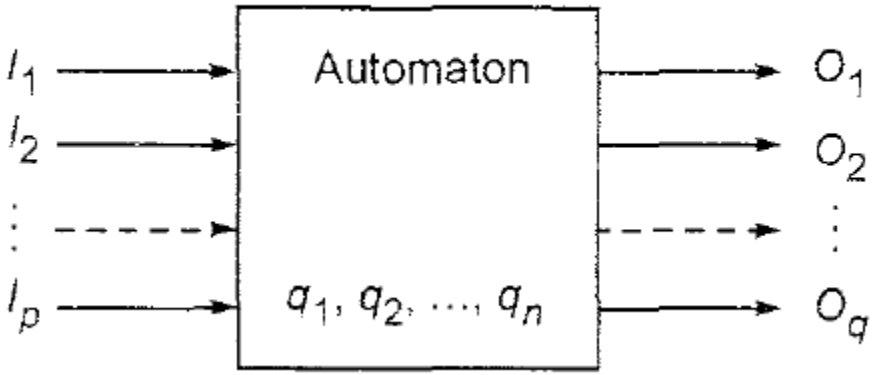
Let's Start: Finite Automata

Definition

An automaton is defined as a system where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man.



Formal Definition

- Finite Automaton (FA)

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

Q : set of states

Σ : input alphabet

δ : transition function

q_0 : initial state

F : set of accepting states

DESCRIPTION OF A FINITE AUTOMATON

Definition 3.1 Analytically, a finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

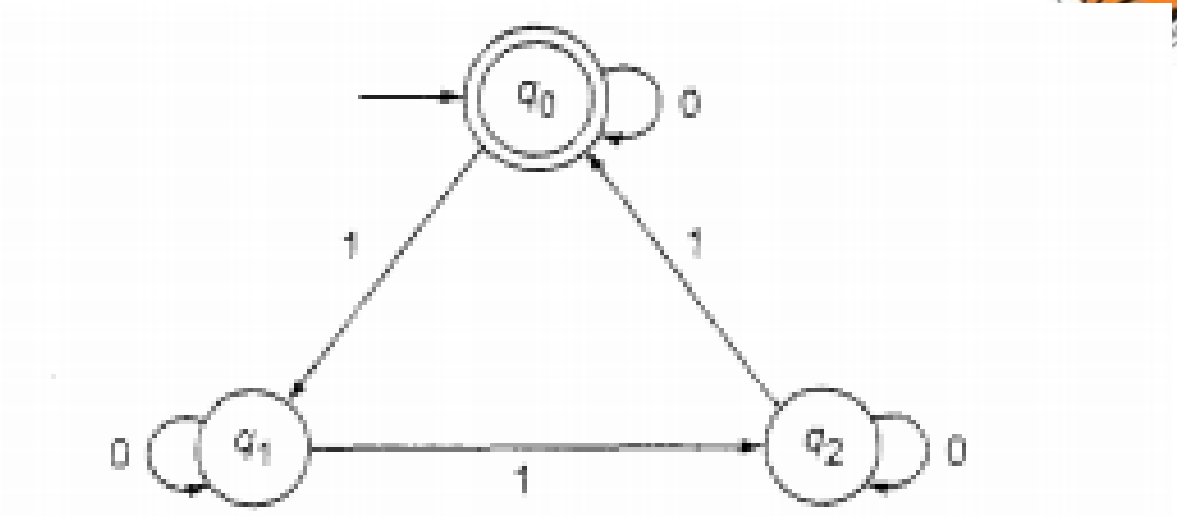
(i) Q is a finite nonempty set of states.

(ii) Σ is a finite nonempty set of inputs called the *input alphabet*.

(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called the *direct transition function*. This is the function which describes the change of states during the transition. This mapping is usually represented by a transition table or a transition diagram.

(iv) $q_0 \in Q$ is the initial state.

(v) $F \subseteq Q$ is the set of final states. It is assumed here that there may be more than one final state.


q_0
0
1
1
0
q_1
1
q_2
0

- The transition function which maps $Q \times \Sigma^*$ into Q (i.e. maps a state and a string of input symbols including the empty stling into a state) is called the *indirect transition function.*

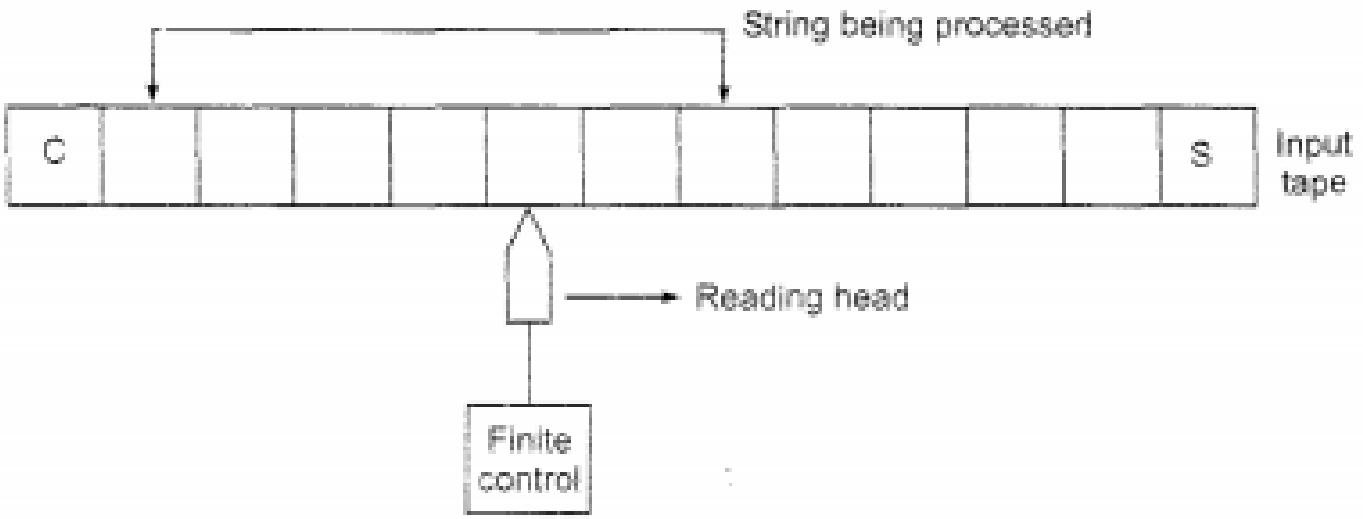


Fig. 3.4 Block diagram of a finite automaton.

(i) *Input tape.* The input tape is divided into squares, each square containing a single symbol from the input alphabet Σ. The end squares of the tape contain the endmarker ¢ at the left end and the endmarker $ at the right end. The absence of endmarkers indicates that the tape is of infinite length. The left-to-right sequence of symbols between the two endmarkers is the input string to be processed.

(ii) *Reading head.* The head examines only one square at a time and can move one square either to the left or to the right. For further analysis, we restrict the movement of the R-head only to the right side.

(iii) *Finite control.* The input to the finite control will usually be the symbol under the R-head, say a, and the present state of the machine, say q, to give the following outputs: (a) A motion of R-head along the tape to the next square (in some a null move, i.e. the R-head remaining to the same square is permitted); (b) the next state of the finite state machine given by $\delta(q, a)$.

Definition 3.2 A transition system is a 5-tuple $(Q, \Sigma, \delta, Q_0, F)$, where

(i) Q, Σ and F are the finite nonempty set of states, the input alphabet, and the set of final states, respectively, as in the case of finite automata;

(ii) $Q_0 \subseteq Q$, and Q_0 is nonempty; and

(iii) δ is a finite subset of $Q \times \Sigma^* \times Q$.

Definition 3.3 A transition system accepts a string w in Σ^* if

(i) there exists a path which originates from some initial state, goes along the arrows, and terminates at some final state; and

(ii) the path value obtained by concatenation of all edge-labels of the path is equal to w.

Consider the transition system given in Fig. 3.6.

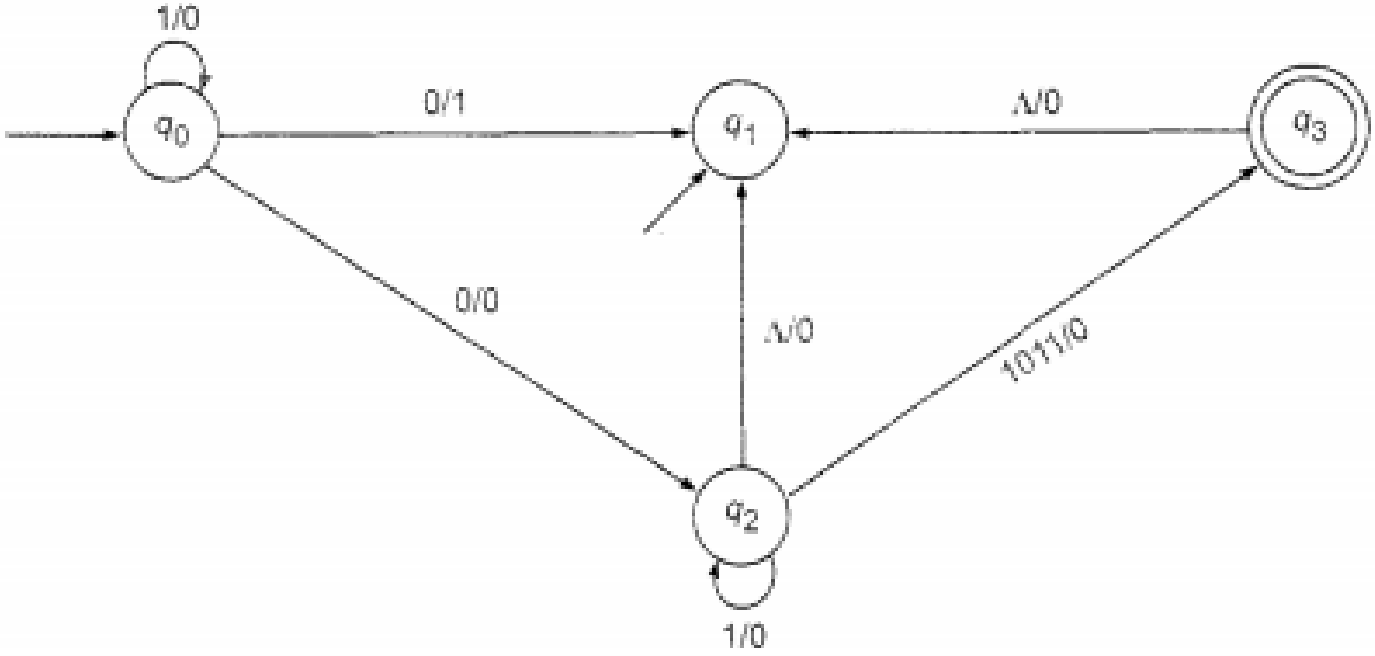


Fig. 3.6 Transition system for Example 3.2.

Determine the initial states. the final states. and the acceptability of 101011, 111010.

Consider the finite state machine whose transition function δ is given by Table 3.1 in the form of a transition table. Here, $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$. Give the entire sequence of states for the input string 110001.

TABLE 3.1 Transition Function Table for Example 3.5

State	Input	
	0	1
$\rightarrow$ (q_0)	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Solution

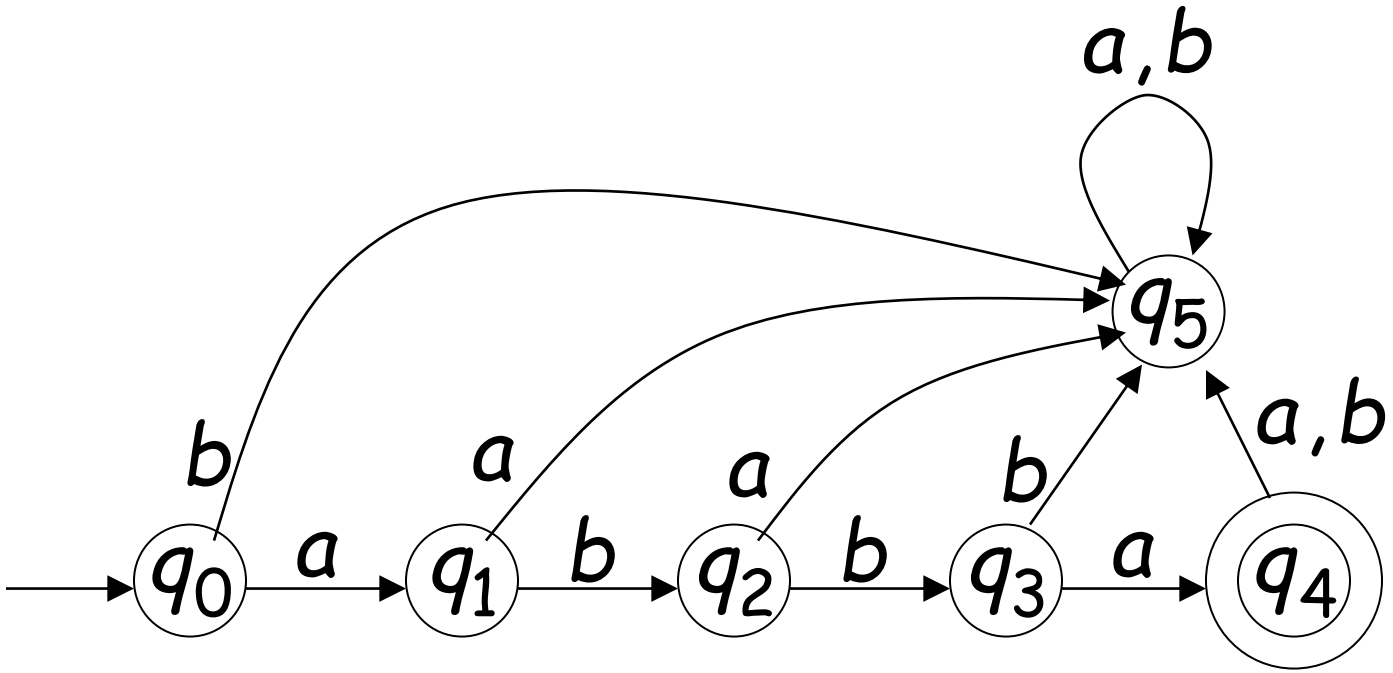
$$\begin{aligned}
\delta(q_0, 110101) &= \delta(q_1, 10101) \\
&= \delta(q_0, 0101) \\
&= \delta(q_2, 101) \\
&= \delta(q_3, 01) \\
&= \delta(q_1, 1) \\
&= \delta(q_0, \Lambda) \\
&= q_0
\end{aligned}$$

