Finite Automata

Formal Languages and Abstract Machines

Week 02

Baris E. Suzek, PhD

Slides are adopted from Dr. Costas Busch

Outline

- Last week
- Introduction - Finite Automata, Types
- Finite Acceptors - Deterministic
- Regular Language
- Finite Acceptors - Nondeterministic

Alphabet and String

- Alphabet: Set of letters e.g. (sigma) $\Sigma = \{a, b\}$

- String: Sequence of letters

$$a \qquad u = ab$$

$$ab \qquad v = bbbaaa$$

$$abba \qquad w = abba$$

$$baba$$

$$aaabbbaabab$$

Language and String

- A language is a set of strings
  - Language of zoo: “cat”, “dog”, “zebra”, …
  - Defined over an alphabet:

$$\Sigma = \{a, b, c, \ldots, z\}$$

Empty String (lambda) λ

- A string with no letters:

$$|\lambda| = 0$$

- Observations:

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = abba$$

Language Example

$$L = \{a^n b^n : n \geq 0\}$$

- An infinite language

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad abb \notin L$$

Language Operations

- The usual set operations

$$\{a,ab,aaaa\}\cup\{bb,ab\}=\{a,ab,bb,aaaa\}$$

$$\{a,ab,aaaa\}\cap\{bb,ab\}=\{ab\}$$

$$\{a,ab,aaaa\}-\{bb,ab\}=\{a,aaaa\}$$

$$\overline{L}=\Sigma^*-L$$

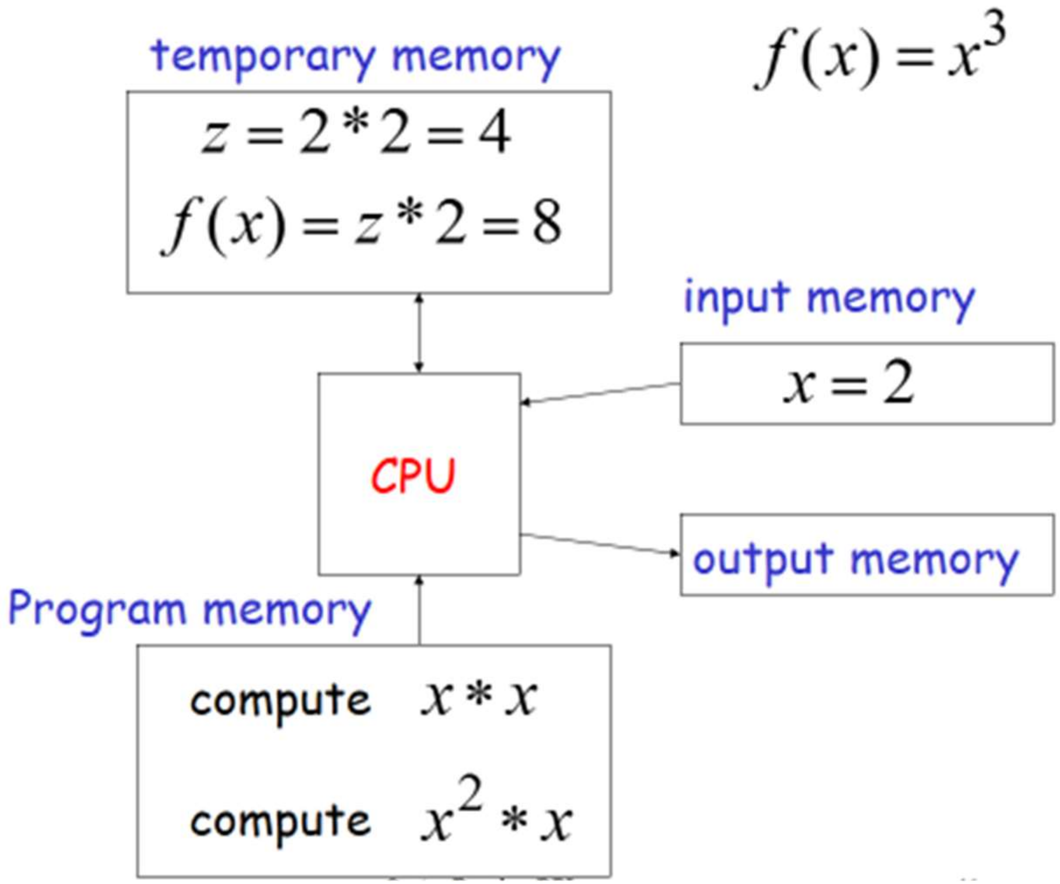
- Complement:

$$\overline{\{a,ba\}}=\{\lambda,b,aa,ab,bb,aaa,\ldots\}$$

Outline

- Last week
- Introduction - Finite Automata, Types
- Finite Acceptors - Deterministic
- Regular Language
- Finite Acceptors - Nondeterministic

Computation – An Abstraction

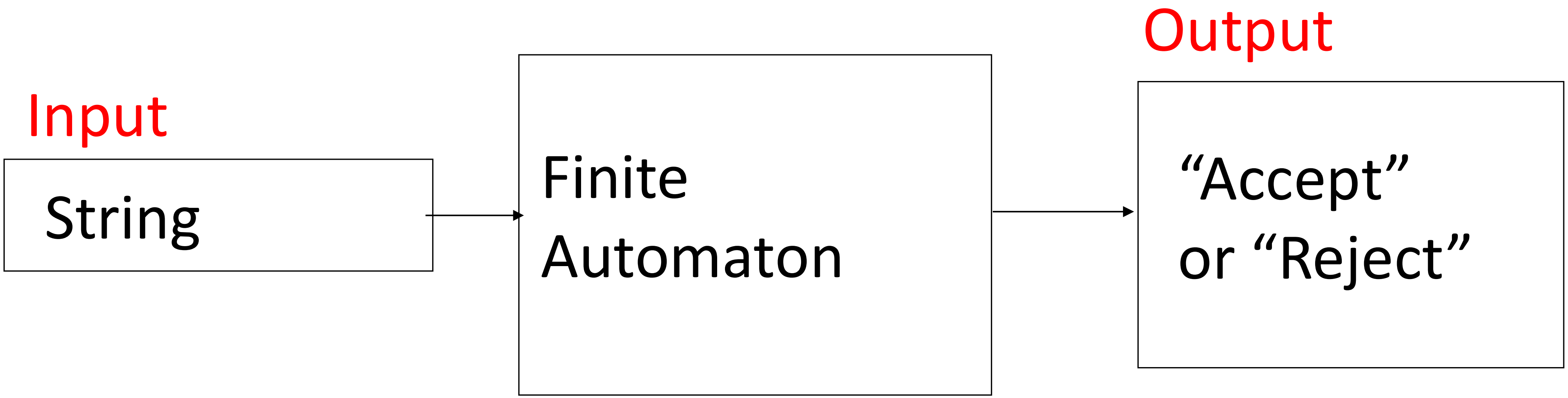


Finite Automaton

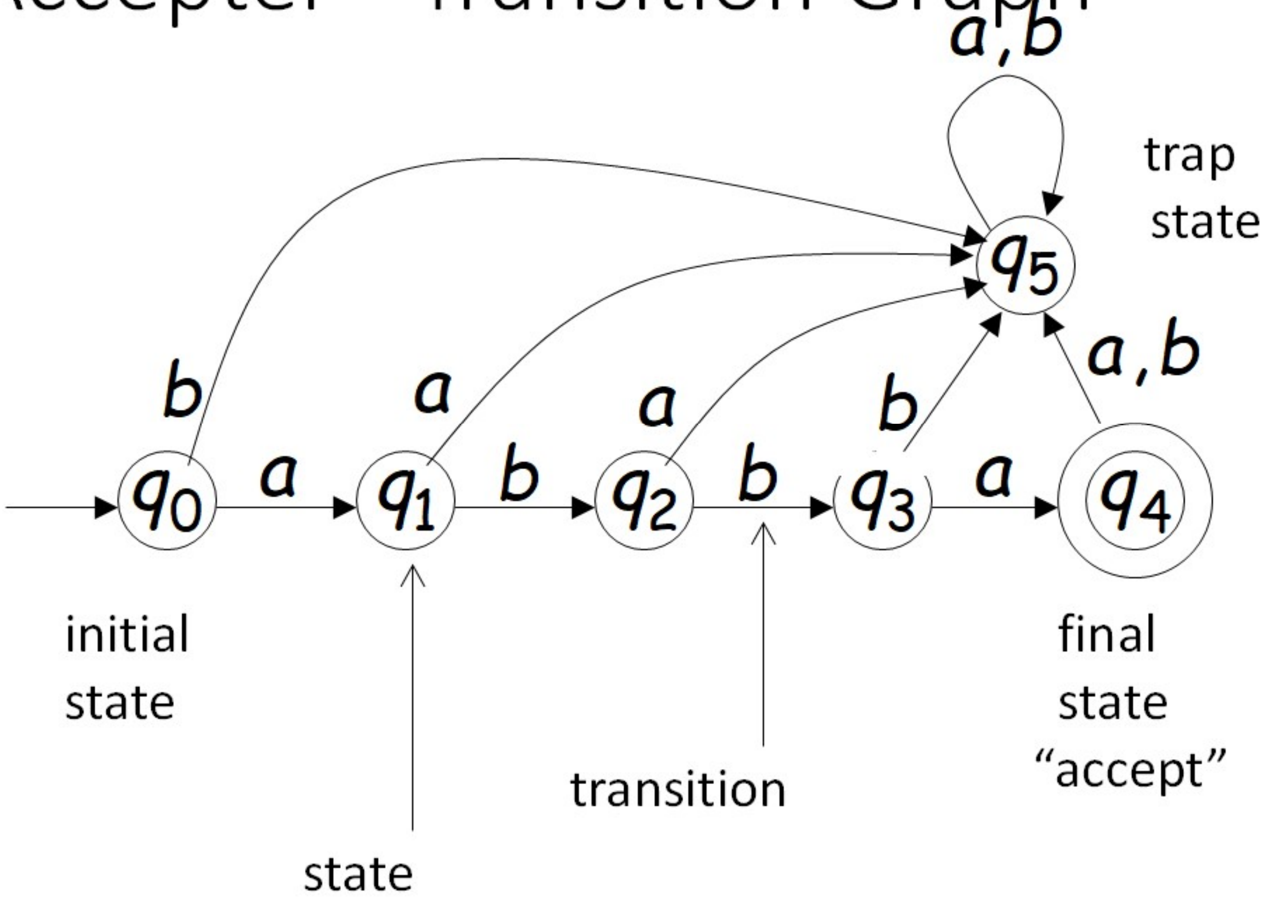
<u>No temporary memory</u>

Finite amount of information is retained through the state machine is in

Finite Automata Type: Finite Acceptor



Finite Accepter - Transition Graph

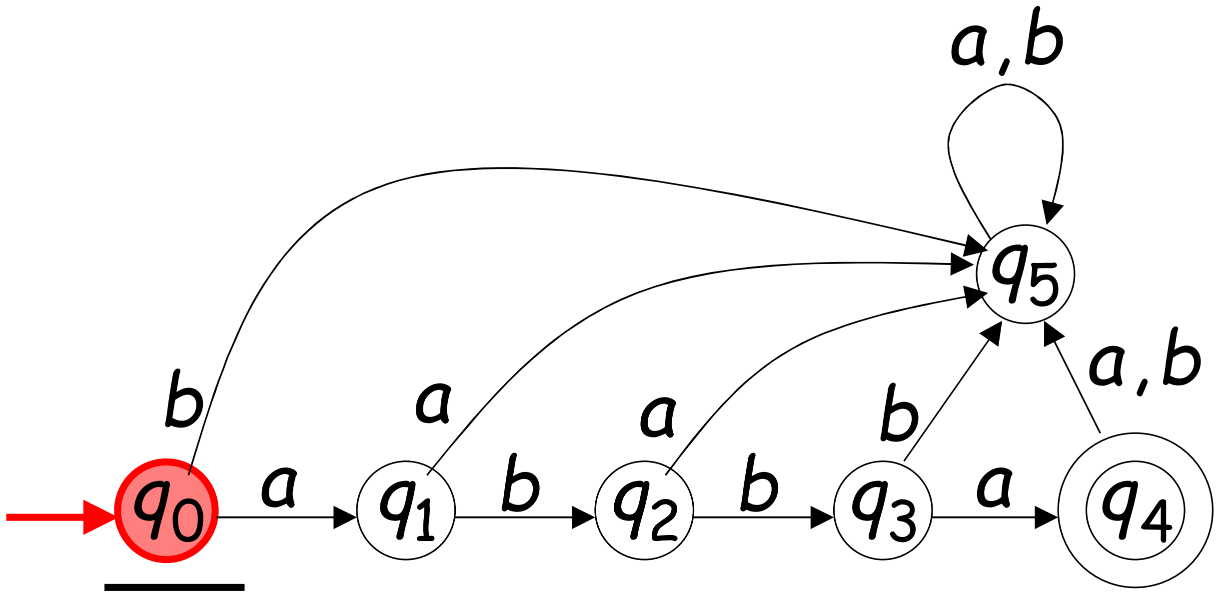


Deterministic FA: produces a unique run of the automaton for each input string

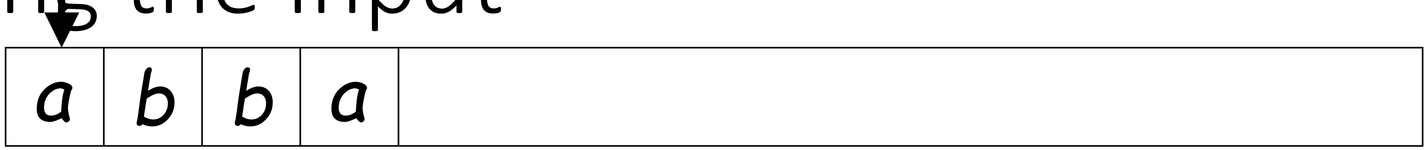
Initial Configuration

Input String

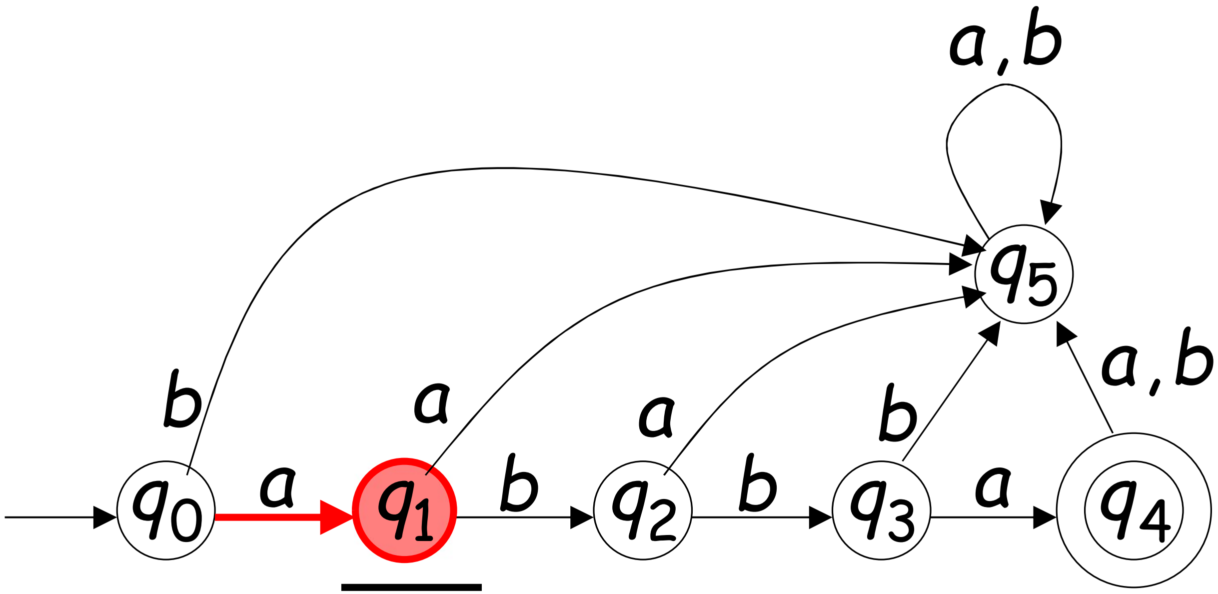
a	b	b	a	

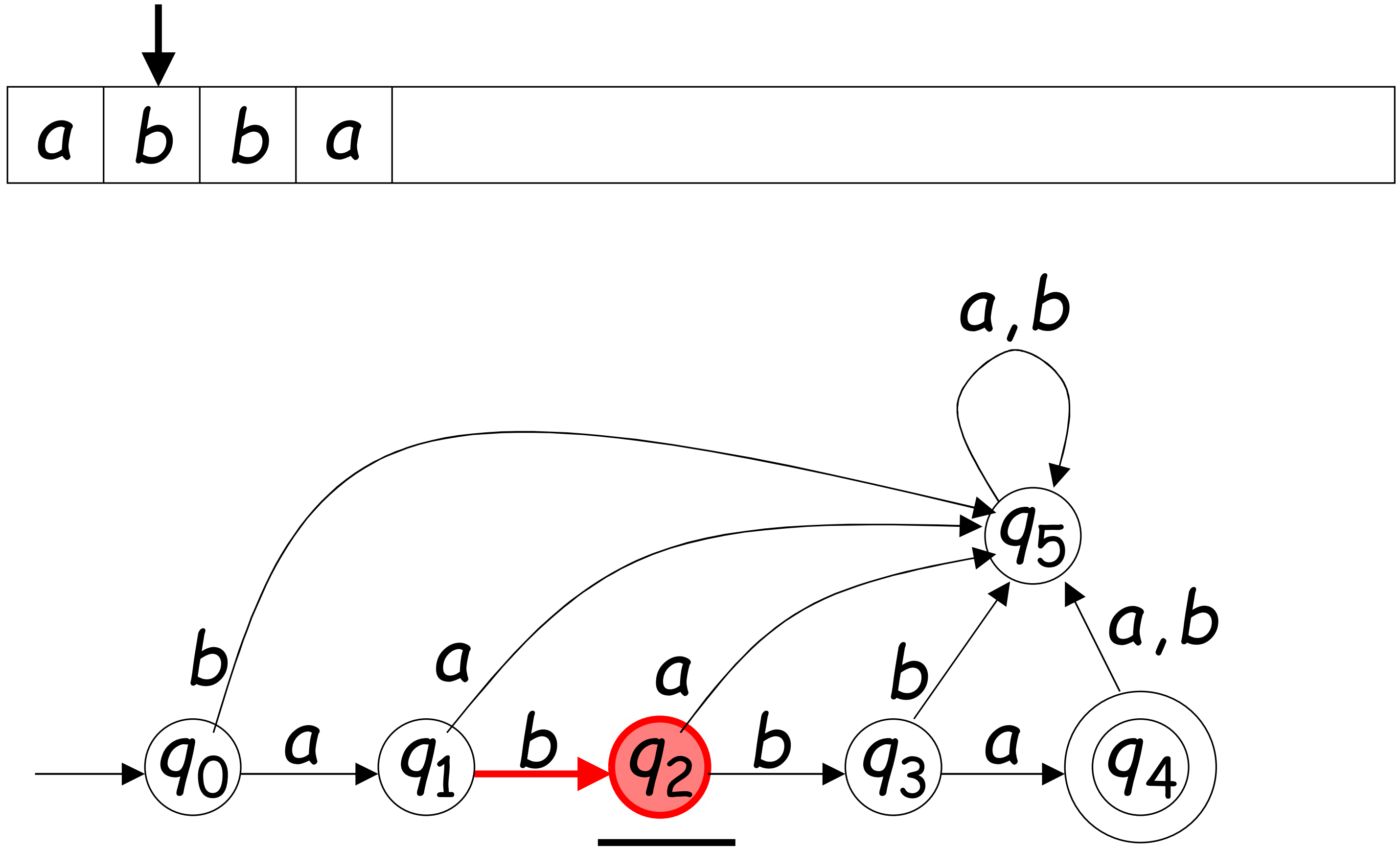


Reading the Input

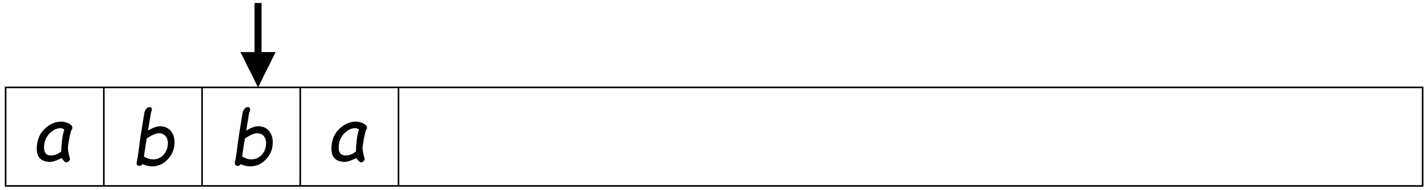


-

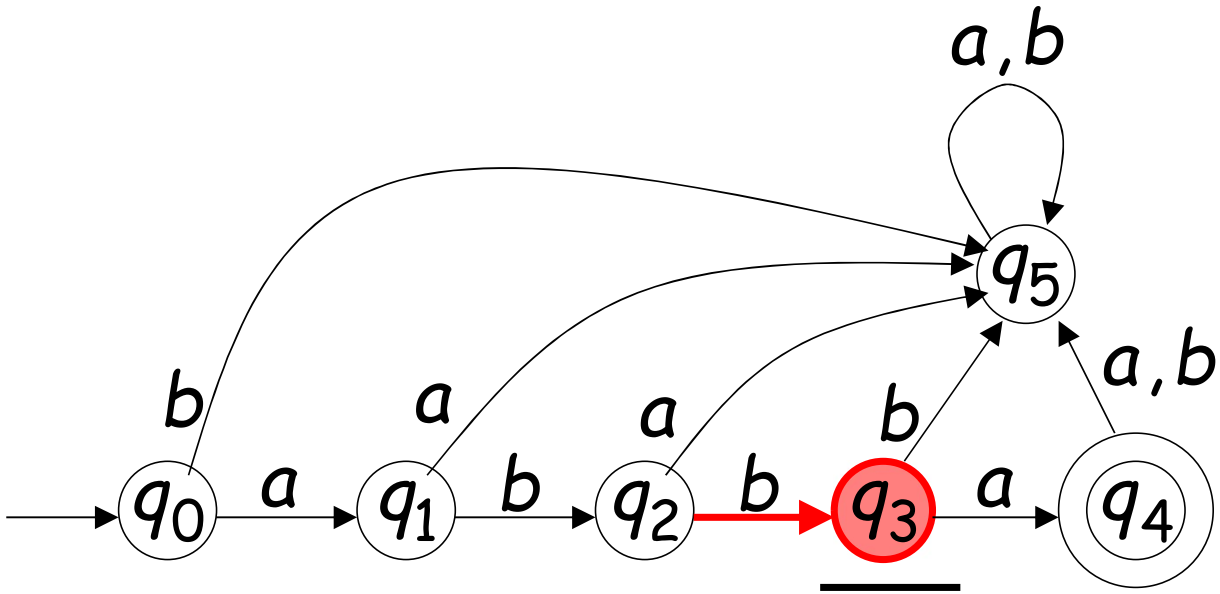


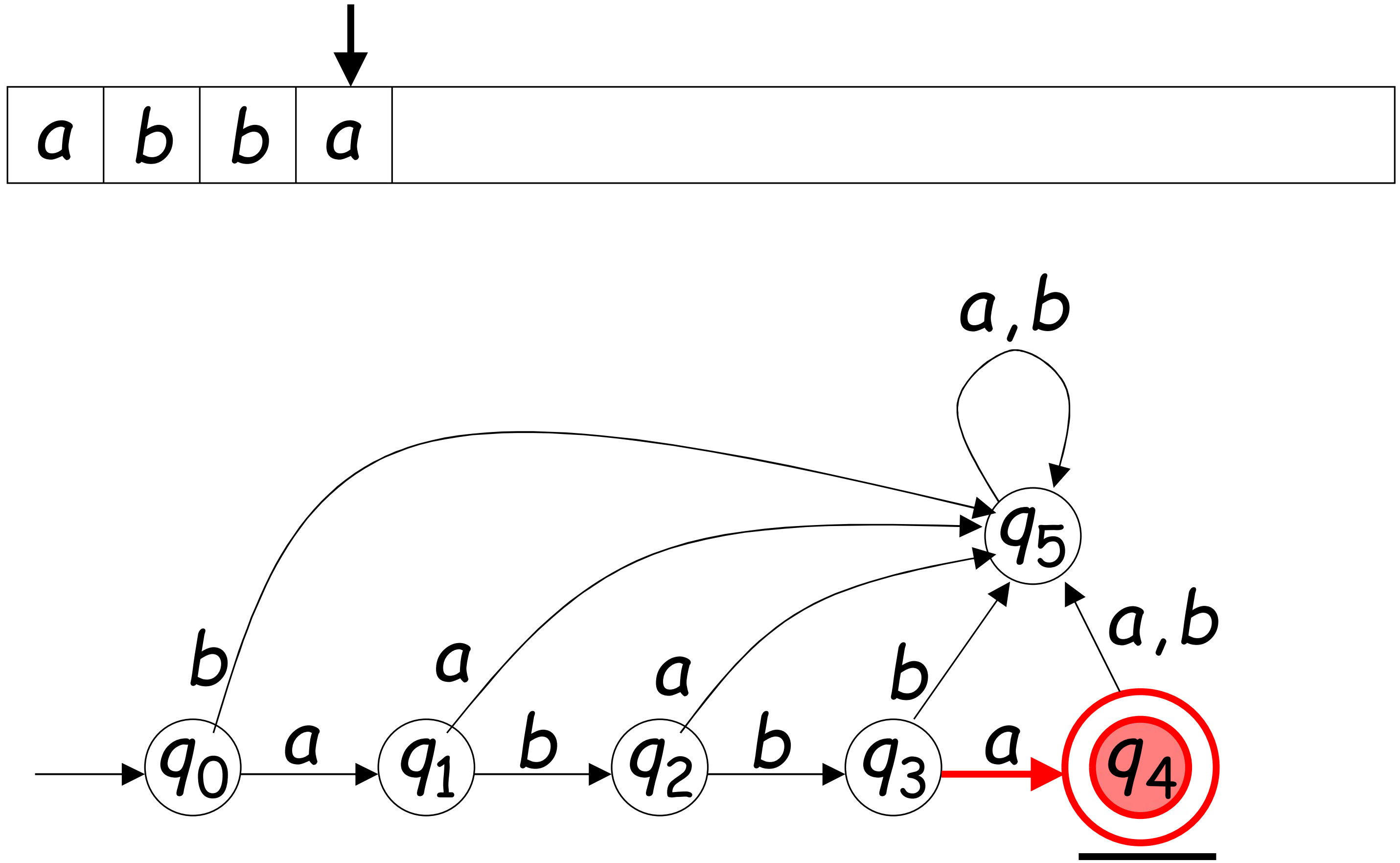


a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


a b b a

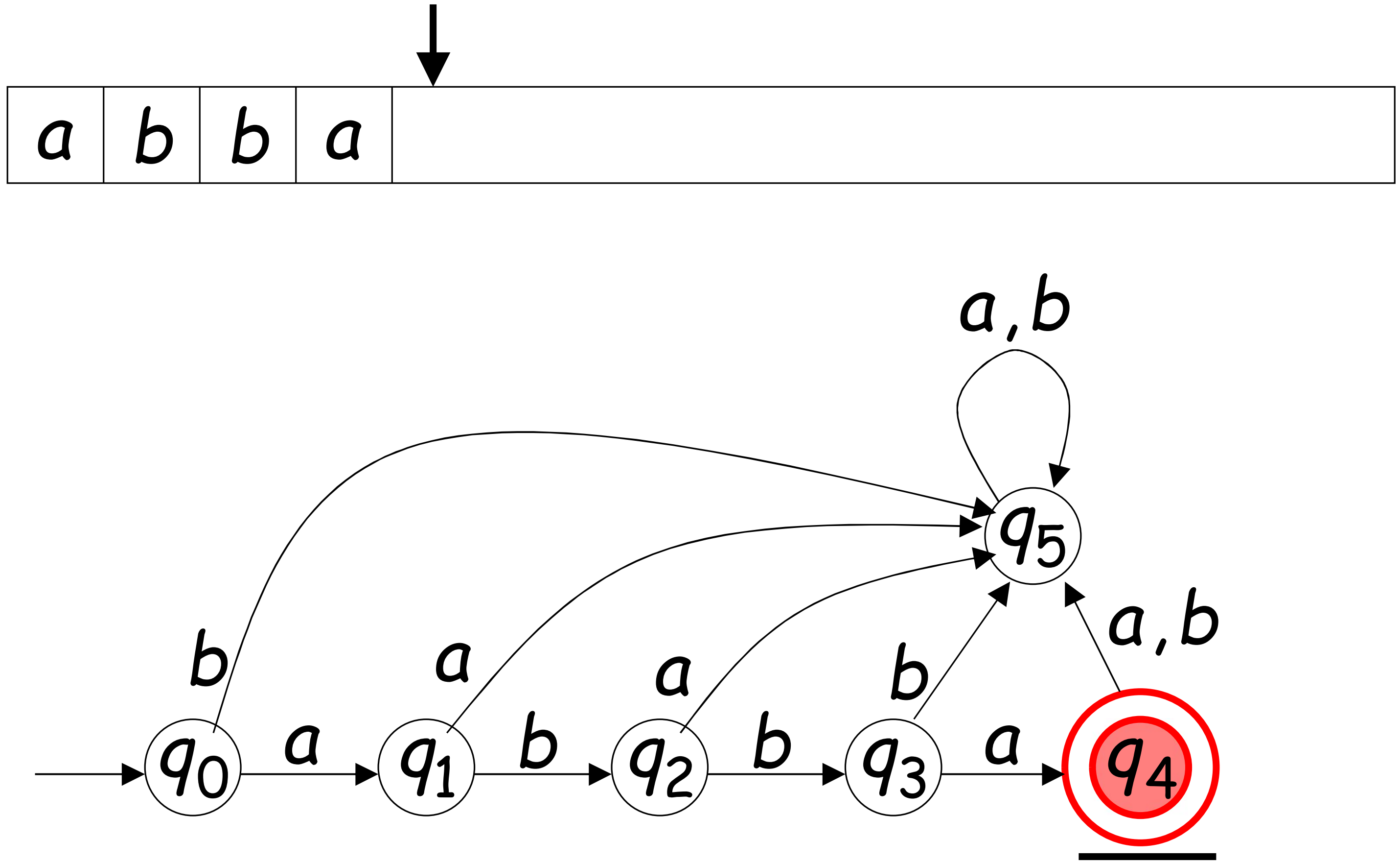
•


a,b
b
a
a
b
a,b
a
b
b
a
q0
q1
q2
q3
q4
q5



a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

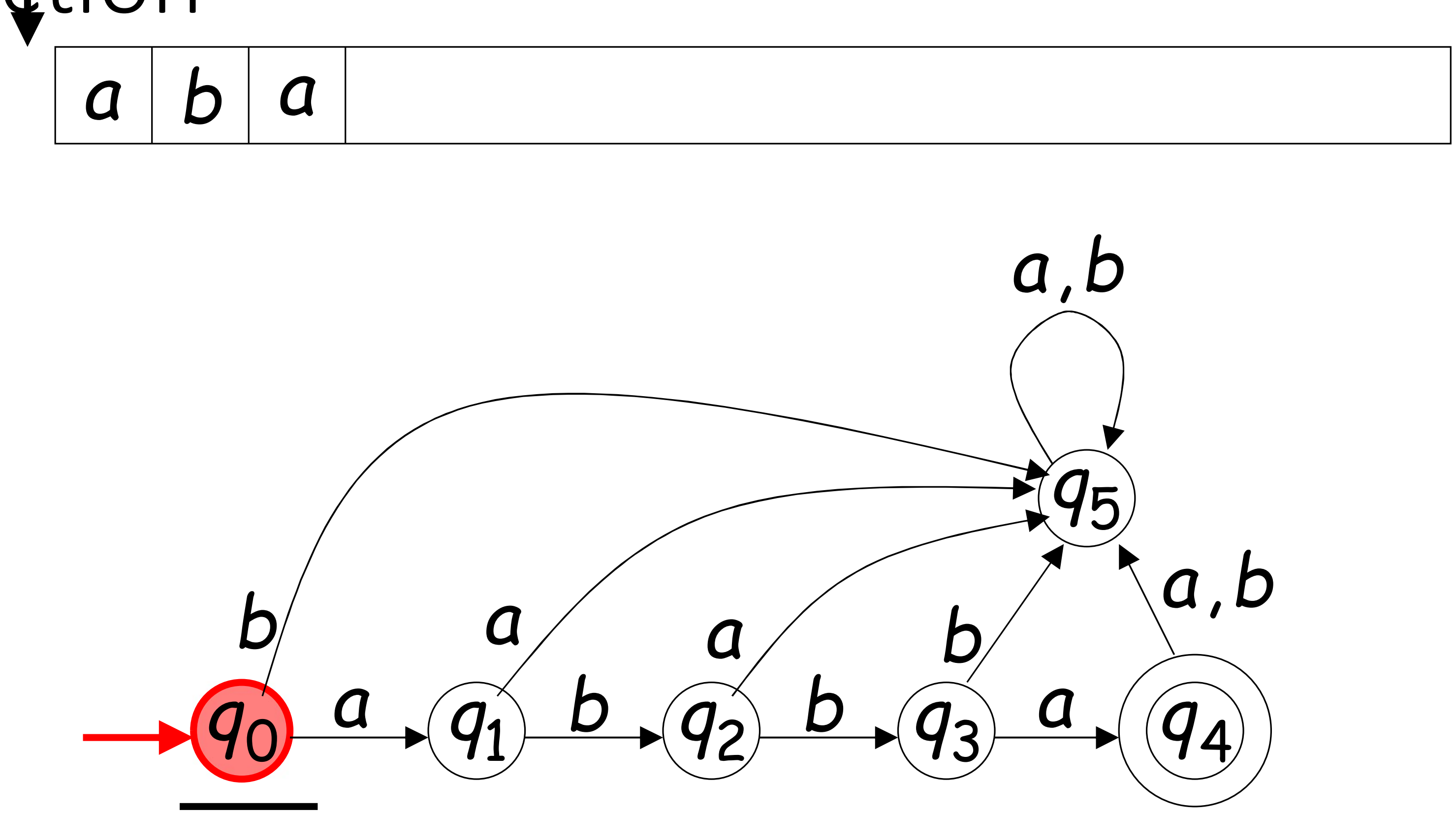
Input finished



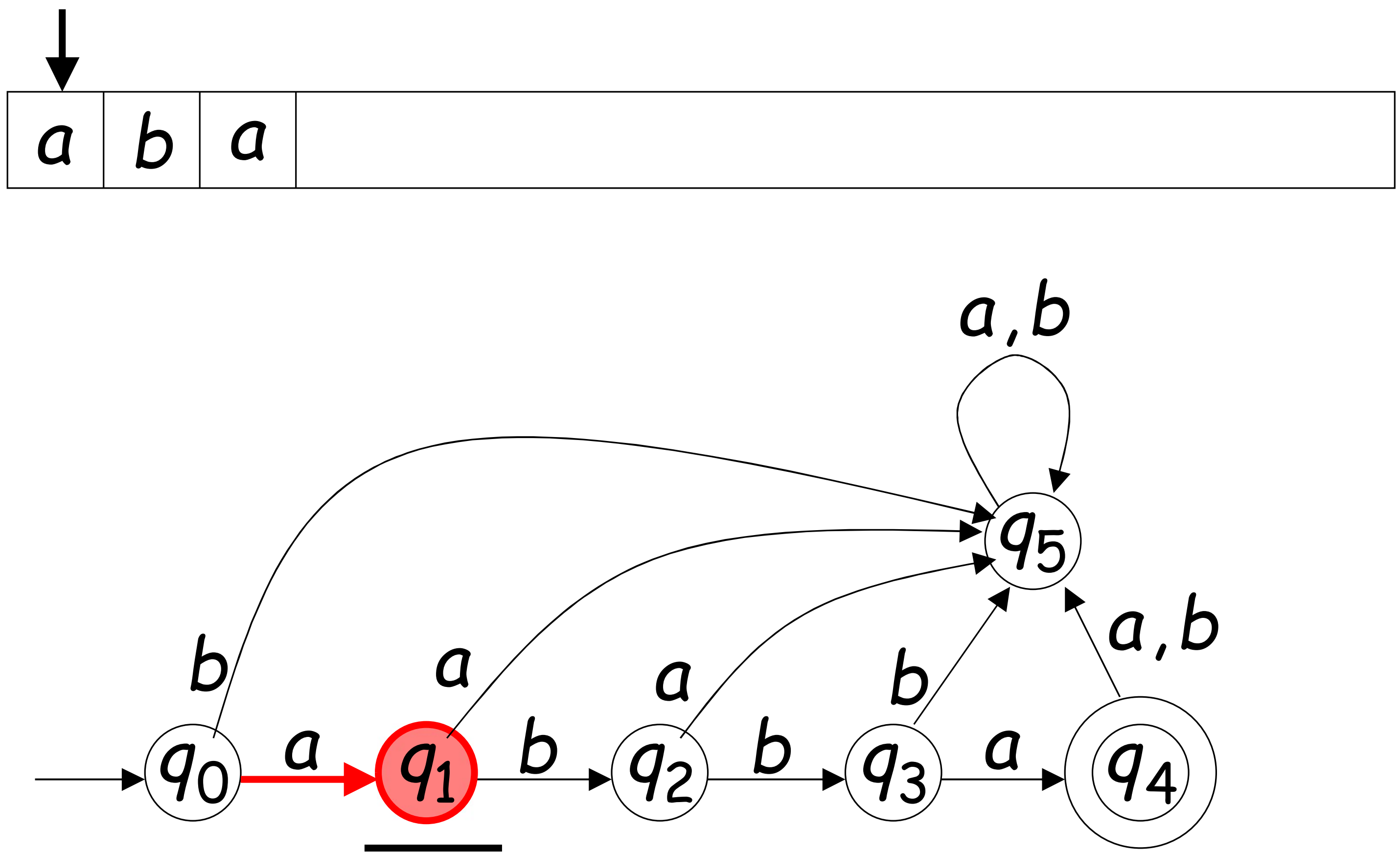
Output: “accept”

