2021

Theory of Computation

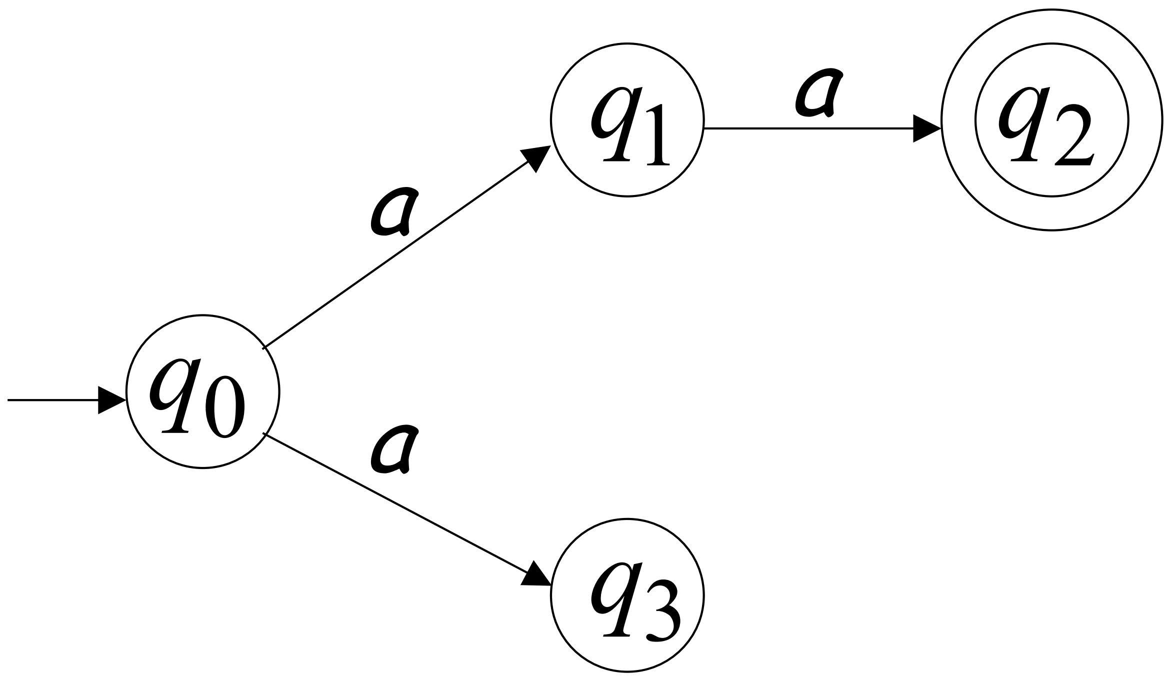
Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1. Deterministic Finite Accepters (DFA)
2. Nondeterministic Finite Accepters (NFA)
3. Equivalence of DFA and NFA
4. Reduction of the Number of States in FA*

Nondeterministic Finite Accepter (NFA)

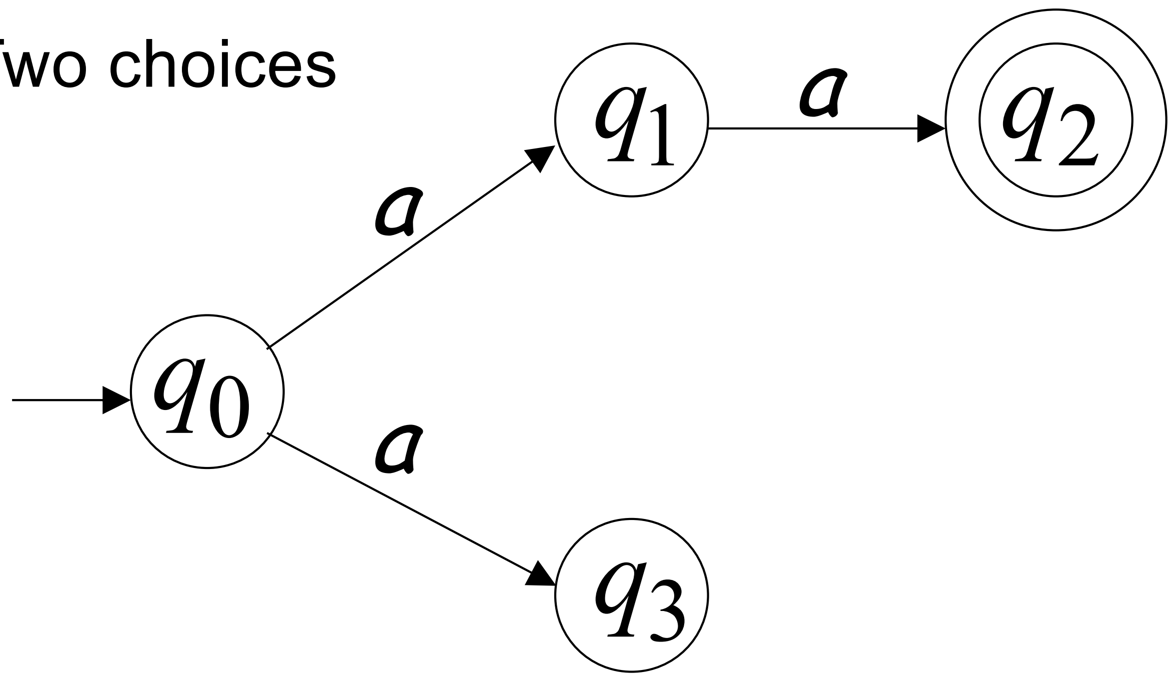
Alphabet = $\{a\}$



Nondeterministic Finite Accepter (NFA)

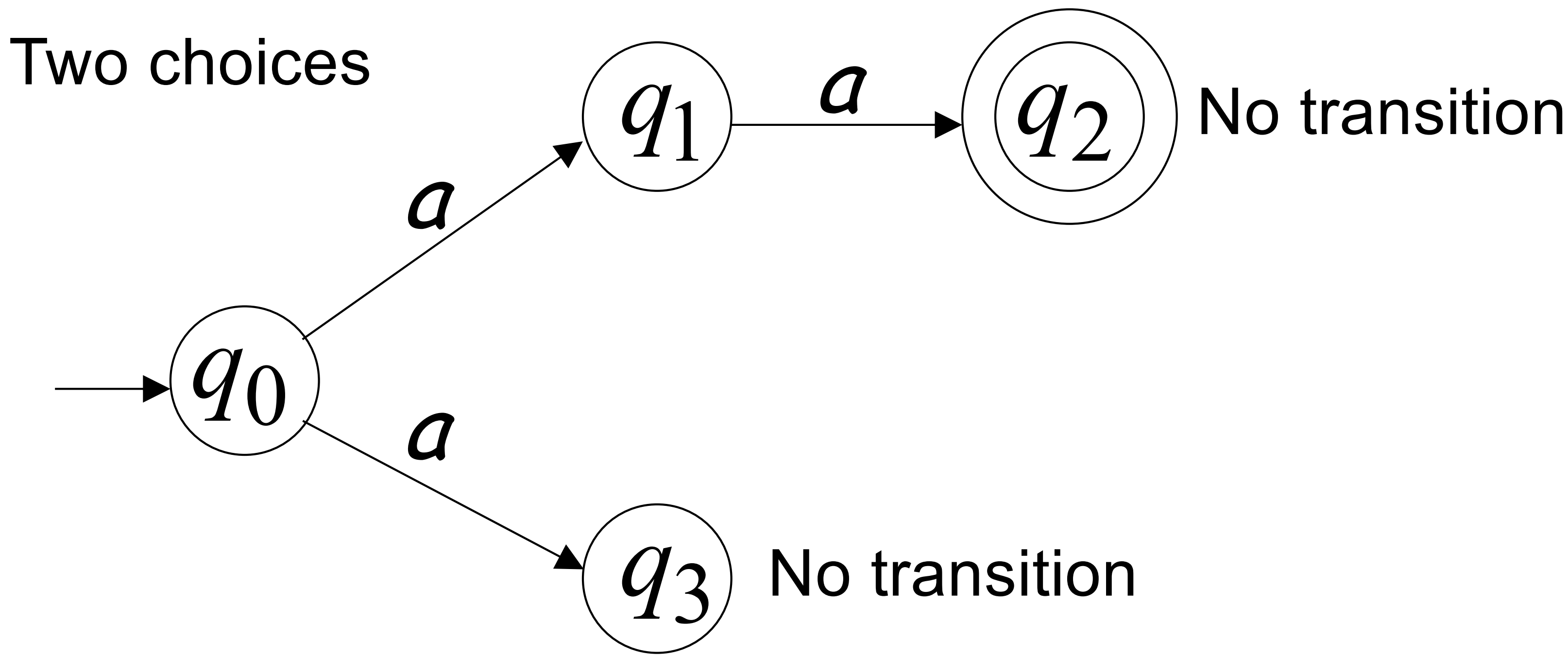
Alphabet = $\{a\}$

Two choices

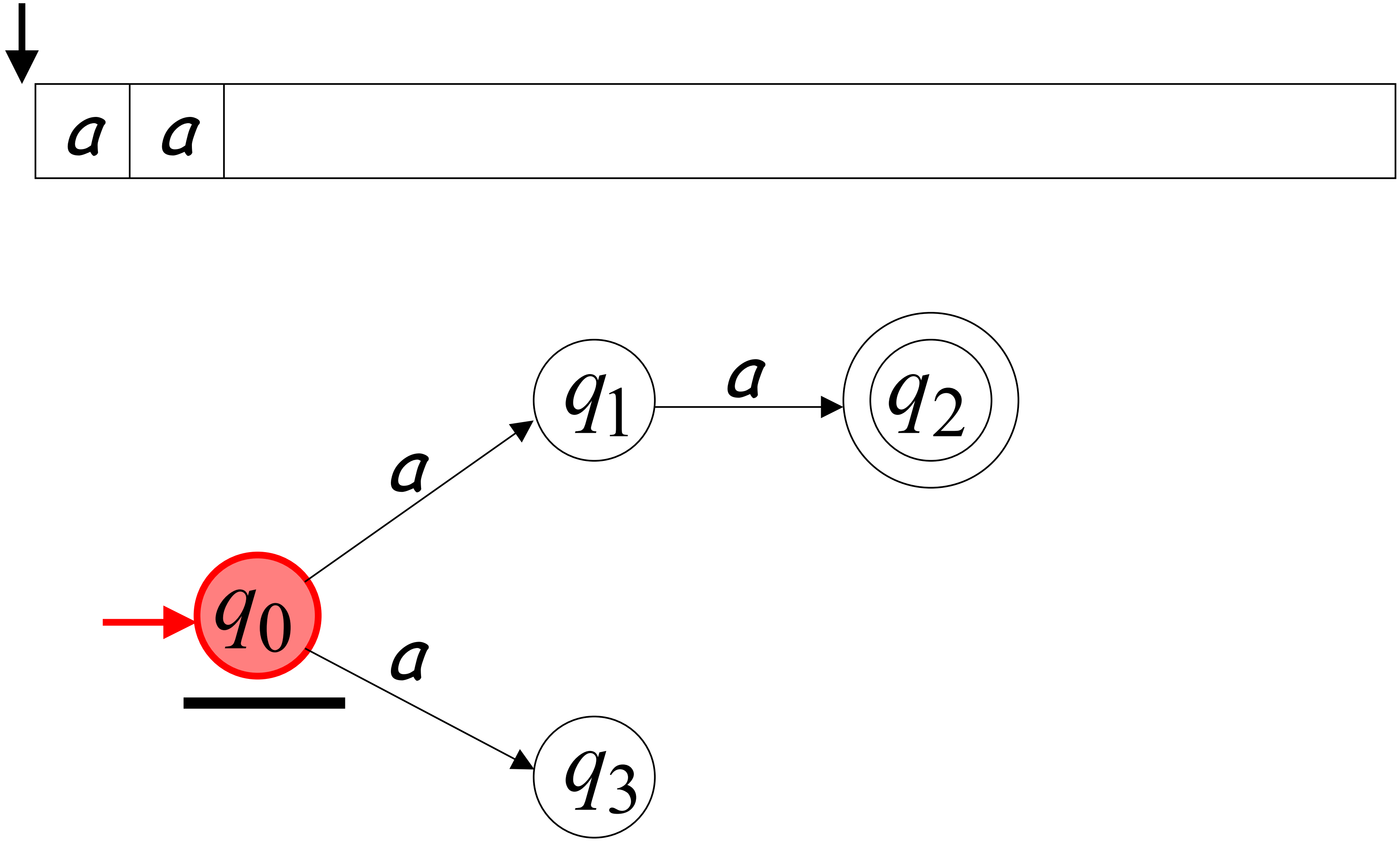


Nondeterministic Finite Accepter (NFA)

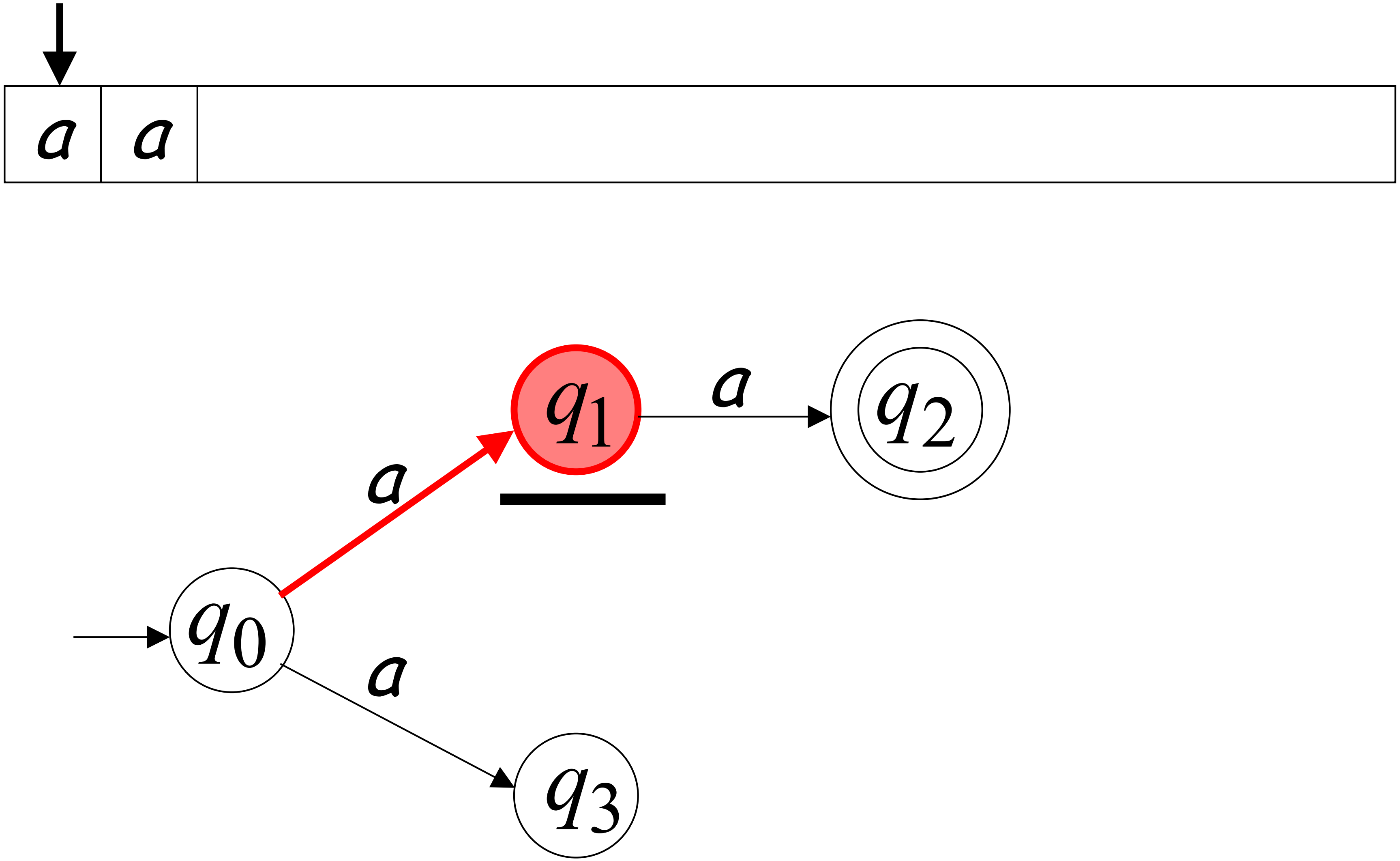
Alphabet = $\{a\}$



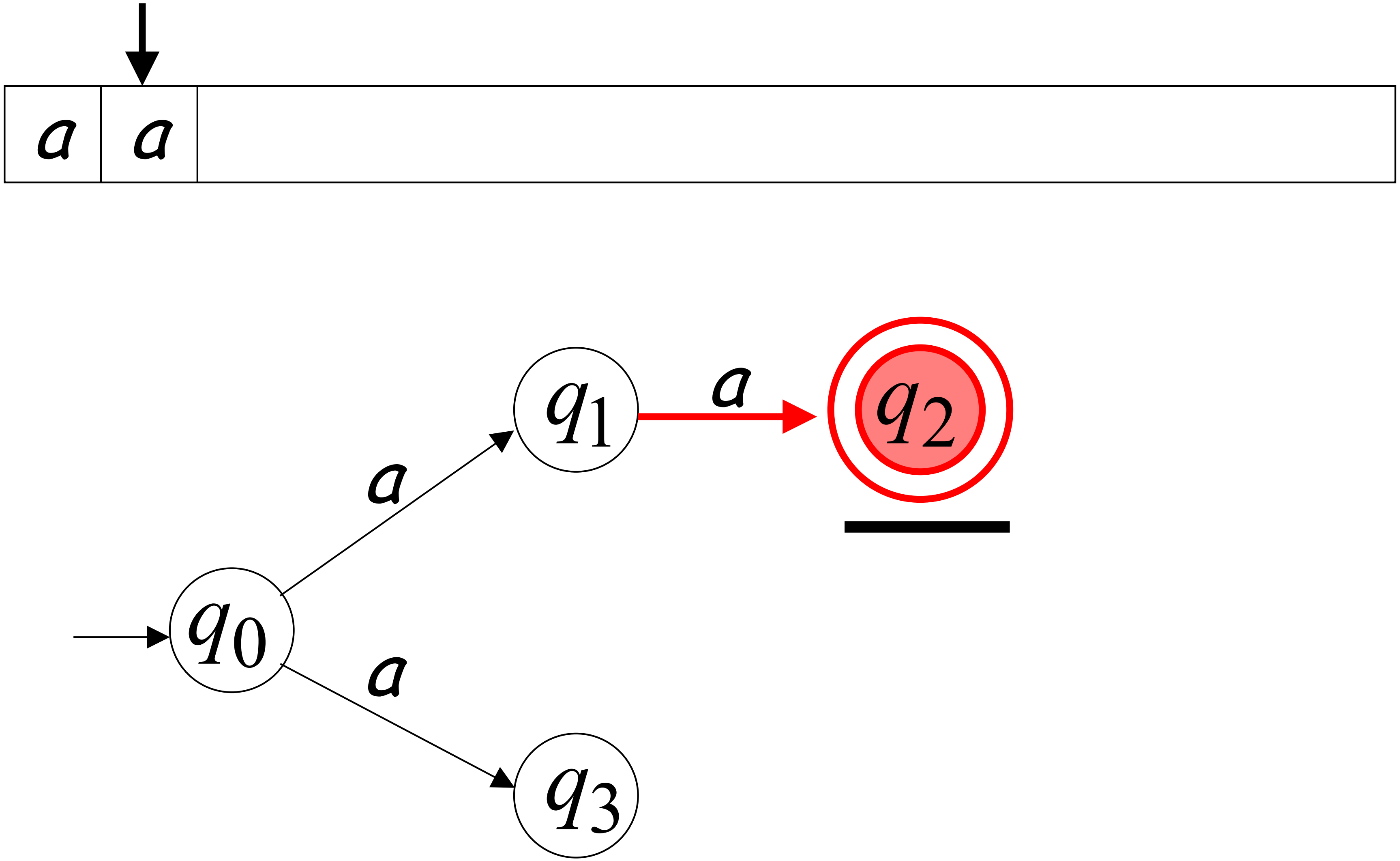
First Choice



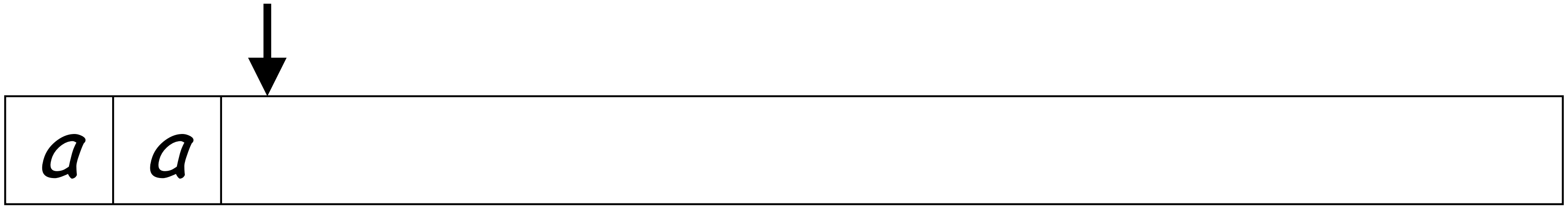
First Choice



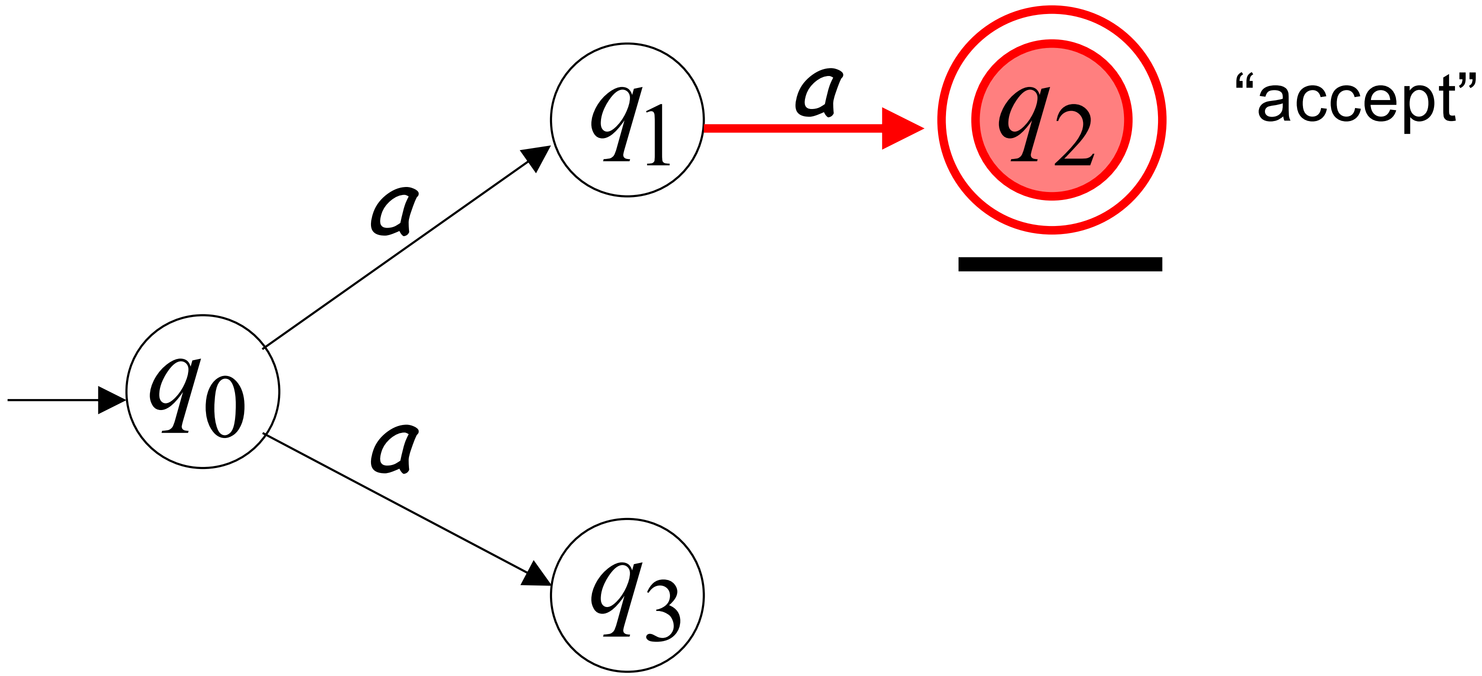
First Choice



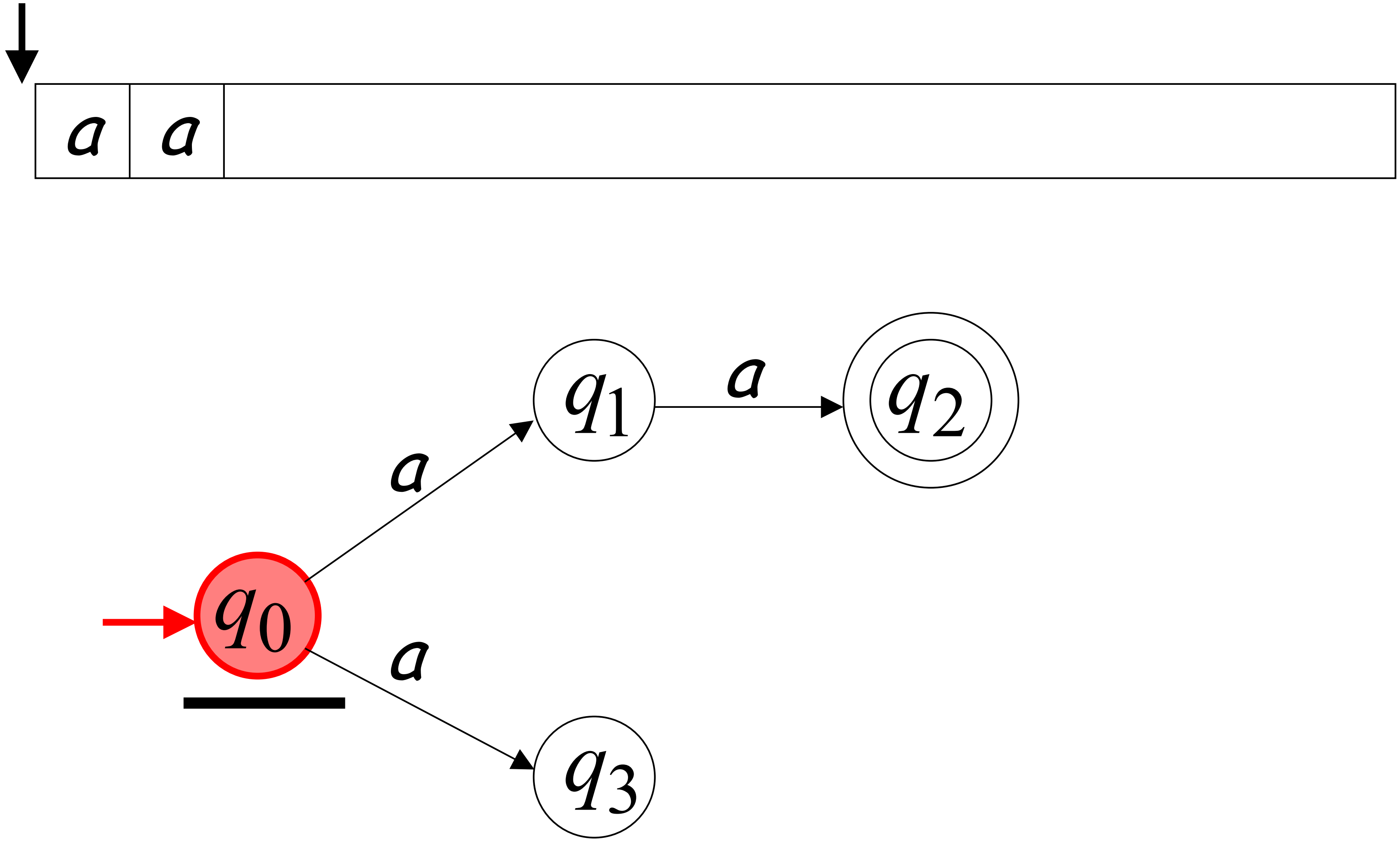
First Choice



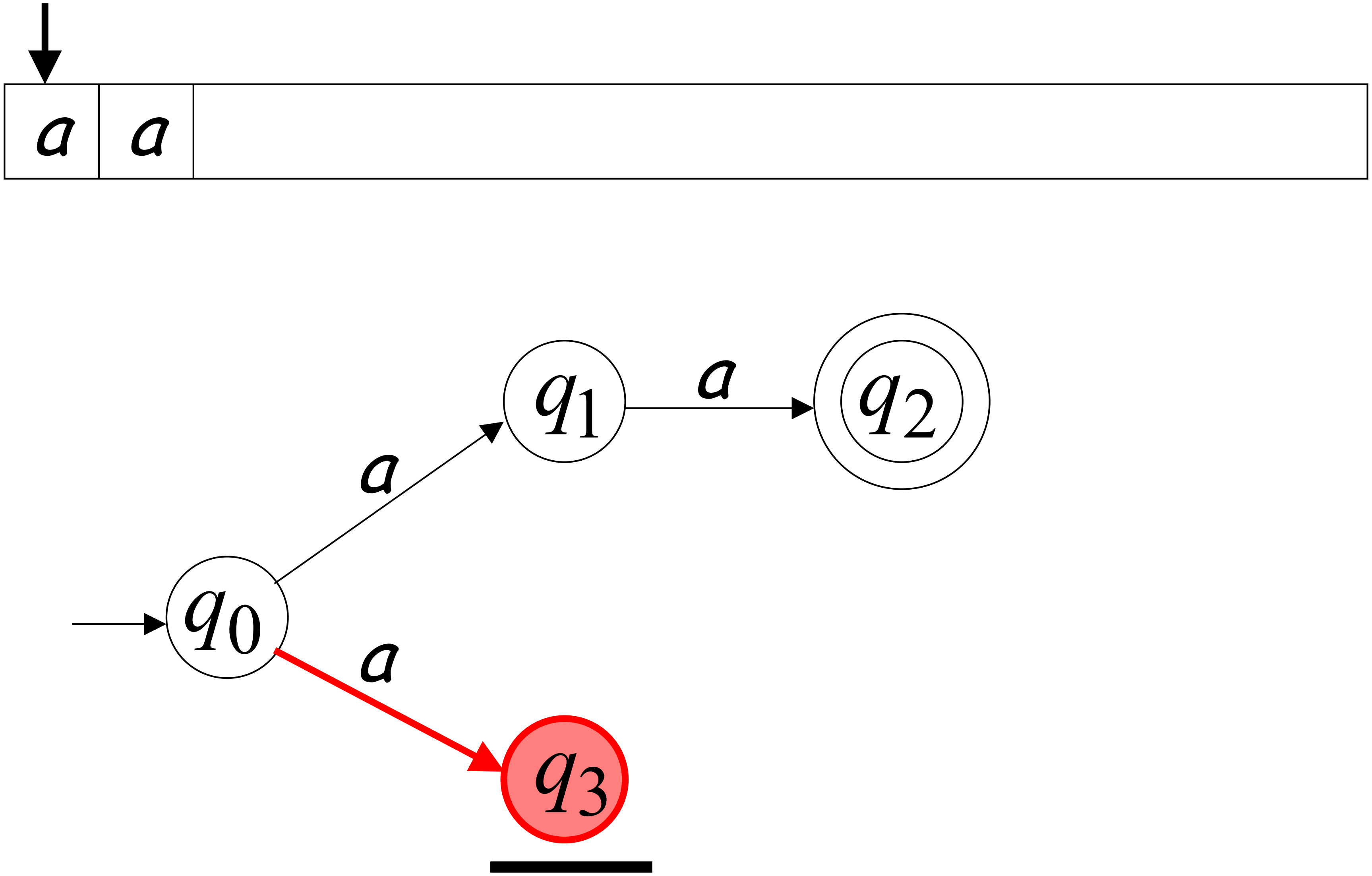
All input is consumed



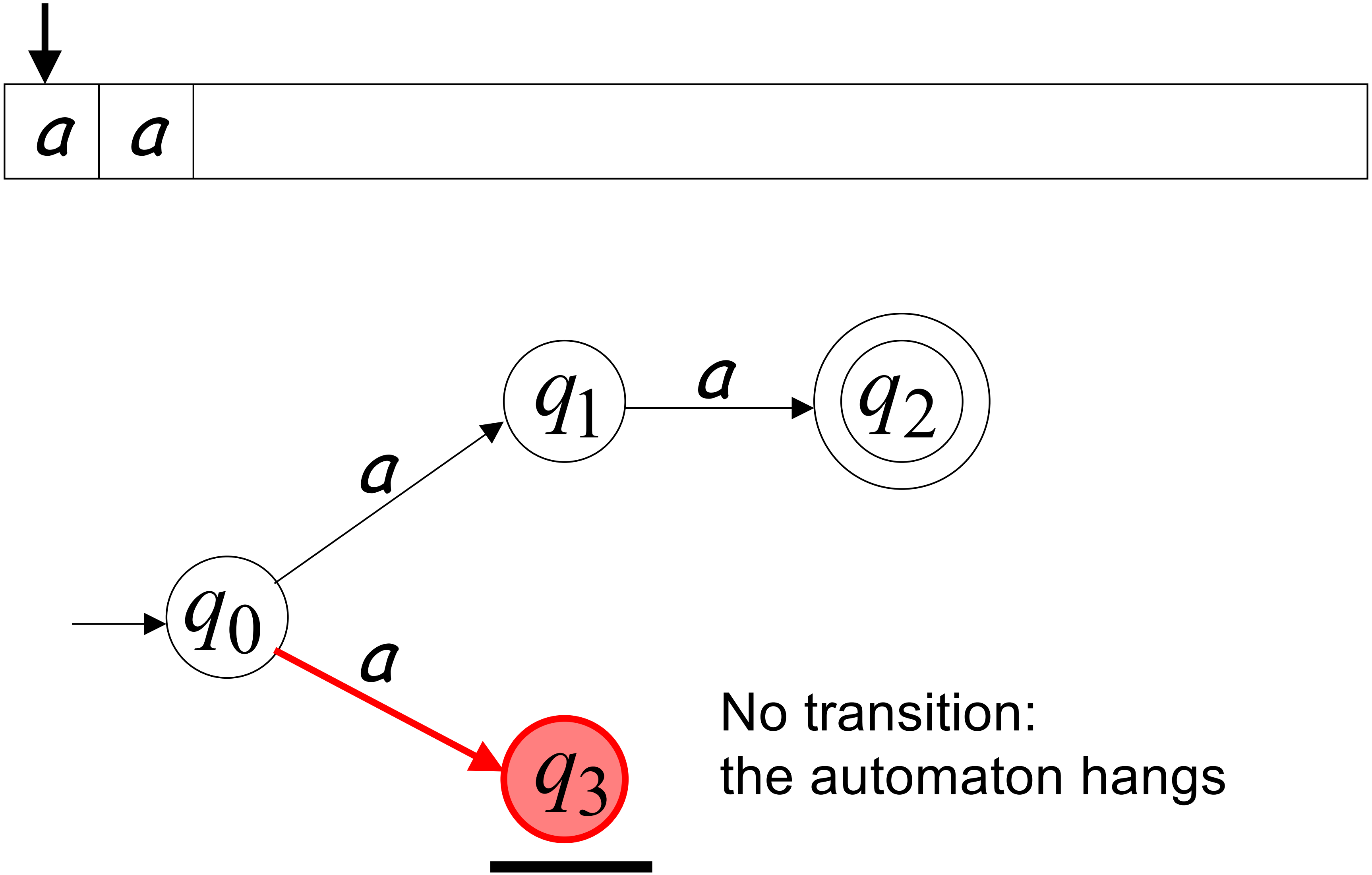
Second Choice



Second Choice

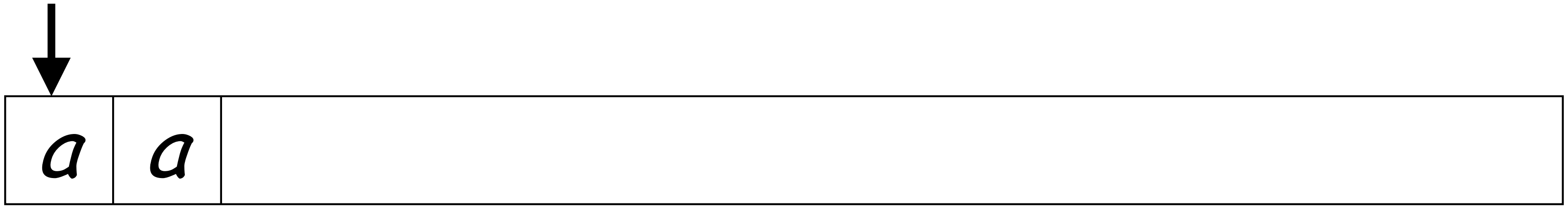


Second Choice

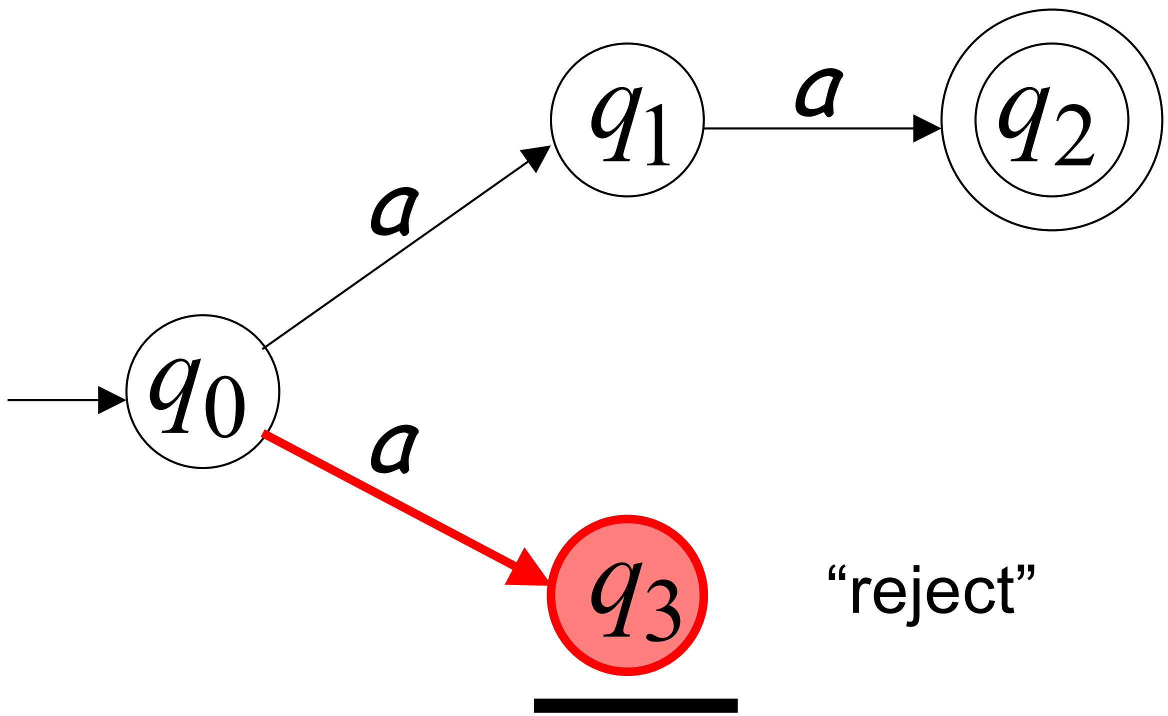


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



An NFA accepts a string:
when there is a computation of the NFA that accepts the string

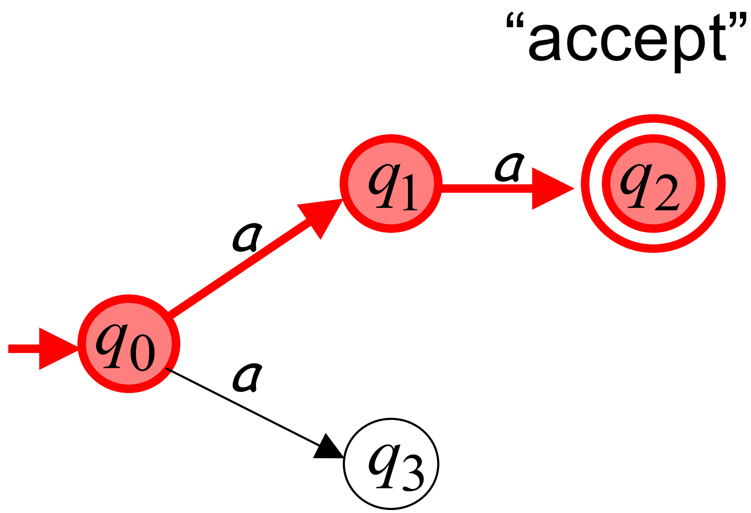
all the input is consumed

AND

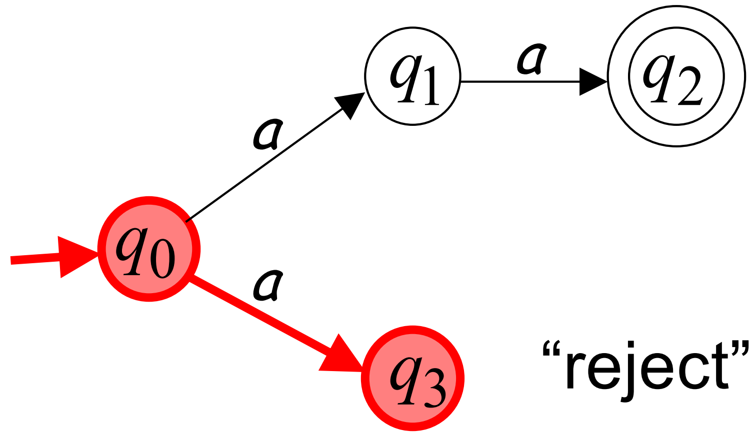
the automaton is in a final state

Example

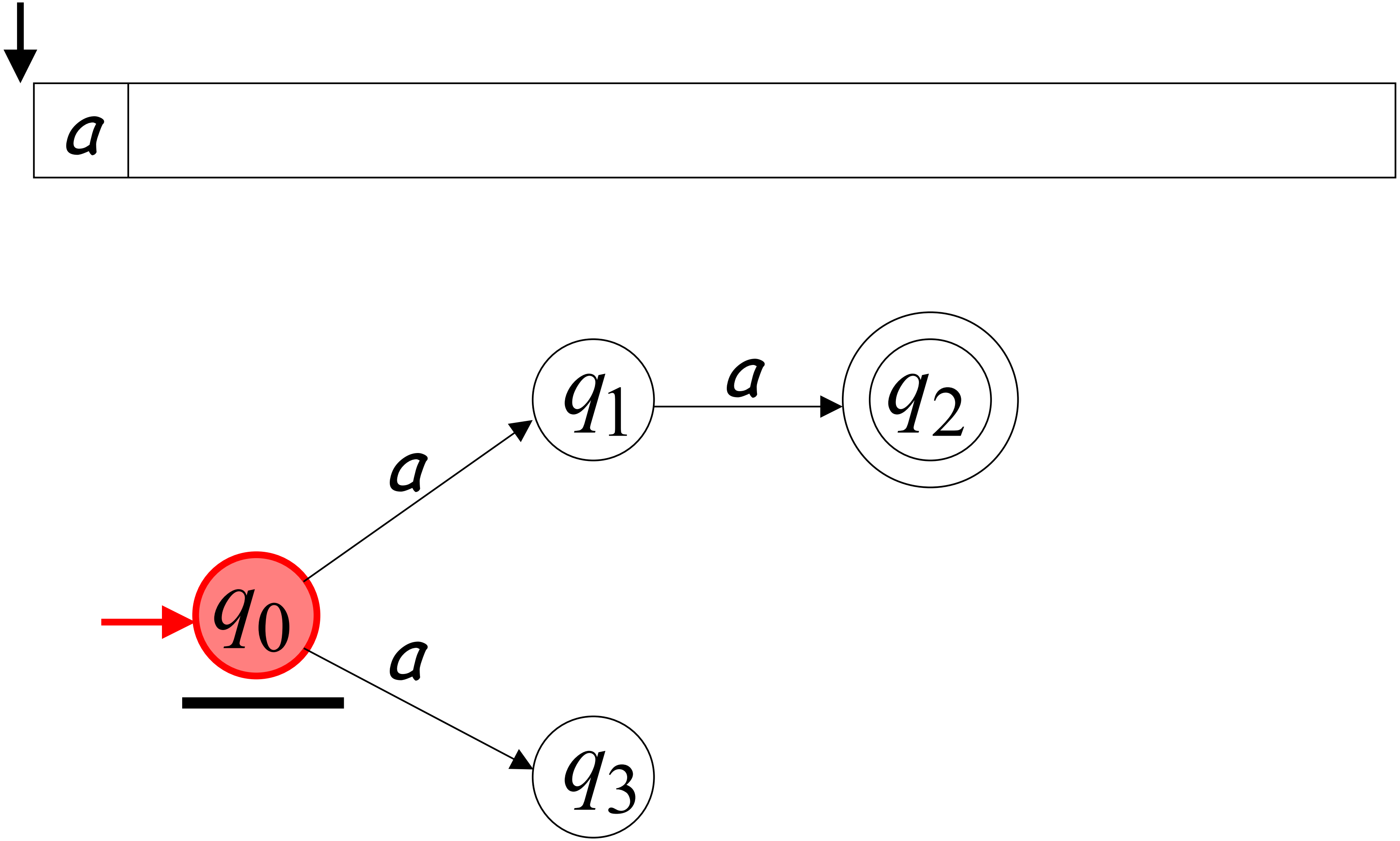
aa is accepted by the NFA:



