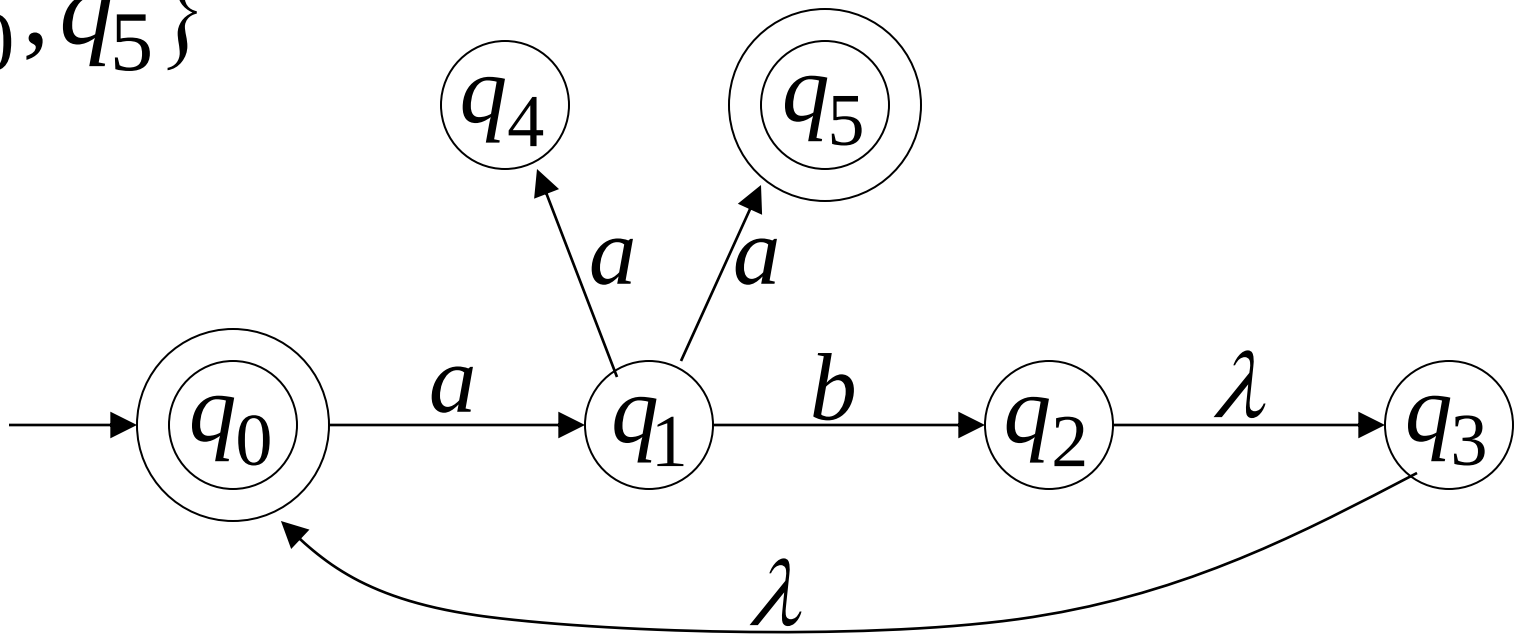
$F = \{q_0, q_5\}$



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

$\searrow \in F$

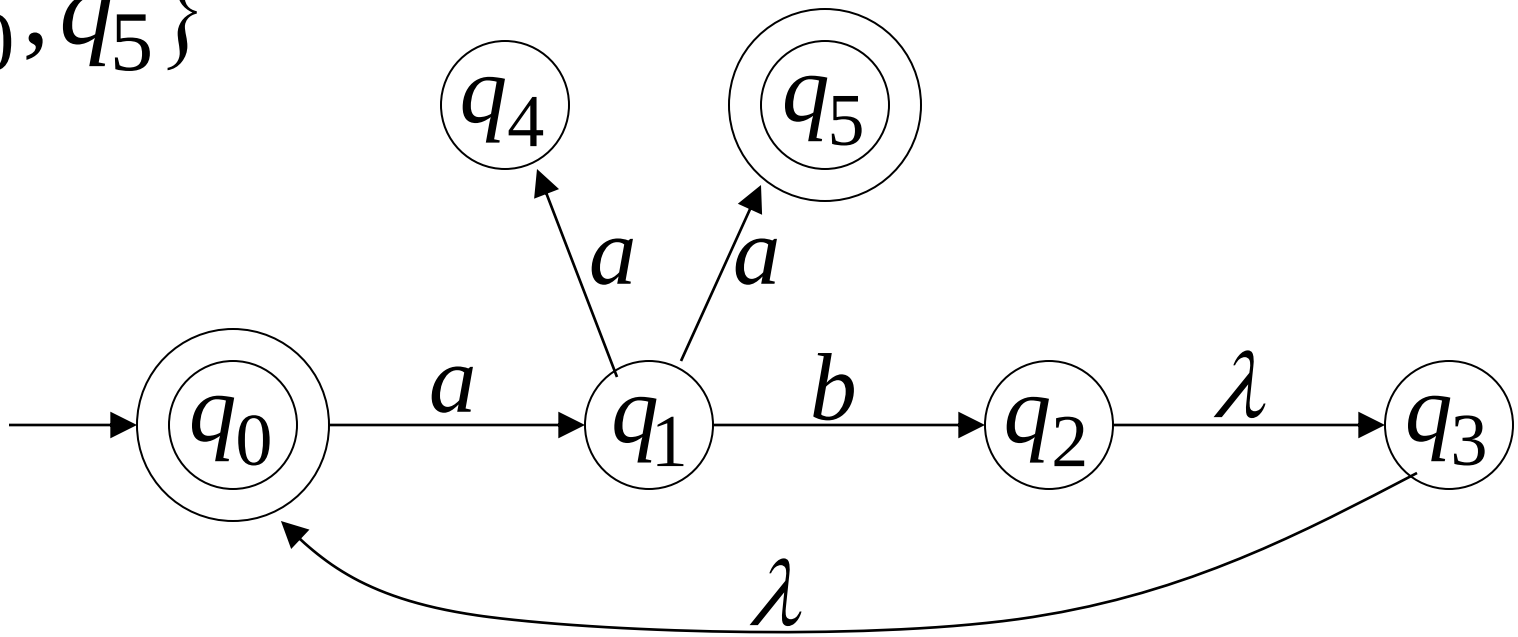
$F = \{q_0, q_5\}$



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

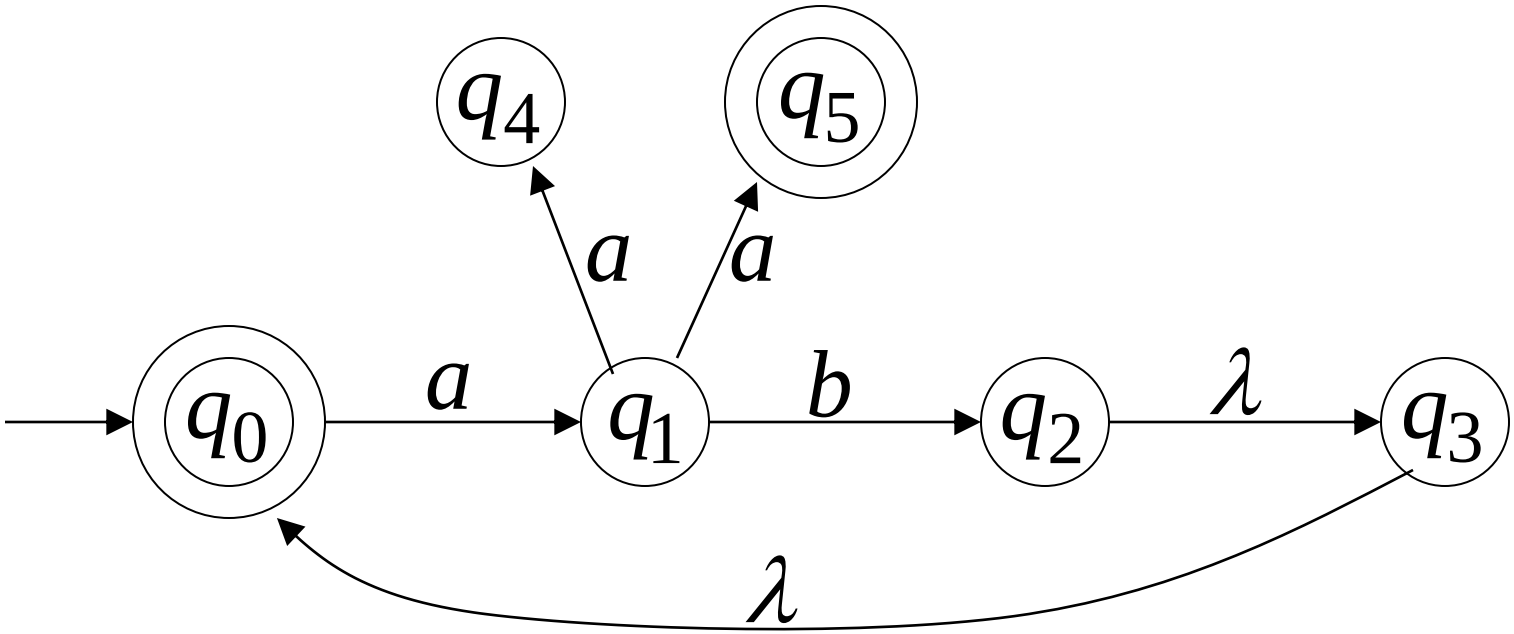
$\searrow \in F$

$F = \{q_0, q_5\}$



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

$\searrow \notin F$



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

Formally

The language accepted by NFA M is:

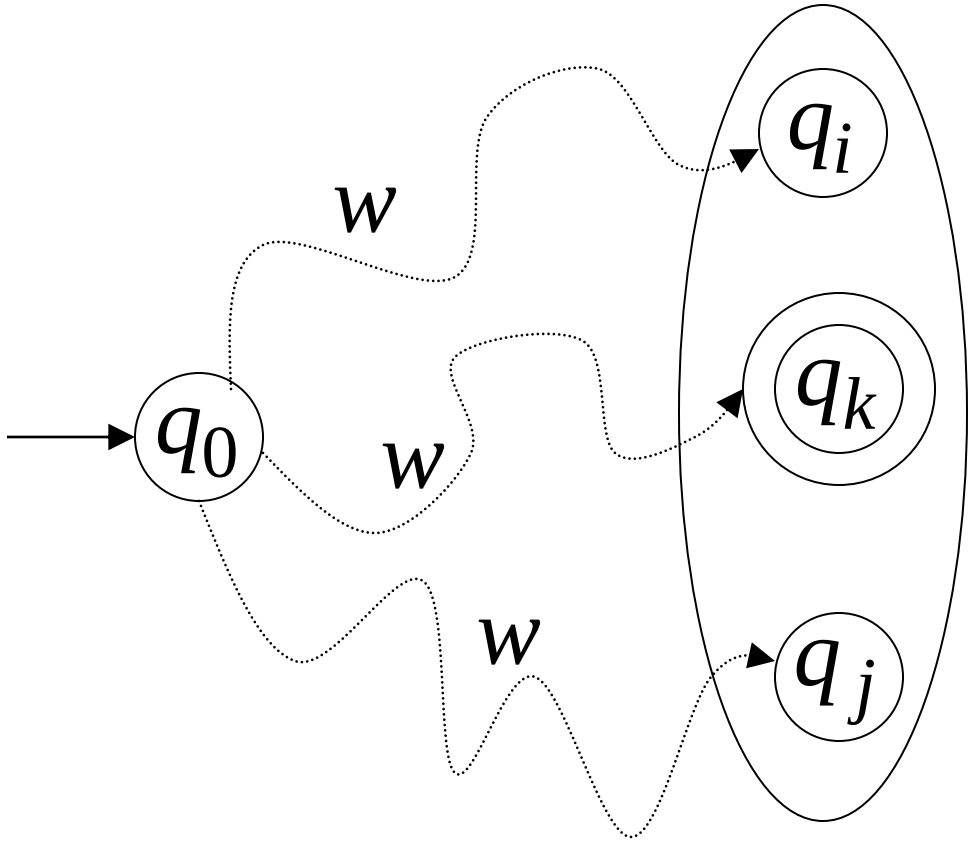
$$L(M) = \{w_1, w_2, w_3, \ldots\}$$

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

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

$w \in L(M)$

$\delta^*(q_0, w)$



$q_k \in F$

NFAs accept the Regular Languages

Equivalence of Machines

Definition for Automata:

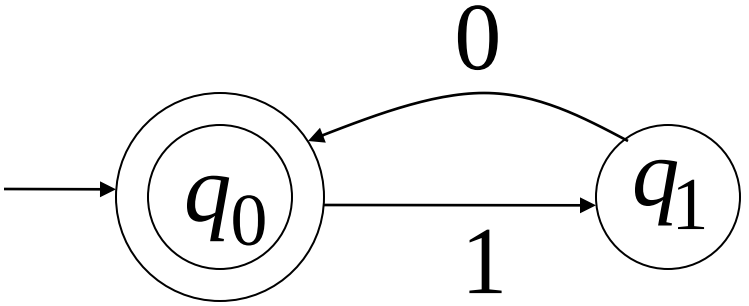
Machine M_1 is equivalent to machine M_2

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

Example of equivalent machines

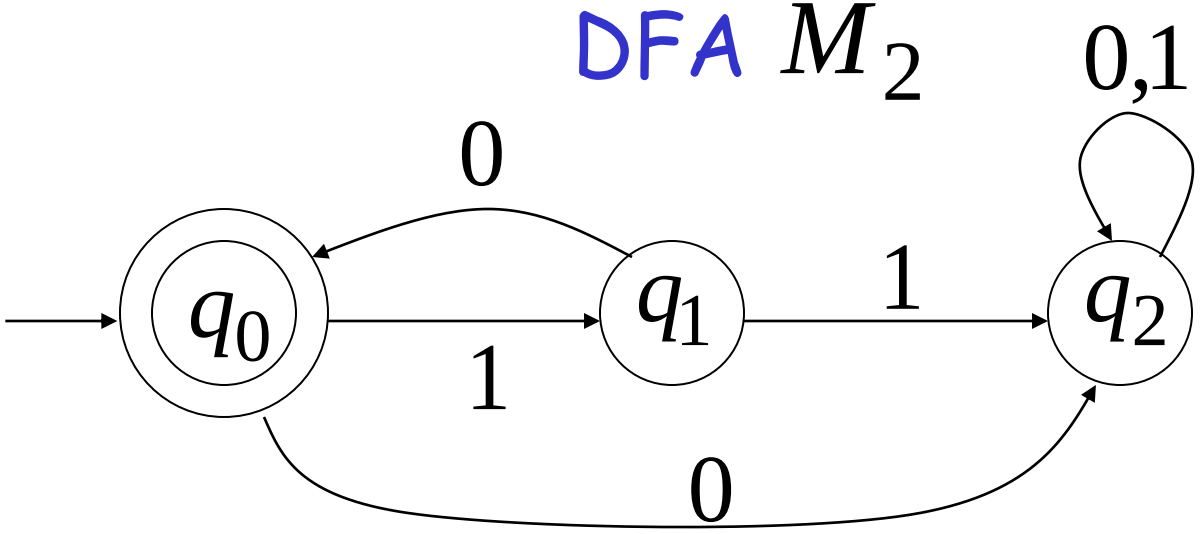
$L(M_1) = \{10\}^*$

NFA M_1



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

DFA M_2



We will prove:

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

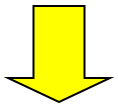
Languages
accepted
by DFAs

NFAs and DFAs have the same computation power

Step 1

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

Proof: Every DFA is trivially an NFA

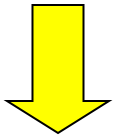


Any language L accepted by a DFA is also accepted by an NFA

Step 2

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

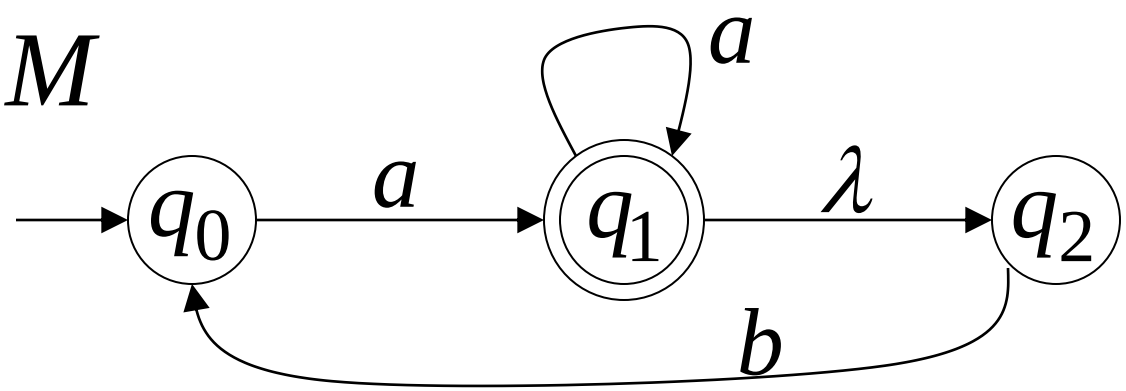
Proof: Any NFA can be converted to an equivalent DFA



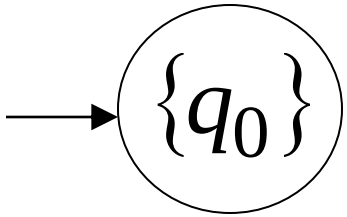
Any language L accepted by an NFA is also accepted by a DFA

Convert NFA to DFA

NFA M

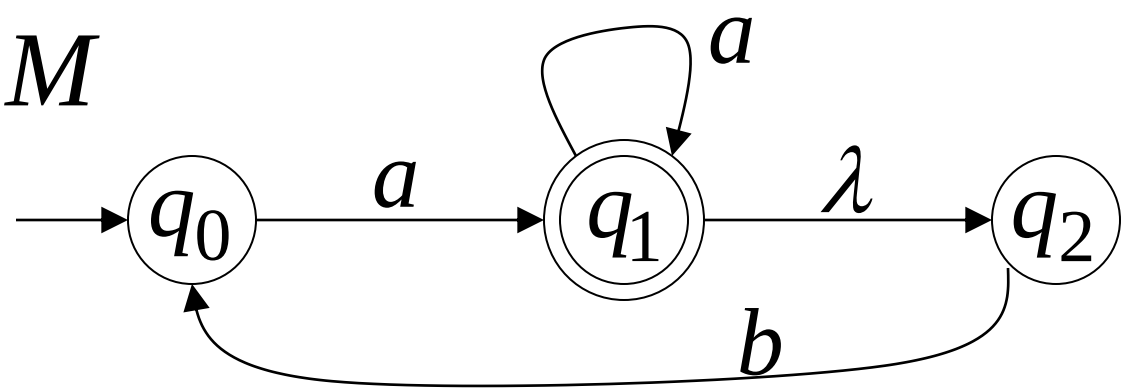


DFA M'

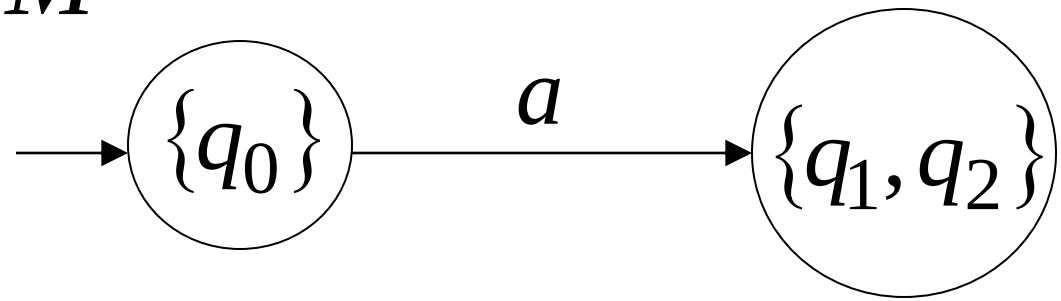


Convert NFA to DFA

NFA M

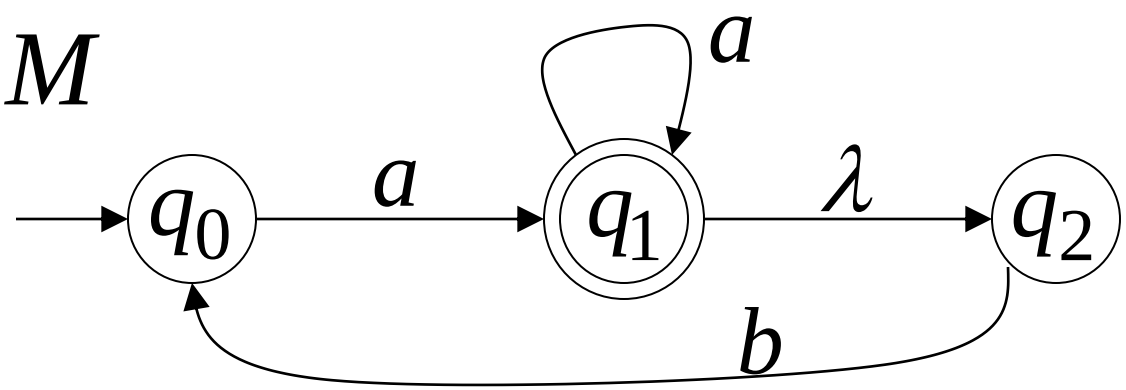


DFA M'

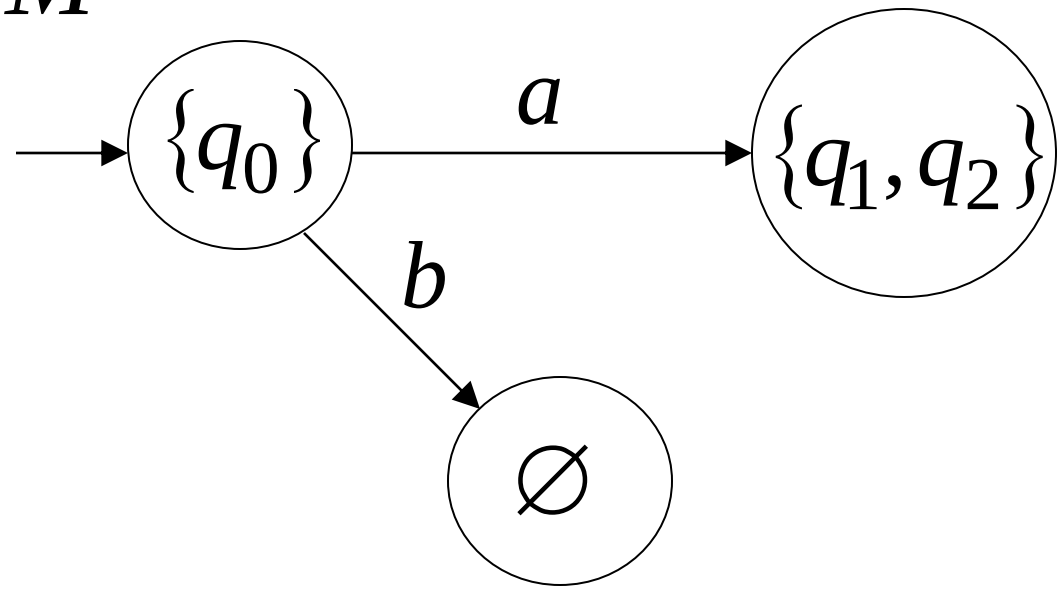


Convert NFA to DFA

NFA M

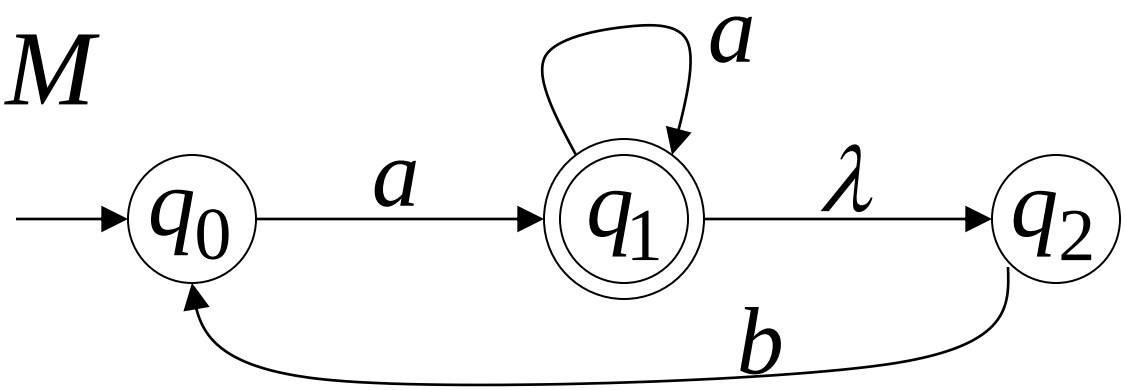


DFA M'