Rejection

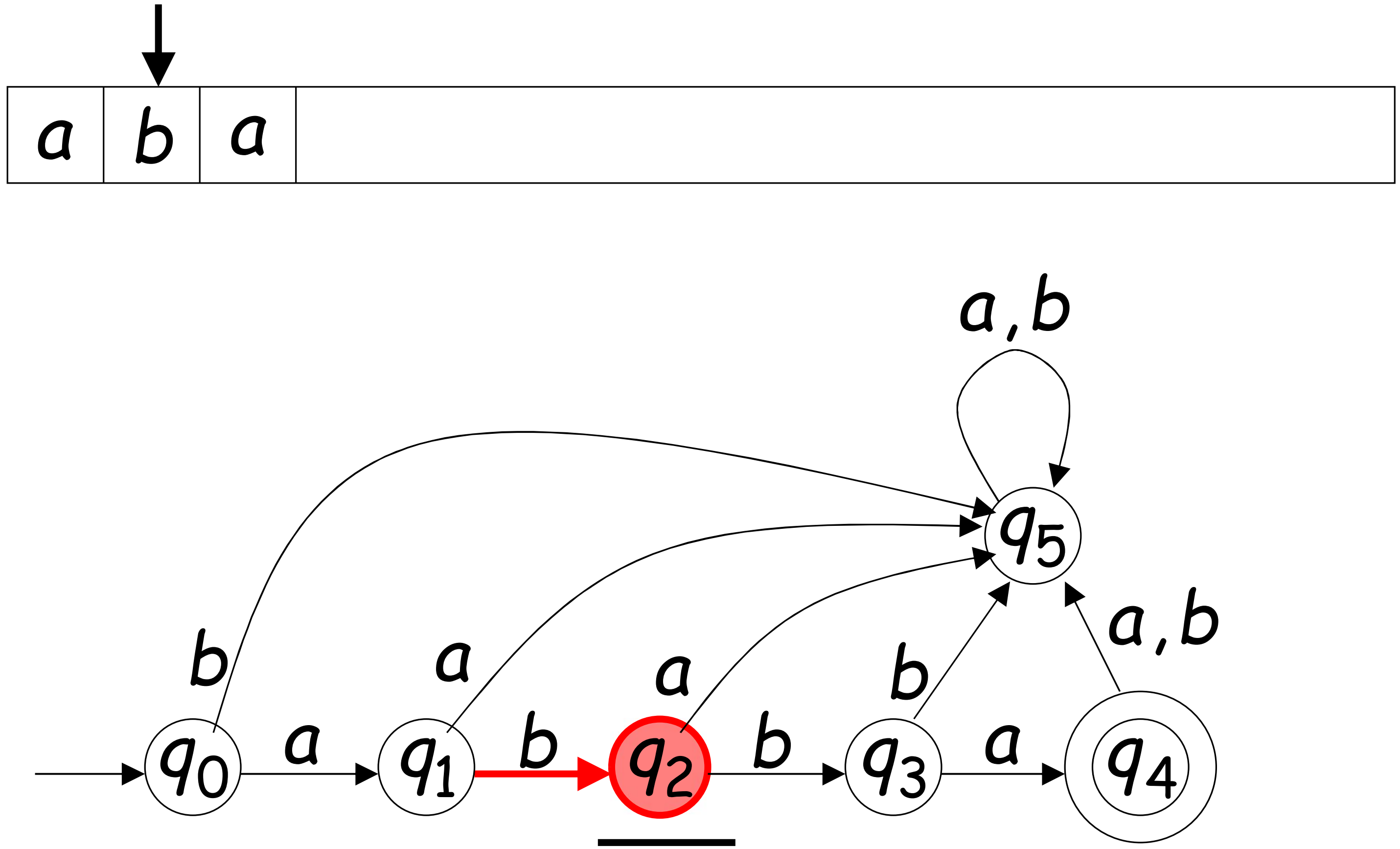
•



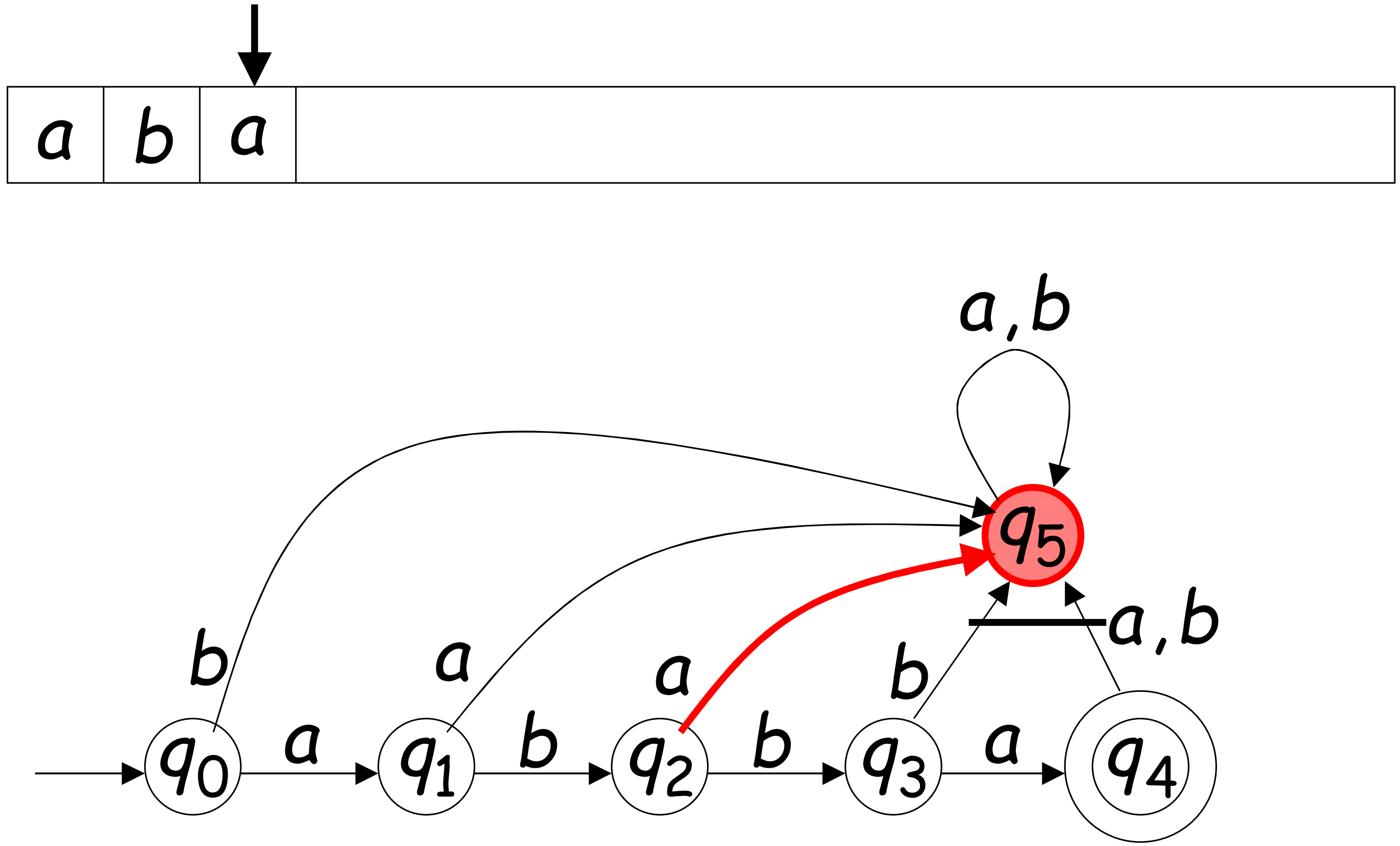
•


a b a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

•

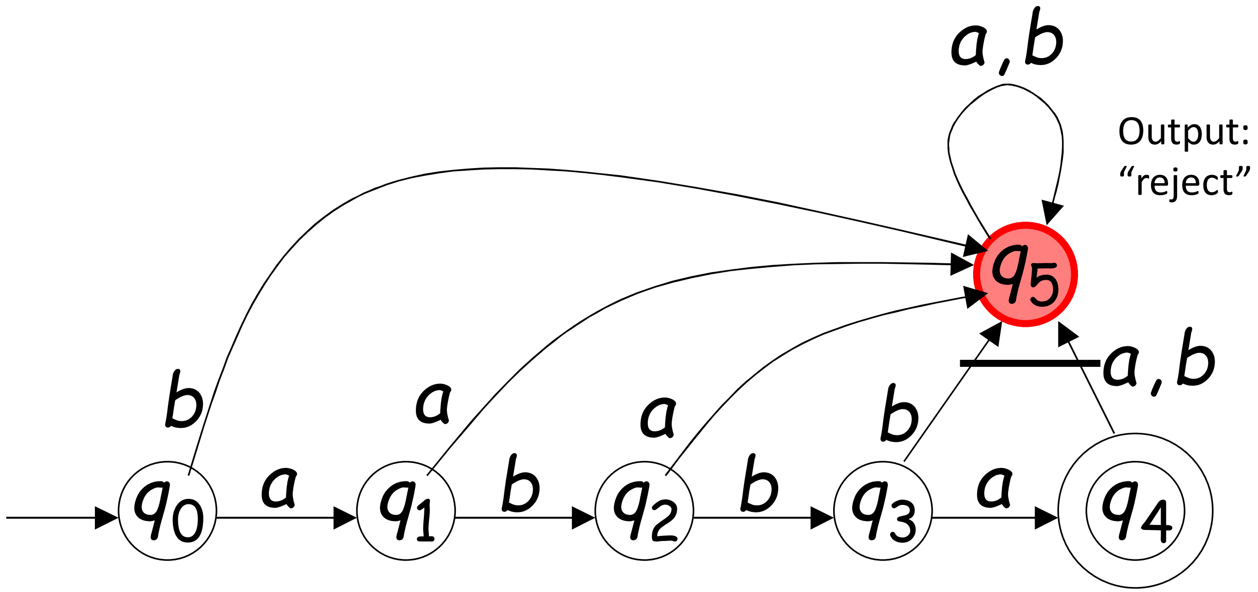

a b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

•

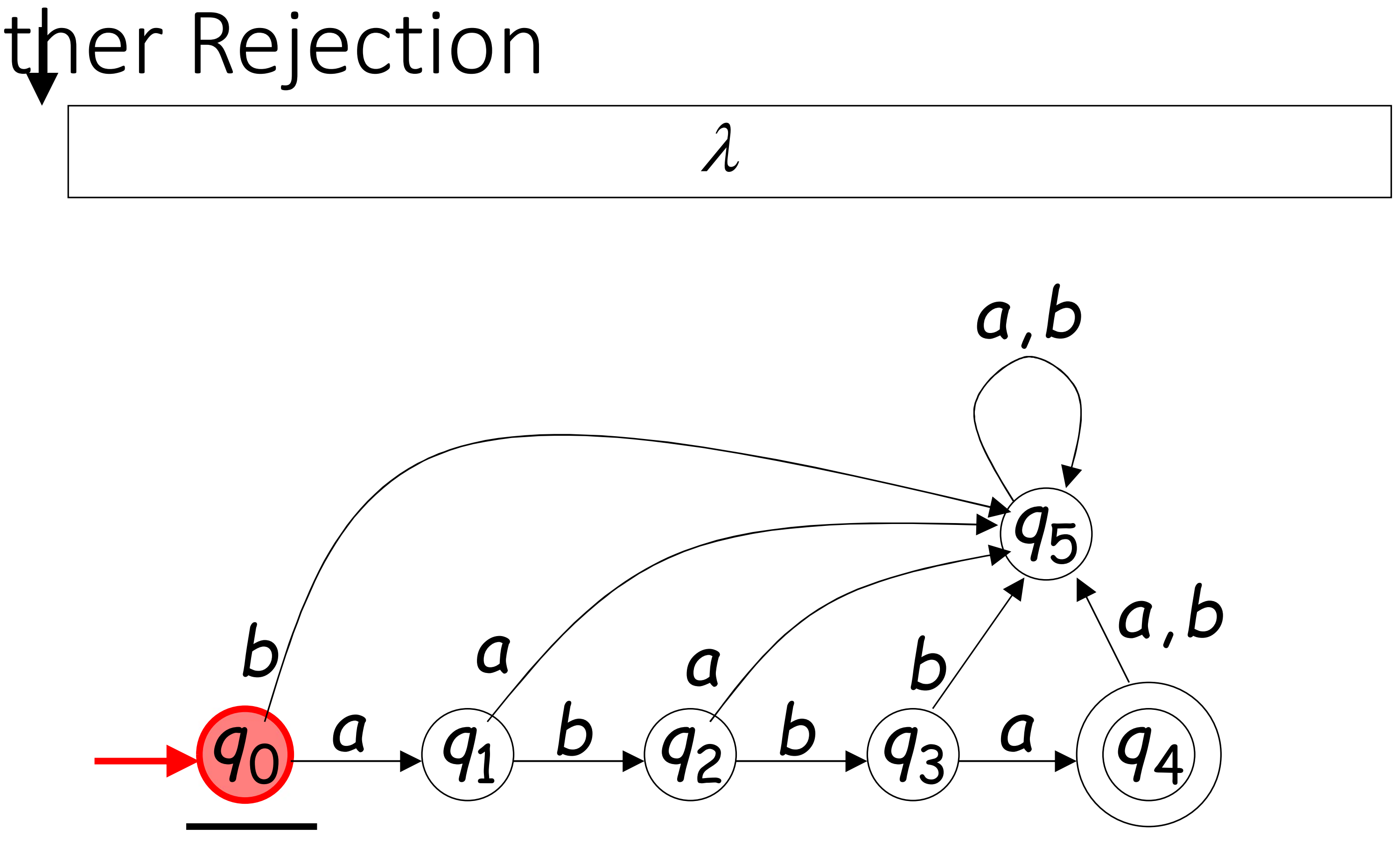

a
b
a
a,b
b
a
a
b
a,b
a
b
b
a
q_0
q_1
q_2
q_3
q_4
q_5

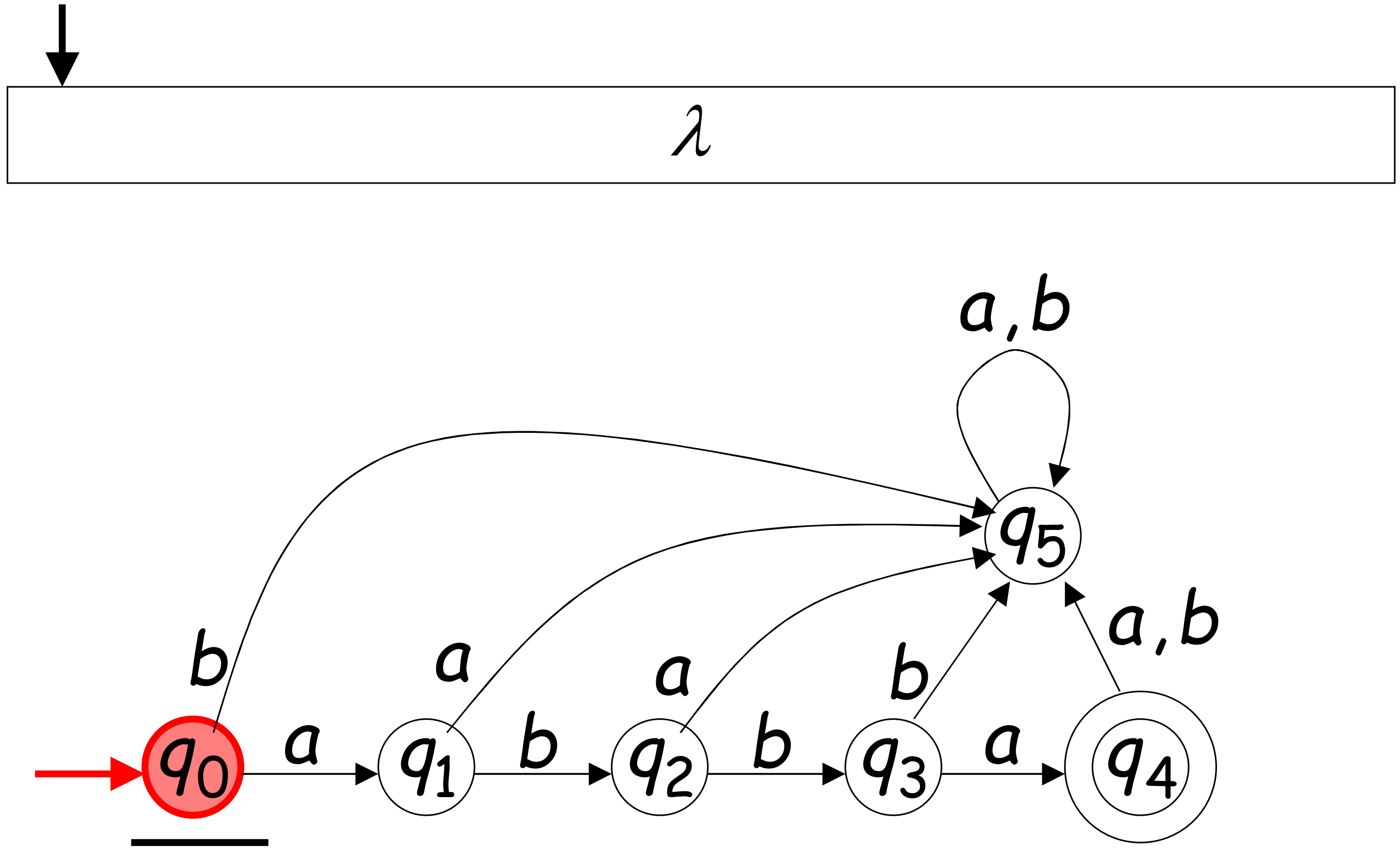
Input finished

a	b	a	



Another Rejection

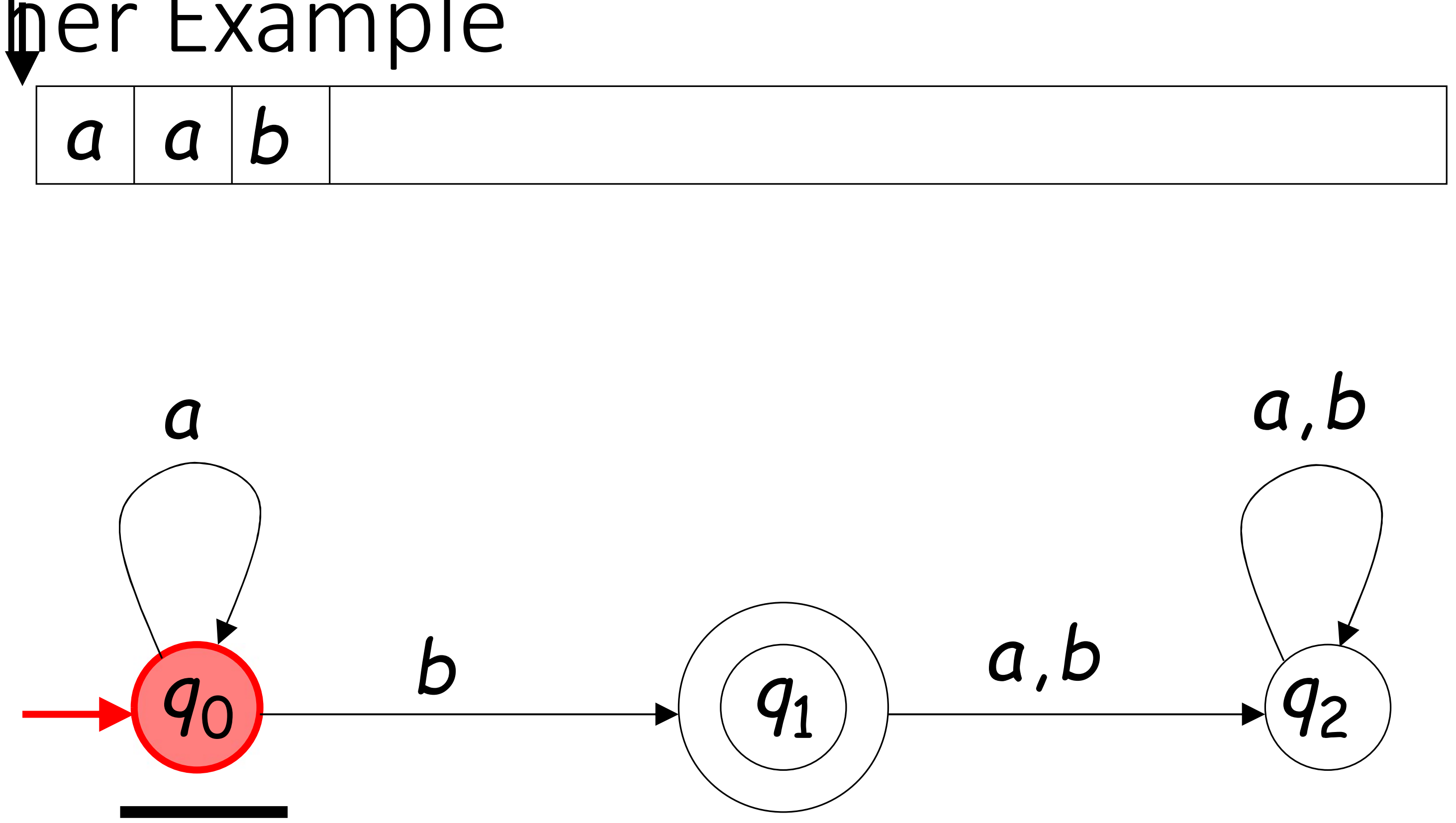


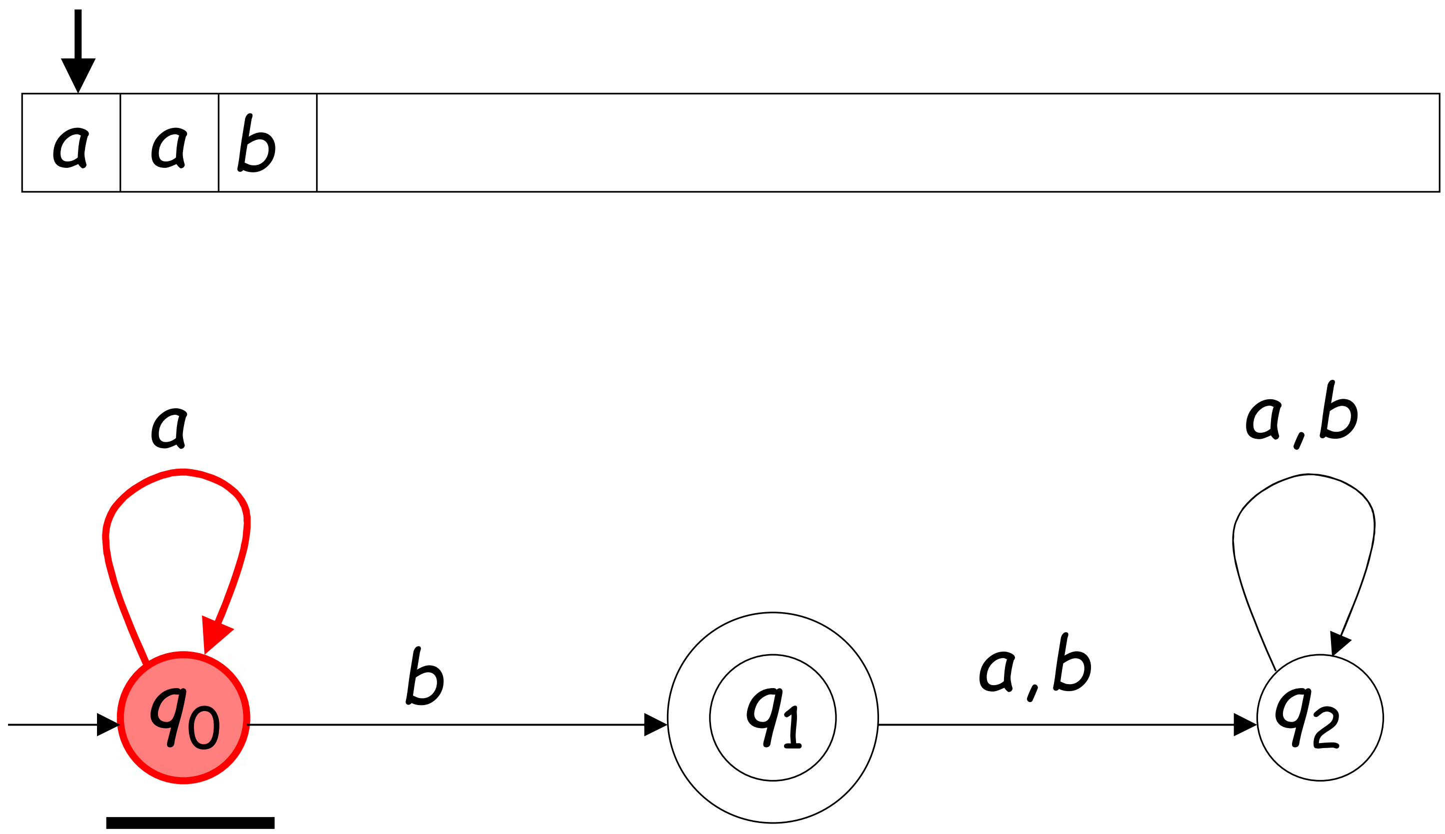


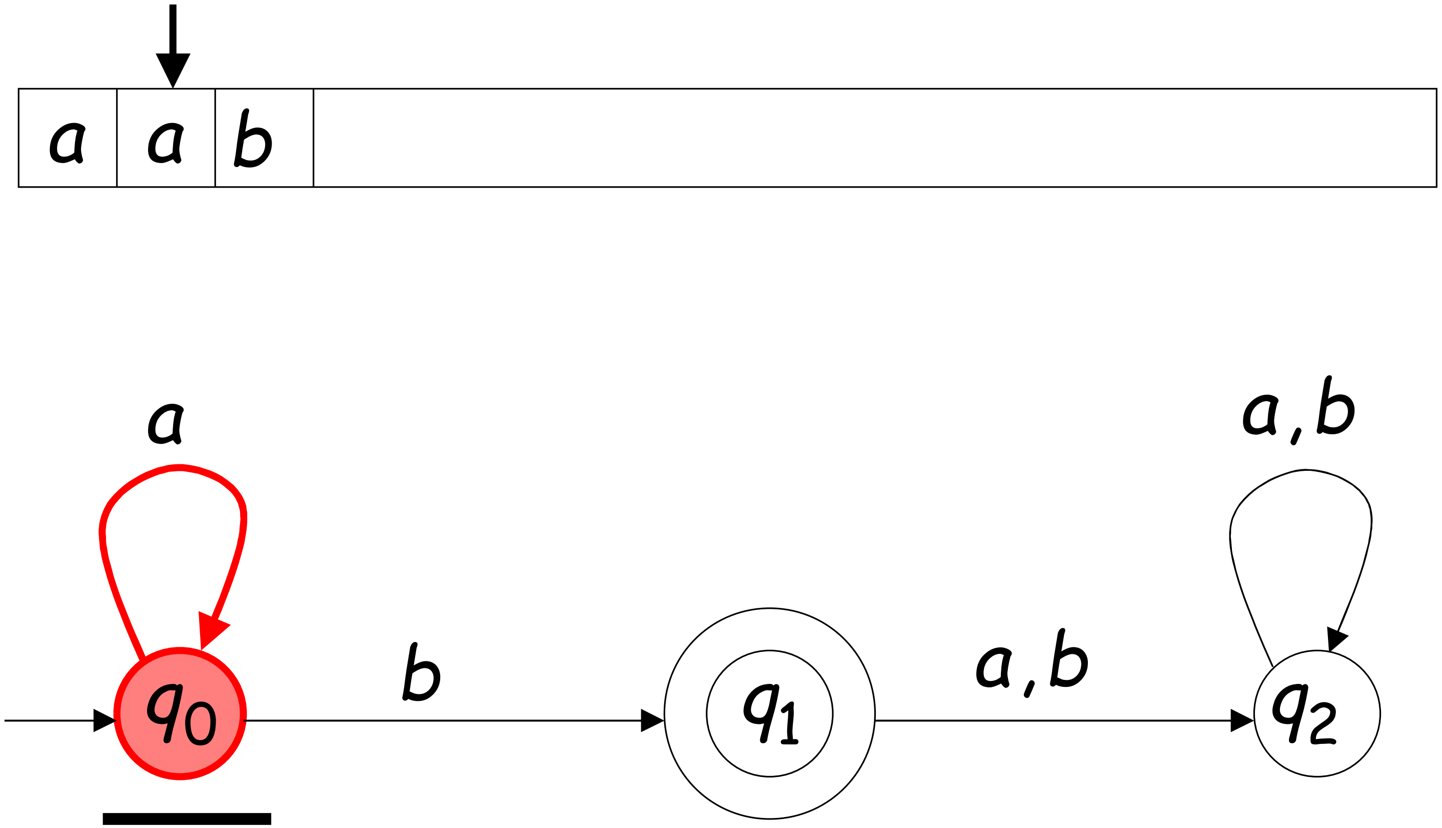
•

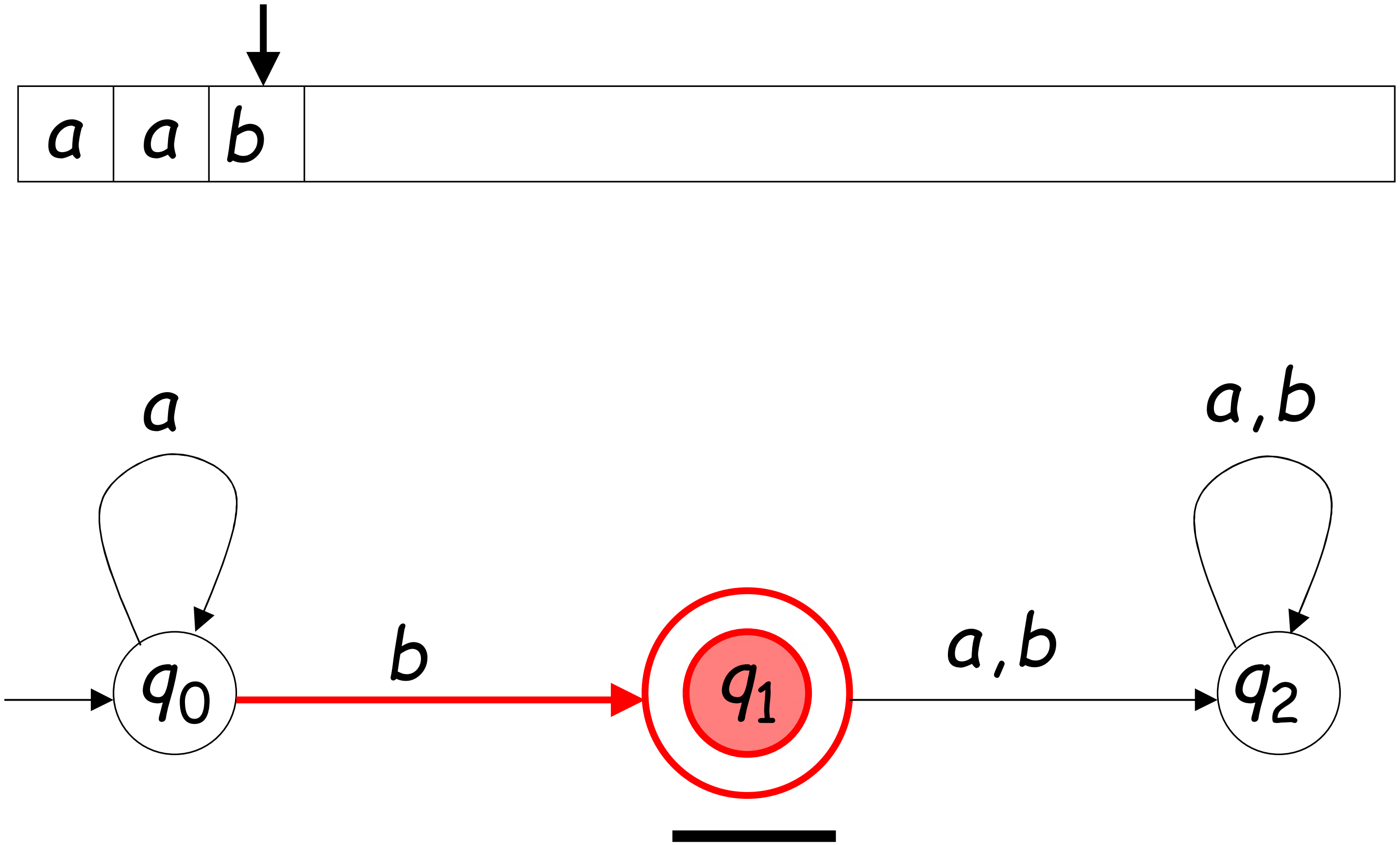
Output:
“reject”