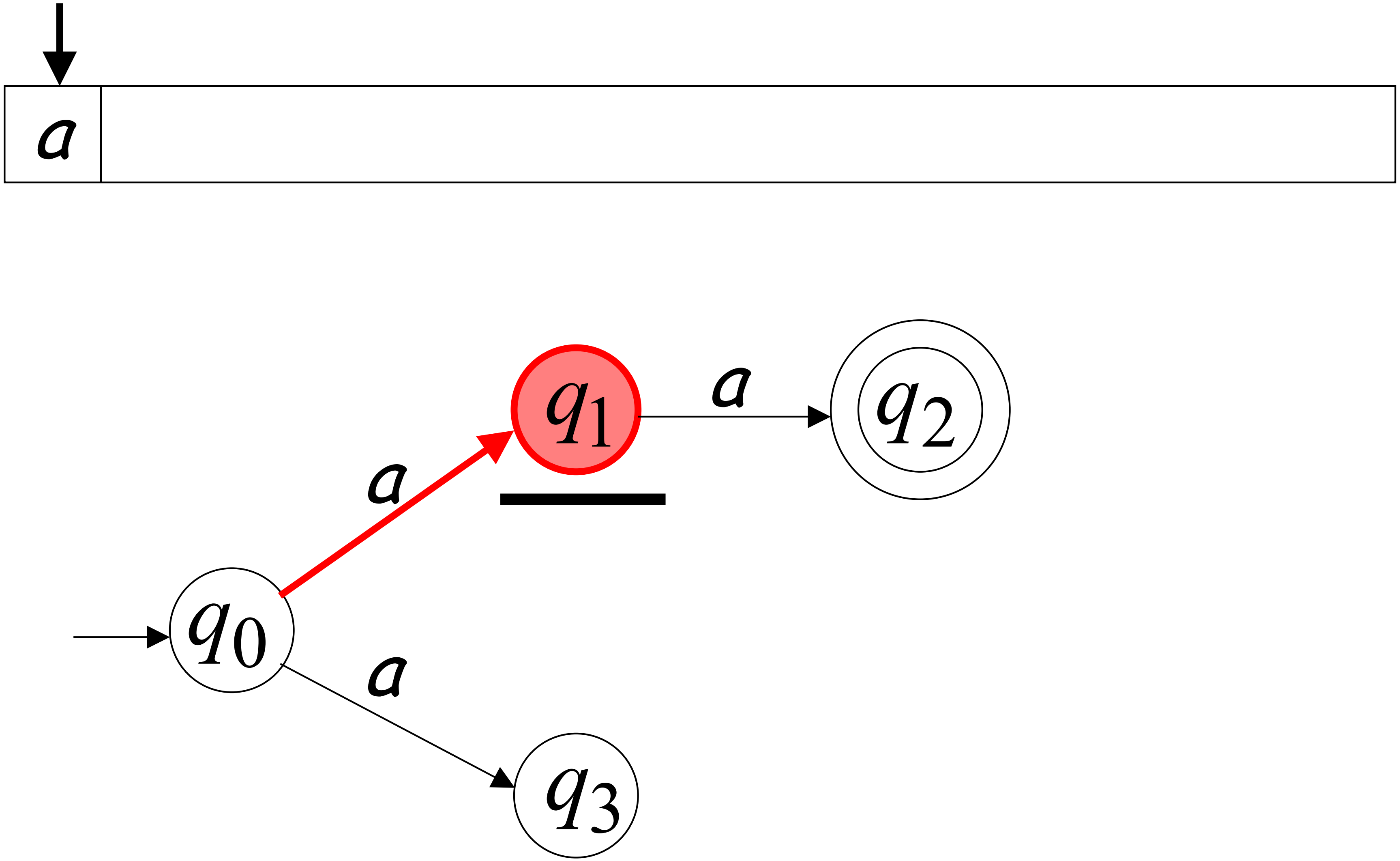
because this computation accepts aa



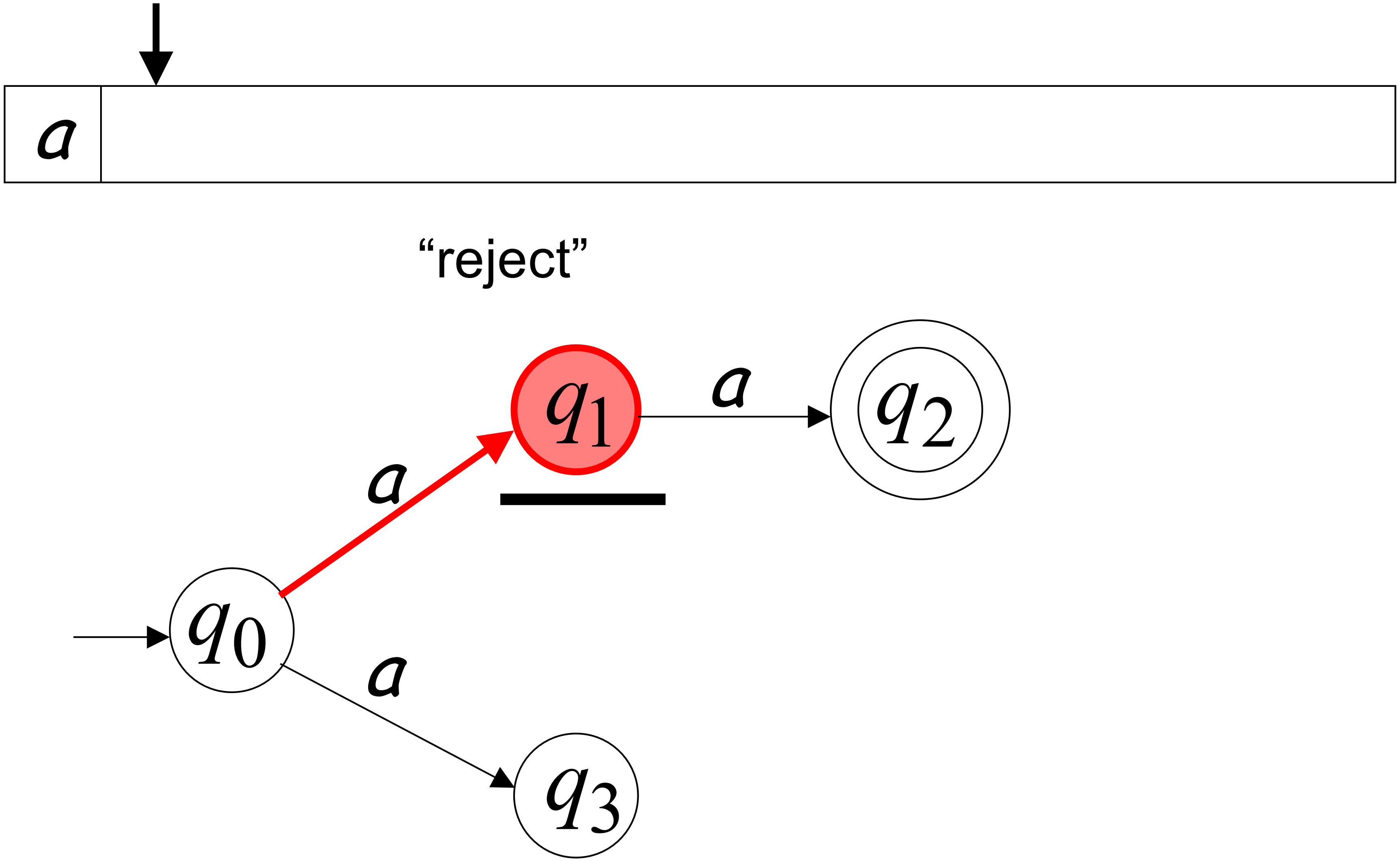
Rejection example



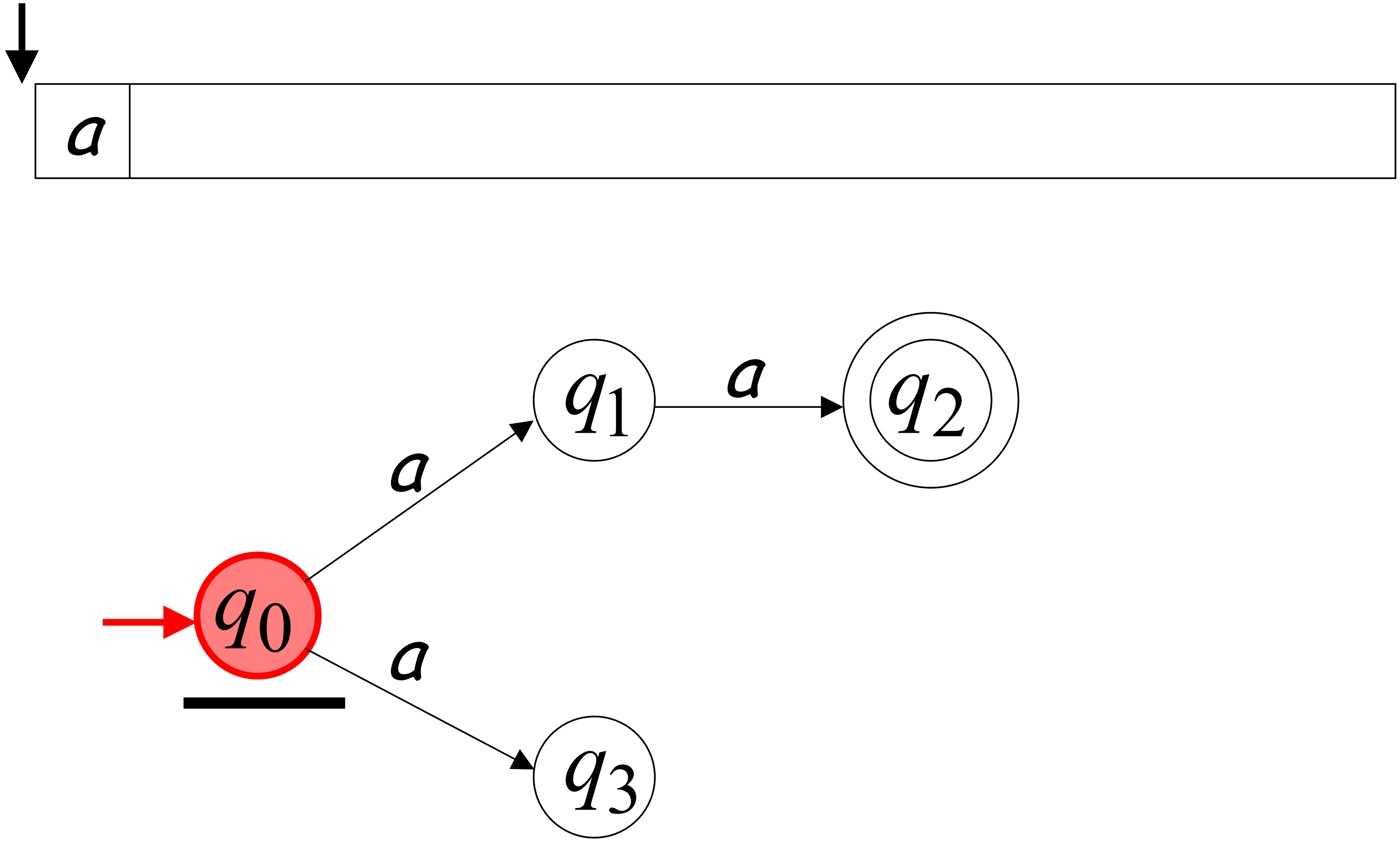
First Choice



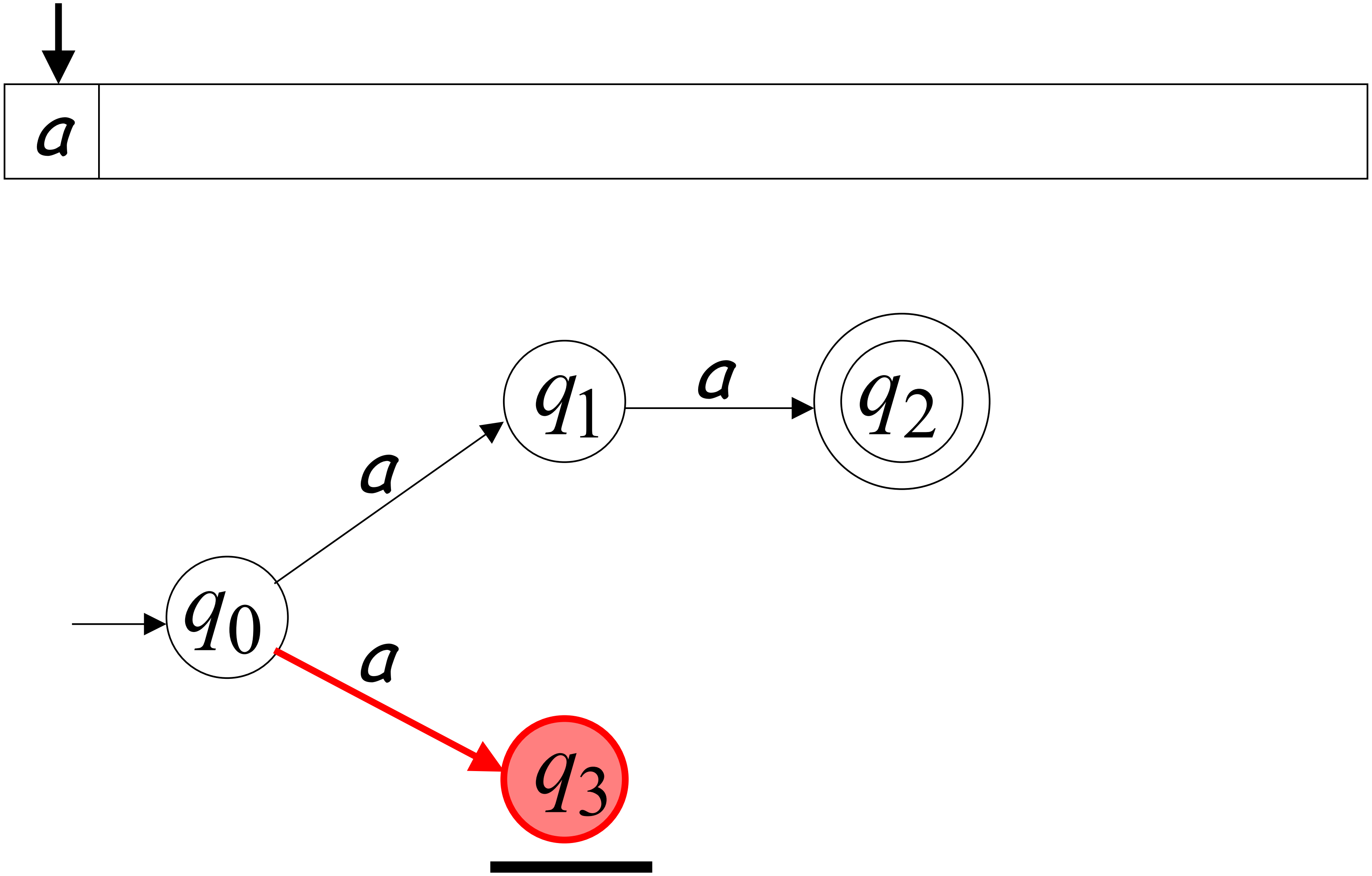
First Choice



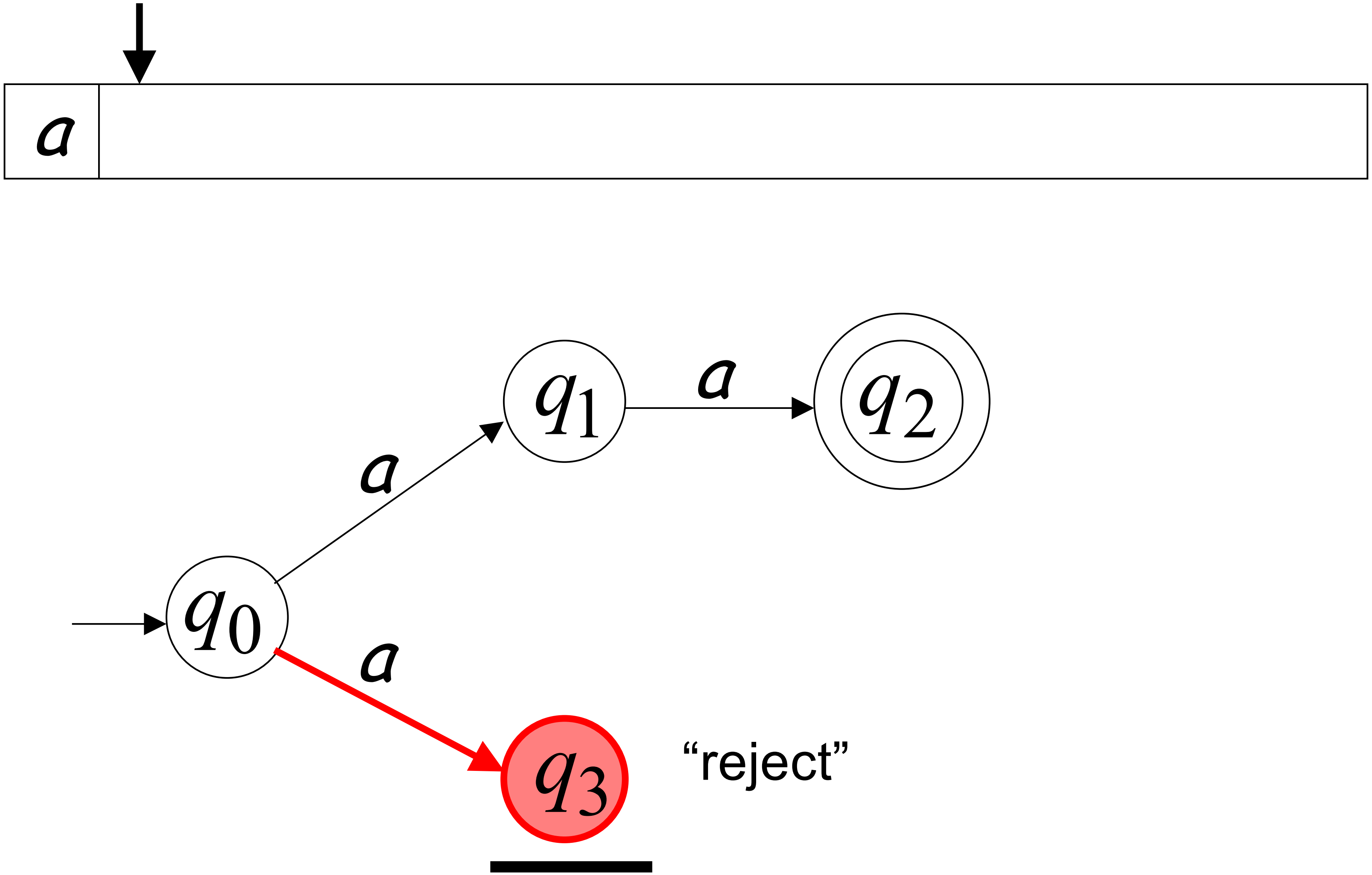
Second Choice



Second Choice



Second Choice



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

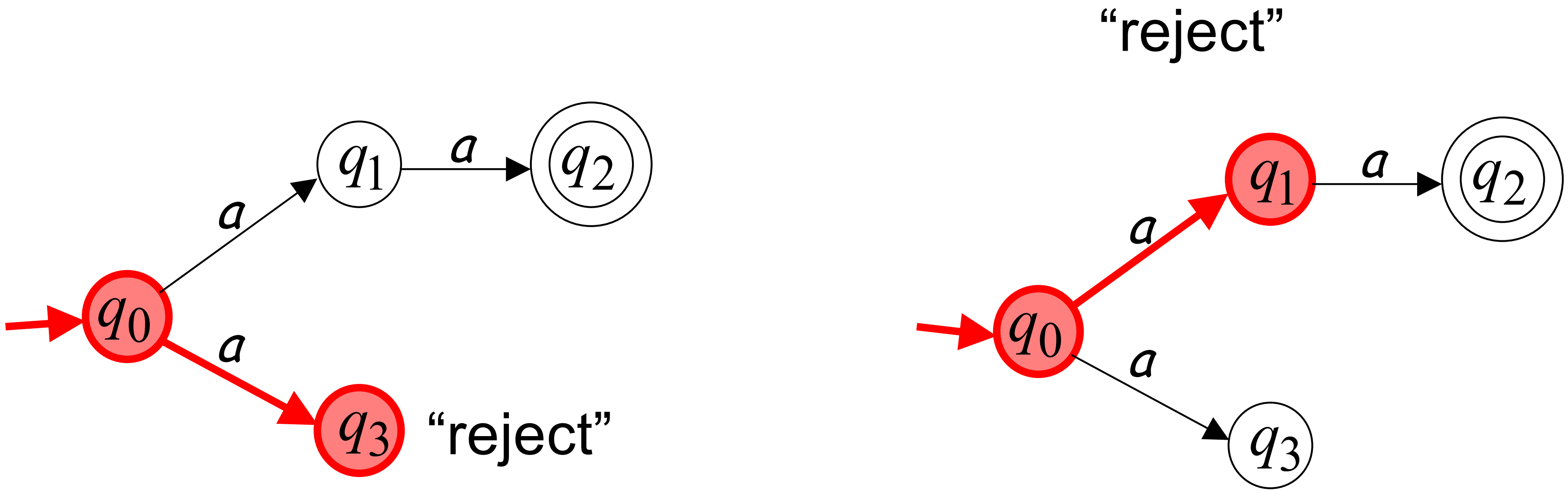
All the input is consumed and the automaton is in a non-final state

OR

The input cannot be consumed

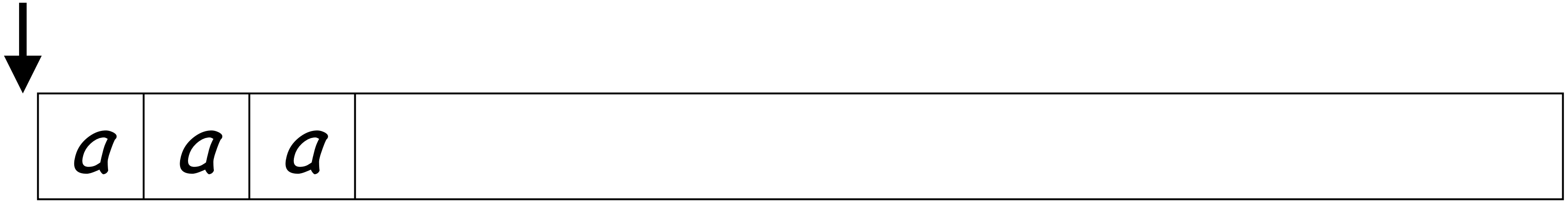
Example

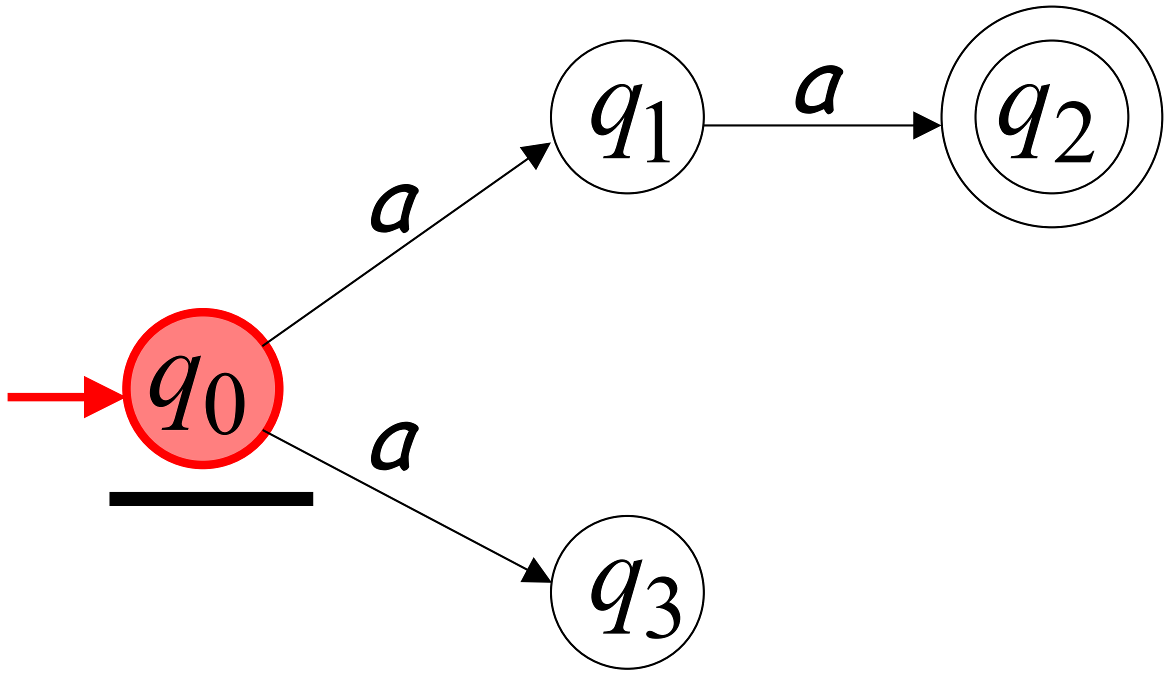
a is rejected by the NFA:



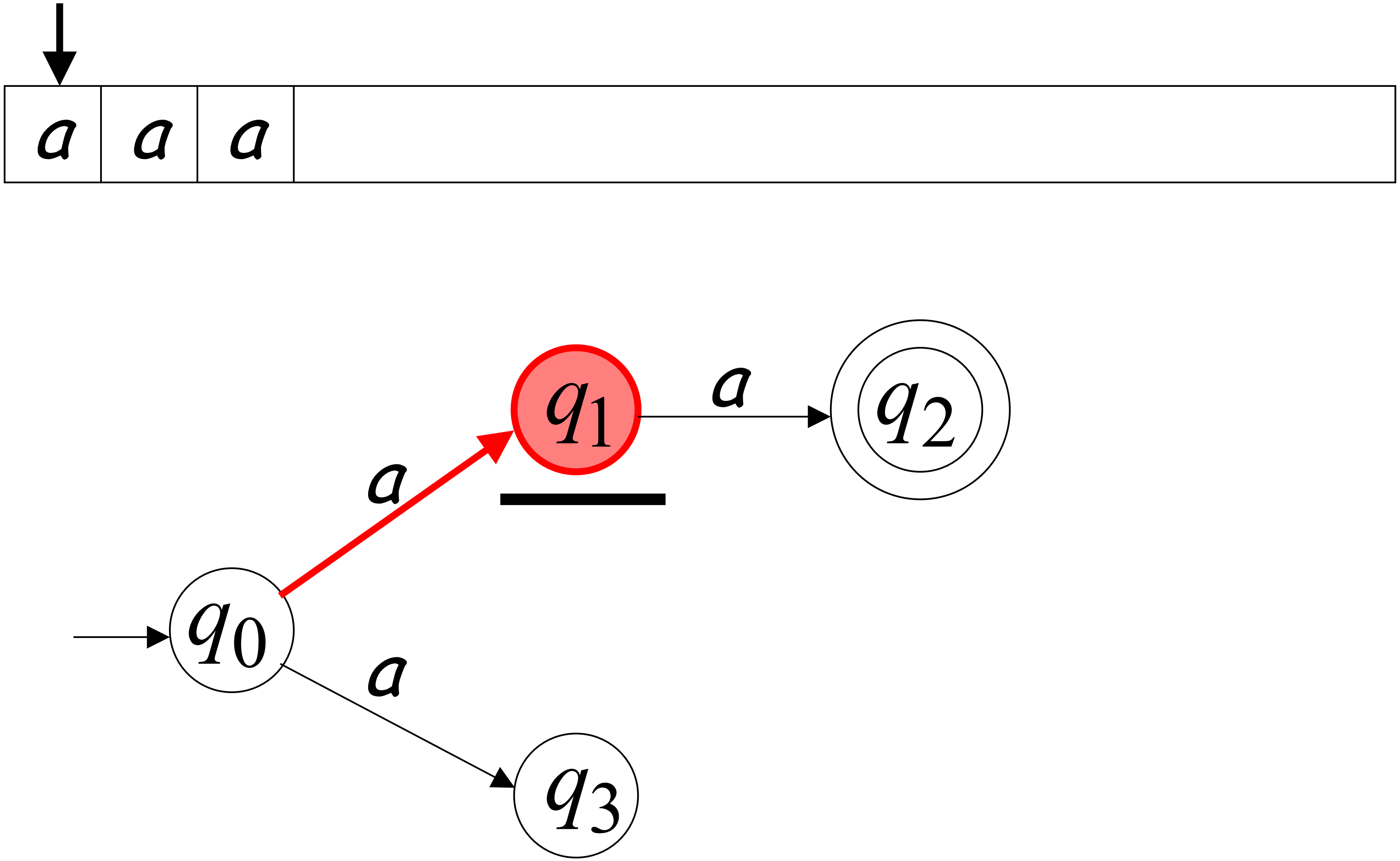
All possible computations lead to rejection