Hence,

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

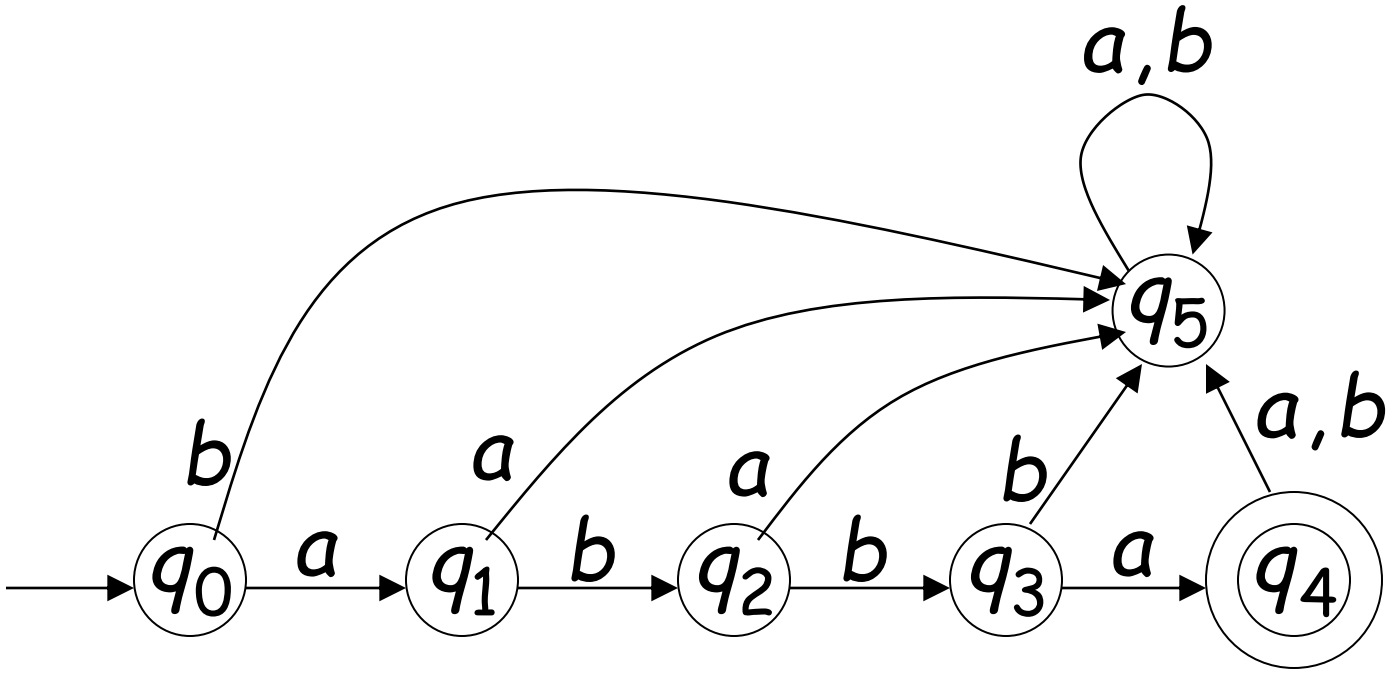
Input alphabet

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

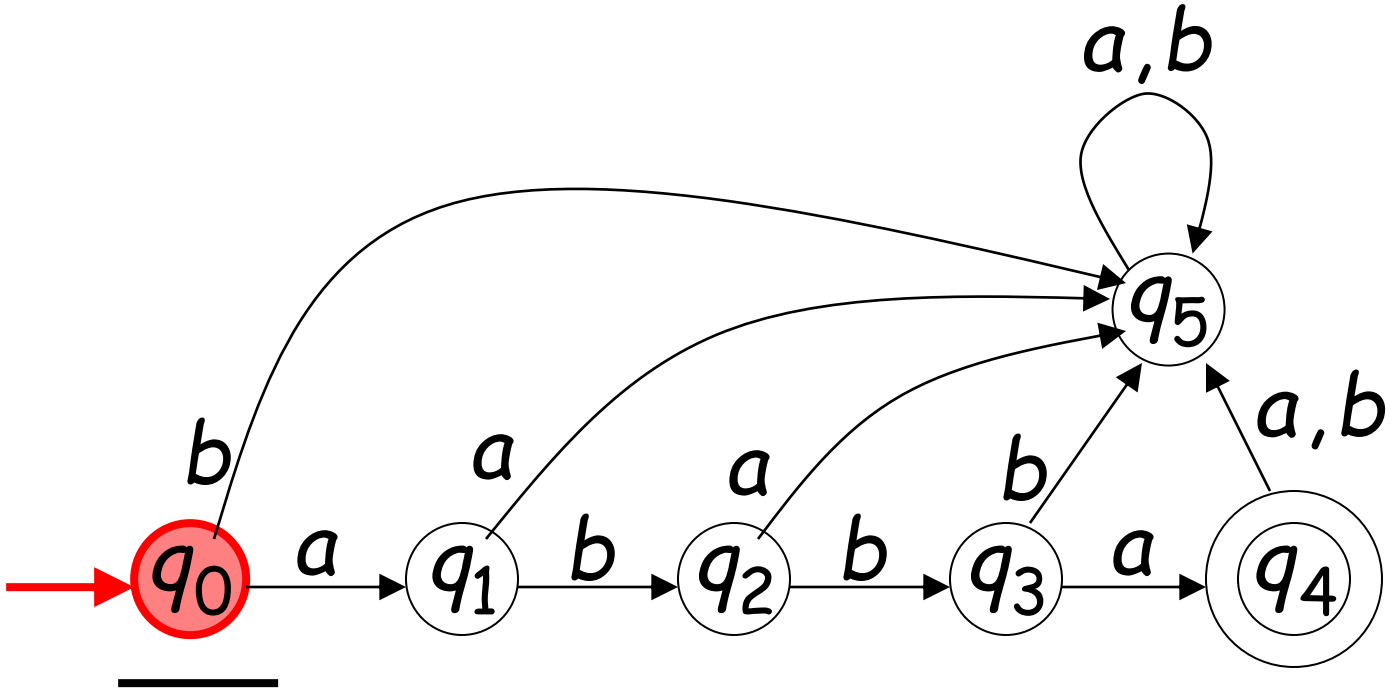


Set of States

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



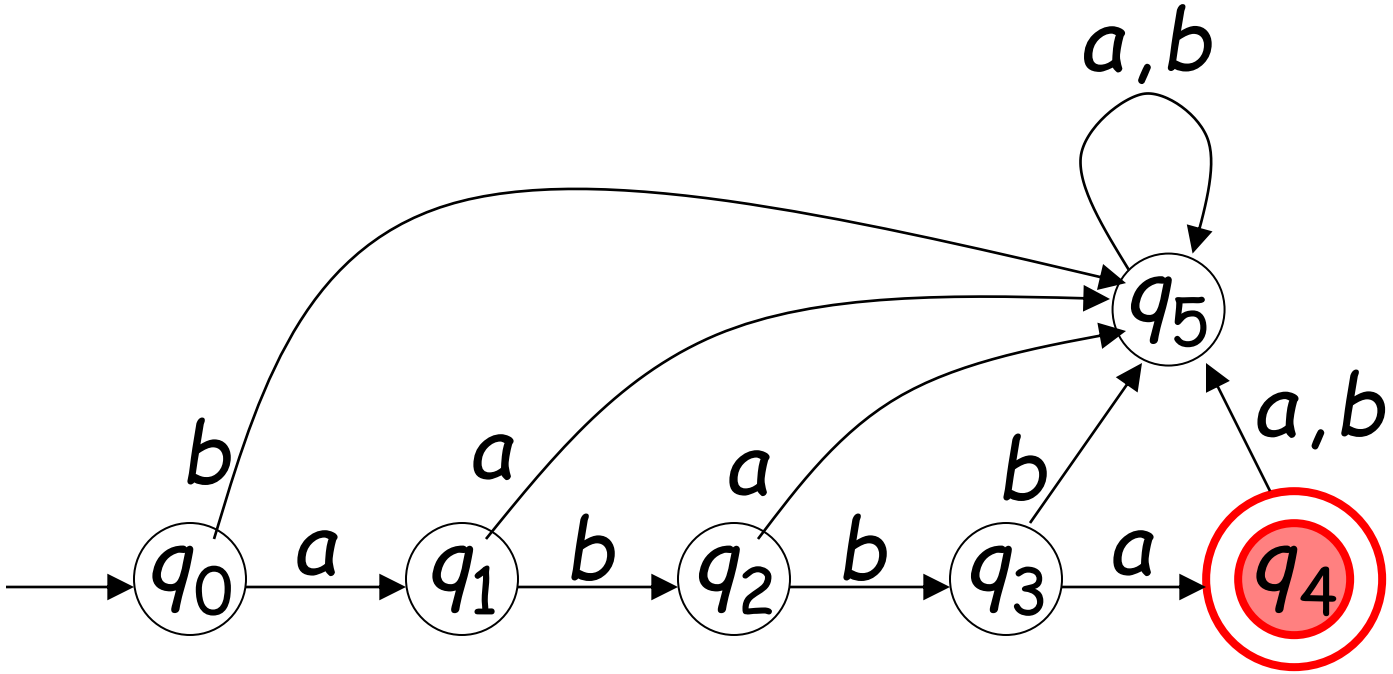
Initial State



Accepting State

Set of Accepting States

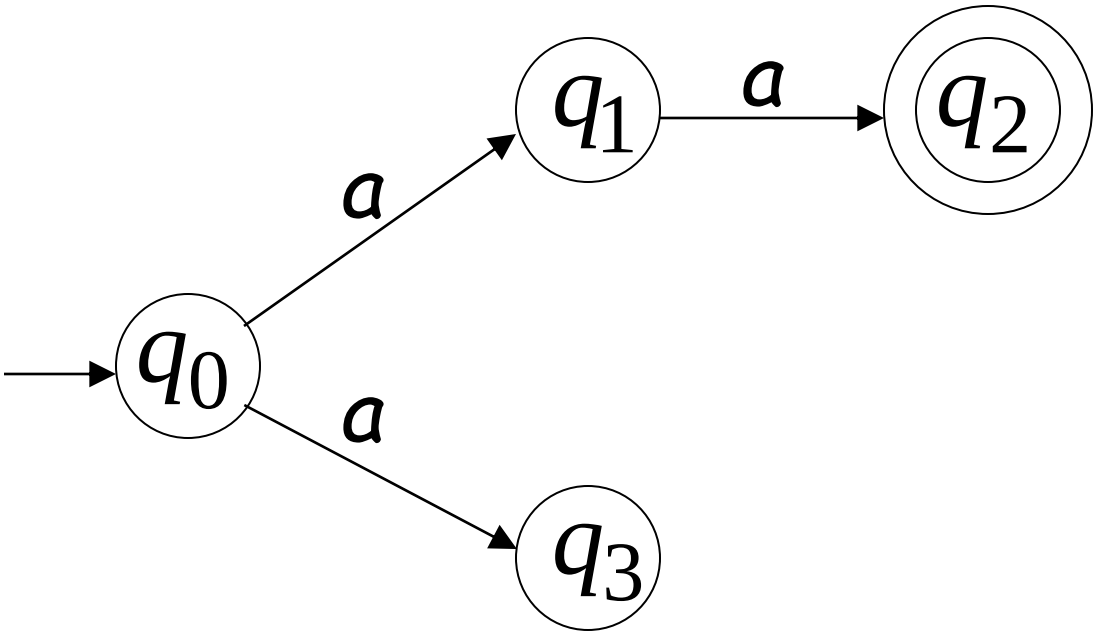
$$F = \{q_4\}$$



Non-Deterministic Finite Automata