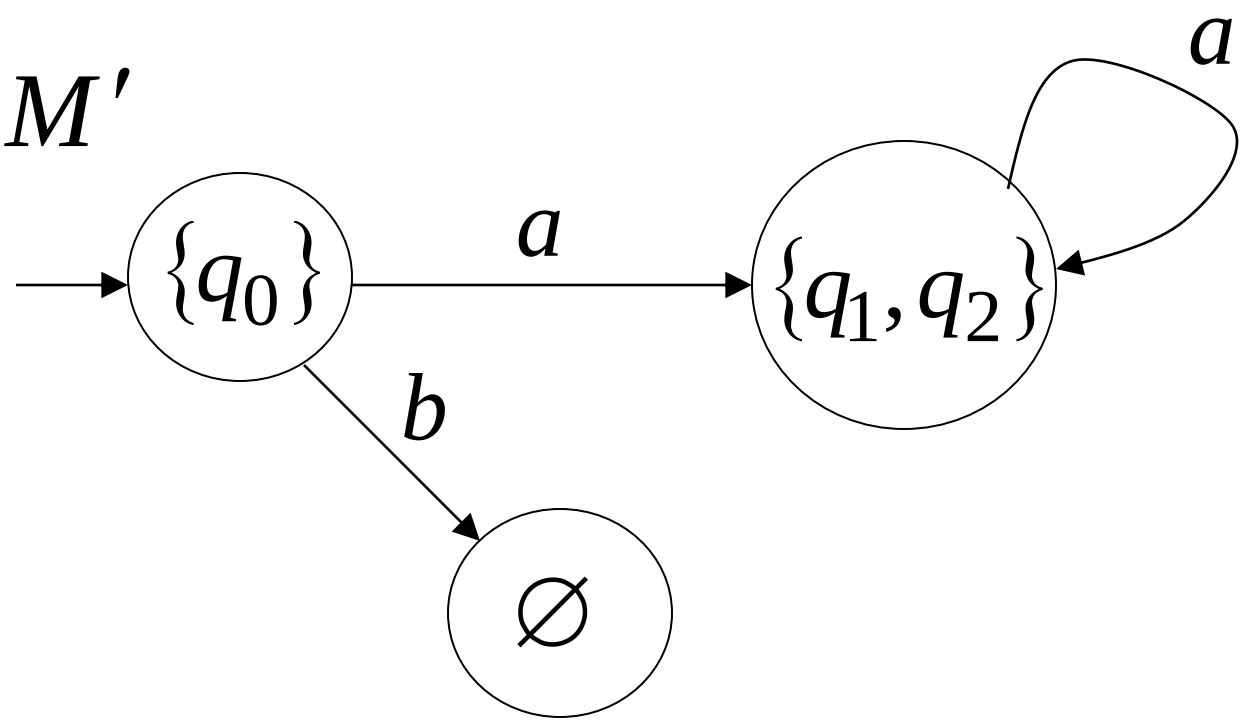


Convert NFA to DFA

NFA M

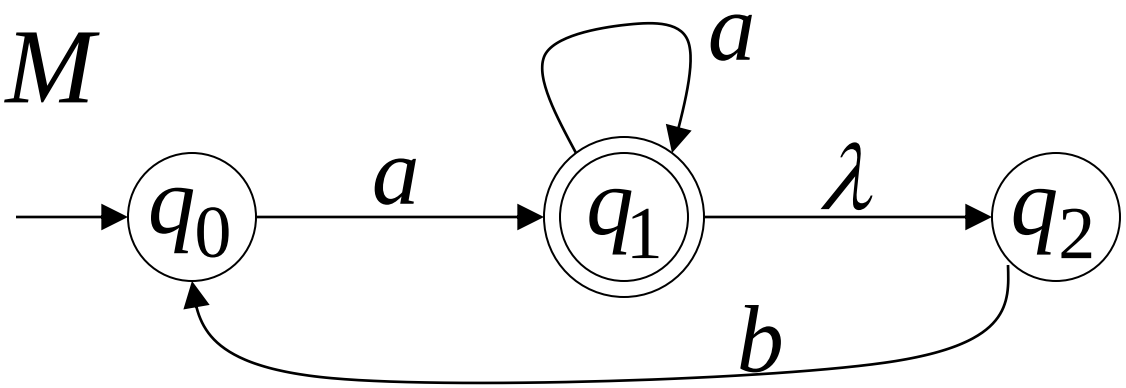


DFA M'

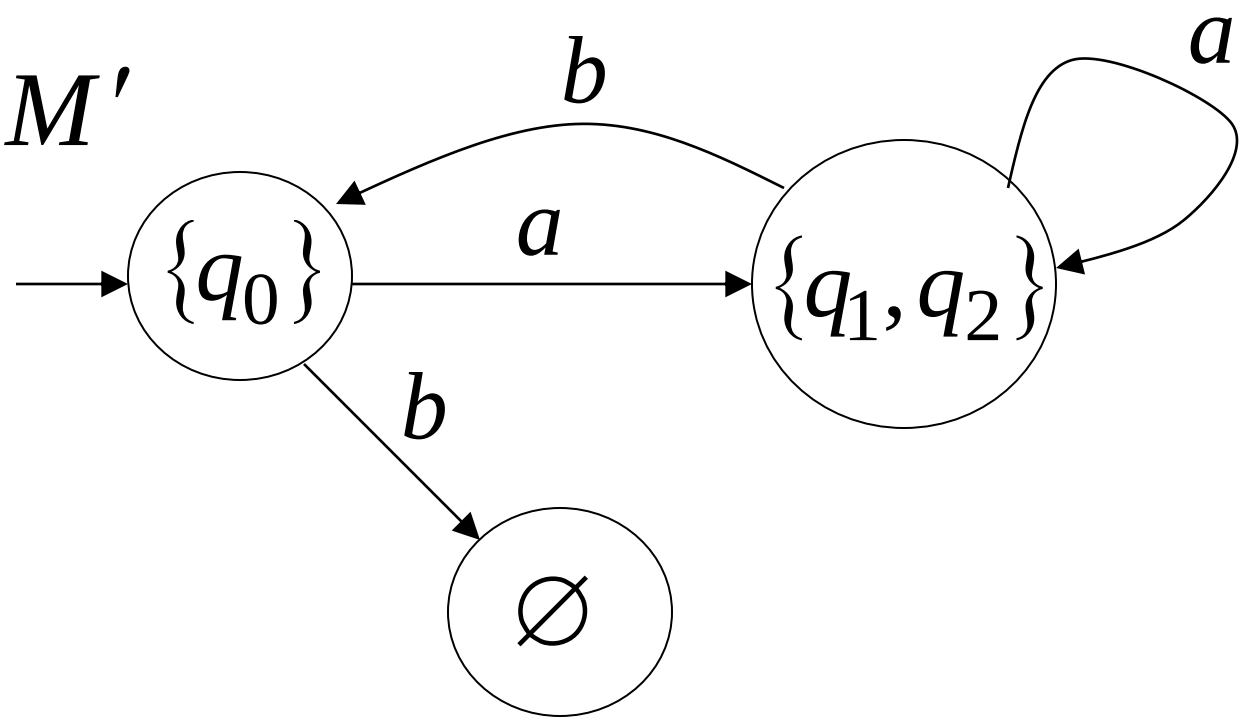


Convert NFA to DFA

NFA M

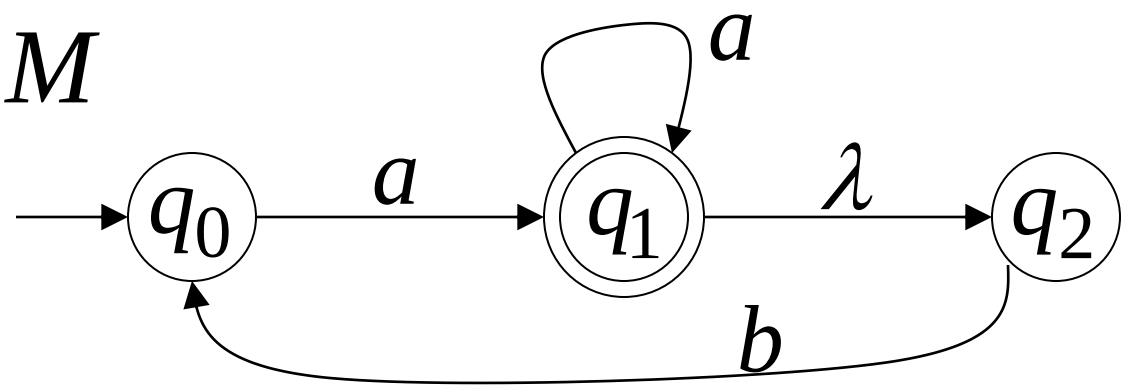


DFA M'