Rejection example

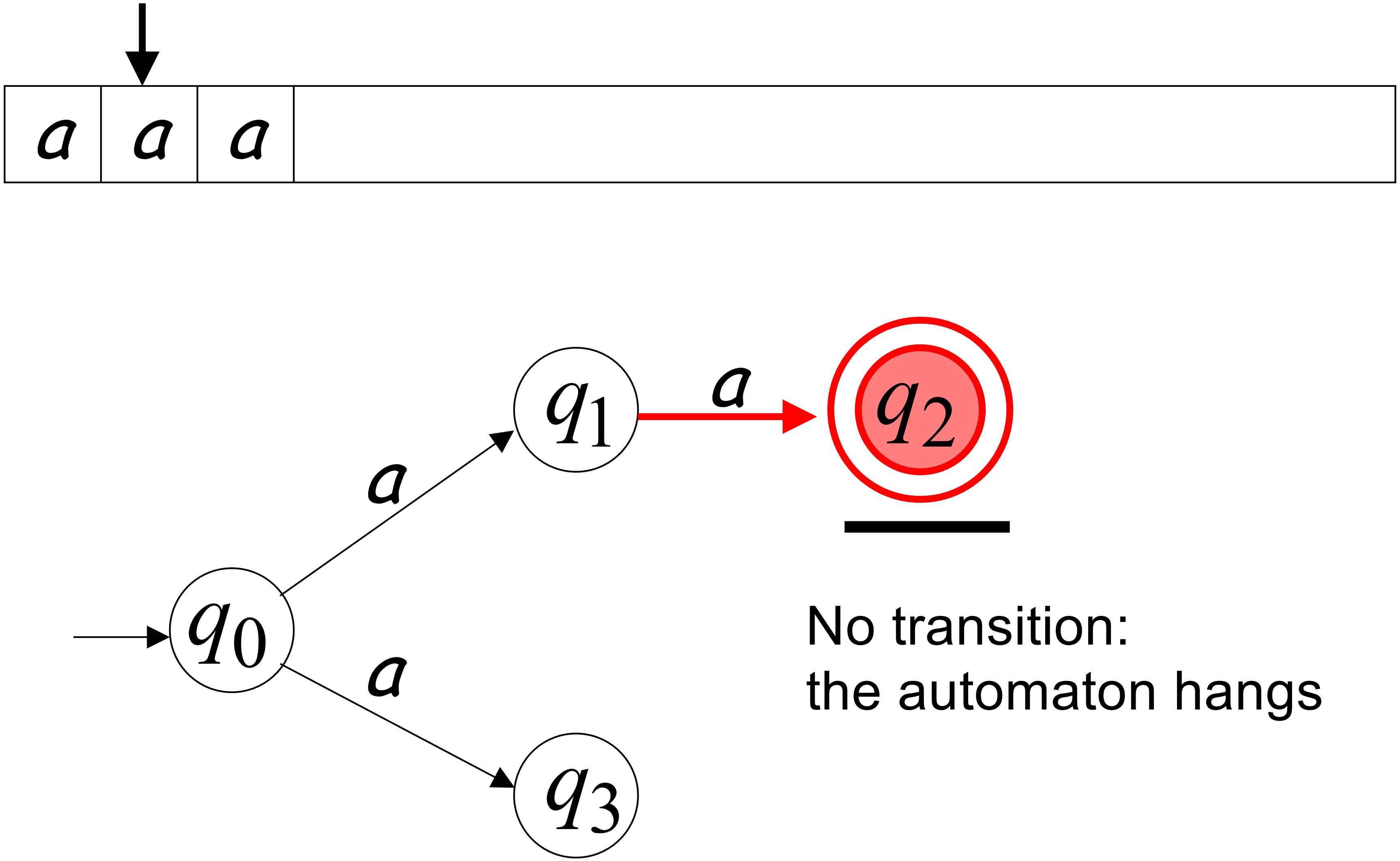




First Choice

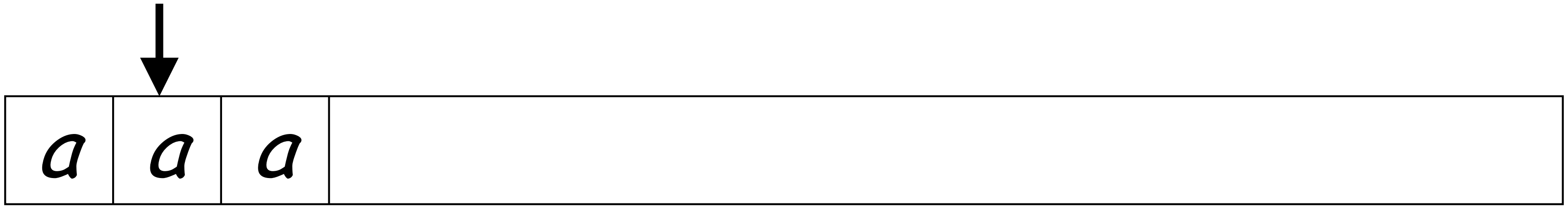


First Choice

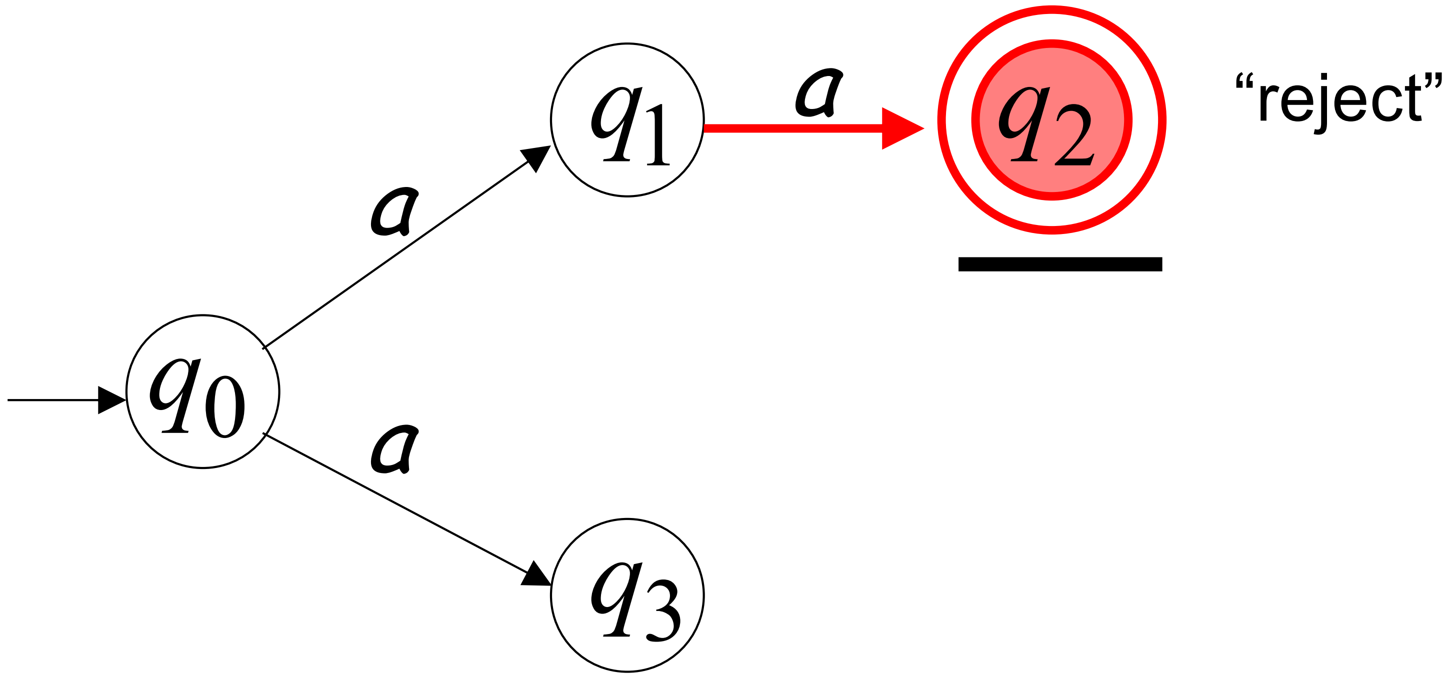


No transition:
the automaton hangs

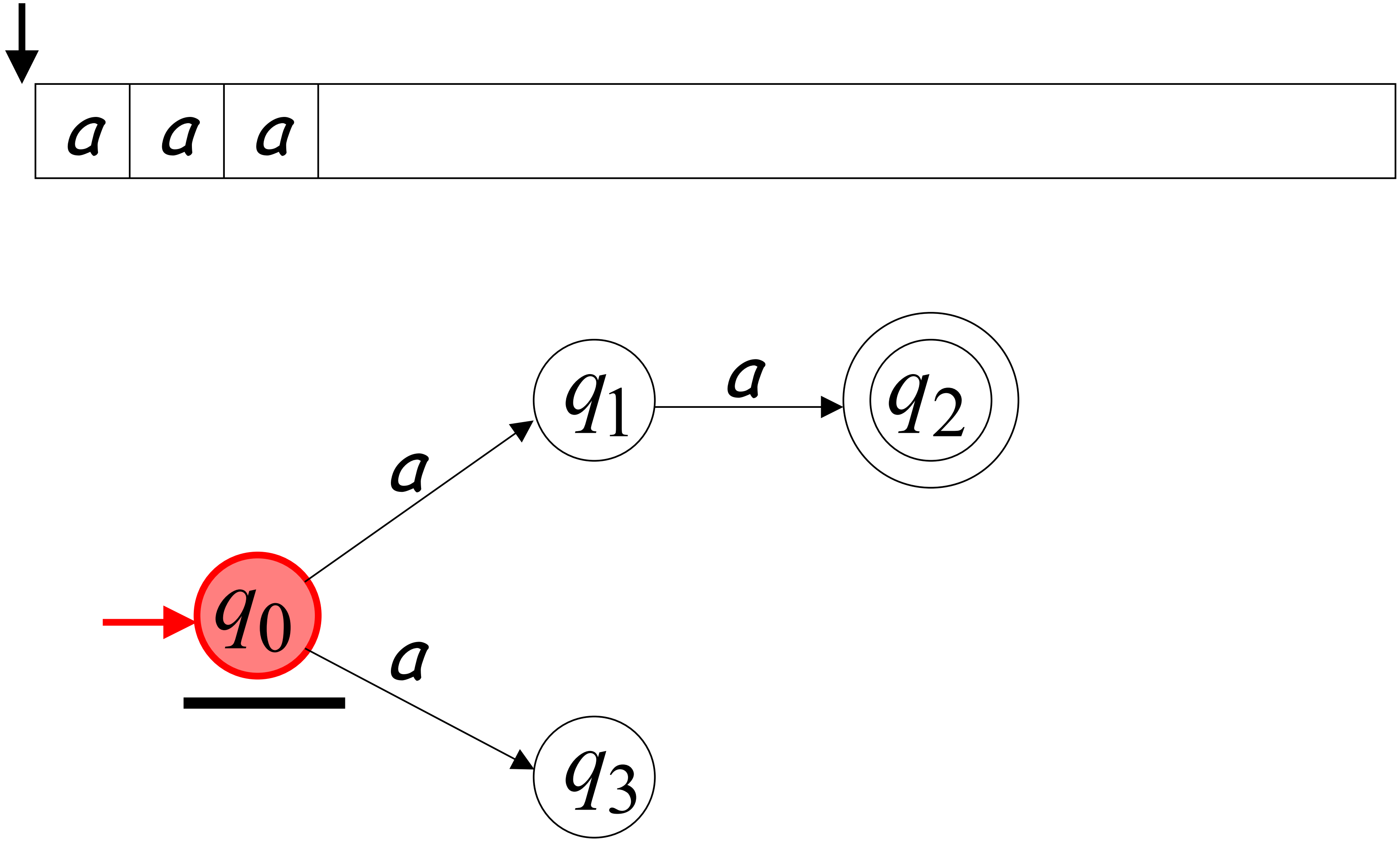
First Choice



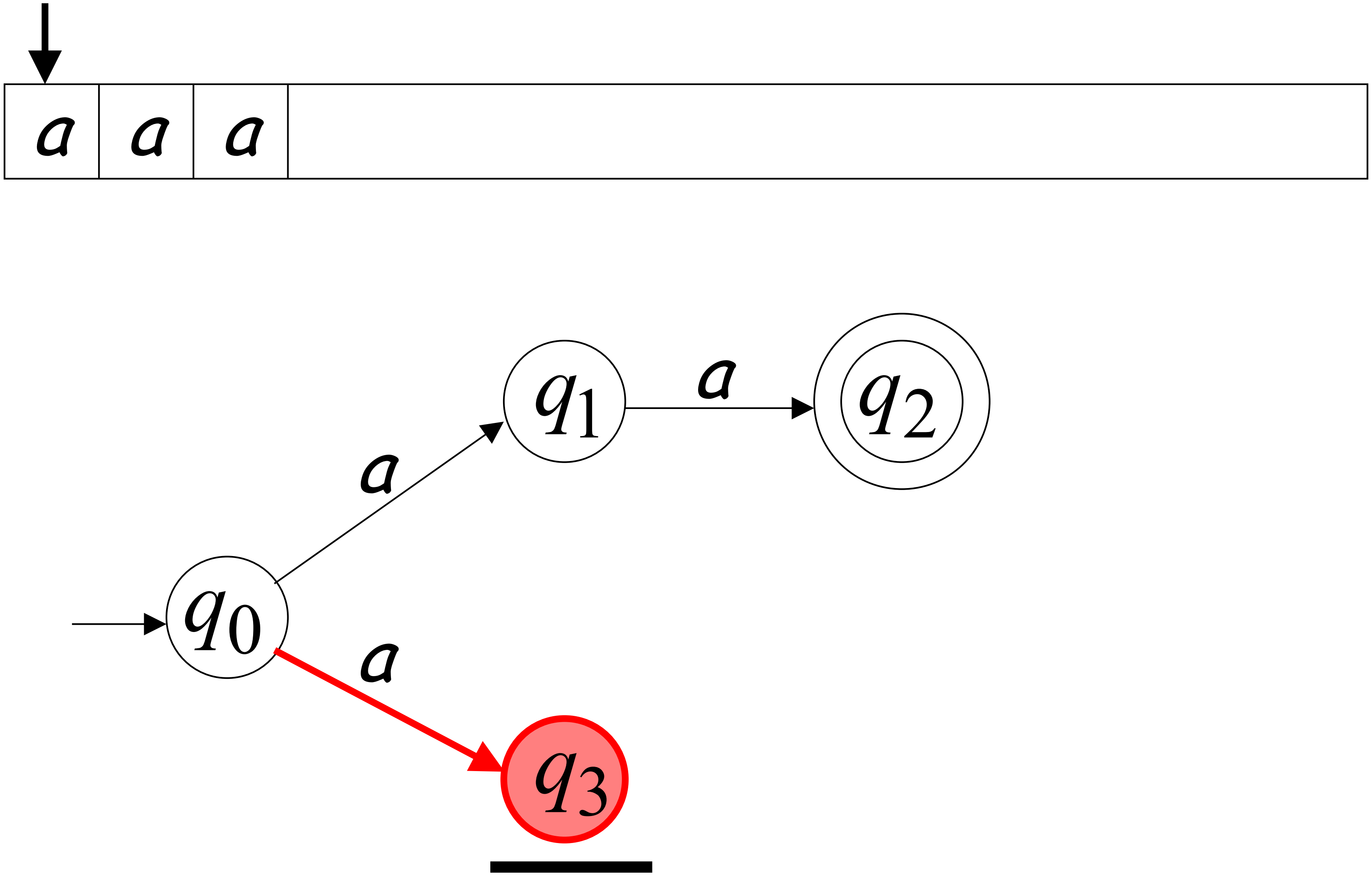
Input cannot be consumed



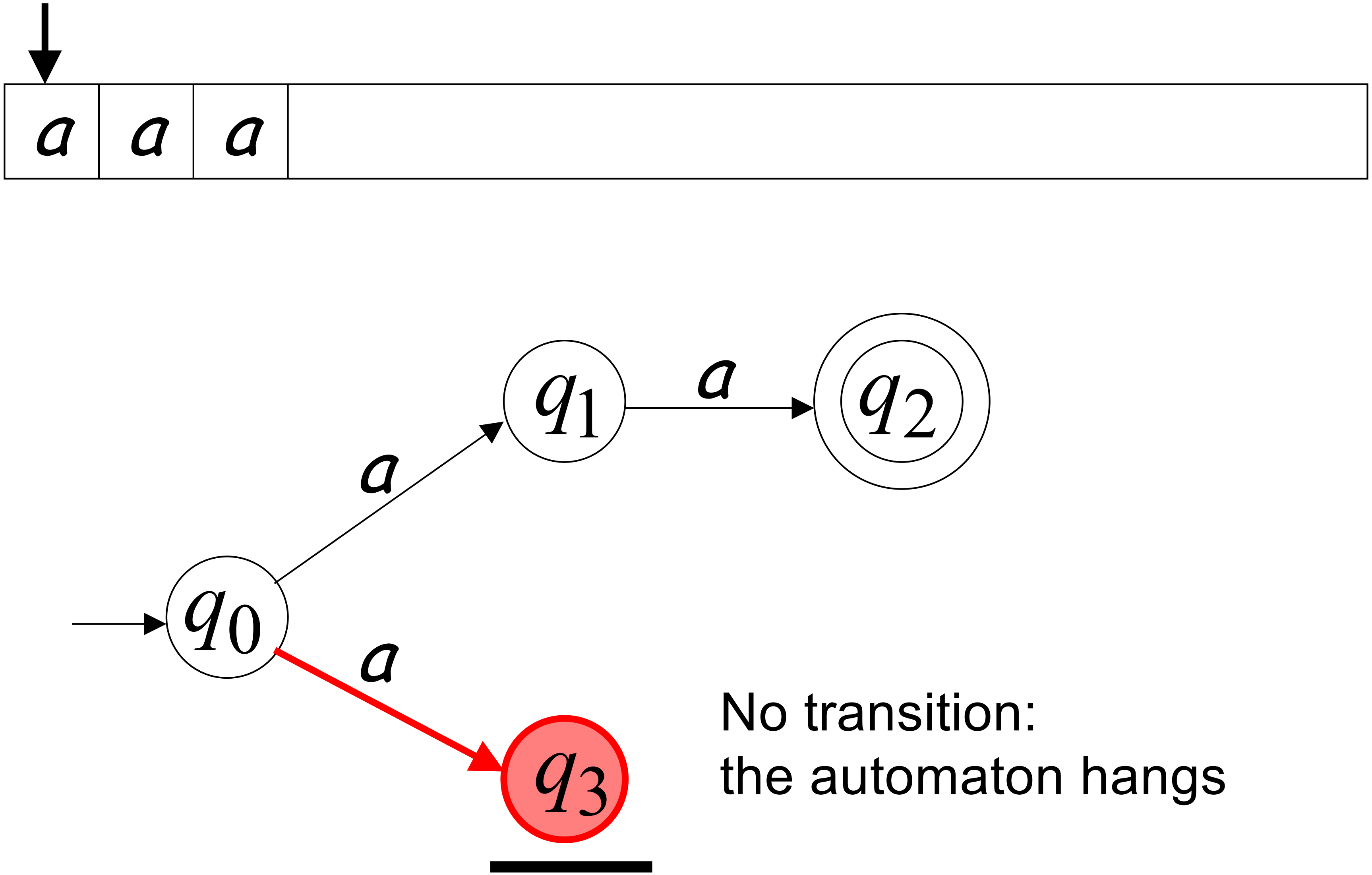
Second Choice



Second Choice

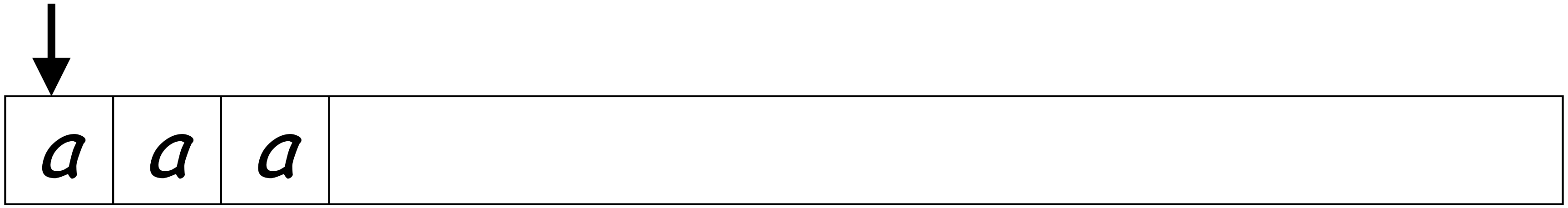


Second Choice

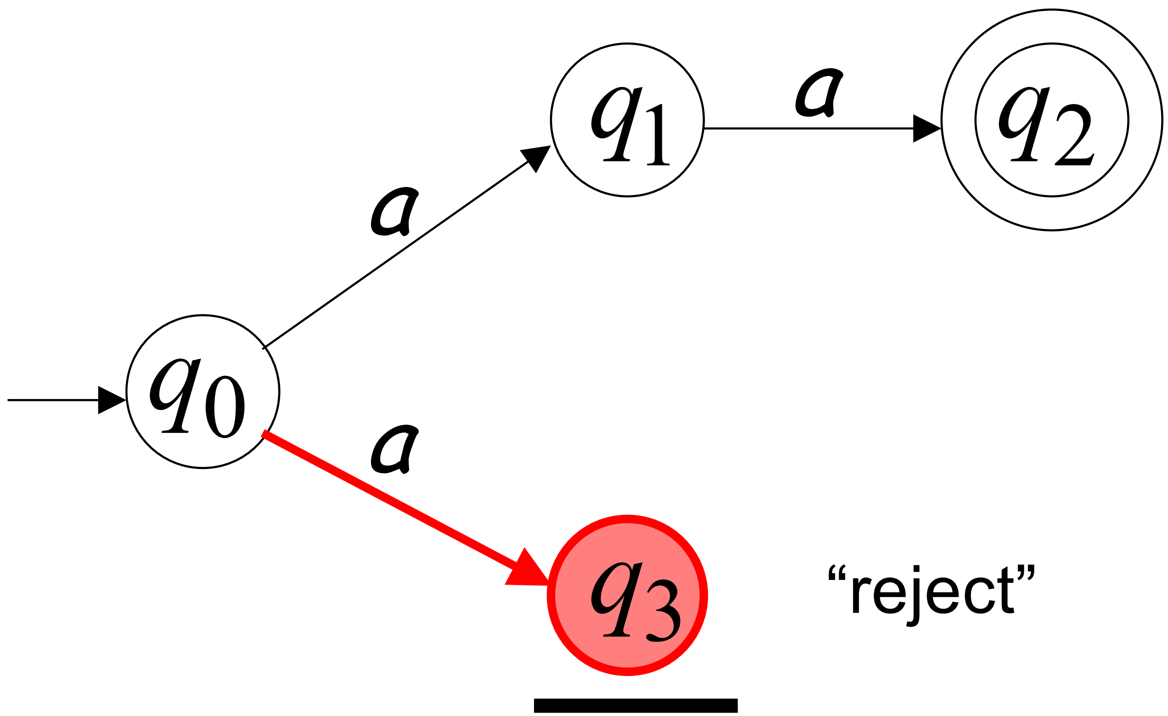


No transition:
the automaton hangs

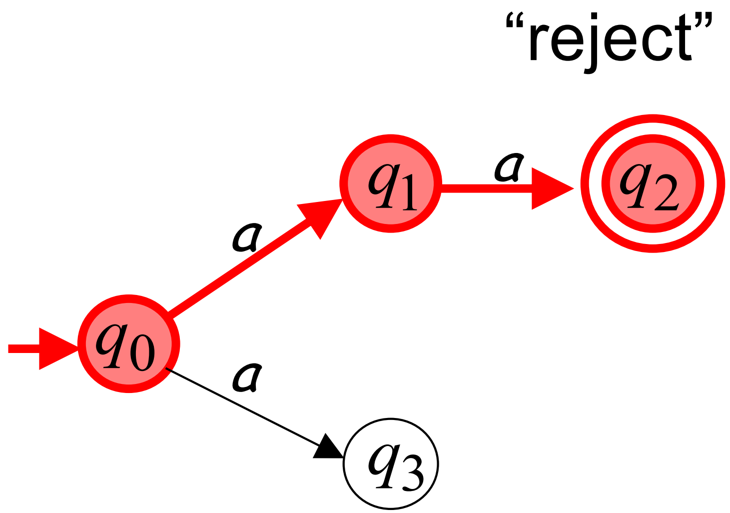
Second Choice

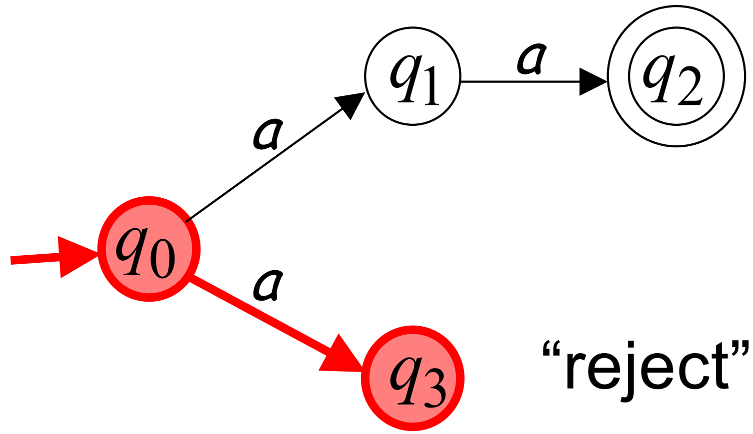


Input cannot be consumed



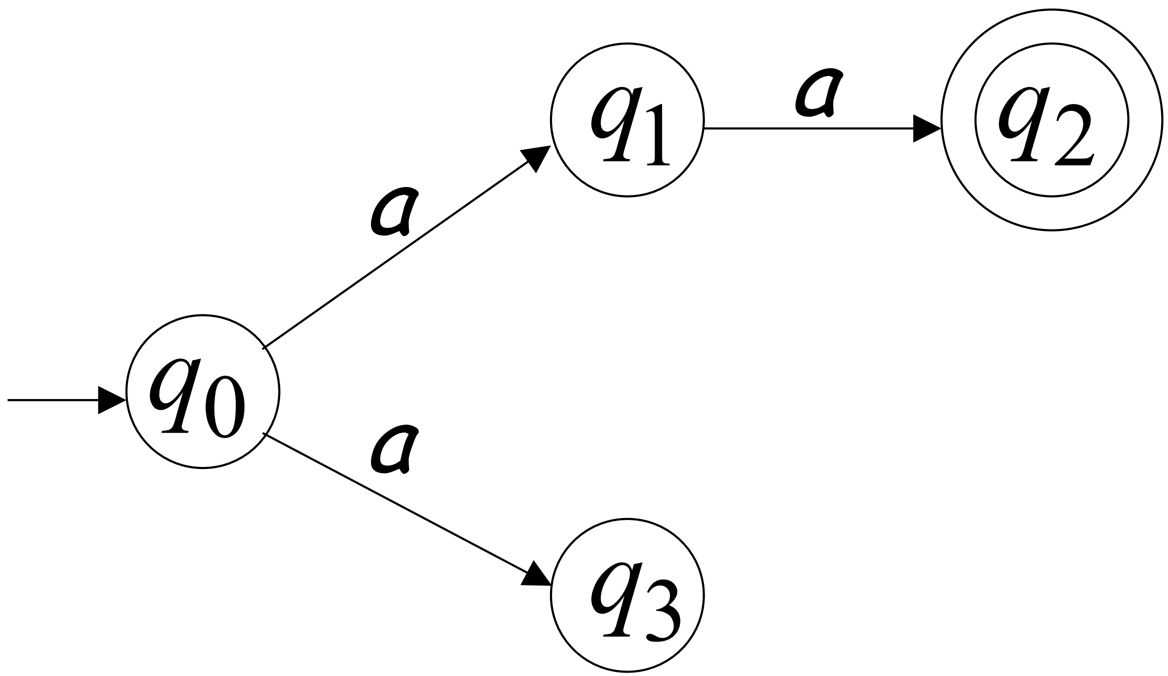
aaa is rejected by the NFA:



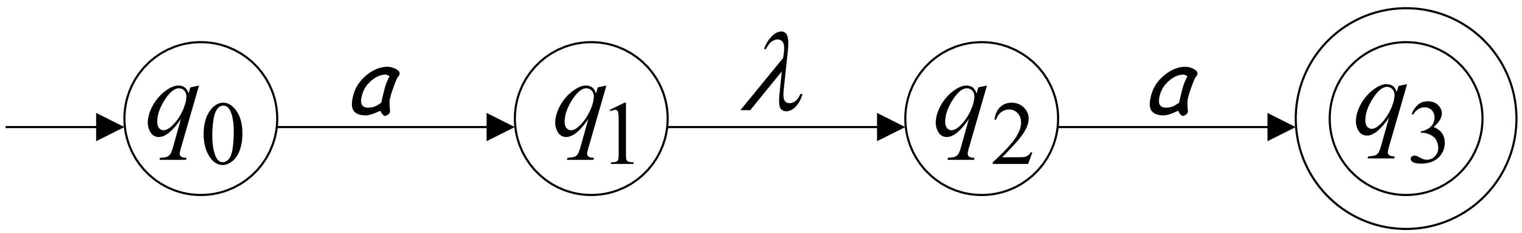


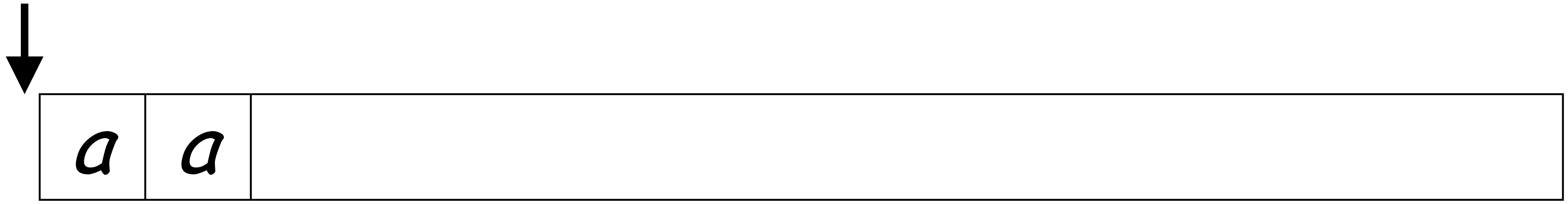
All possible computations lead to rejection

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

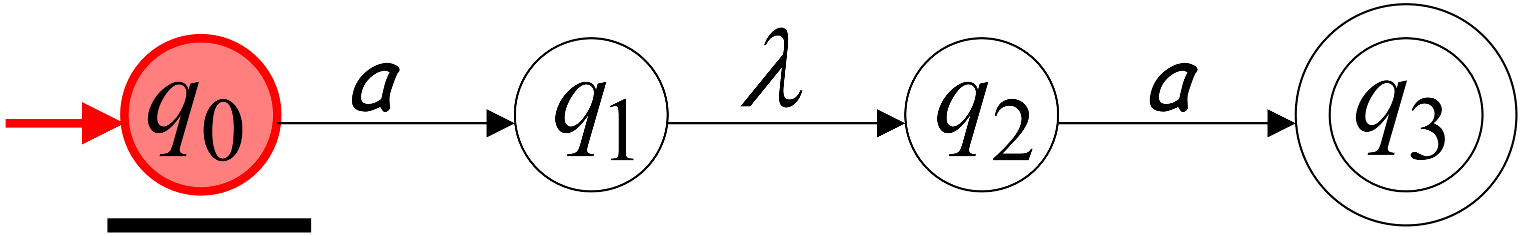


Lambda(λ) Transitions

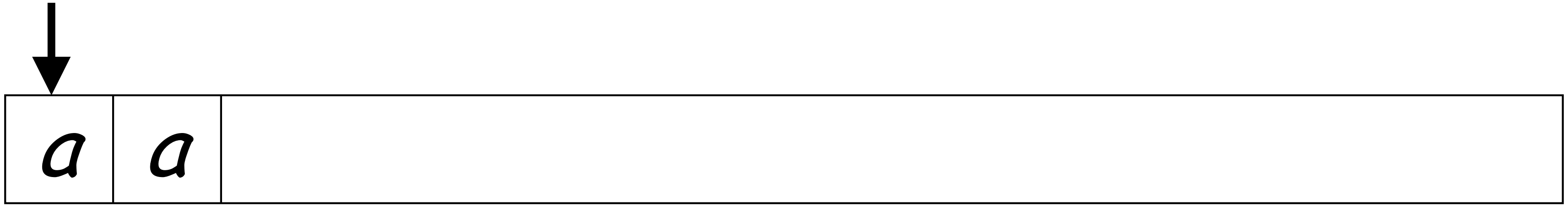


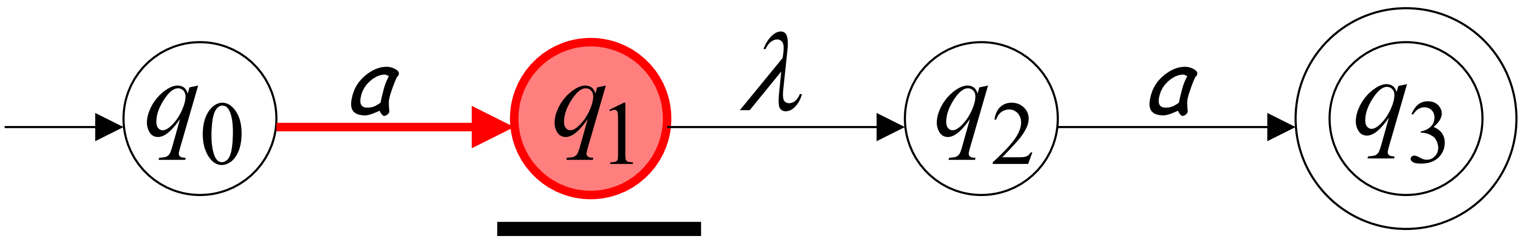


a
a

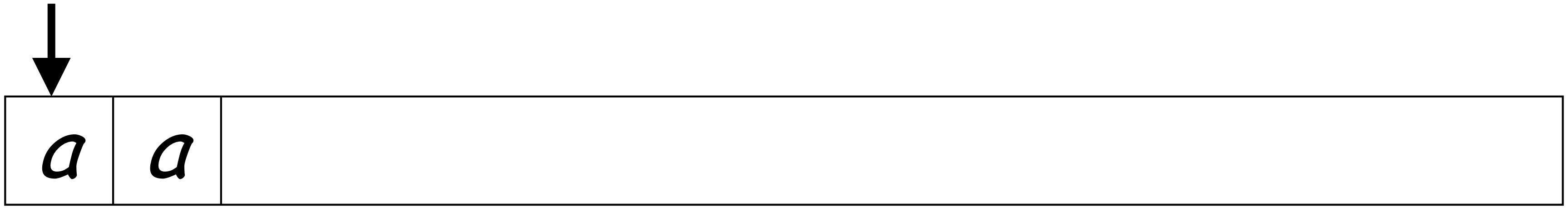


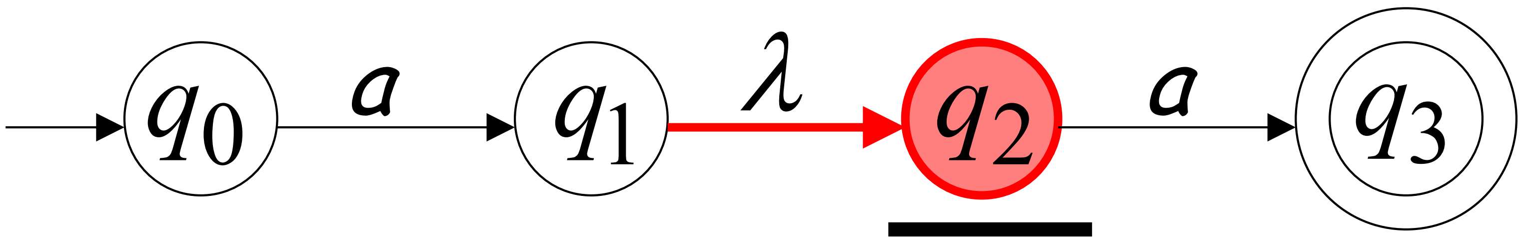
q_0
a
q_1
λ
q_2
a
q_3

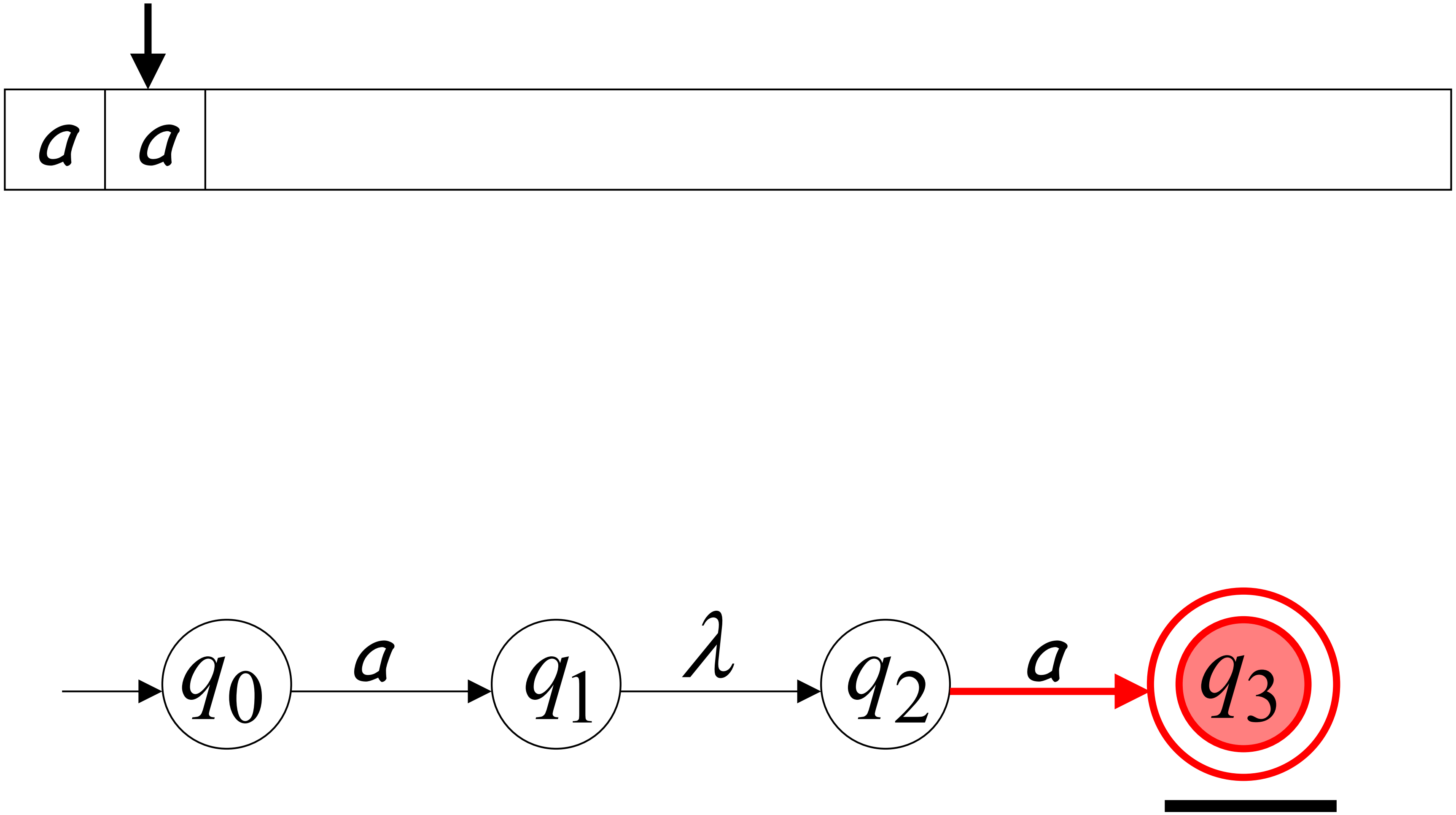

a
a


q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

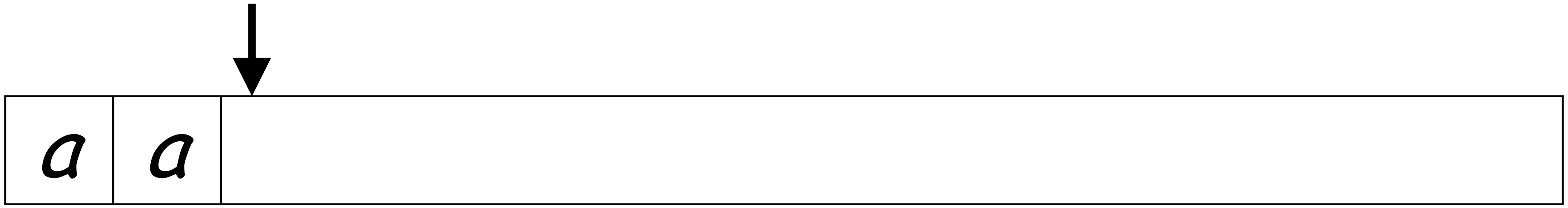


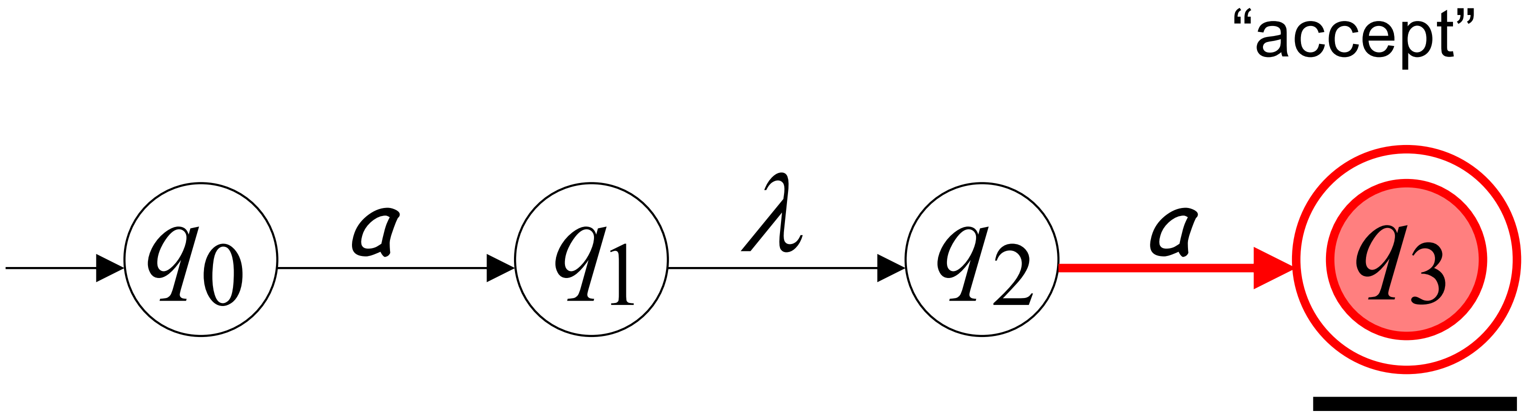




a a
q0
a
q1
λ
q2
a
q3

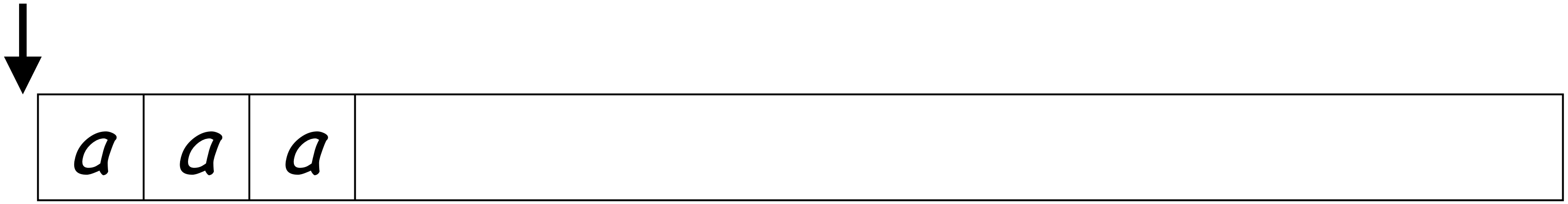
all input is consumed

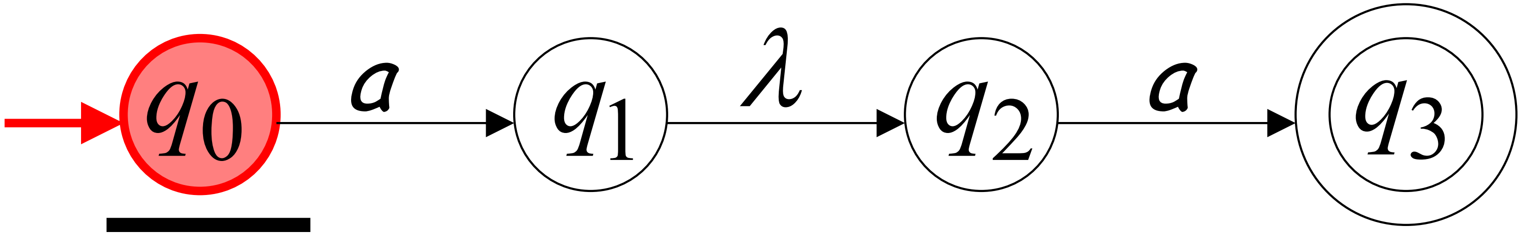


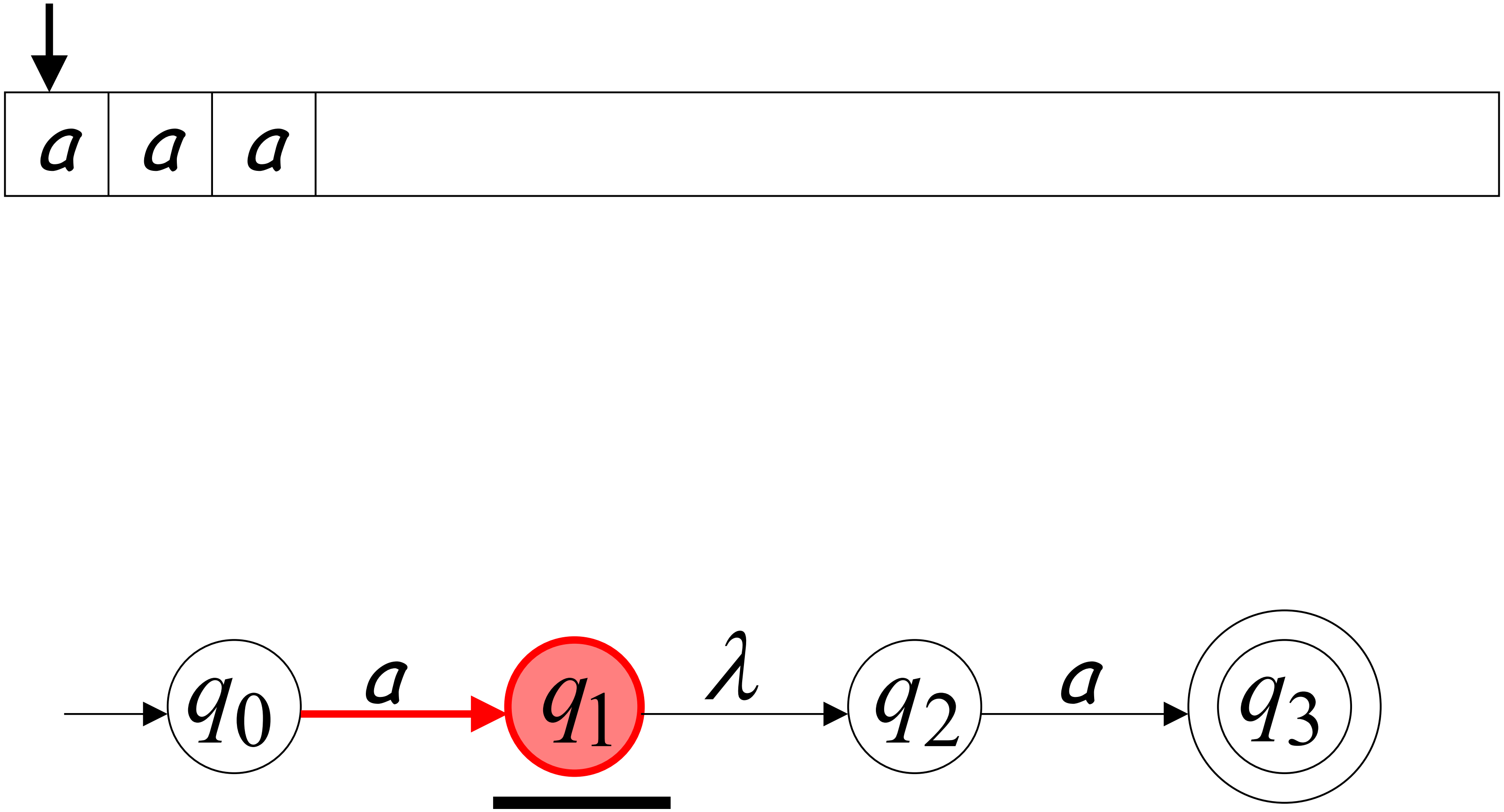


String aa is accepted

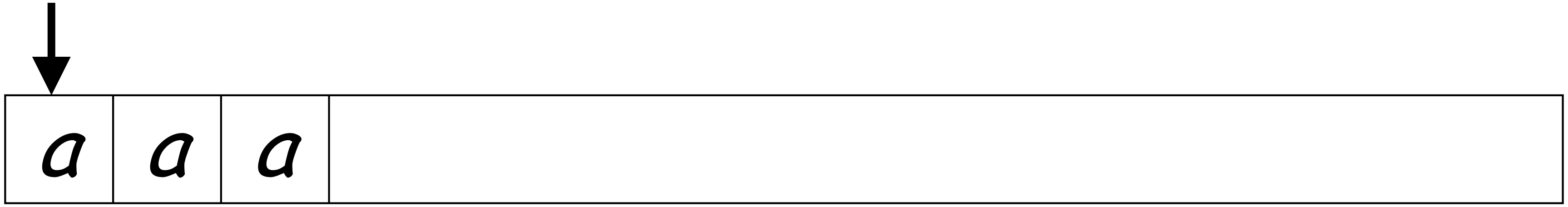
Rejection Example

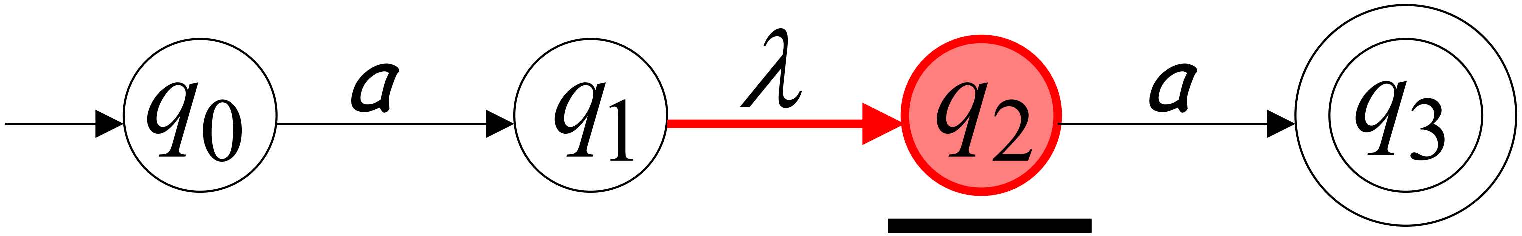


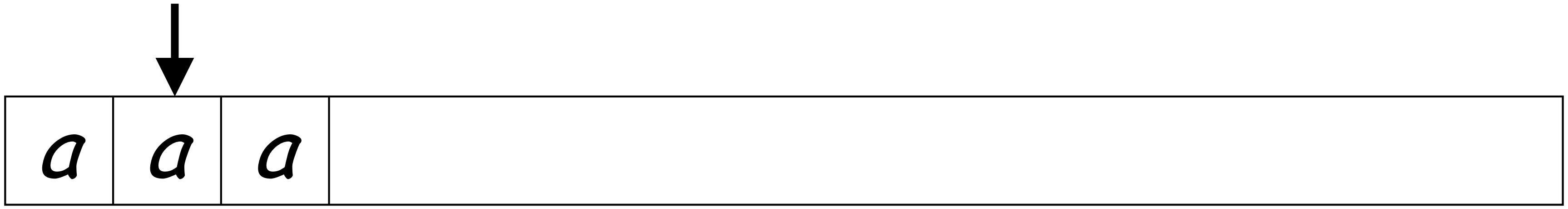


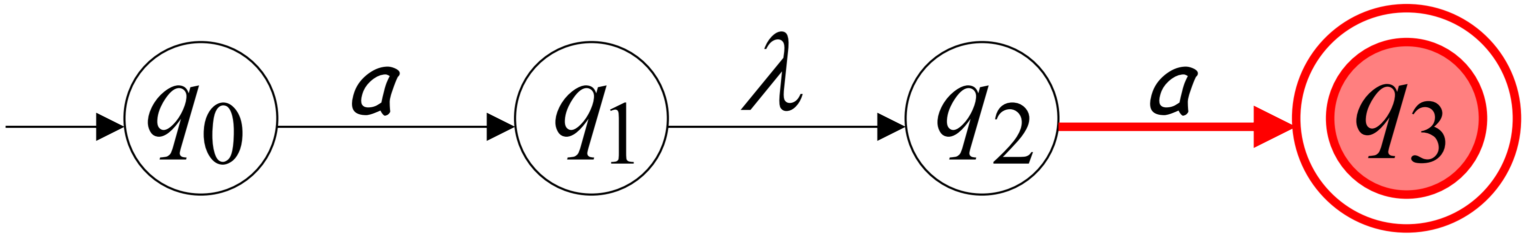

a
a
a
q0
a
q1
λ
q2
a
q3

(read head doesn't move)