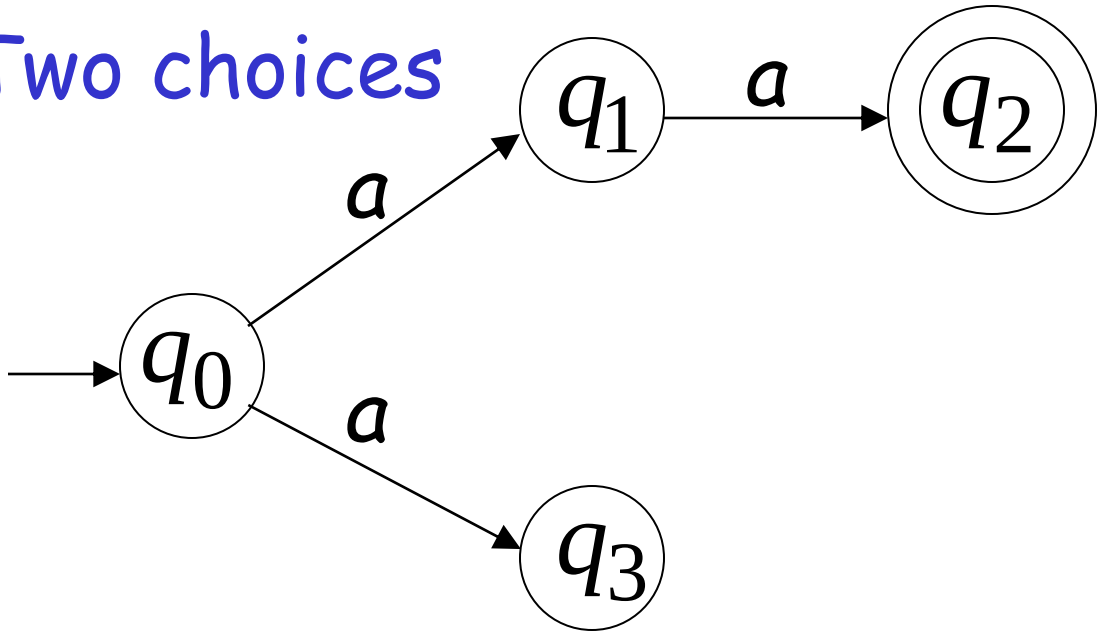
Nondeterministic Finite Automaton

Alphabet = $\{a\}$

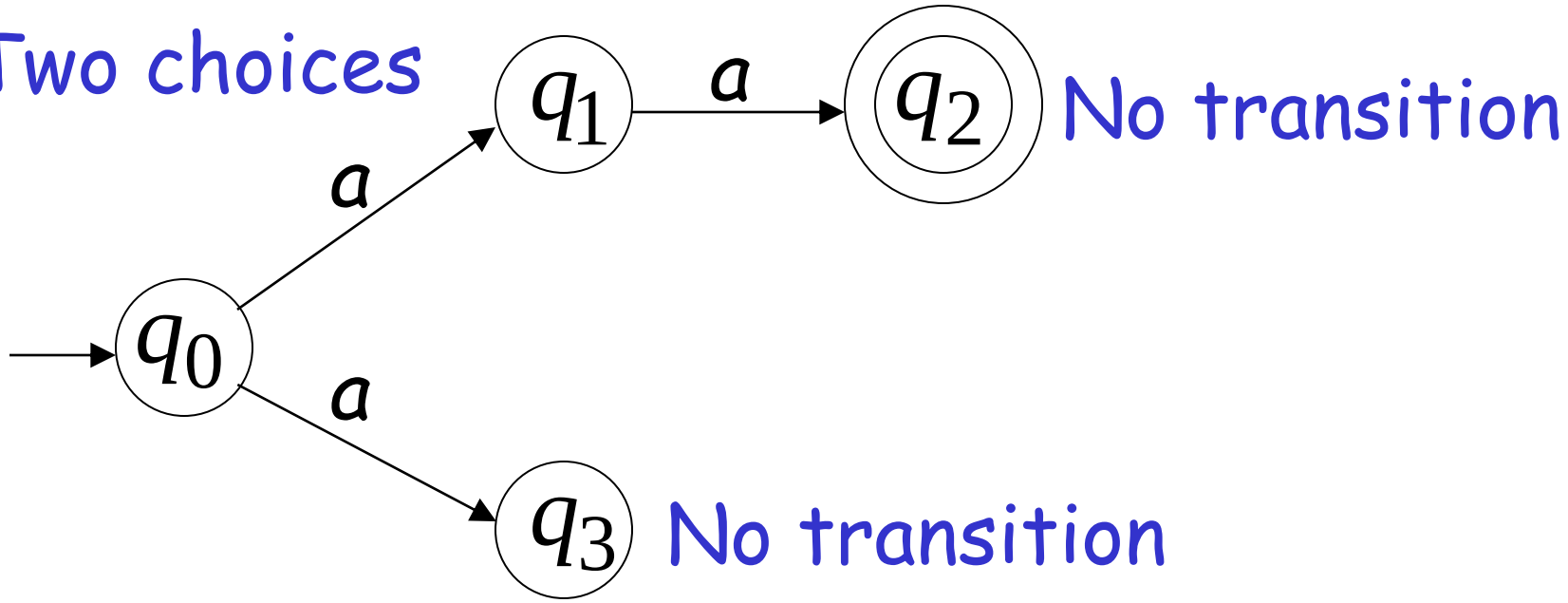


Alphabet = $\{a\}$

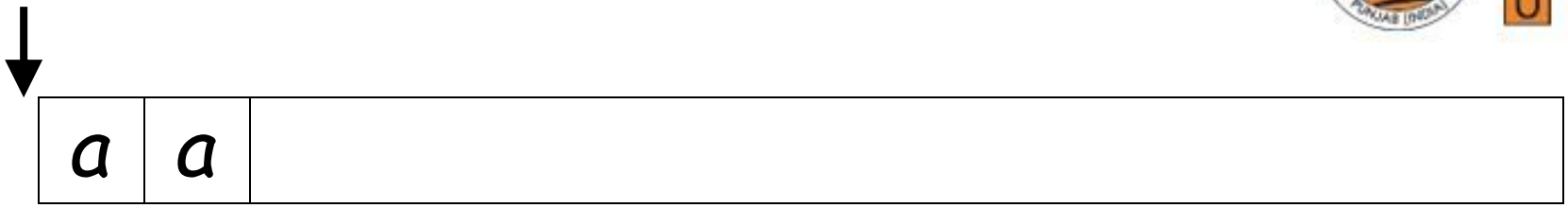
Two choices

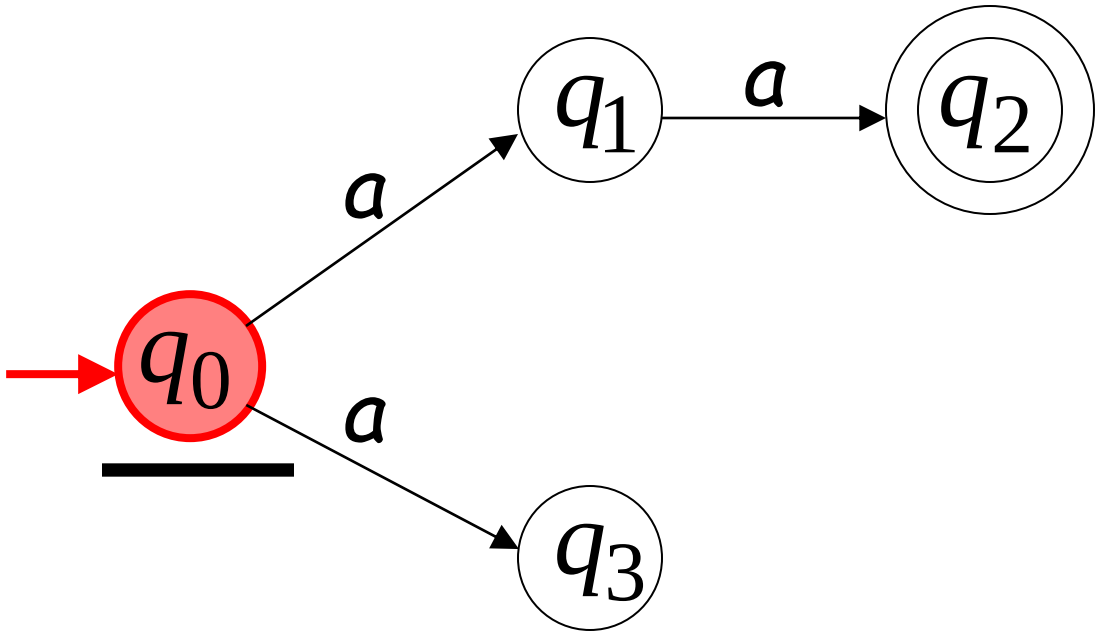


Alphabet = $\{a\}$

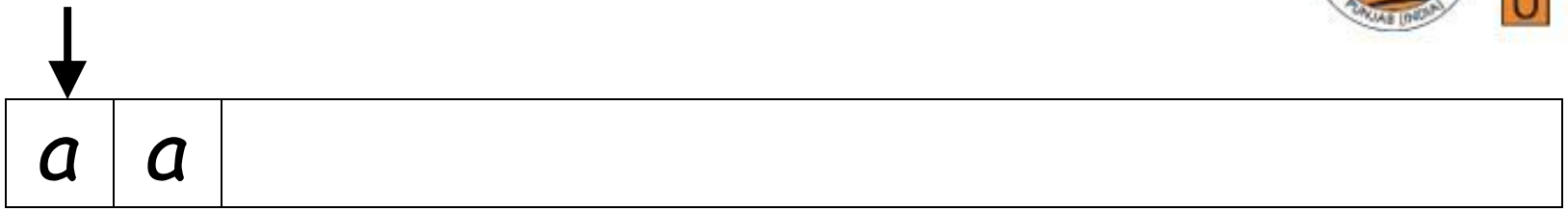


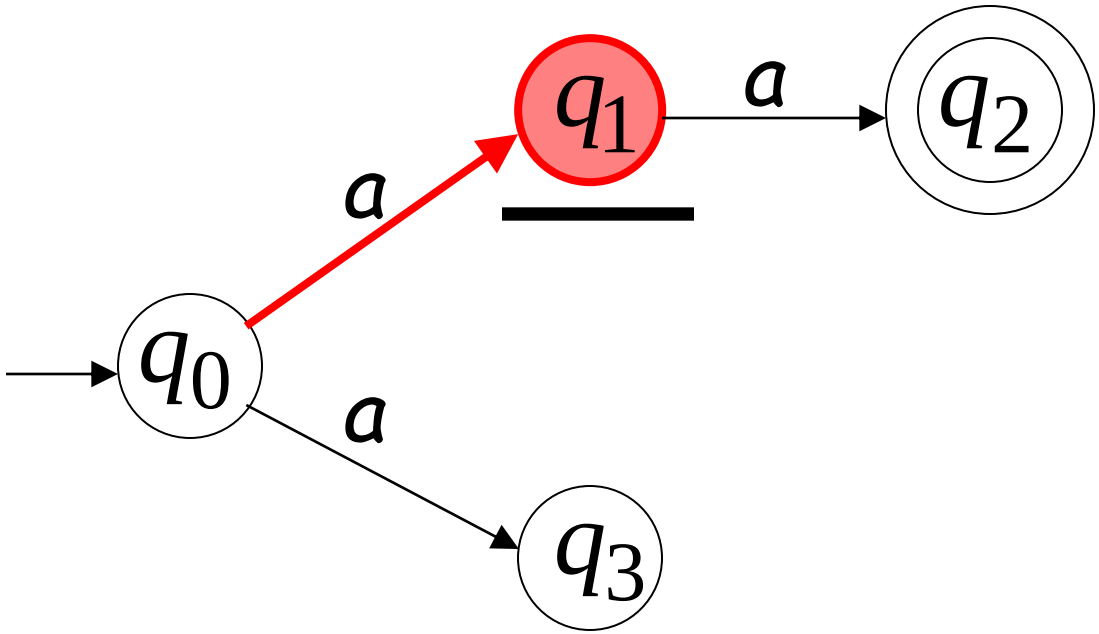
First Choice



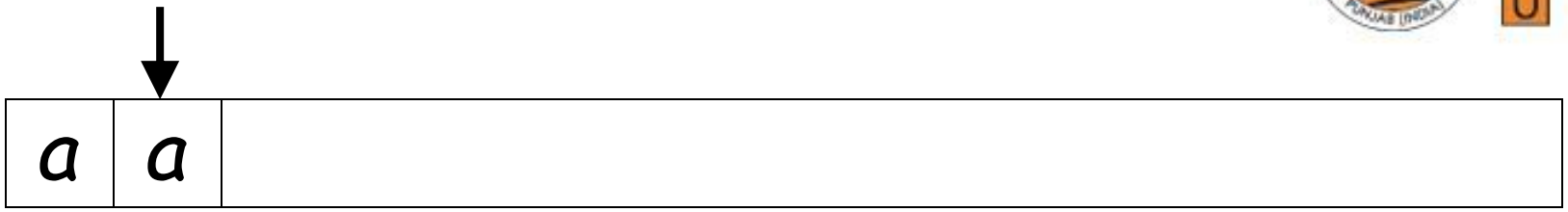


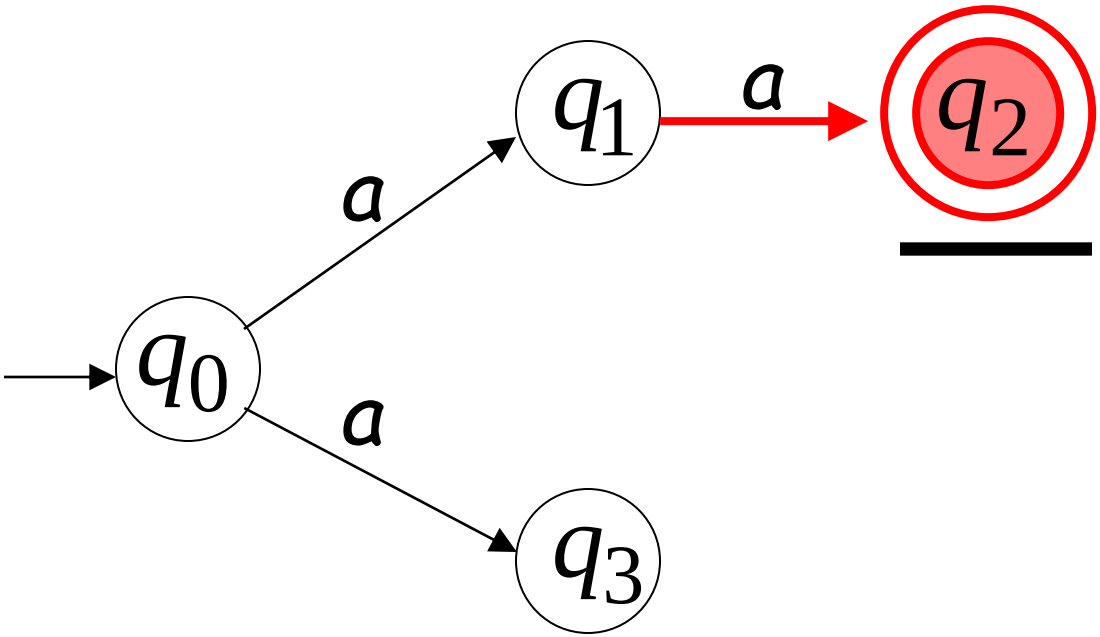