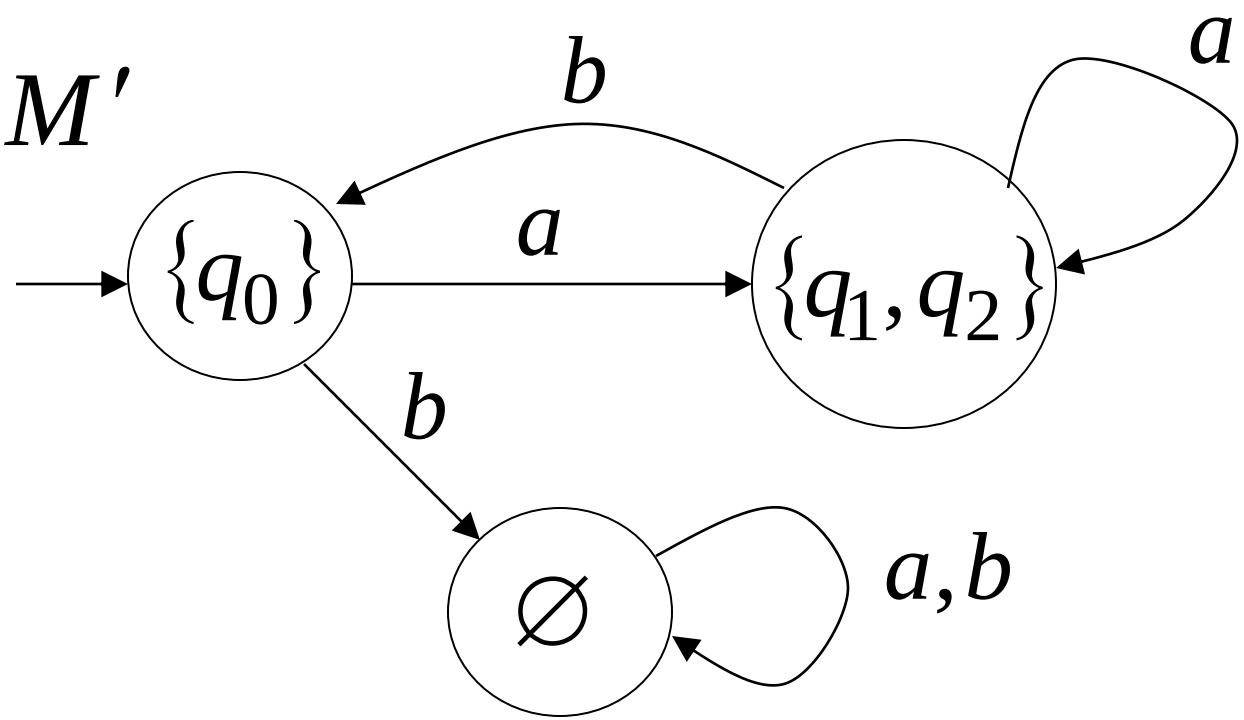


Convert NFA to DFA

NFA M

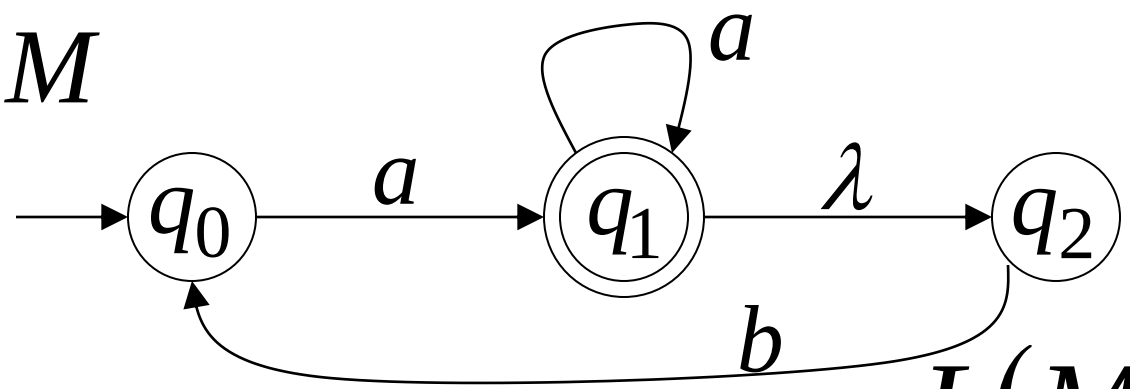


DFA M'



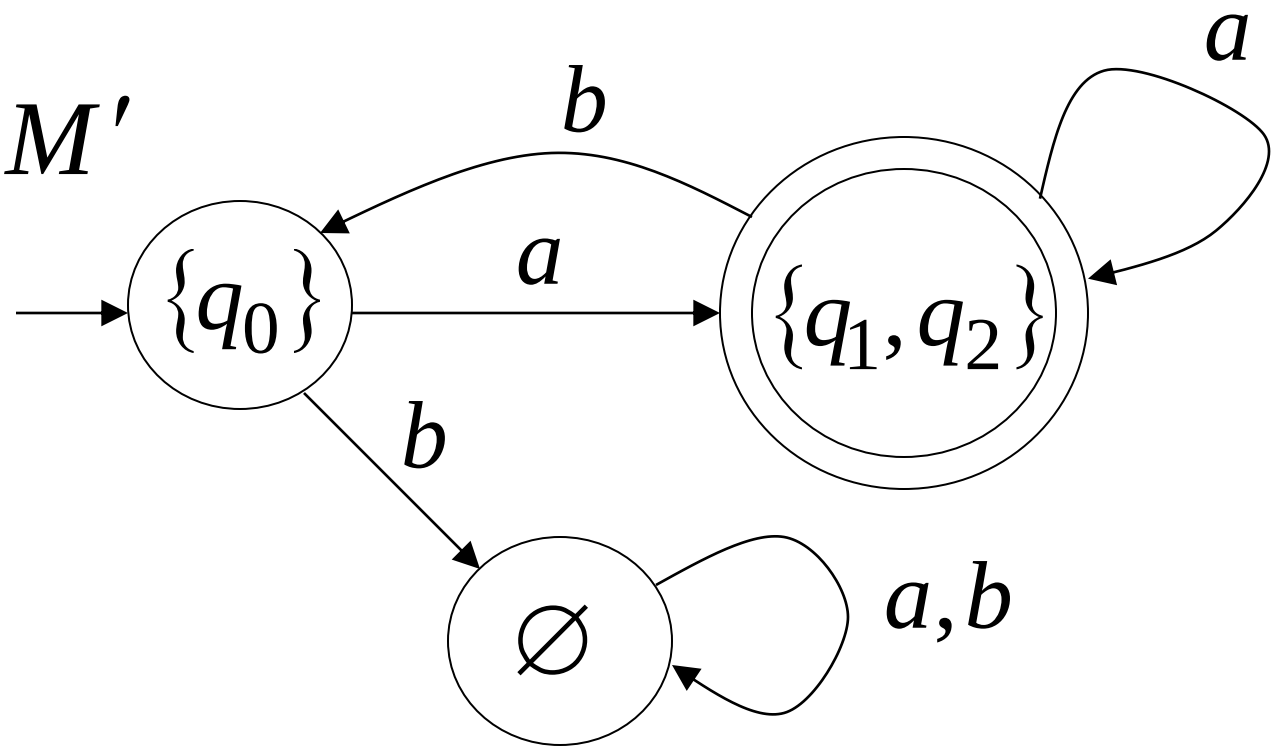
Convert NFA to DFA

NFA M



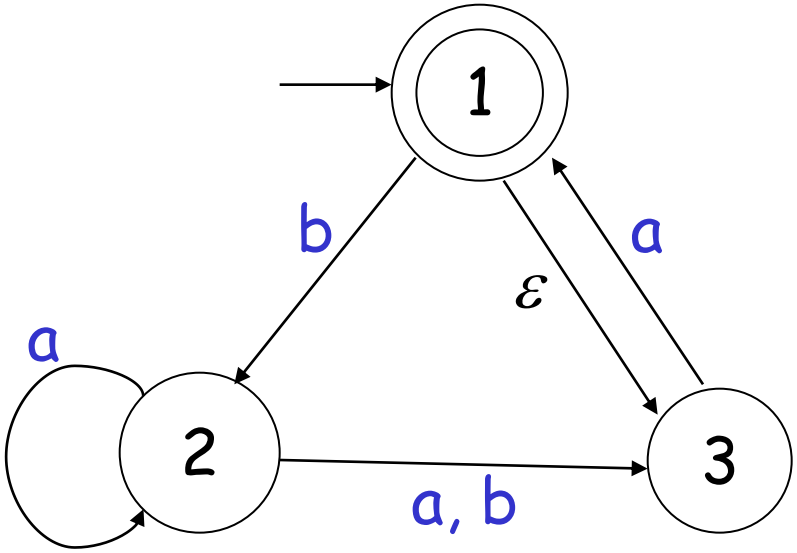
$$L(M) = L(M')$$

DFA M'



More example: Converting NFA to DFA

- NFA N_4 = (Q,{a,b},δ,1,{1}), the set of states Q is {1,2,3} as shown in the following figure.



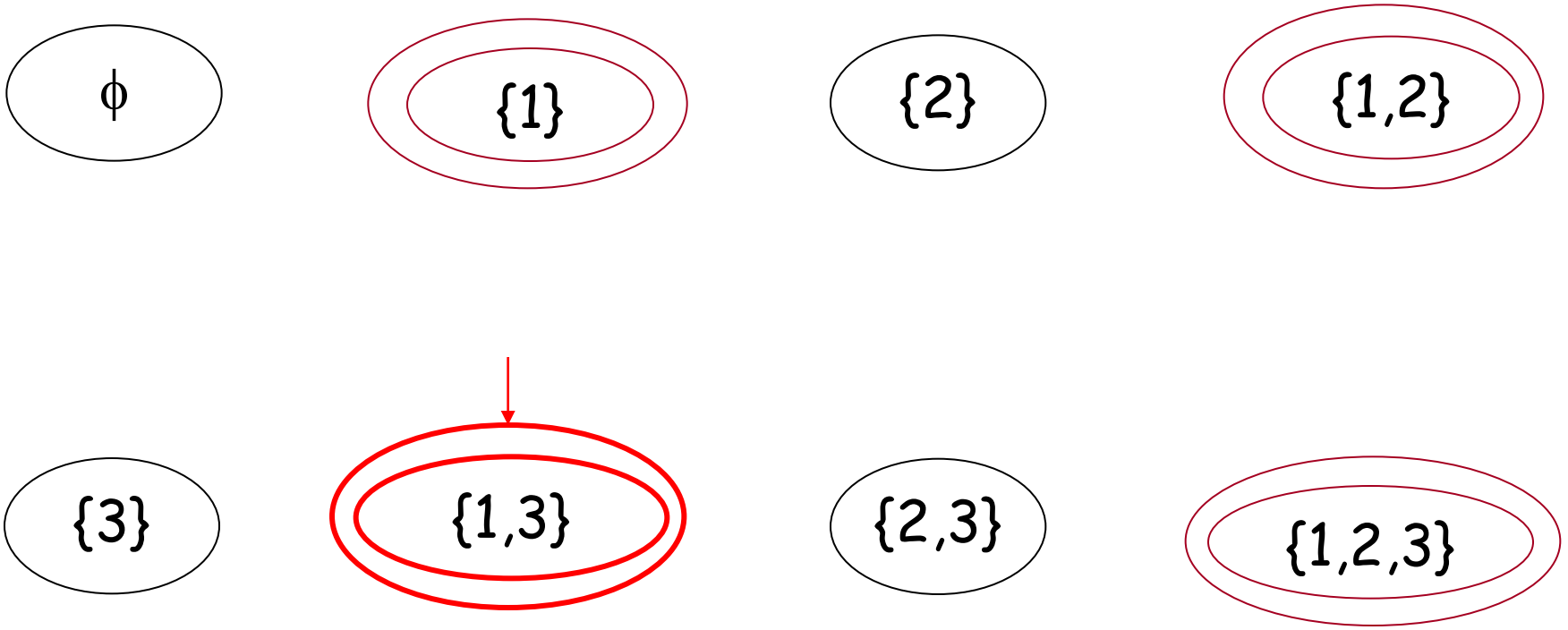
The NFA N_4

Step1: Determine DFA's states

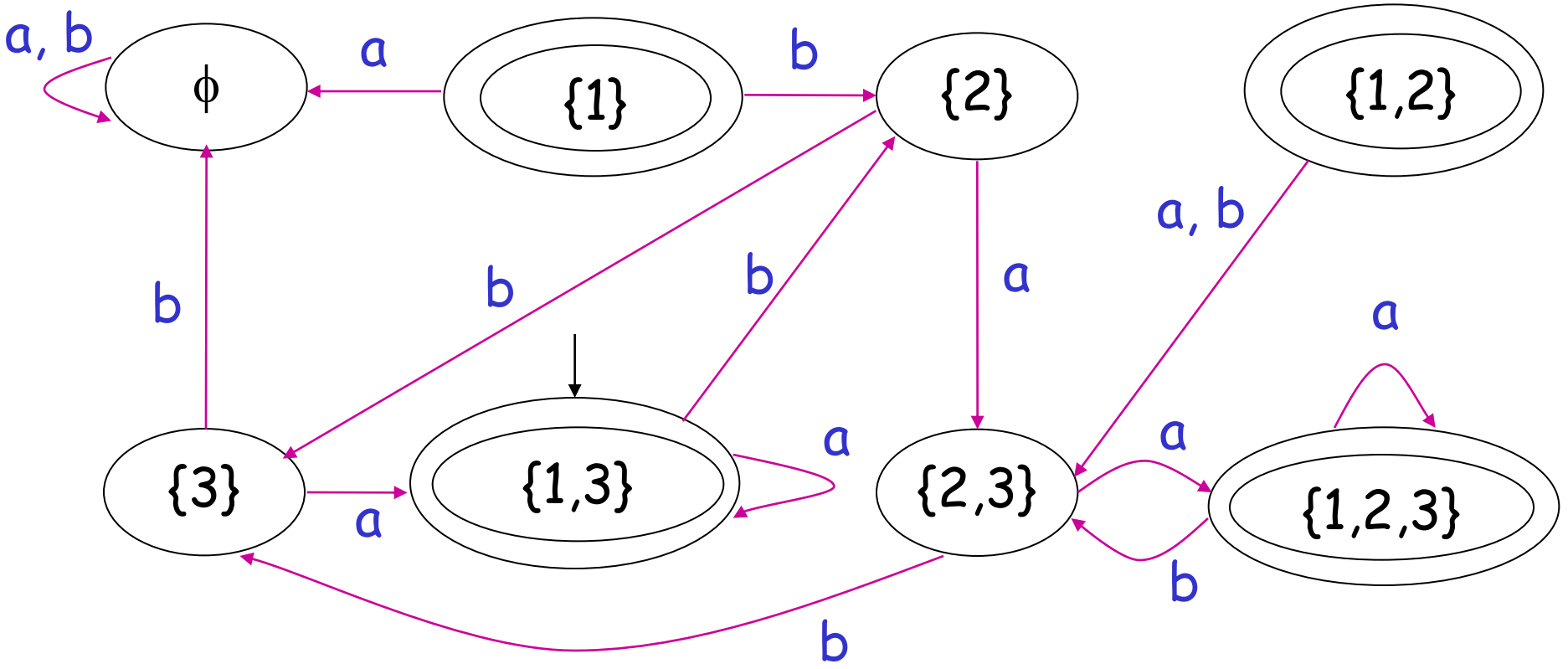
- N_4 has three states {1,2,3}, so we construct DFA D with eight.