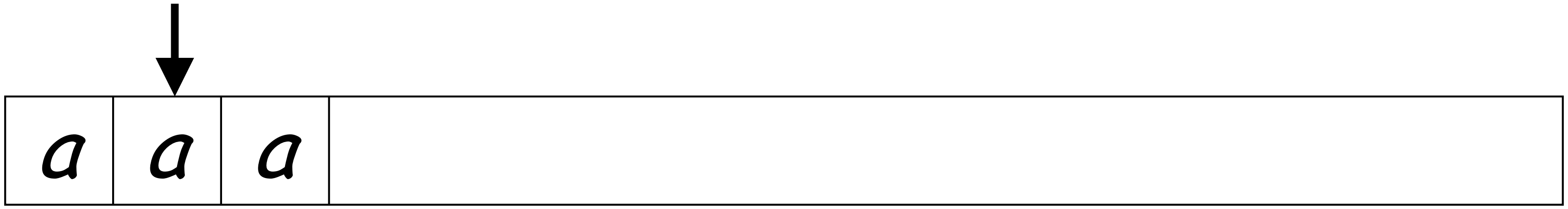


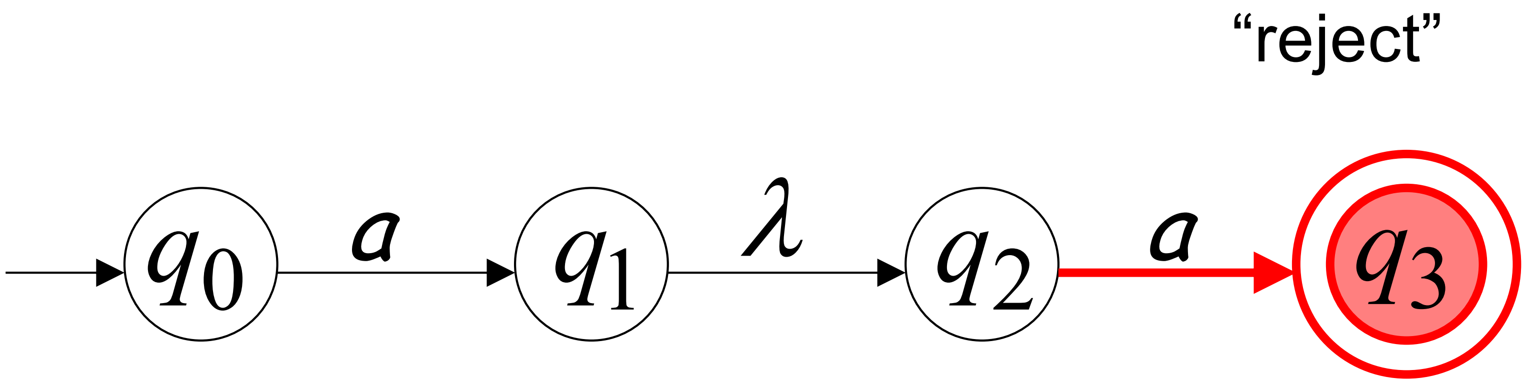




No transition:
the automaton hangs

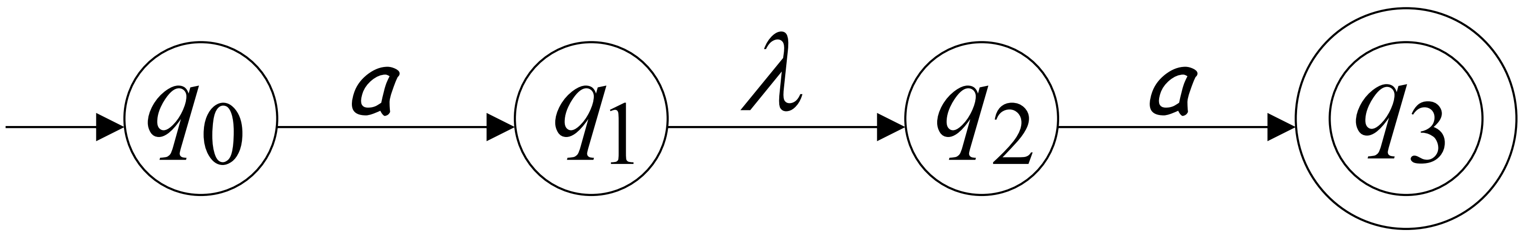
Input cannot be consumed



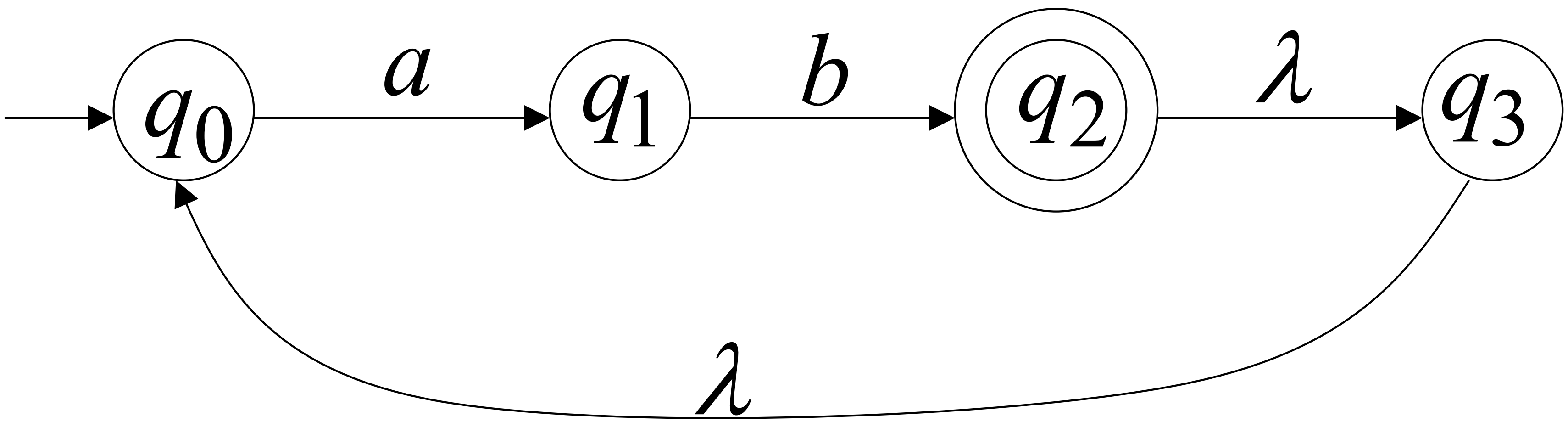


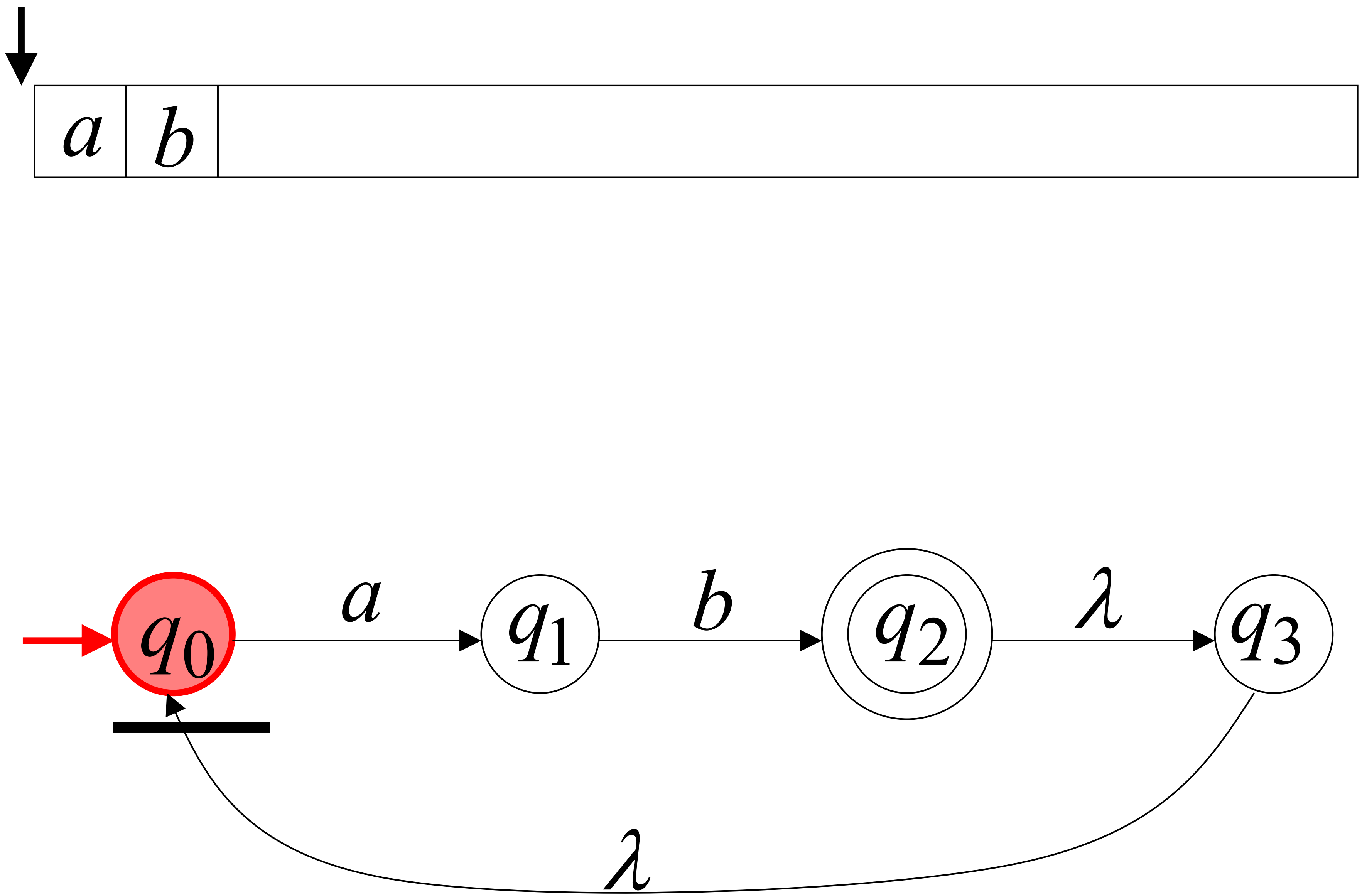
String aaa is rejected

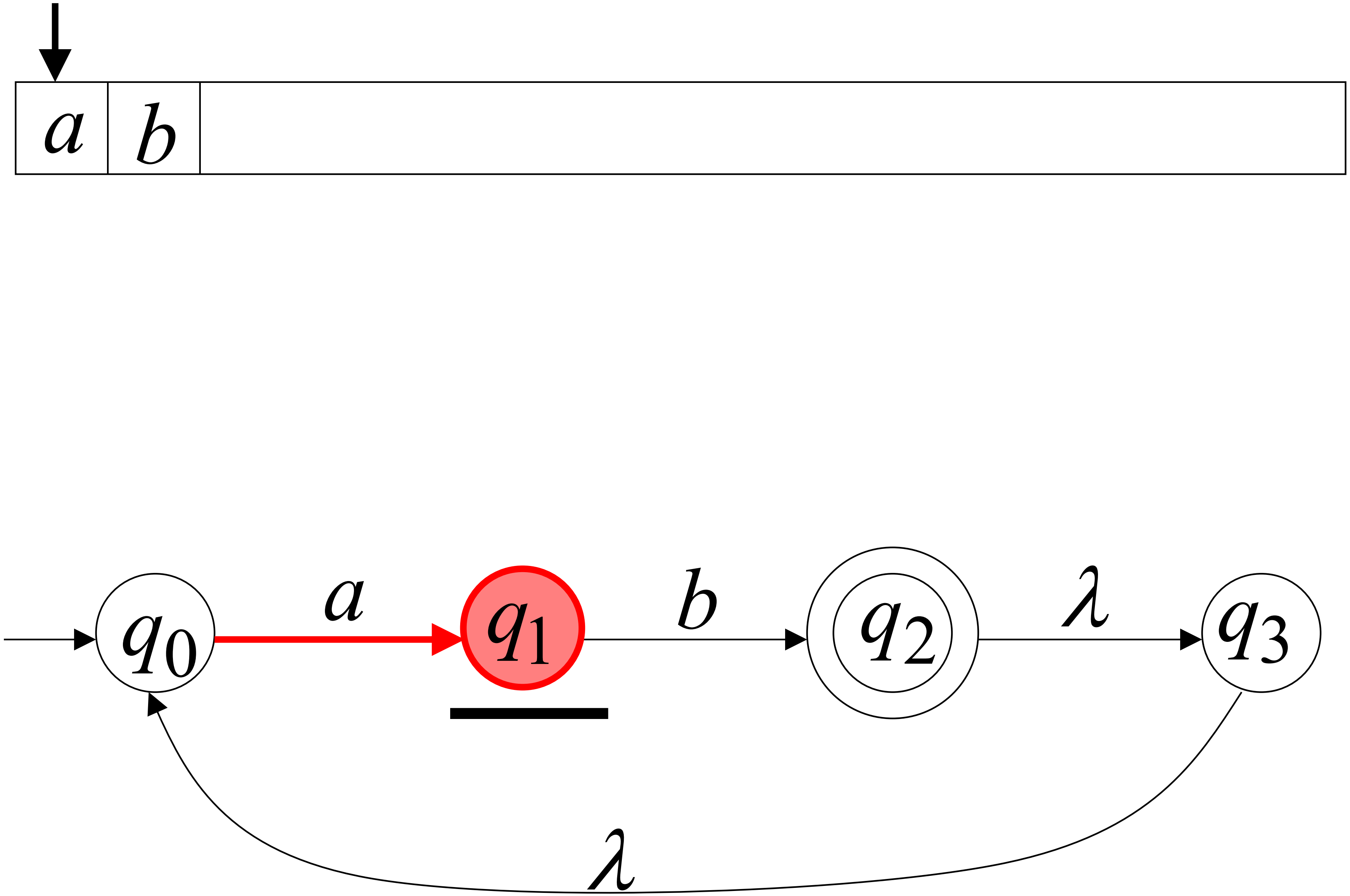
Language accepted: $L = \{aa\}$

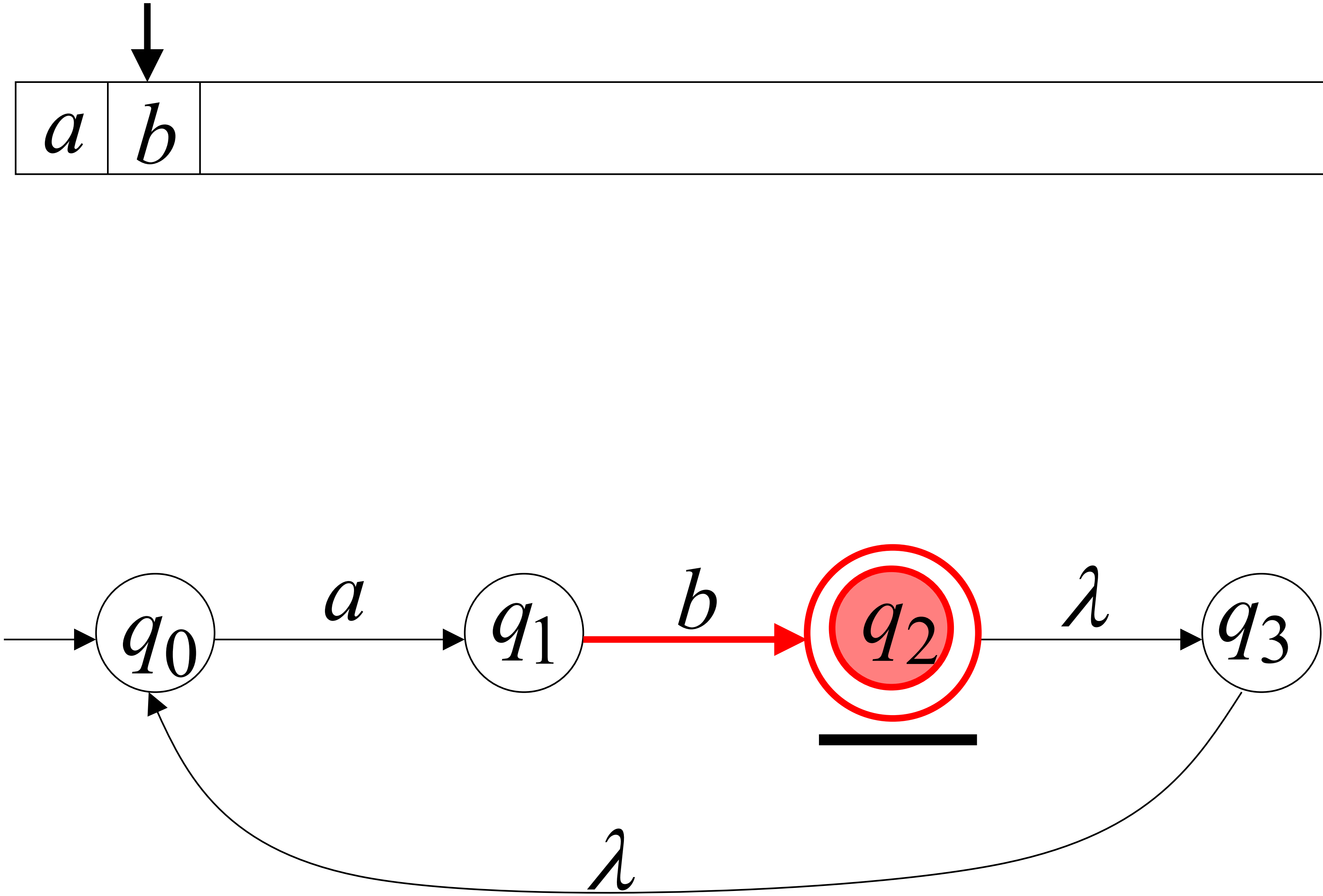


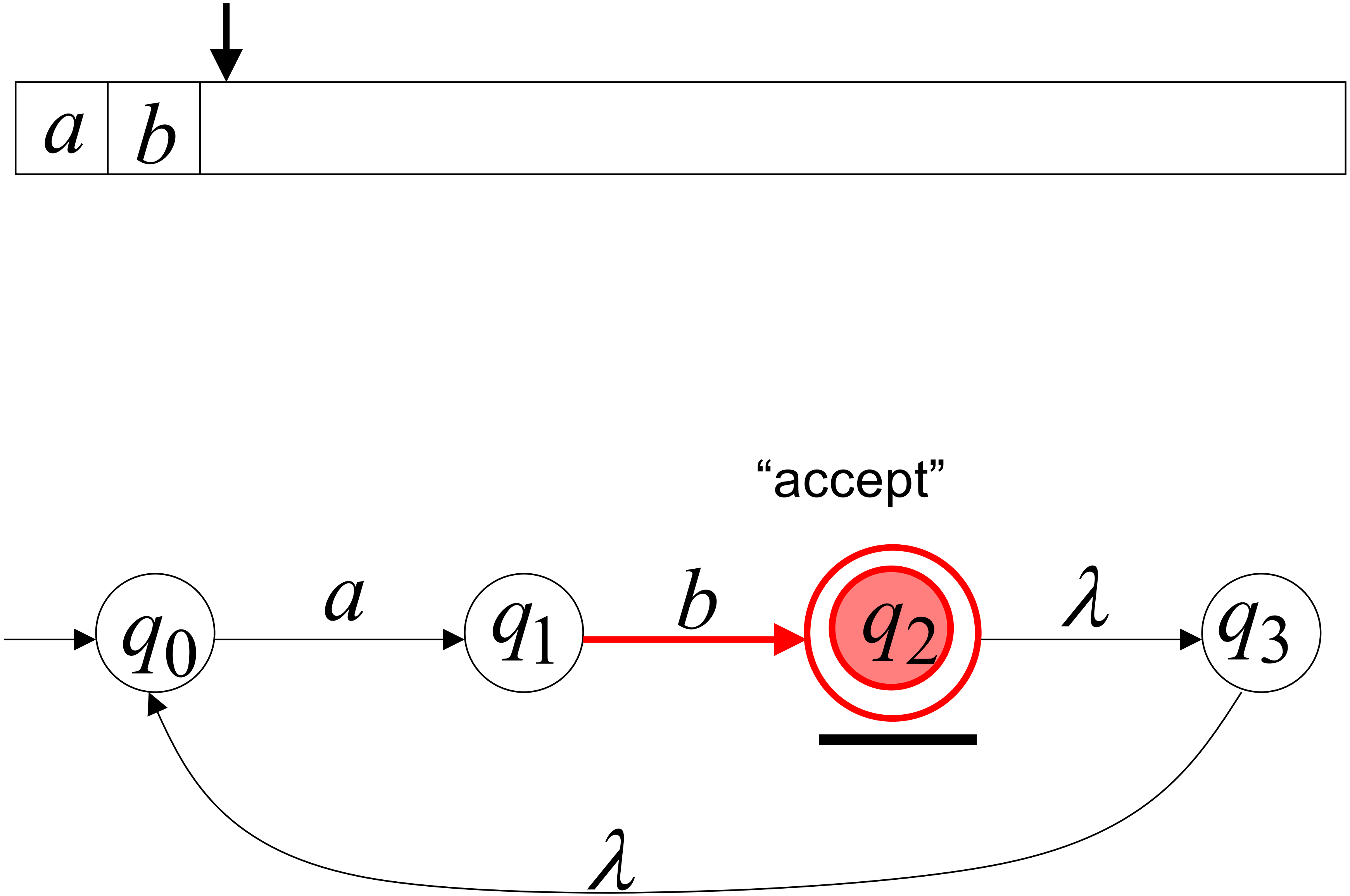
Another NFA Example



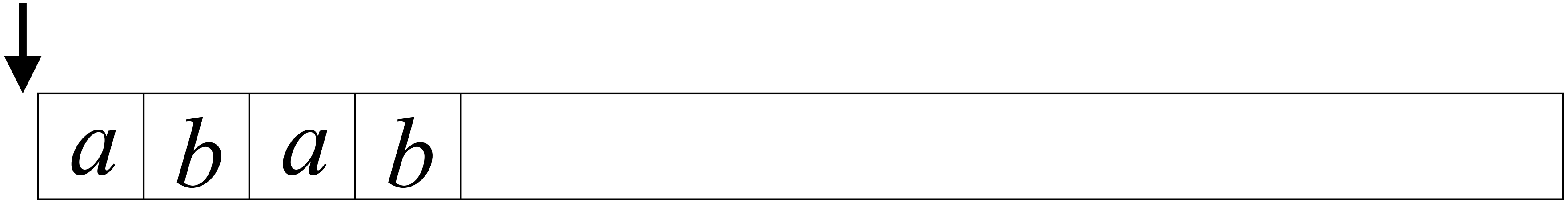

a
b
a
b
λ
λ
q0
q1
q2
q3

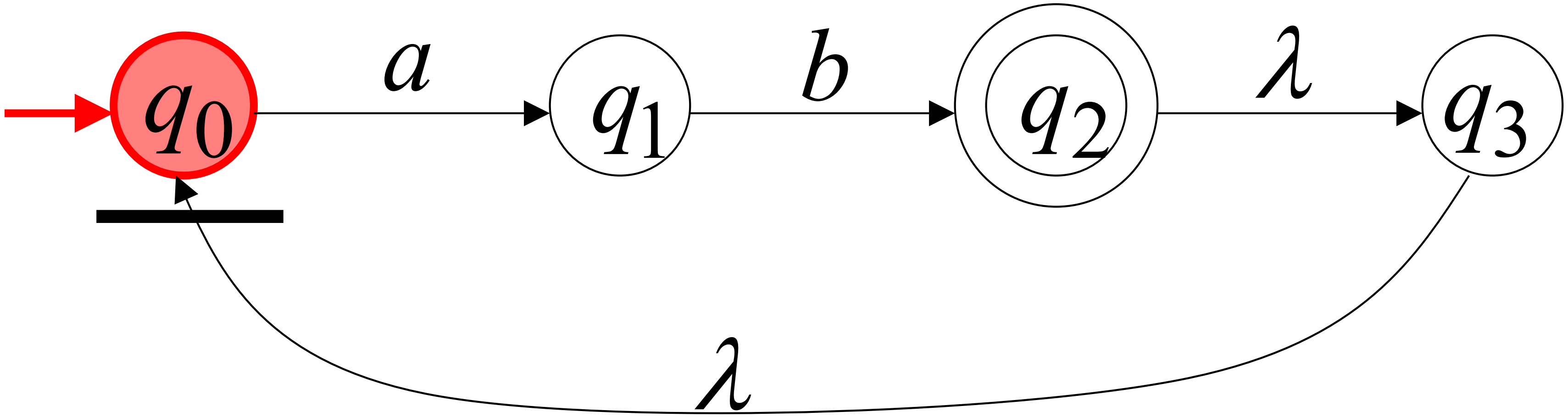

a
b
a
b
λ
λ
q_0
q_1
q_2
q_3

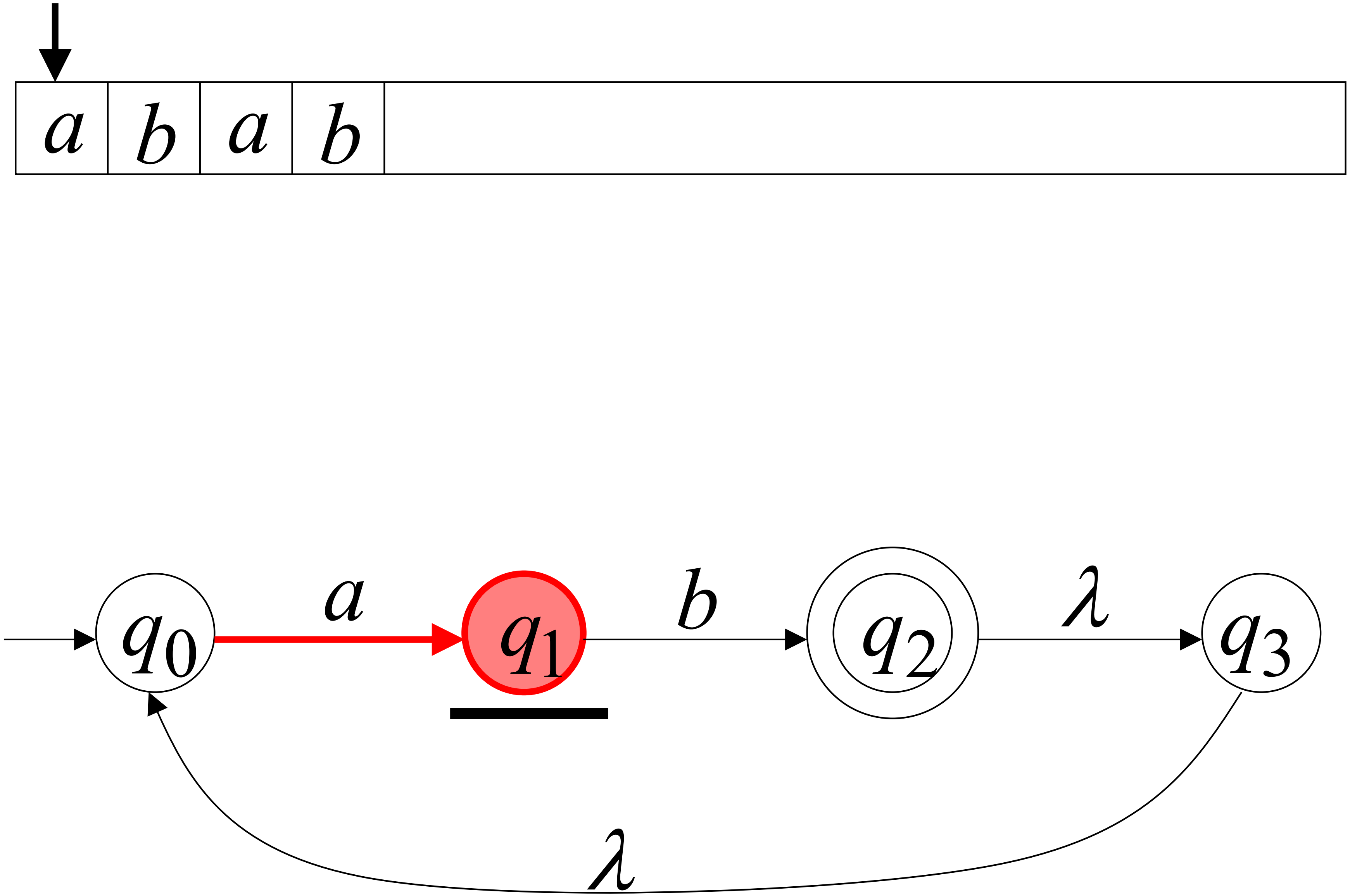

a
b
a
b
λ
λ
q_0
q_1
q_2
q_3

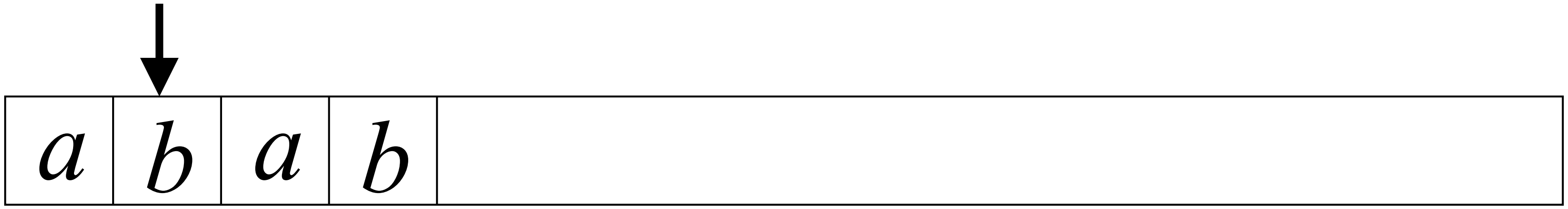

a
b
"accept"
q0
a
q1
b
q2
λ
q3
λ

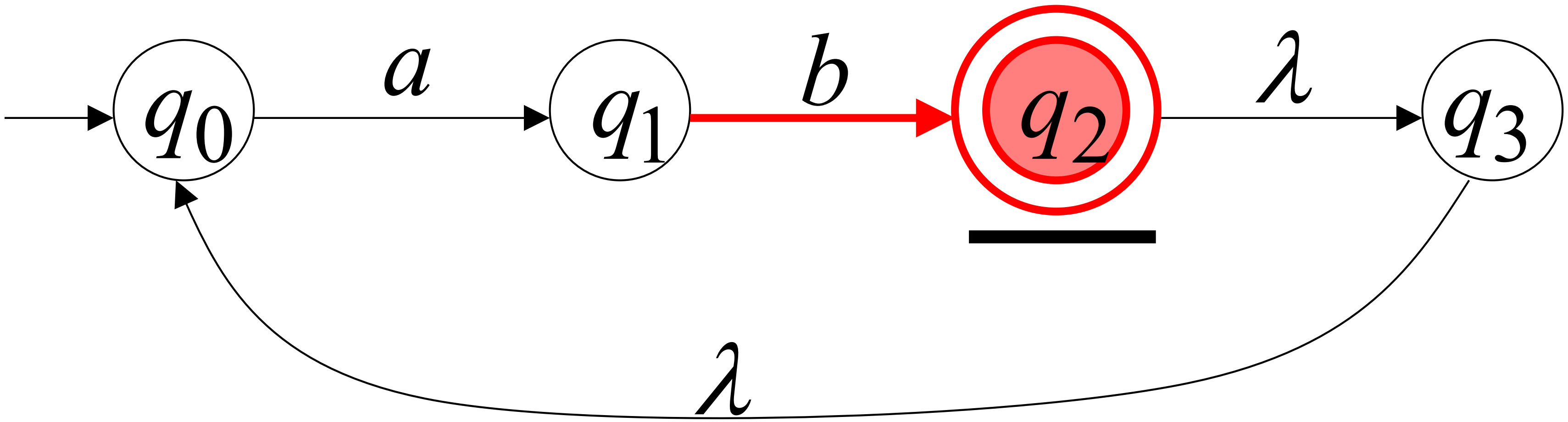
Another String

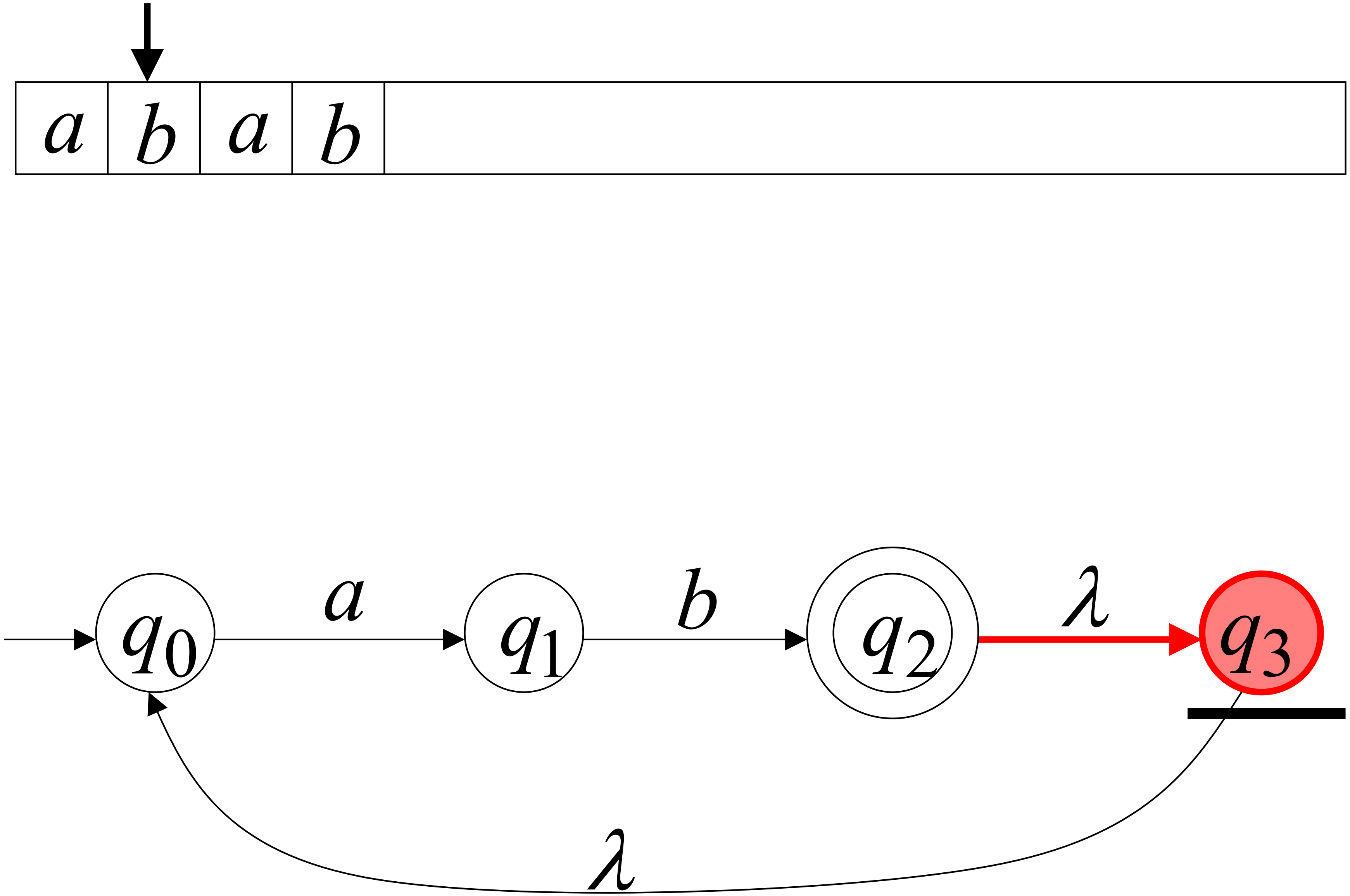


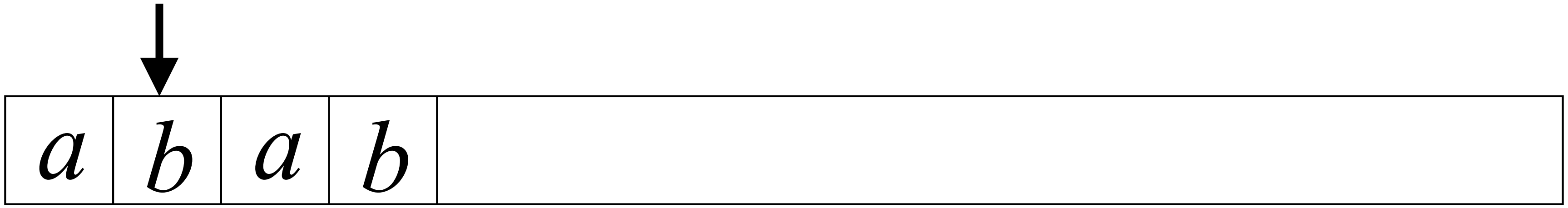


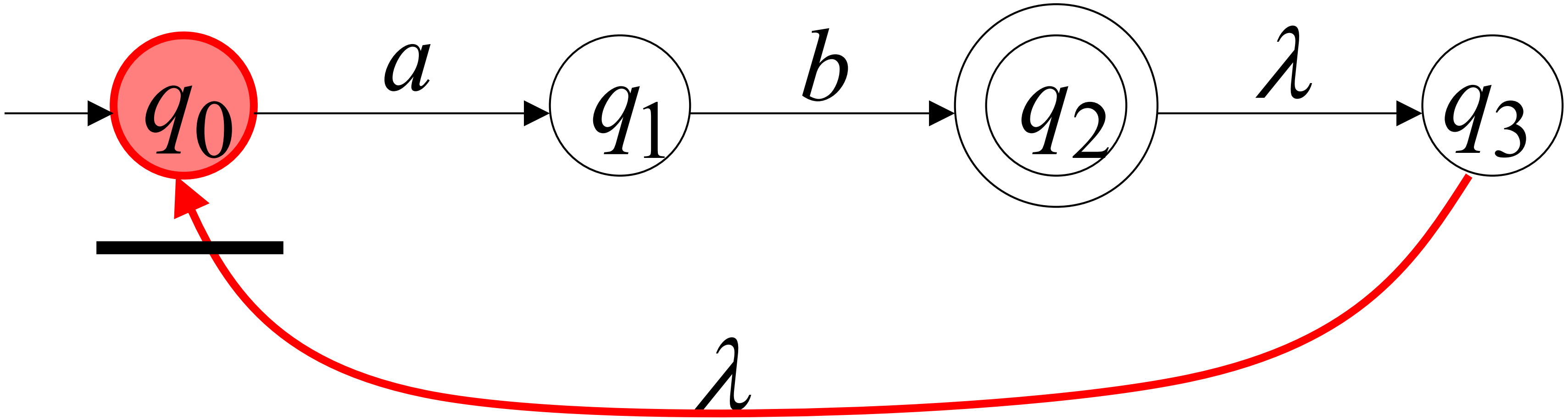

a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

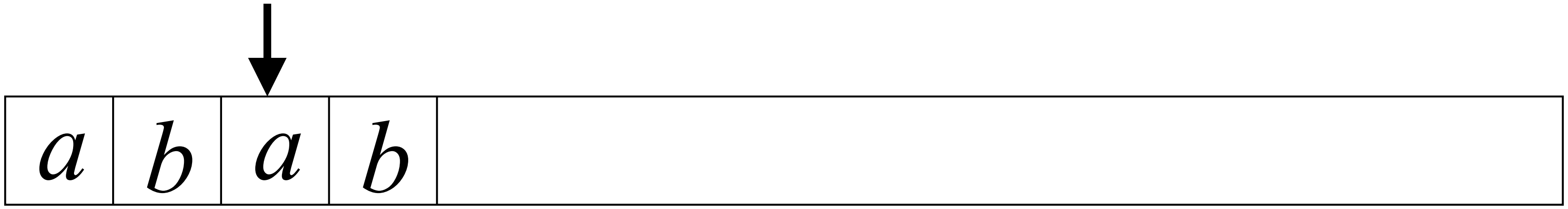

a
b
a
b

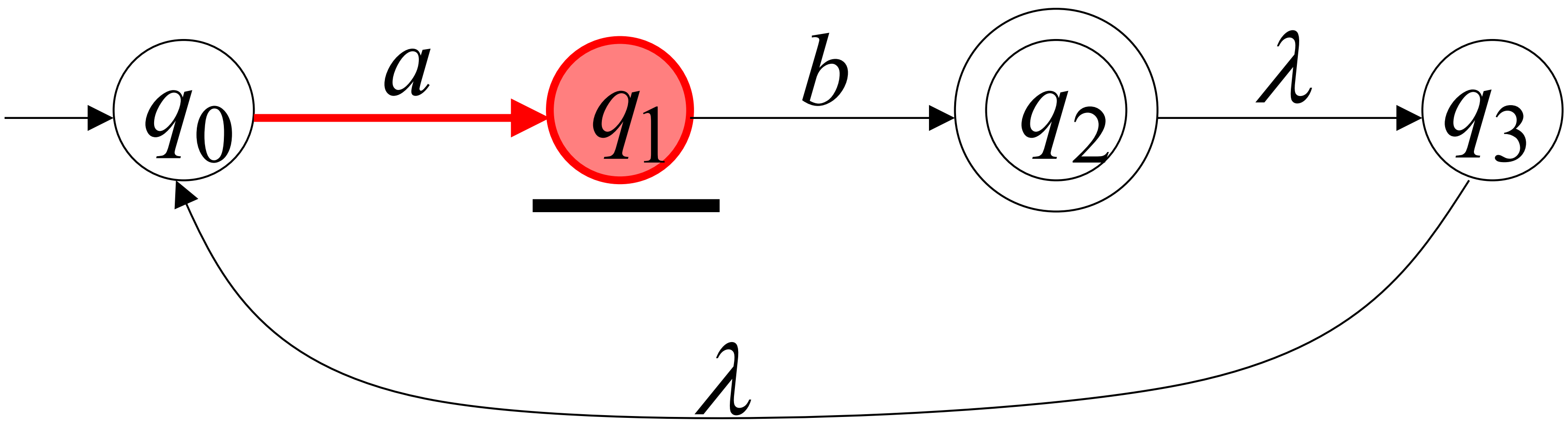

q_0
a
q_1
b
q_2
λ
q_3
λ

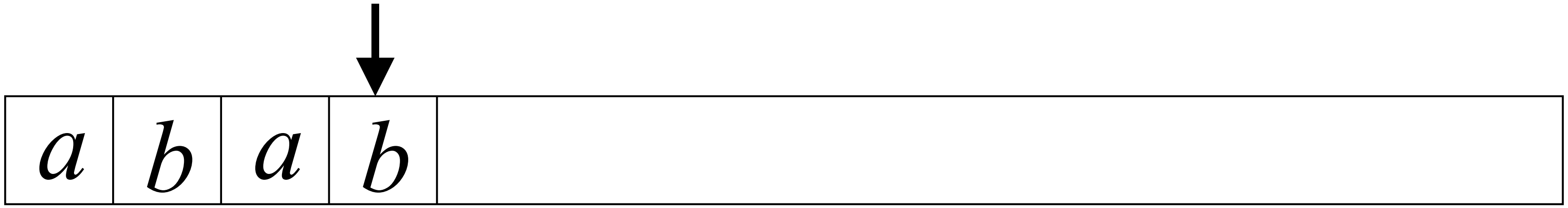

a
b
a
b
a
b
λ
λ
q_0
q_1
q_2
q_3

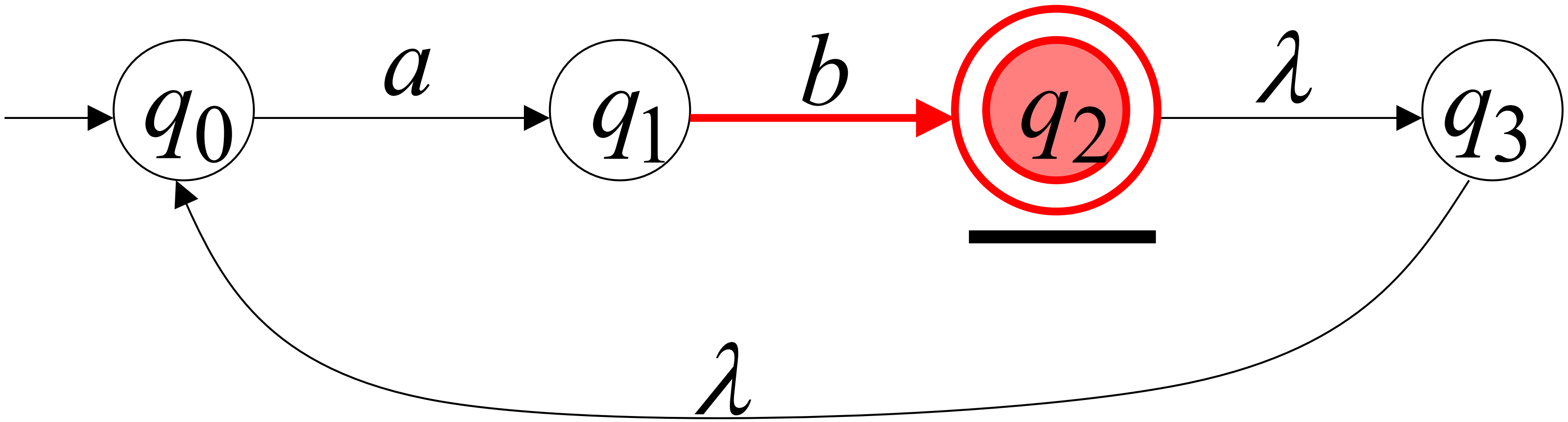

a
b
a
b

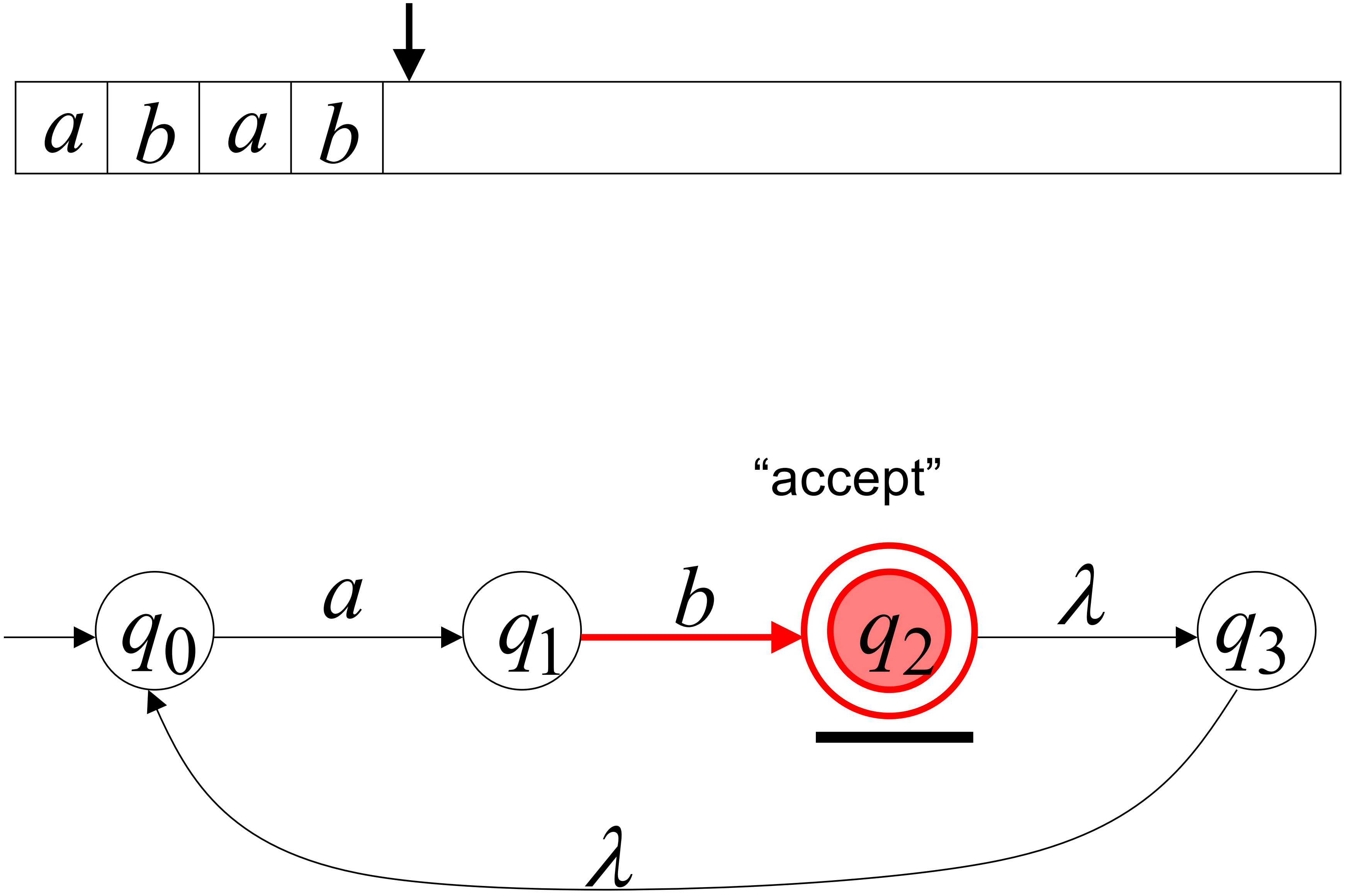

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b


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

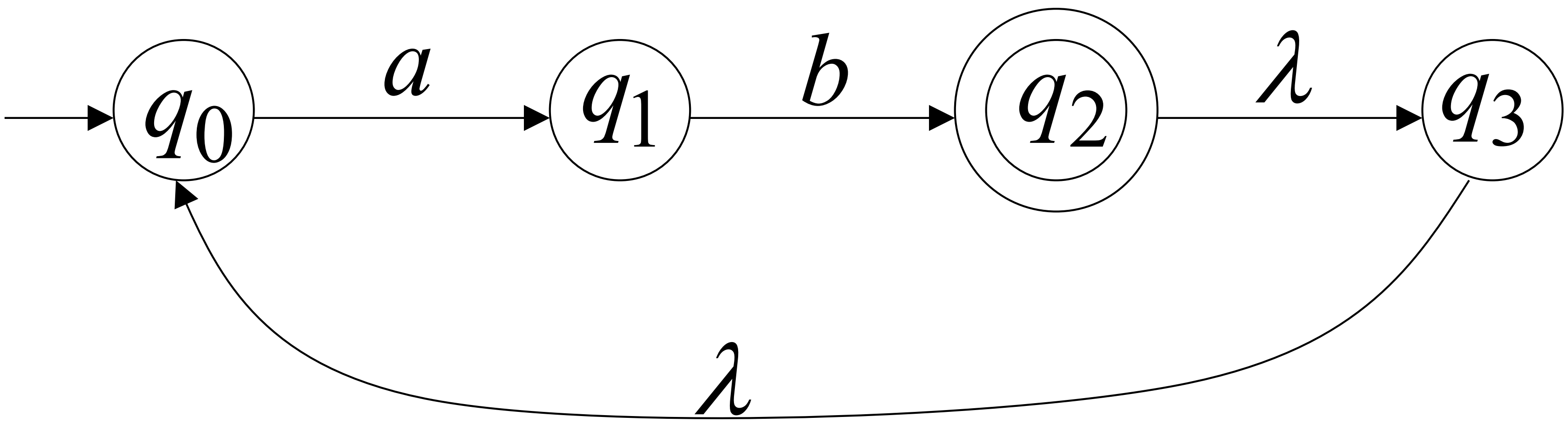


a
b
a
b
"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

Language accepted

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

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

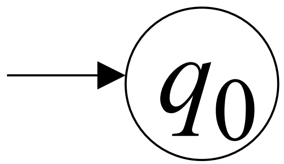


Remarks:

- The λ symbol never appears on the input tape

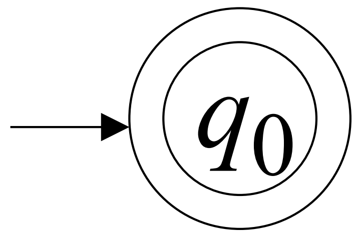
- Simple automata:

M_1



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

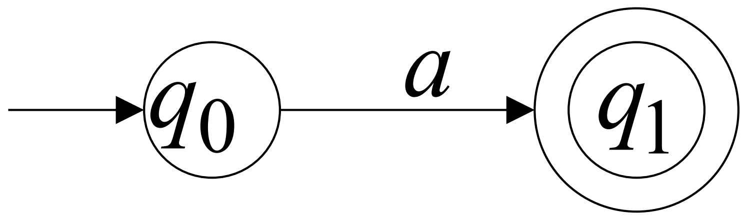
M_2



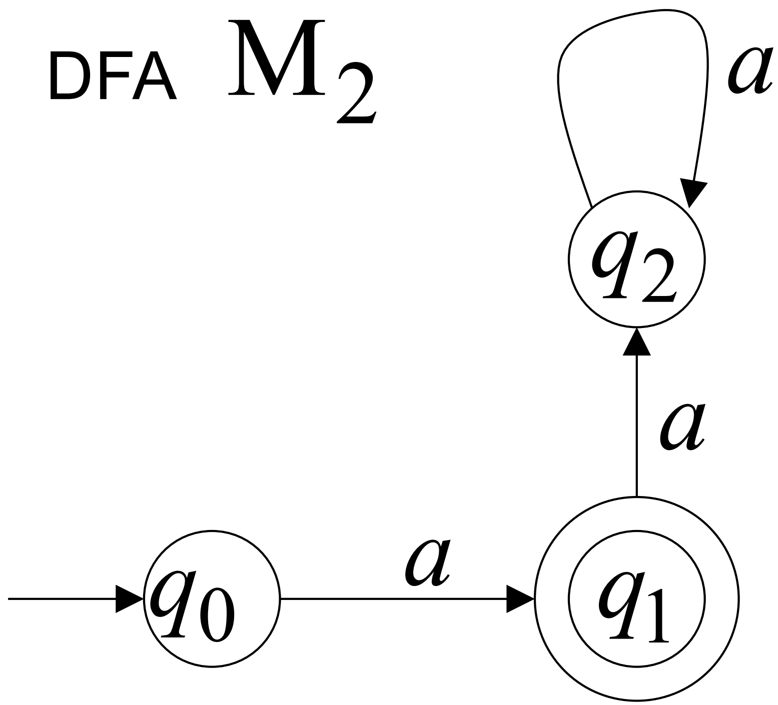
$$L(M_2) = \{\lambda\}$$

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

NFA M_1



DFA M_2



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