First Choice



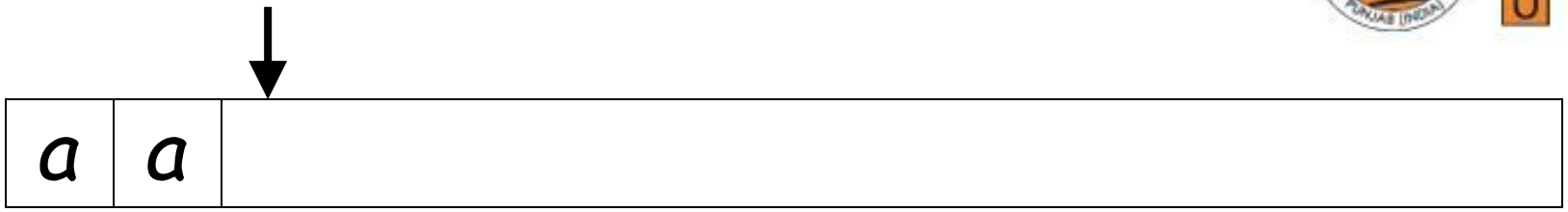


First Choice

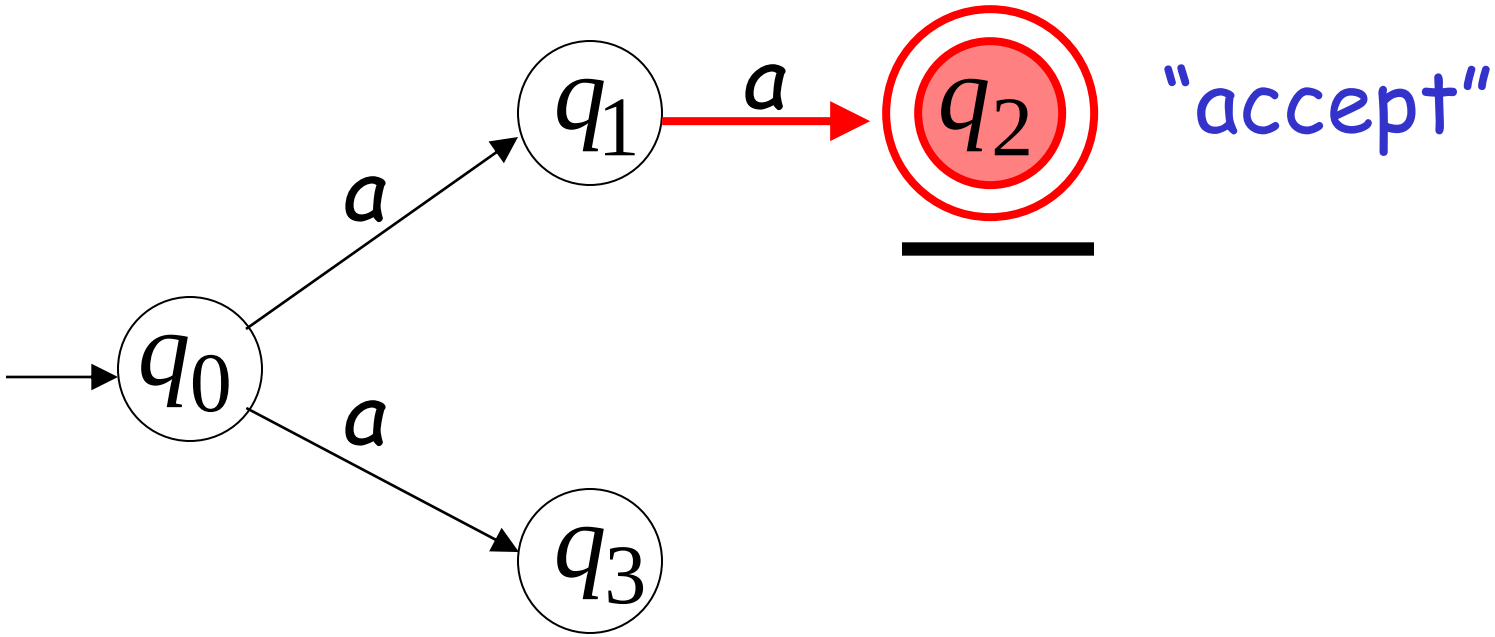




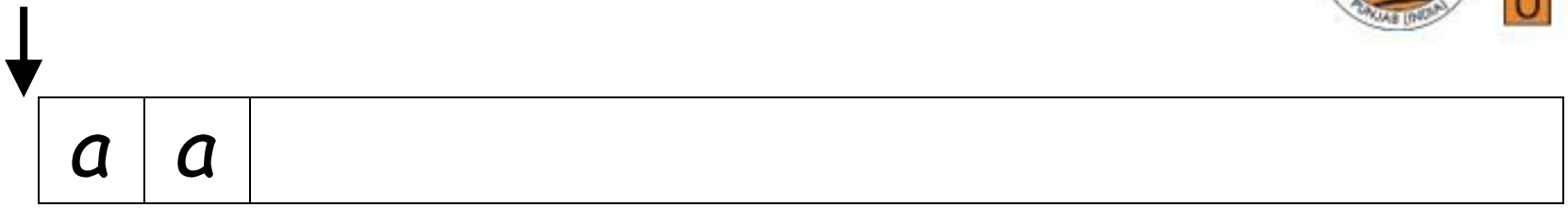
First Choice

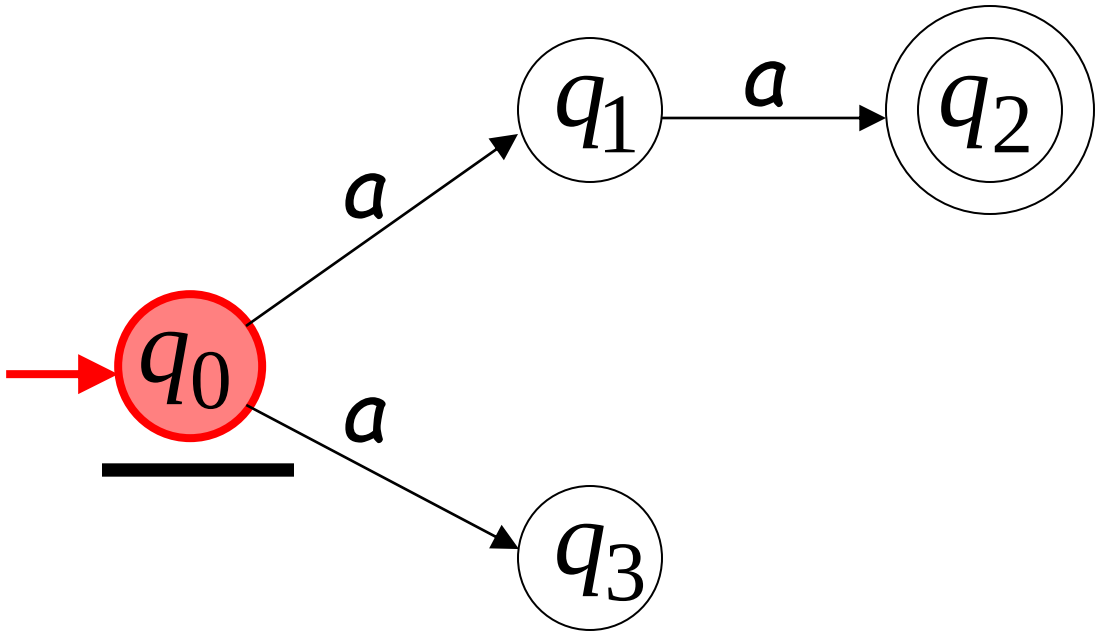


All input is consumed

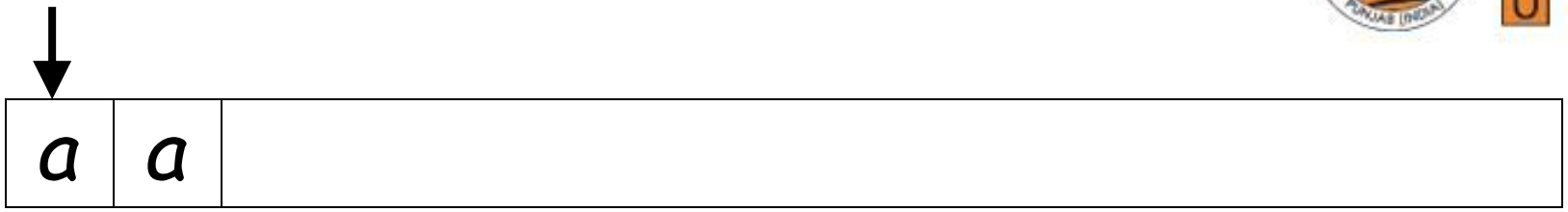


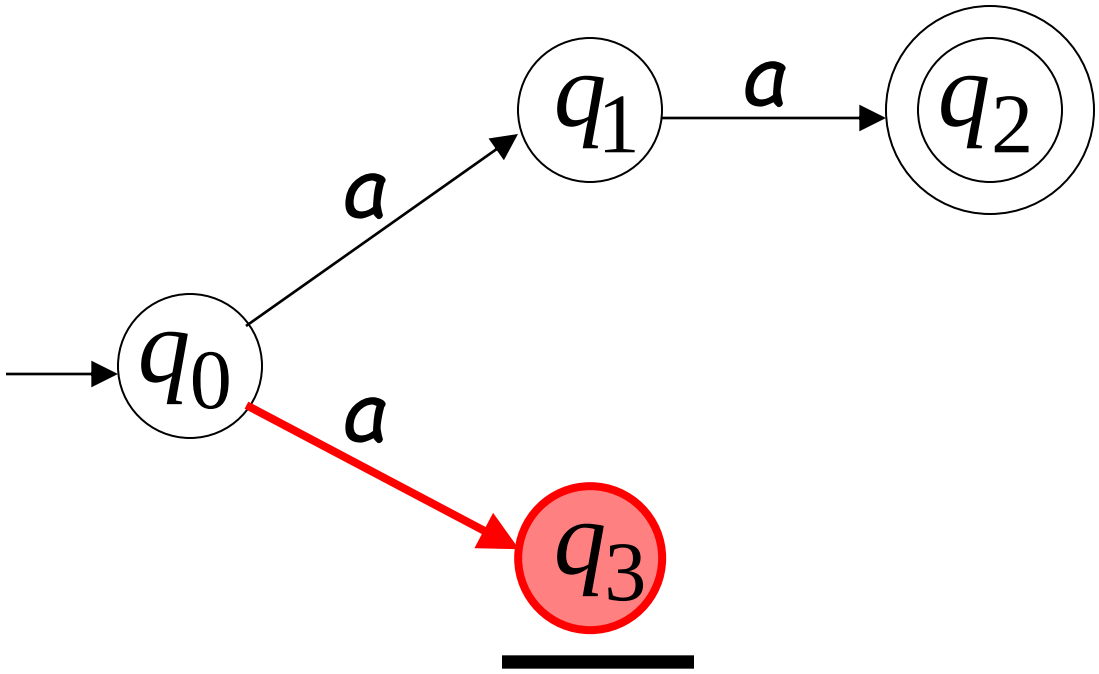
Second Choice



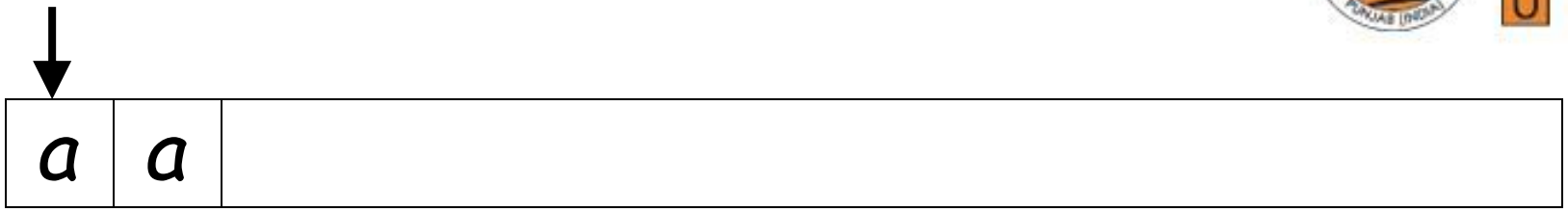


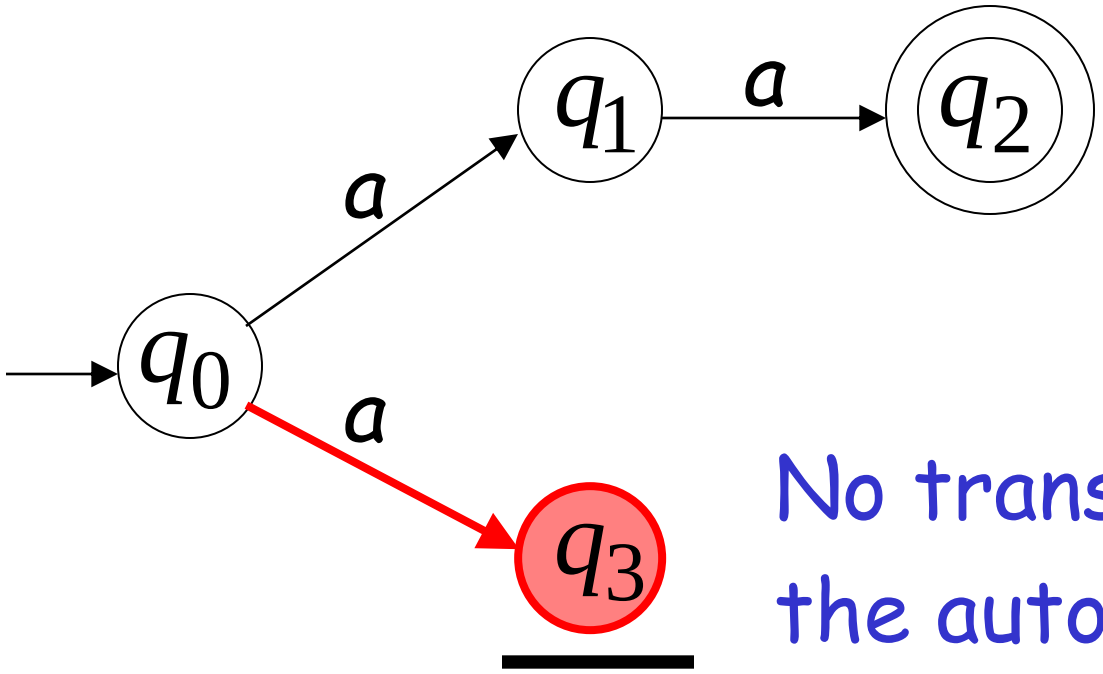
Second Choice





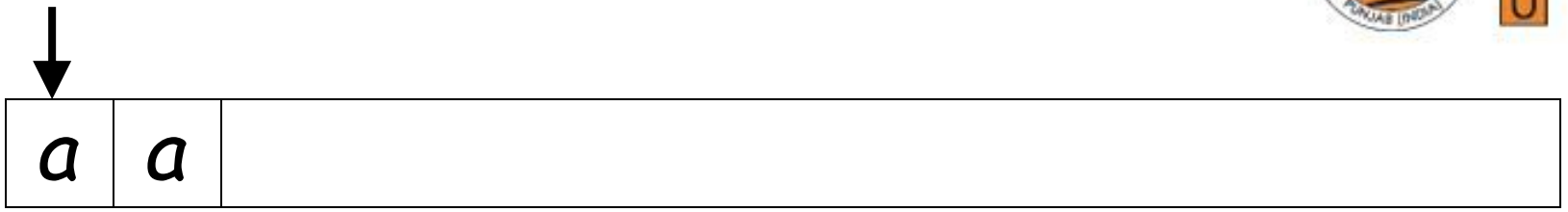