- We label each of D's states with the corresponding subset. Thus D's state set is $\{\varnothing,\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$

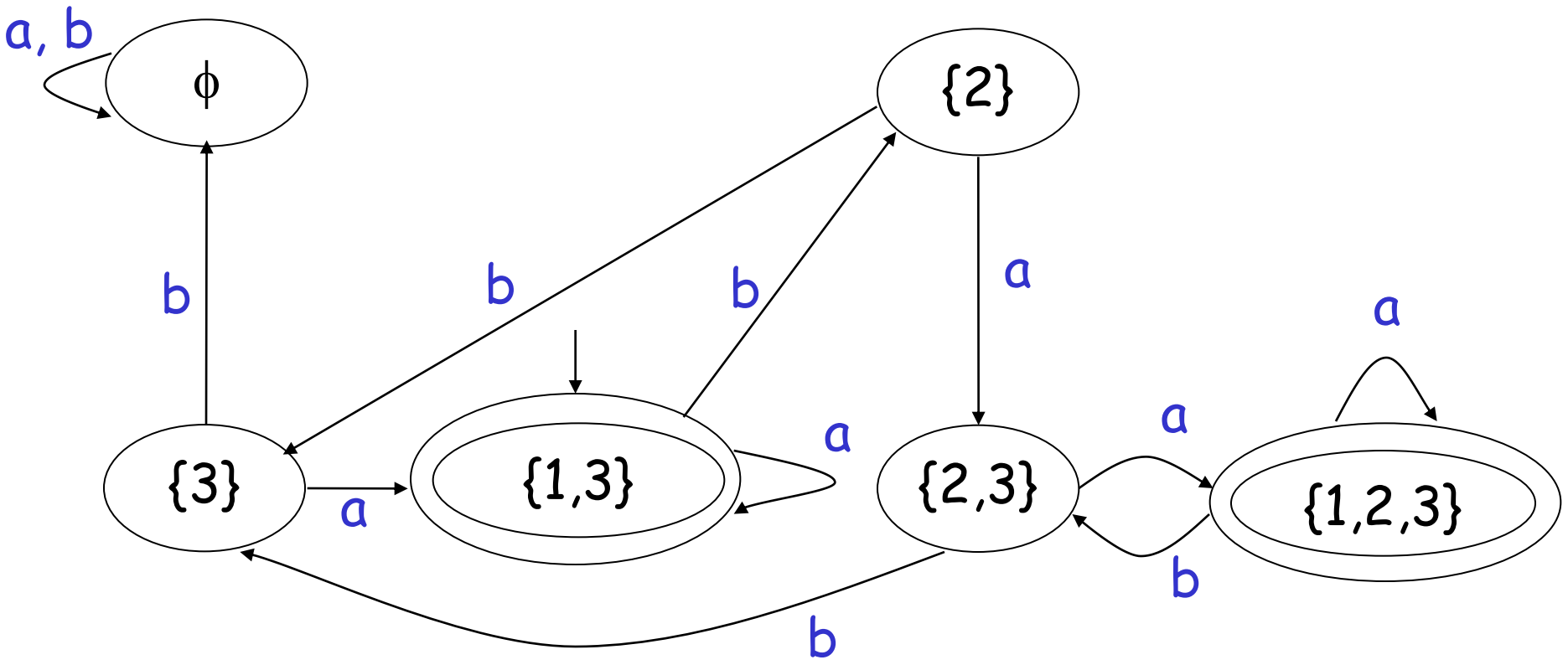
Step2:Determine the start and accept states