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

Formal Definition of NFAs

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

Q : a finite set of **internal states**

Σ : a finite set of symbols called **input alphabet**

δ : $Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ called **transition function**
: $Q \times \Sigma \rightarrow Q$ (DFA)

q_0 : $q_0 \in Q$ is the **initial state**

F : $F \subseteq Q$ is a set of **final states**

Difference Between DFA and NFA

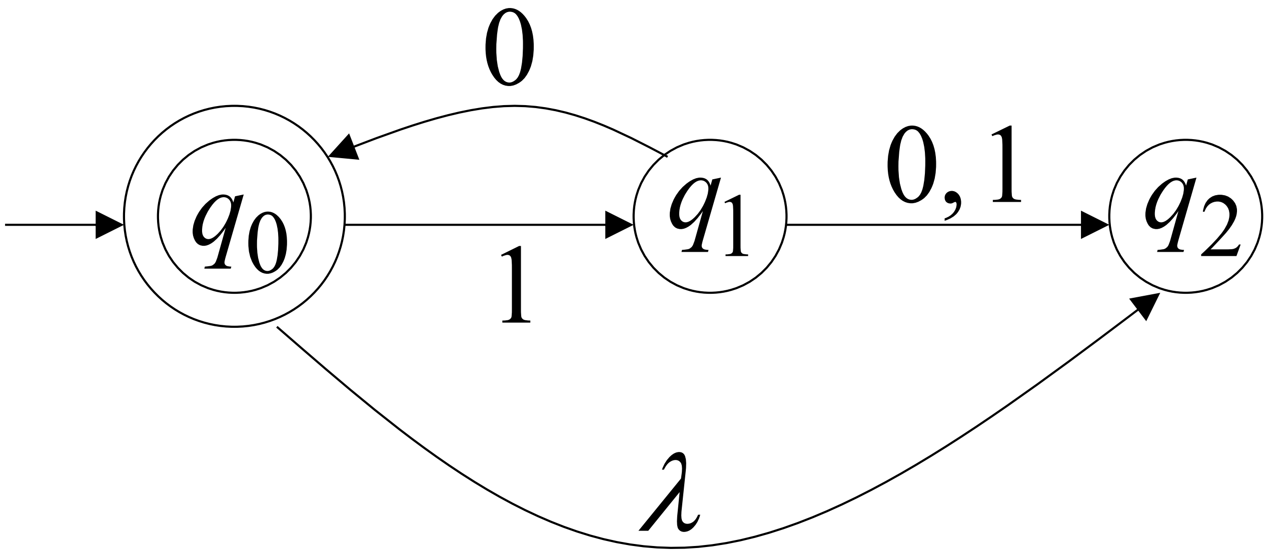
$\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$

In NFA

- The range of δ is in the powerset 2^Q
- It allows λ as the second argument of δ
- The set $\delta(q_i, a)$ may be empty

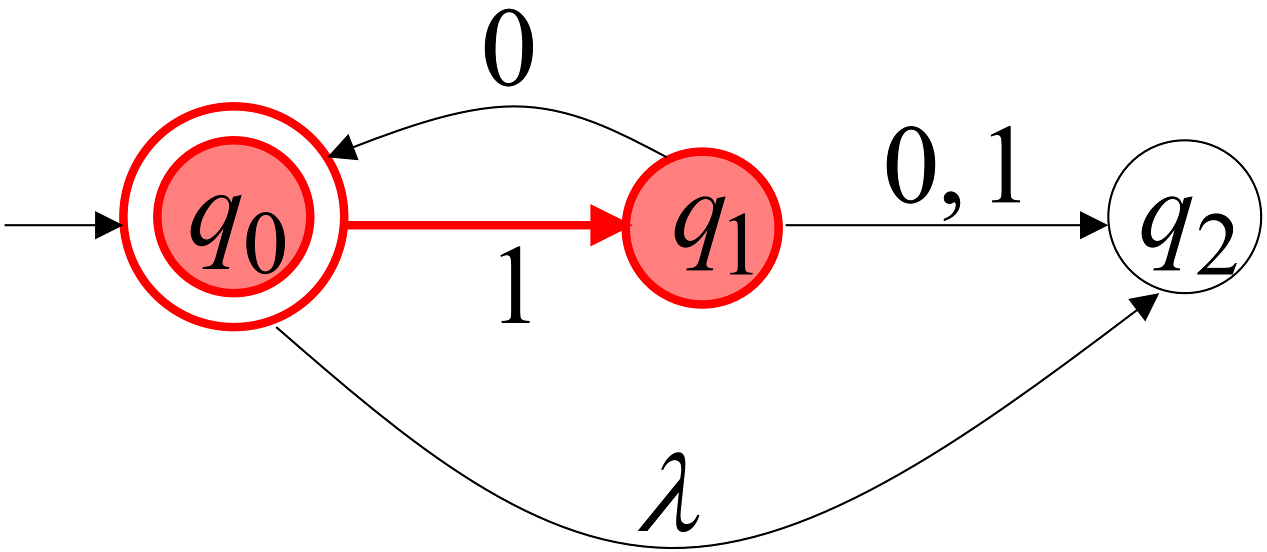
Assume NFA wants to accept every string (try the best move)

Example 2.8

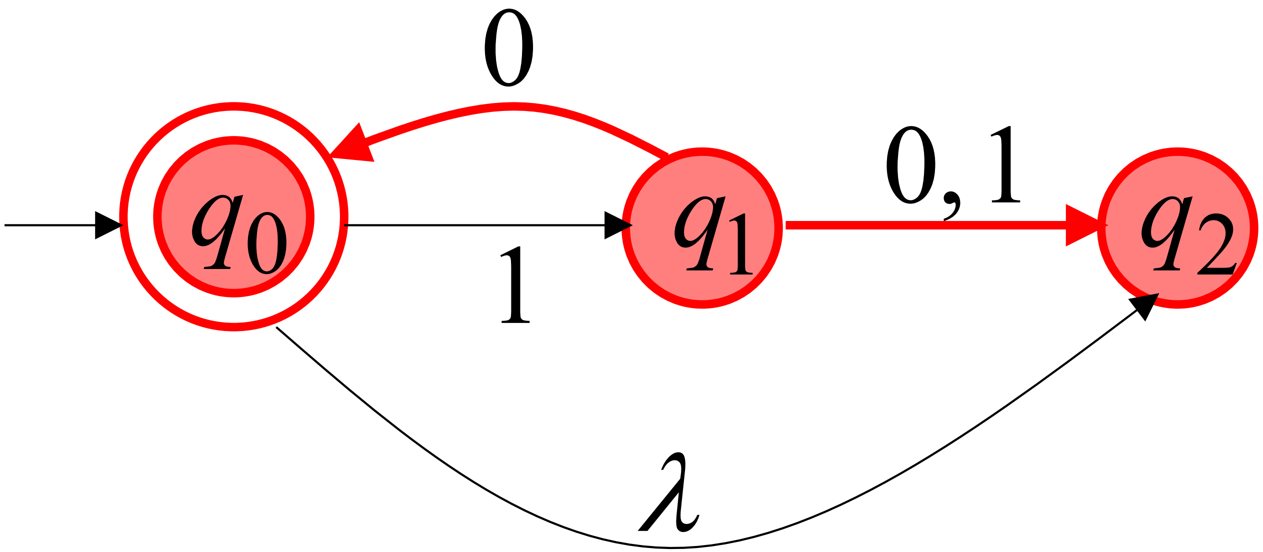


Transition Function δ

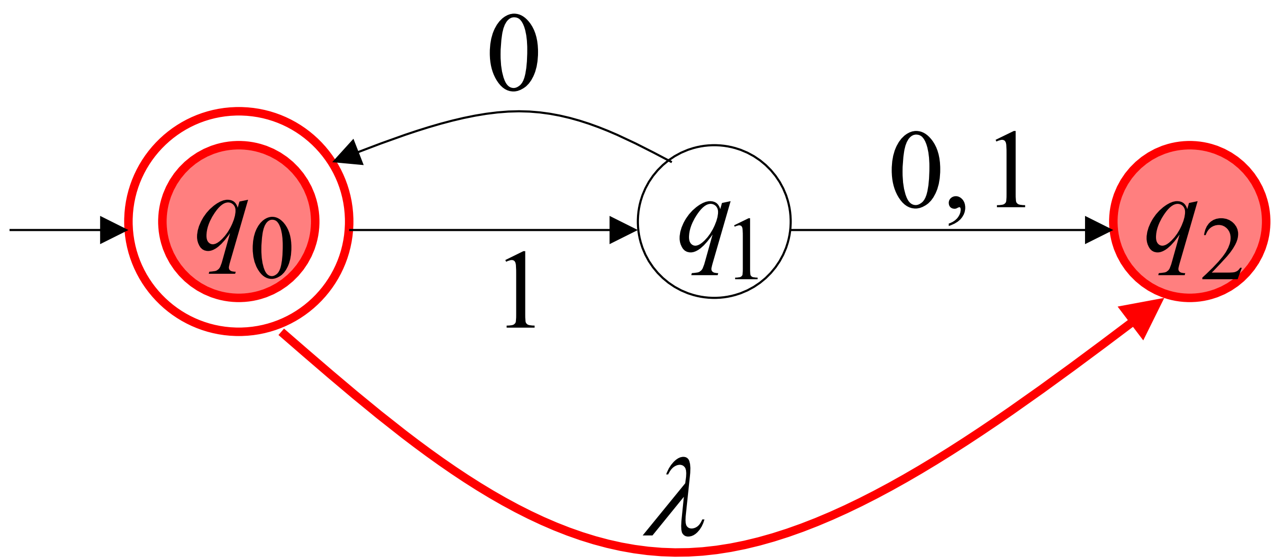
$$\delta(q_0, 1) = \{q_1\}$$



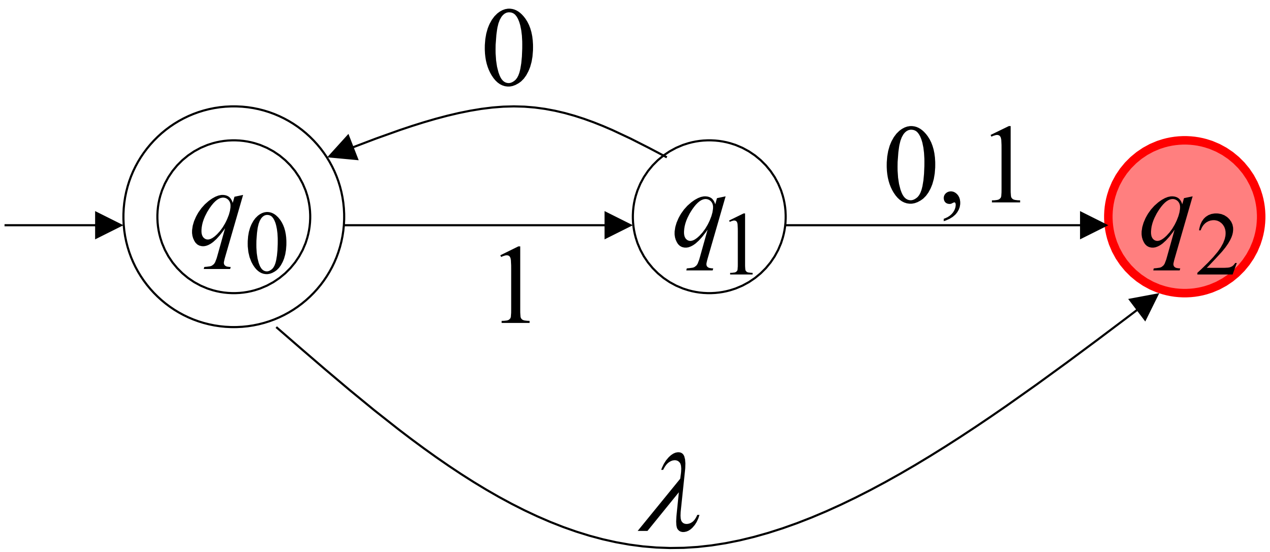
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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



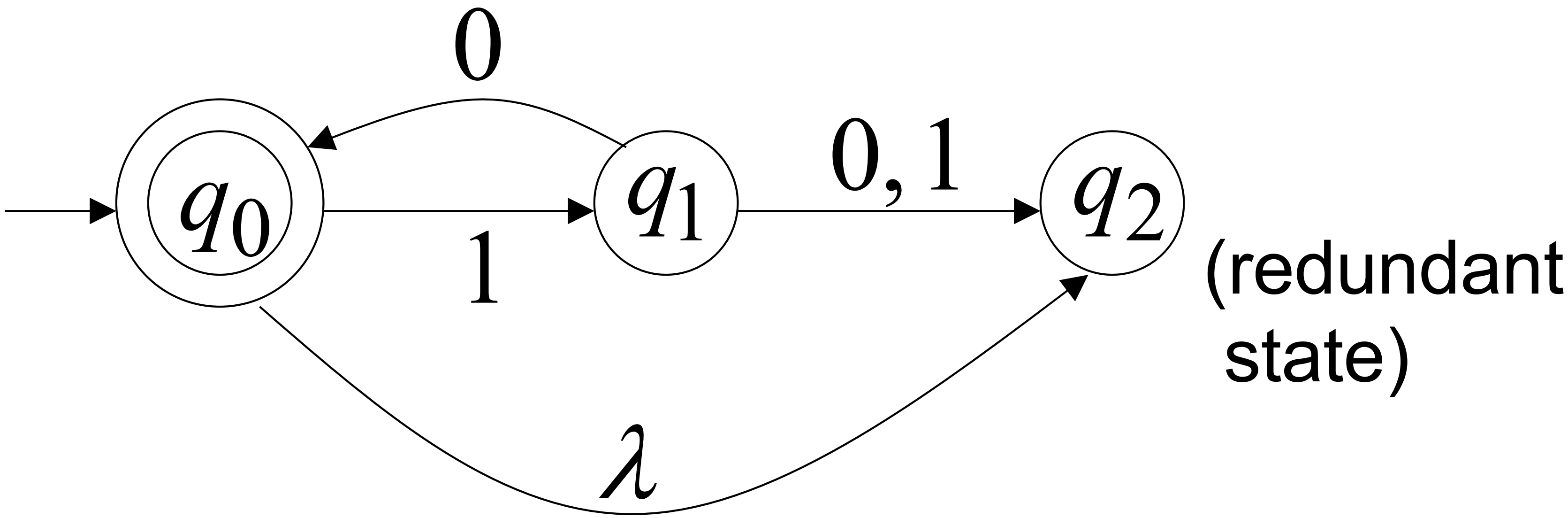
$$\delta(q_2, 1) = \varnothing$$



Language accepted

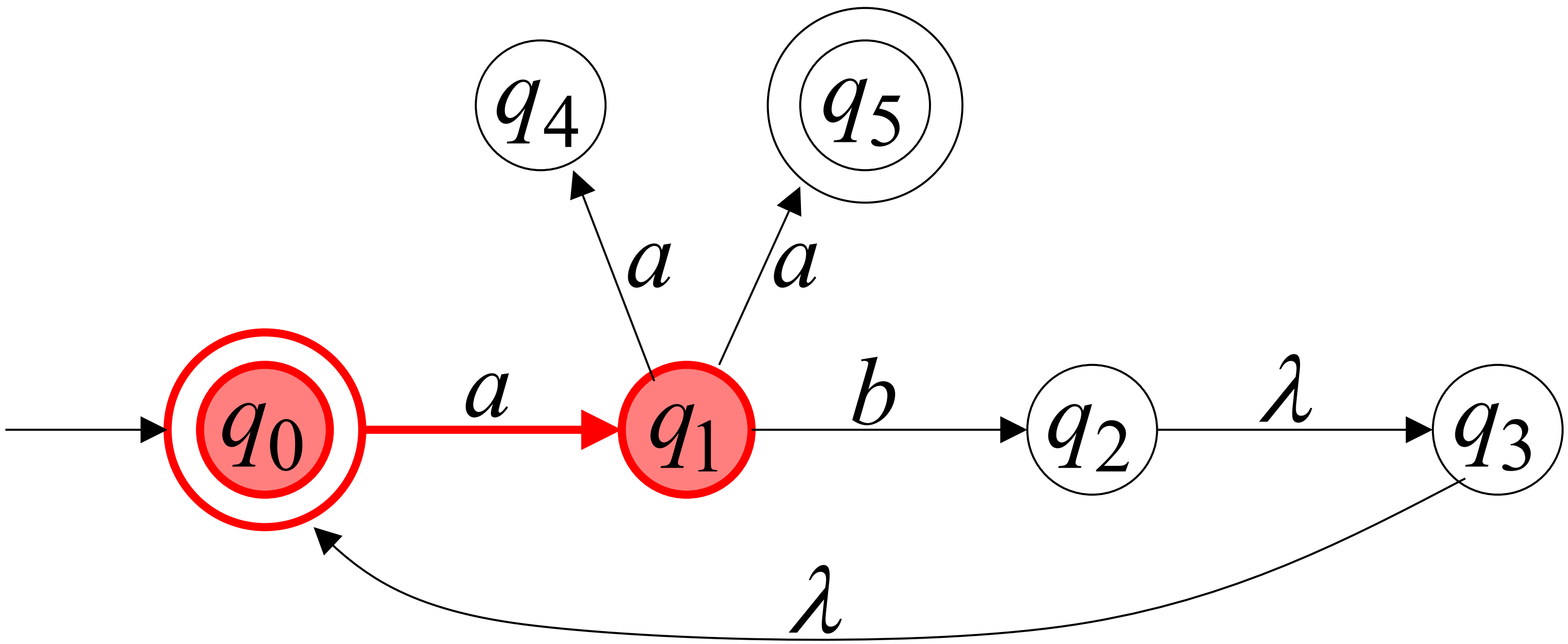
$$L(M) = \{\lambda,\ 10,\ 1010,\ 101010,\ \ldots\}$$

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

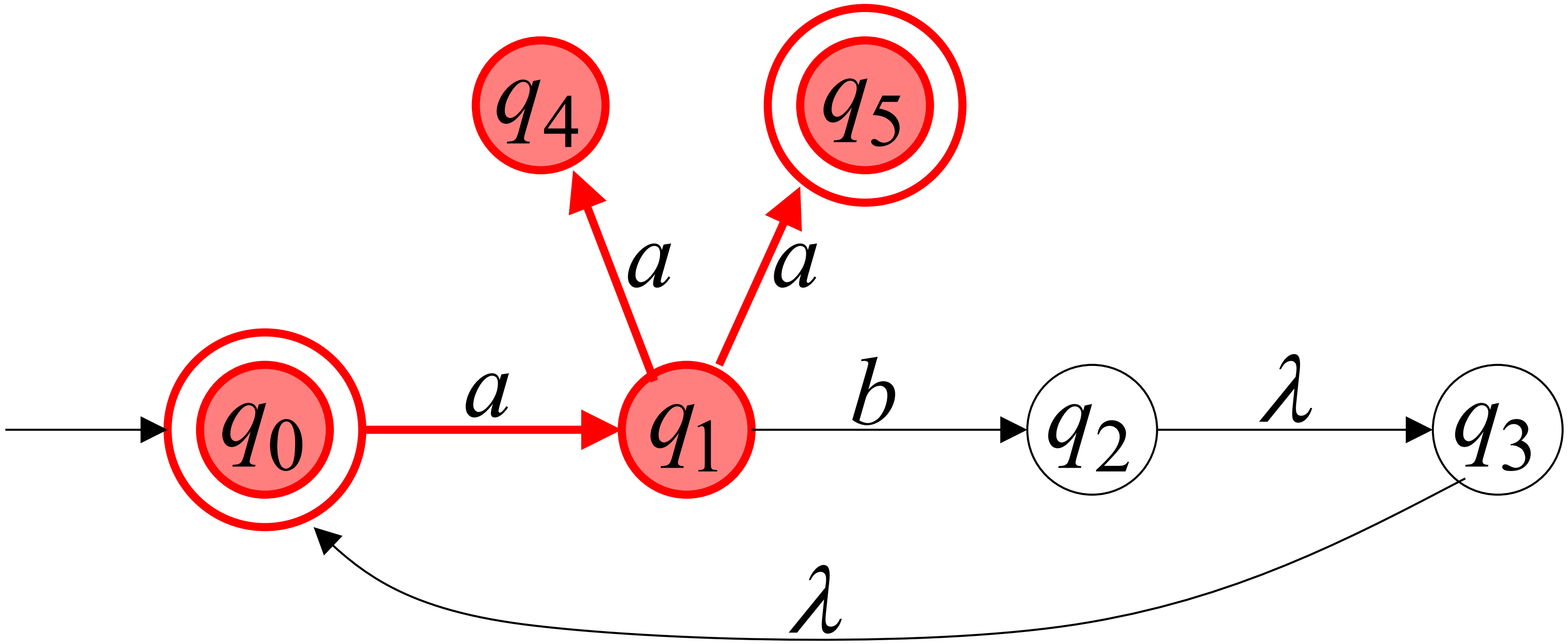


Extended Transition Function δ^*

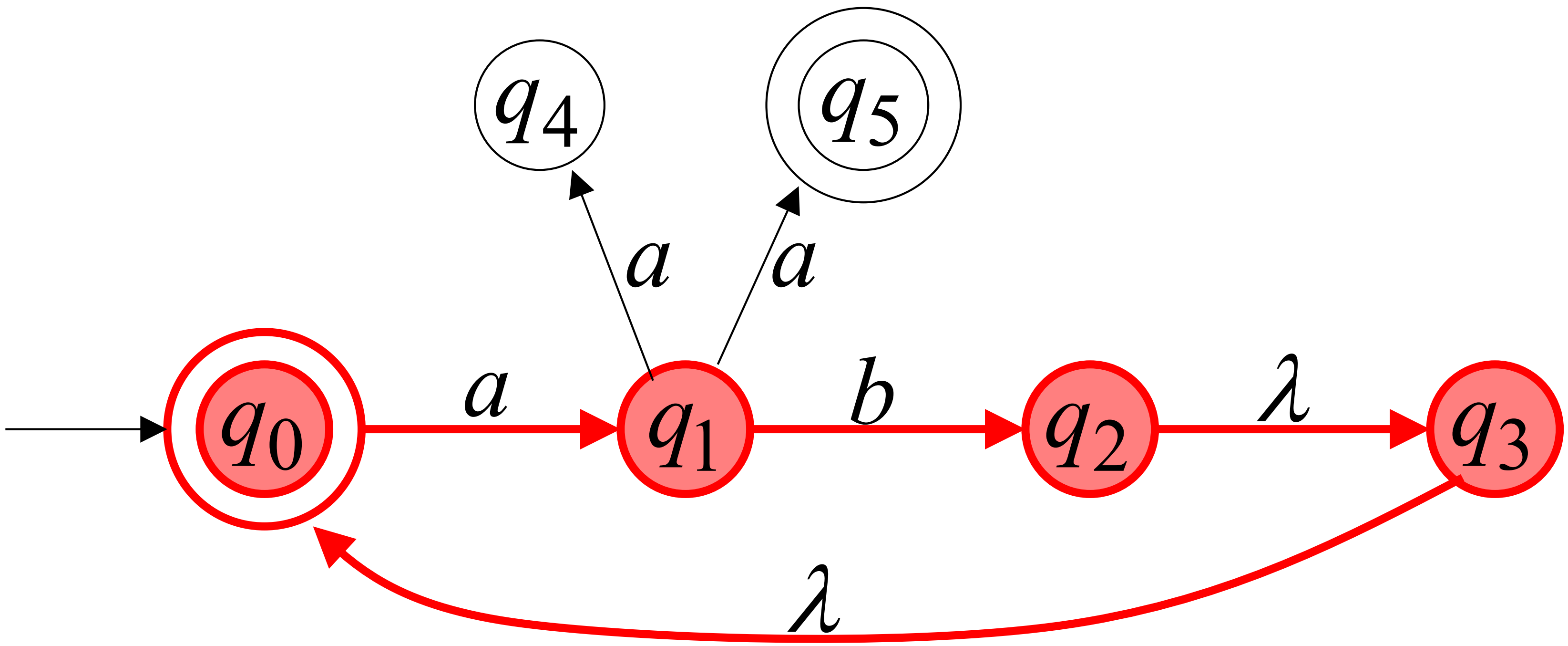
$$\delta^*(q_0, a) = \{q_1\}$$



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

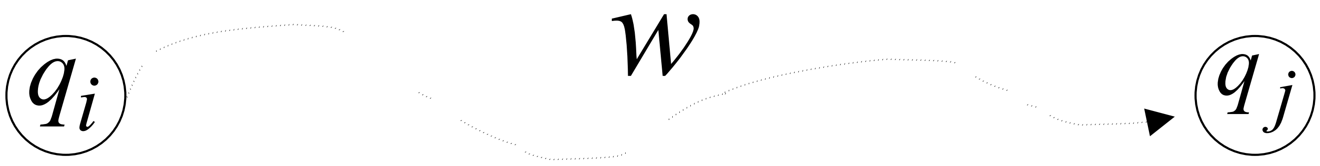


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

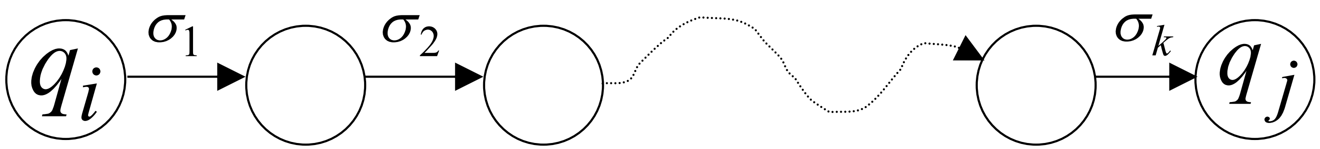


Formally

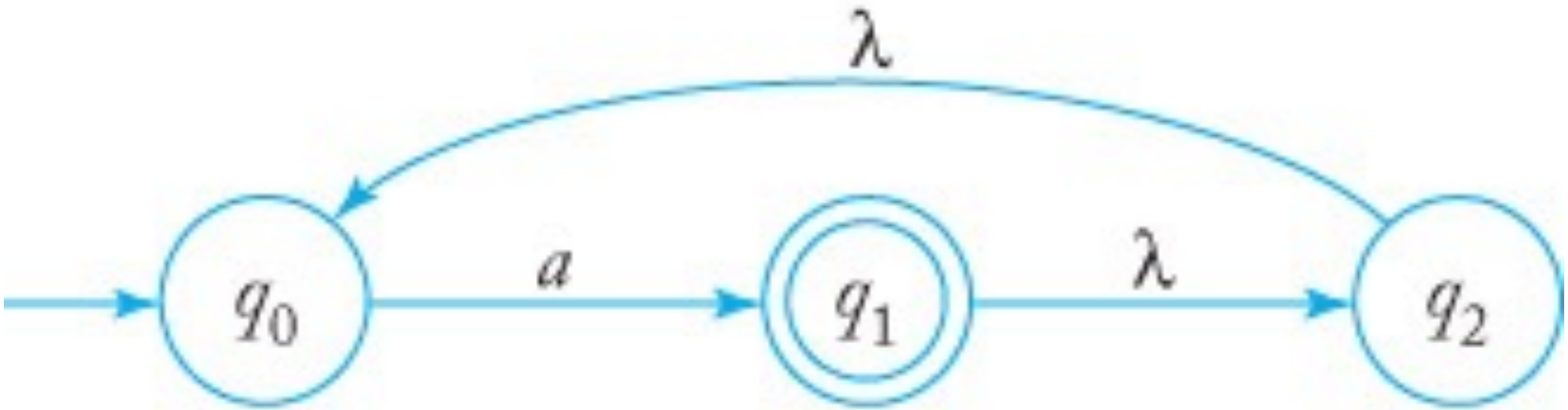
$q_j \in \delta^*(q_i, w)$: there is a walk from q_i to q_j with label w



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



Example 2.9

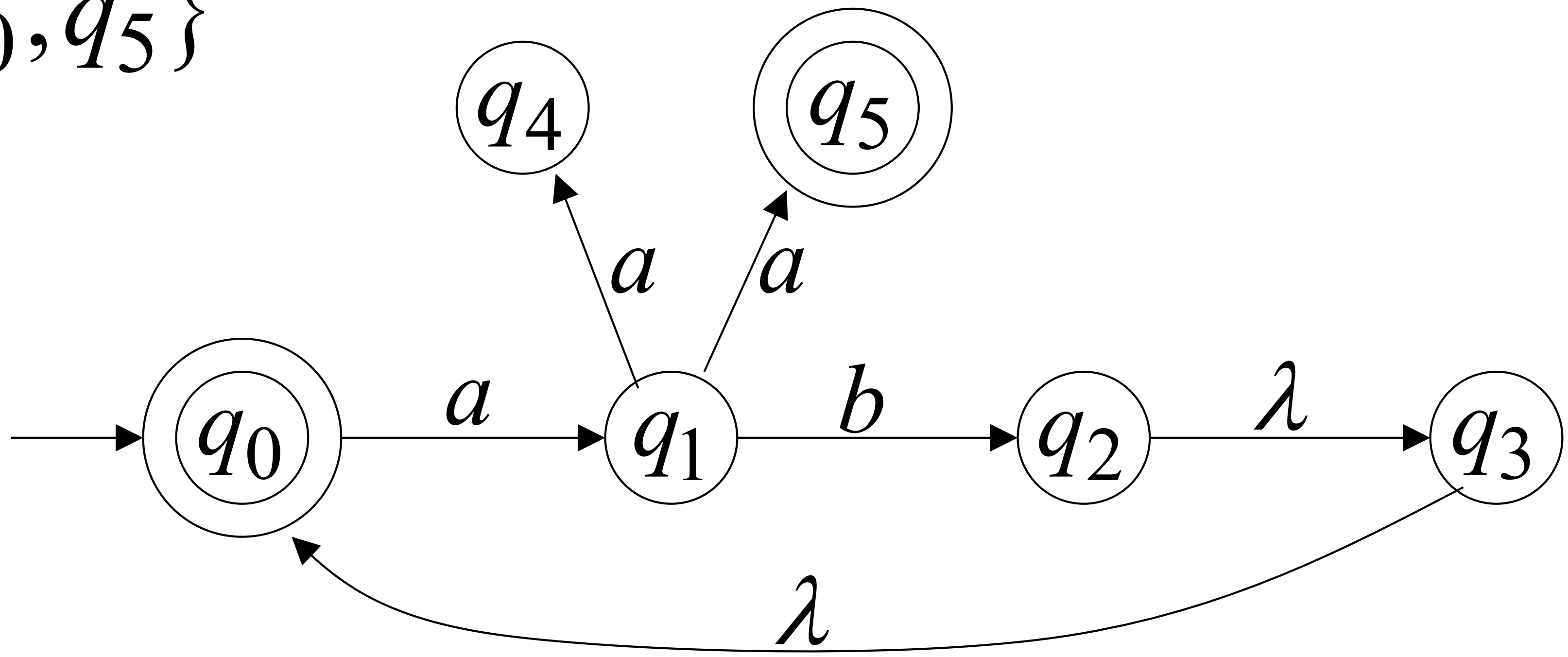


$$\delta^*(q_1, a) = \{q_0, q_1, q_2\}$$
$$\delta^*(q_2, \lambda) = \{q_0, q_2\}$$
$$\delta^*(q_2, aa) = \{q_0, q_1, q_2\}$$