Second Choice



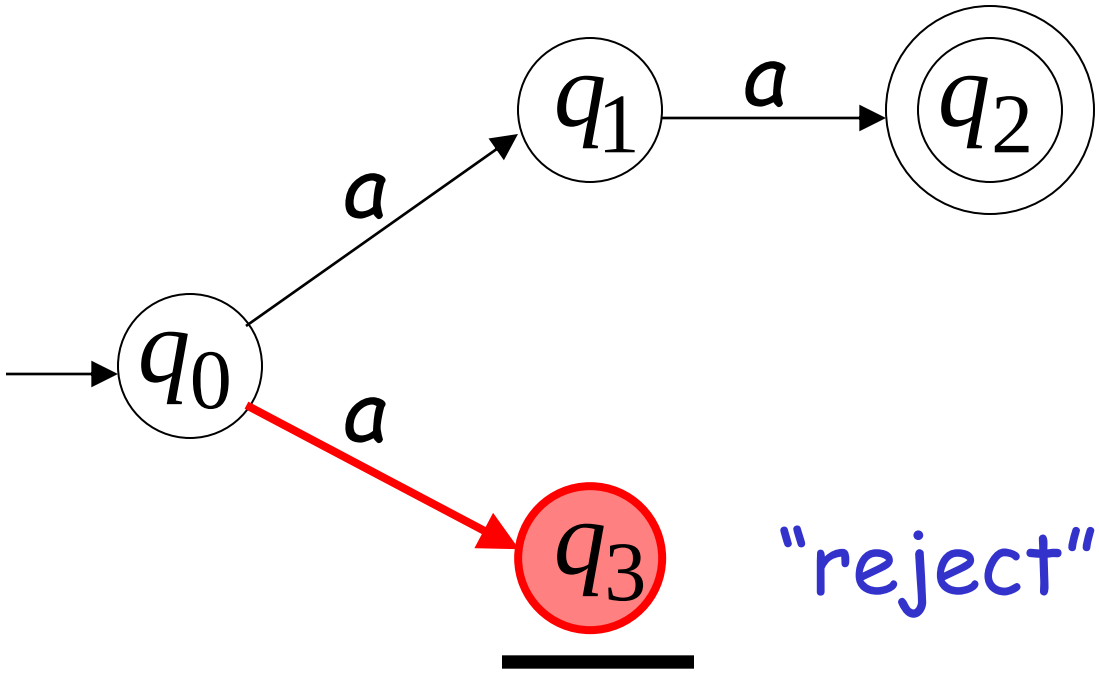


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



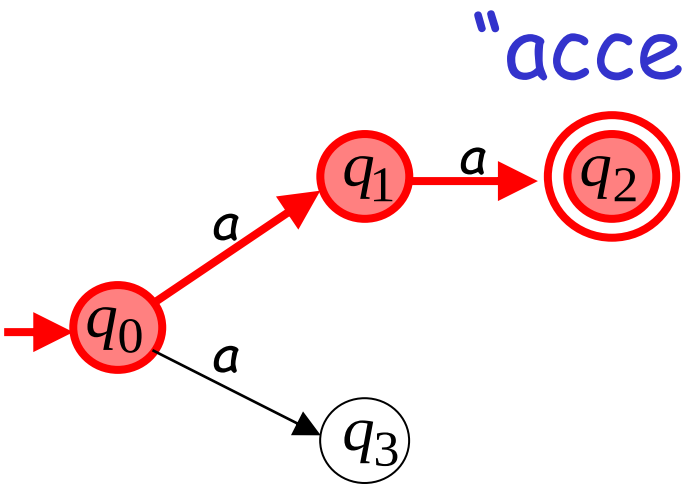
An NFA accepts a string:

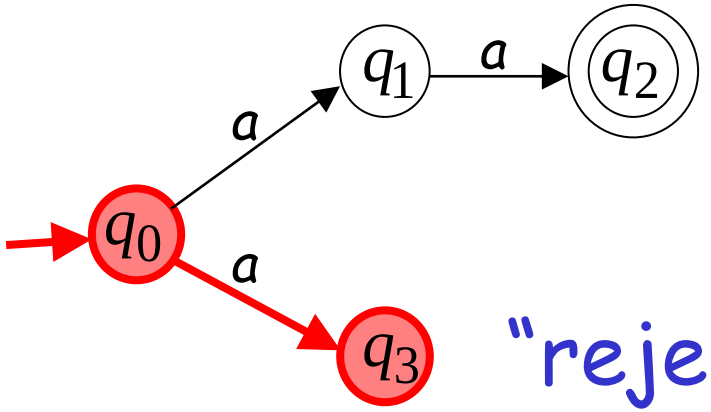
when there is a computation of the NFA that accepts the string

There is a computation:
all the input is consumed and the automaton is in an accepting state

Example

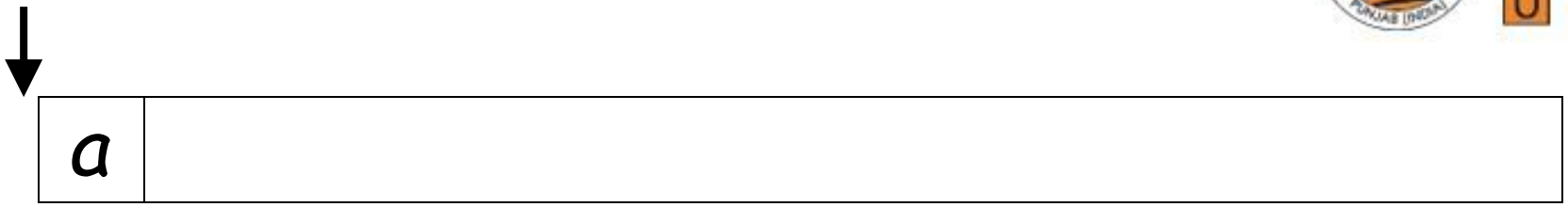
aa is accepted by the NFA:

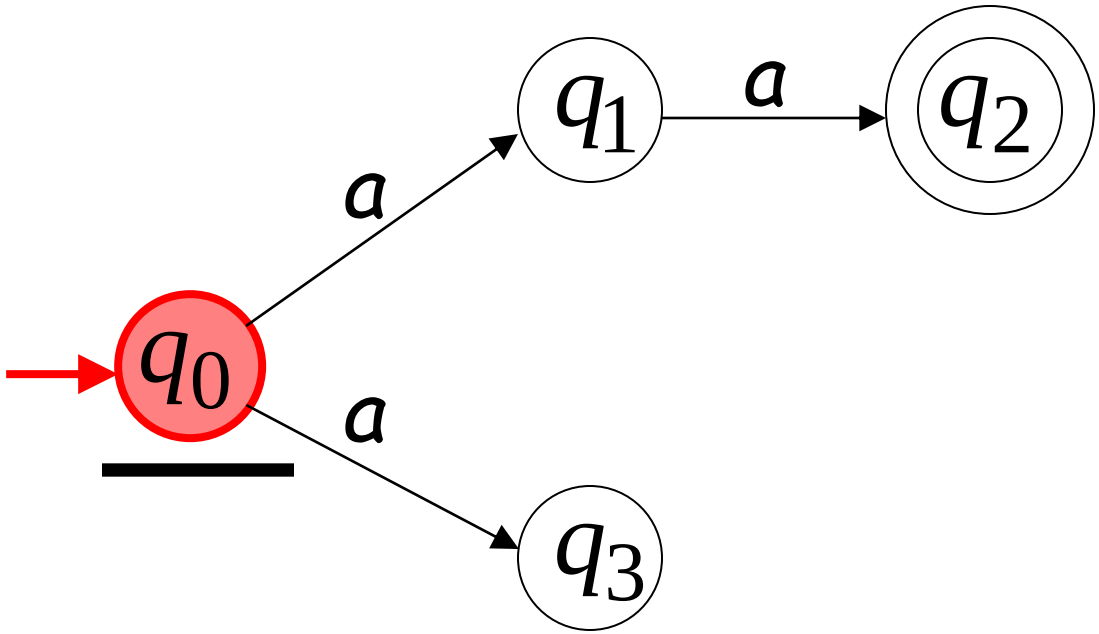




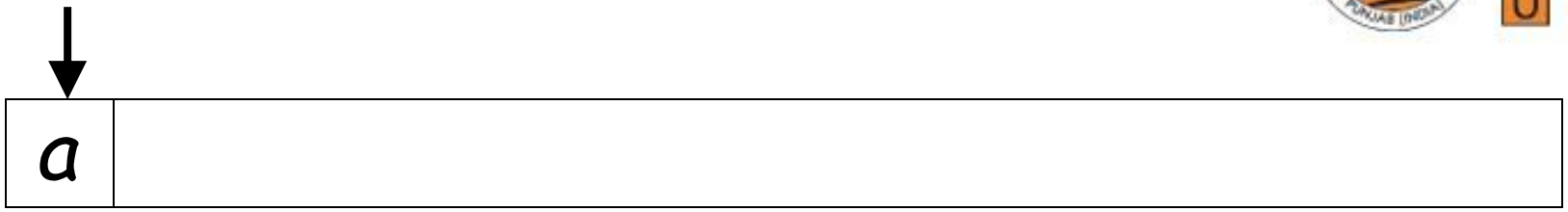
because this computation accepts aa

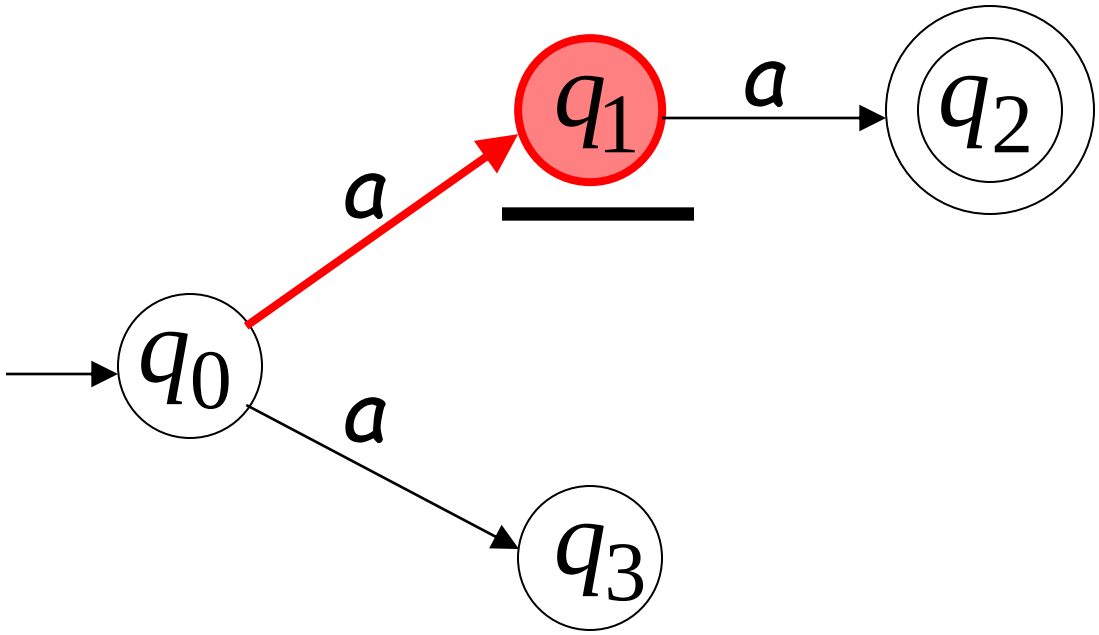
Rejection example



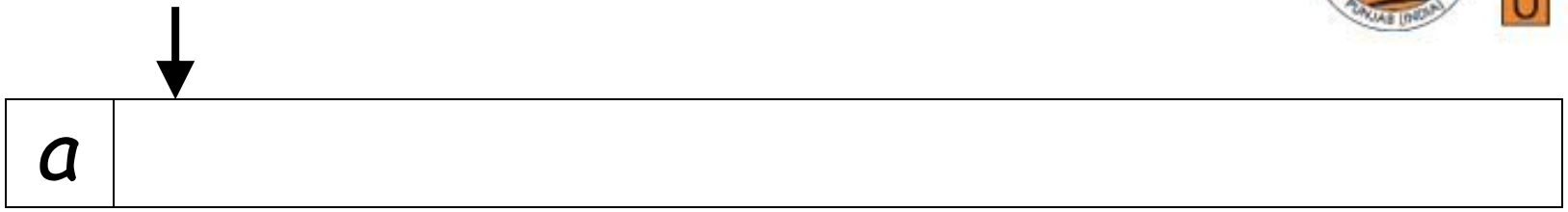


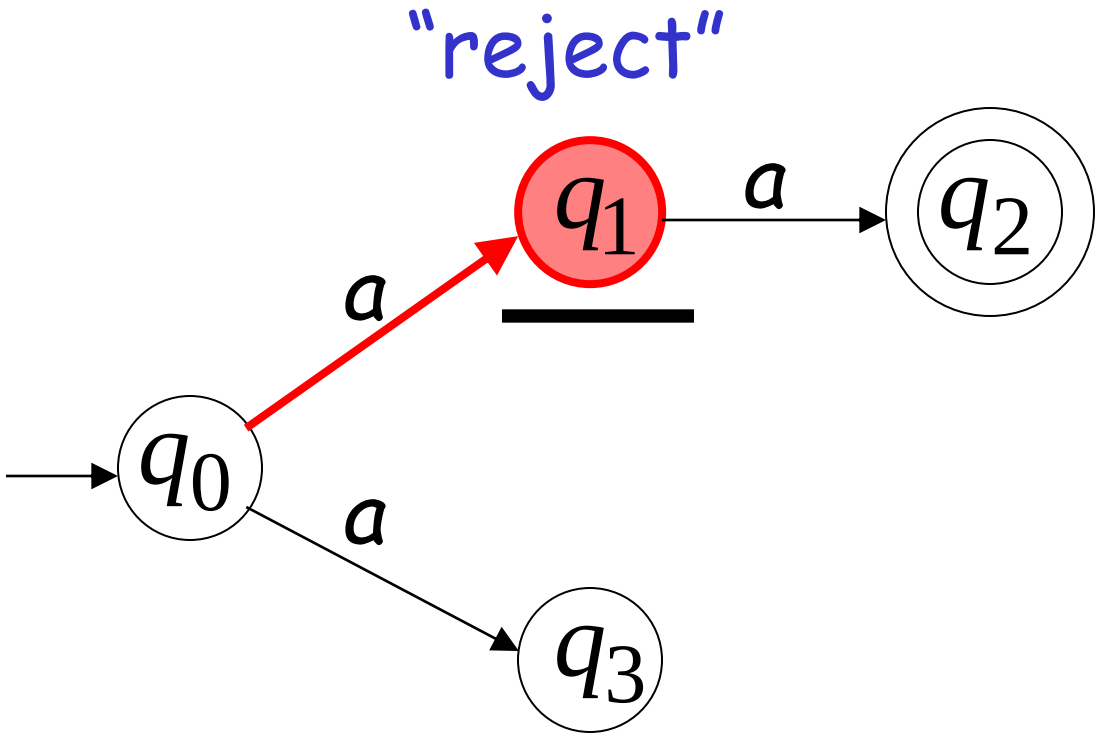
First Choice



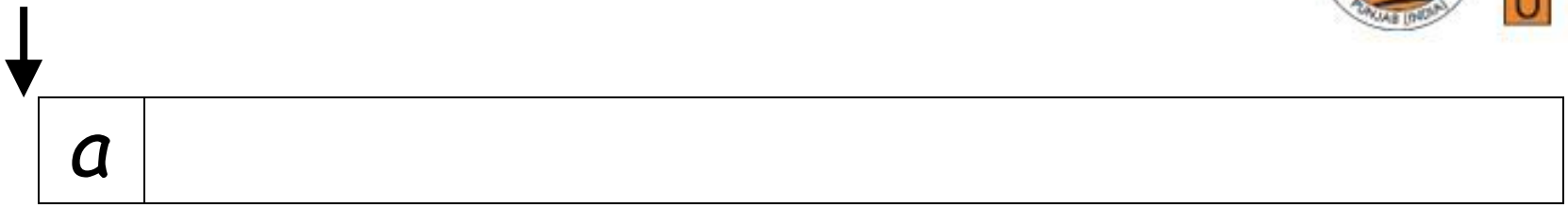


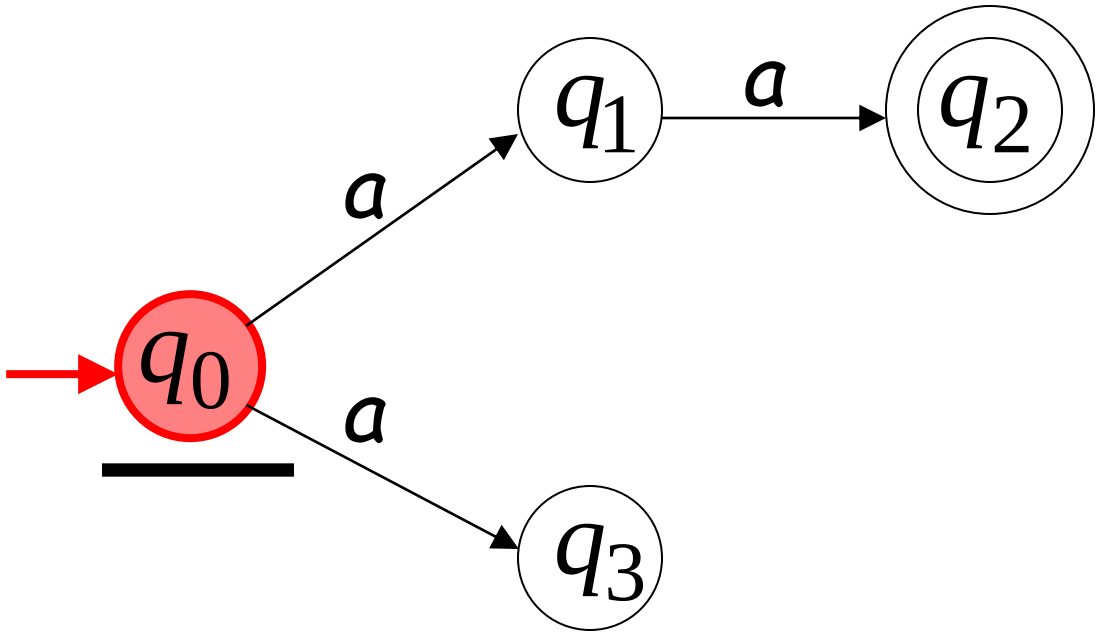
First Choice



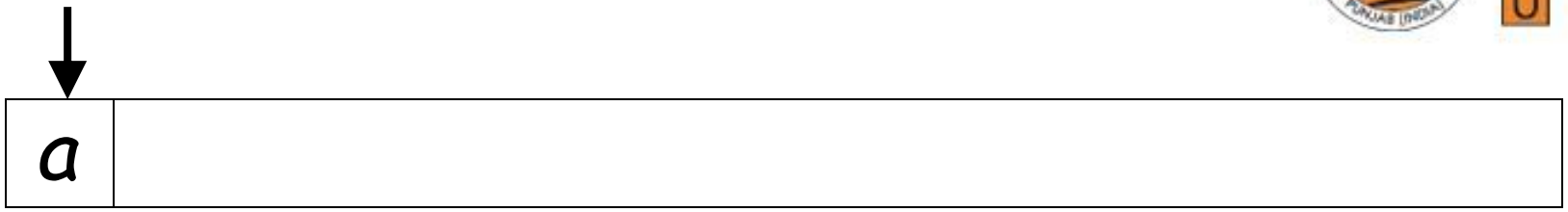


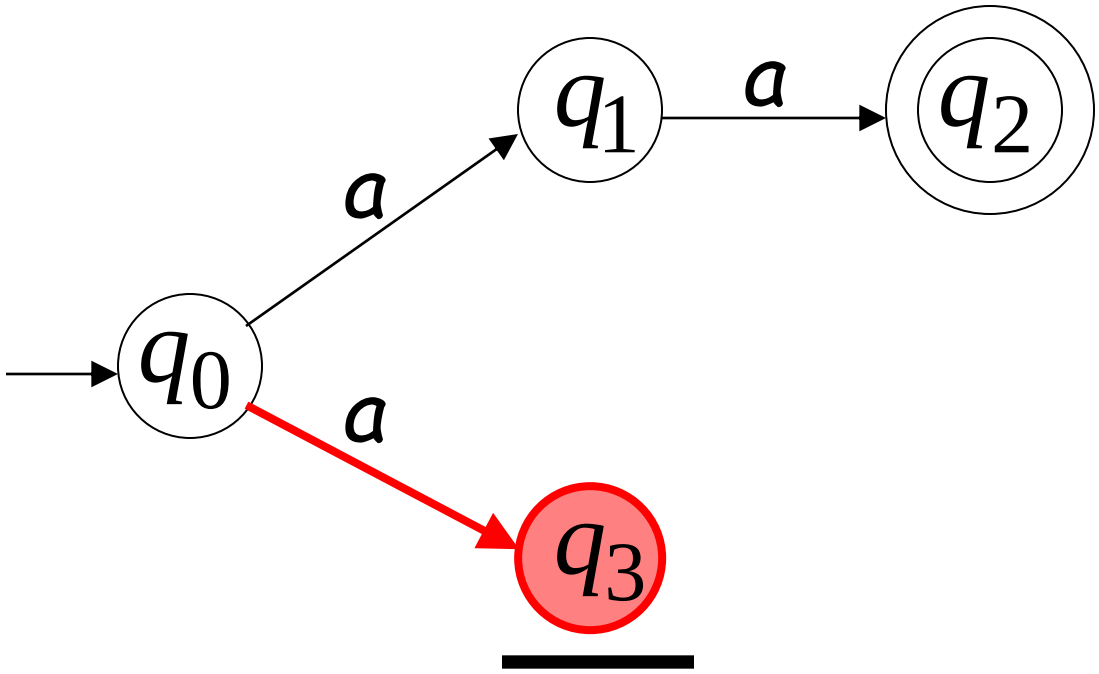
Second Choice



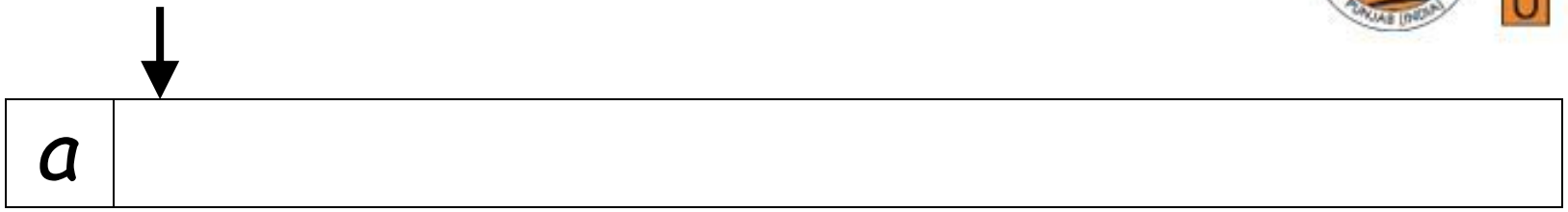


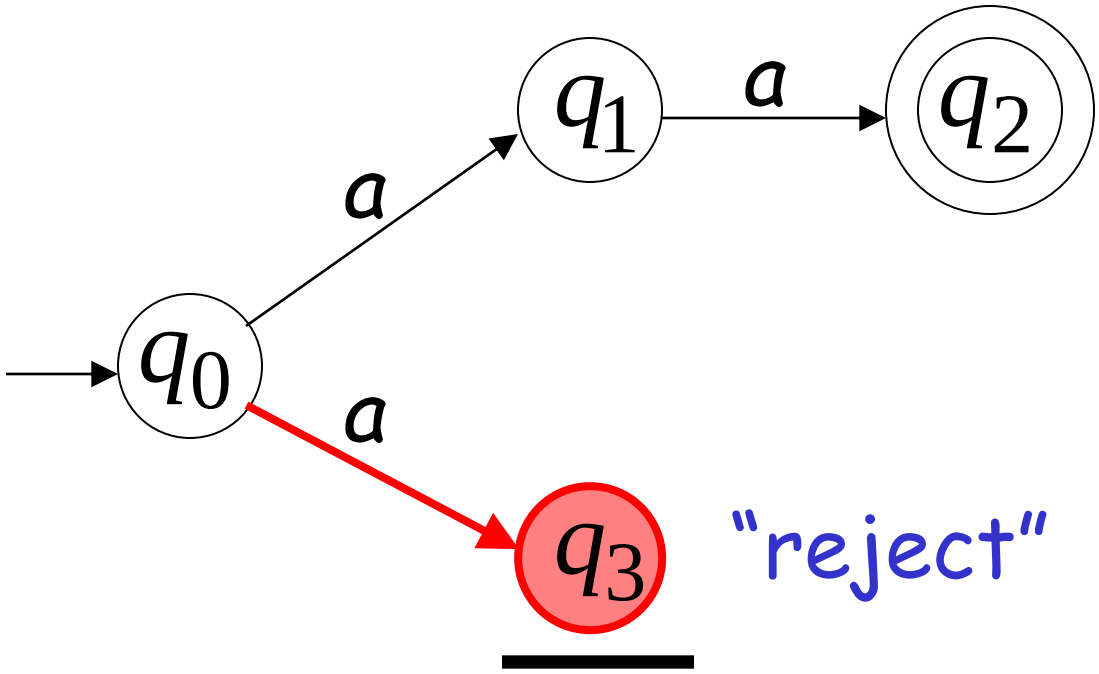