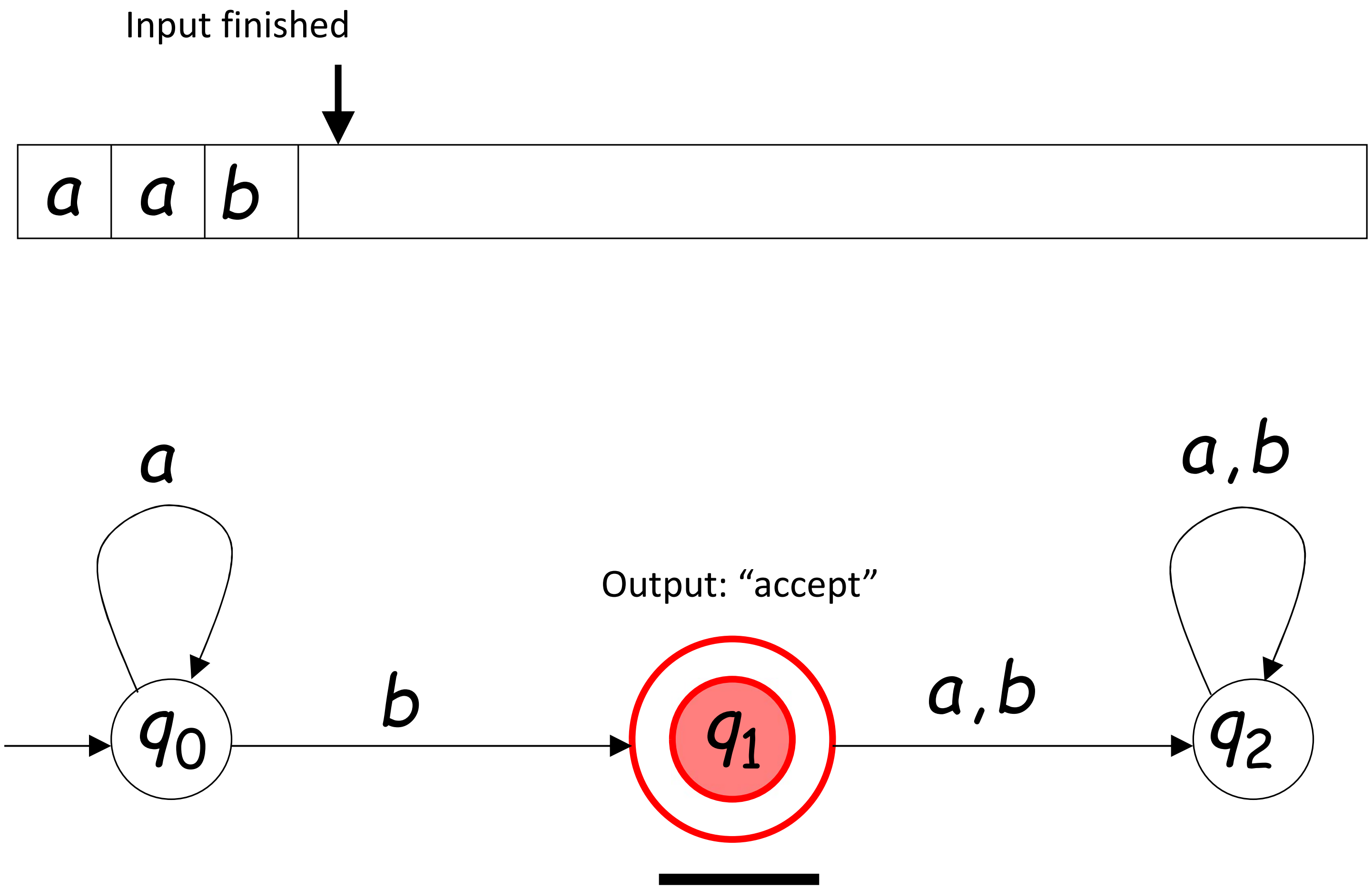
Another Example



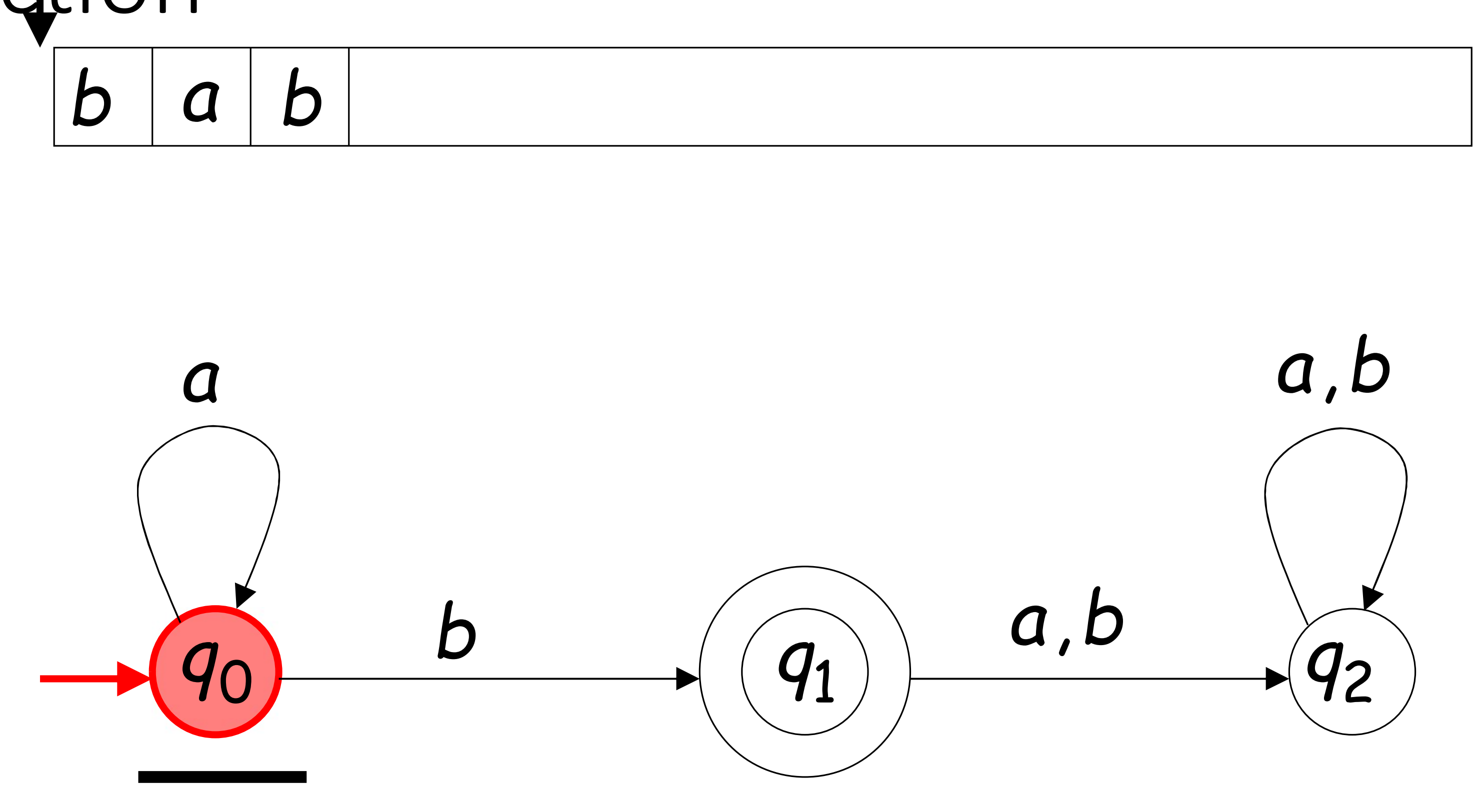

a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

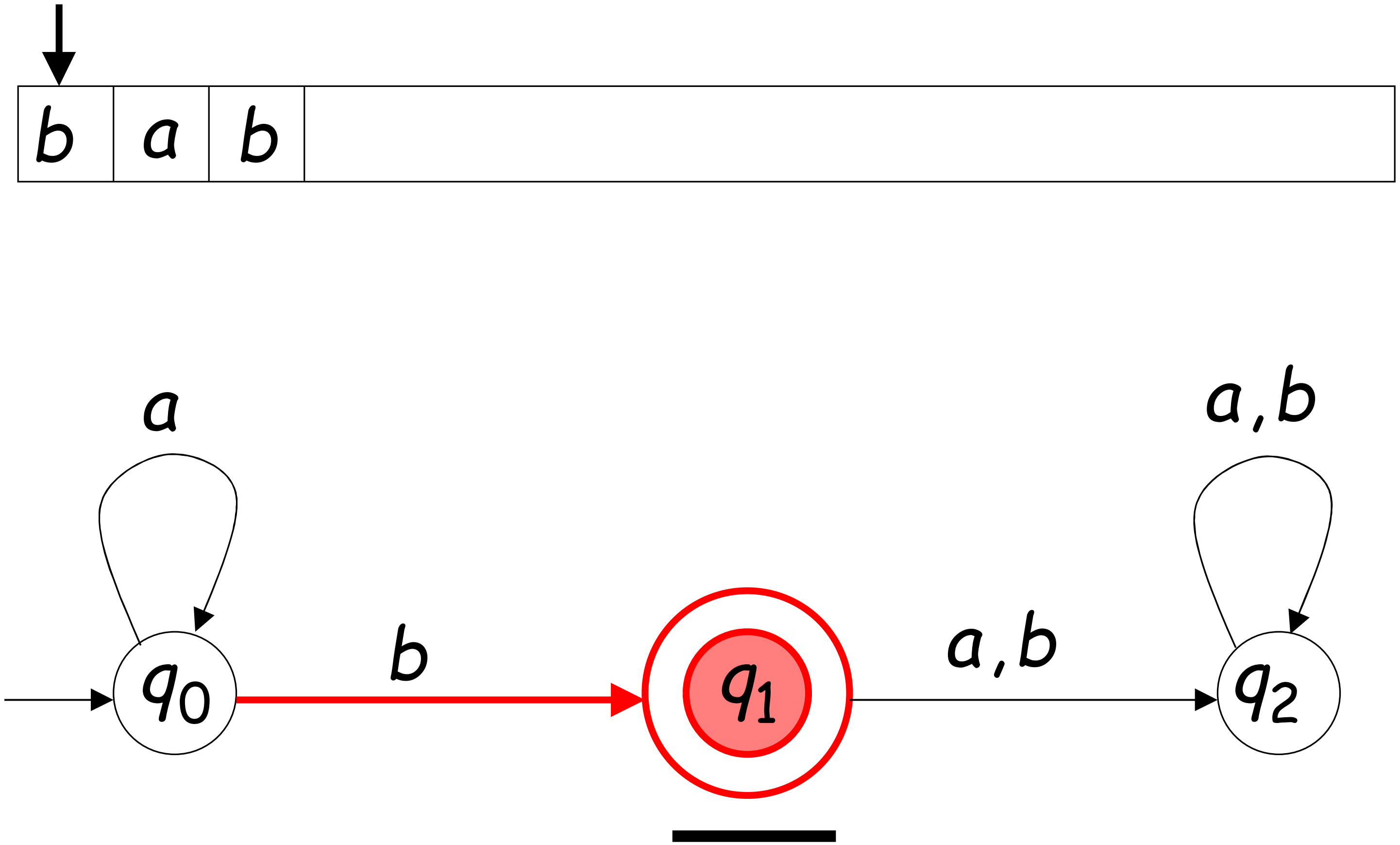

a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

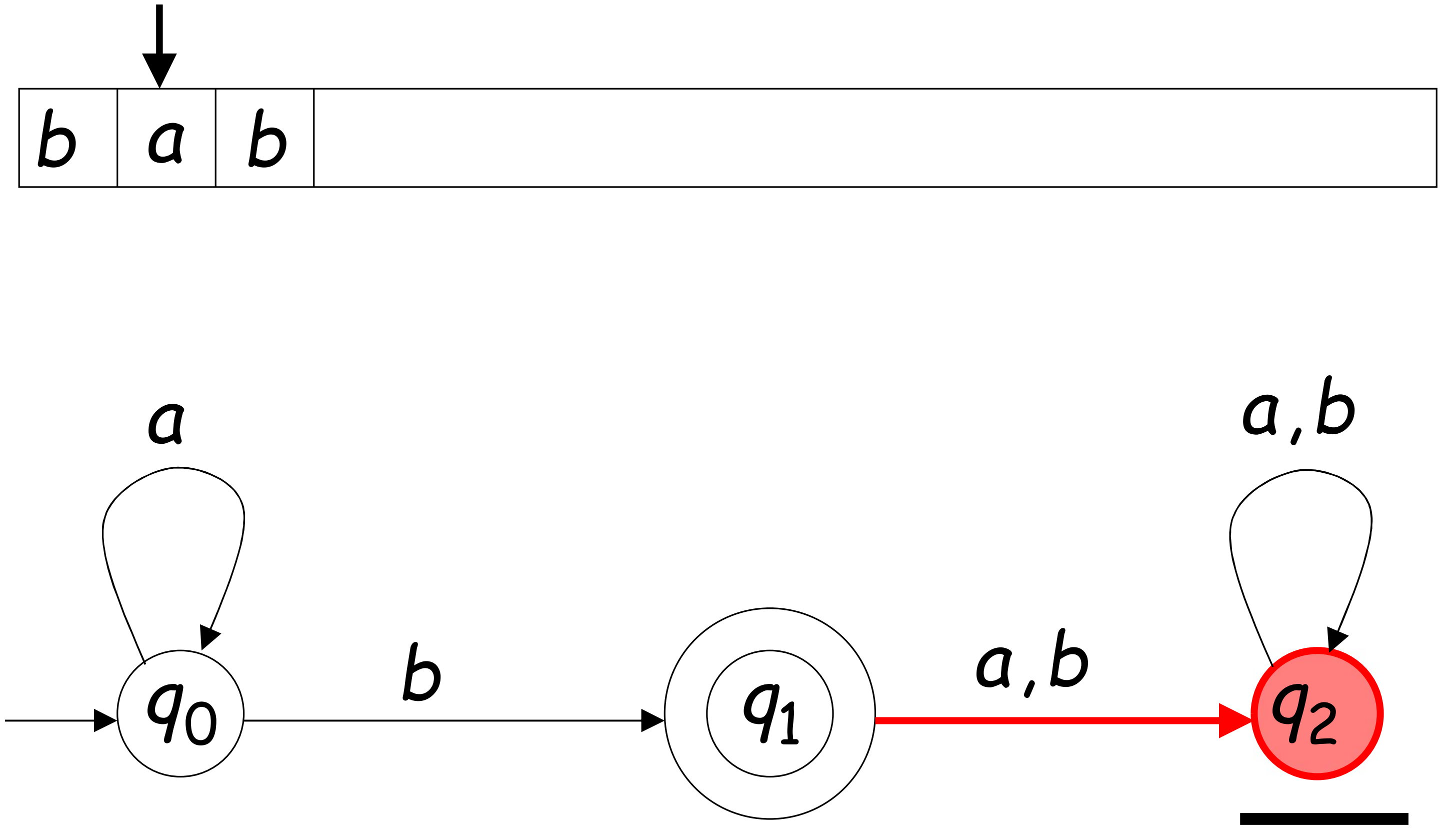

a
a
b
a
a,b
b
q0
q1
a,b
q2

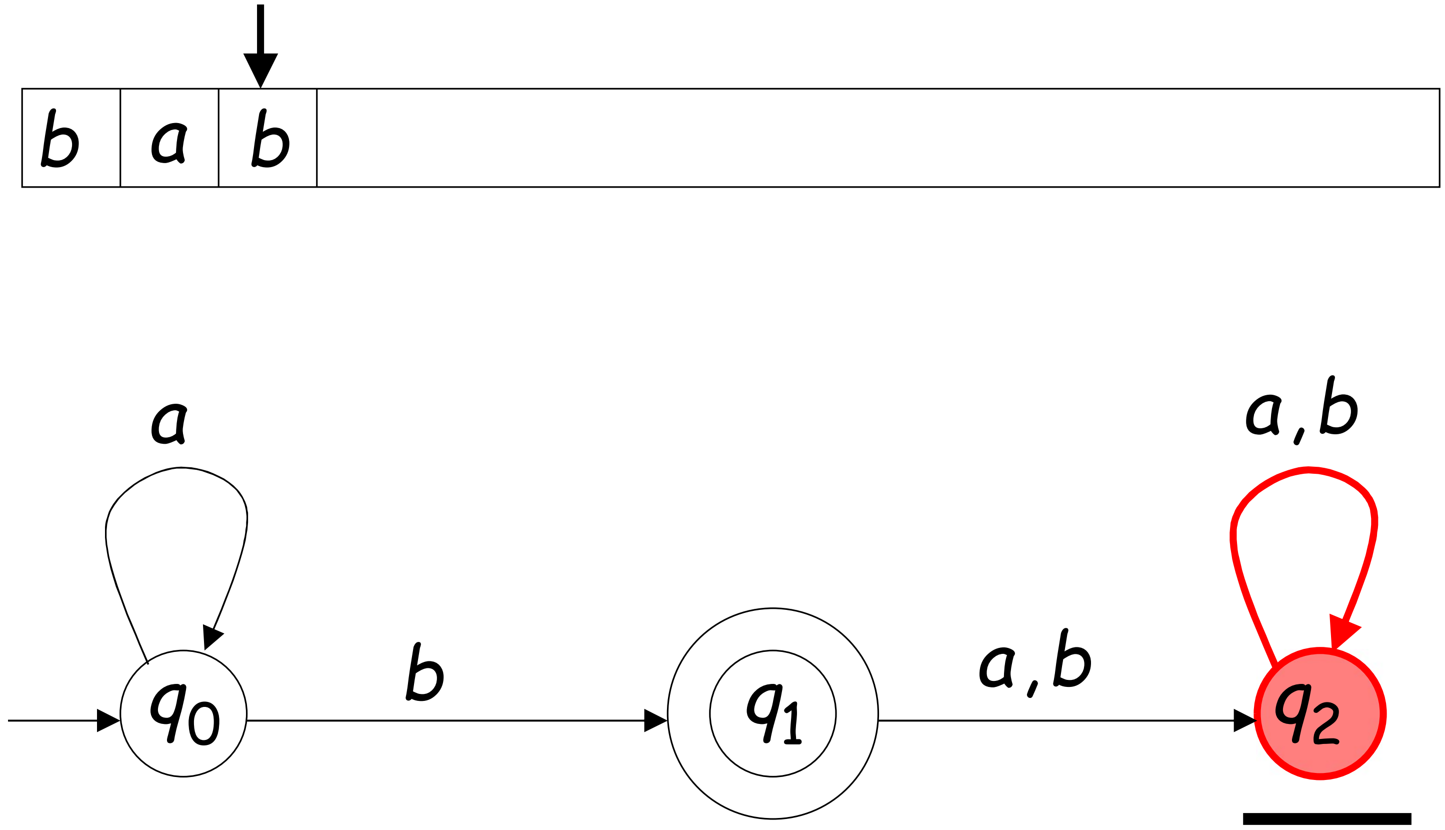

Input finished
a
a
b
a
b
Output: “accept”
a,b
a,b
q_0
q_1
q_2

Rejection



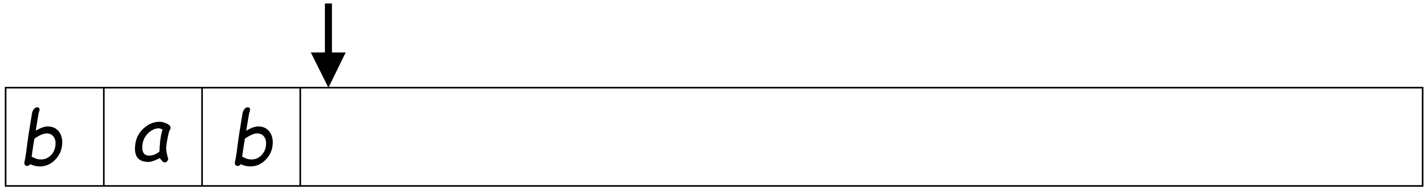

b a b
a
a,b
b
a,b
q0
q1
q2

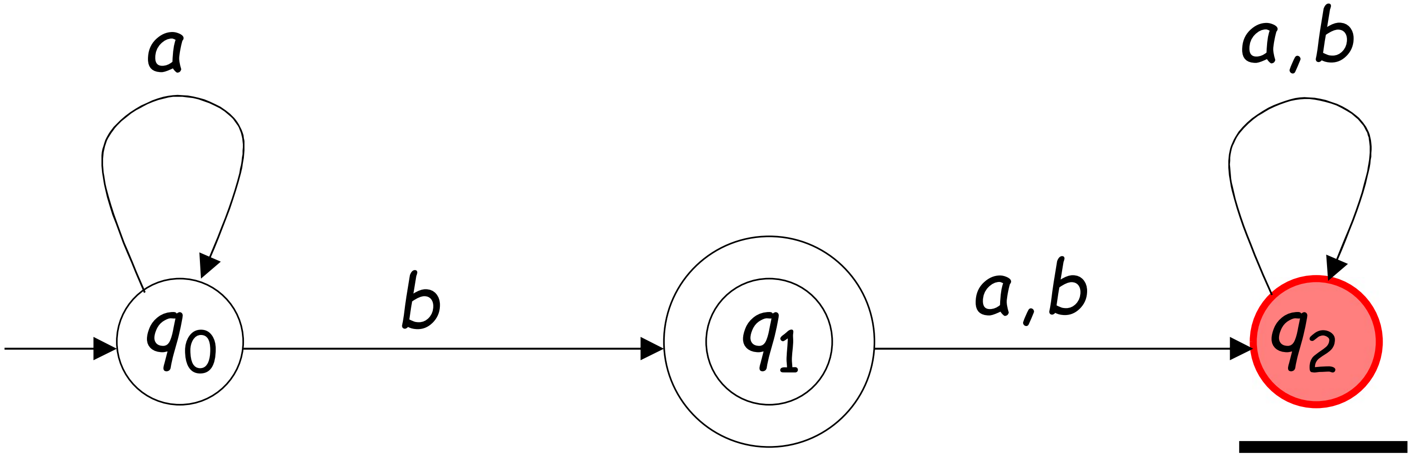

b
a
b
a
a,b
b
q_0
q_1
a,b
q_2



b
a
b
a
b
a, b
a, b
q_0
q_1
q_2

Input finished





Output: “reject”

Formalities

- Deterministic Finite Accepter (DFA) $M = (Q, \Sigma, \delta, q_0, F)$

Q : set of states

Σ : input alphabet

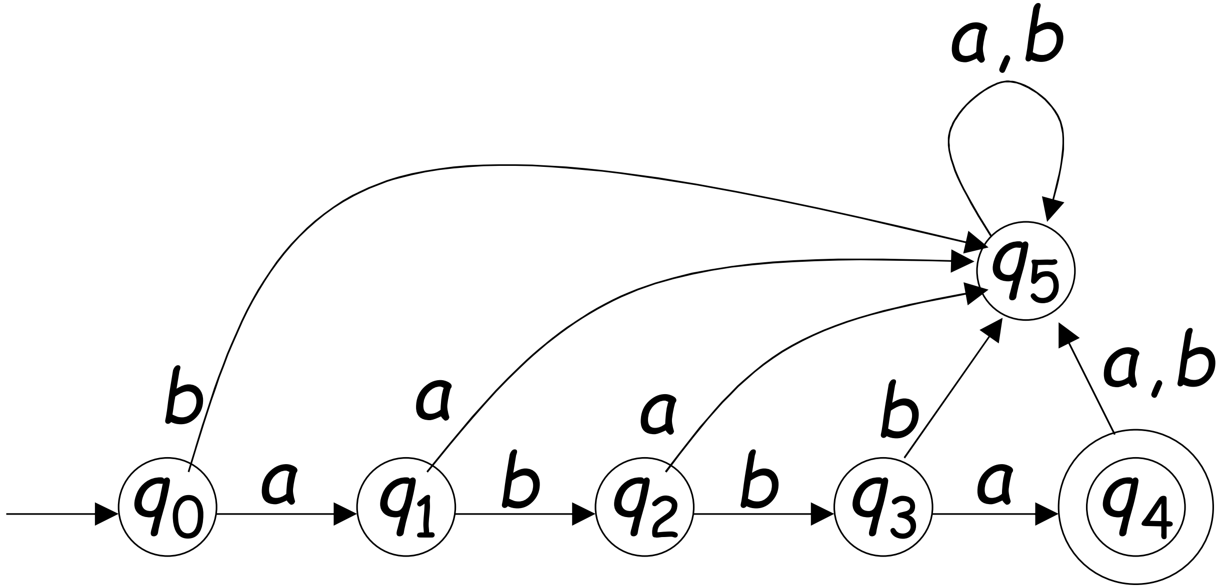
δ : transition function

q_0 : initial state

F : set of final states

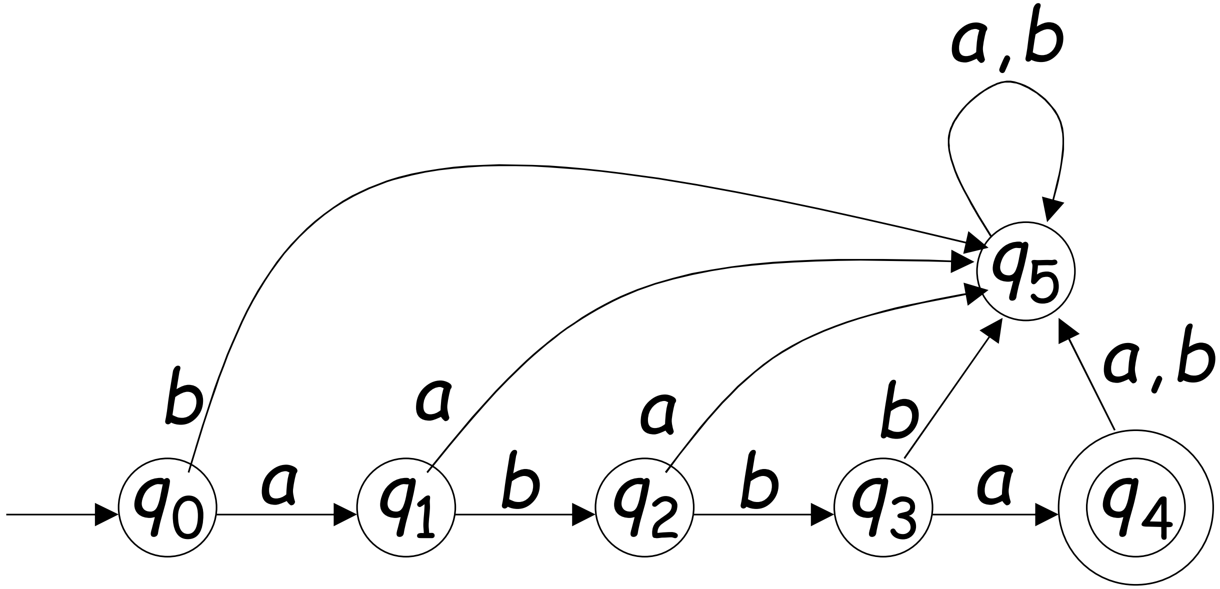
Input Alphabet Σ

$$\Sigma = \{a, b\}$$



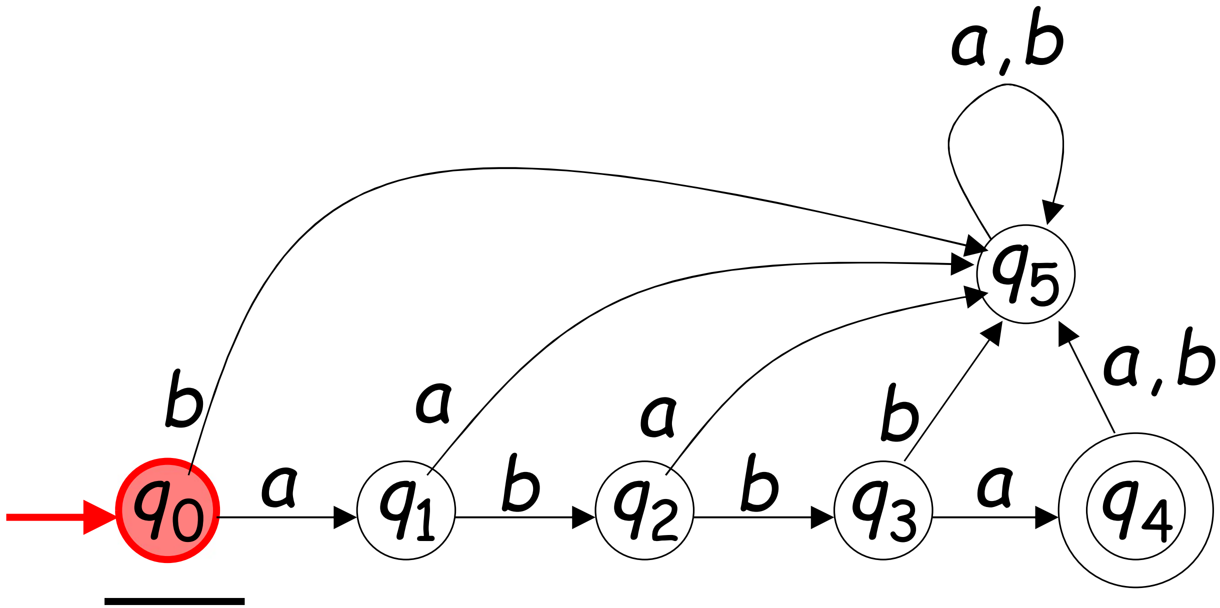
Set of States Q

- $$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$$



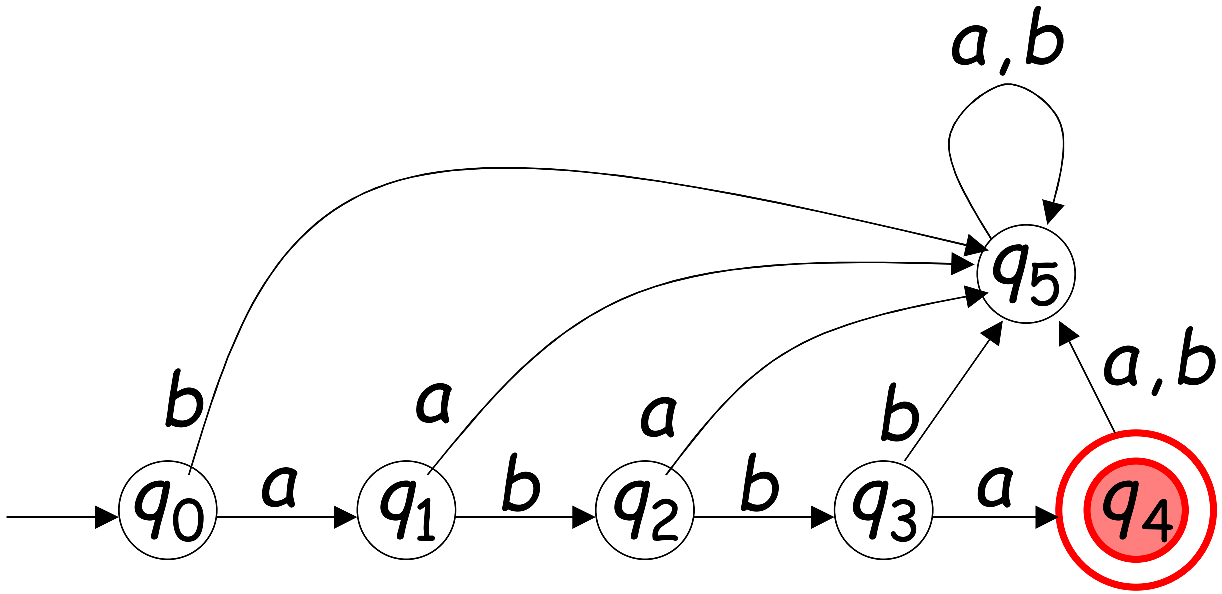
Initial State q_0

-



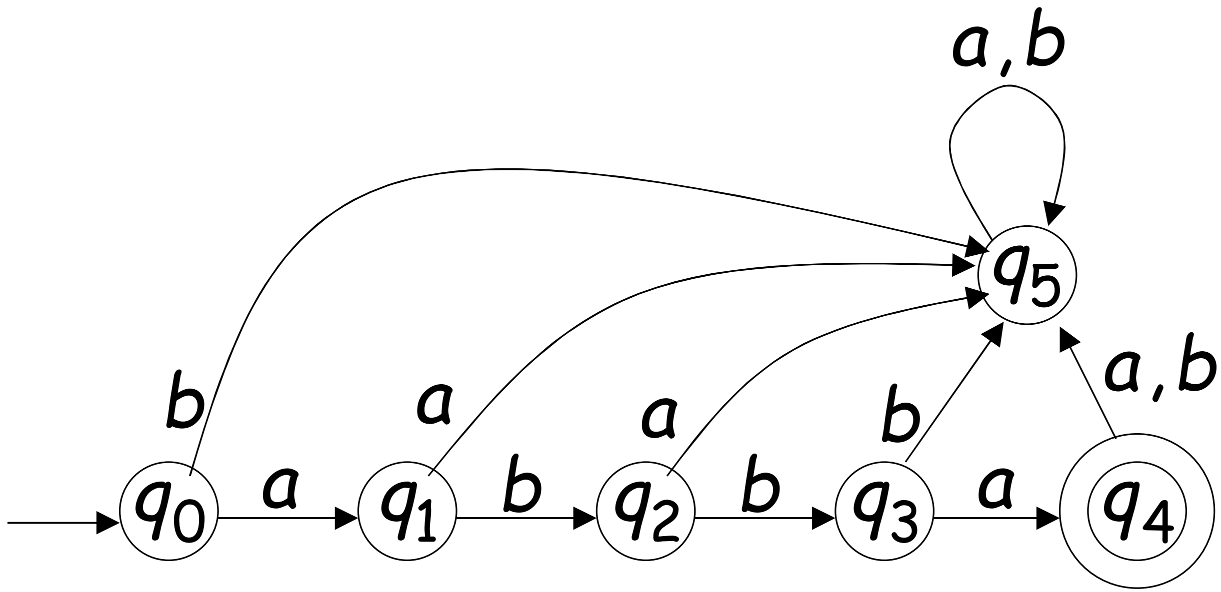
Set of Final States F

- $$F = \{q_4\}$$

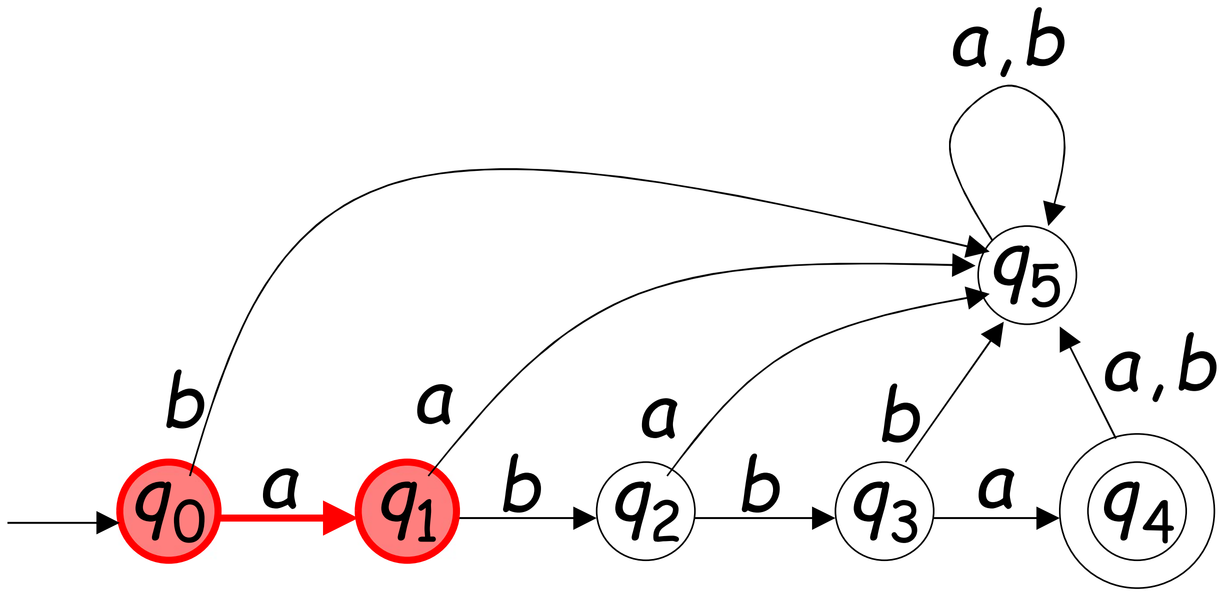


Transition Function δ

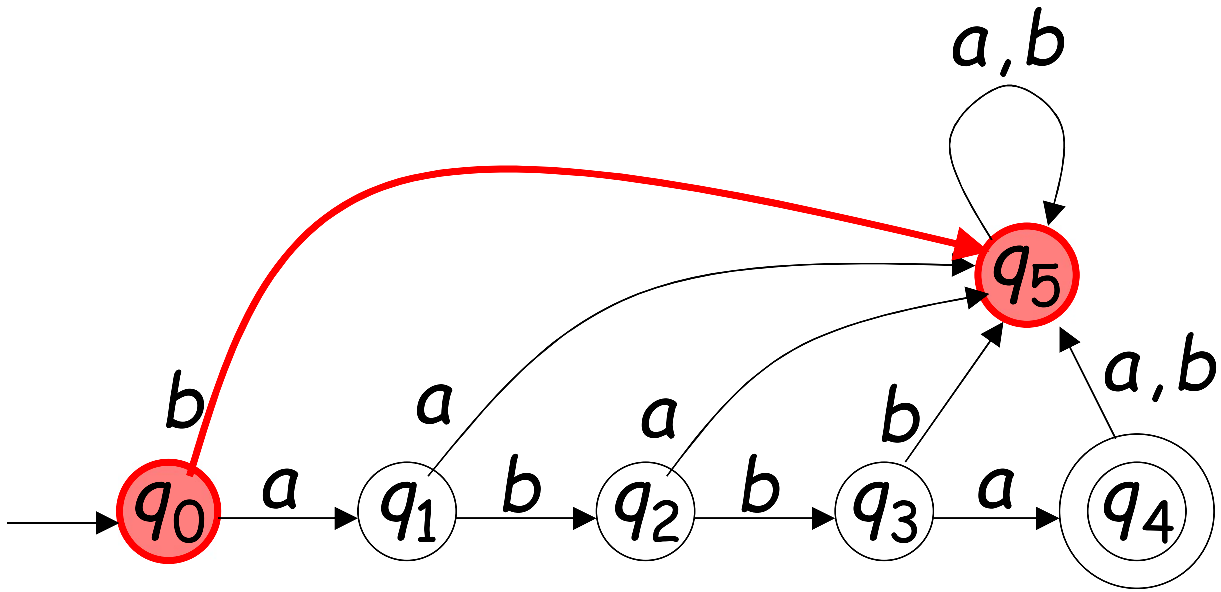
- $$\delta : Q \times \Sigma \rightarrow Q$$



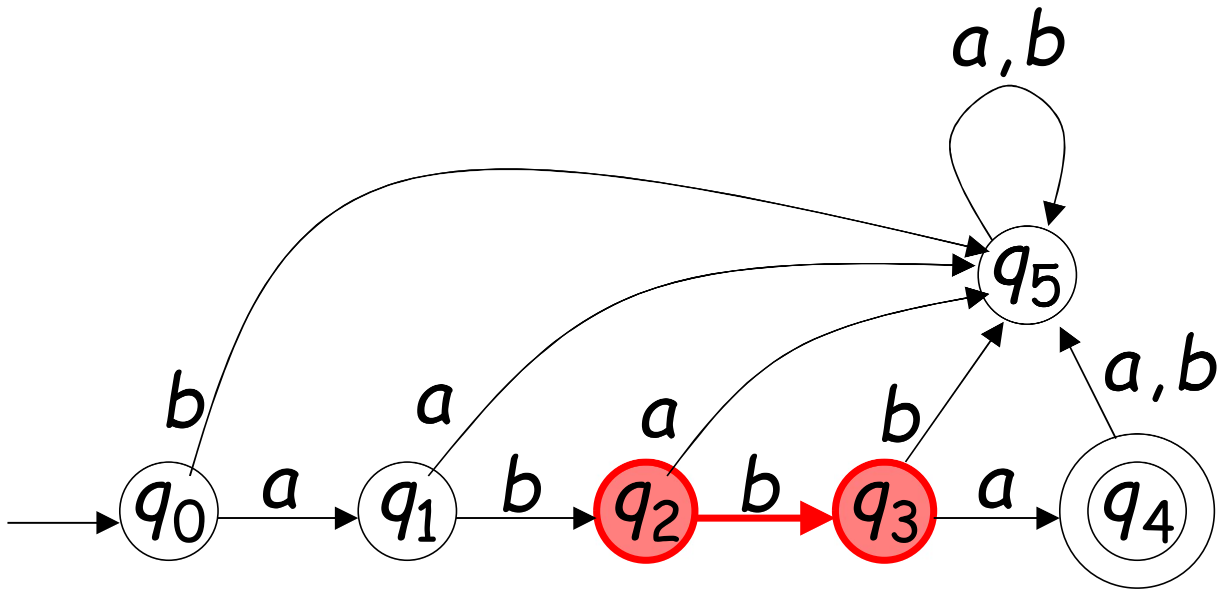
$$\delta(q_0, a) = q_1$$



$$\delta(q_0, b) = q_5$$

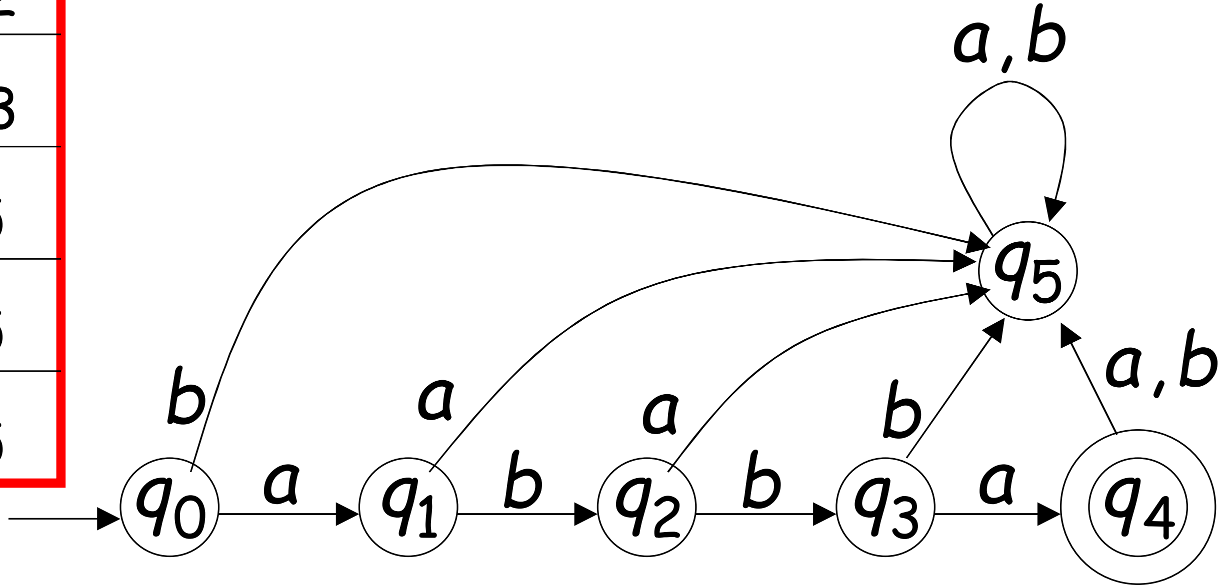


$$\delta(q_2, b) = q_3$$



Transition Function δ

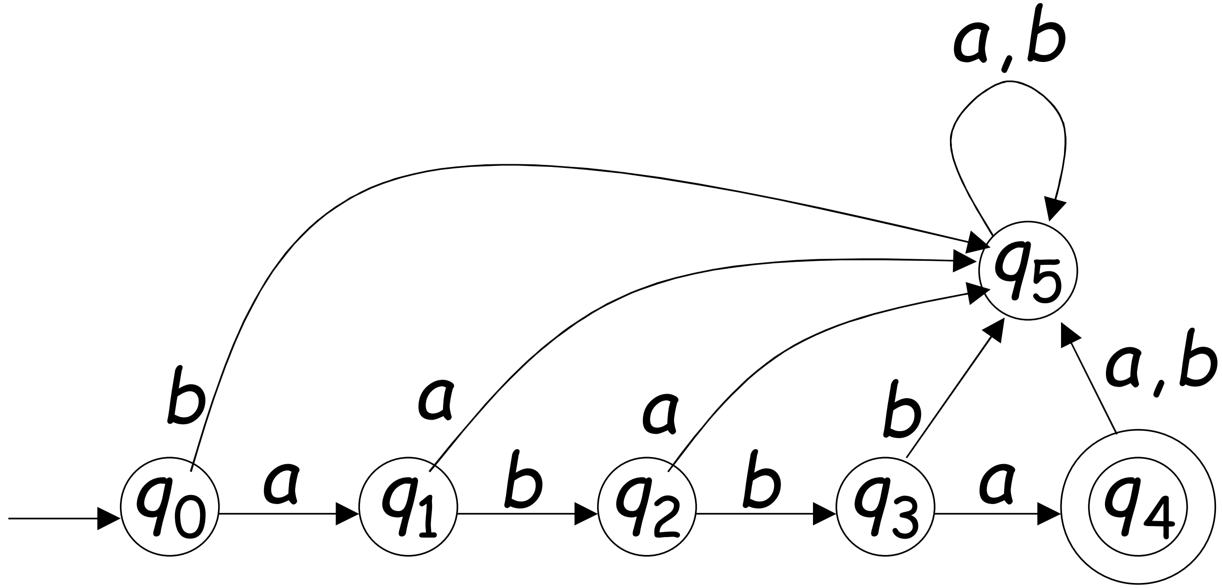
δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5



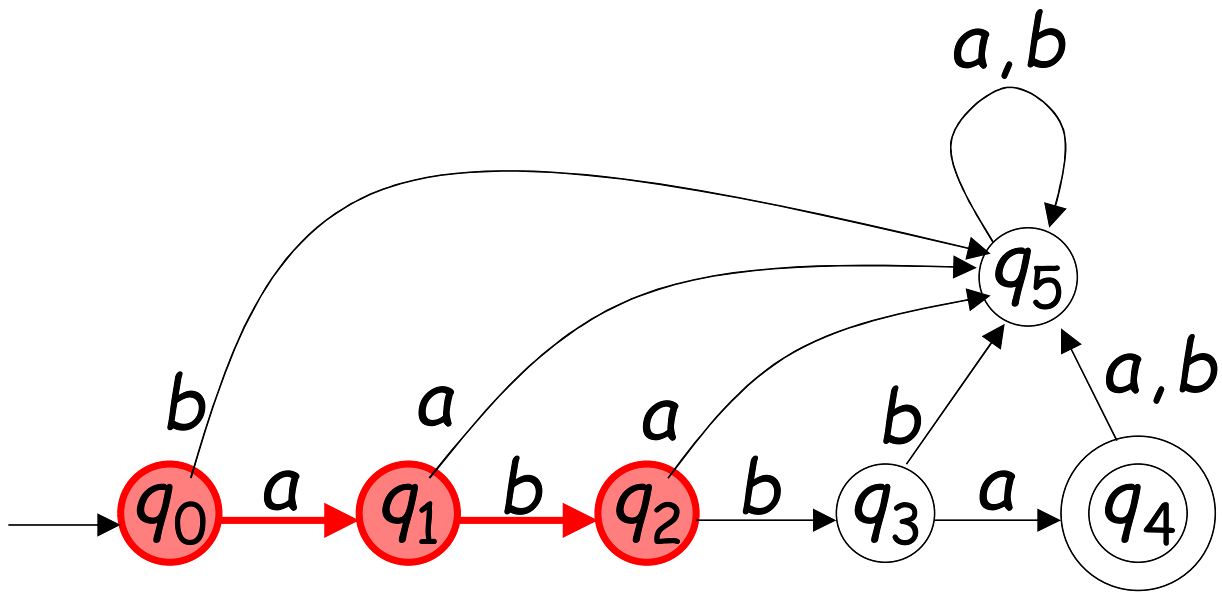
Extended Transition Function δ^*

- $$\delta^* : Q \times \Sigma^* \rightarrow Q$$

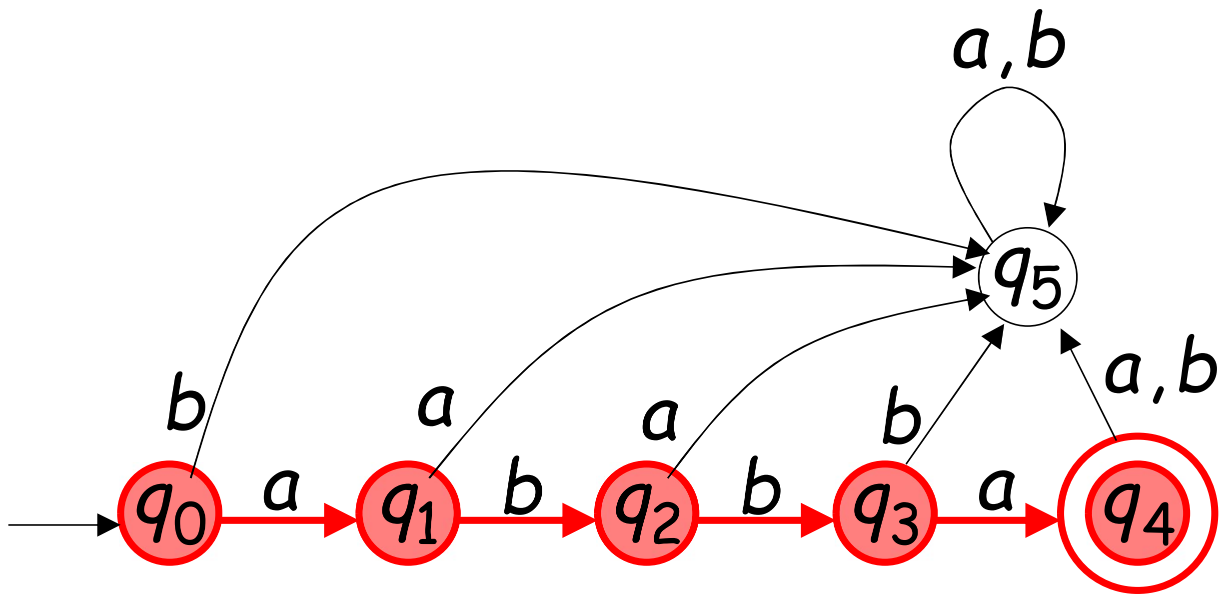
Note that the second argument of function is a string



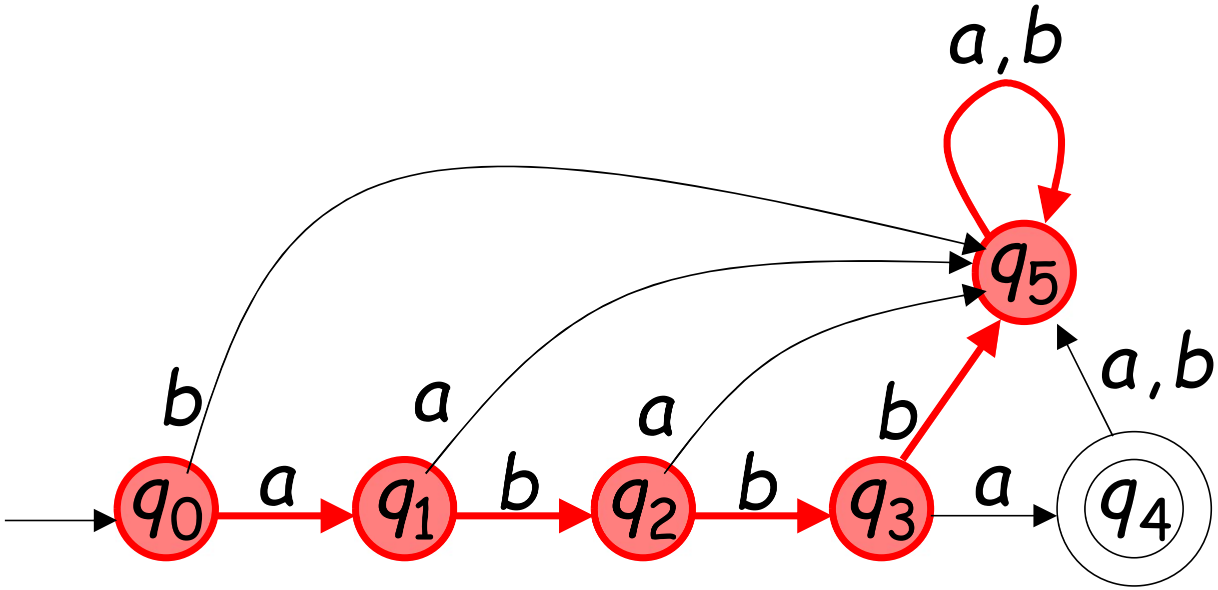
$$\delta^*(q_0, ab) = q_2$$



$$\delta^*(q_0, abba) = q_4$$

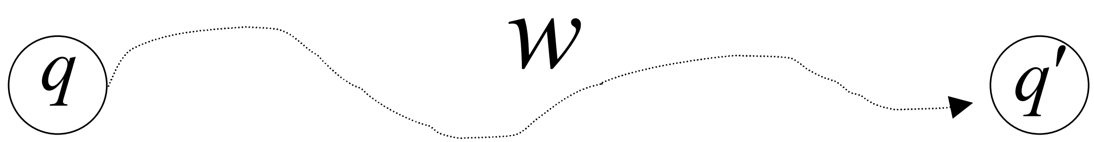


$$\delta^*(q_0, abbbaa) = q_5$$

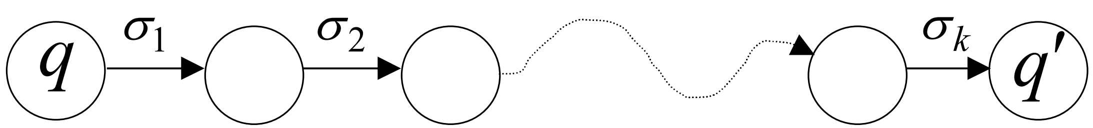


Observation: There is a walk from q to q' with label w

$$\delta^*(q, w) = q'$$

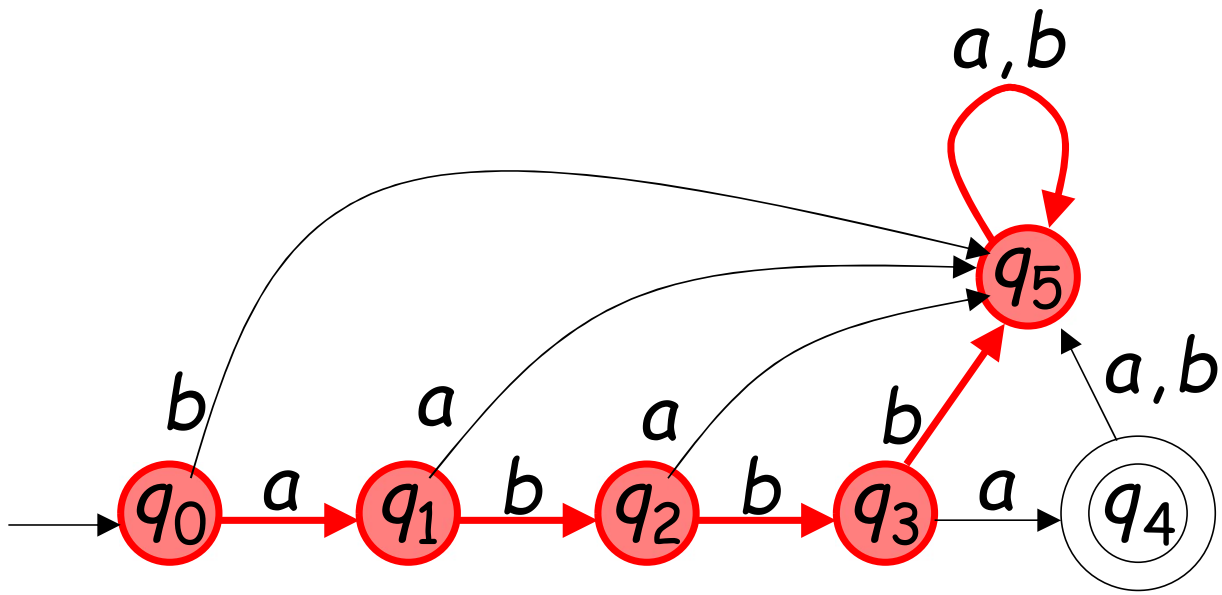


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



Example: There is a walk from q_0 to q_5 with label $abbbaa$

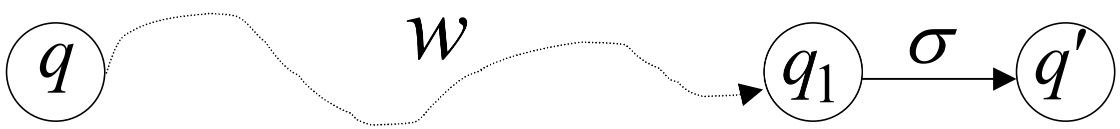
$$\delta^*(q_0, abbbaa) = q_5$$



Recursive Definition

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

$$\delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$



$$\left.\begin{array}{l}\delta^*(q,w\sigma)=q' \\ \\ \delta(q_1,\sigma)=q'\end{array}\right\} \Rightarrow \delta^*(q,w\sigma)=\delta(q_1,\sigma)$$

$$\left.\begin{array}{l}\delta^*(q,w\sigma)=\delta(q_1,\sigma) \\ \\ \delta^*(q,w)=q_1\end{array}\right\} \Rightarrow \delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$

$$\delta^*(q_0, ab) =$$

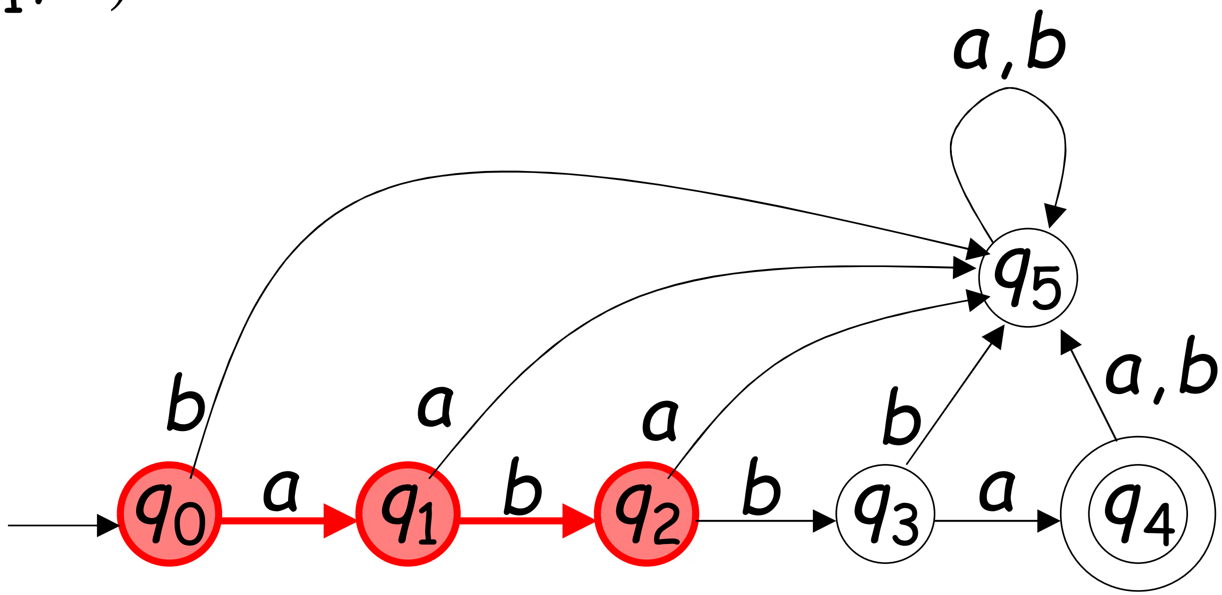
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

$$\delta(\delta(q_0, a), b) =$$

$$\delta(q_1, b) =$$

$$q_2$$



JFLAP

Download and install

https://www.jflap.org/

Run

Outline

- Last week
- Introduction - Finite Automata, Types
- Finite Acceptors - Deterministic
- Regular Language
- Finite Acceptors - Nondeterministic