The length of a walk labeled *a* between q_1 and q_2 is 4

The Language of an NFA M

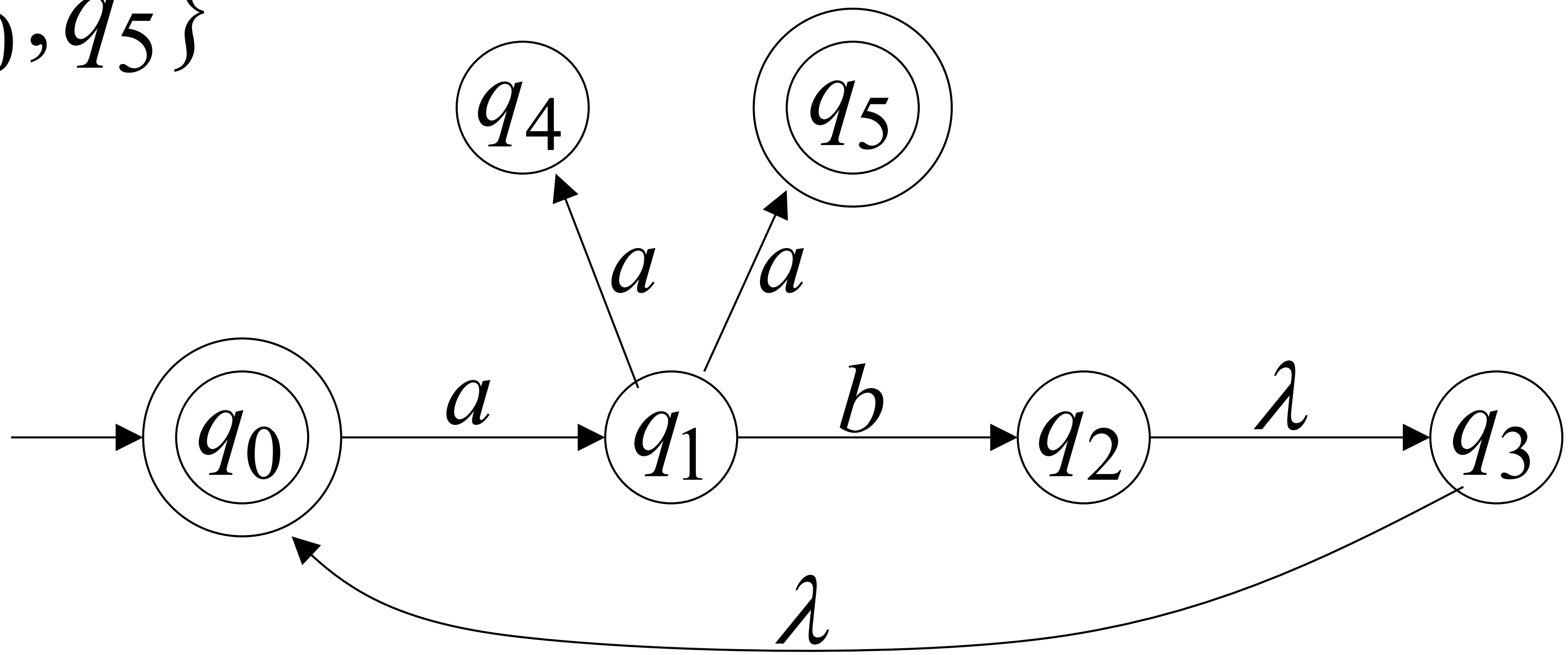
$$F = \{q_0, q_5\}$$



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

$$\searrow \in F$$

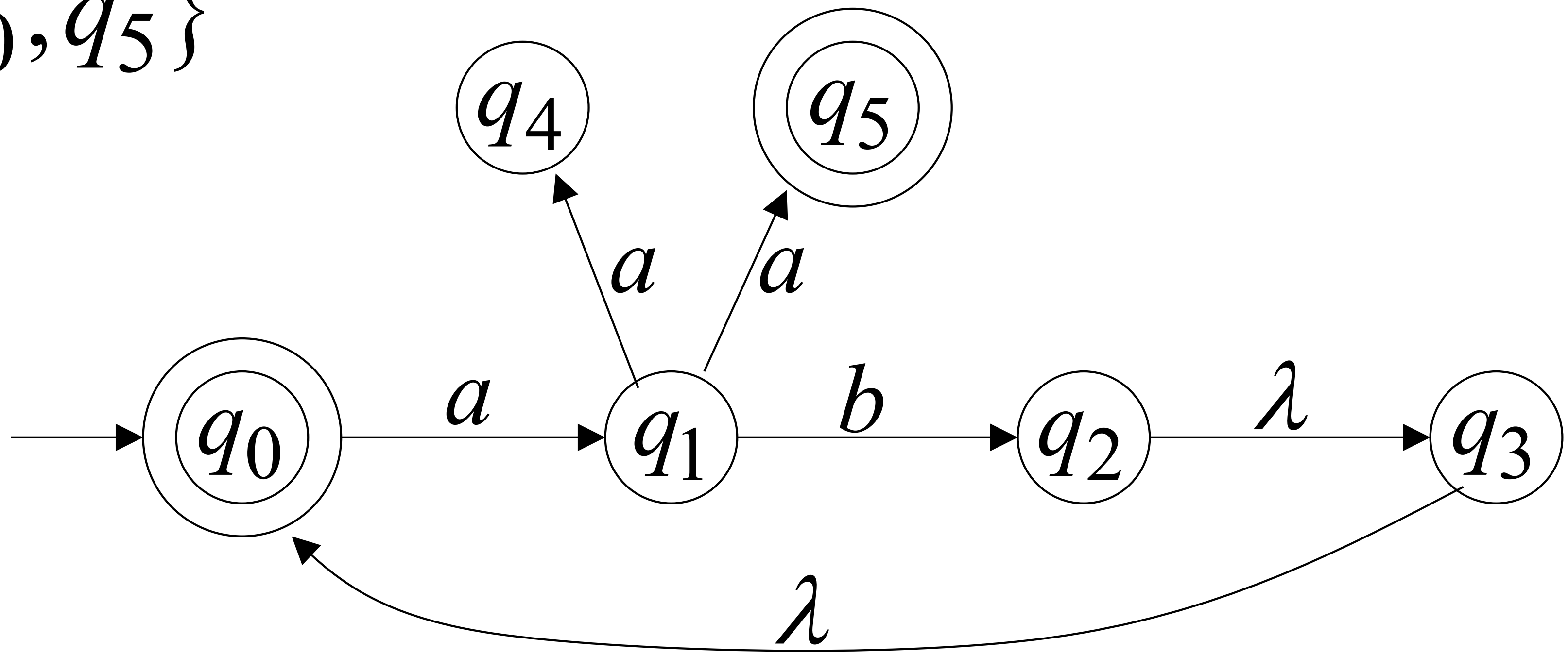
$$F = \{q_0, q_5\}$$



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

$$\searrow \in F$$

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

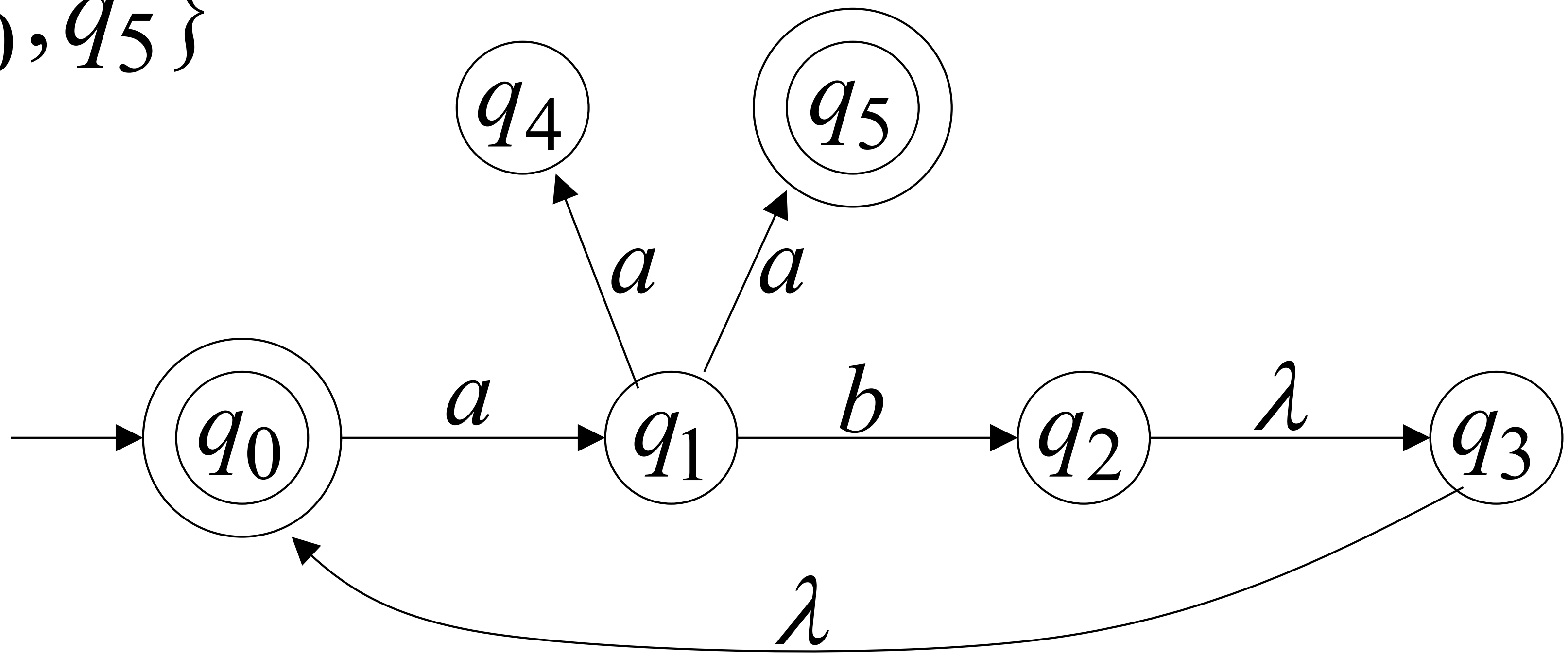


$$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\}$$

$\in F$

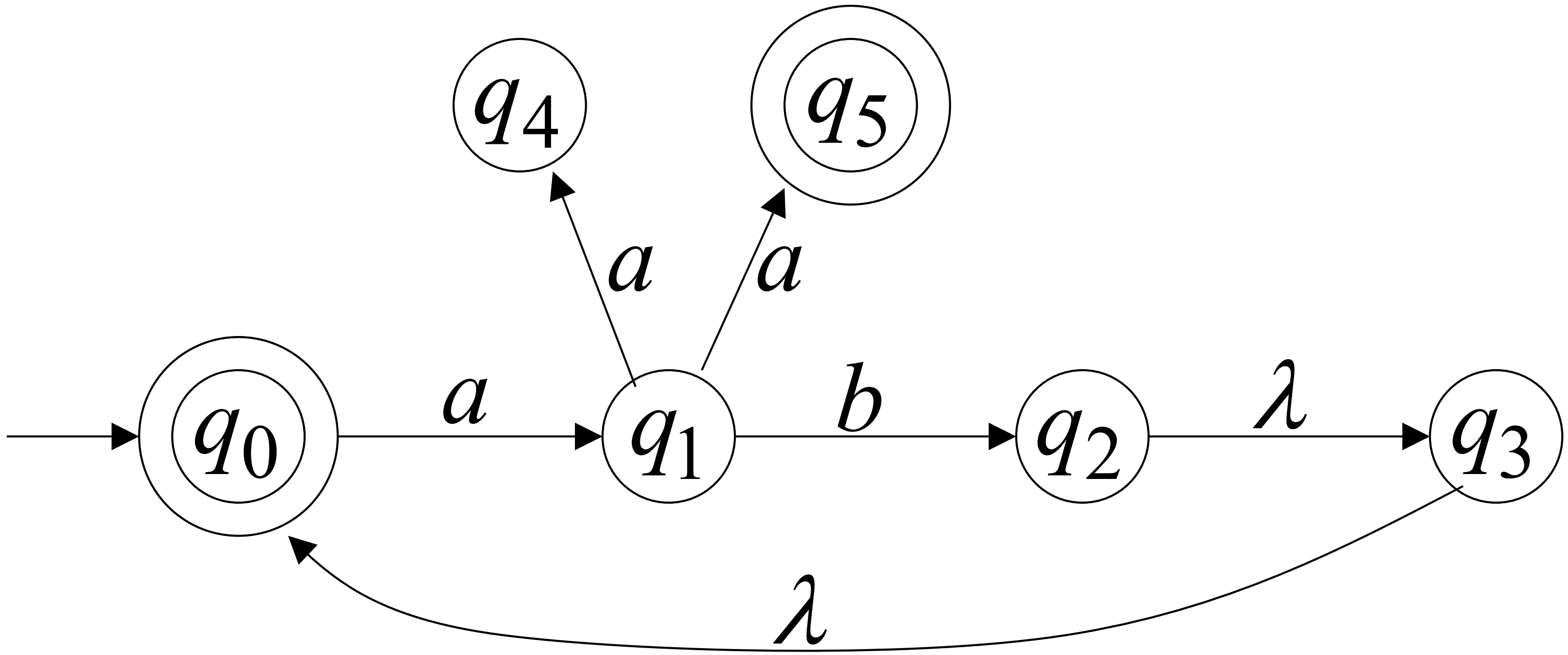
$$abaa \in L(M)$$

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



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

$$\notin F$$



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

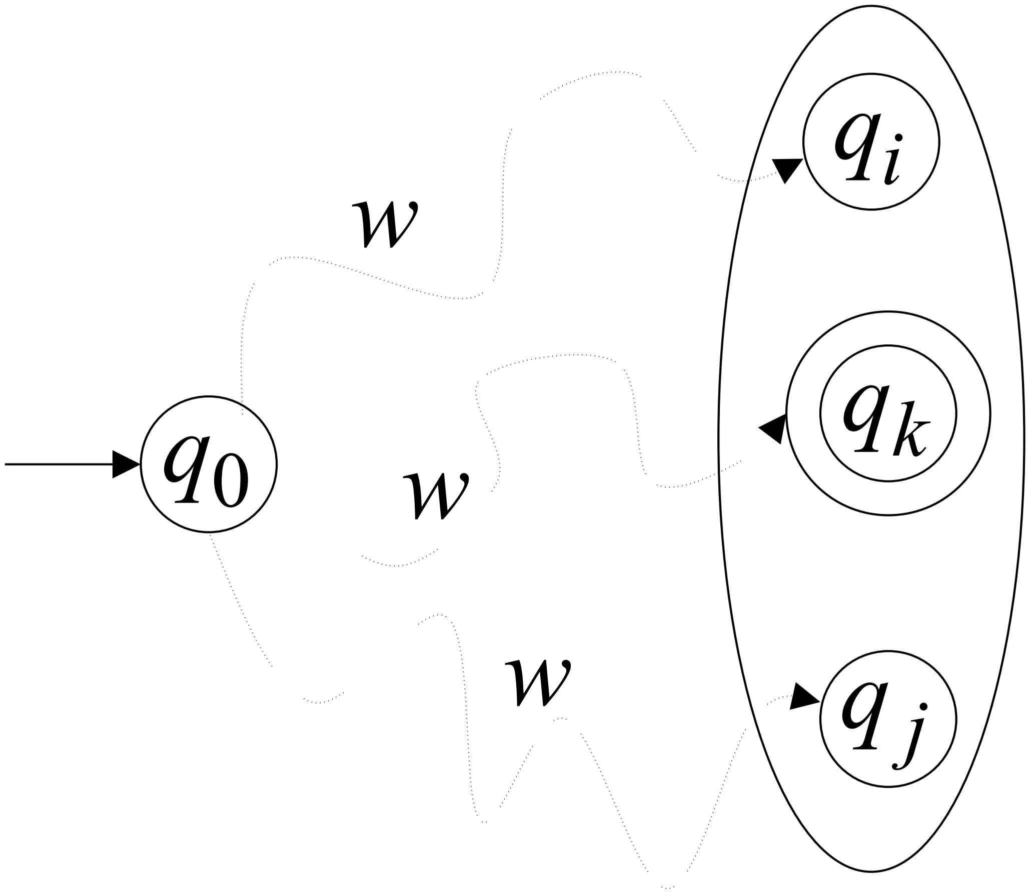
Definition 2.6

The language L accepted by an NFA M is defined as the set of all accepted strings :

$$L(M) = \left\{ w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \phi \right\}$$

$w \in L(M)$

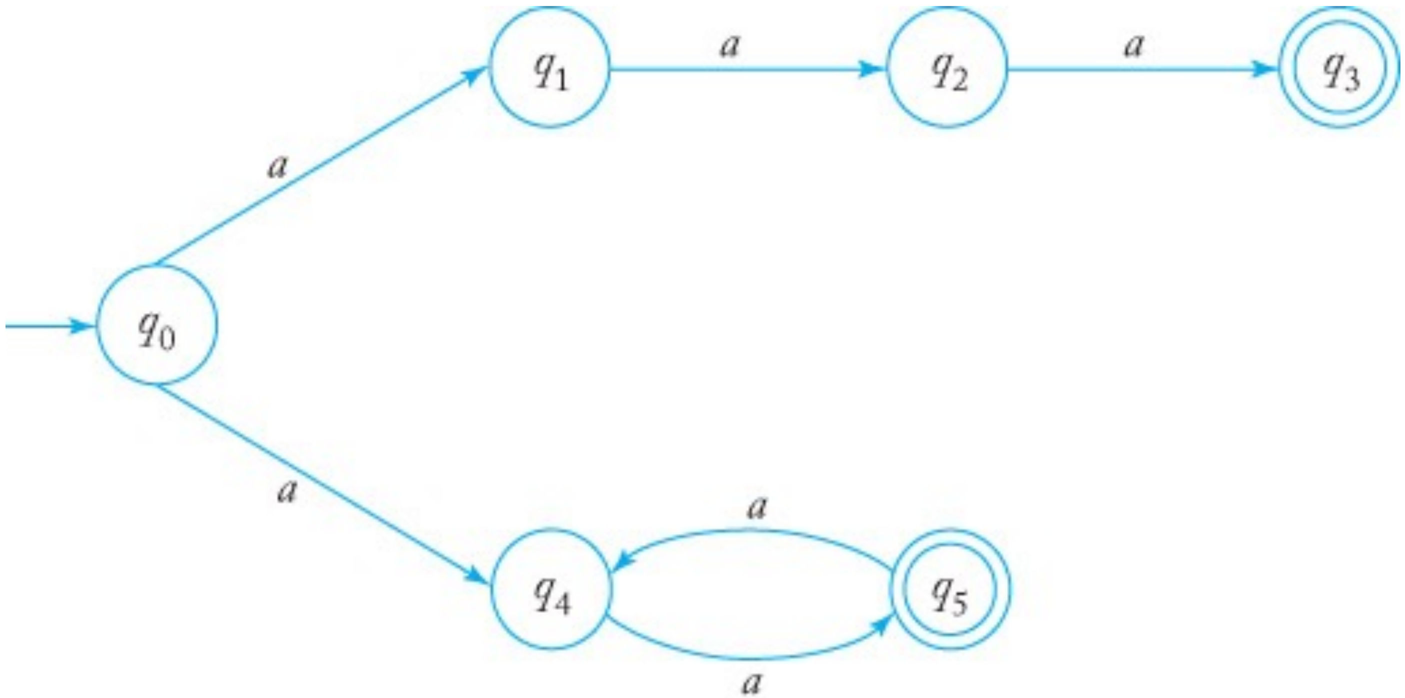
$\delta^*(q_0, w)$



$q_k \in F$

Why Nondeterminism?

- Many deterministic algorithms require that one make a choice at some stage (game-playing program, TSP, etc)
- Nondeterminism is sometimes helpful in solving problems easily



The language accepted by the NFA is

$\{a^3\} \cup \{a^{2n}: n \geq 1\}$

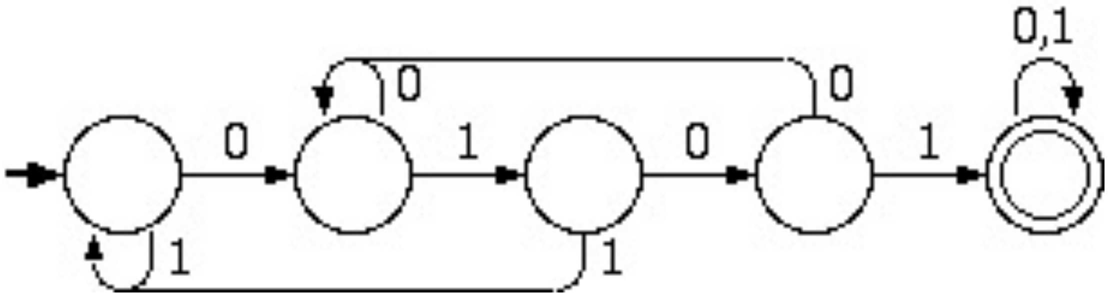
Why Nondeterminism?

- Nondeterminism is an effective mechanism for describing some complicated languages concisely.

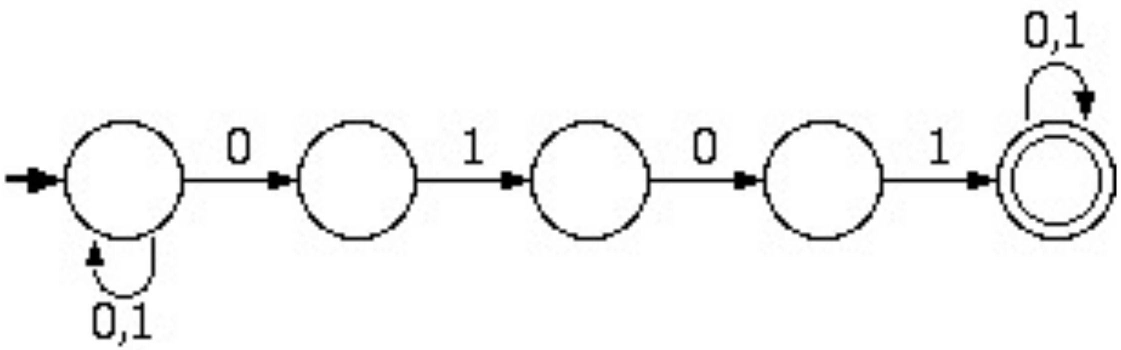
  Ex: $S \to aSb \mid \lambda$

More Examples

– All strings that contain the substring 0101.



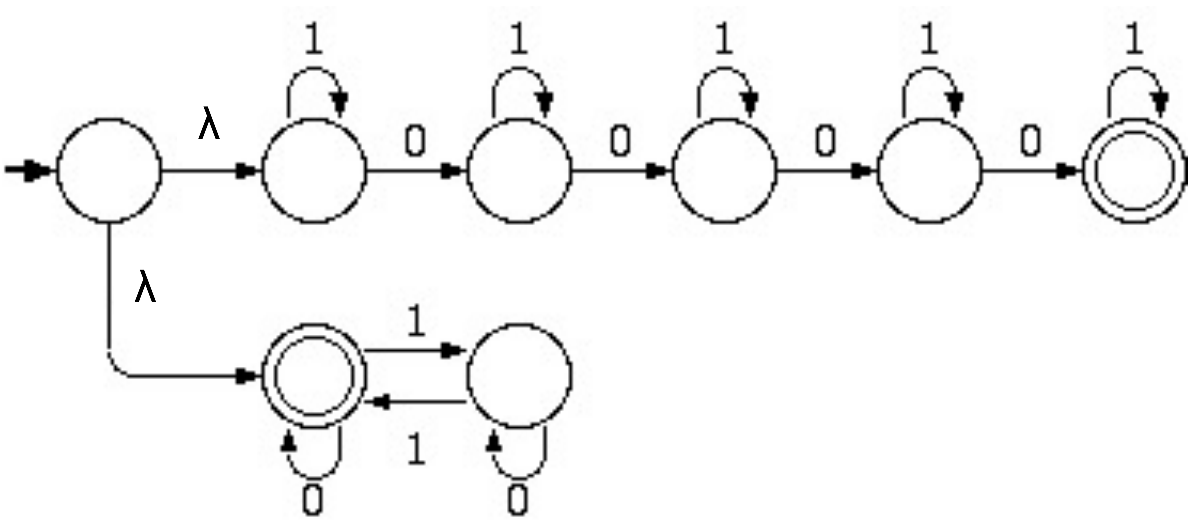
DFA



NFA
(5 states)

More Examples

– All strings containing exactly 4 0s or an even number of 1s. (8 states)



NFA

Outline

1. Deterministic Finite Accepters (DFA)
2. Nondeterministic Finite Accepters (NFA)
3. Equivalence of DFA and NFA
4. Reduction of the Number of States in FA*

Equivalence of Machines

- Definition 2.7:

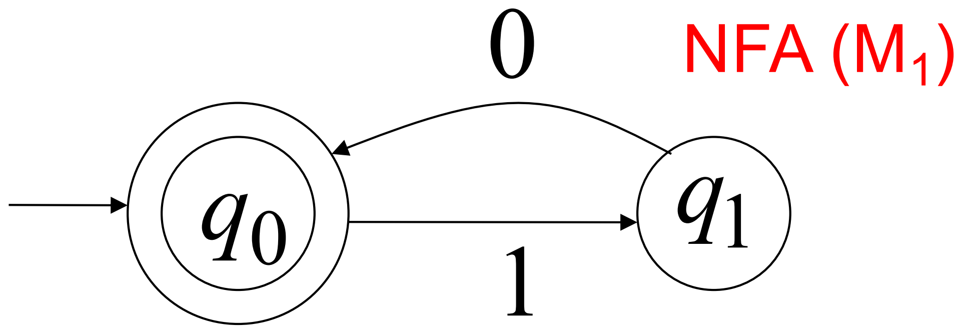
Two finite accepters M_1 and M_2 are said to be equivalent if

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

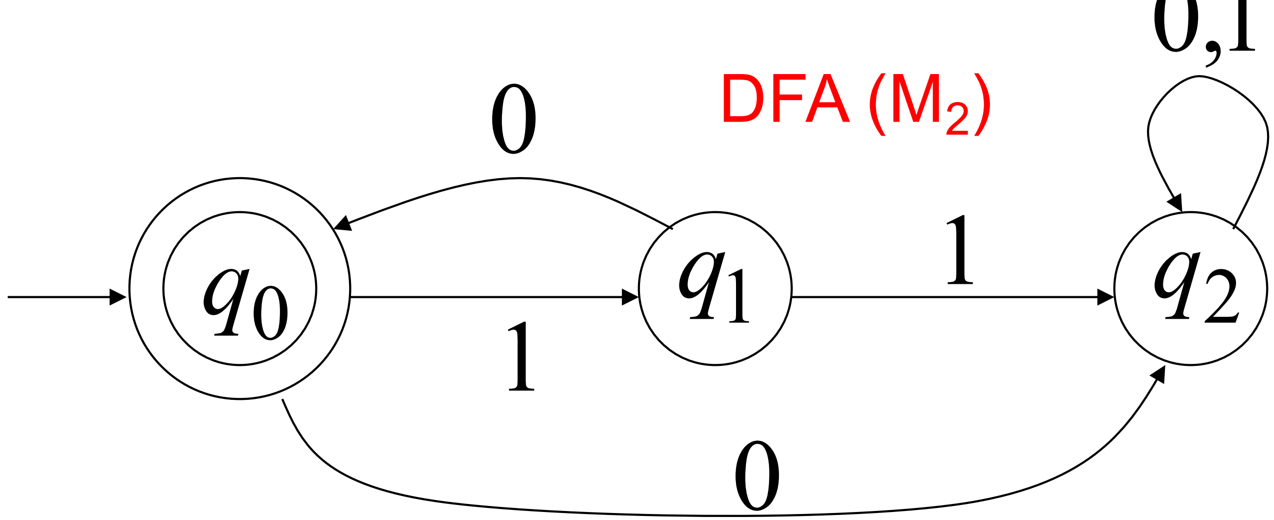
that is, if they both accept the same language.

Example of equivalent machines

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



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



DFA v.s. NFA

- Which one is more powerful?
- “More powerful” means
  – An automaton of one kind can achieve something that cannot be done by any automaton of the other kind
- Trivially, DFA is a restricted kind of NFA

NFAs and DFAs have the same computation power

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

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

$\Downarrow$

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

Step 2

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

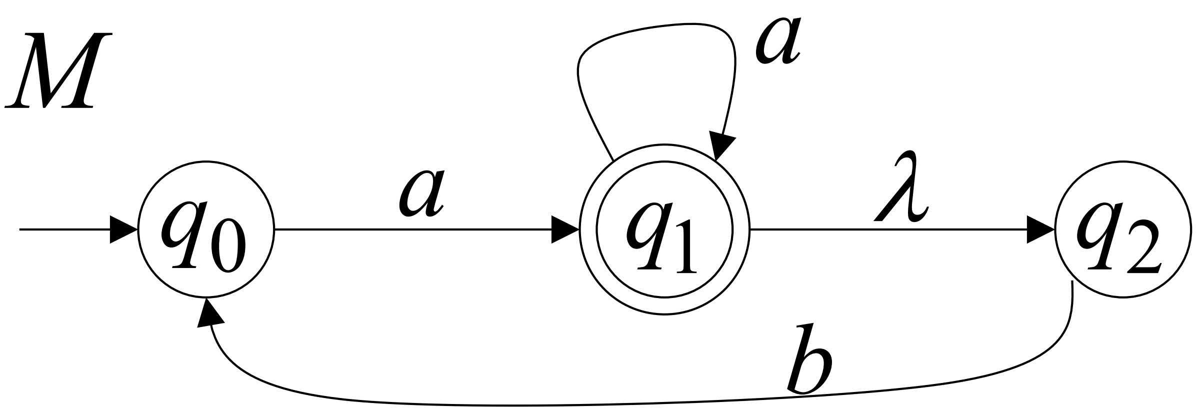
Proof: Any NFA can be converted to an equivalent DFA

$\Downarrow$

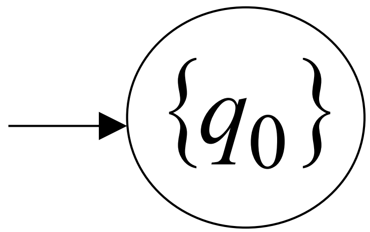
Any language L accepted by an NFA is also accepted by a DFA

Convert NFA to DFA

NFA M

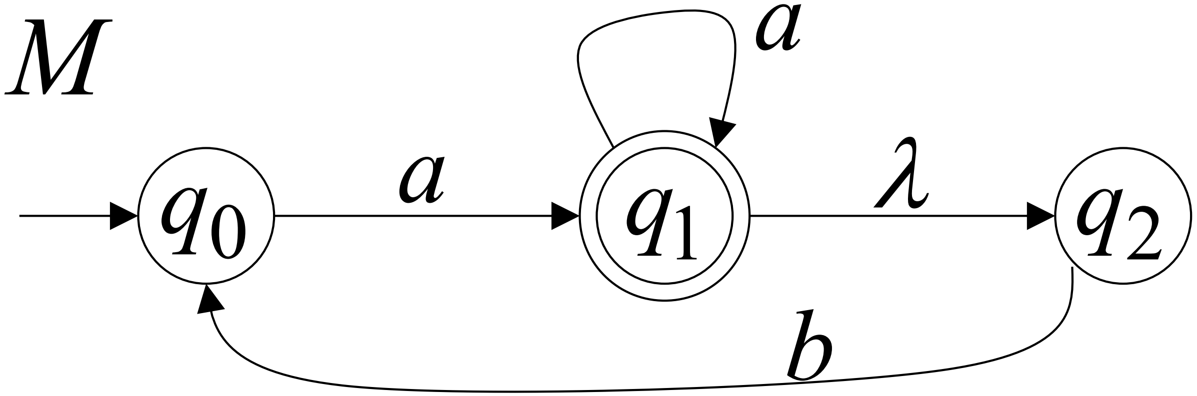


DFA M'

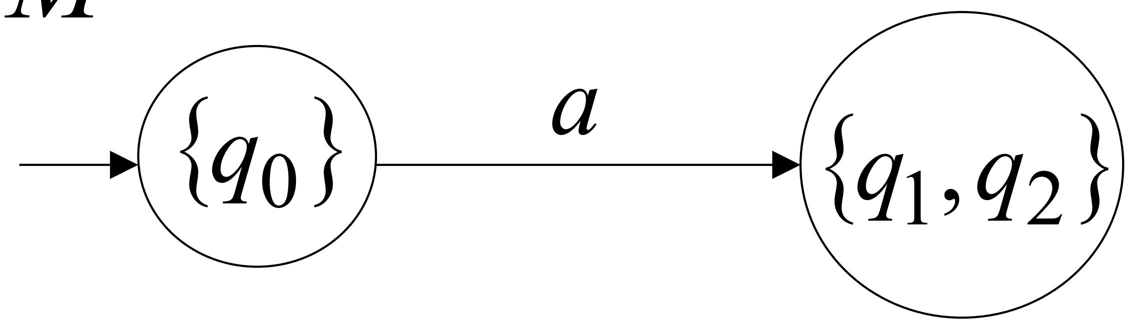


Convert NFA to DFA

NFA M

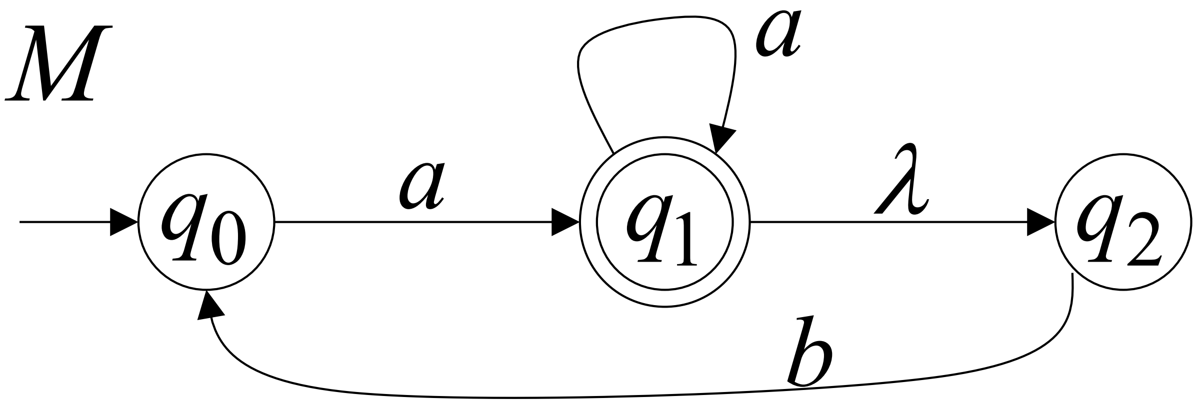


DFA M'

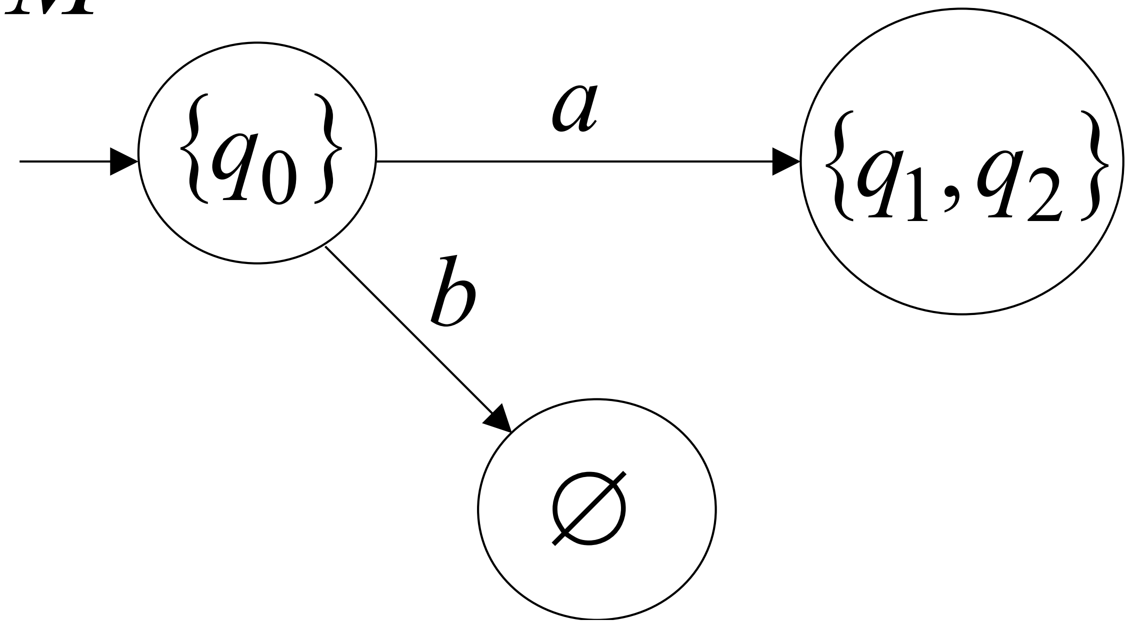


Convert NFA to DFA

NFA M

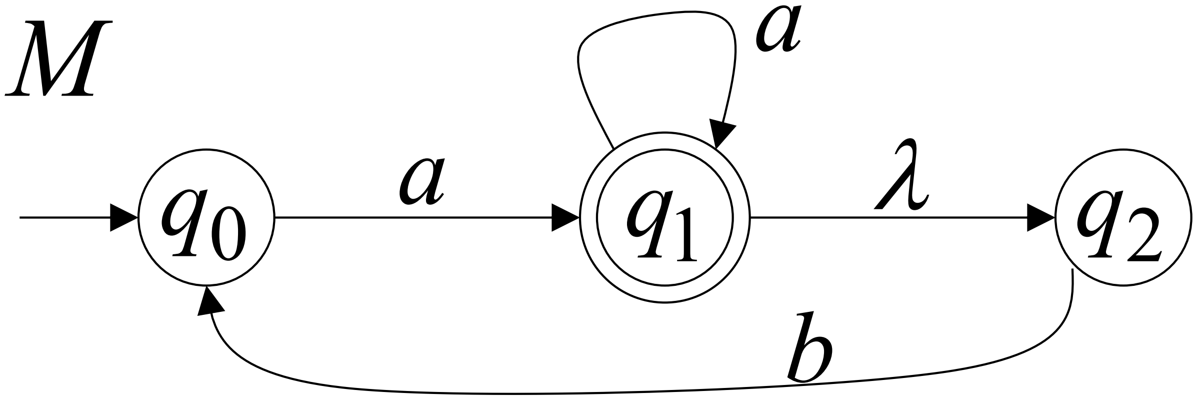


DFA M'

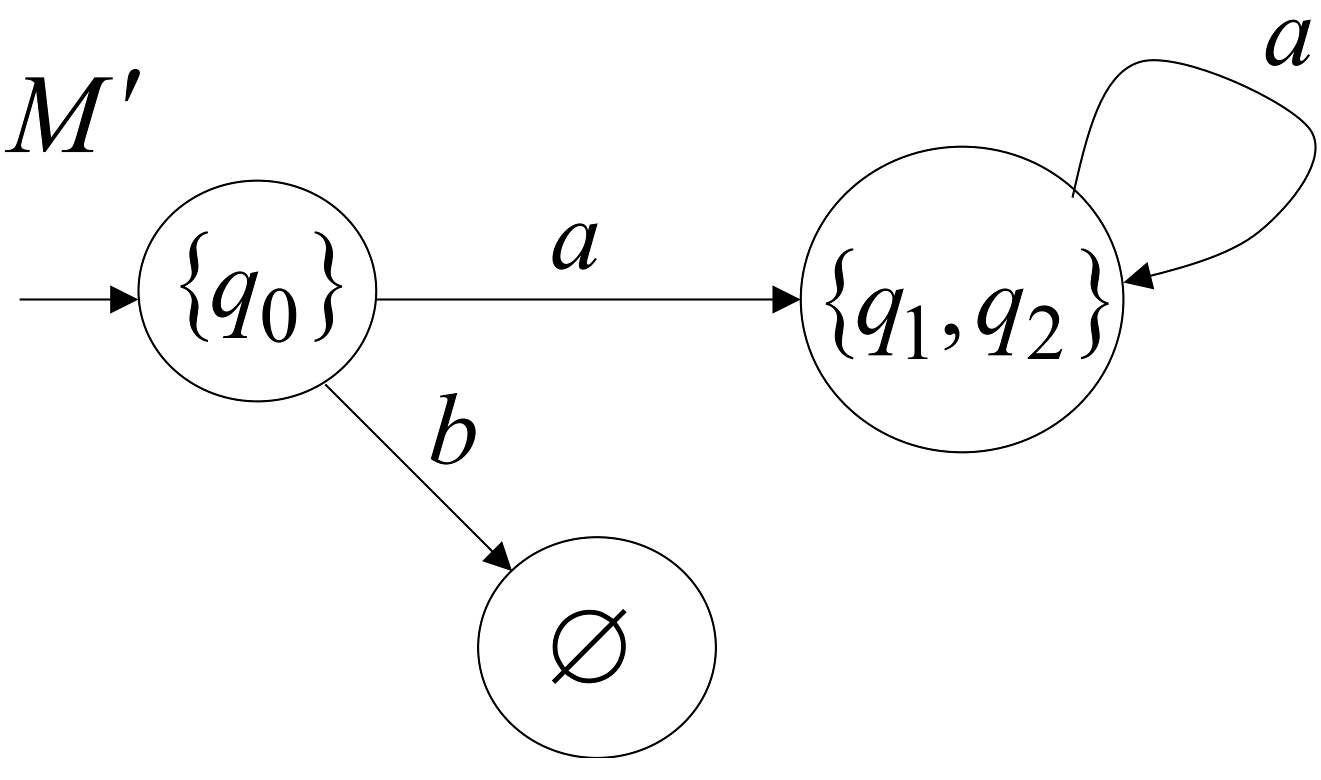


Convert NFA to DFA

NFA M

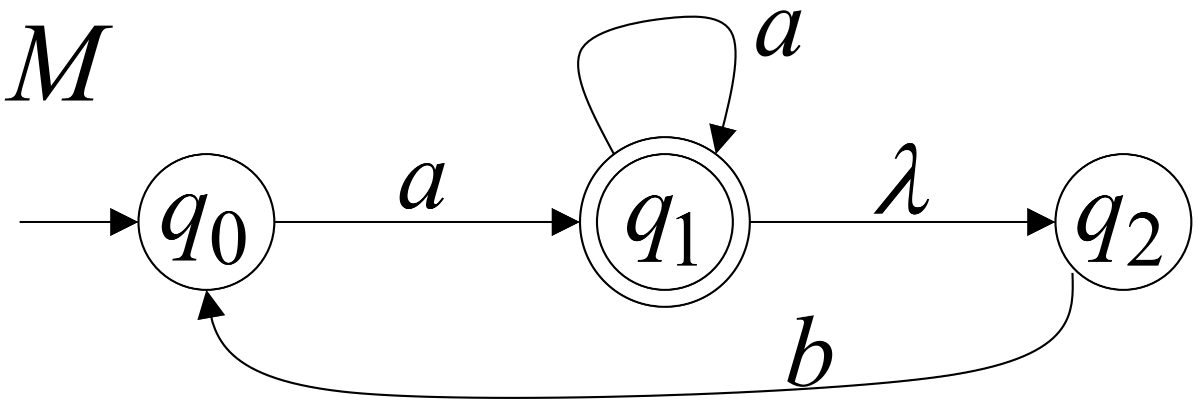


DFA M'