Second Choice





Second Choice





An NFA rejects a string:

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

For each computation:

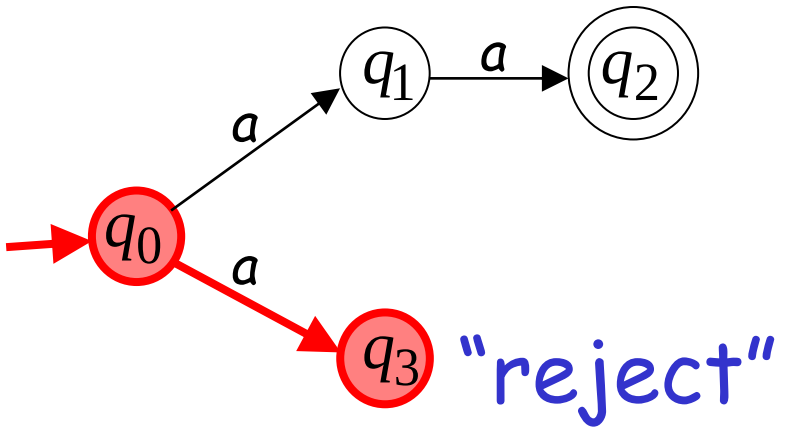
- All the input is consumed and the automaton is in a non final state

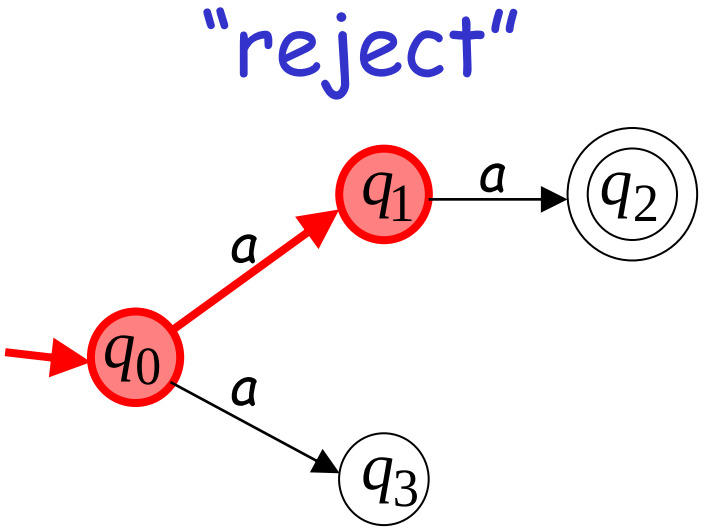
OR

- The input cannot be consumed

Example

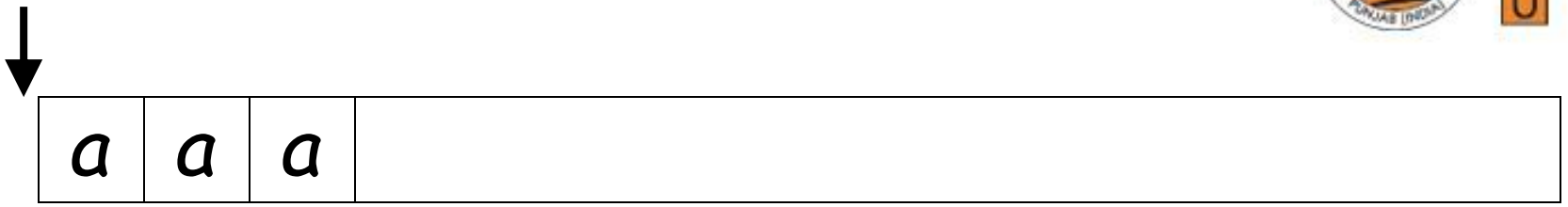
a is rejected by the NFA:

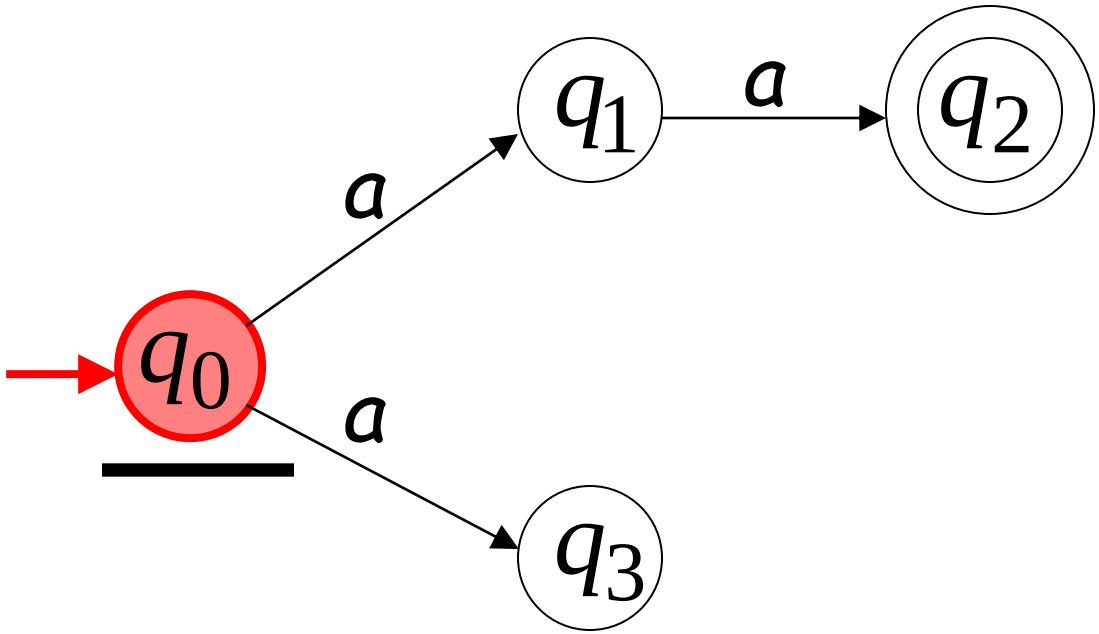




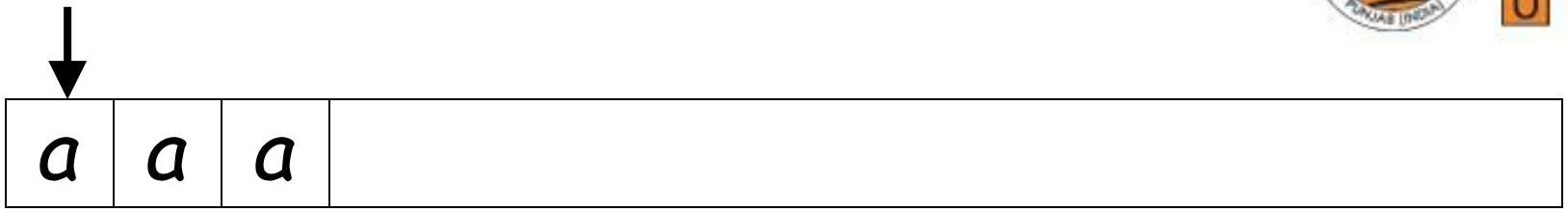
All possible computations lead to rejection