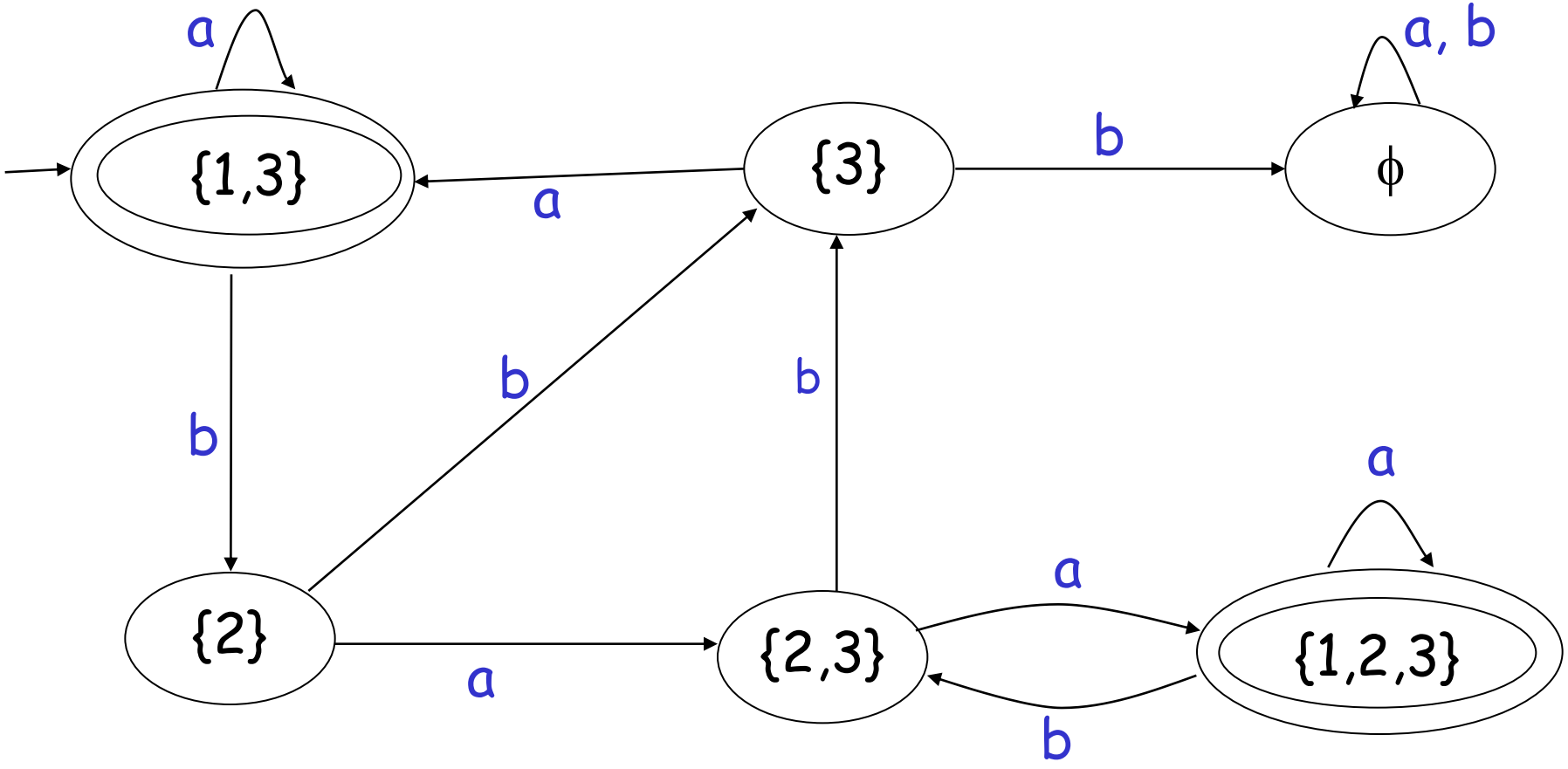
Step3: Determine transition function



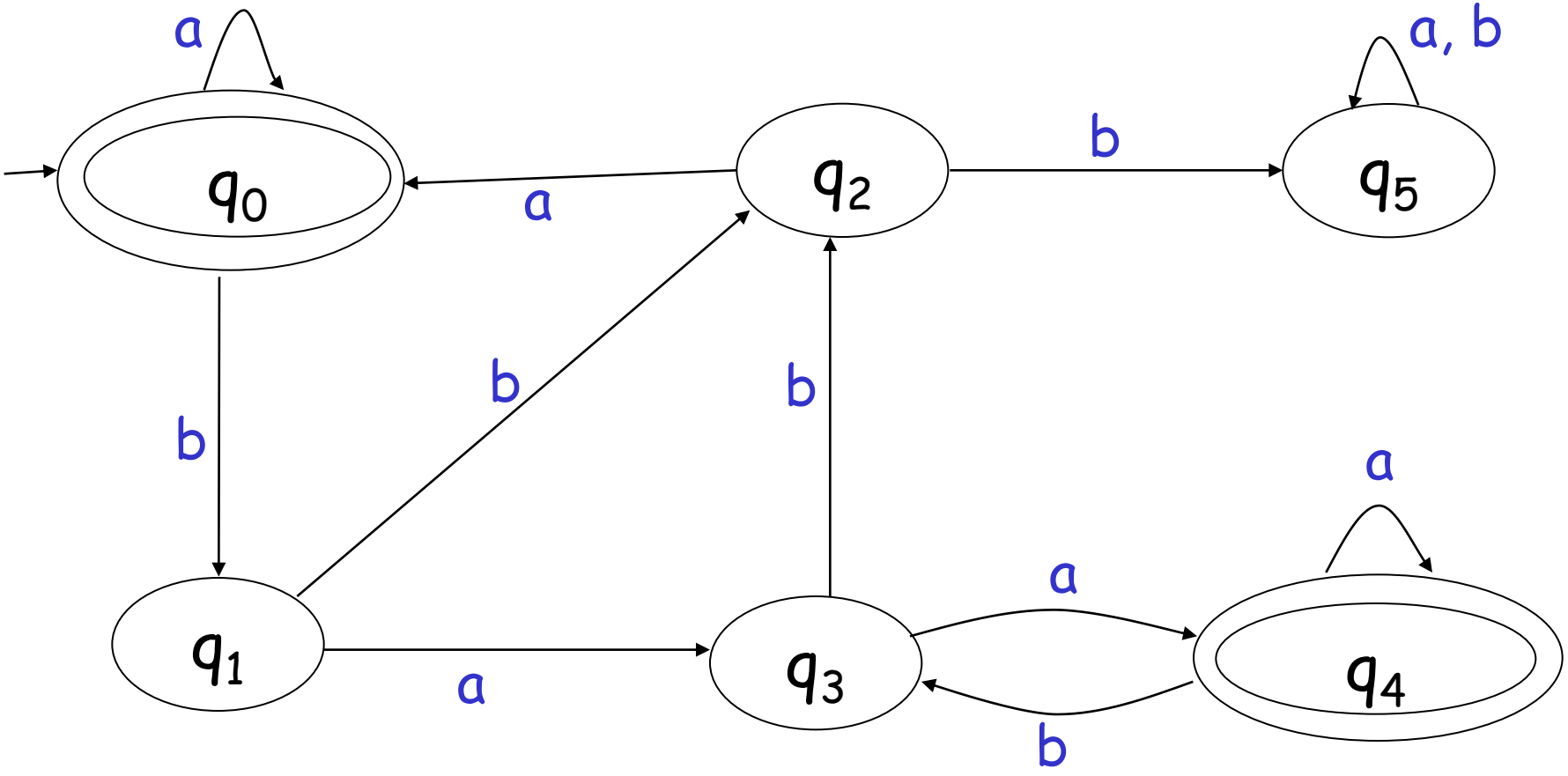
After removing unnecessary states



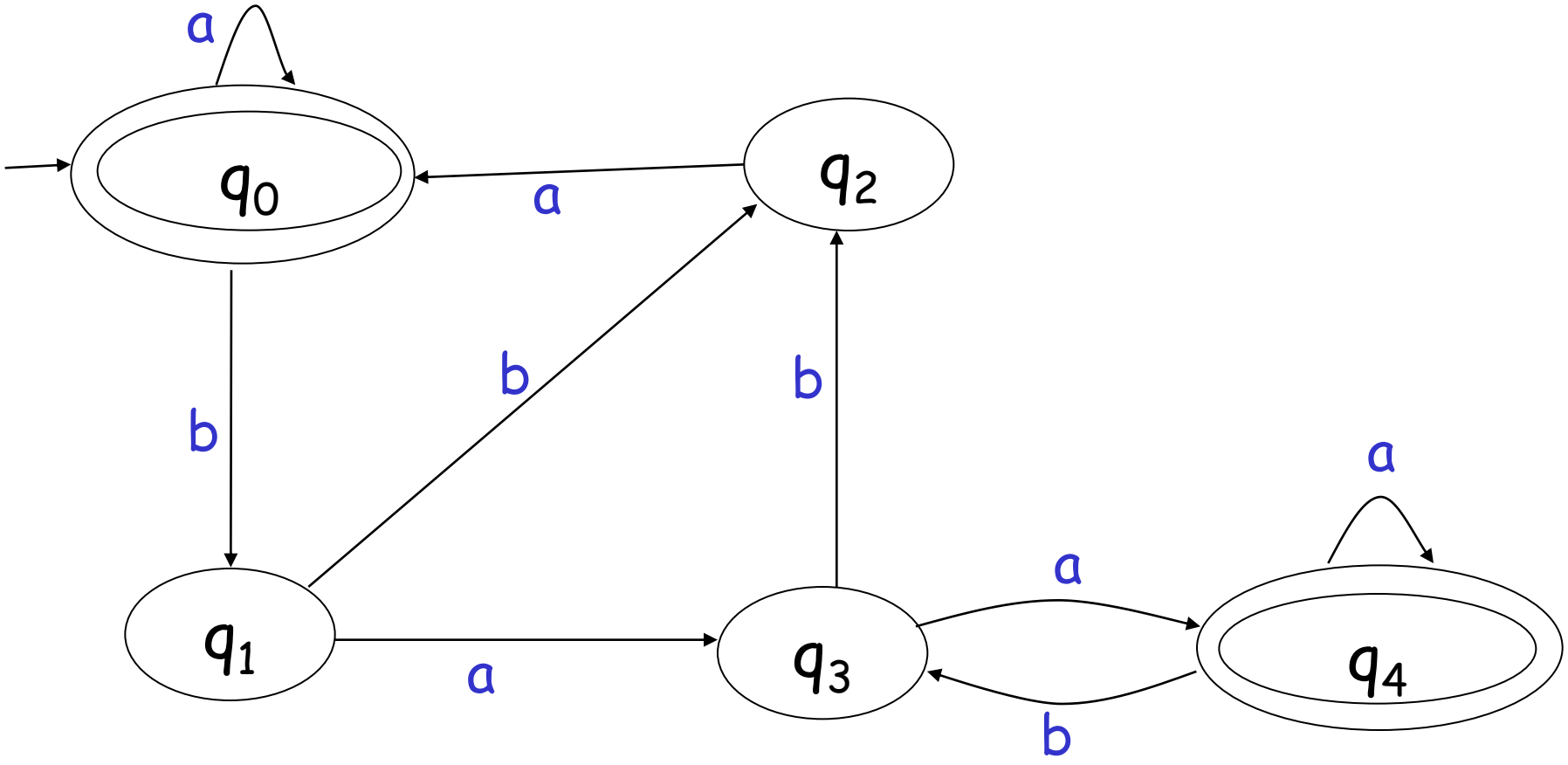
Rearranging states



Renaming states



More simplified



NFA to DFA: Remarks

We are given an NFA M

We want to convert it
to an equivalent DFA M'

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

If the NFA has states

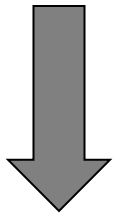
$$q_0, q_1, q_2, \ldots$$

the DFA has states in the powerset

$$\varnothing, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots.$$

Procedure NFA to DFA

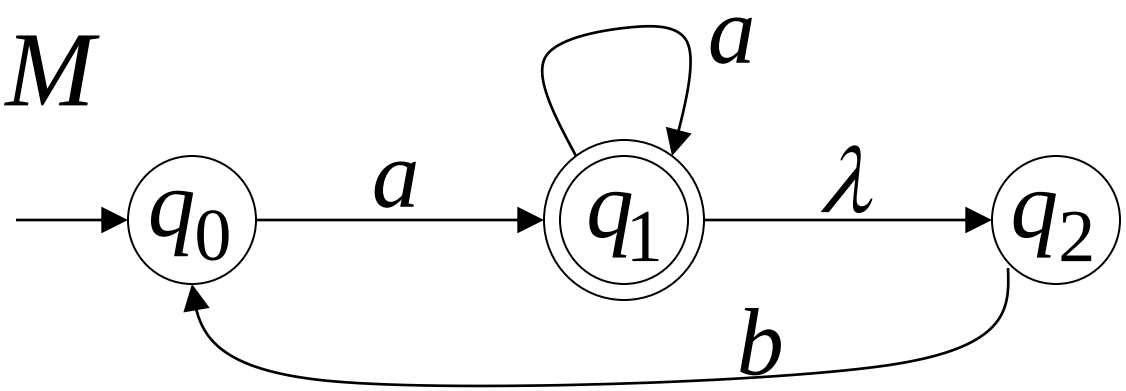
1. Initial state of NFA: q_0



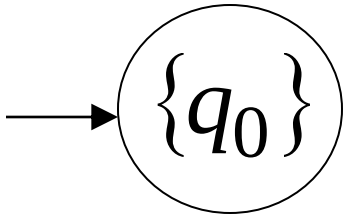
Initial state of DFA: $\{q_0\}$

Example

NFA M



DFA M'



Procedure NFA to DFA

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

Compute in the NFA

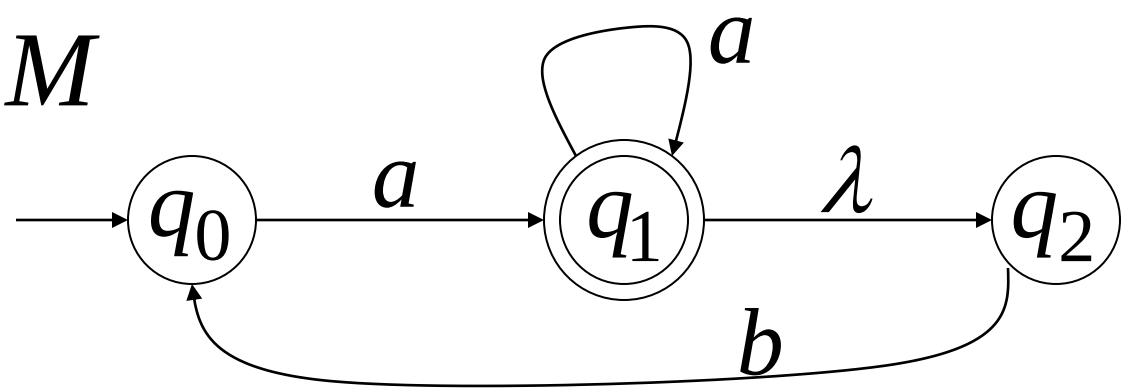
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to DFA

$$\delta(\{q_i, q_j, \ldots, q_m\},\ a) = \{q'_i, q'_j, \ldots, q'_m\}$$

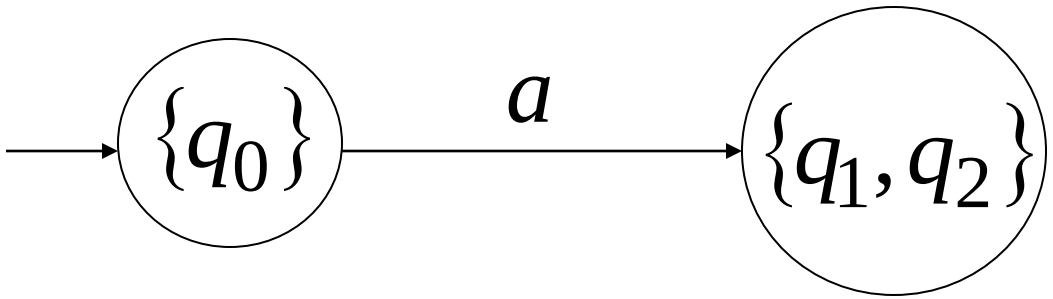
Exampe

NFA M



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

DFA M'



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

$$\delta^*(2, b) = \{3\},\ \delta^*(3, b) = \varphi$$

$$\delta(\{2,\ 3\},\ b) = \{3\} \cup \varphi = \{3\}$$

Procedure NFA to DFA

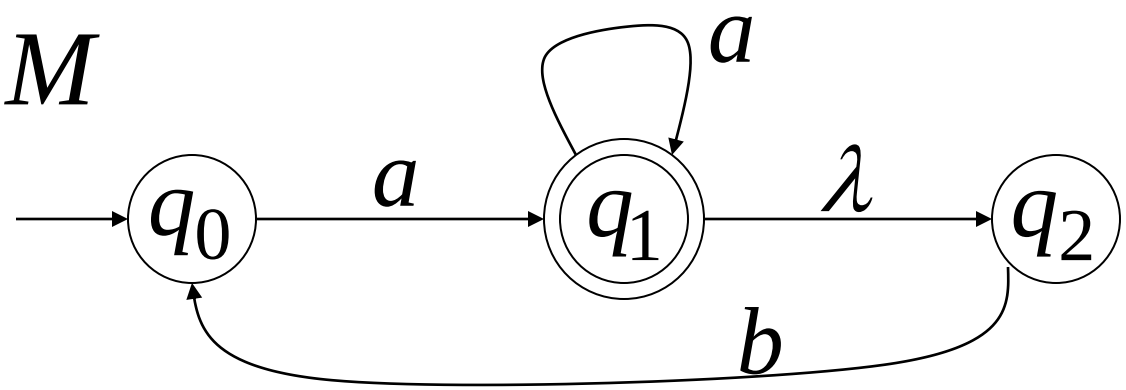
Repeat Step 2 for all letters in alphabet,

until

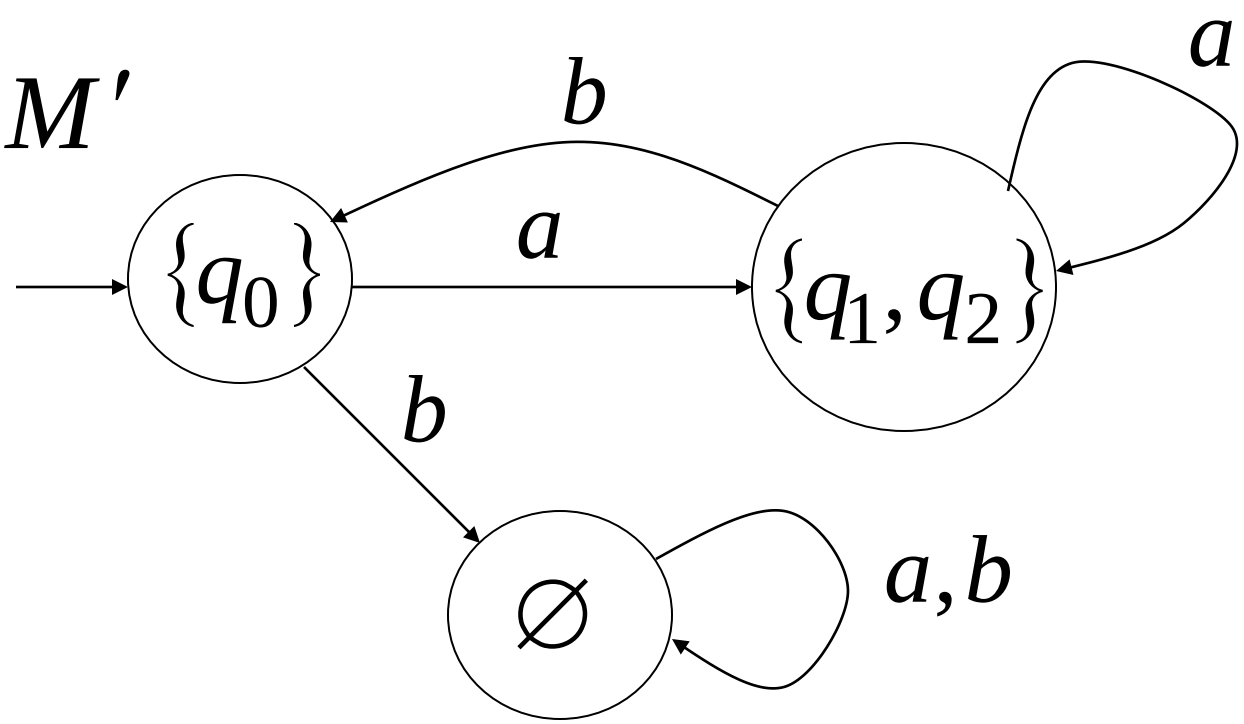
no more transitions can be added.