Languages Accepted by DFAs

- Take DFA M

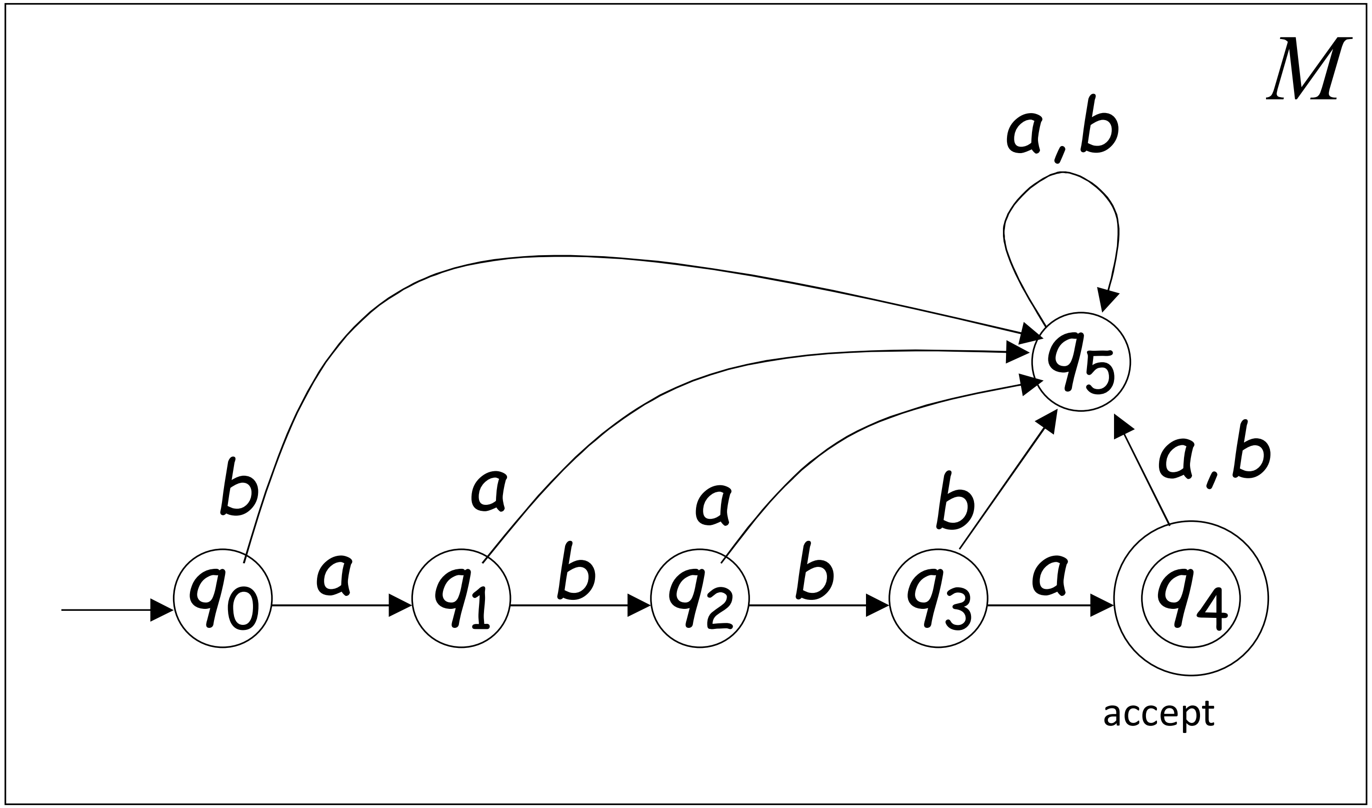
- Definition:
  - The language $L(M)$ contains all input strings accepted by M

$L(M)$ = { strings that drive M to a final state}

Example

$$L(M) = \{abba\}$$

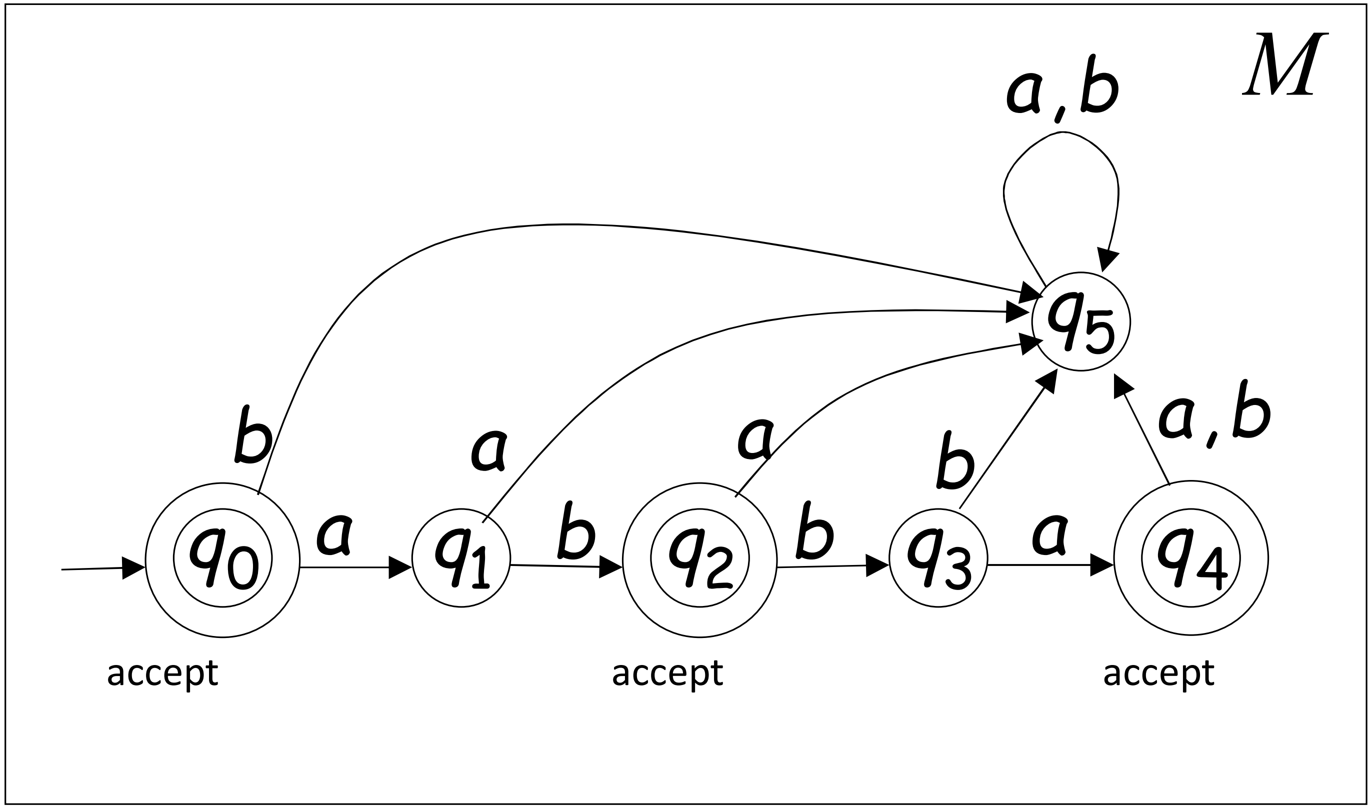
•



Another Example

$$L(M) = \{\lambda, ab, abba\}$$

-

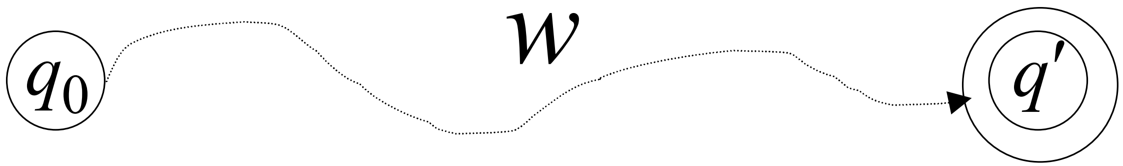


Formally

- For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

- Language accepted by M:

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$



$q' \in F$

Observation

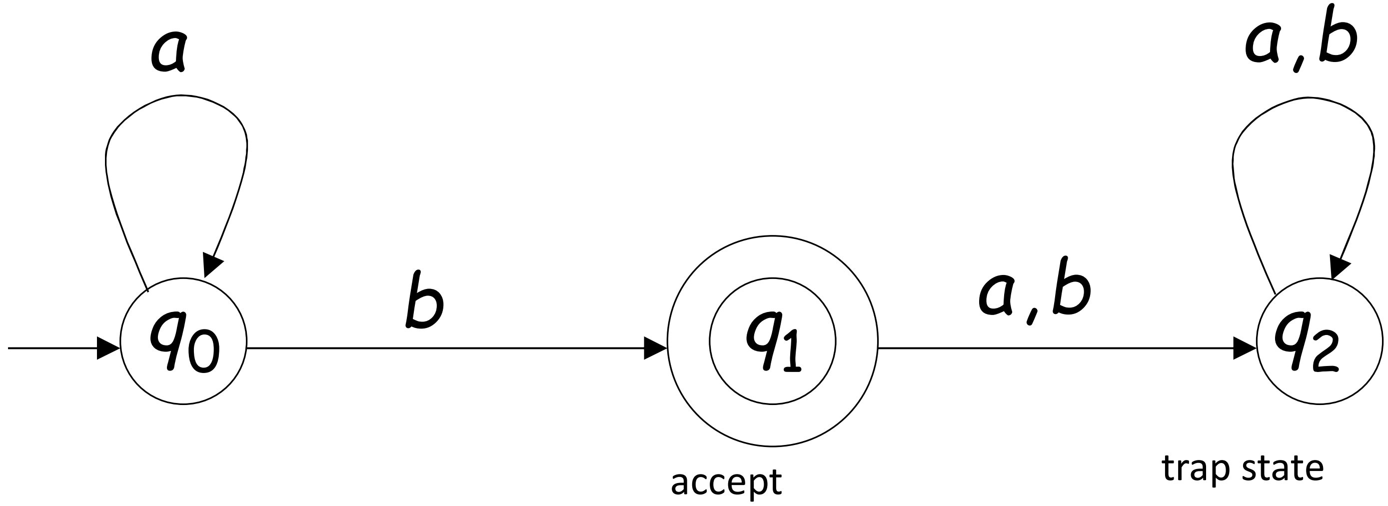
- Language rejected by M :

$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$

$$q_0 \xrightarrow{w} q' \qquad q' \notin F$$

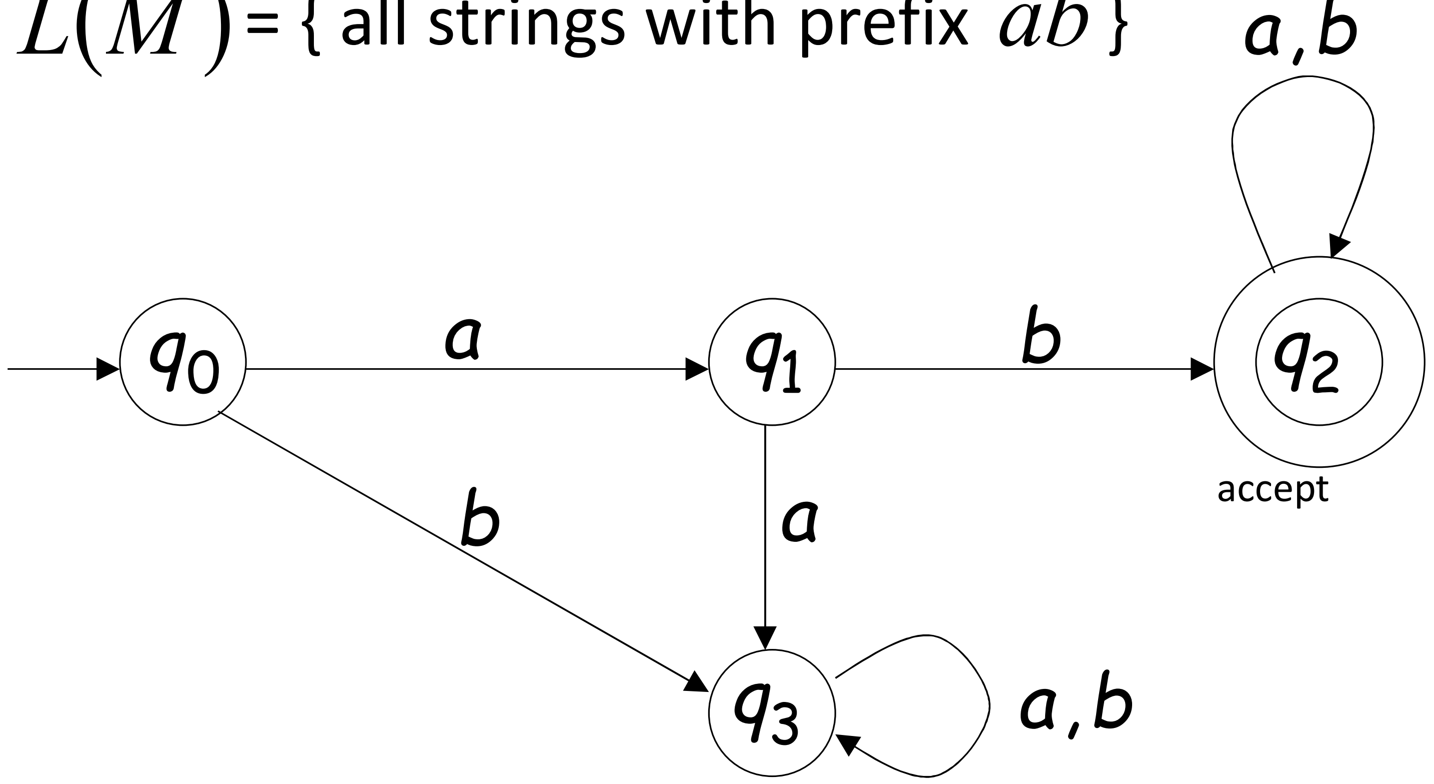
More Examples

$$L(M) = \{a^n b : n \geq 0\}$$



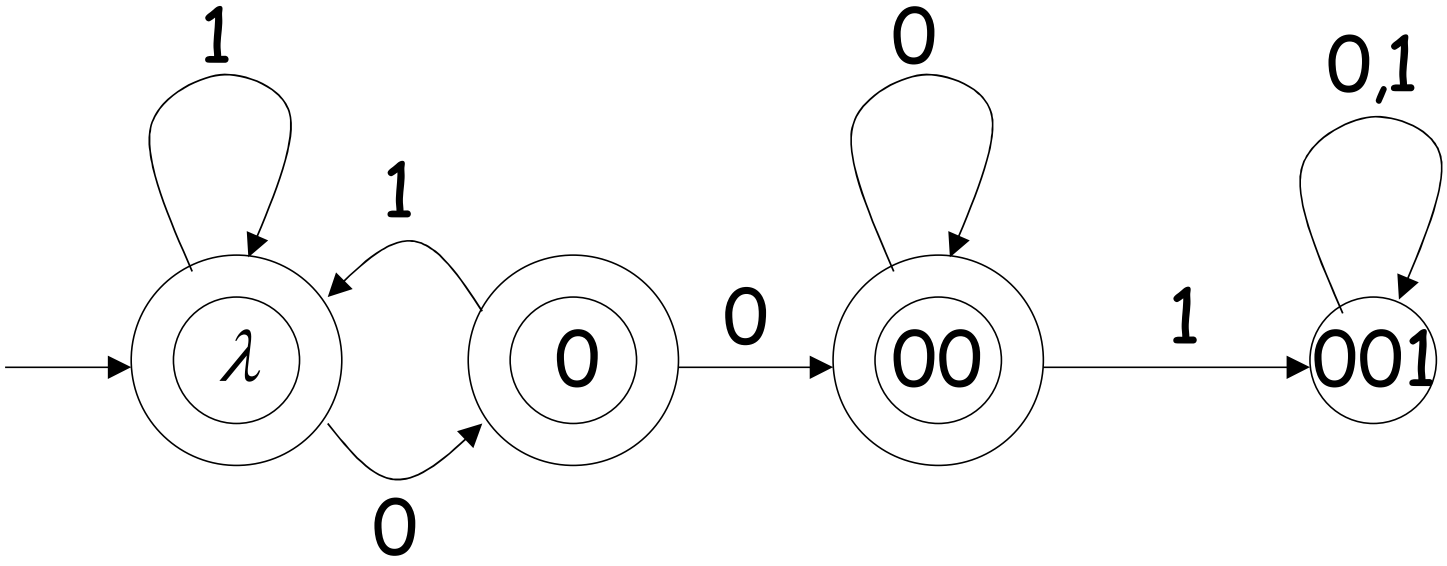
More Examples

$L(M) =$ { all strings with prefix ab }



More Examples

$L(M)$ = { all strings without substring 001 }



Regular Languages

- A language L is regular if there is a DFA M such that $L = L(M)$

- All regular languages form a language family

Regular Language Examples

$\{abba\}$ $\{\lambda, ab, abba\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab}

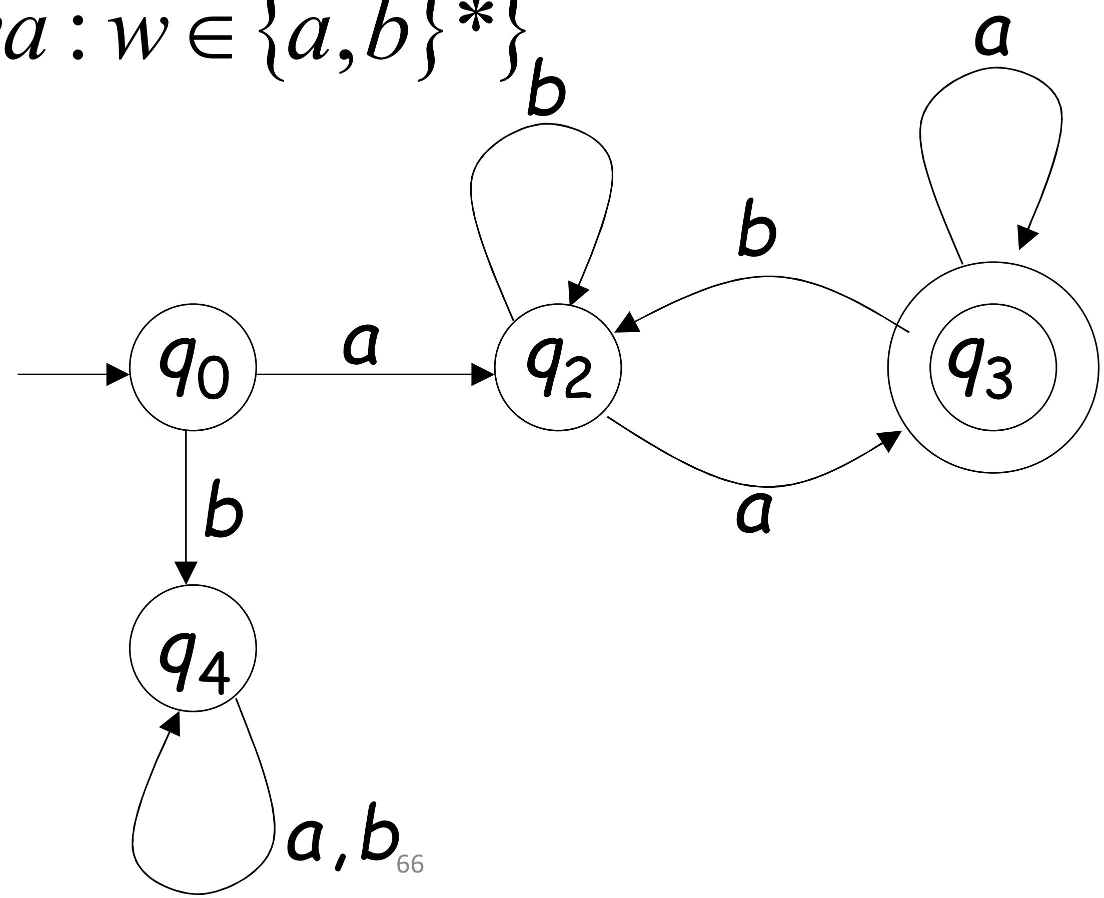
{ all strings without substring 001}

There exist automata that accept these Languages (see previous slides).

Regular Language Example

- The language $L = \{awa : w \in \{a,b\}^*\}$

is regular:



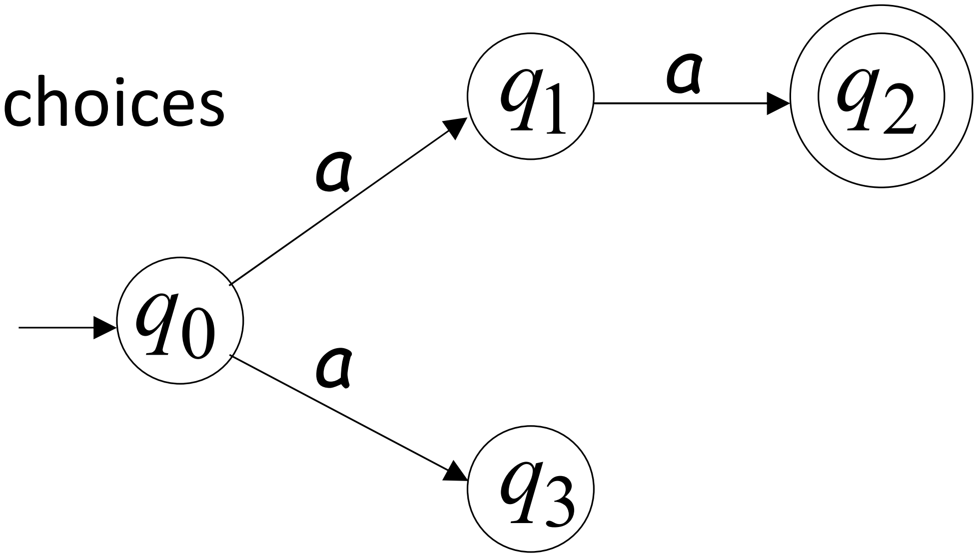
Outline

- Last week
- Introduction - Finite Automata, Types
- Finite Acceptors - Deterministic
- Regular Language
- Finite Acceptors - Nondeterministic

Nondeterministic Finite Accepter (NFA)