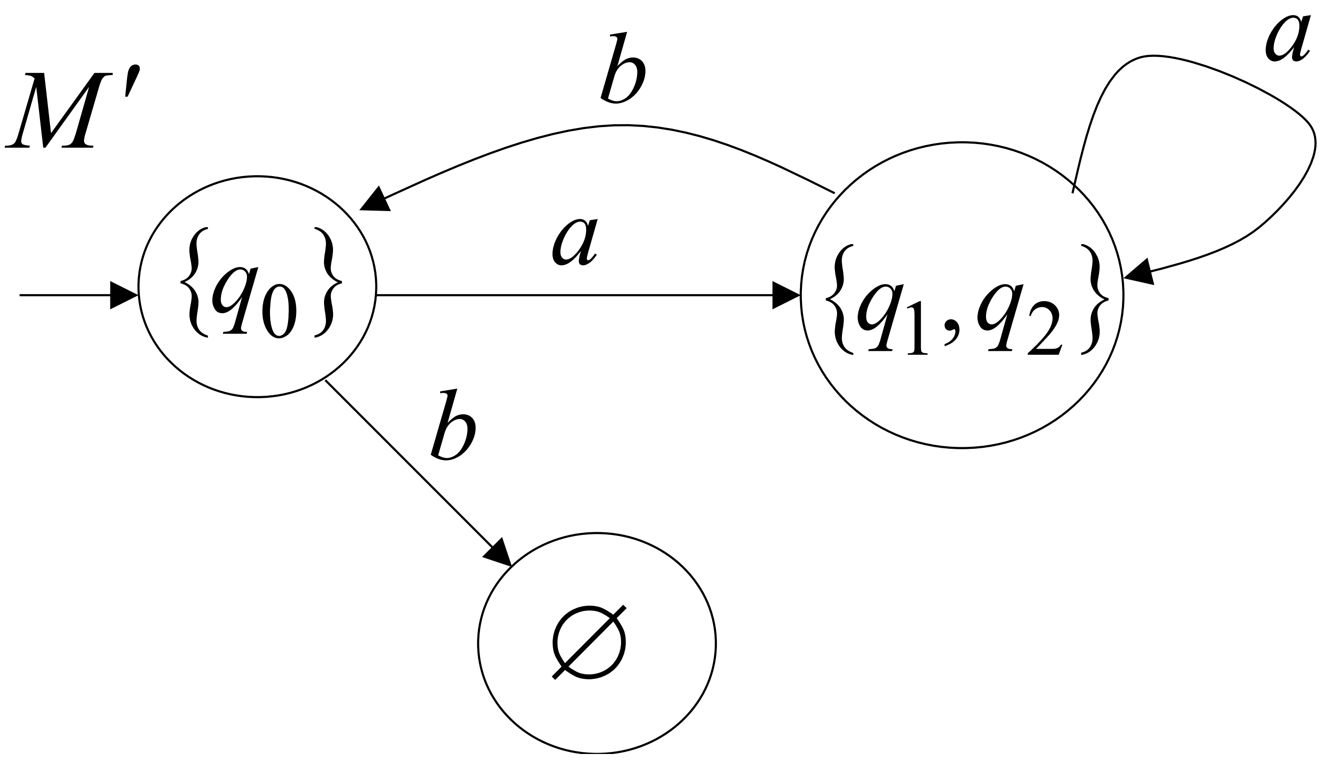


Convert NFA to DFA

NFA M

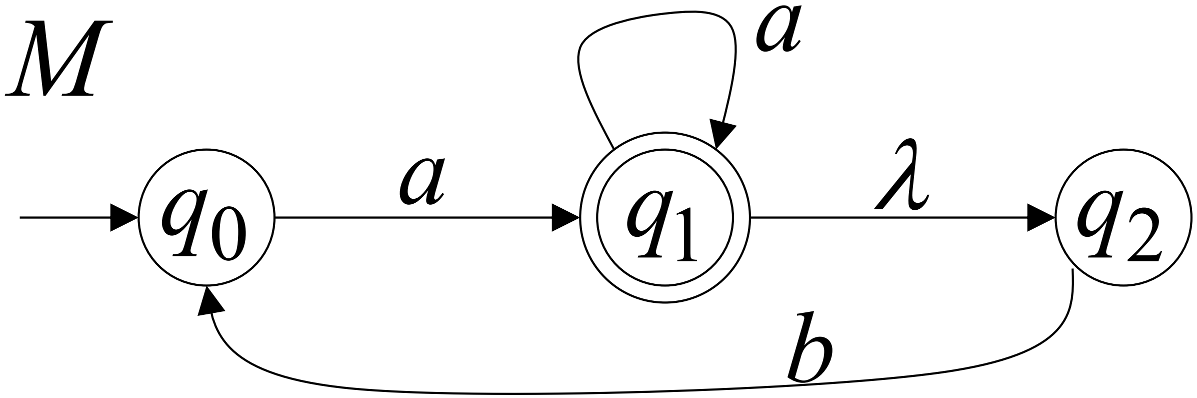


DFA M'

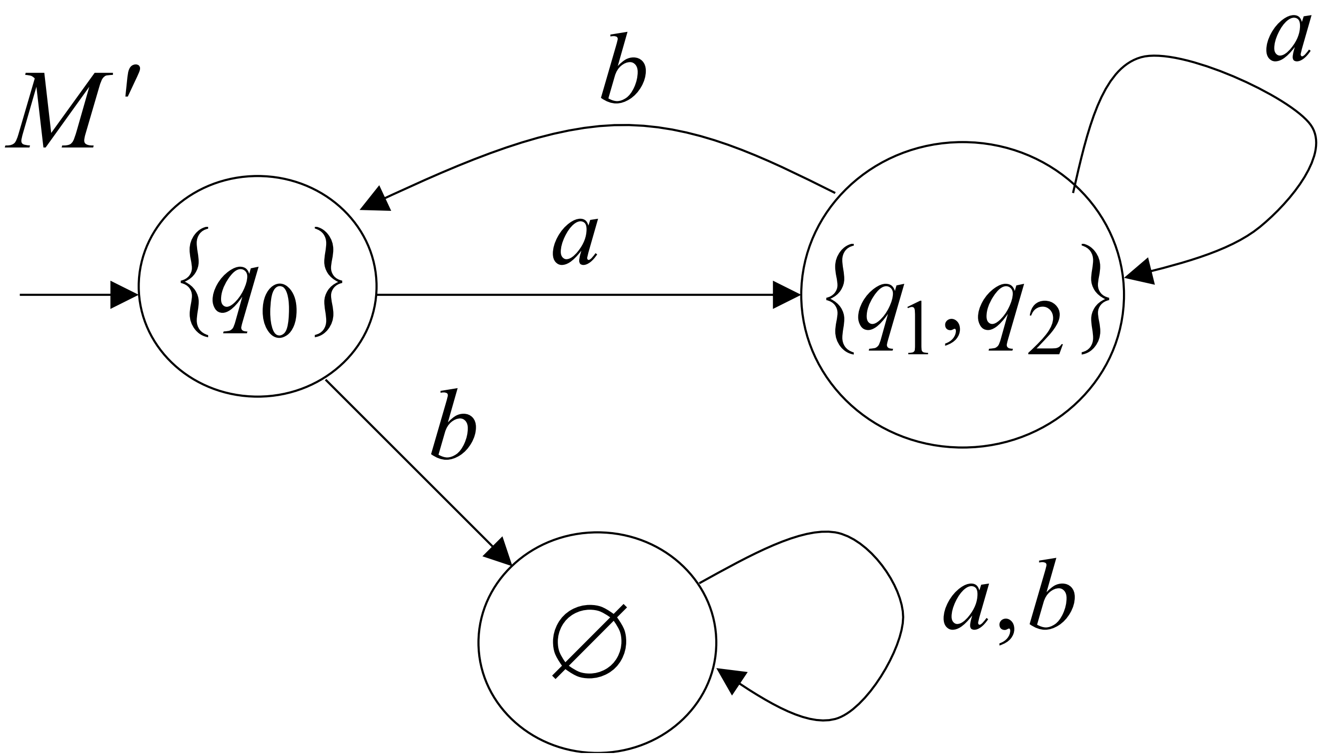


Convert NFA to DFA

NFA M

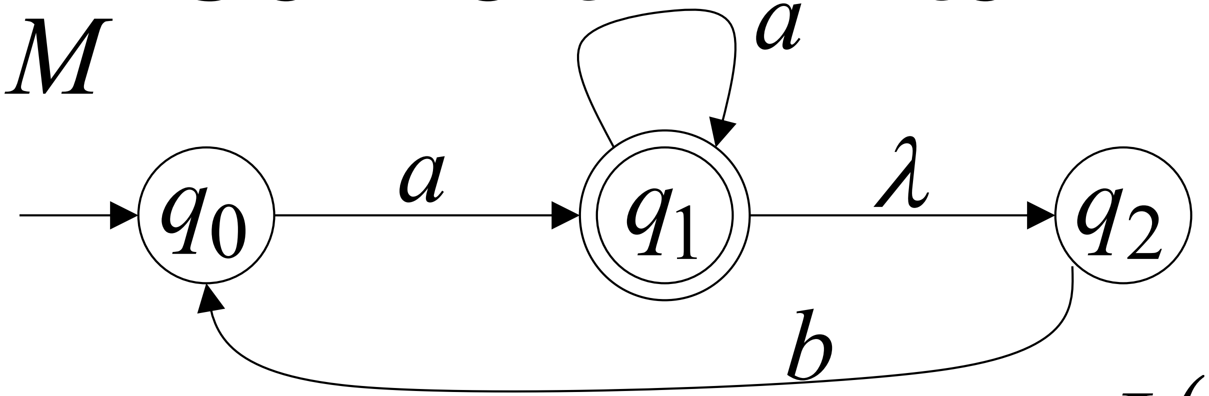


DFA M'



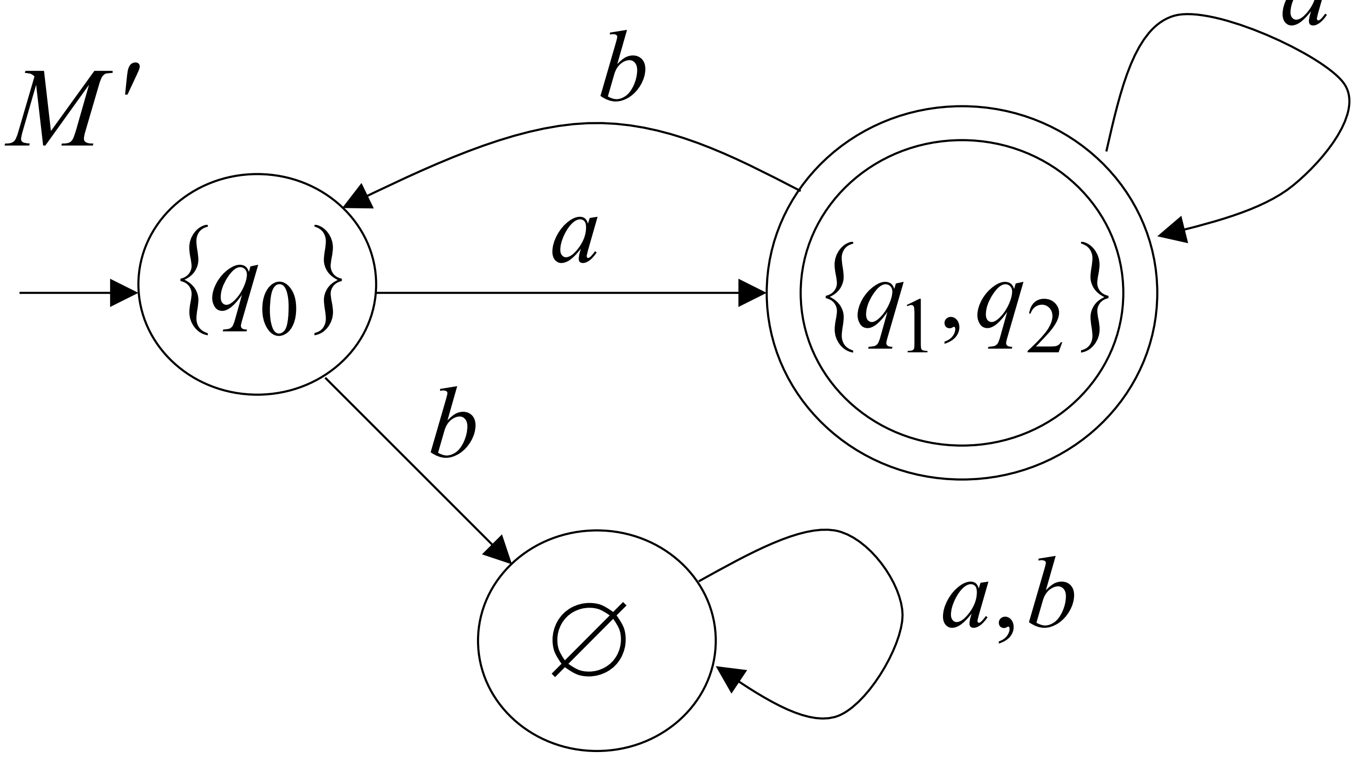
Convert NFA to DFA

NFA M



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

DFA M'



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

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

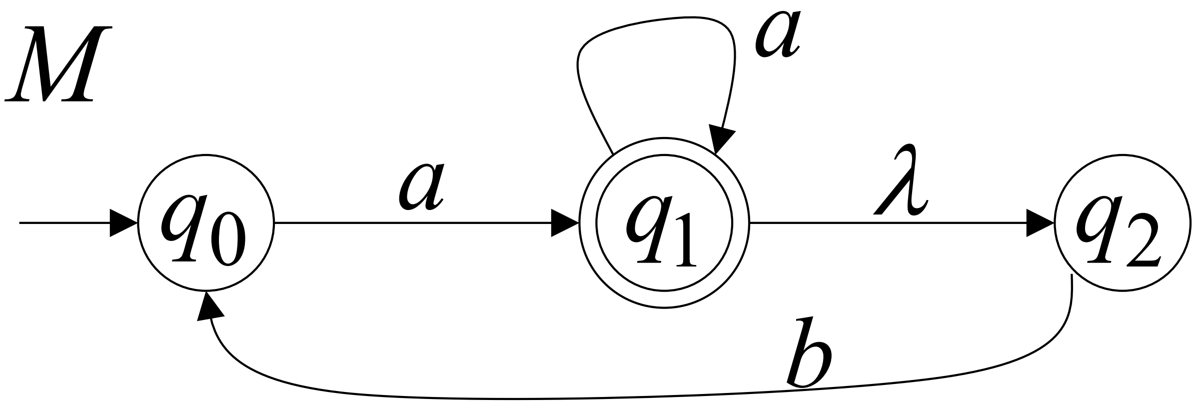
Procedure NFA to DFA

1. Initial state of NFA: q_0

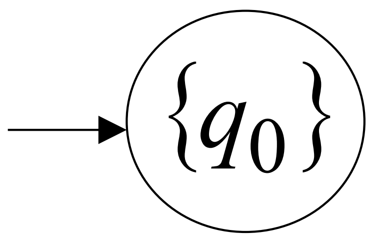
Initial state of DFA: $\{q_0\}$

Example 2.12

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

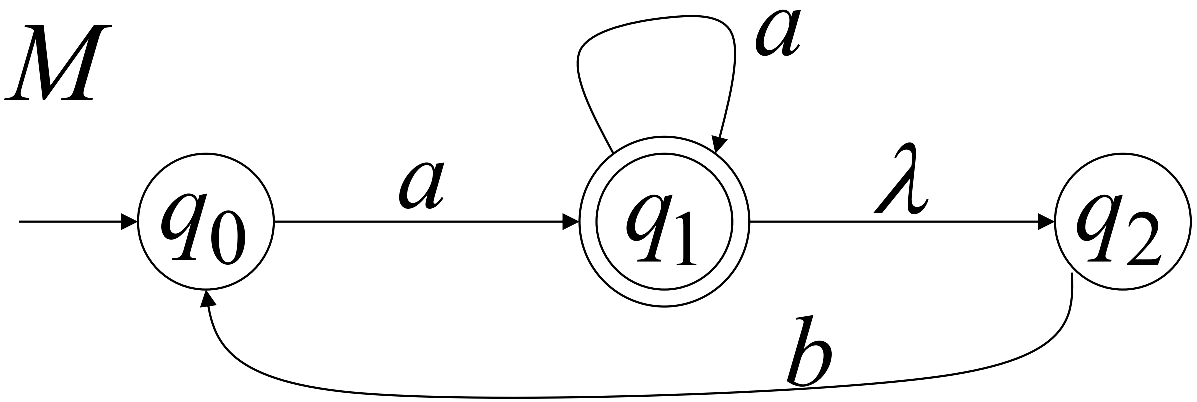
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to DFA

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

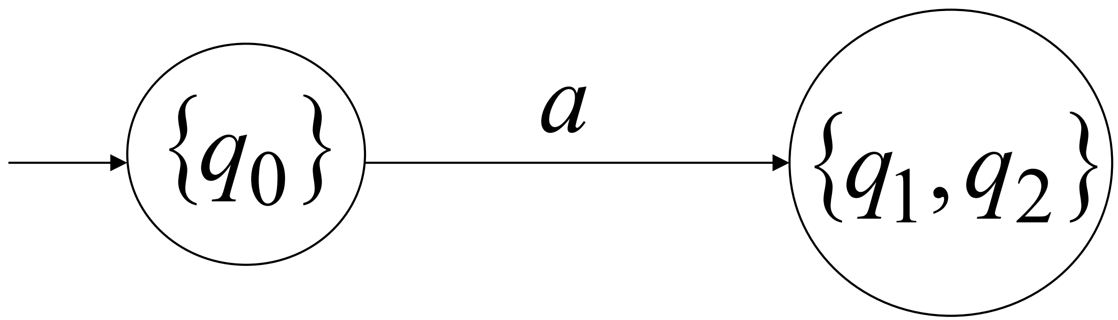
Example 2.12

NFA M



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

DFA M'



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

Procedure NFA to DFA

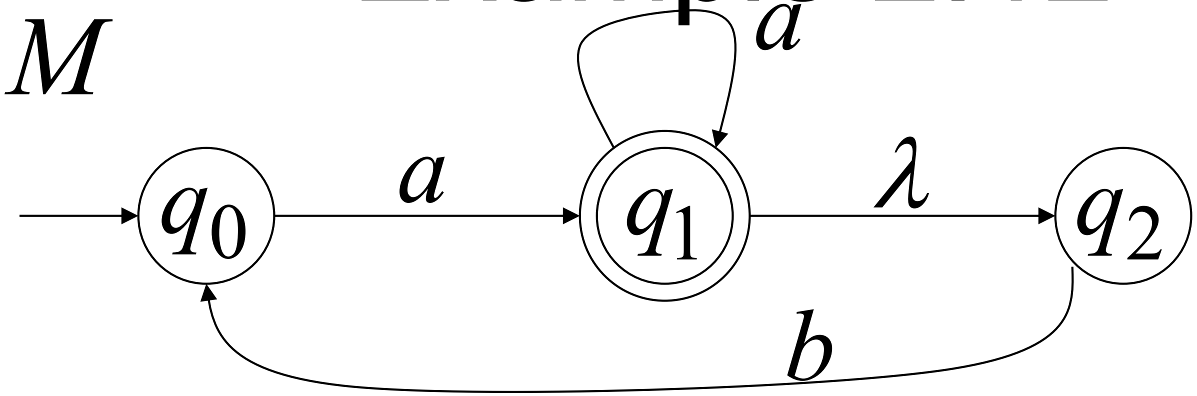
Repeat Step 2 for all letters in alphabet,

until

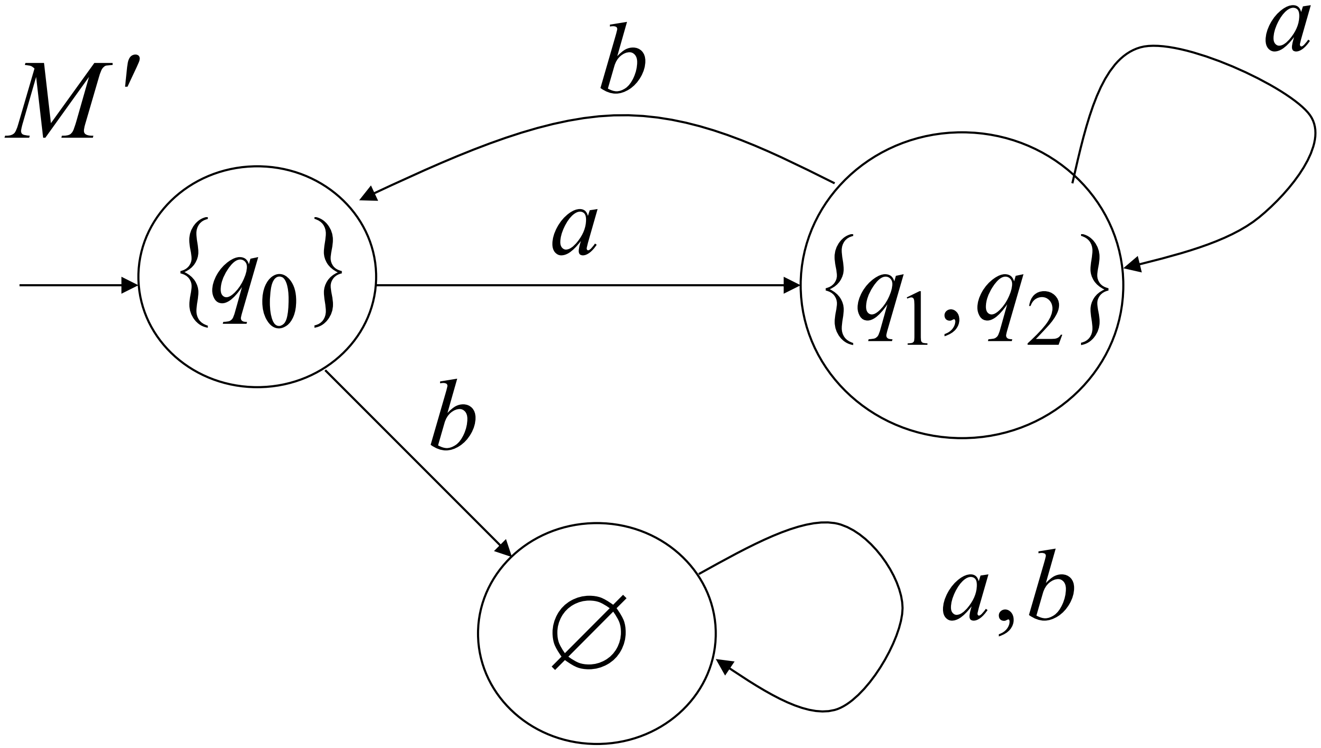
no more transitions can be added.

Example 2.12

NFA M



DFA M'



Procedure NFA to DFA

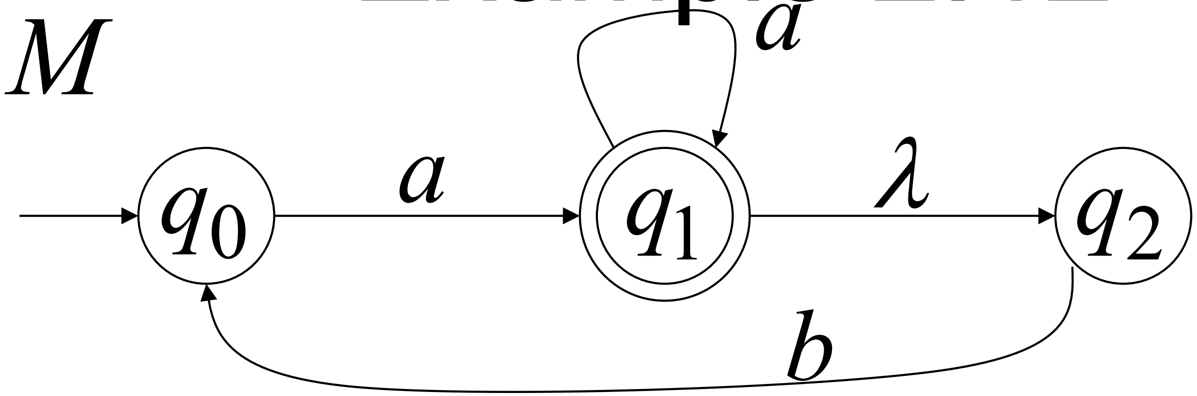
3. For any DFA state $\{q_i, q_j, \ldots, q_m\}$

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

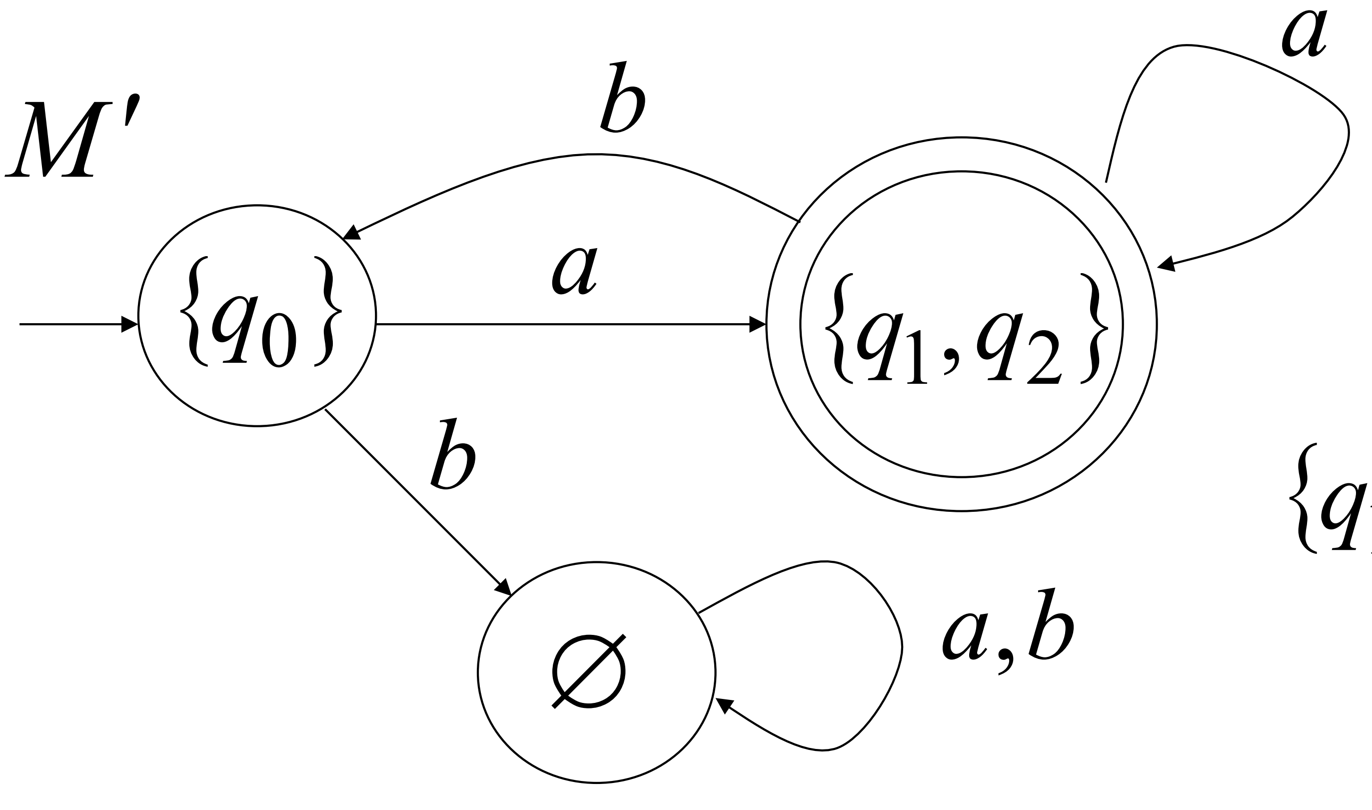
Example 2.12

NFA M



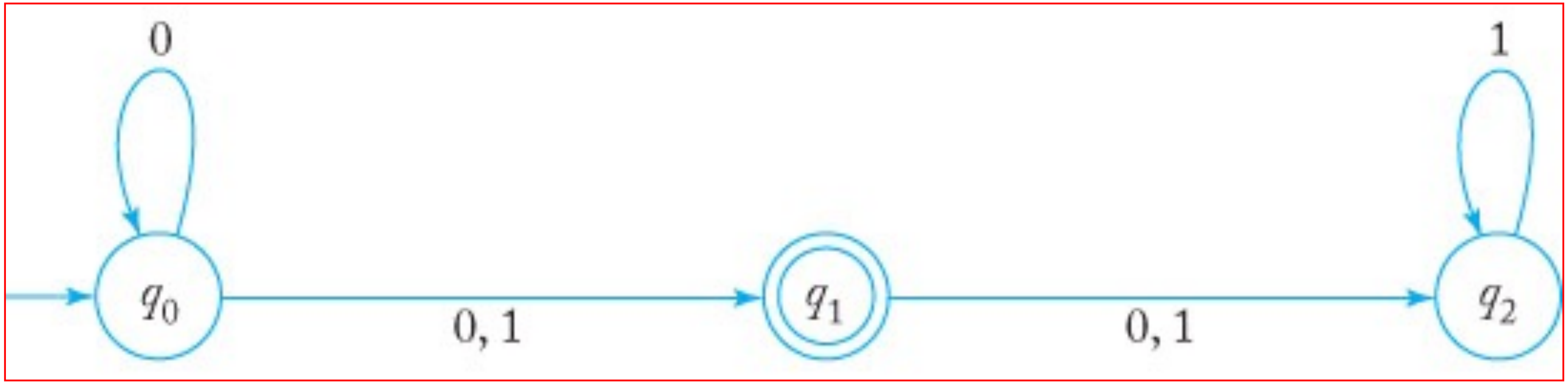
$q_1 \in F$

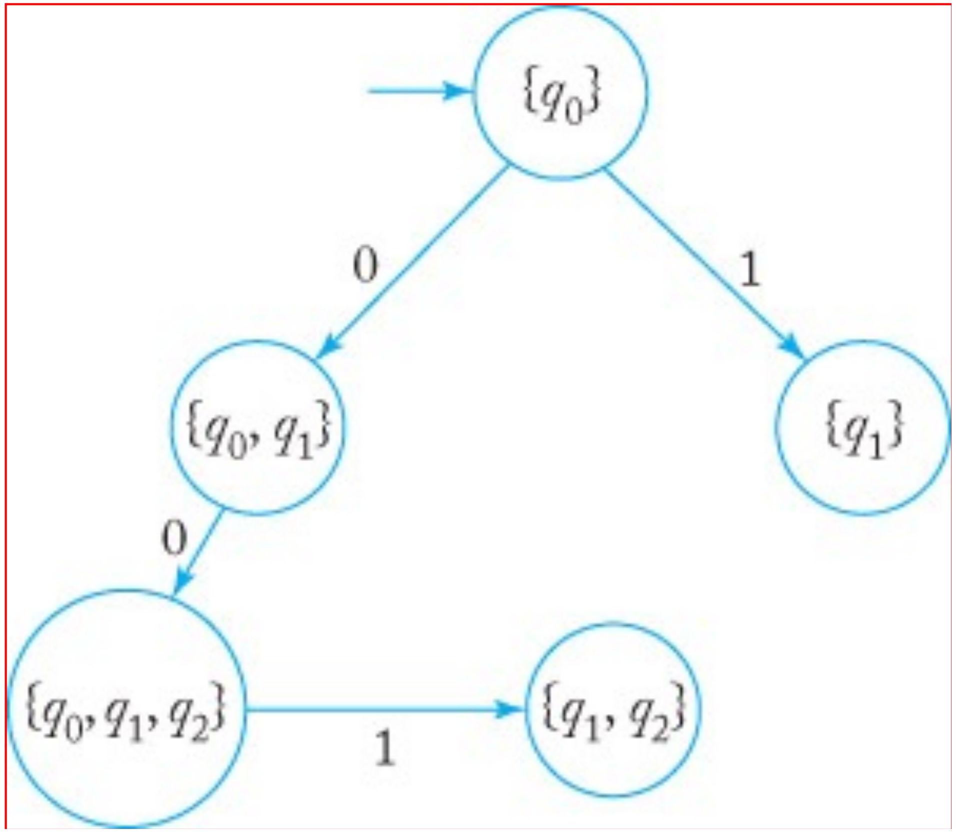
DFA M'



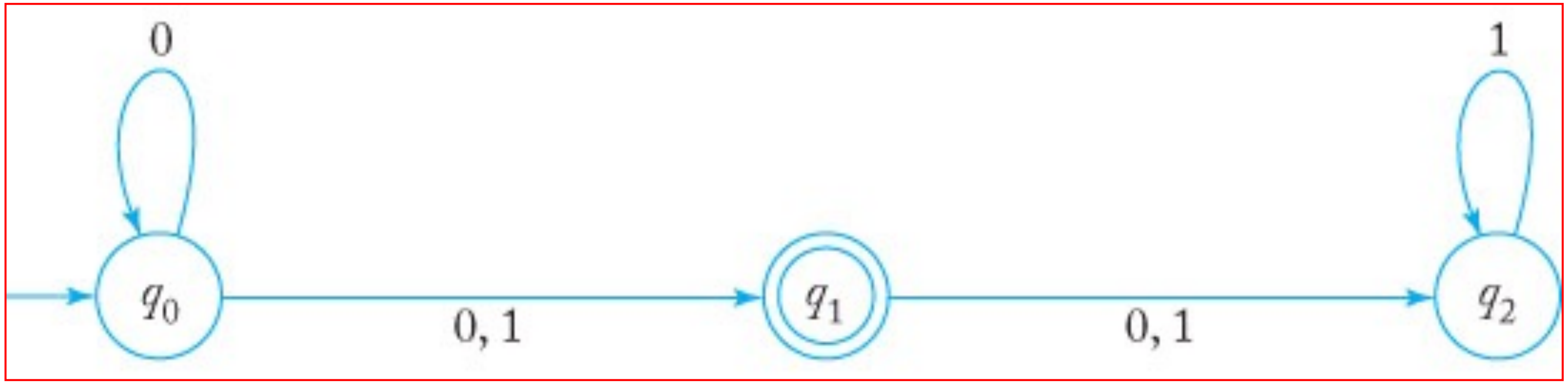
$\{q_1, q_2\} \in F'$

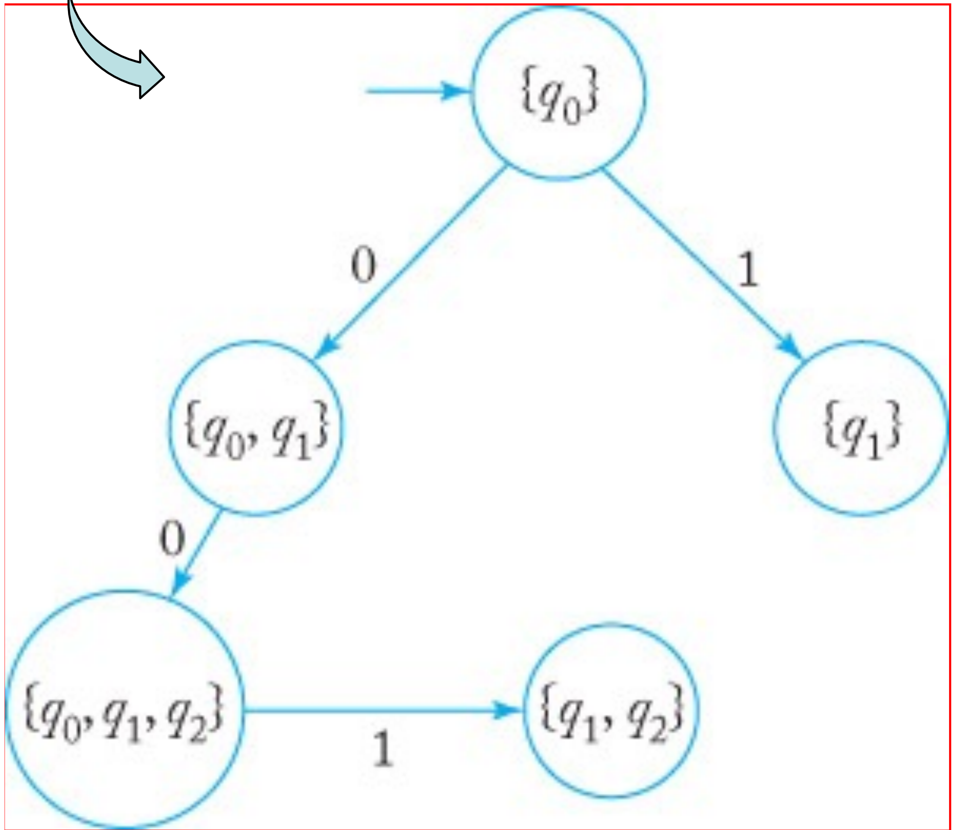
Example 2.13

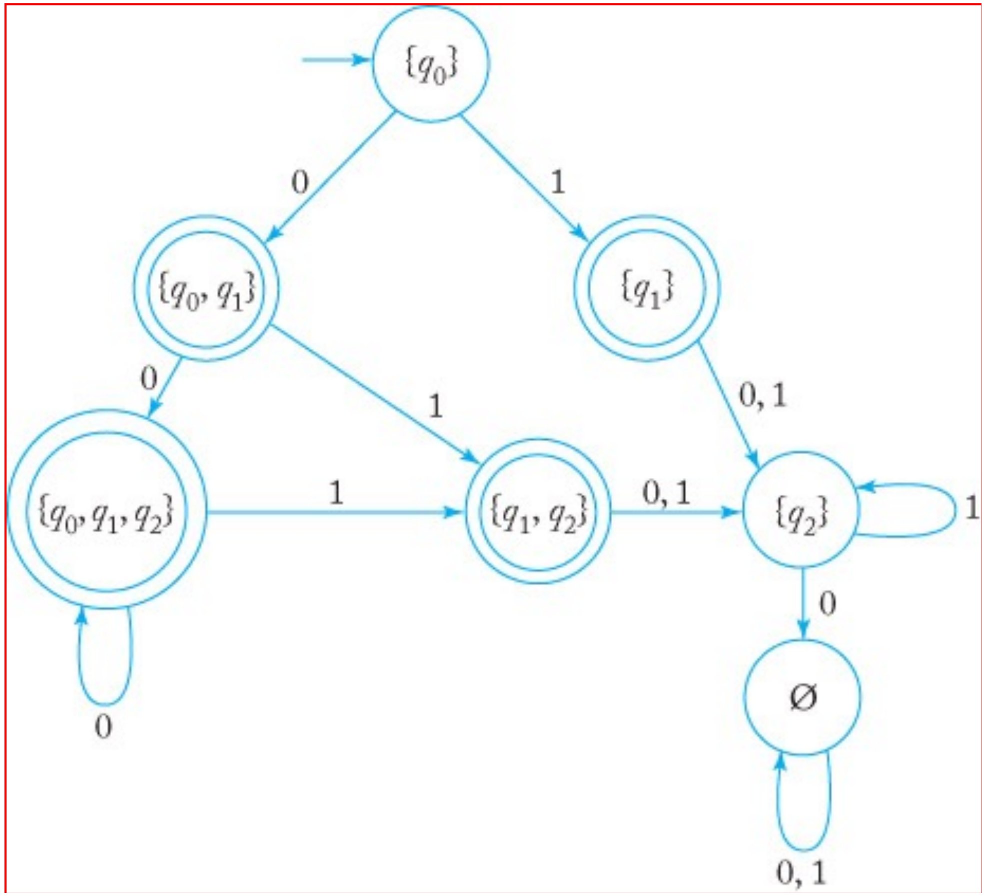




Example 2.13


0
1
q_0
0, 1
q_1
0, 1
q_2


$\{q_0\}$
0
1
$\{q_0, q_1\}$
$\{q_1\}$
0
$\{q_0, q_1, q_2\}$
1
$\{q_1, q_2\}$


$\{q_0\}$
0
1
$\{q_0, q_1\}$
$\{q_1\}$
0
1
0, 1
$\{q_0, q_1, q_2\}$
1
$\{q_1, q_2\}$
0, 1
$\{q_2\}$
1
0
0
$\varnothing$
0, 1

NFA M
DFA M'

Theorem 2.2

Take NFA M

Apply procedure to obtain DFA M'

Then M and M' are equivalent :

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

NFA M
DFA M'

Proof

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

$$\Updownarrow$$

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

NFA M
DFA M'

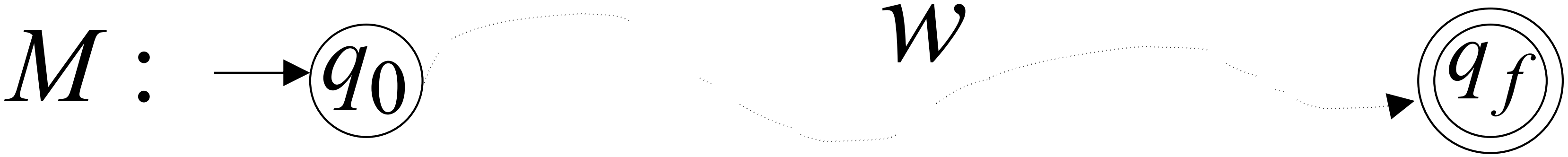
First we show: $L(M) \subseteq L(M')$

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

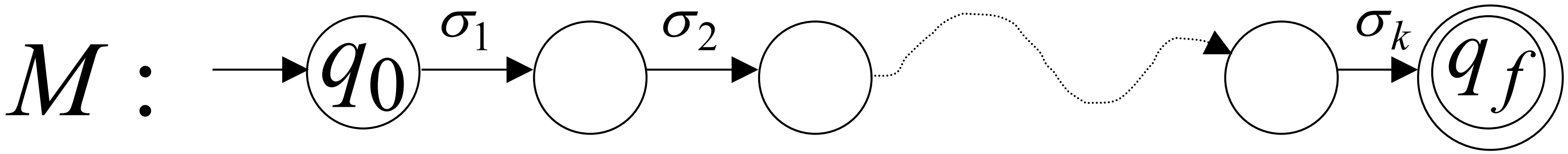
We will prove: $w \in L(M')$

NFA M
DFA M'

$$w \in L(M)$$



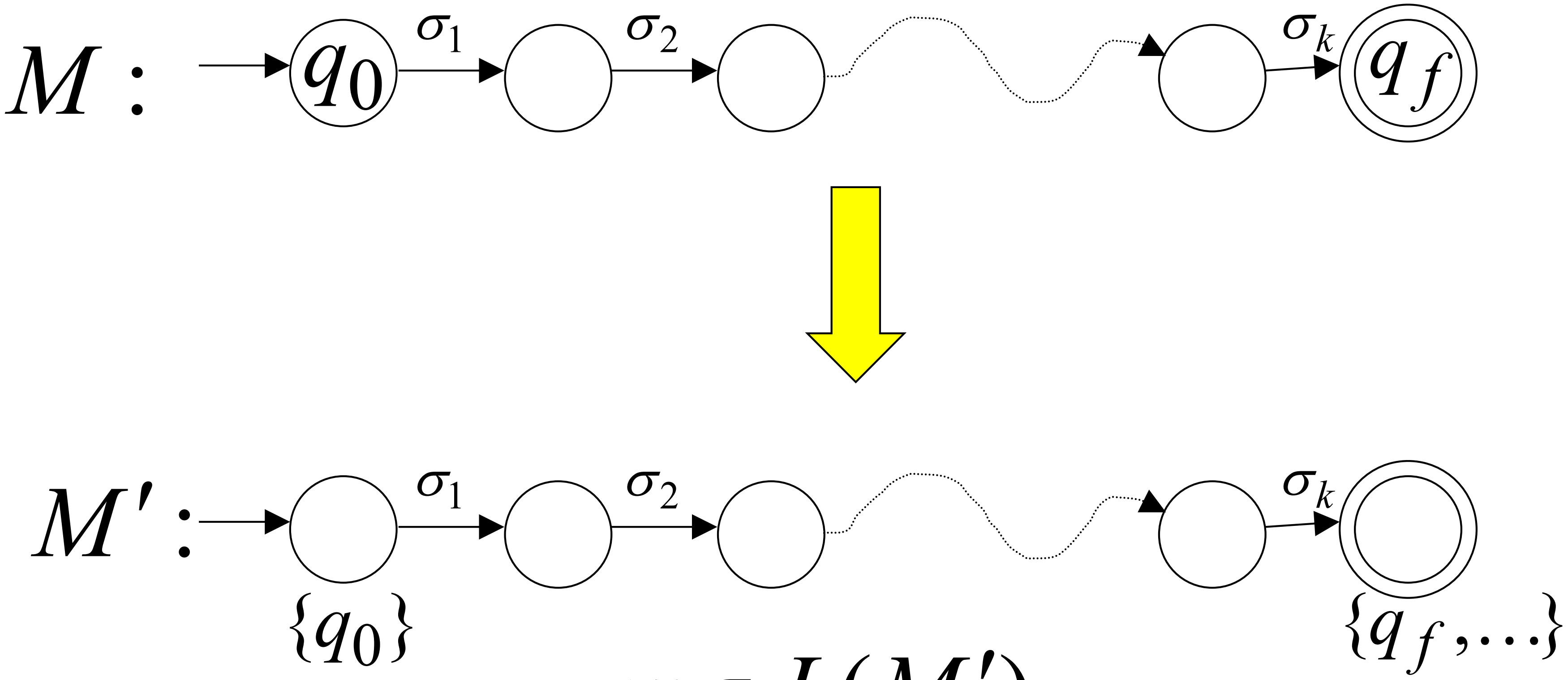
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