Example

NFA M



DFA M'



Procedure NFA to DFA

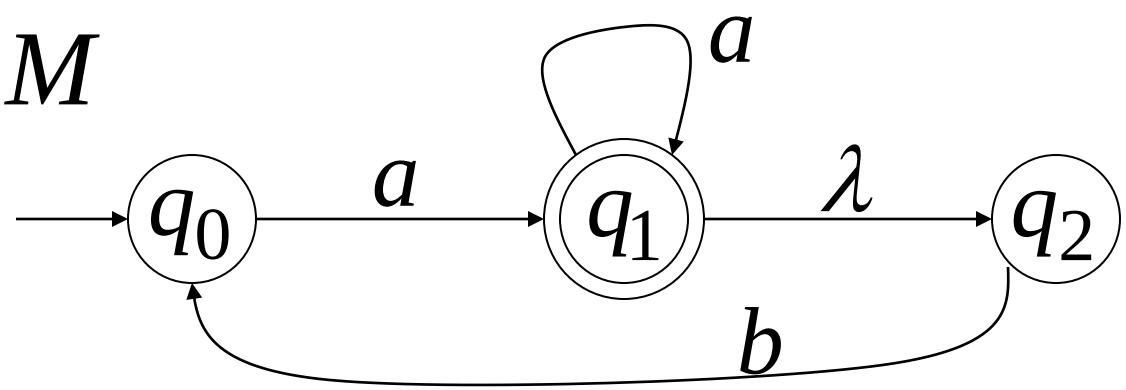
3. For any DFA state $\{q_i, q_j, \ldots, q_m\}$

If some q_j is a final state in the NFA

Then, $\{q_i, q_j, \ldots, q_m\}$
is a final state in the DFA

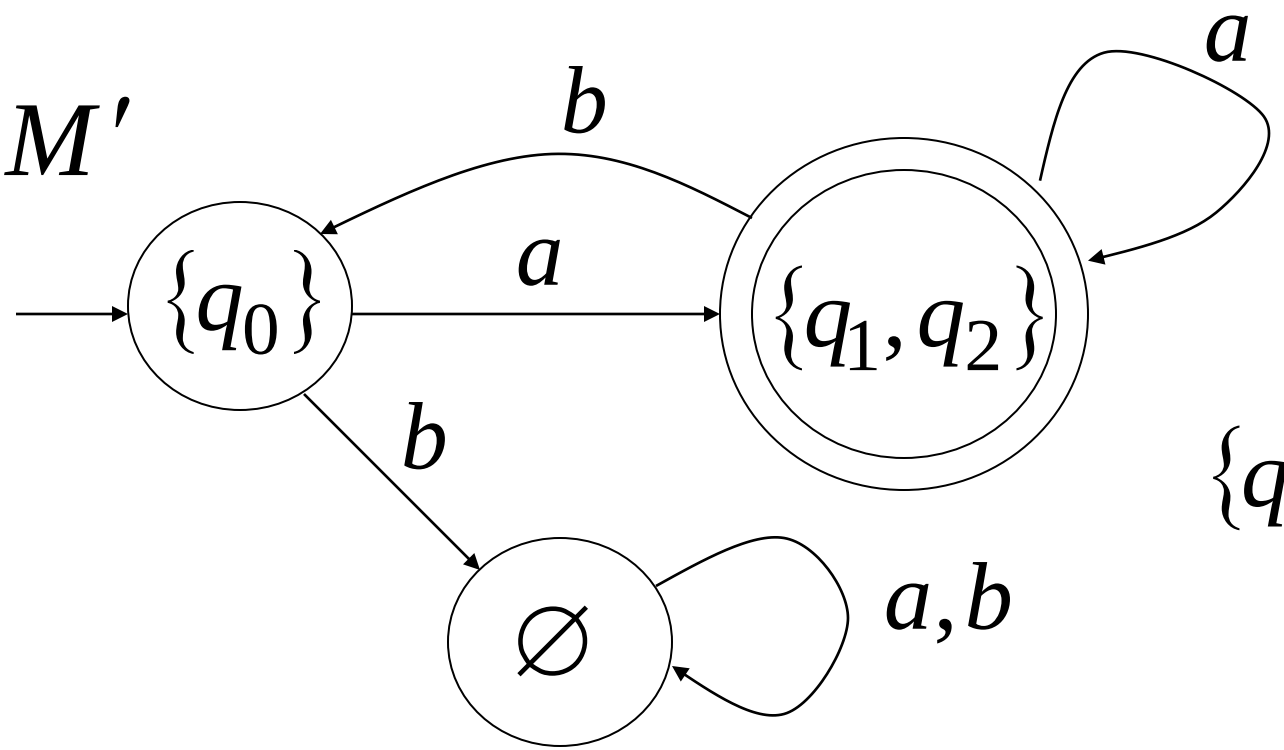
Example

NFA M



$q_1 \in F$

DFA M'



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

Theorem

Take NFA M

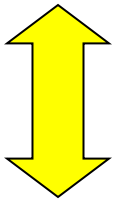
Apply procedure to obtain DFA M'

Then M and M' are equivalent :

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

Proof

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



$$L(M) \subseteq L(M') \quad \text{AND} \quad L(M) \supseteq L(M')$$

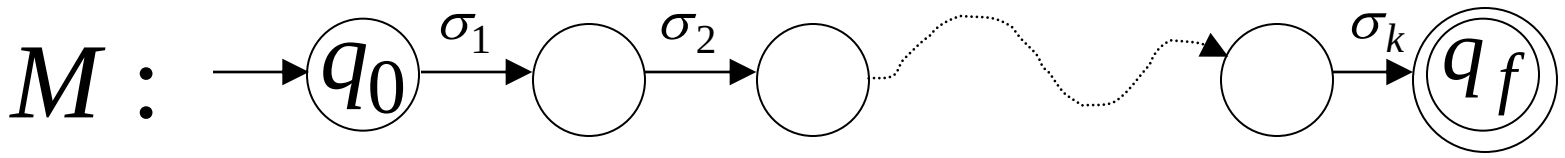
First we show: $L(M) \subseteq L(M')$

Take arbitrary: $w \in L(M)$

We will prove: $w \in L(M')$

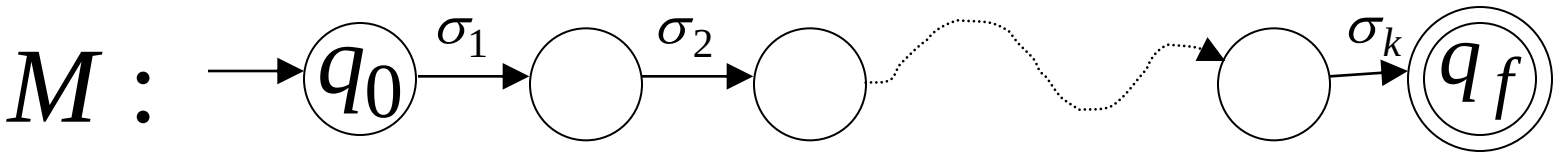
$$w \in L(M)$$

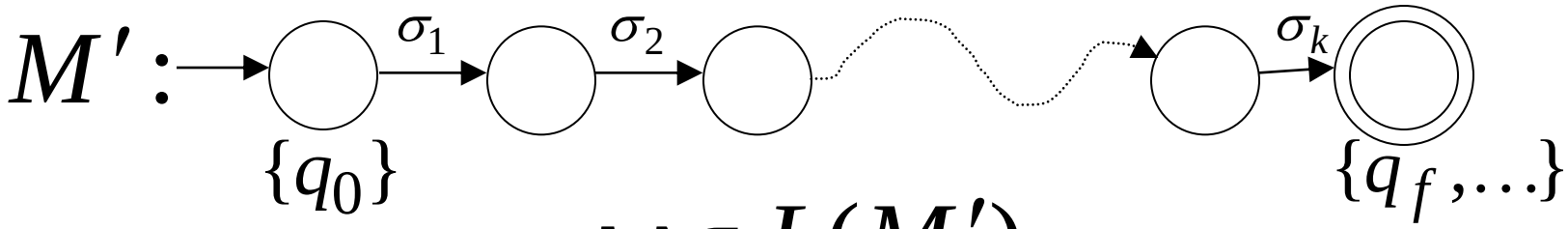
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



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

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

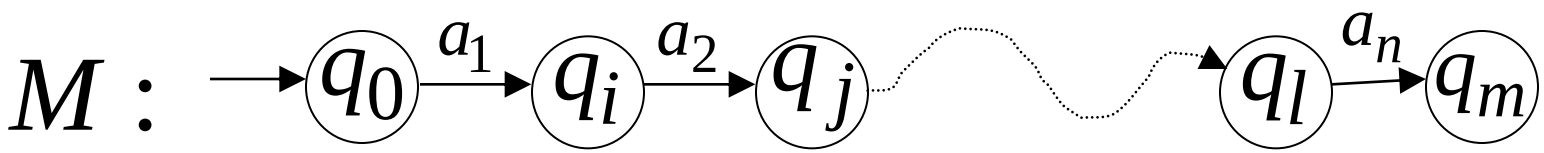


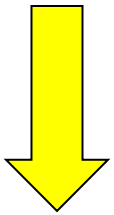


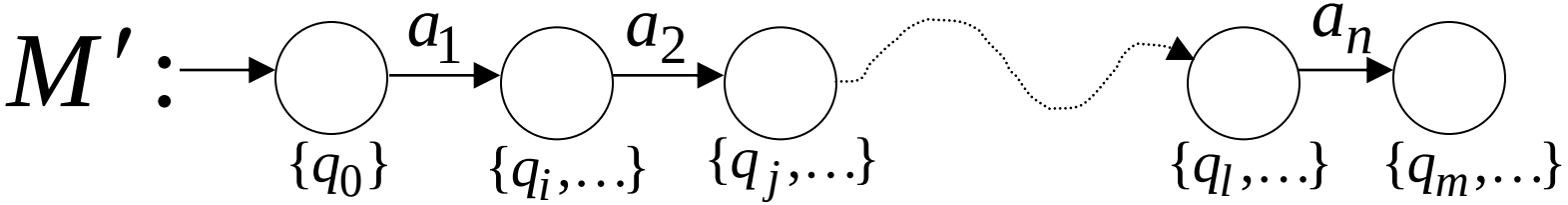
$$w \in L(M')$$

More generally, we will show that if in M:

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

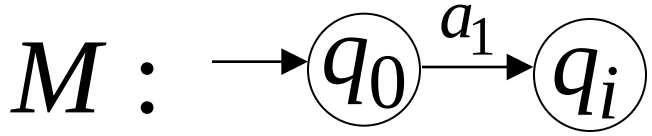


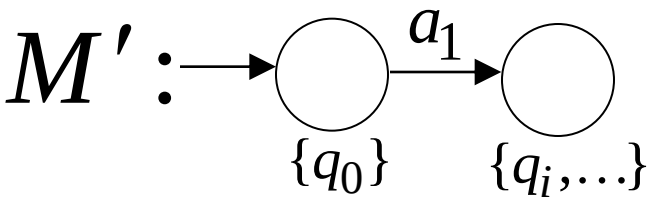




Proof by induction on $|v|$

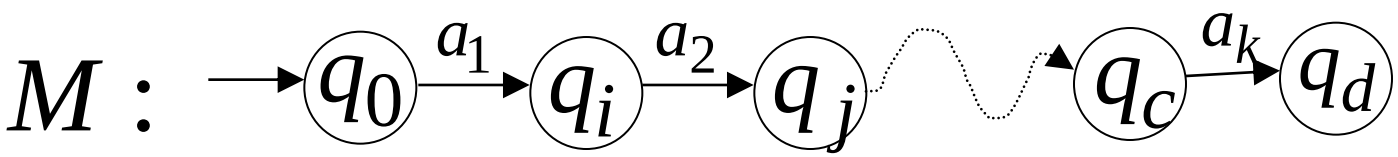
Induction Basis: $v = a_1$

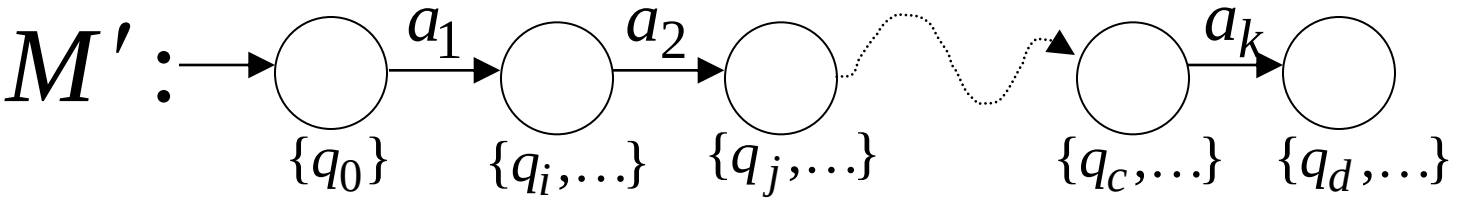




Induction hypothesis: $1 \leq |v| \leq k$

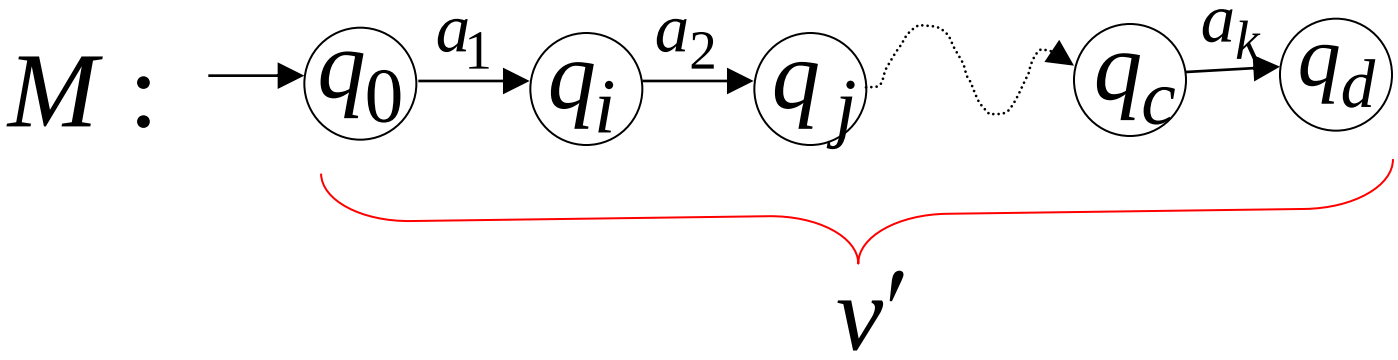
$$v = a_1 a_2 \cdots a_k$$

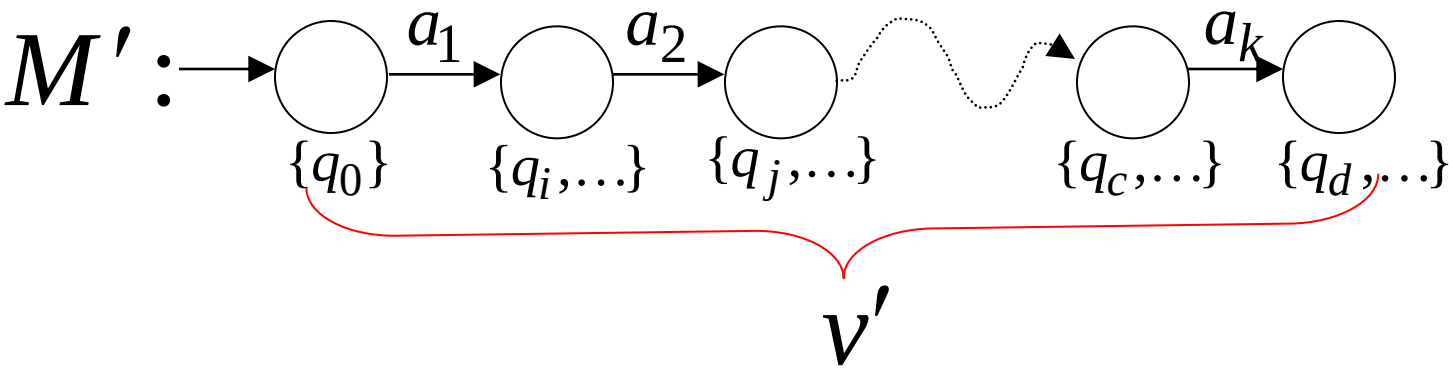




Induction Step: $|v| = k+1$

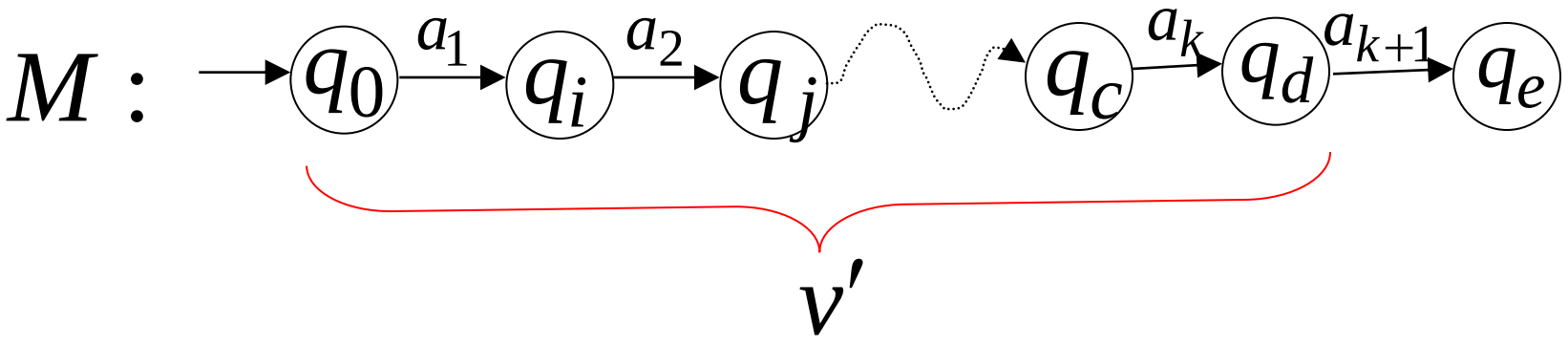
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

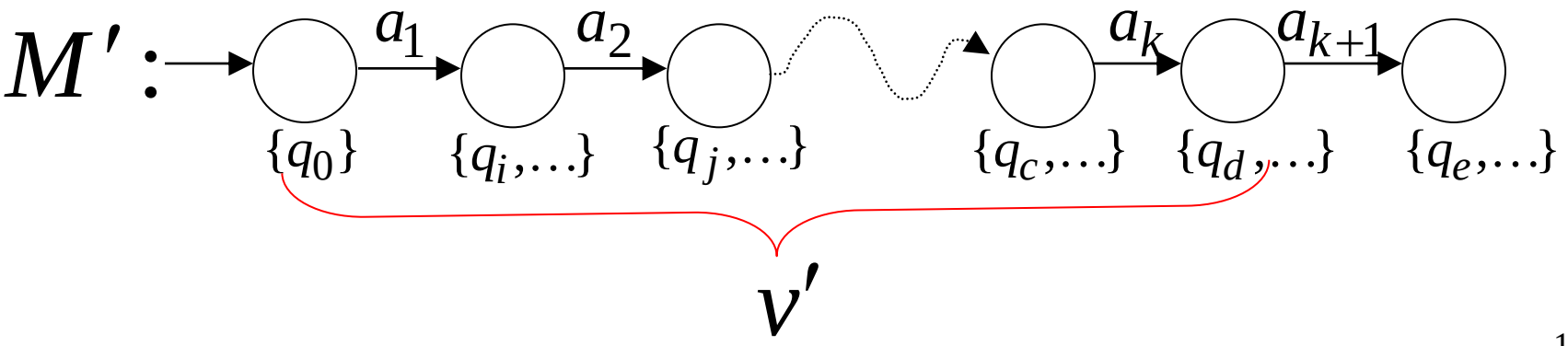




Induction Step: $|v| = k+1$

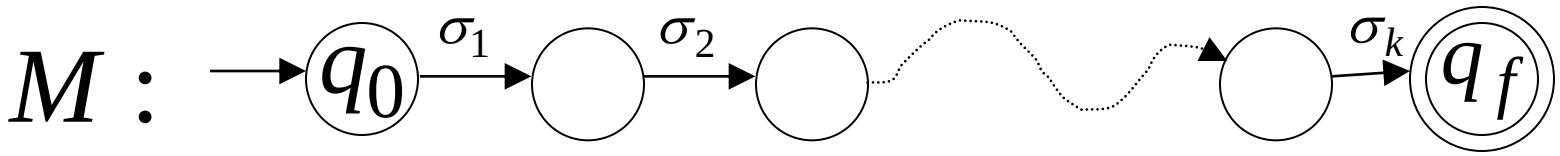
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

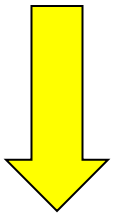


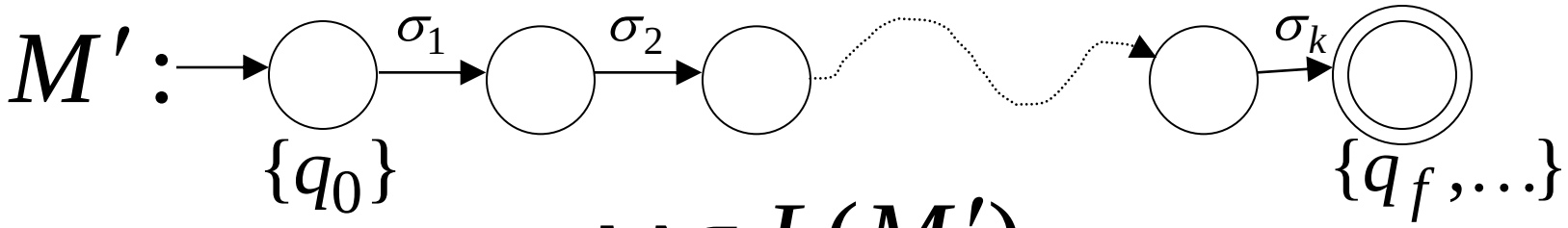


Therefore if $w \in L(M)$

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







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

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

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

(proof is similar)

The End