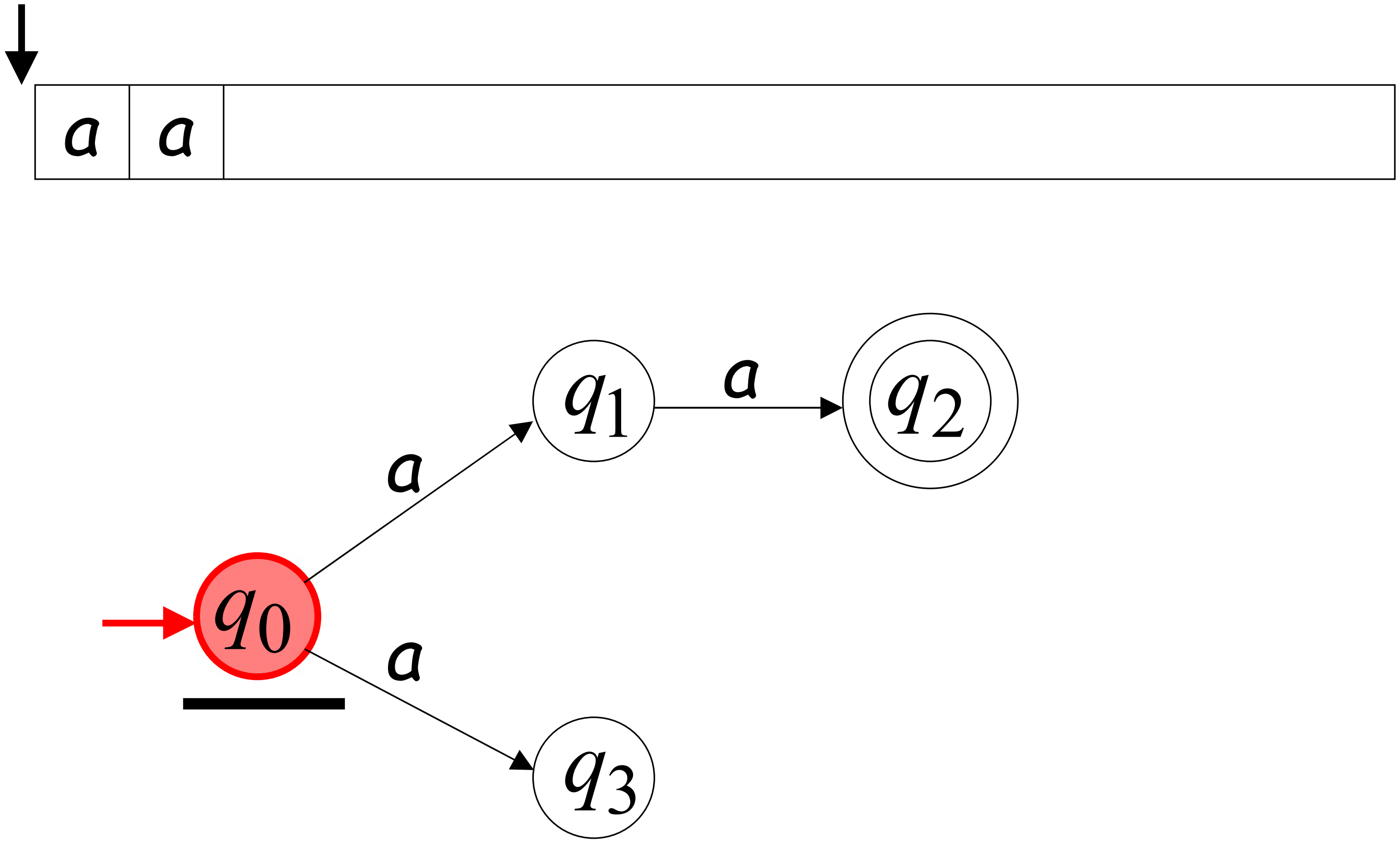
Alphabet = $\{a\}$

Two choices

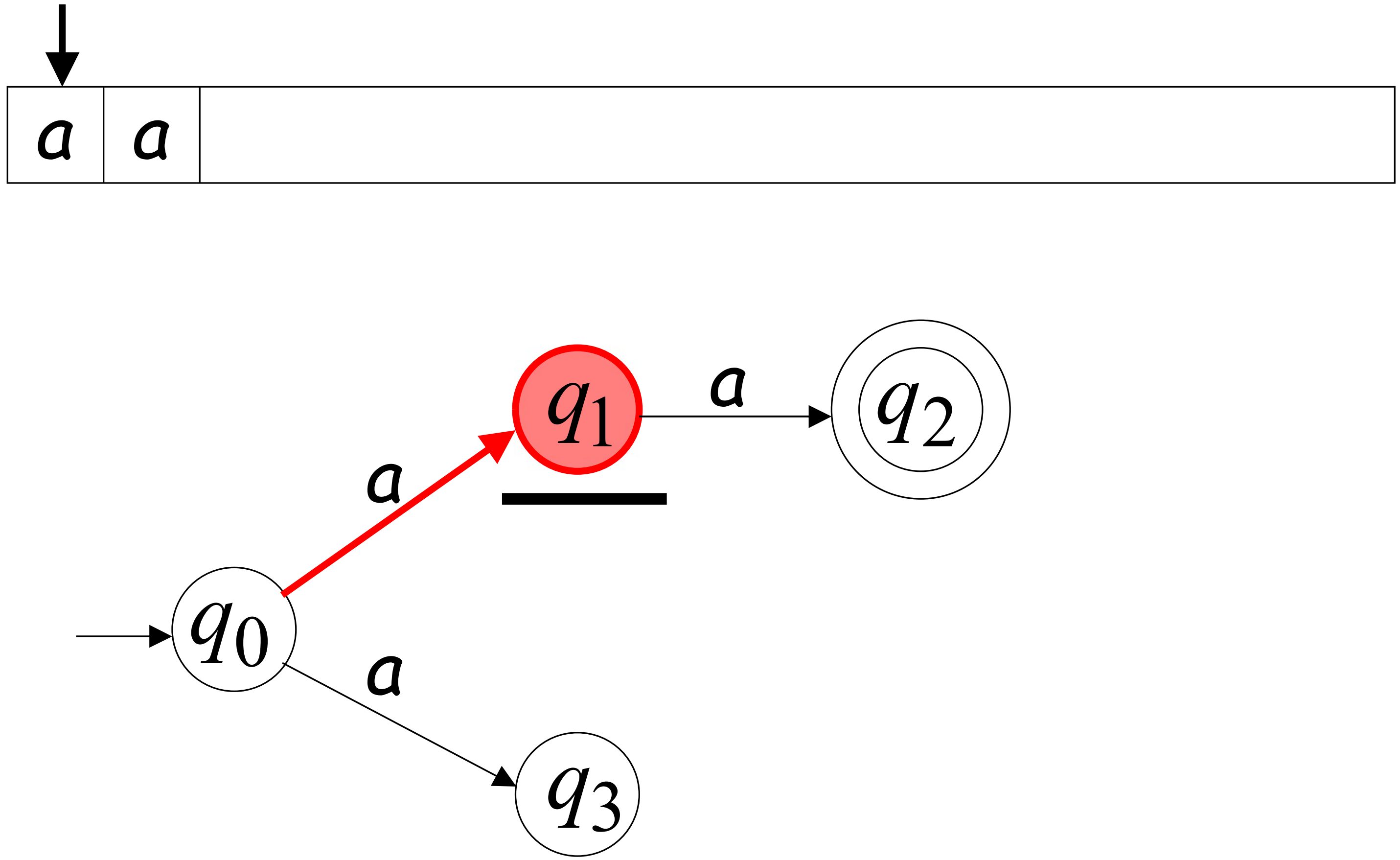


No transition

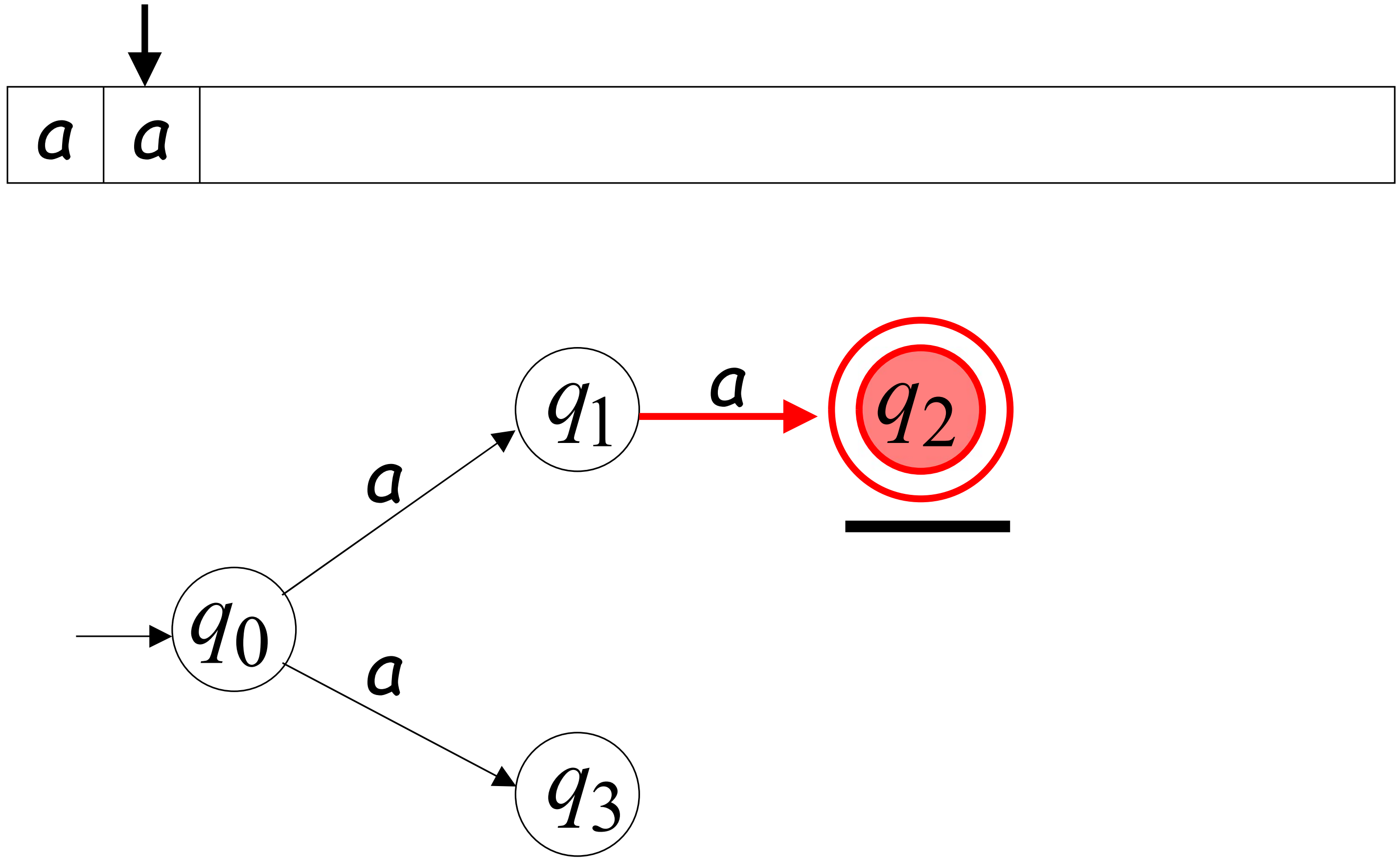
First Choice



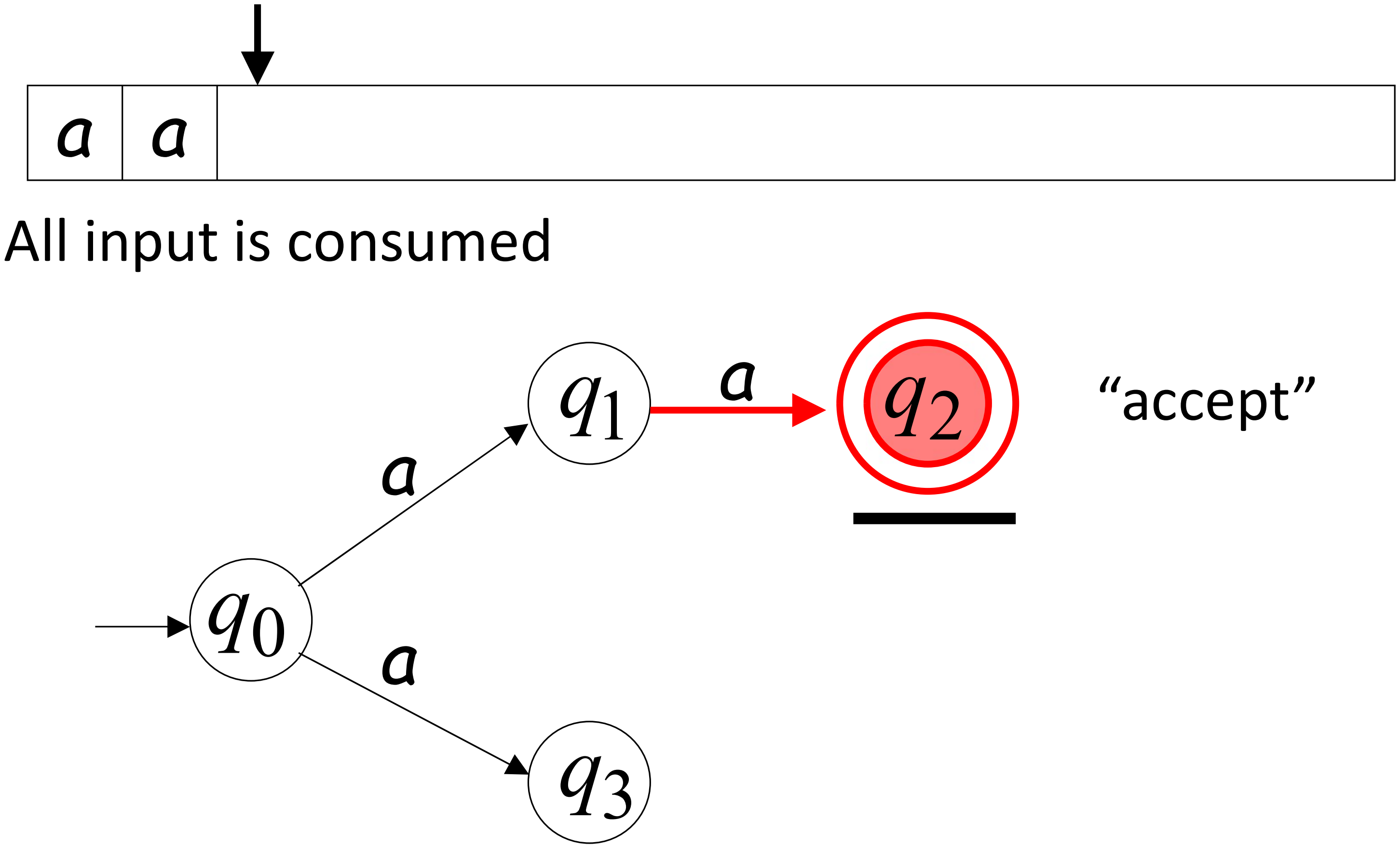
First Choice



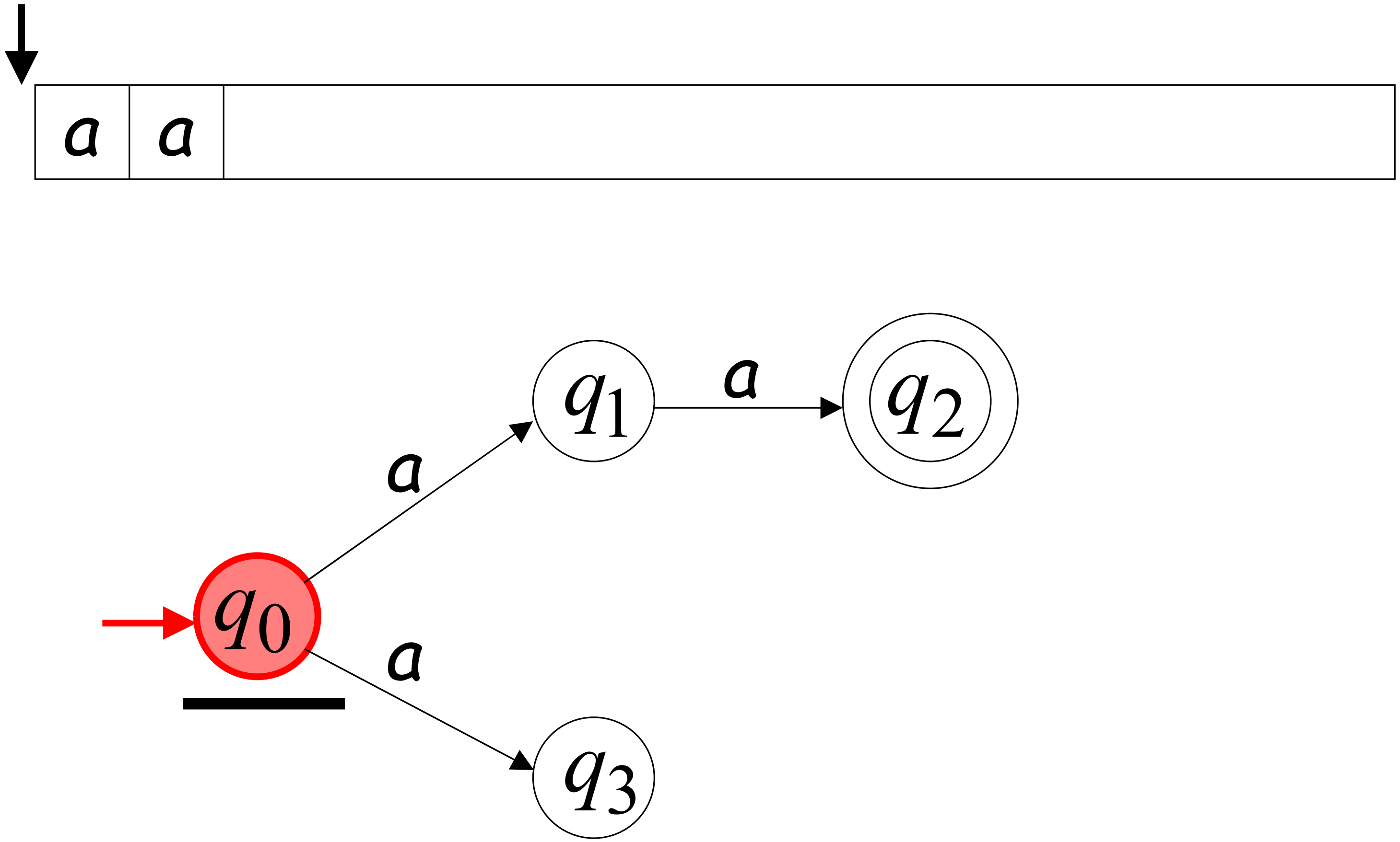
First Choice



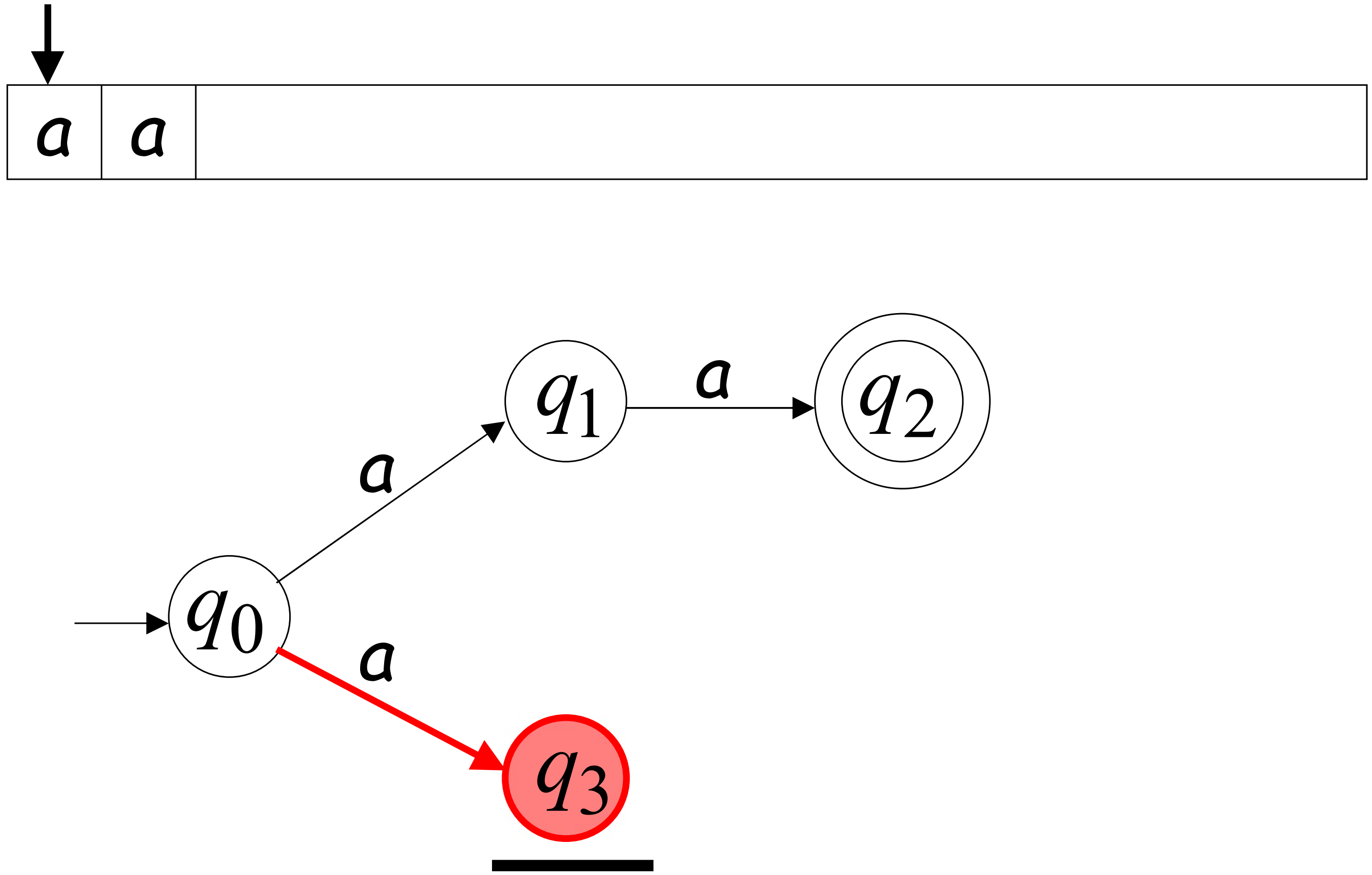
First Choice



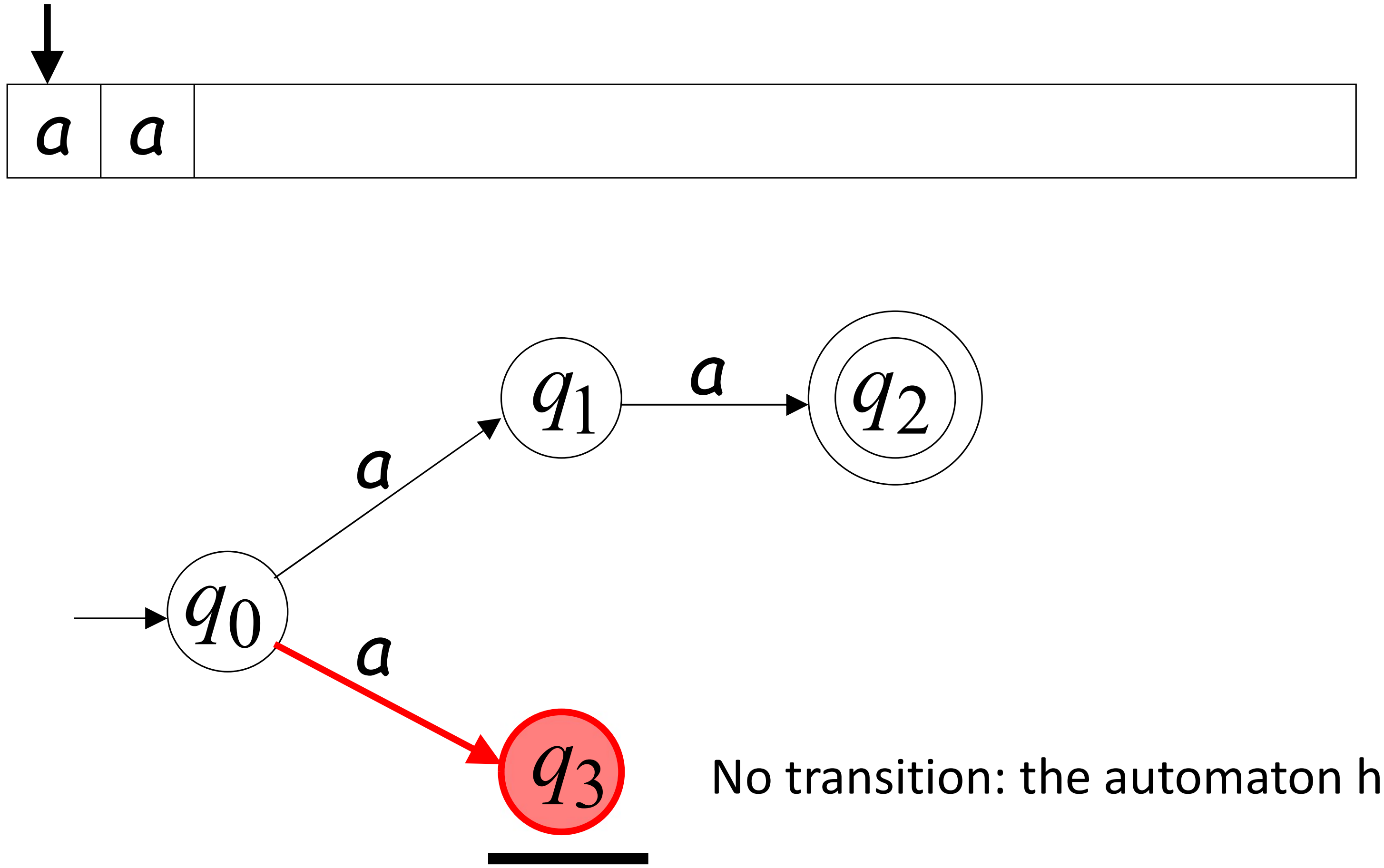
Second Choice



Second Choice

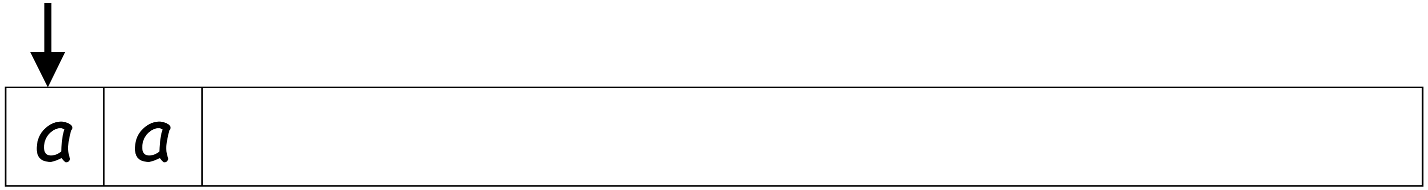


Second Choice

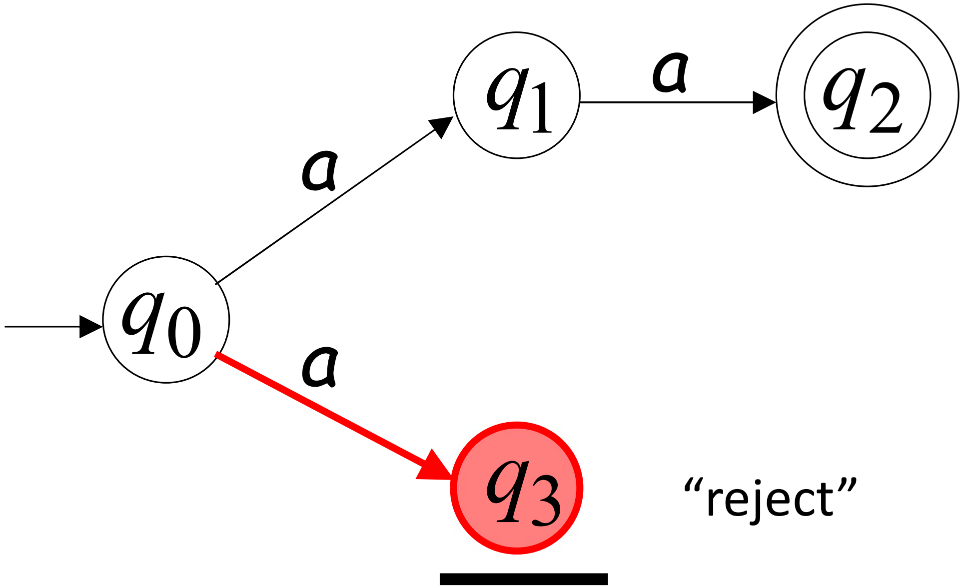


No transition: the automaton hangs

Second Choice



Input cannot be consumed



"reject"

An NFA accepts a string when:

- there is at least one computation of the NFA that accepts the string

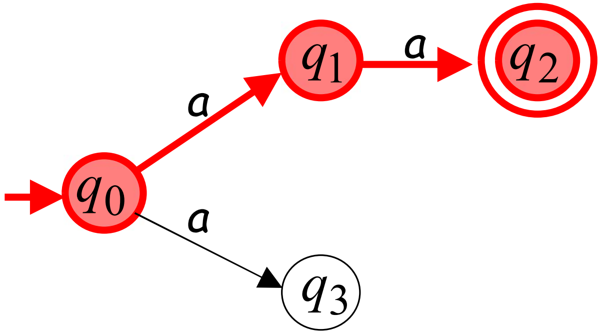
AND

- all the input is consumed and the automaton is in a final state

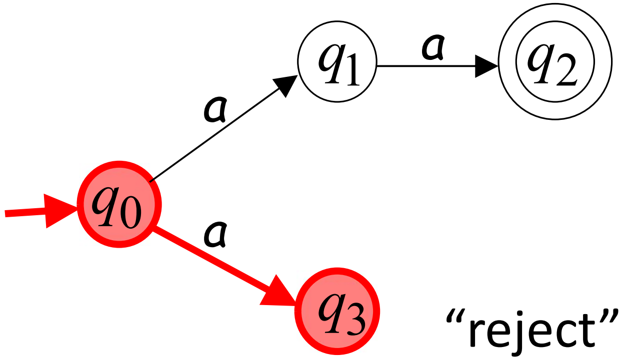
Example

aa is accepted by the NFA:

“accept”

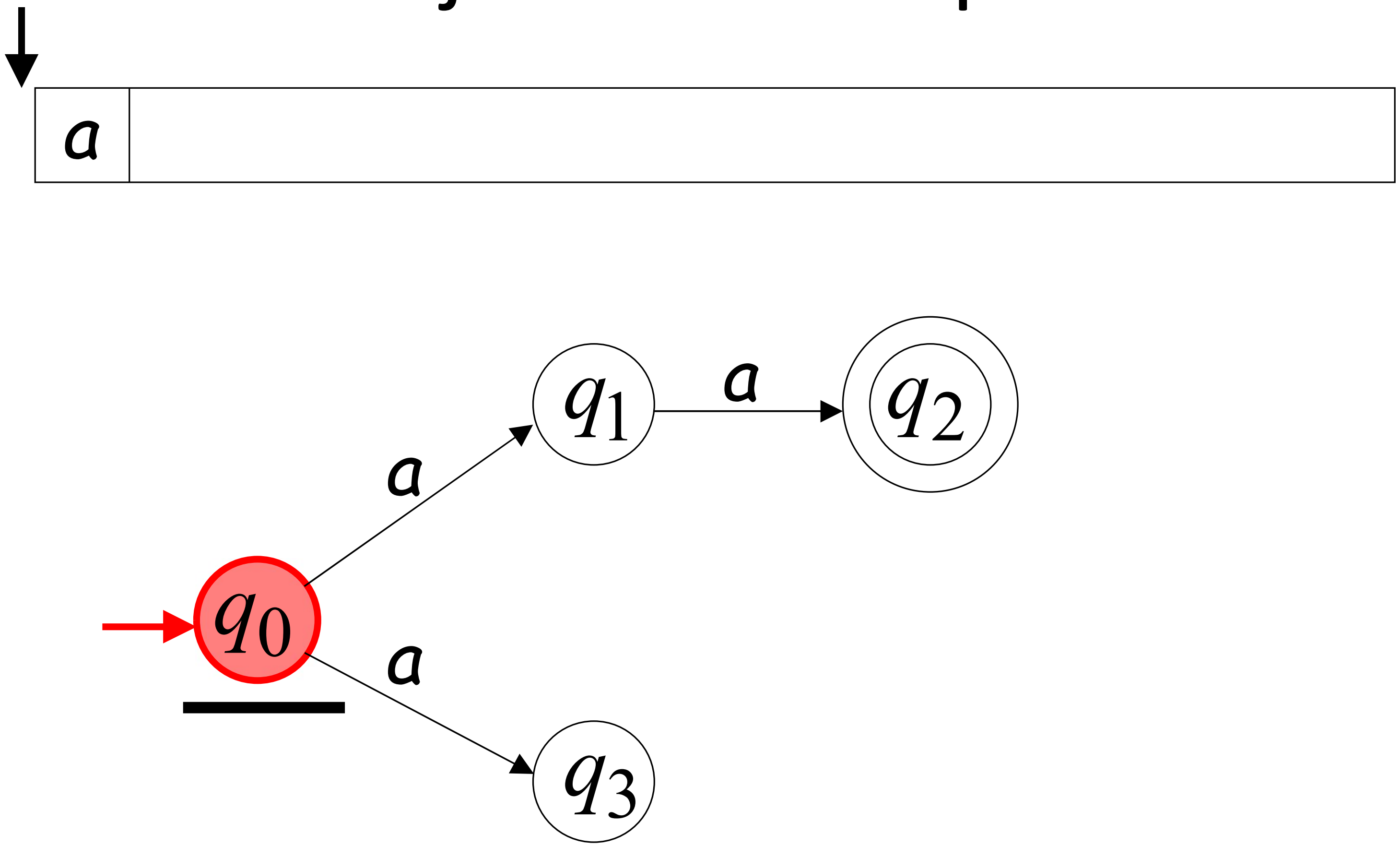


because this
computation
accepts

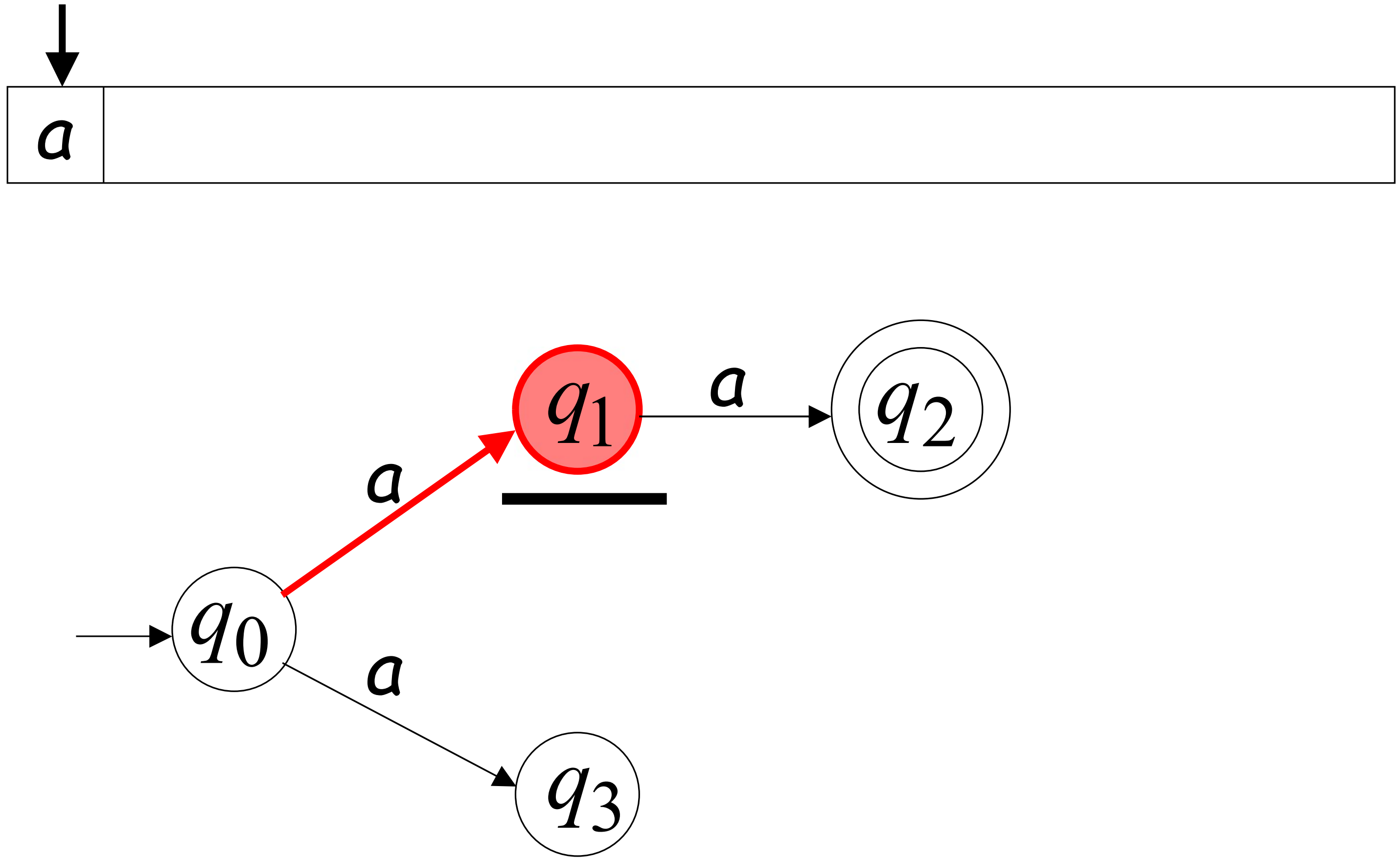


“reject”

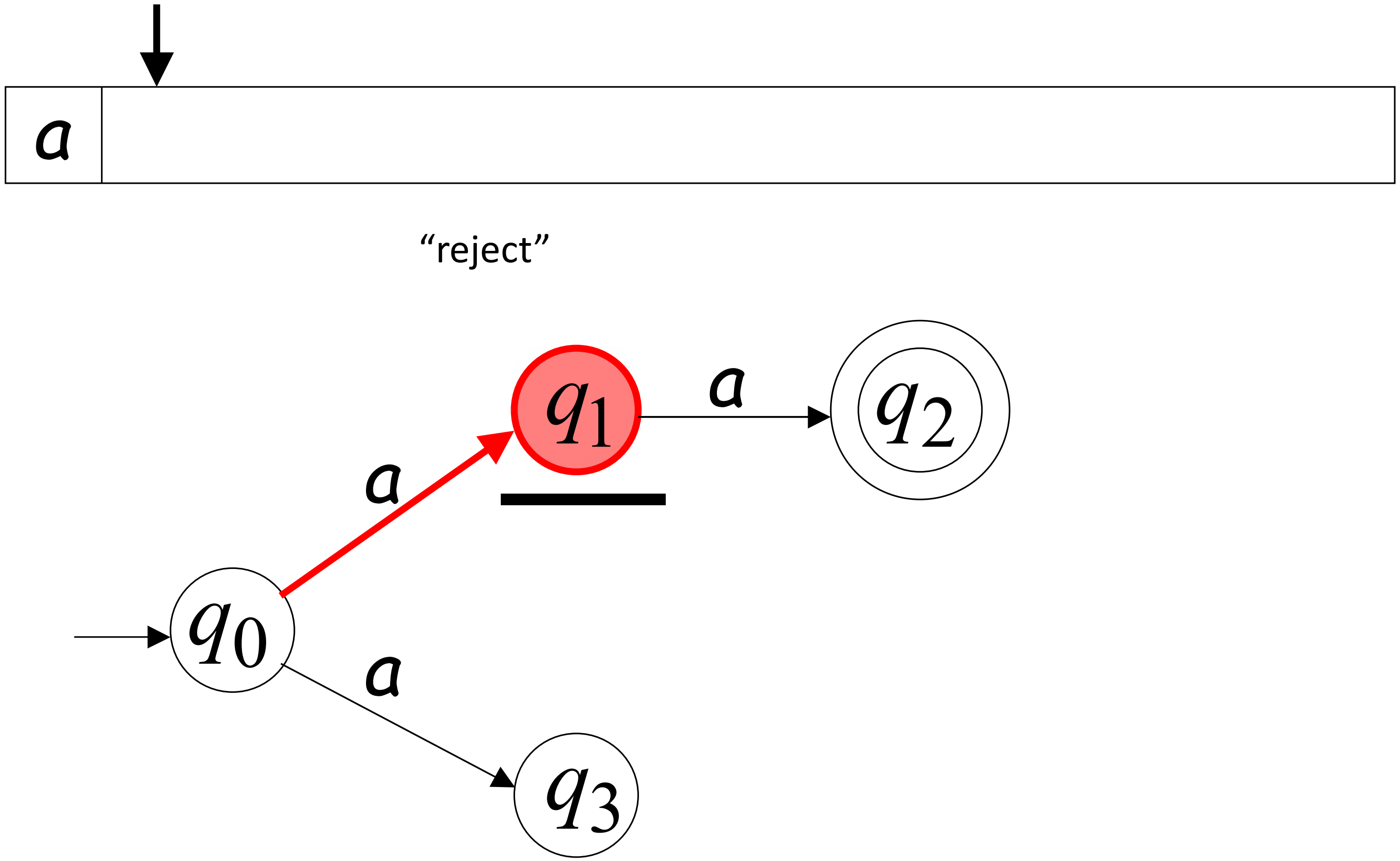
Rejection example



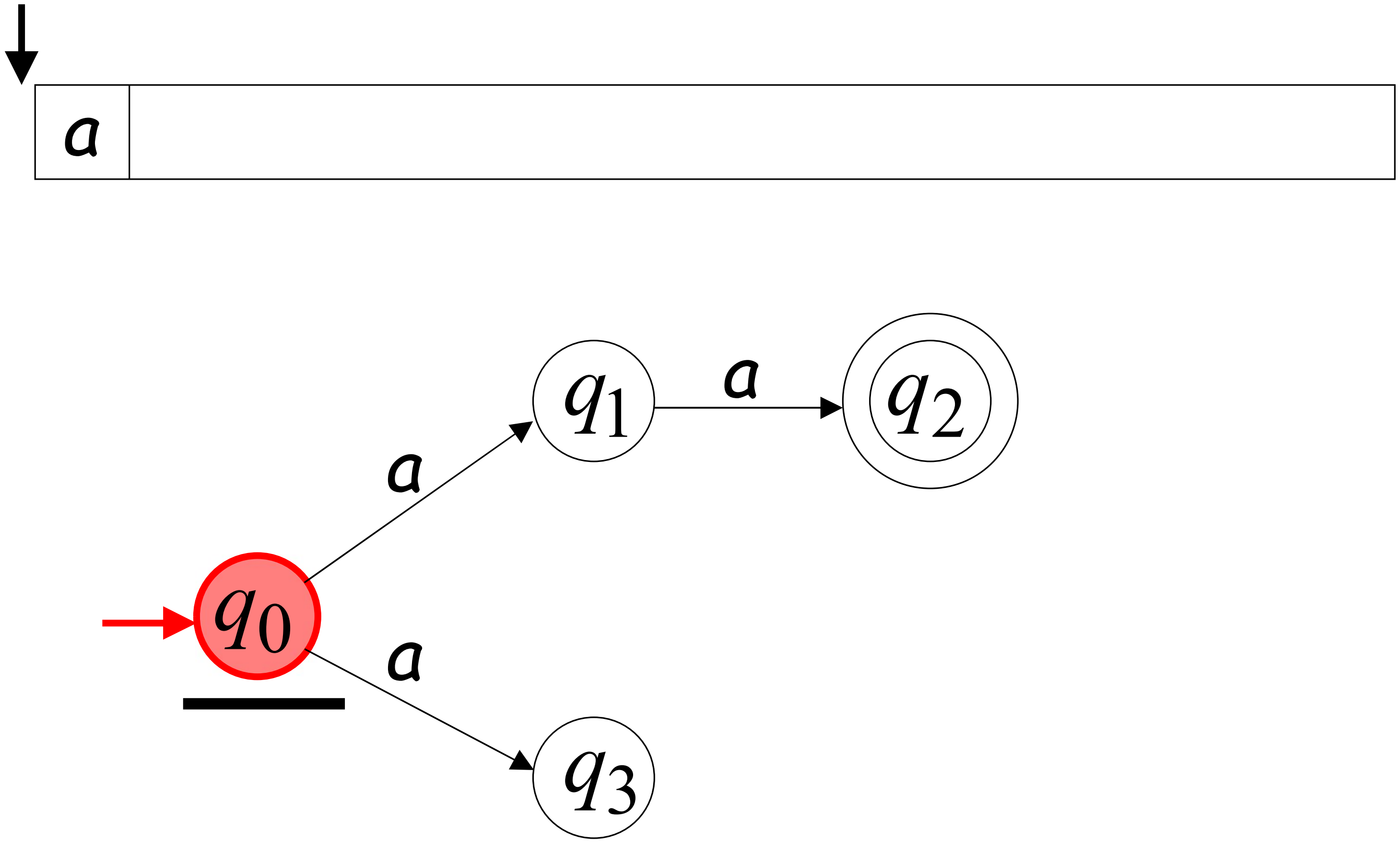
First Choice



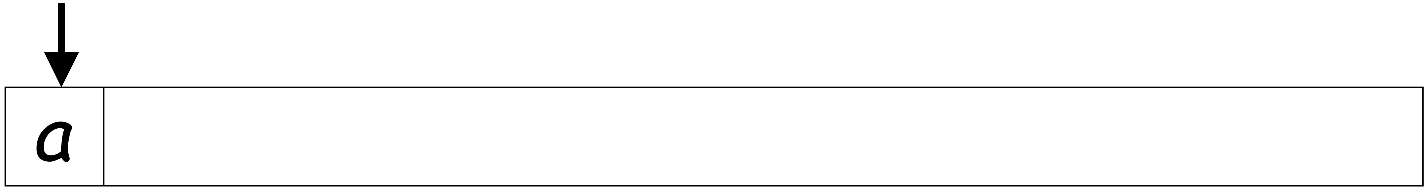
First Choice

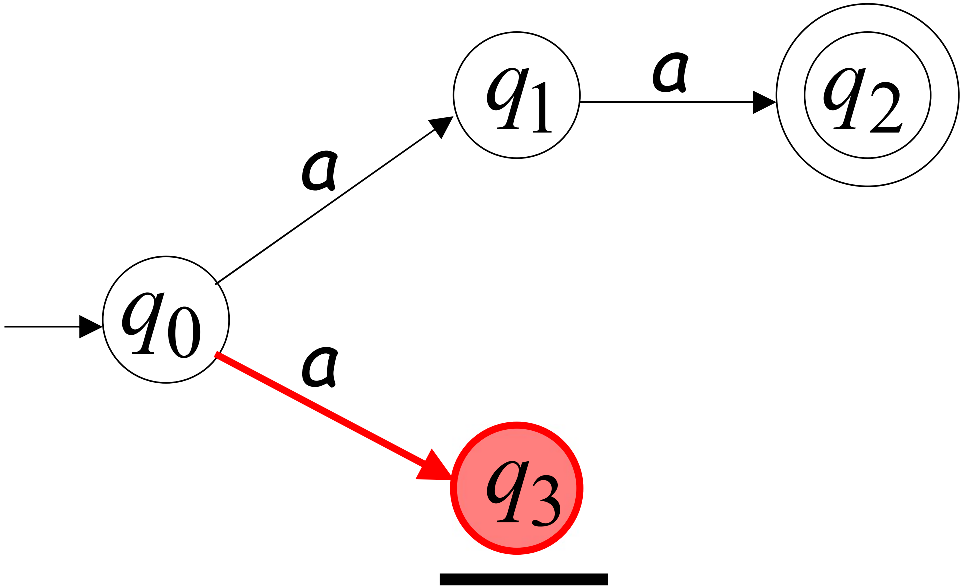


Second Choice

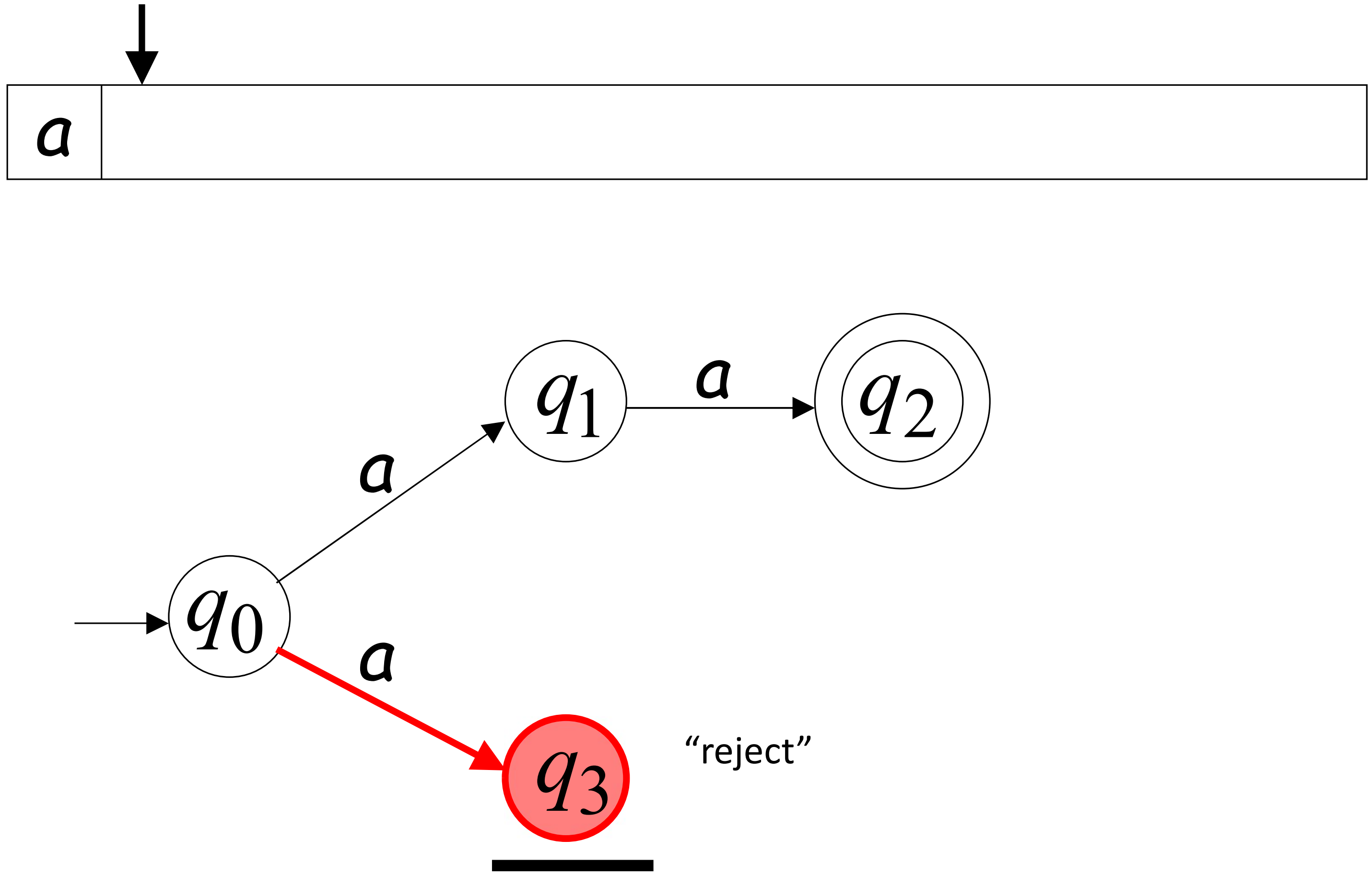


Second Choice





Second Choice



An NFA rejects a string when:

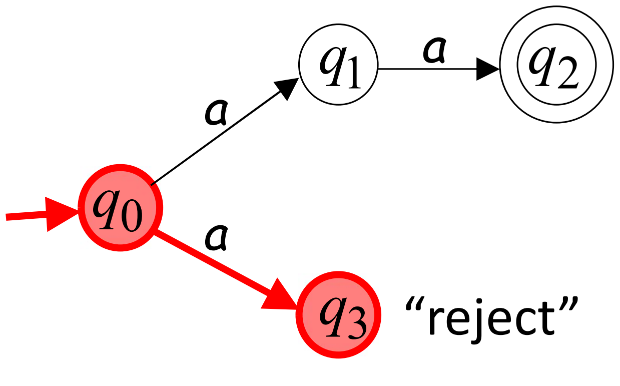
- All the input is consumed and the automaton is in a non-final state

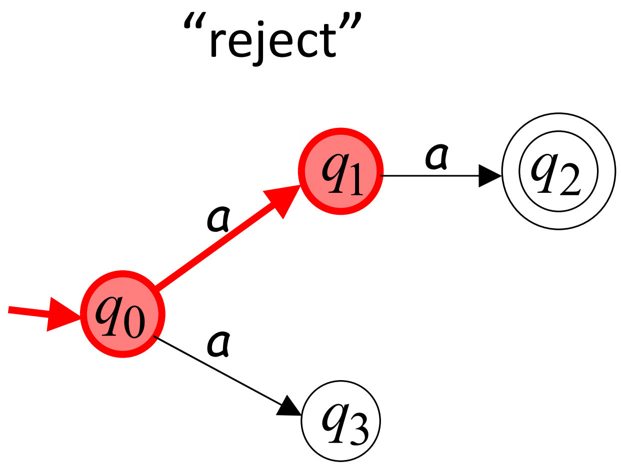
OR

- The input cannot be consumed

Example

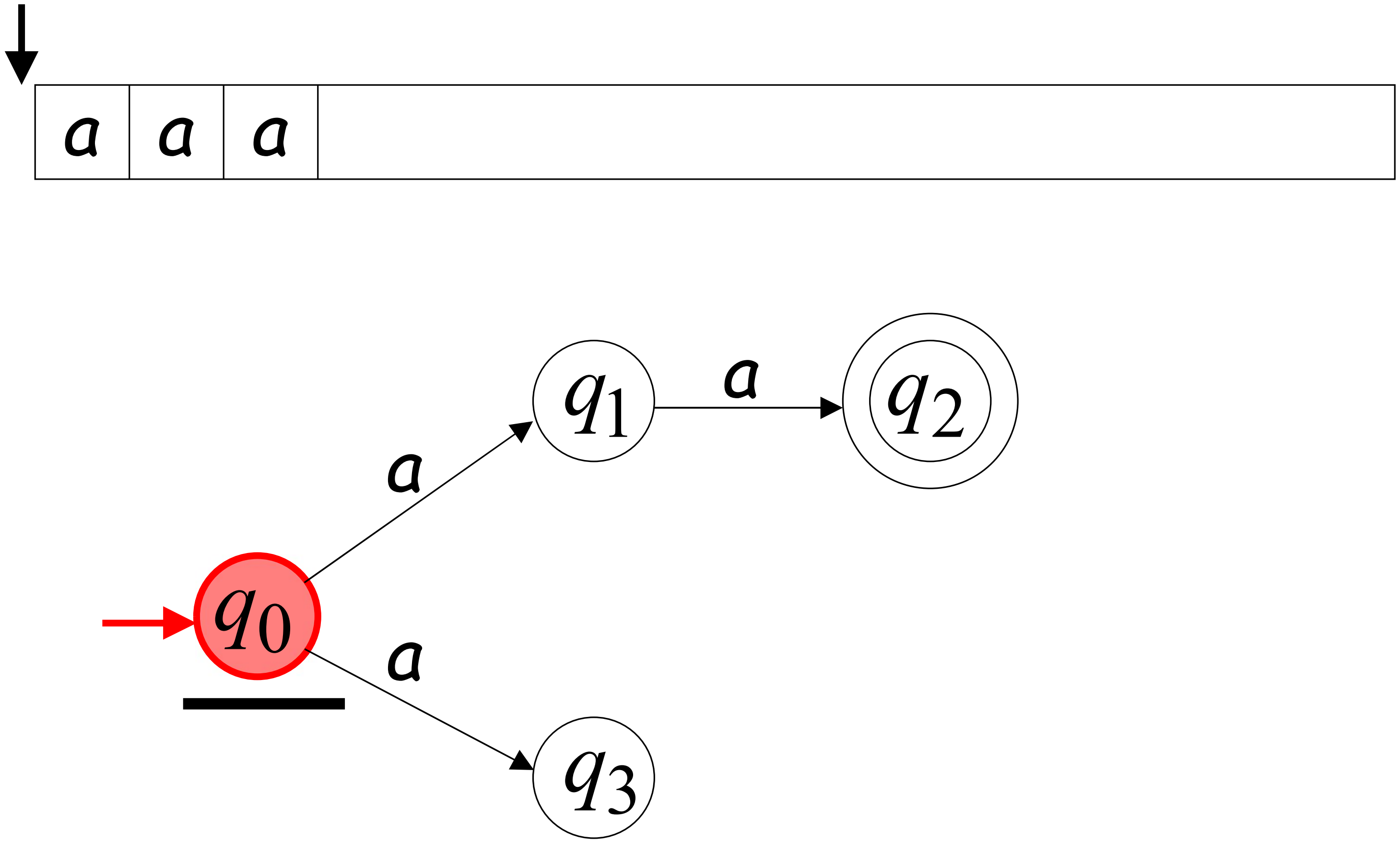
a is rejected by the NFA:



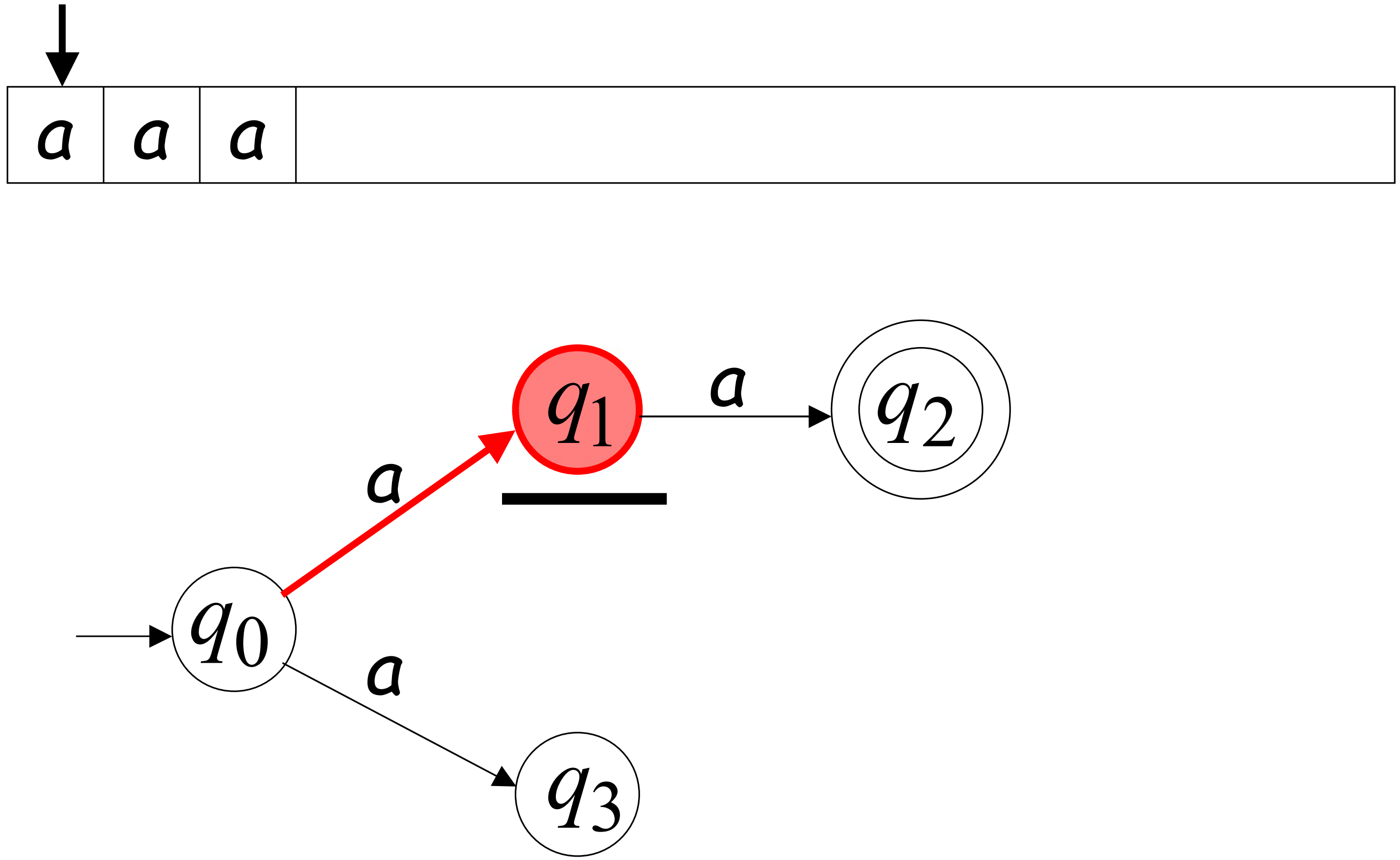


All possible computations lead to rejection

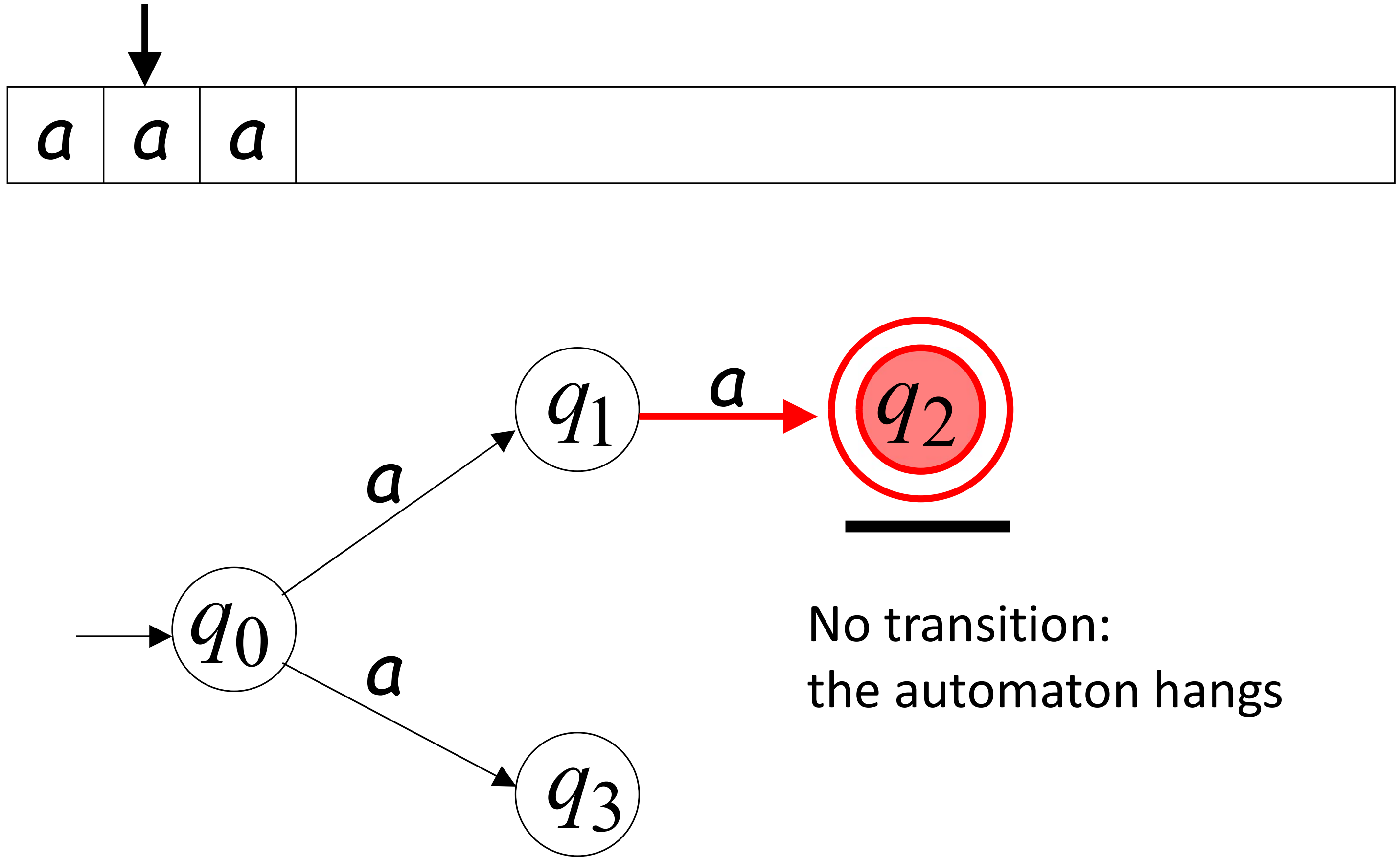
Another Rejection Example



First Choice

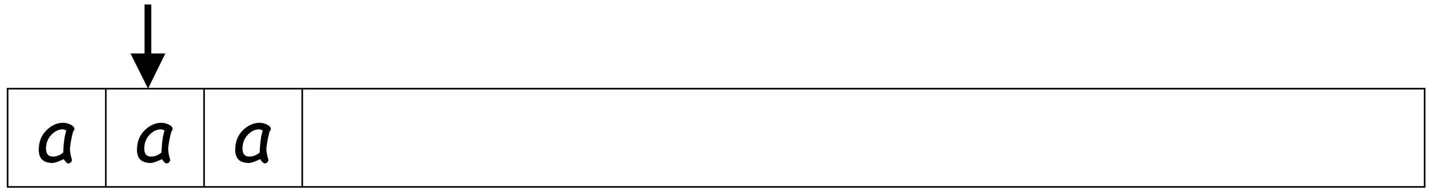


First Choice

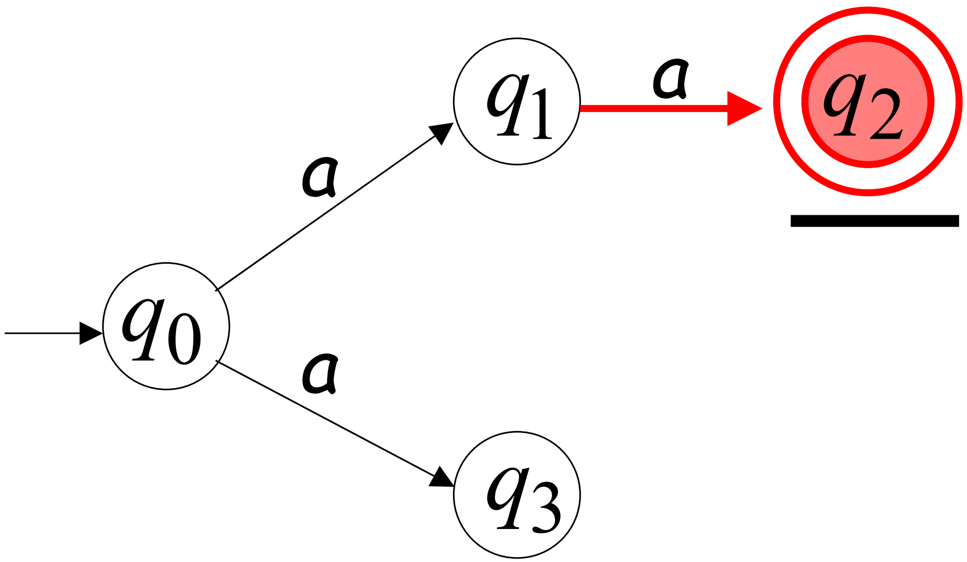


No transition:
the automaton hangs

First Choice

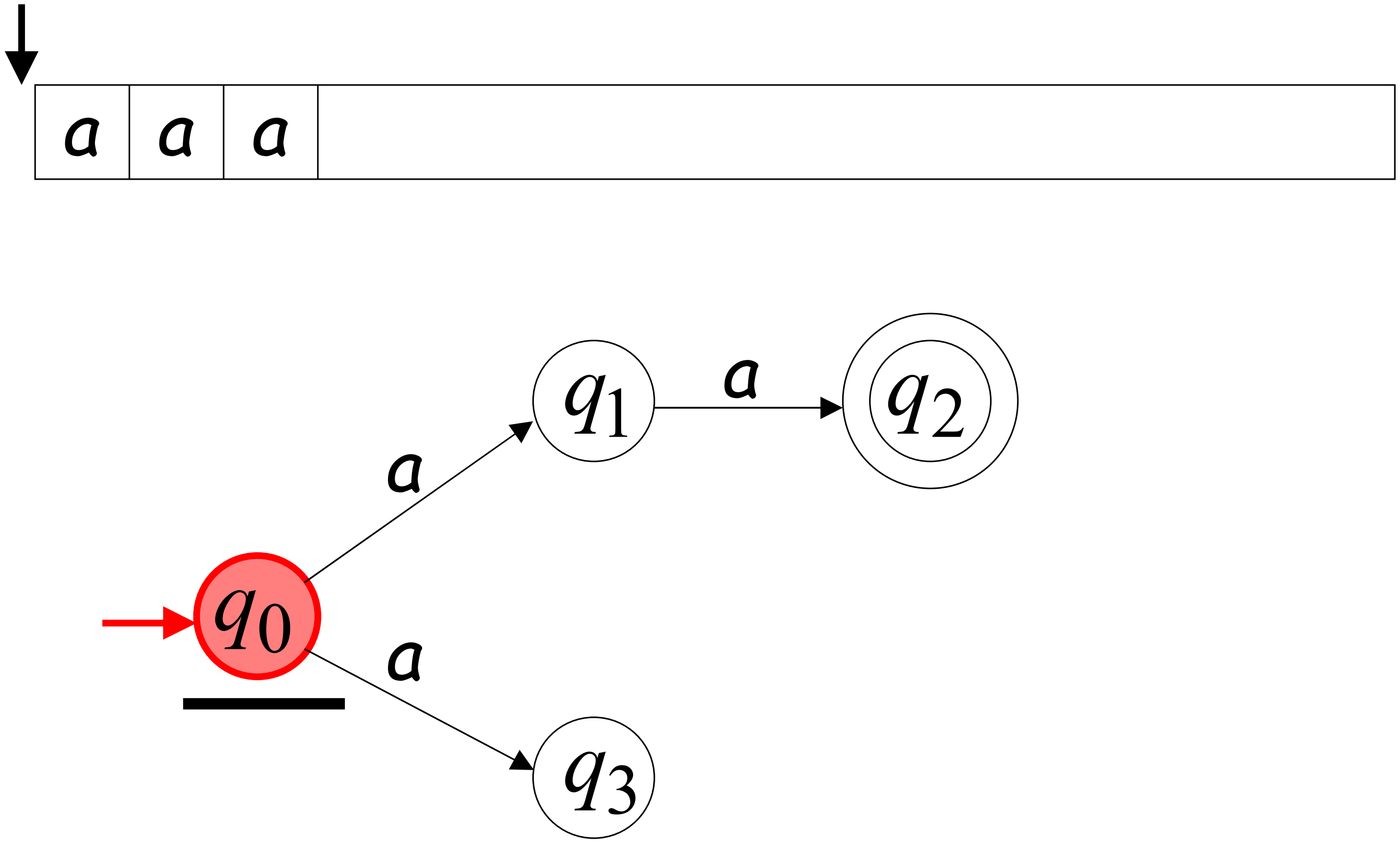


Input cannot be consumed

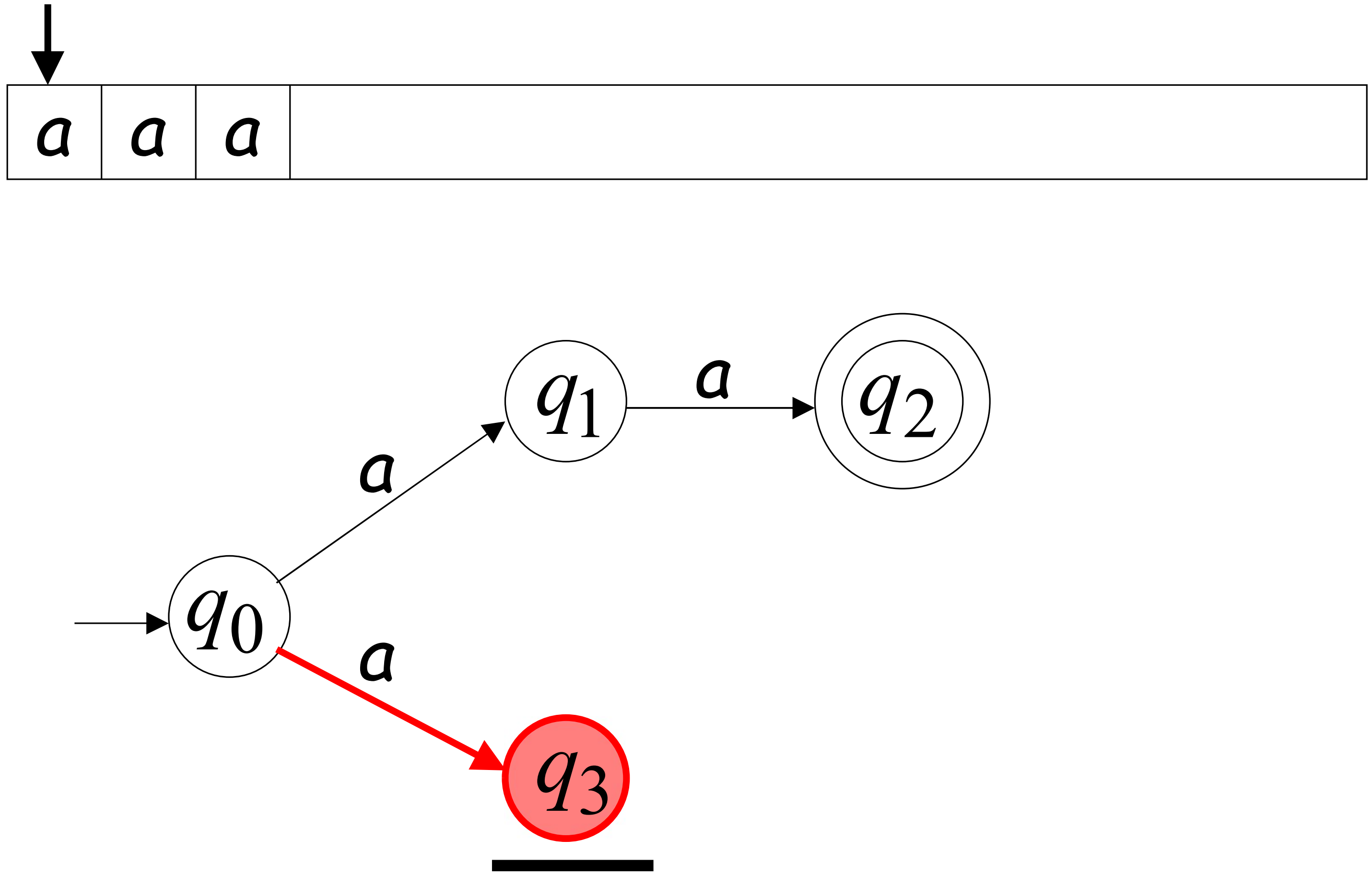


"reject"

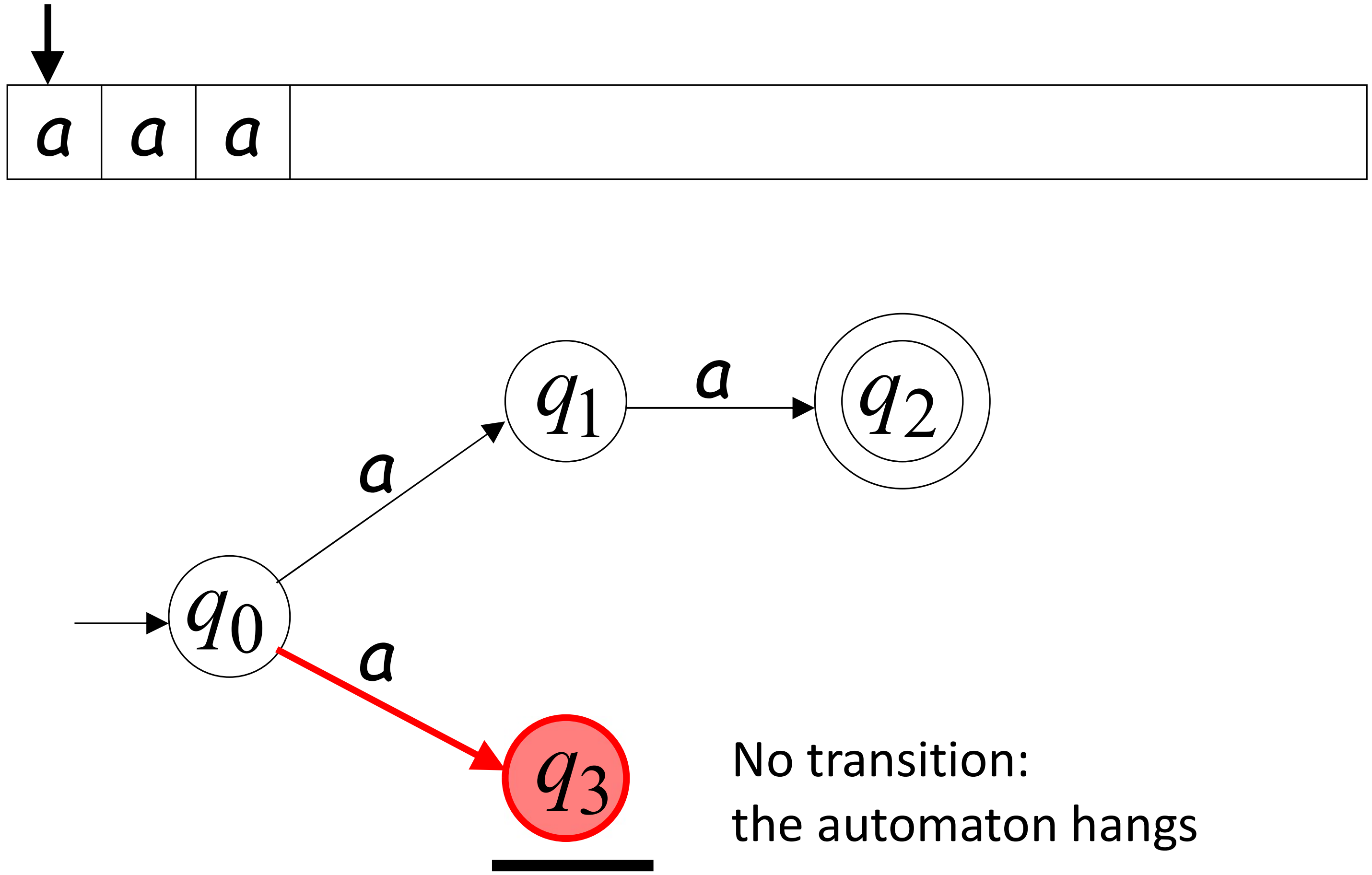
Second Choice



Second Choice

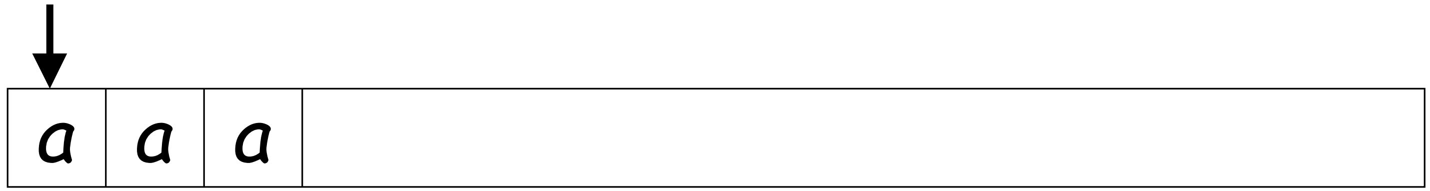


Second Choice

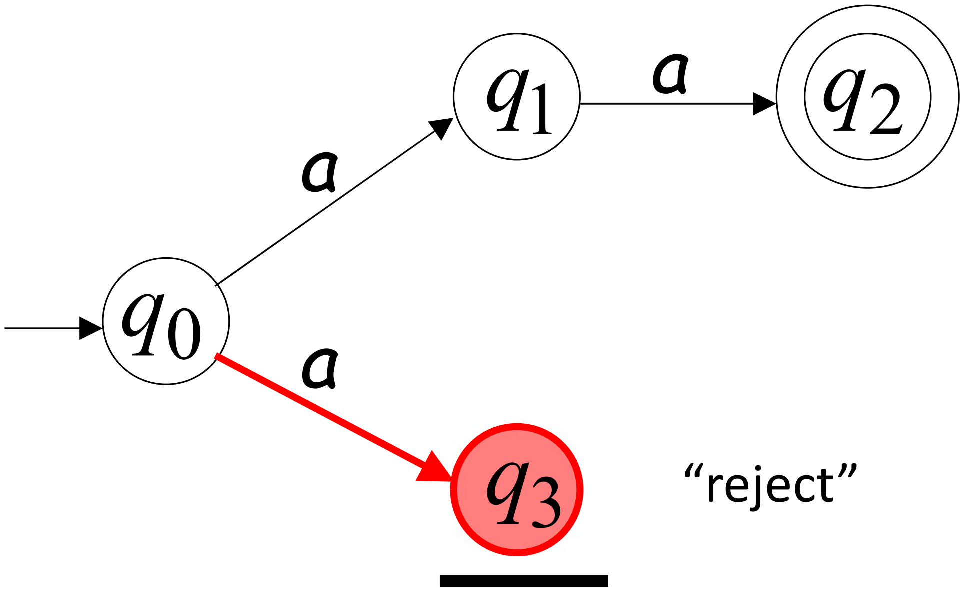


No transition:
the automaton hangs

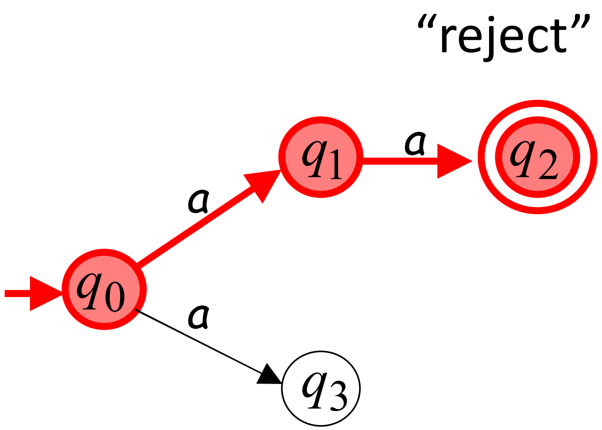
Second Choice

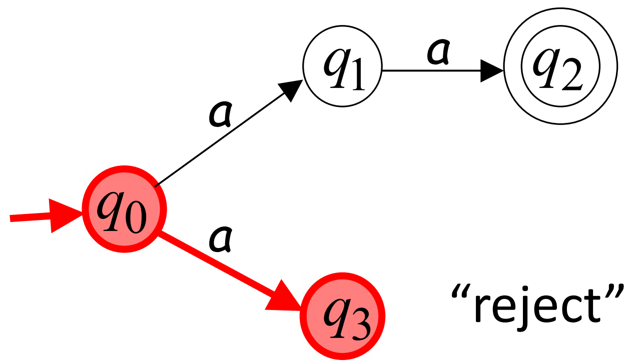


Input cannot be consumed



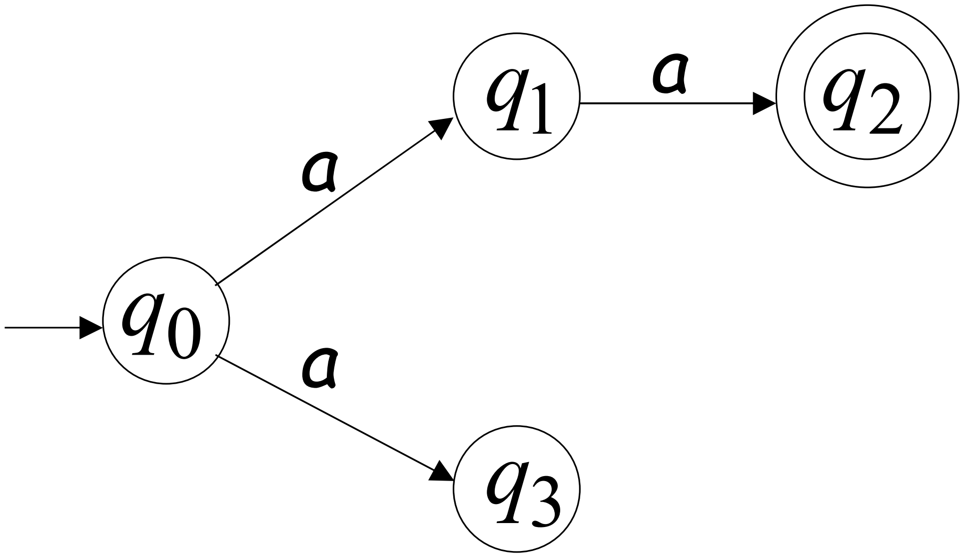
aaa is rejected by the NFA:



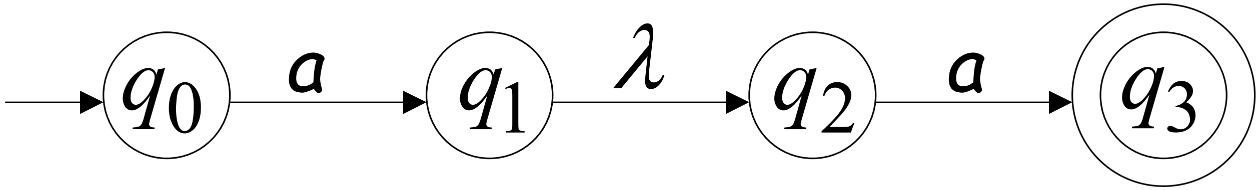


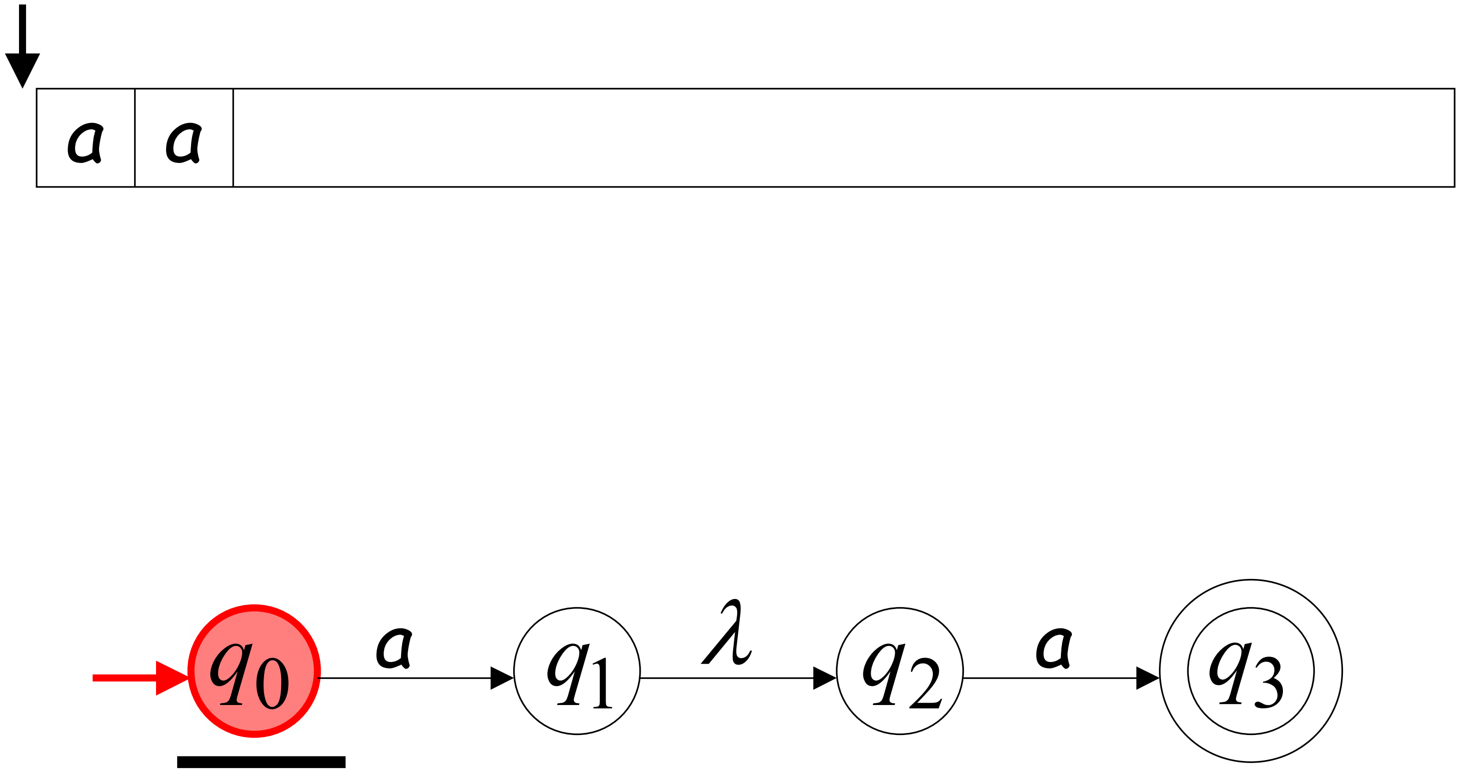
All possible computations lead to rejection

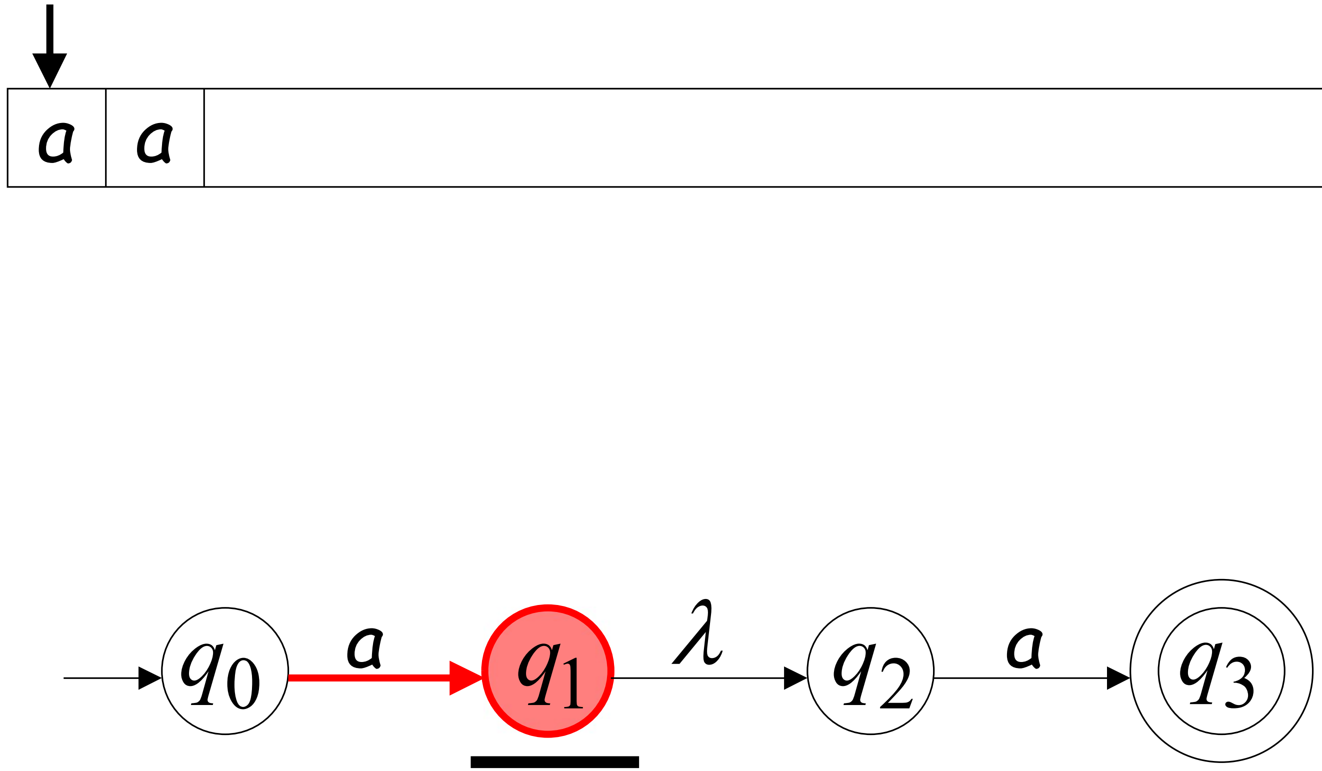
Language accepted: $L = \{aa\}$

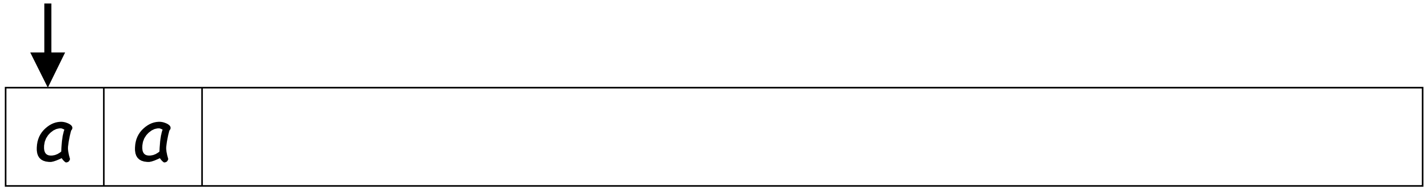


Lambda Transitions

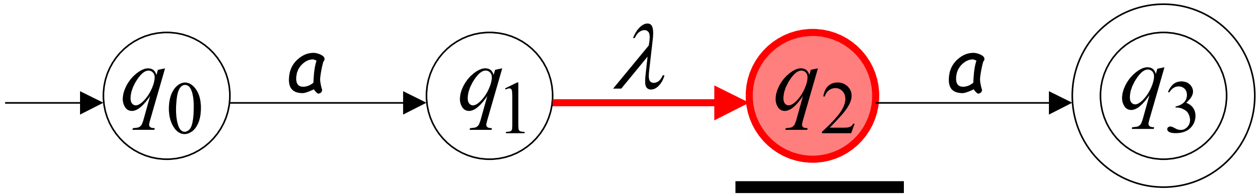


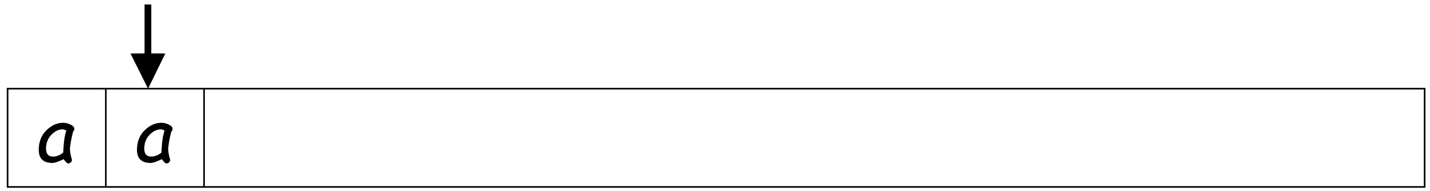

a
a
a
λ
a
q_0
q_1
q_2
q_3


a a
q_0 a q_1 λ q_2 a q_3

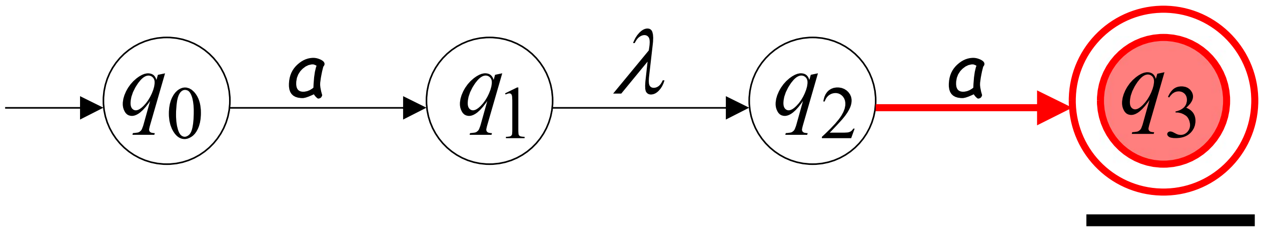


(read head does not move)



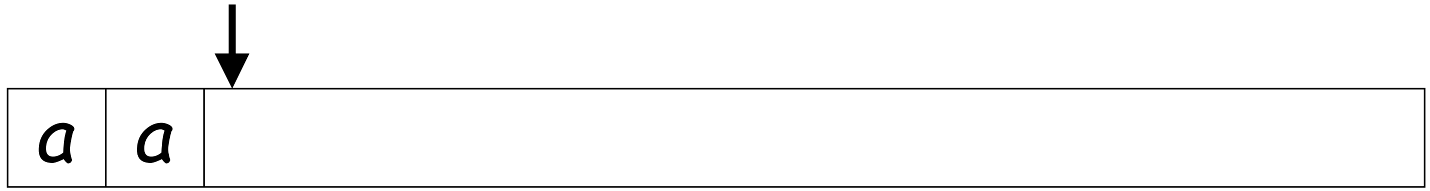


a
a

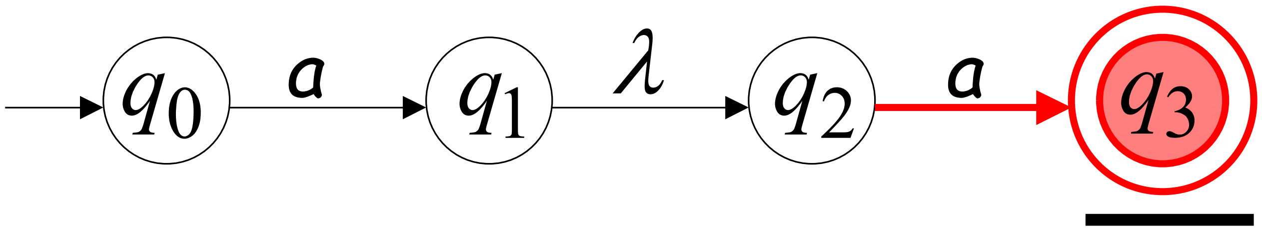


q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

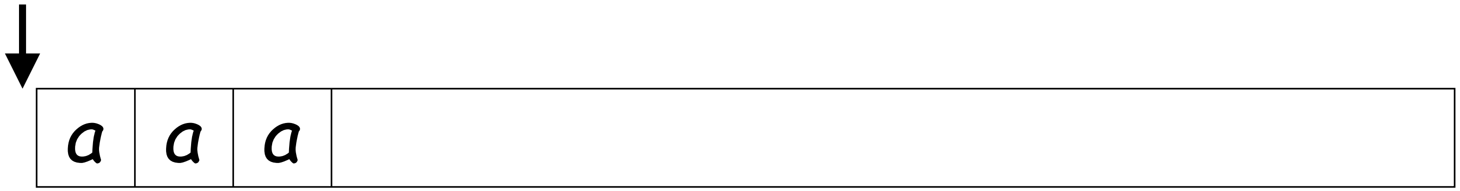


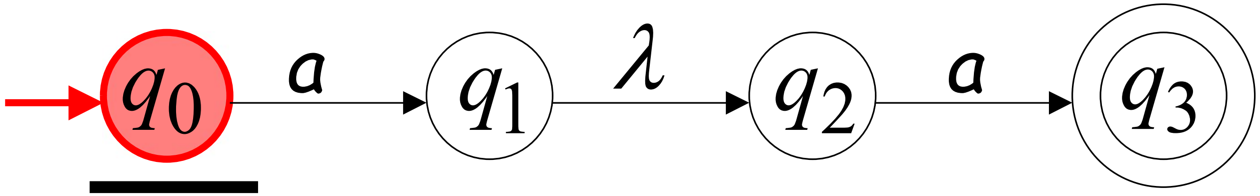
“accept”