NFA M
DFA M'

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

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

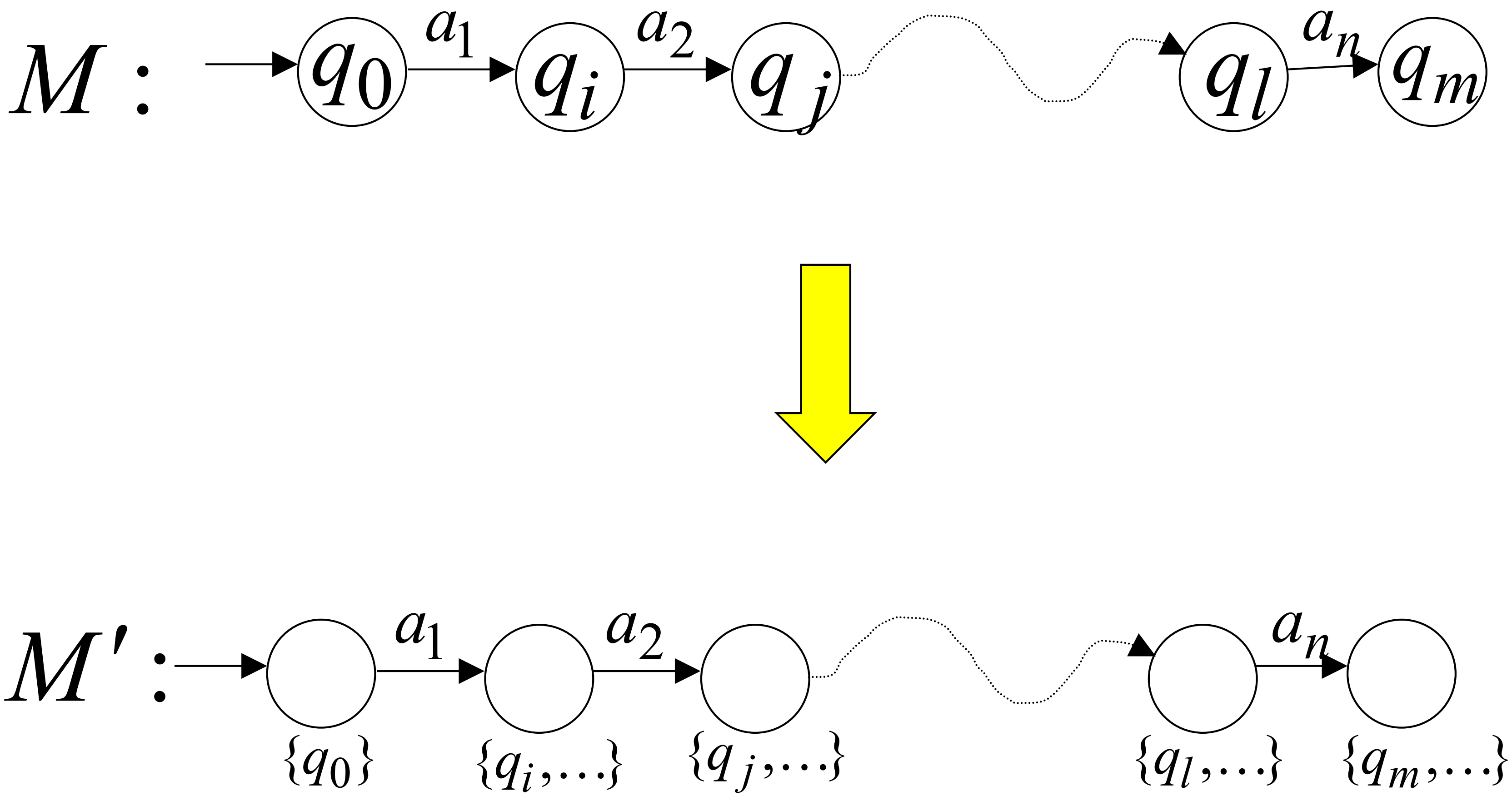


$$w \in L(M')$$

NFA M
DFA M'

More generally, we will show that if in M:

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

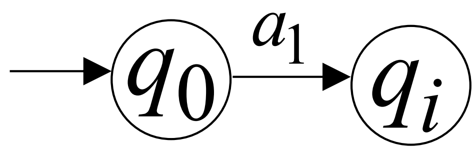


Proof by induction on $|v|$

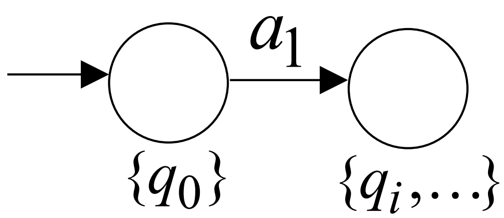
NFA M
DFA M'

Induction Basis: $v = a_1$

$M:$



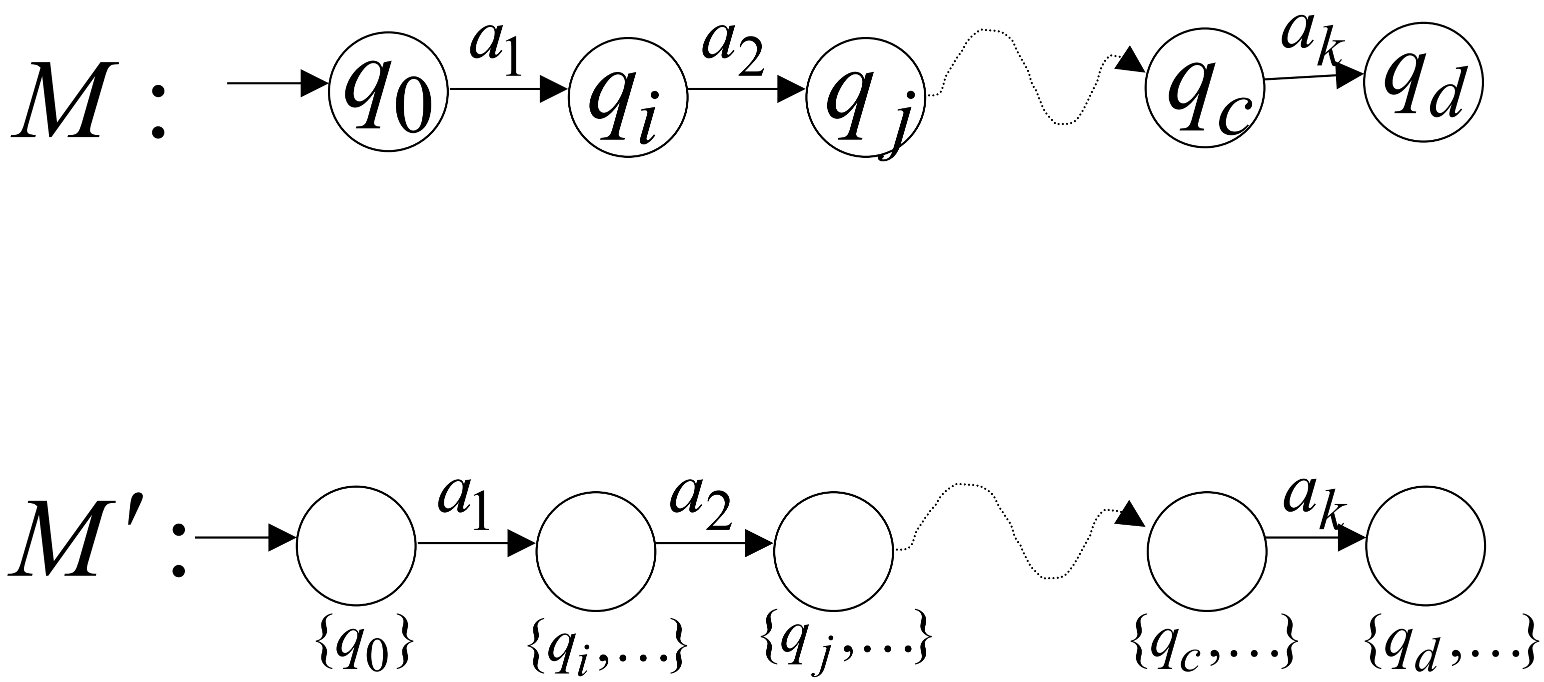
$M':$



NFA M
DFA M'

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

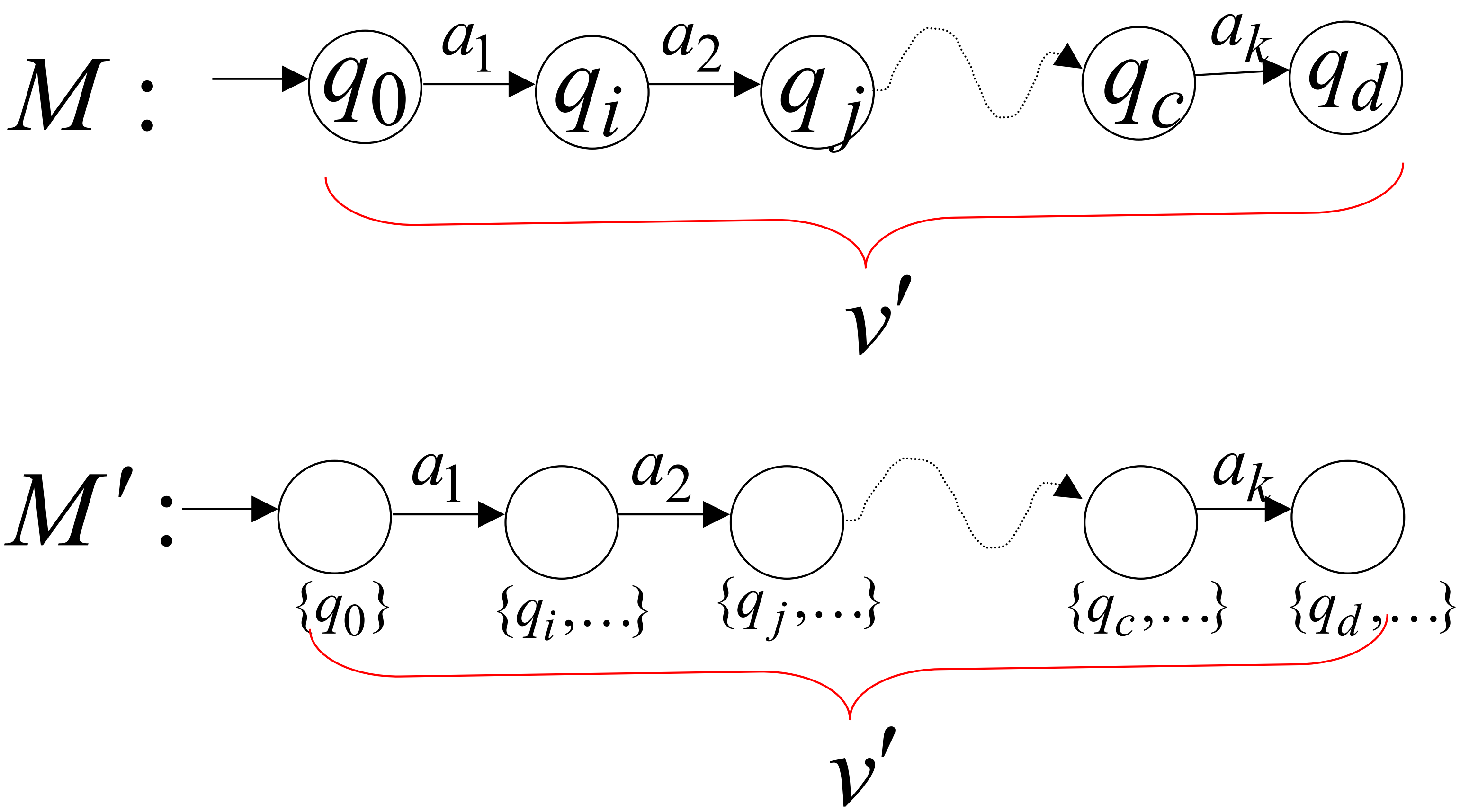
$$v = a_1 a_2 \cdots a_k$$



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

NFA M
DFA M'

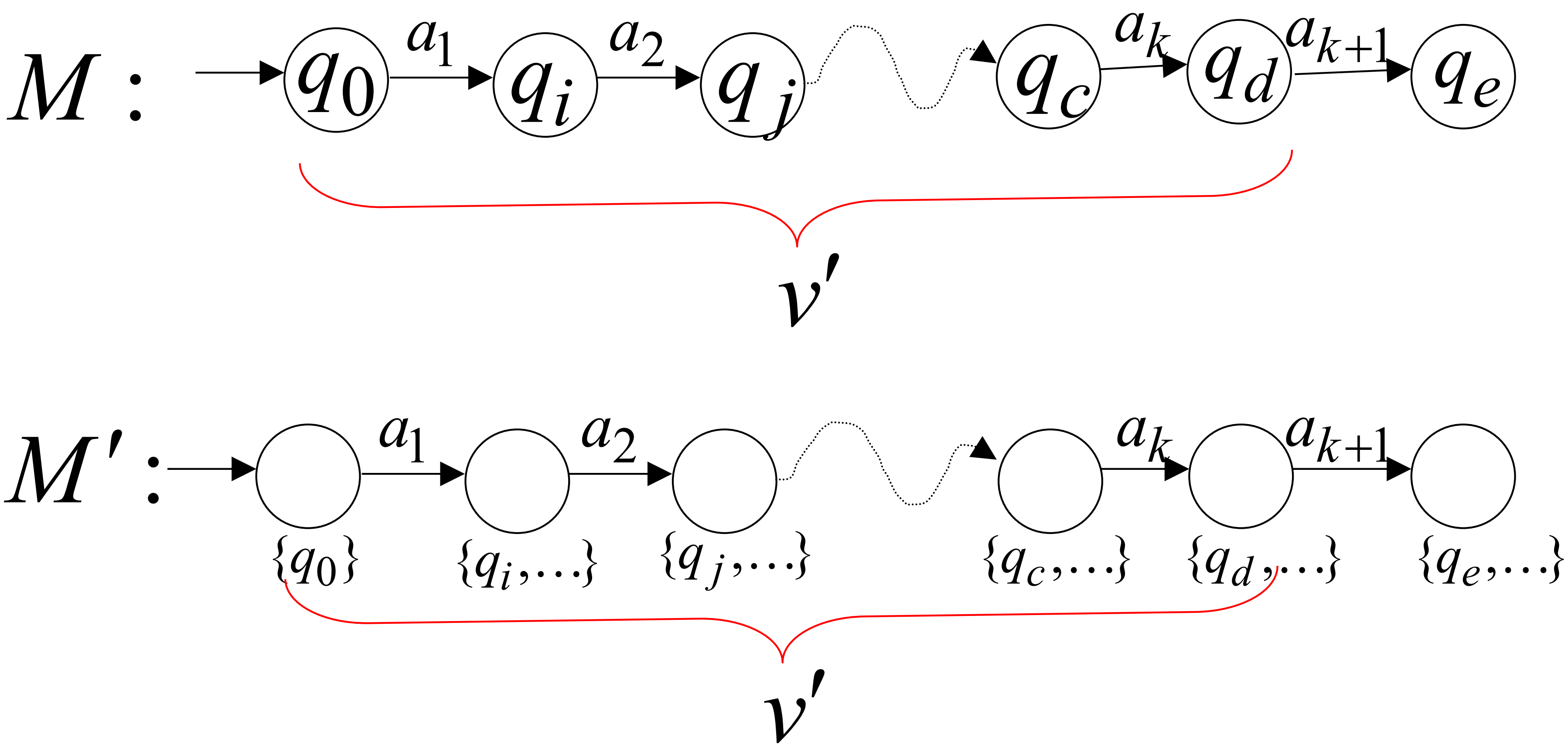
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$



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

NFA M
DFA M'

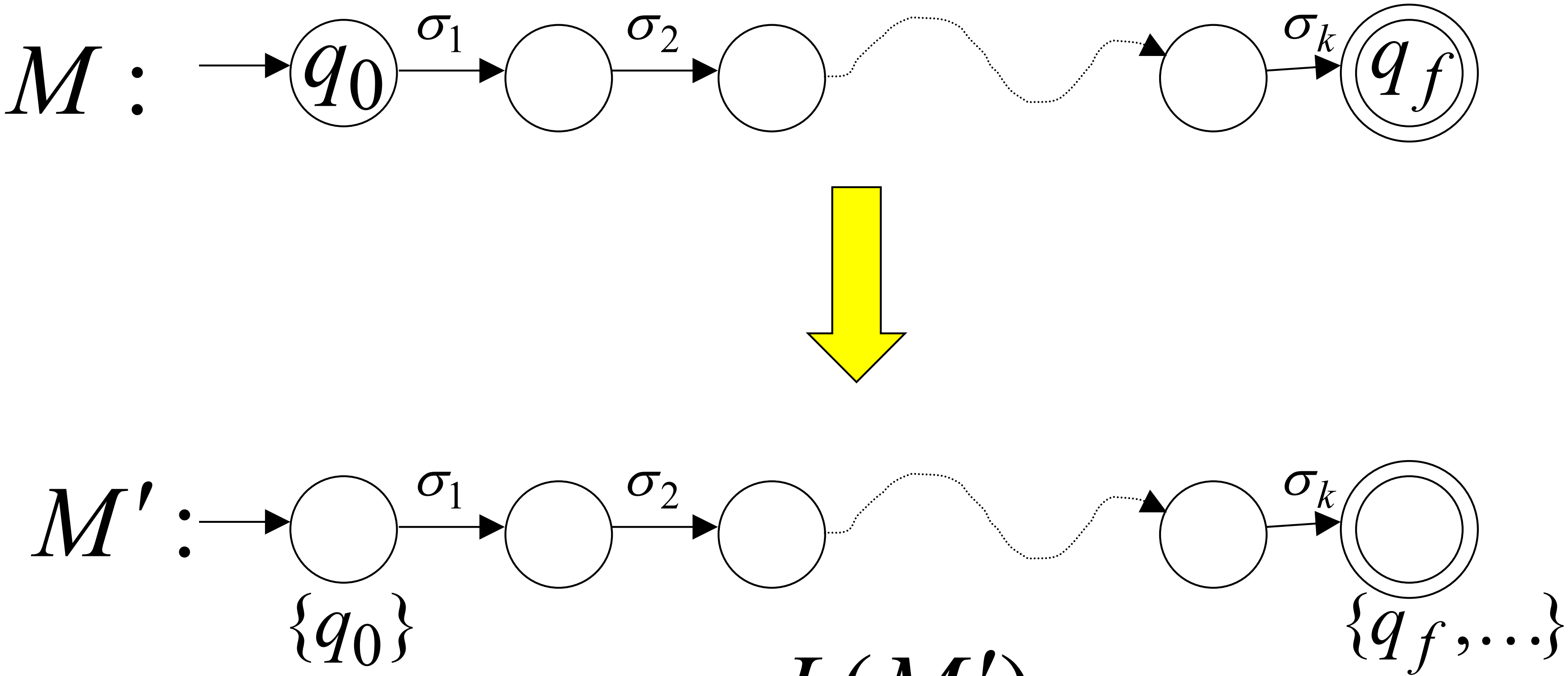
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$



NFA M
DFA M'

Therefore if $w \in L(M)$

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



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

NFA M
DFA M'

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

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

(proof is similar)

NFAs accept the Regular Languages

Exercise 2.3.7

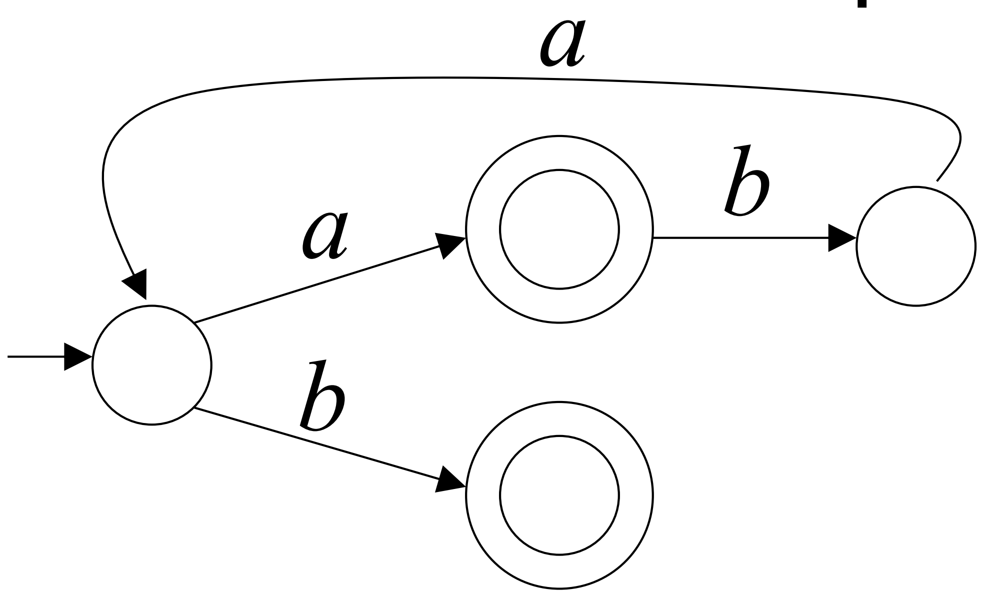
Any NFA can be converted

to an equivalent NFA

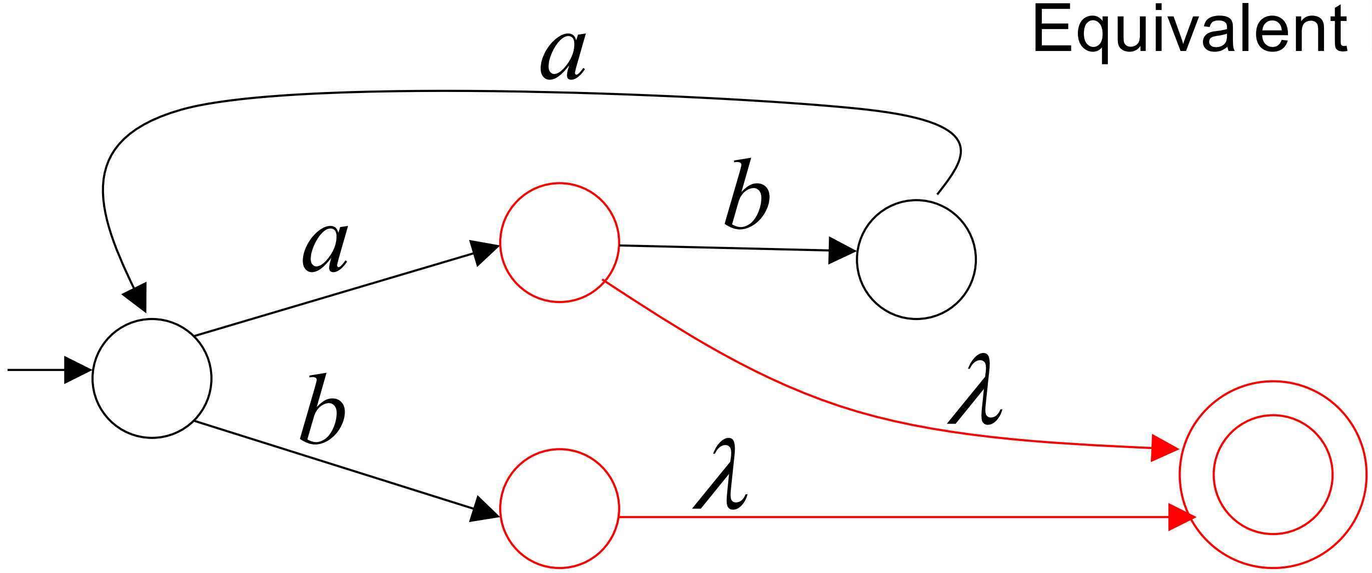
with a single final state

Example

NFA

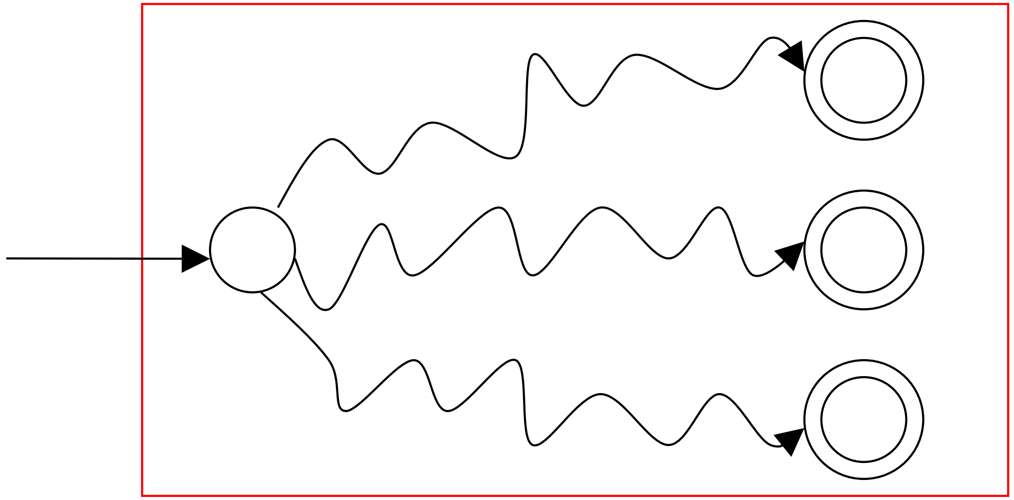


Equivalent NFA

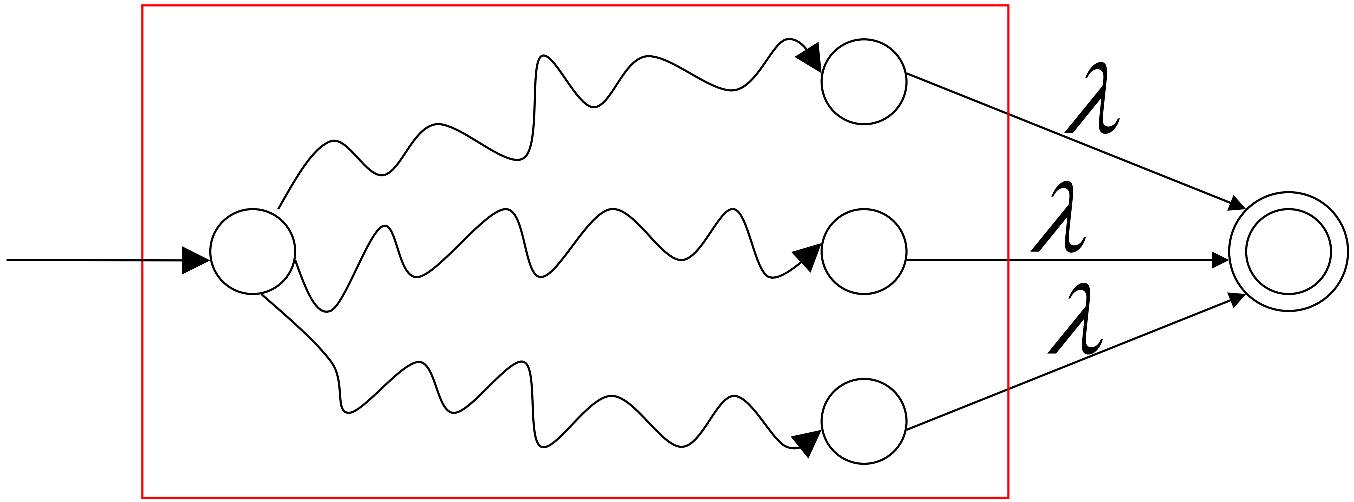


In General

NFA



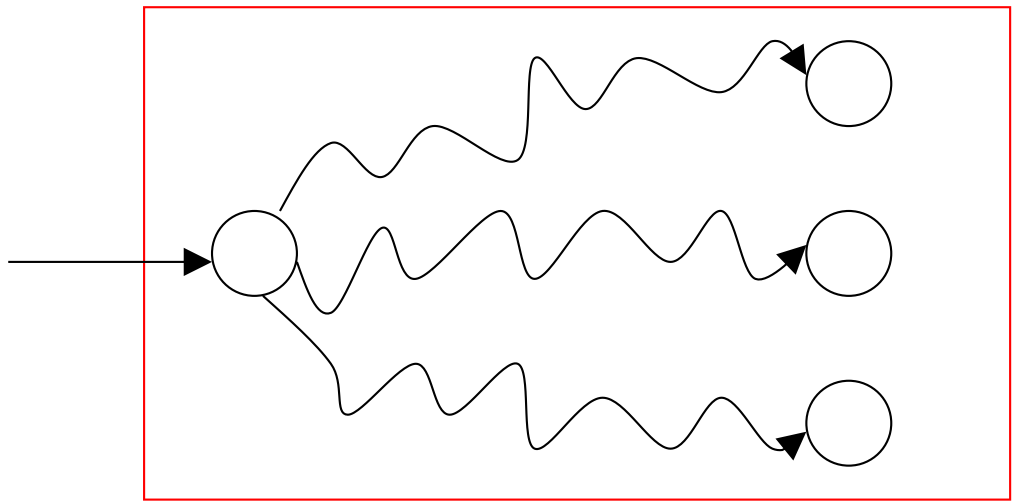
Equivalent NFA

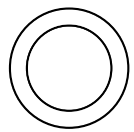


Single
final state

Extreme Case

NFA without final state (it accepts ϕ)



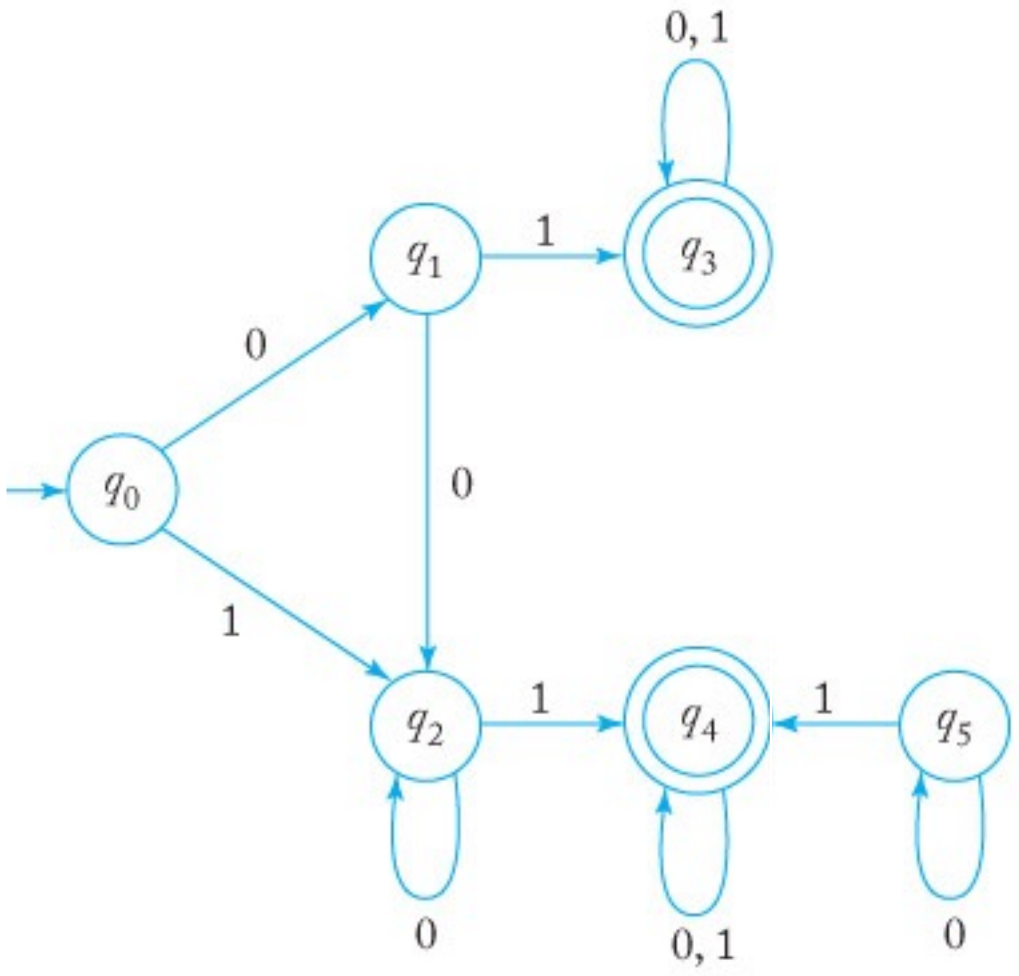


Add a final state
Without transitions

Outline

1. Deterministic Finite Accepters (DFA)
2. Nondeterministic Finite Accepters (NFA)
3. Equivalence of DFA and NFA
4. Reduction of the Number of States in FA*

Example 2.14



(a)

Inaccessible state

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_2$

Definition 2.8

Two steps p and q of a DFA are called indistinguishable if

$$\delta^*(p,w) \in F \text{ implies } \delta^*(q,w) \in F,$$

And

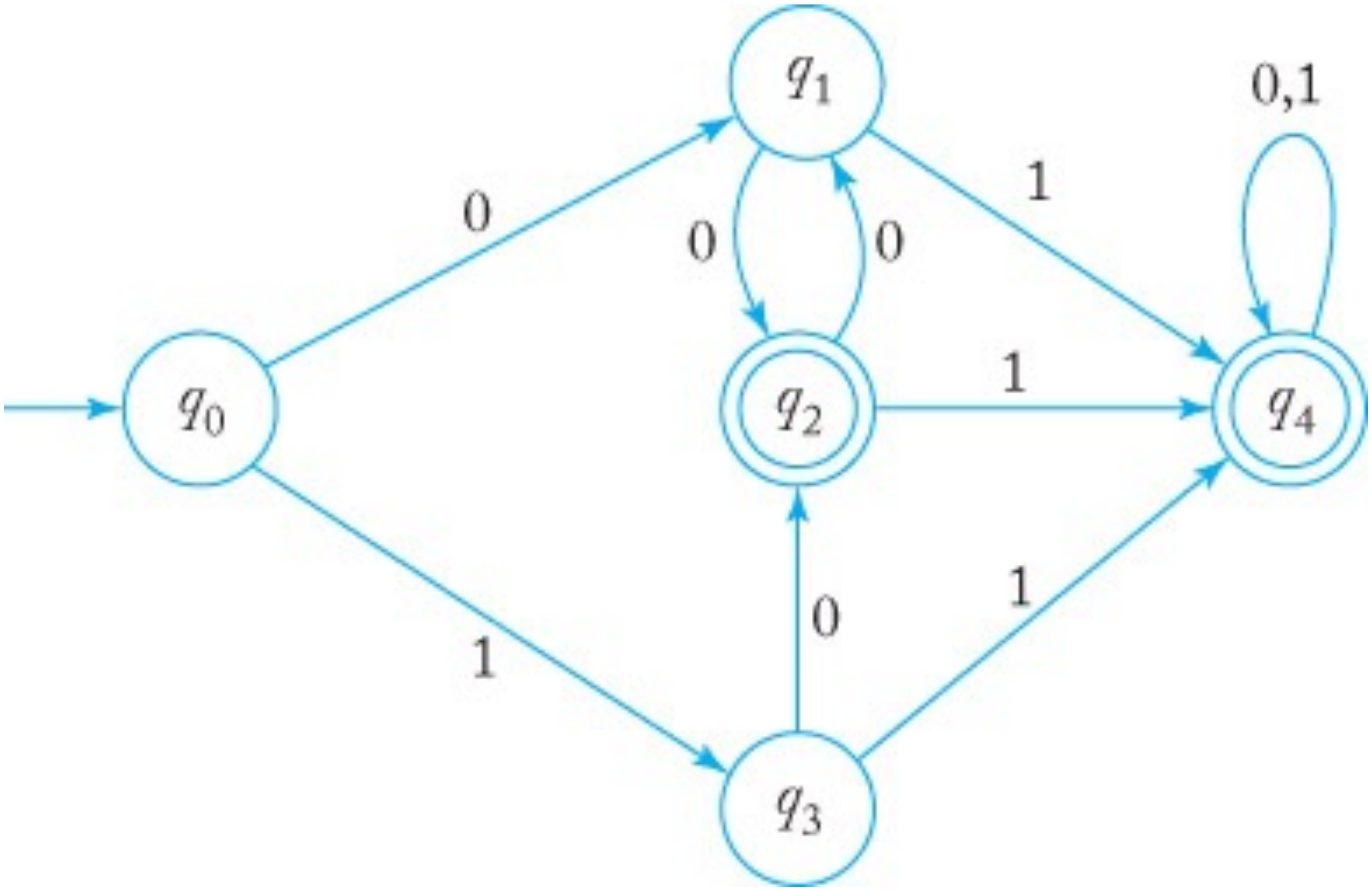
$$\delta^*(p,w) \notin F \text{ implies } \delta^*(q,w) \notin F,$$

For all w ϵ Σ*. If on the other hand, there exists some string w ϵ Σ* such that

$$\delta^*(p,w) \in F \text{ implies } \delta^*(q,w) \notin F,$$

Or vice versa, then the state p and q are said to be distinguishable by a string w.

Example 2.15

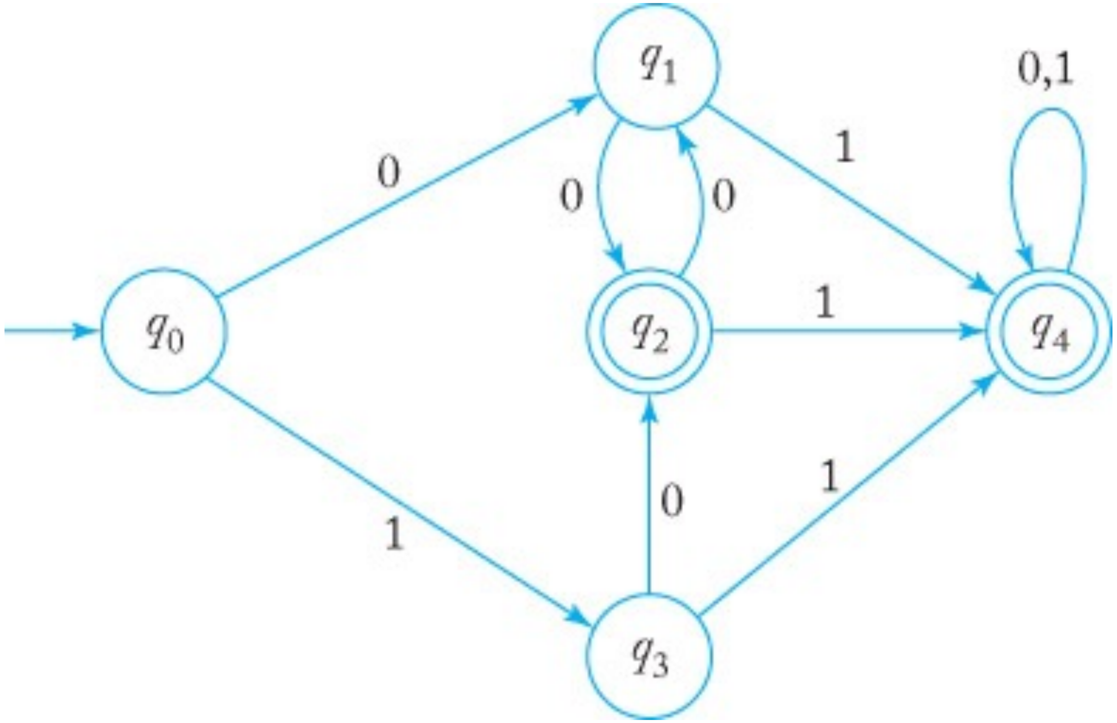


$\{q_0, q_1, q_3\} \notin F$

$\{q_2, q_4\} \in F$

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

Thus q_0 and q_1 are distinguishable
$\{q_0, q_1, q_3\} \rightarrow \{q_0\}\ \{q_1, q_3\}$

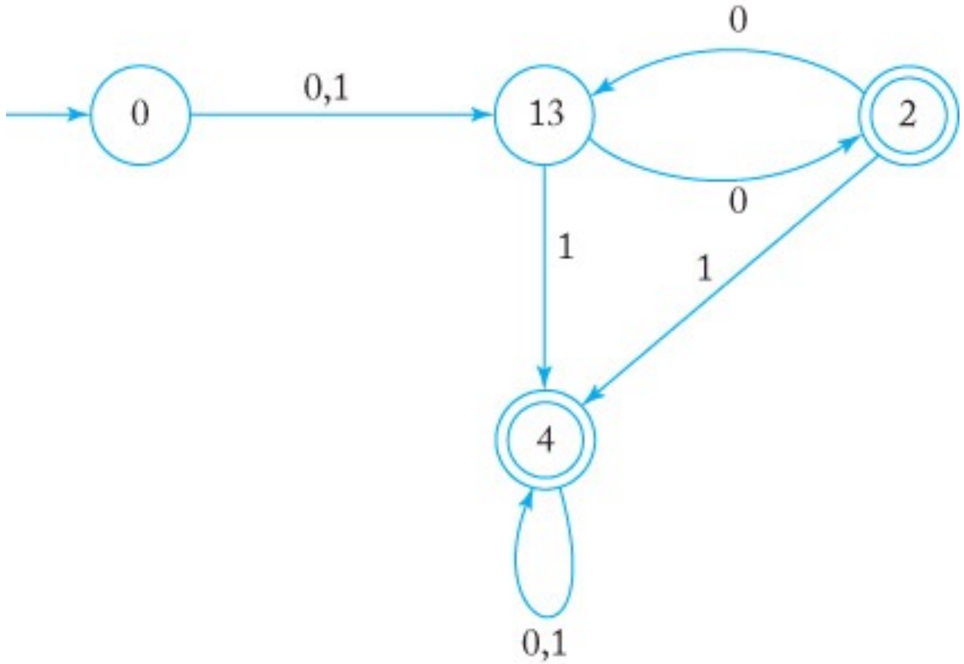


$\delta(q_0, 0) = q_1$
$\delta(q_1, 0) = q_2$

Thus q_0 and q_1 are distinguishable
$\{q_0, q_1, q_3\} \rightarrow$
$\{q_0\}\ \{q_1, q_3\}$

$\delta(q_2, 0) = q_1$
$\delta(q_4, 0) = q_4$

Thus q_2 and q_4 are distinguishable
$\{q_2, q_4\} \rightarrow$
$\{q_2\}\ \{q_4\}$



Questions?