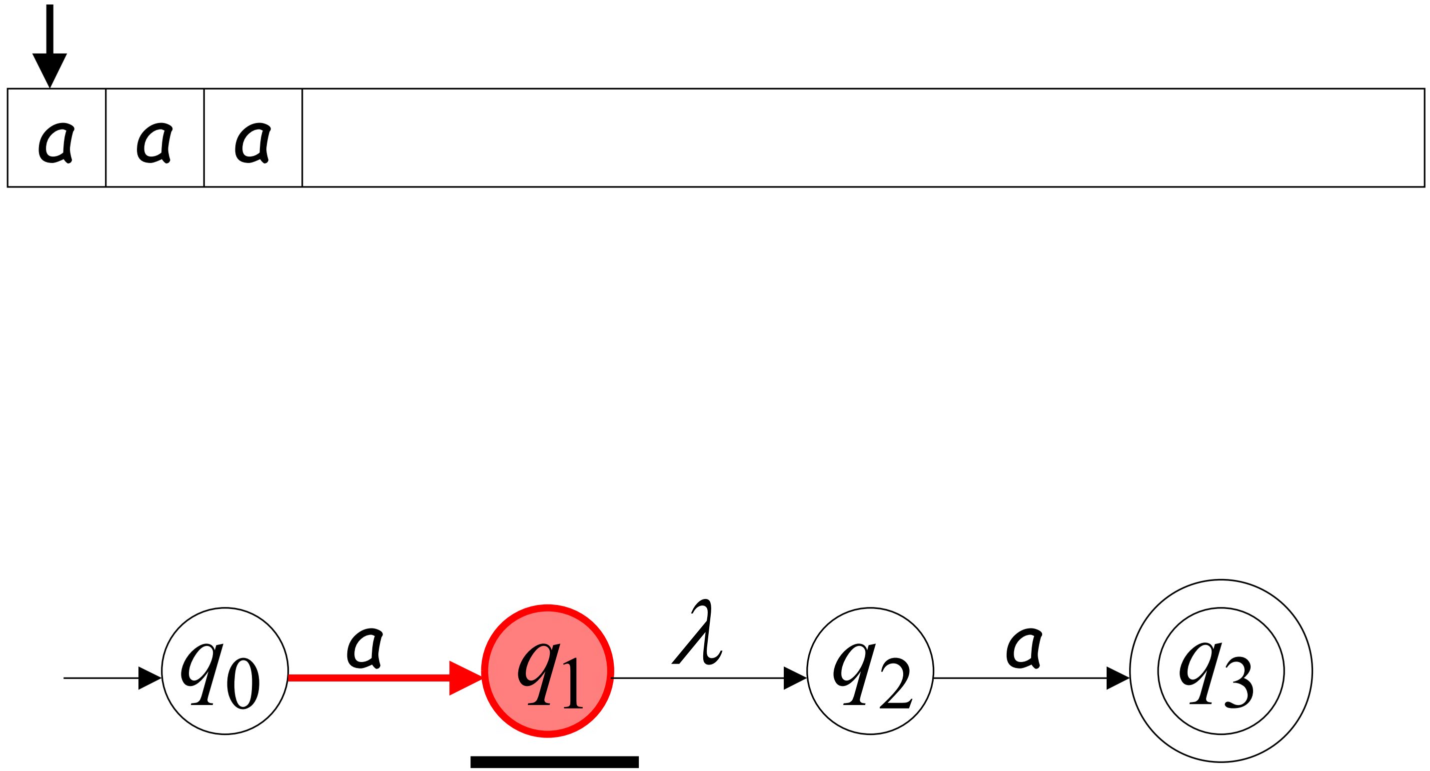


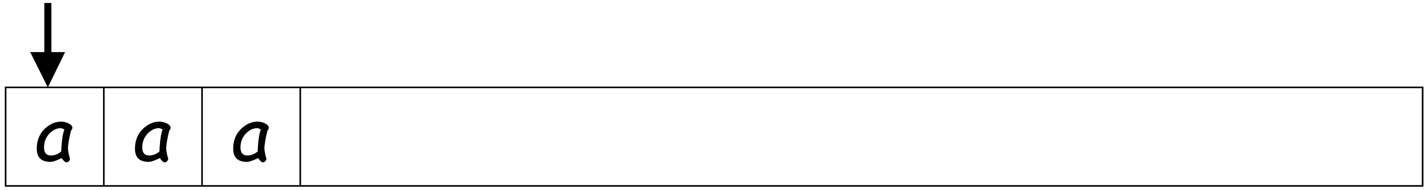
String aa is accepted

Rejection Example

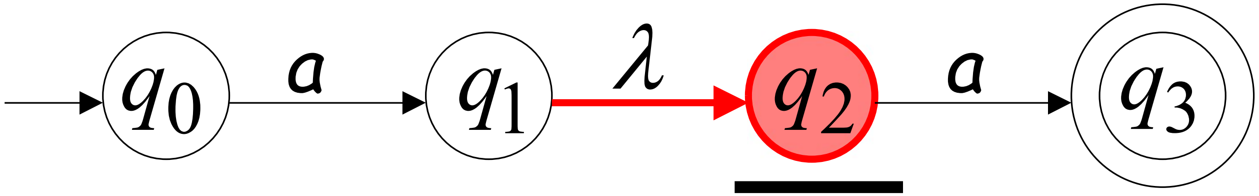


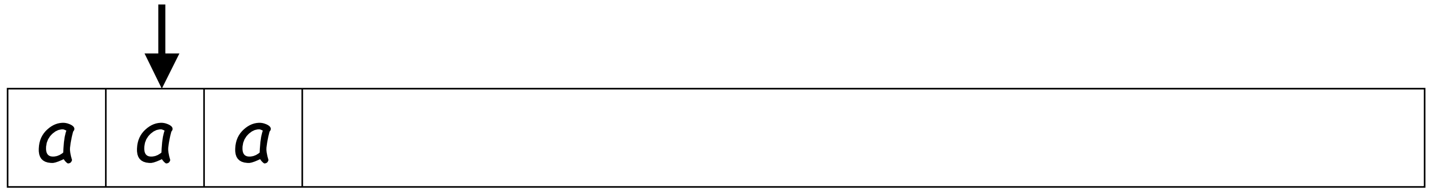


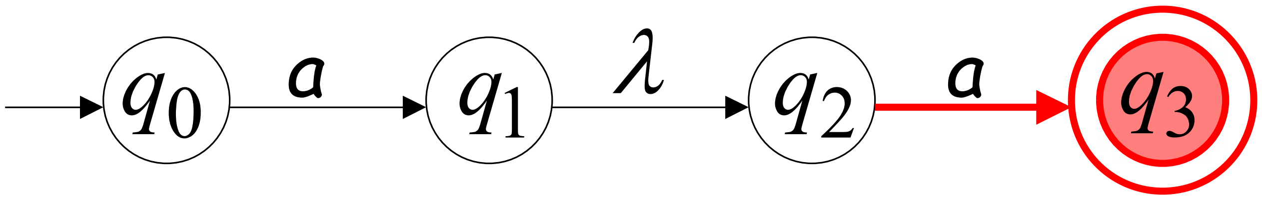

a
a
a
q_0
a
q_1
λ
q_2
a
q_3



(read head doesn't move)

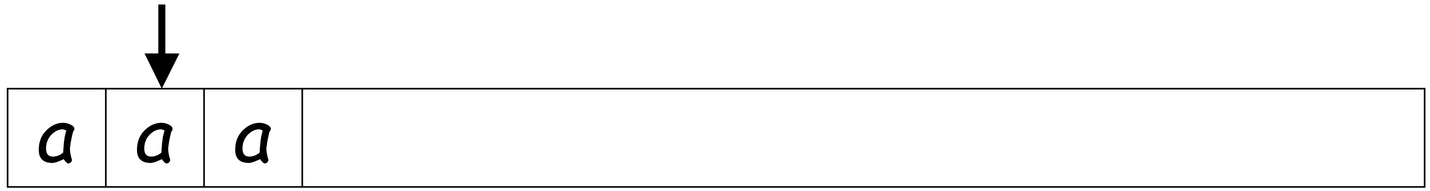




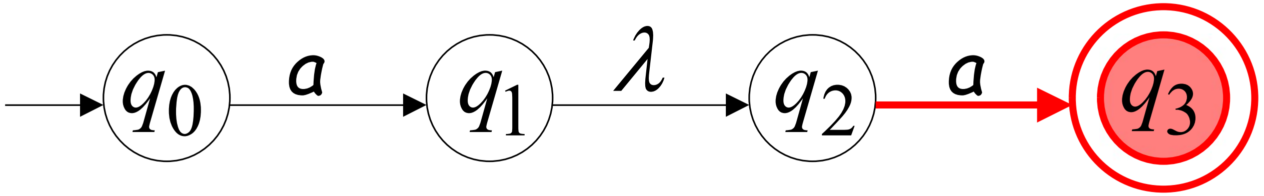


No transition:
the automaton hangs

Input cannot be consumed

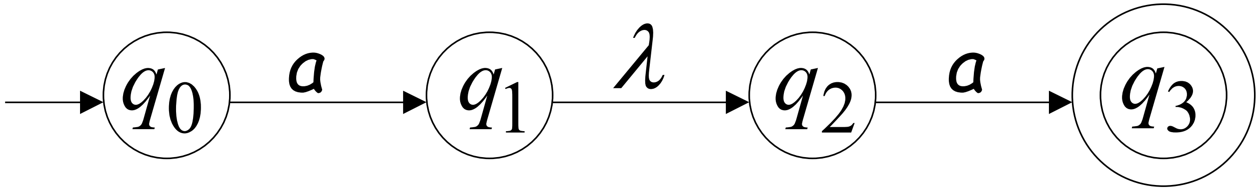


“reject”

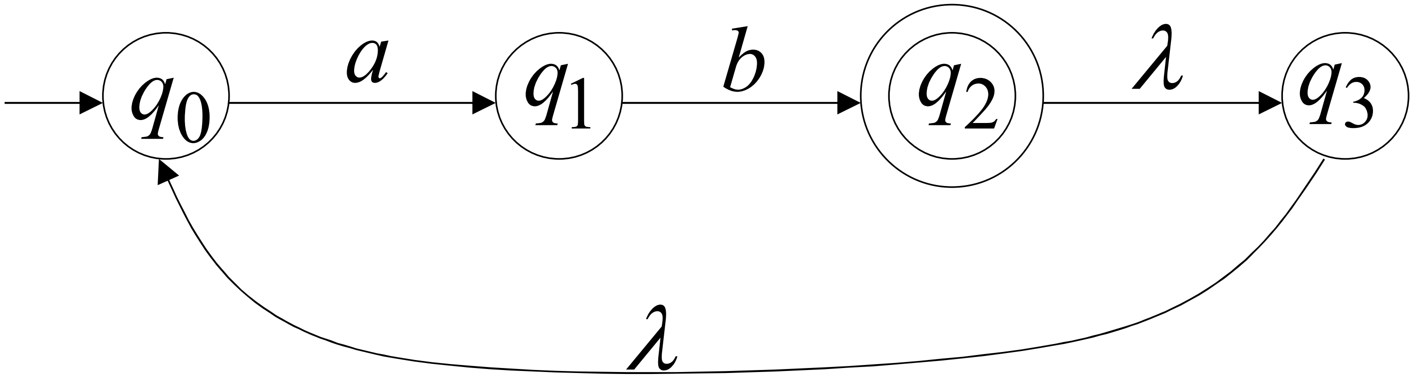


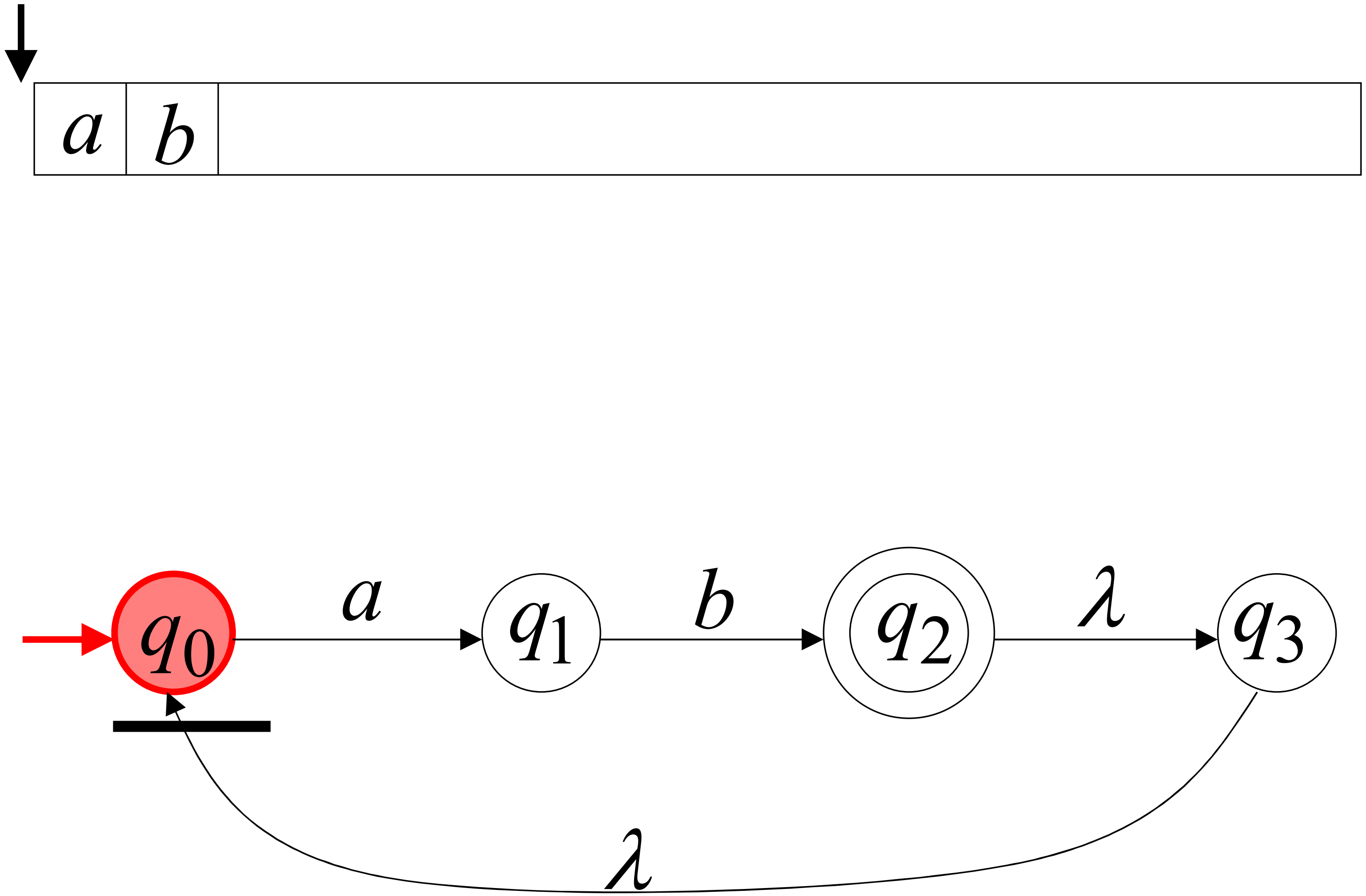
String aaa is rejected

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

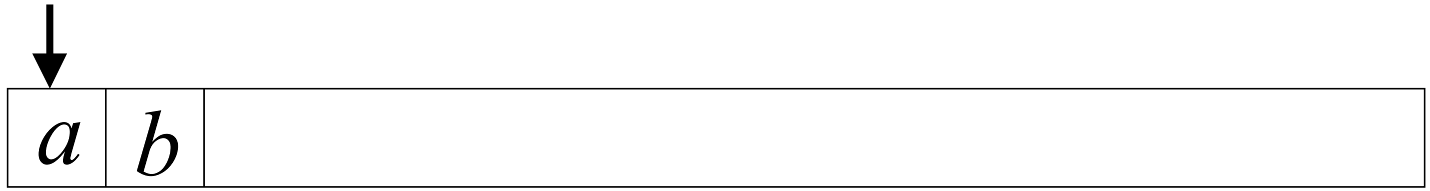


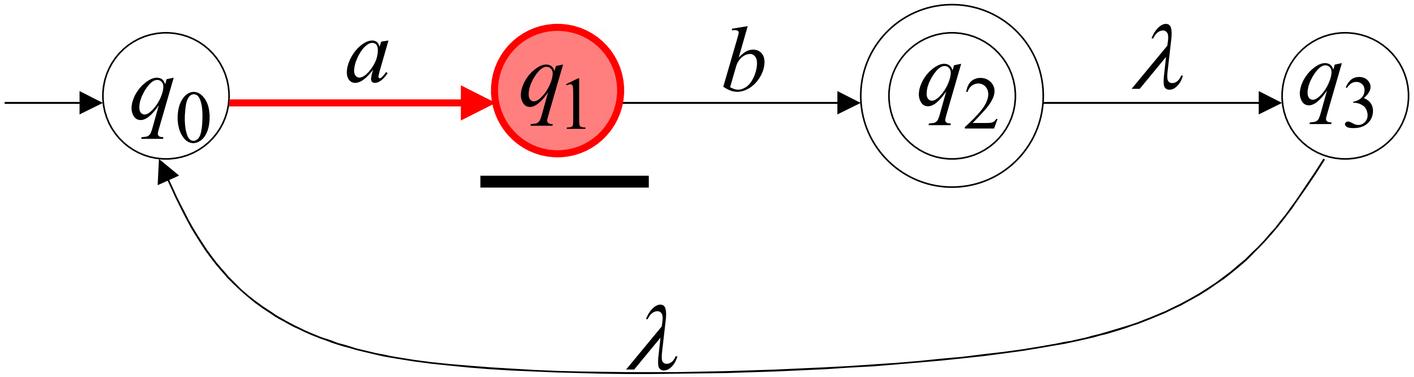
Another NFA Example

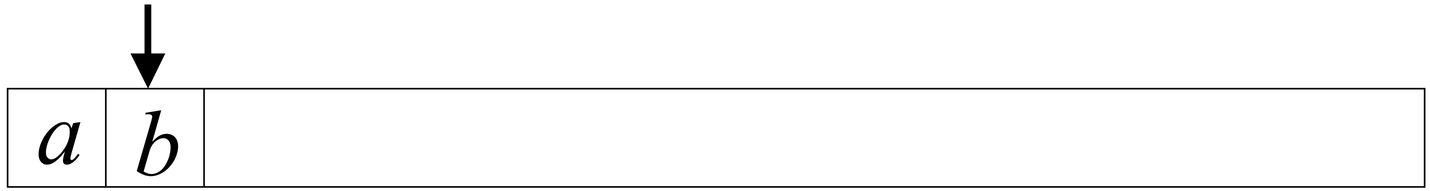


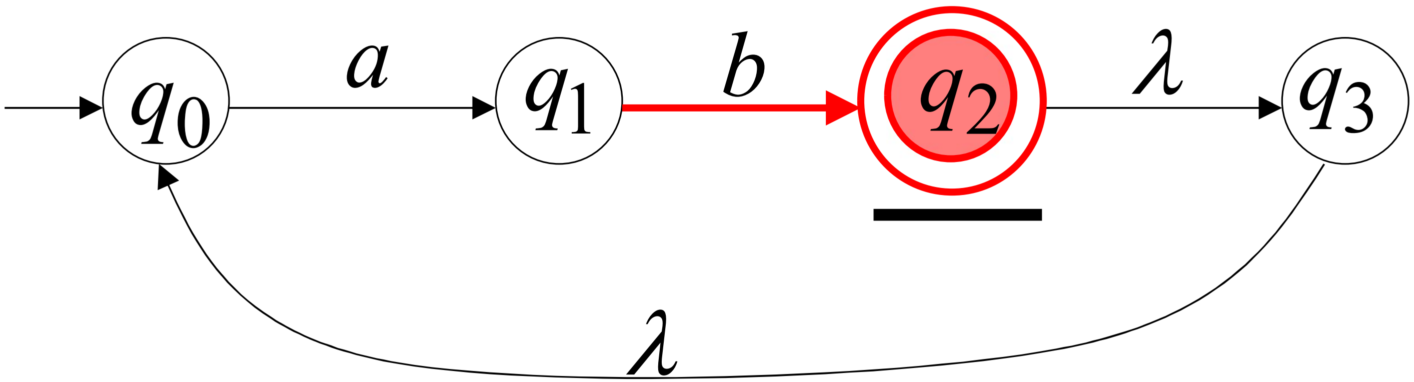


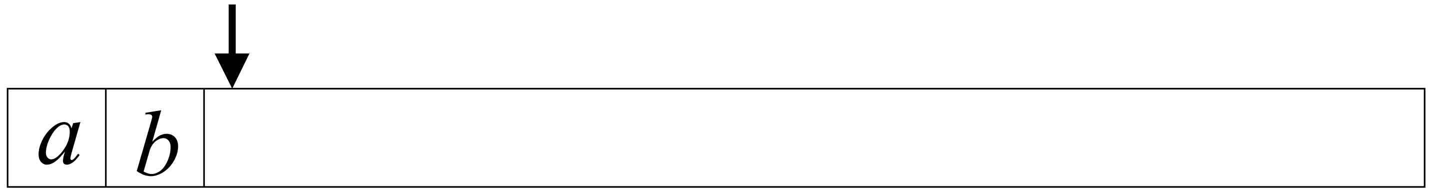
a
b
q0
a
q1
b
q2
λ
q3
λ

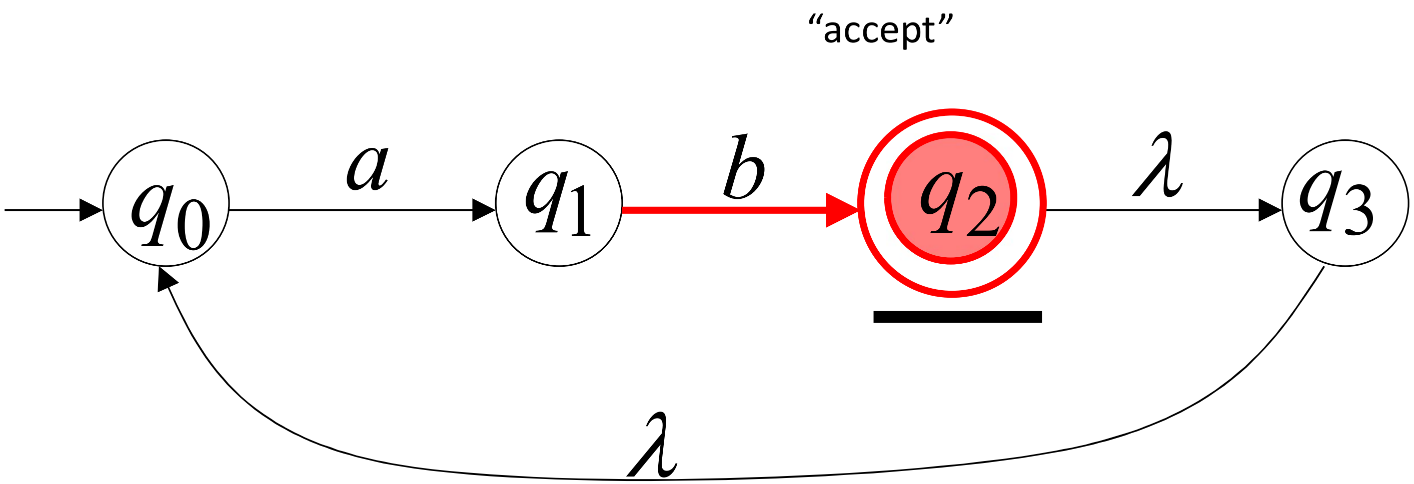

a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

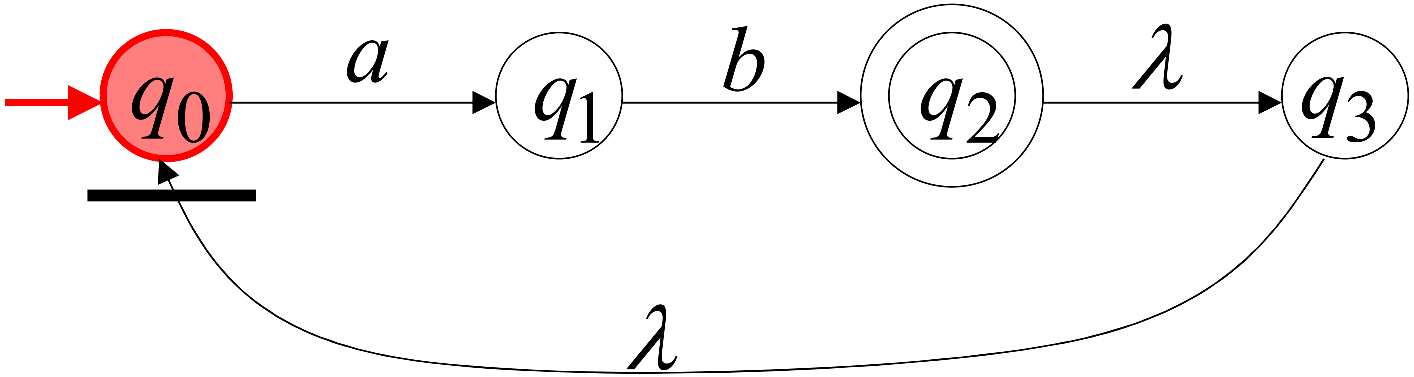

a
b

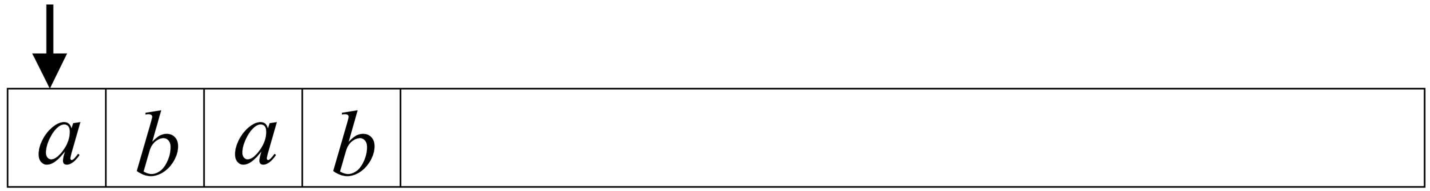

q0
a
q1
b
q2
λ
q3
λ

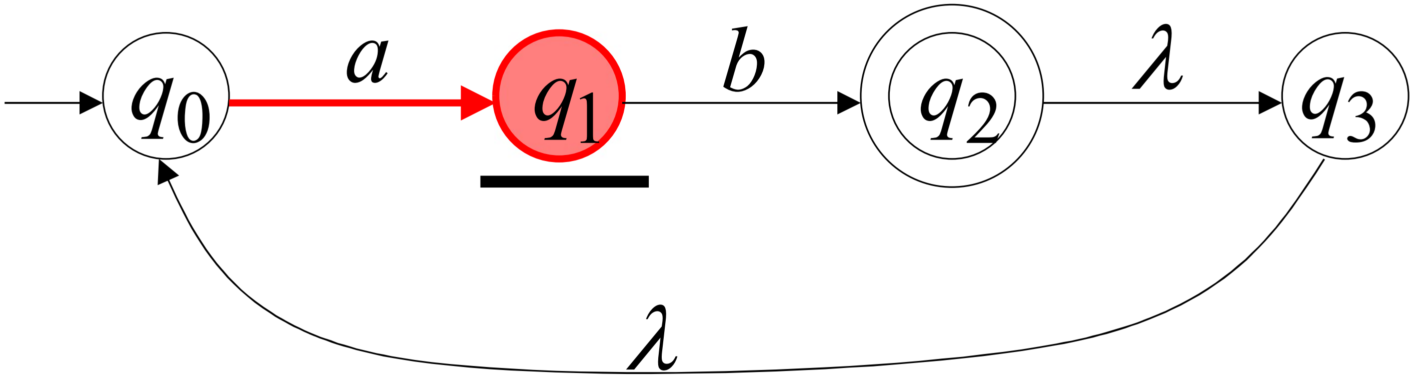

a
b

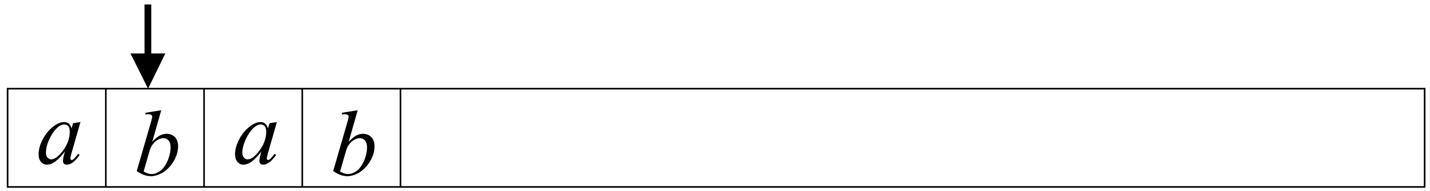

“accept”
q0
a
q1
b
q2
λ
q3
λ

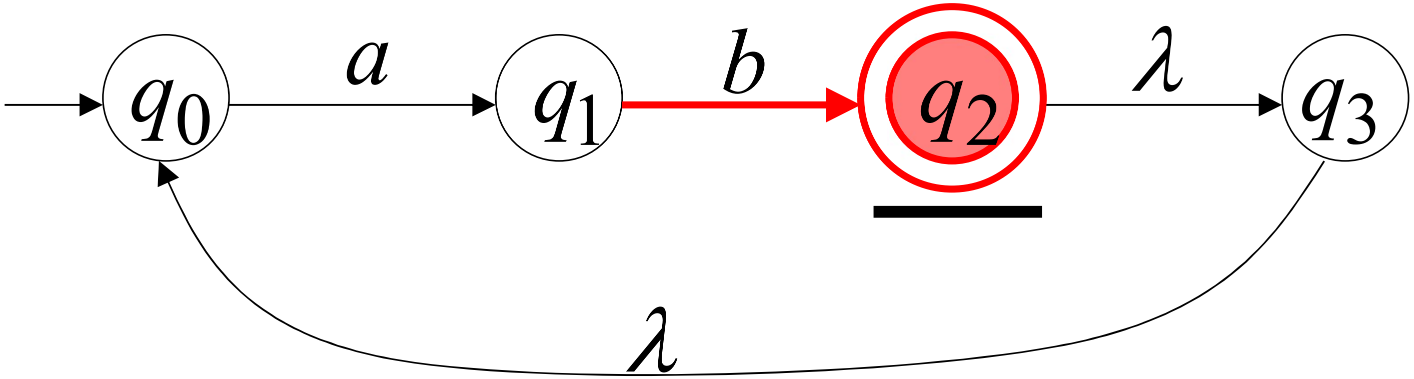
Another String

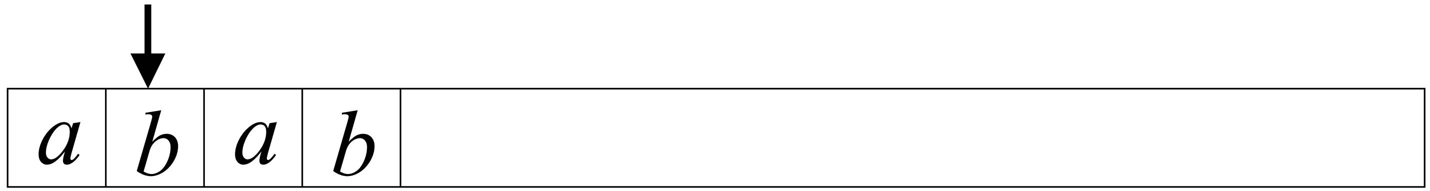


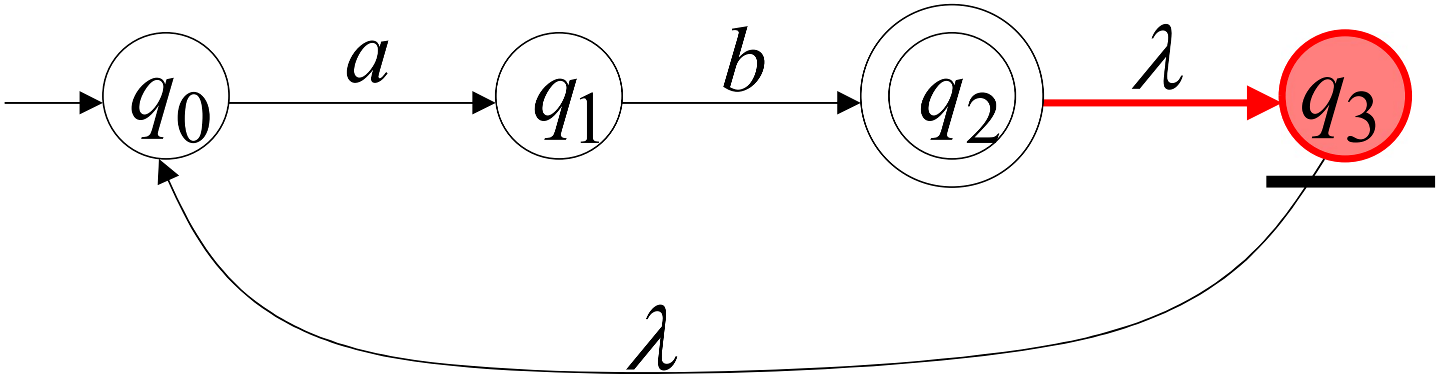

a
b
a
b

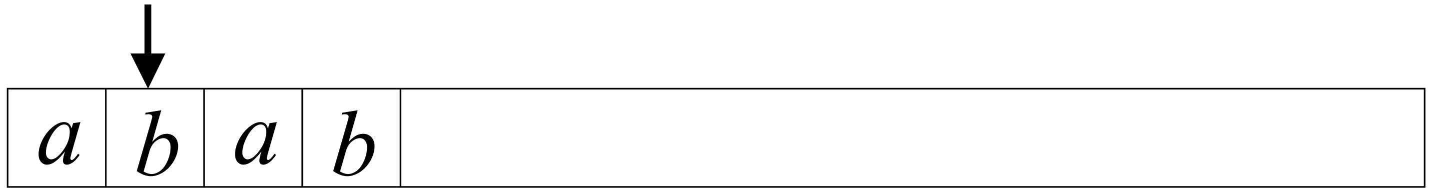

q_0
a
q_1
b
q_2
λ
q_3
λ

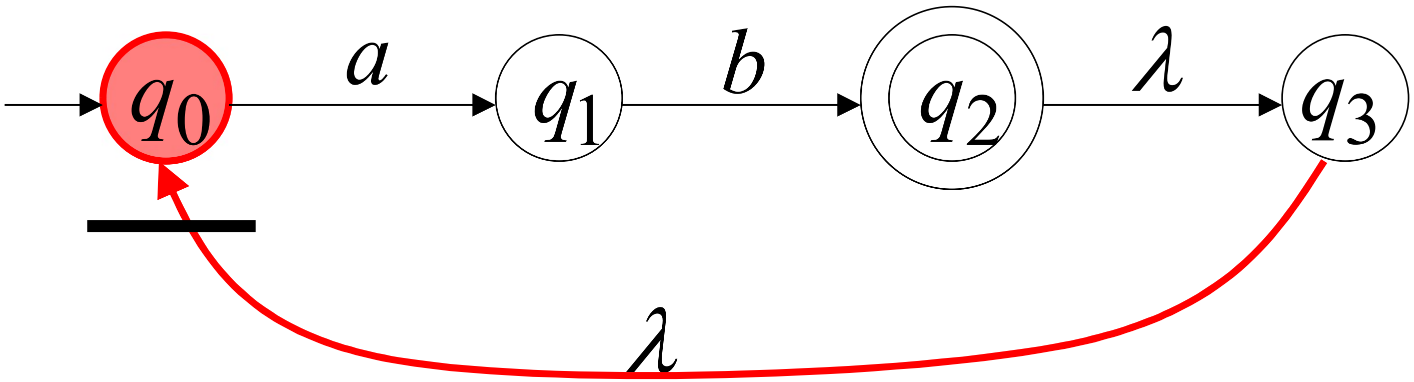

a
b
a
b

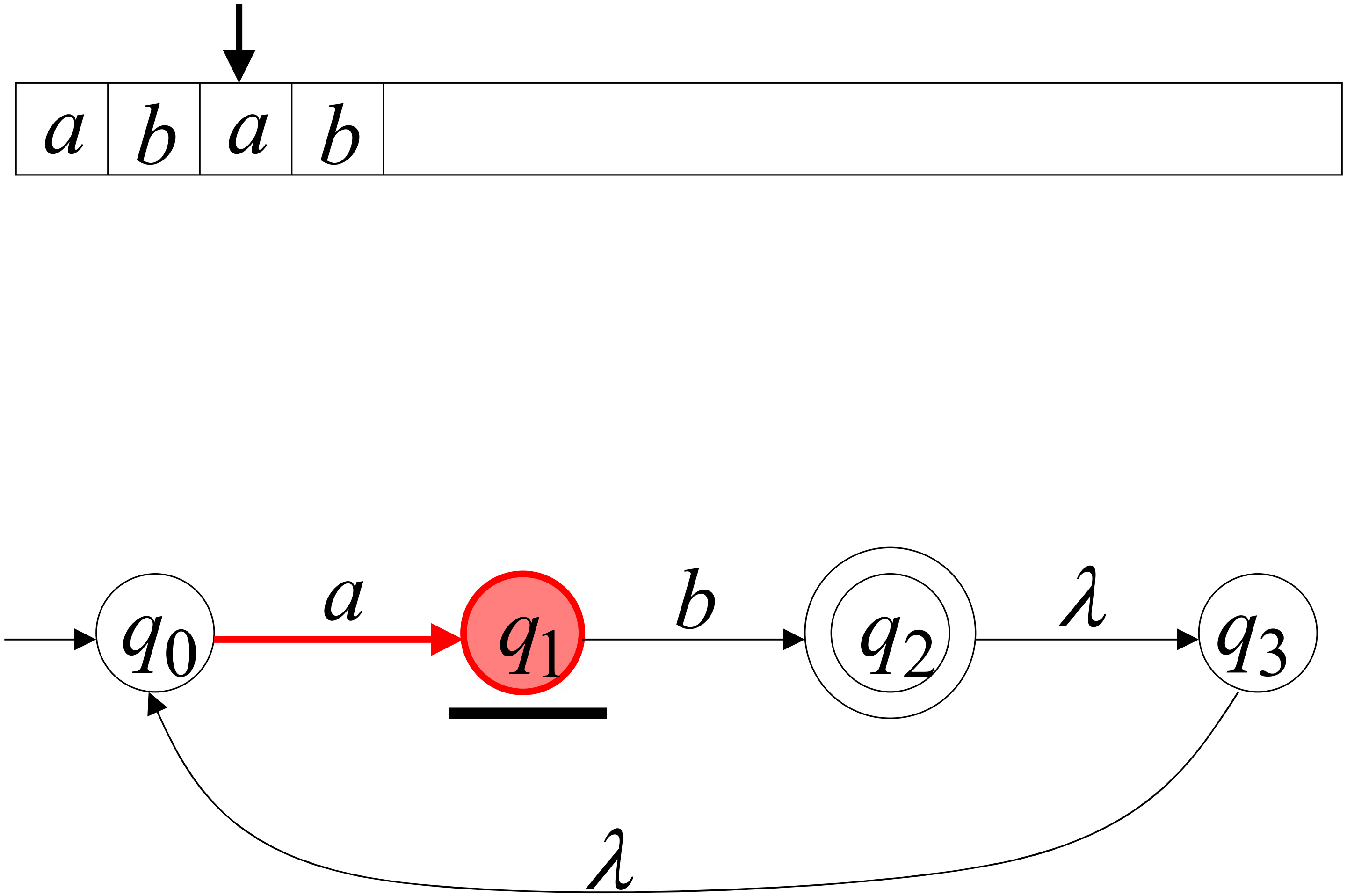

q0
a
q1
b
q2
λ
q3
λ

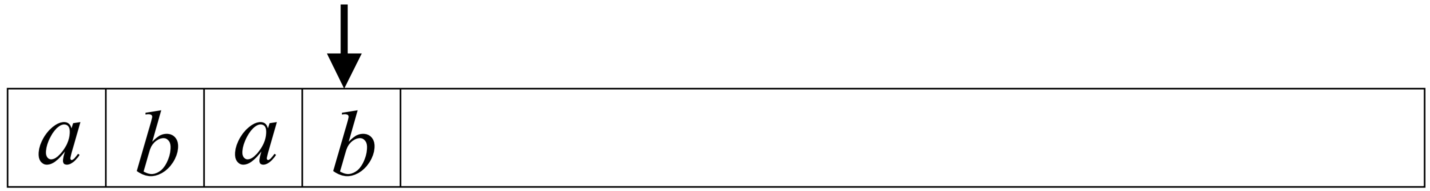

a
b
a
b

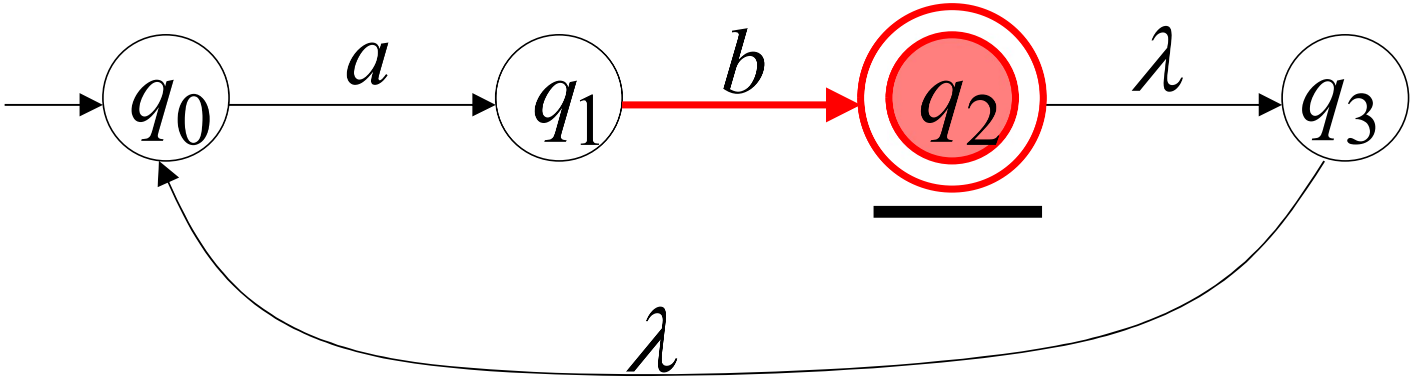

q_0
a
q_1
b
q_2
λ
q_3
λ

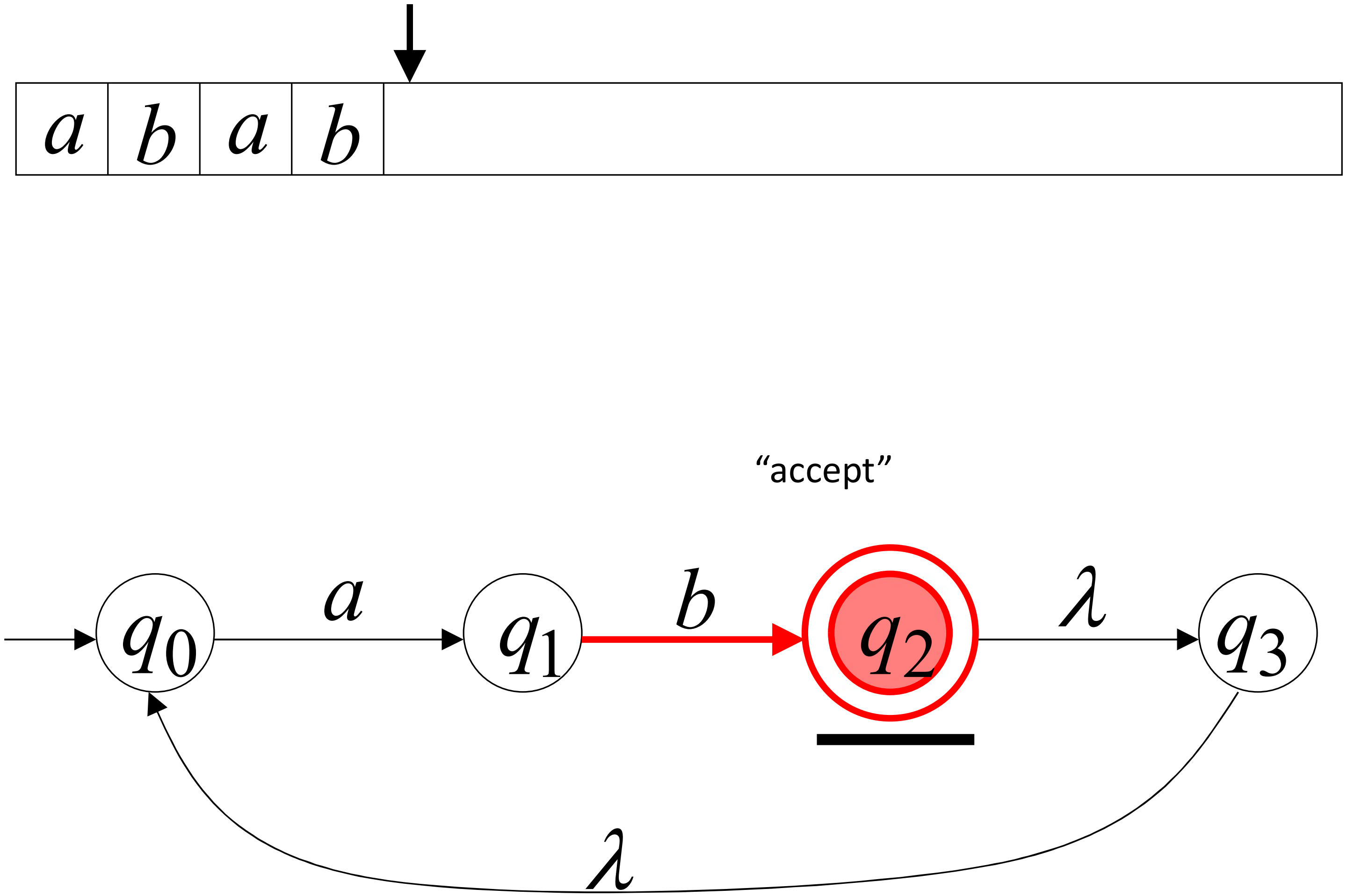

a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b
a
b
λ
λ

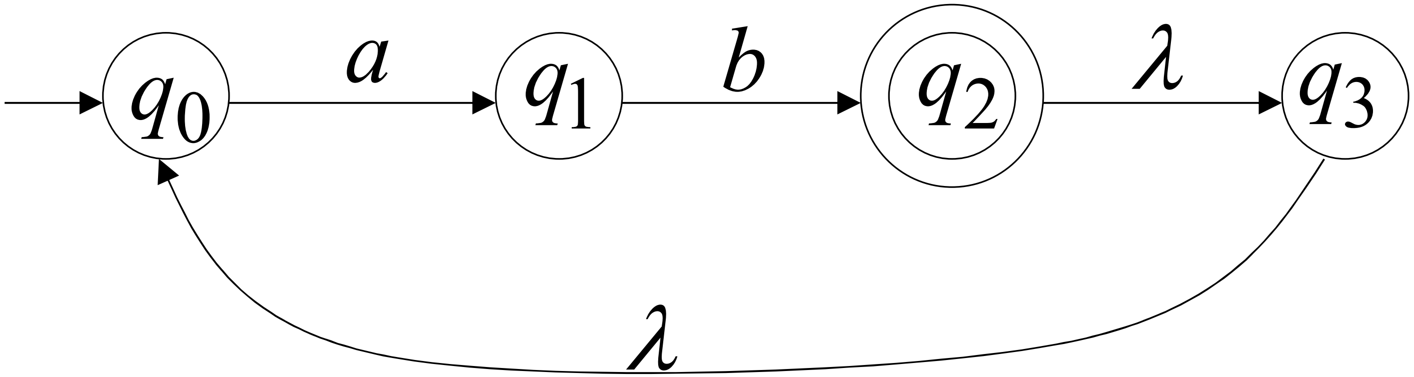

a
b
a
b


q0
a
q1
b
q2
λ
q3
λ

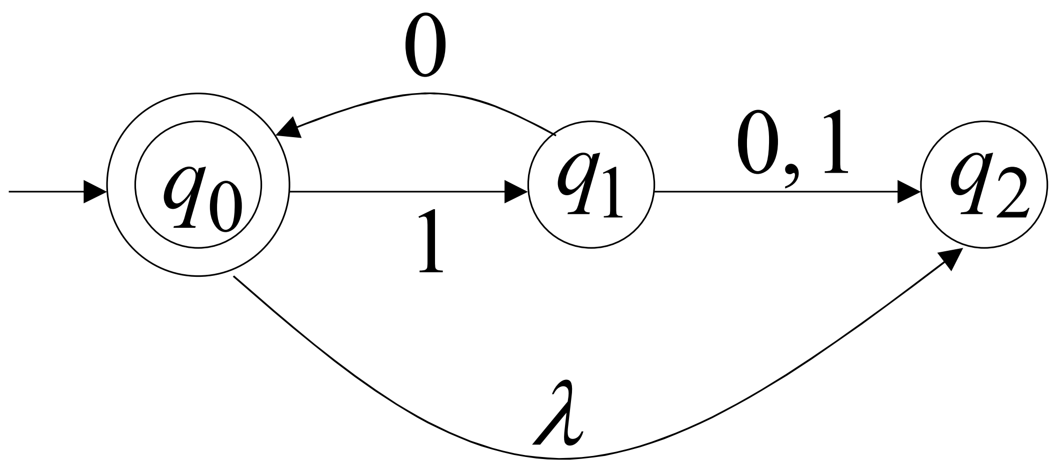

a b a b
"accept"
q0
a
q1
b
q2
λ
q3
λ

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

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

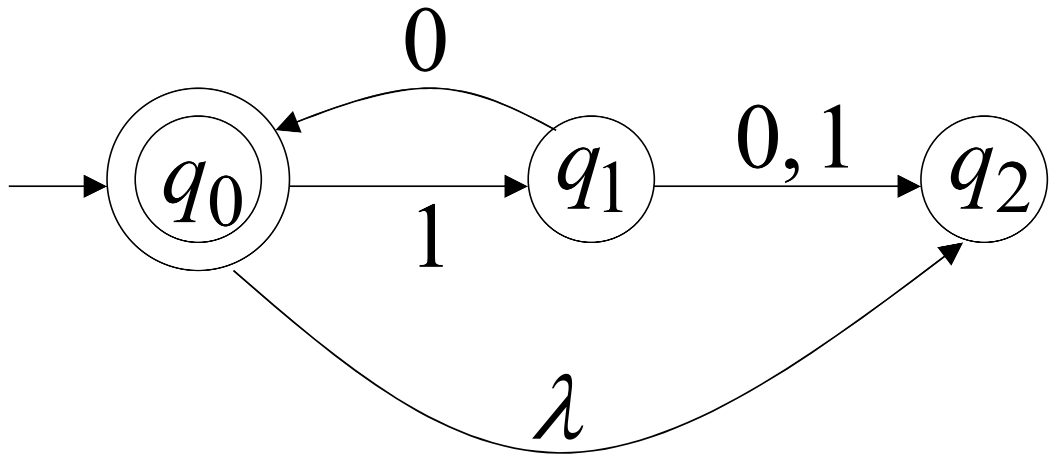


Another NFA Example



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

$= \{10\}^*$



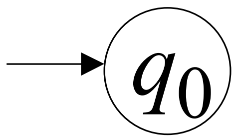
(redundant state)

Remarks:

- The λ symbol never appears on the input tape

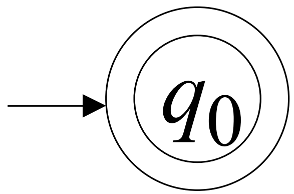
- Simple automata:

M_1



$L(M_1) = \{\}$

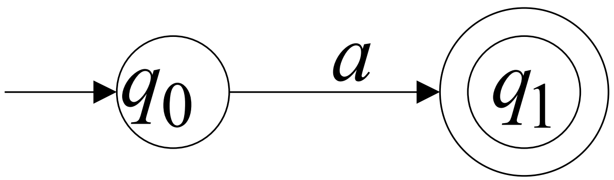
M_2



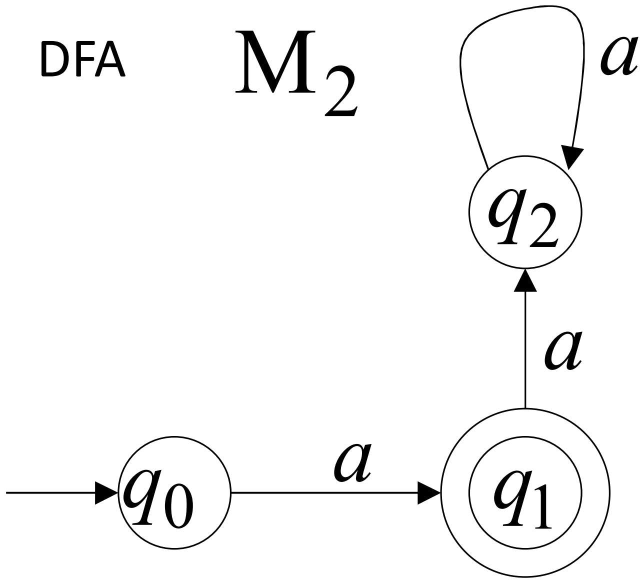
$L(M_2) = \{\lambda\}$

•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

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

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

  $\Sigma:$ Input alphabet, i.e. $\{a, b\}$

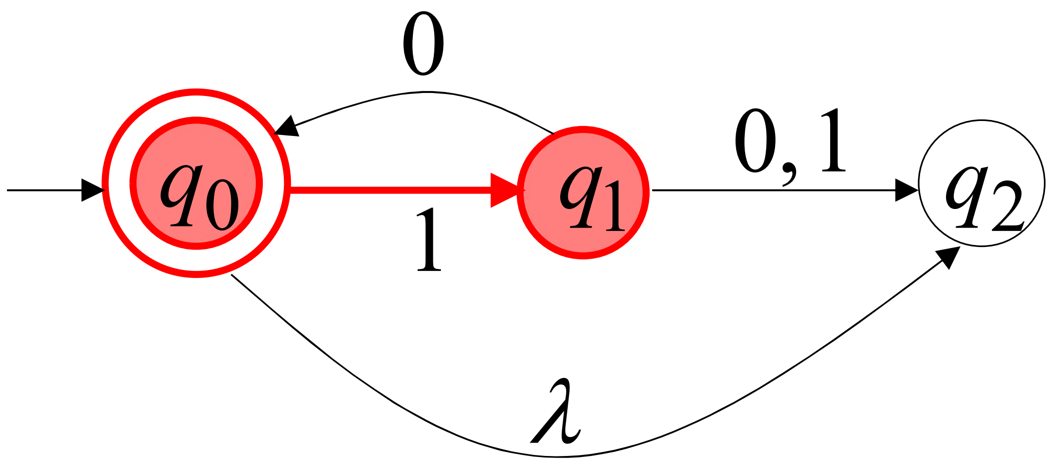
  $\delta:$ Transition function $\delta : Q \times \boxed{(\Sigma \cup \{\lambda\}}) \to 2^Q.$

  $q_0:$ Initial state

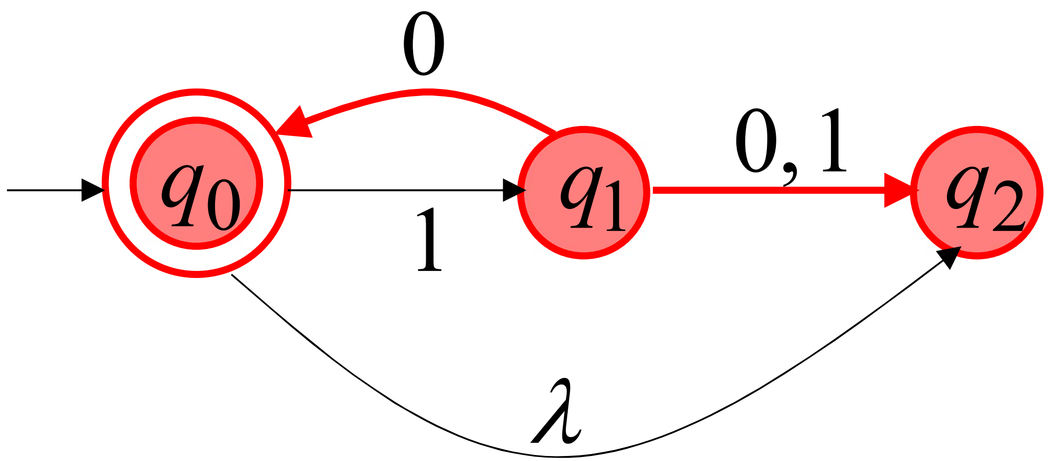
  $F:$ Final states

Transition Function δ

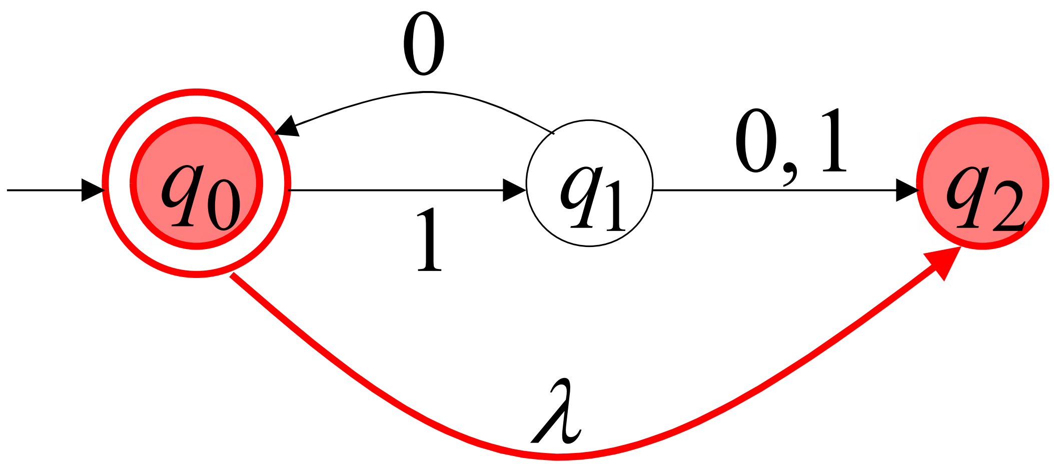
$$\delta(q_0, 1) = \{q_1\}$$



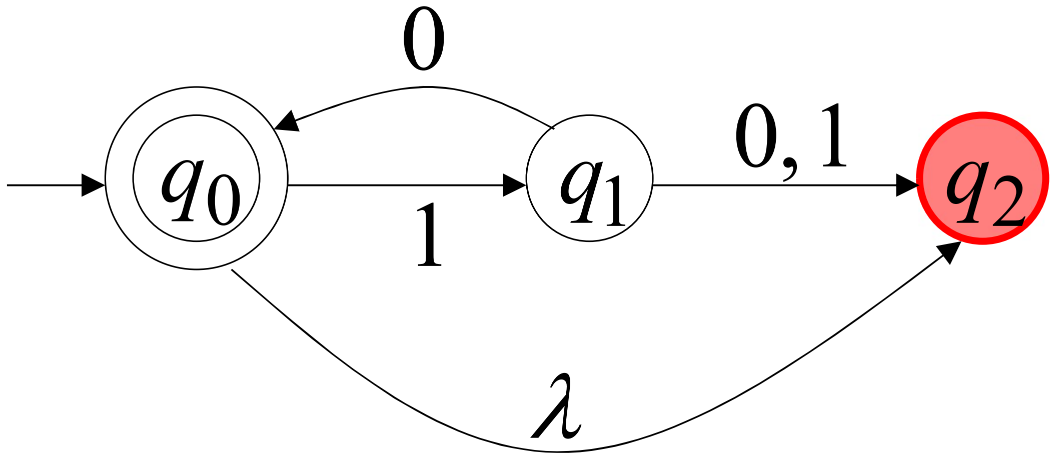
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

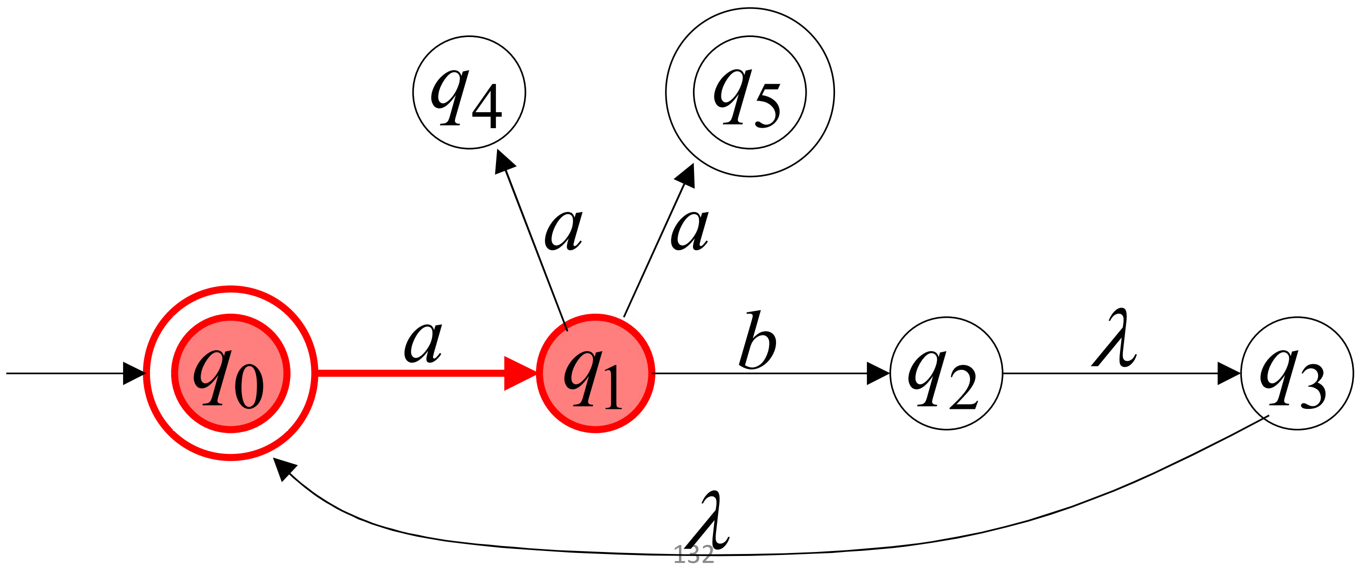


$$\delta(q_2, 1) = \varnothing$$

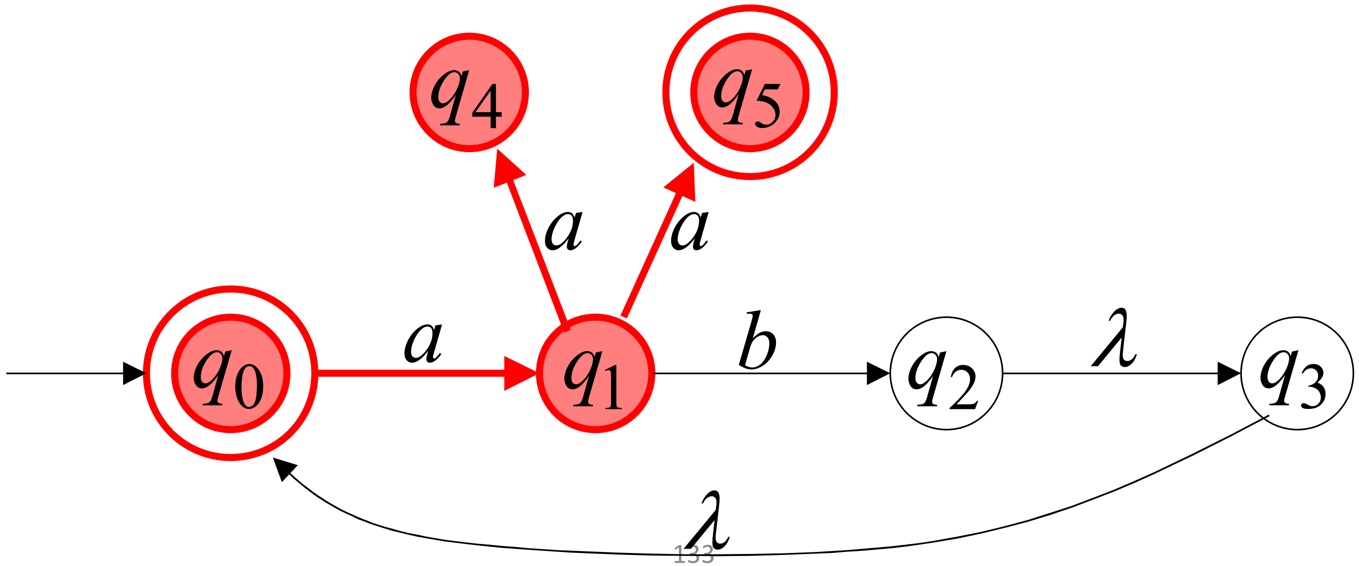


Extended Transition Function

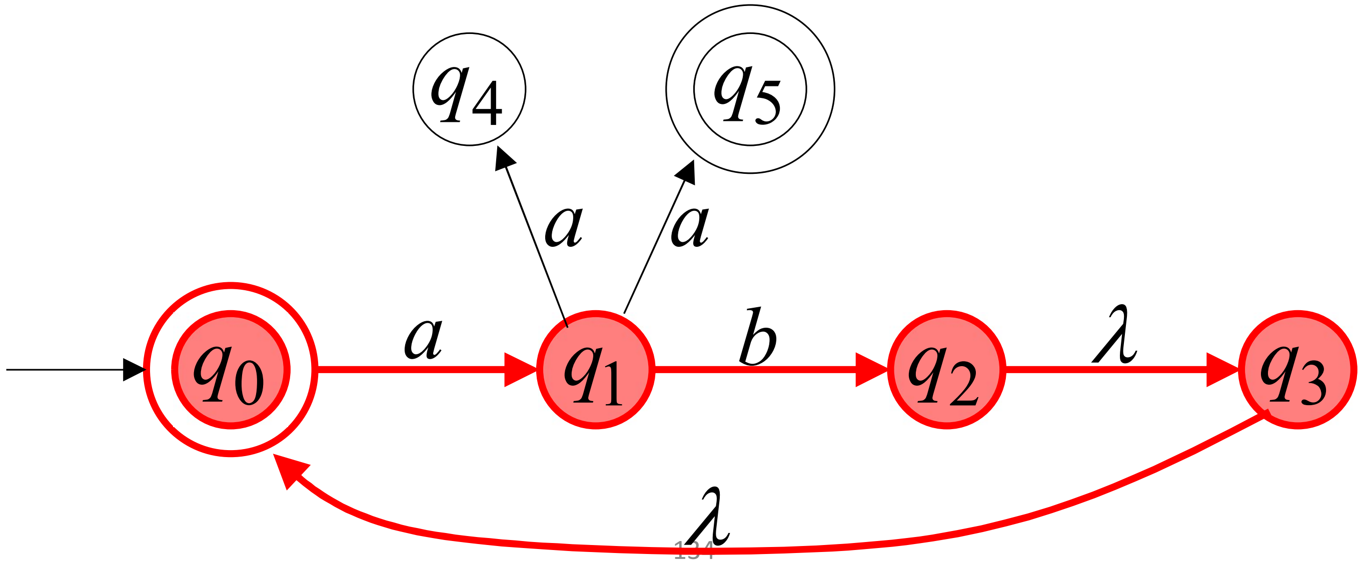
- $$\delta^*(q_0, a) = \{q_1\}$$



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

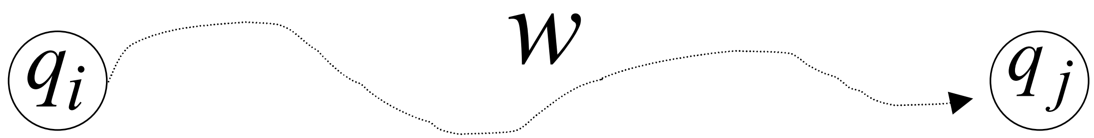


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

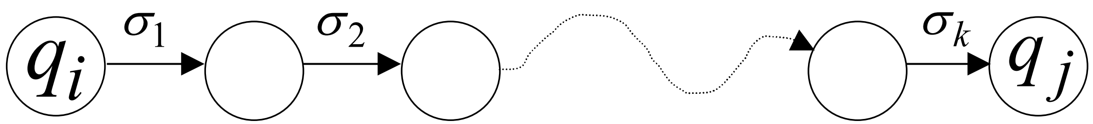


Formally

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

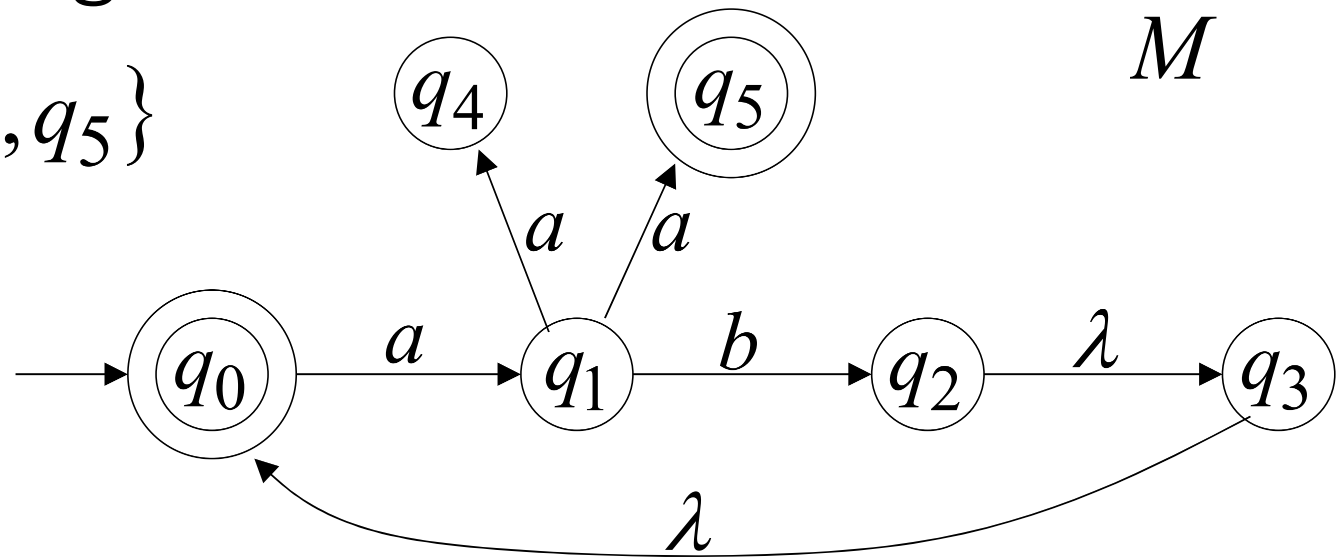


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



The Language of an NFA

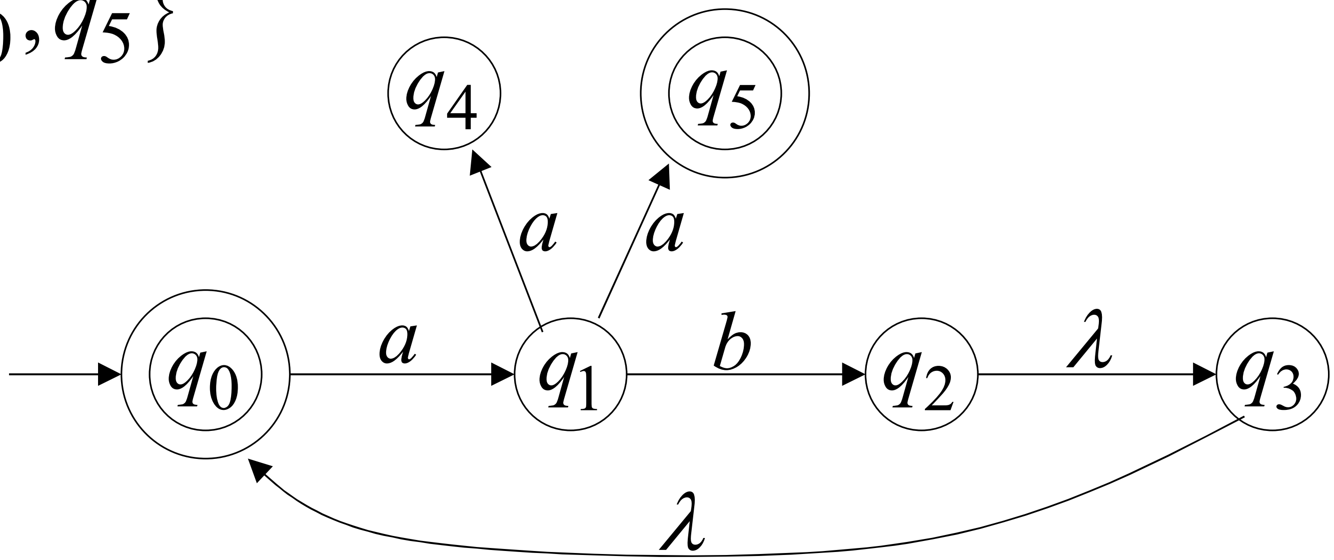
- $F = \{q_0, q_5\}$



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

$$\in F$$

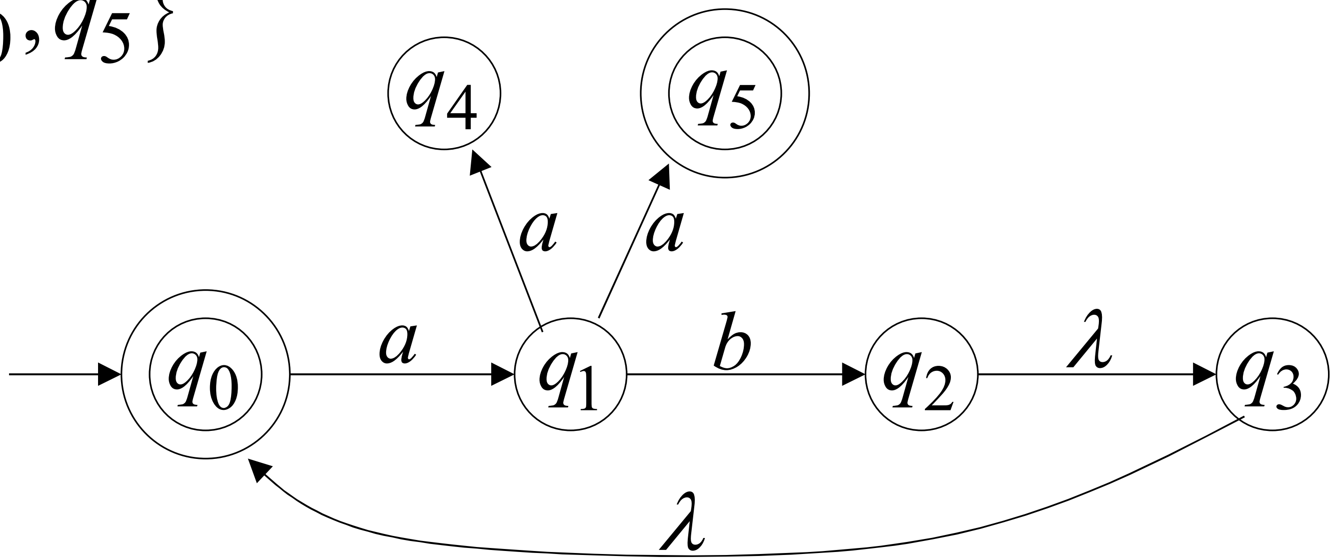
$$F = \{q_0, q_5\}$$



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

$$\in F$$

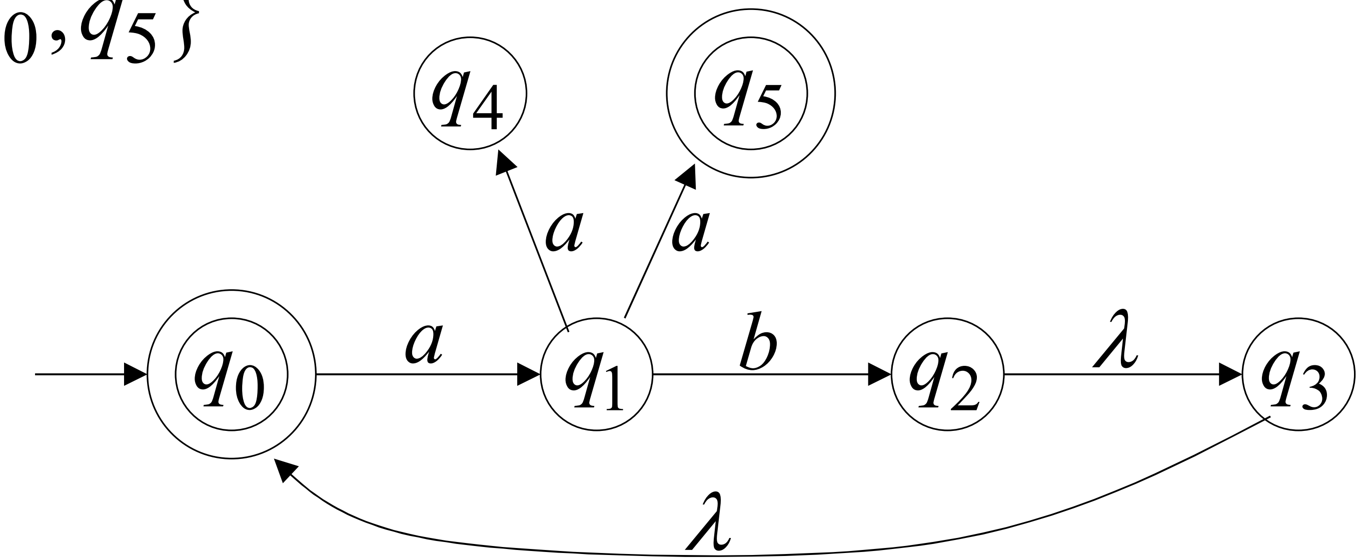
$$F = \{q_0, q_5\}$$

- 

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

$$\in F$$

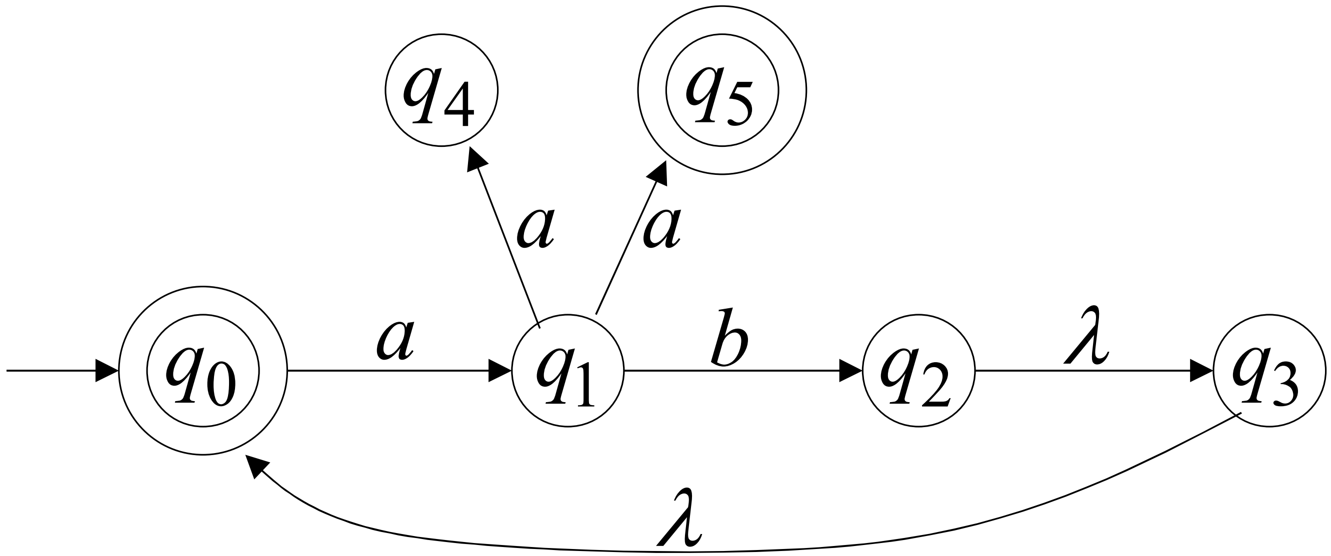
$$F = \{q_0, q_5\}$$



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

$$\notin F$$

•



$$L(M) = \{(ab)^*(aa)^n : n = \{0,1\}\}$$

Formally

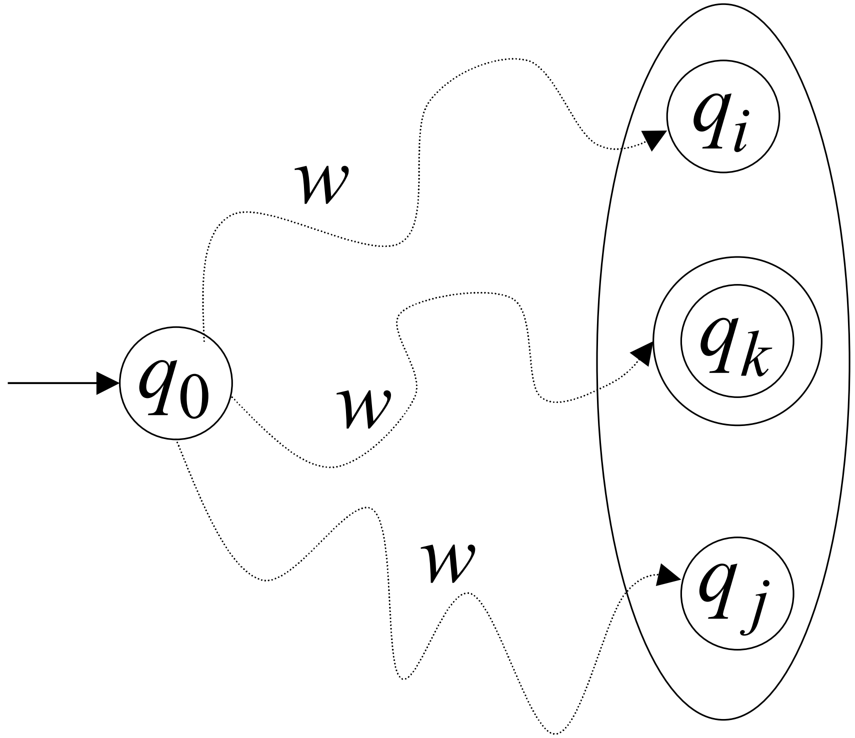
- The language accepted by NFA M is: $L(M) = \{w_1, w_2, w_3, \ldots\}$

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

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

$$w \in L(M)$$

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

- 

$$q_k \in F$$

NFA vs. DFA

- Transition functions range is Q vs. 2^Q (powersets of Q)
- λ can be an argument of transition function; transition without consuming a symbol
- $\delta(q_k, a)$ can be empty (not a total function)

δ	a	b
q_0	q_1	
q_1		q_2

Equivalence of Machines

- NFAs accept the Regular Languages

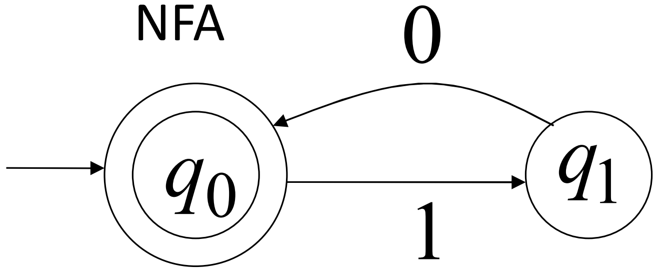
- Machine M_1 is equivalent to machine M_2

- 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