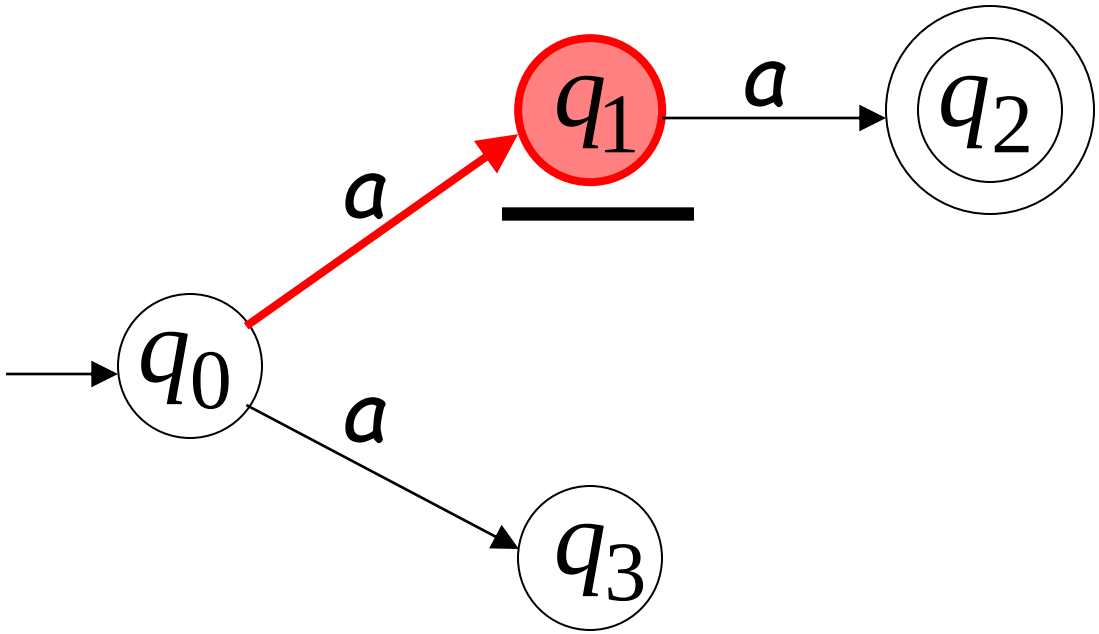
Rejection example



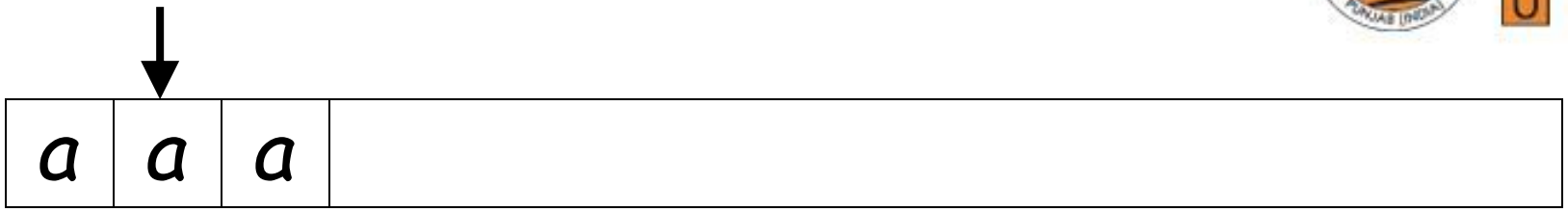


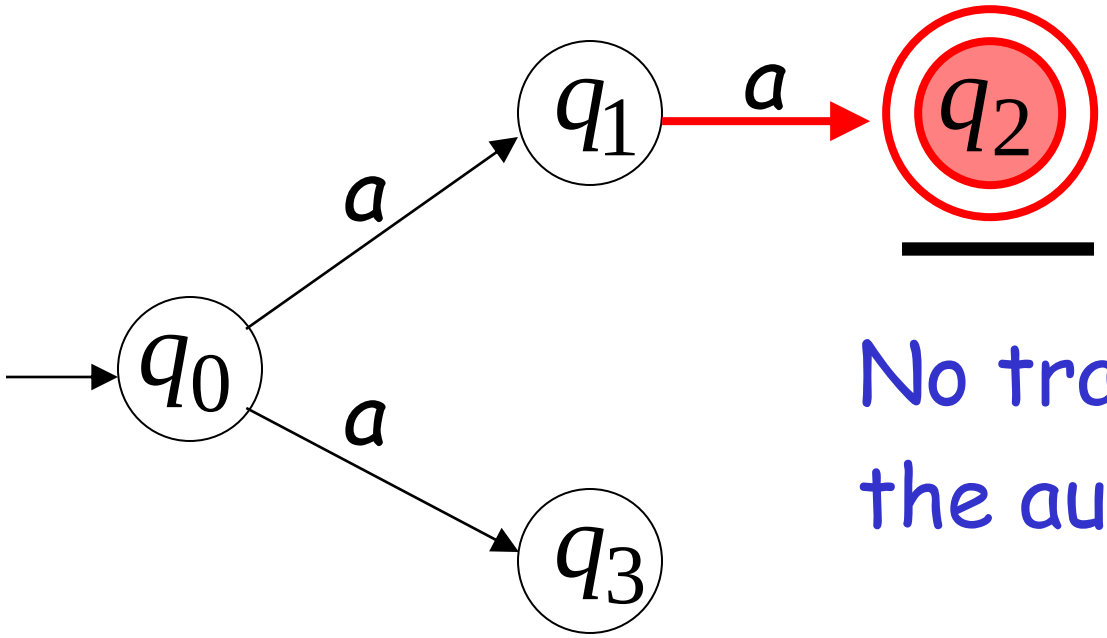
First Choice





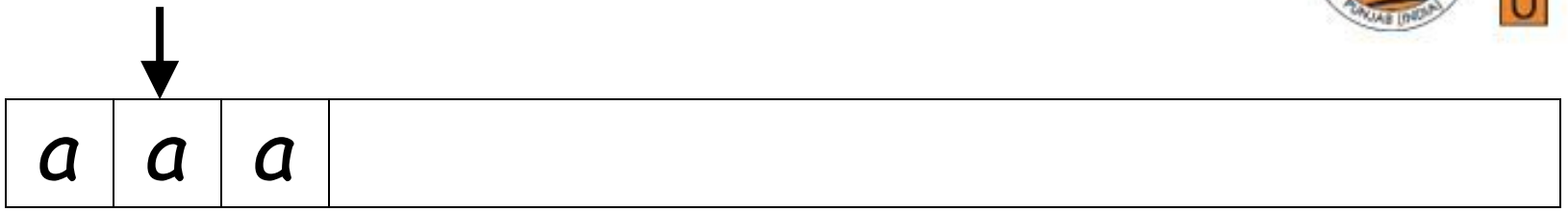
First Choice



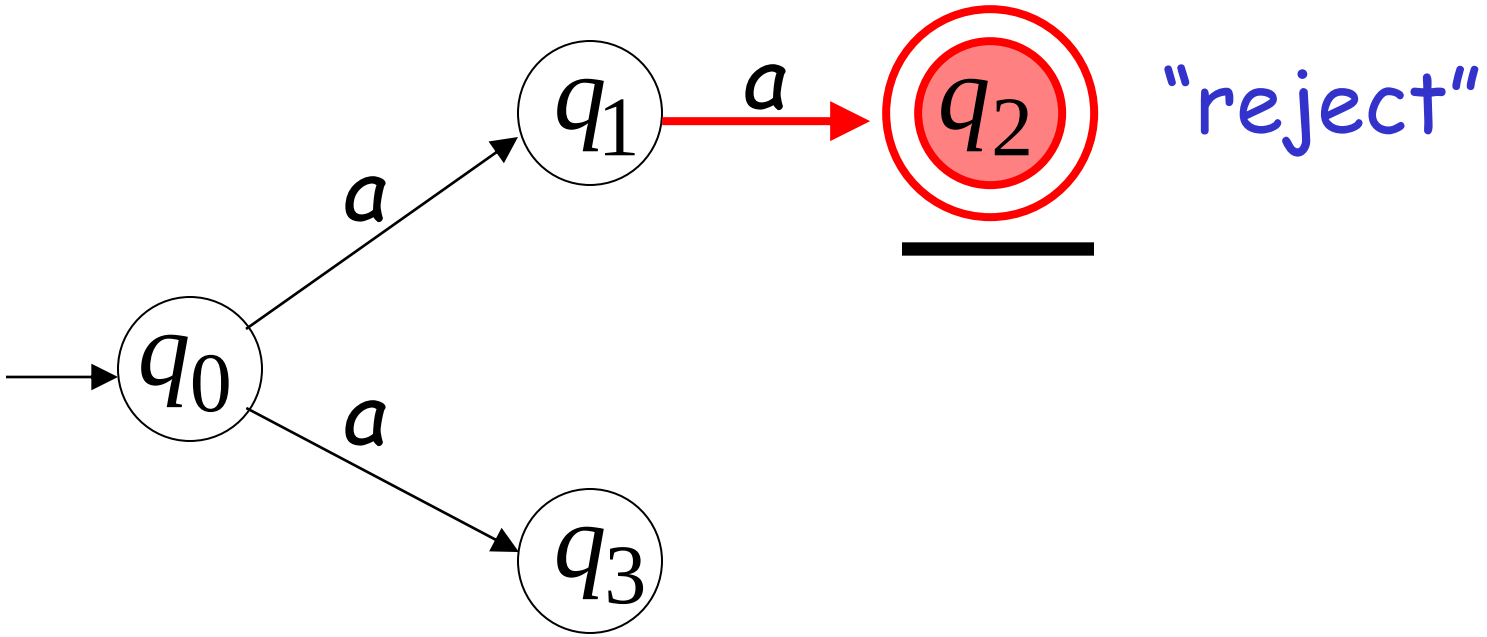


No transition:
the automaton hangs

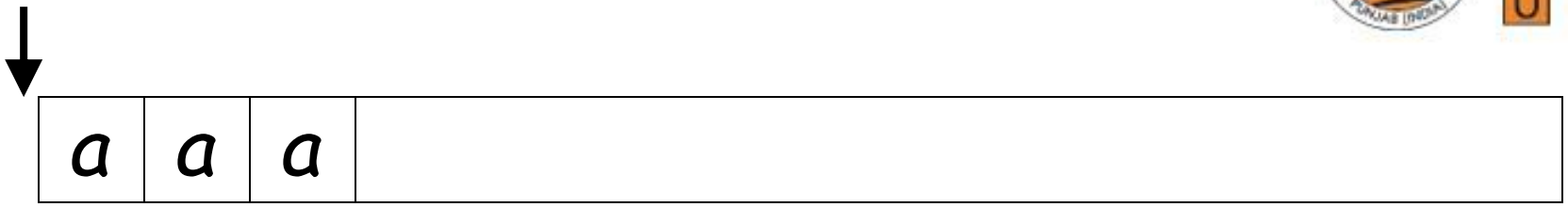
First Choice

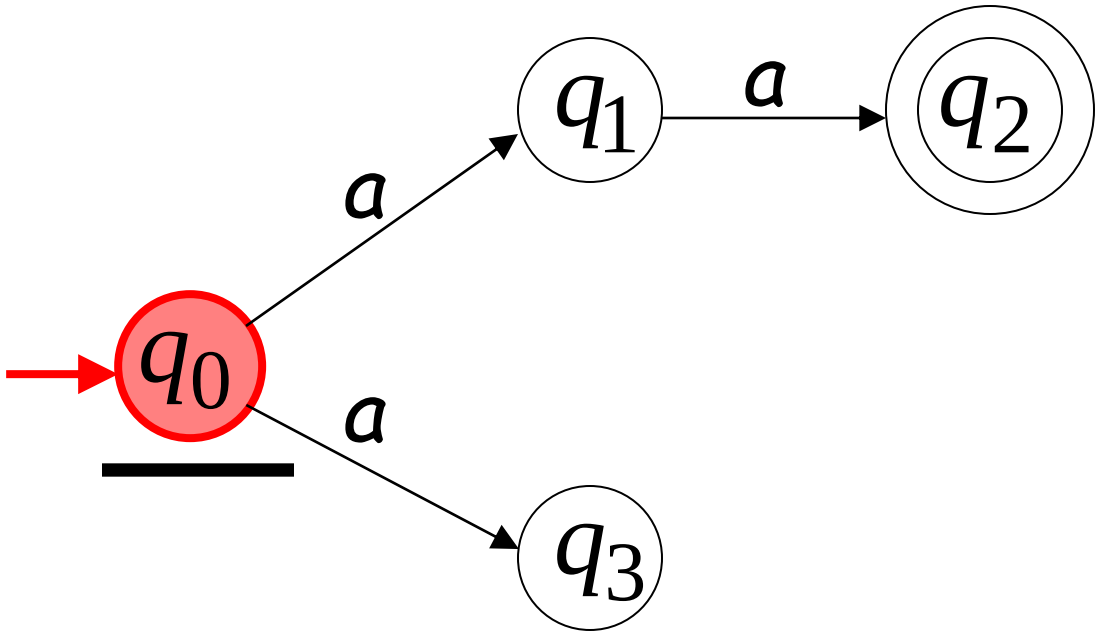


Input cannot be consumed

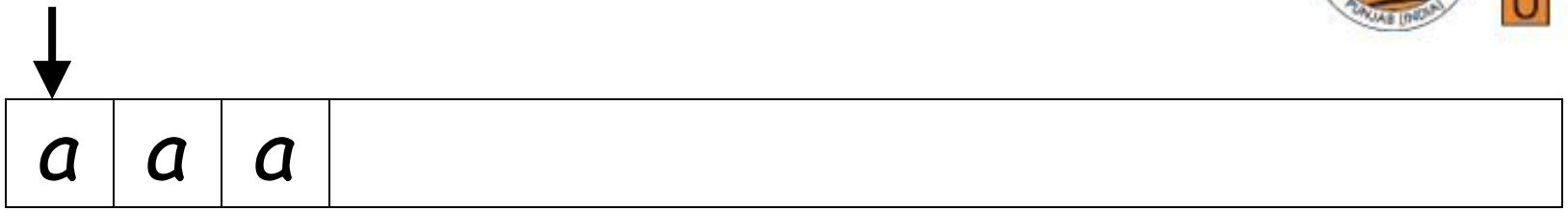


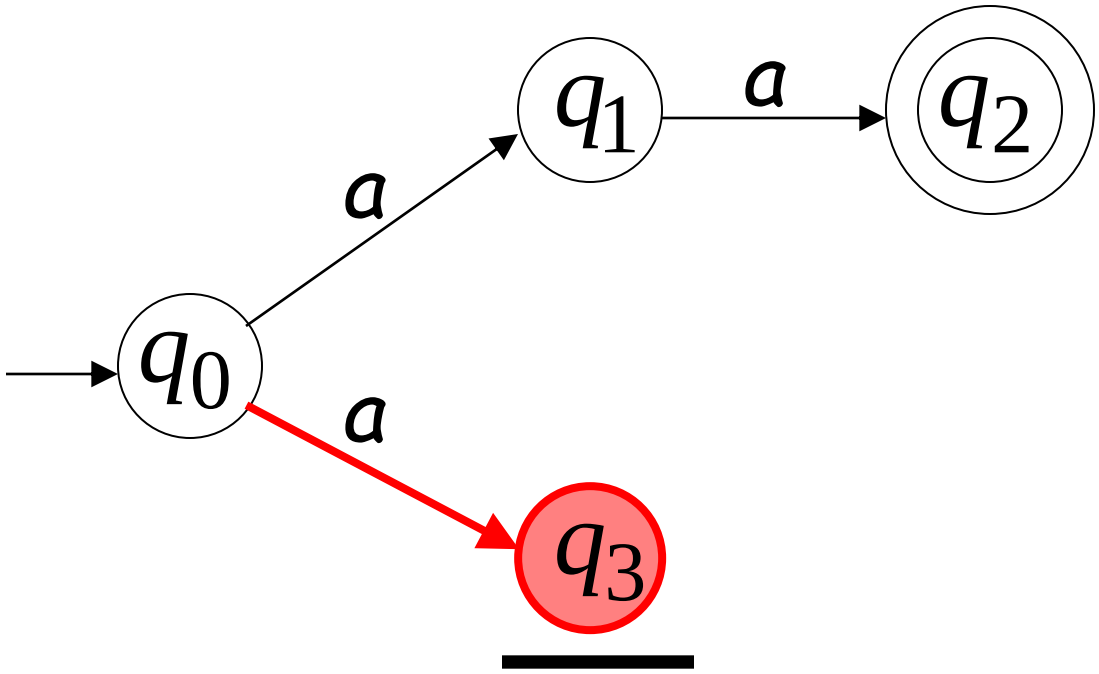
Second Choice



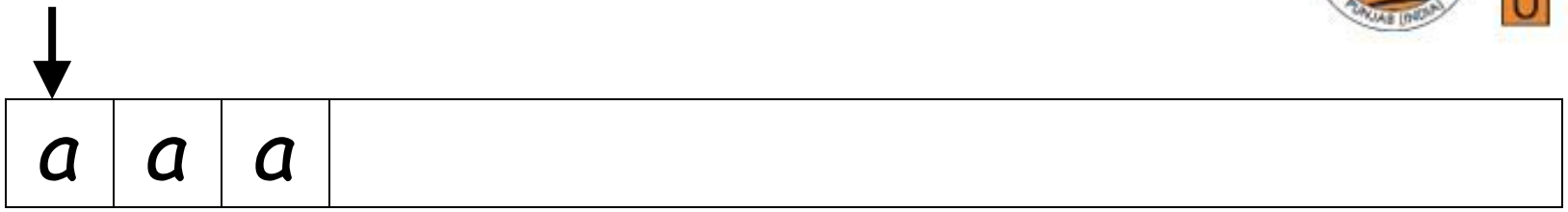


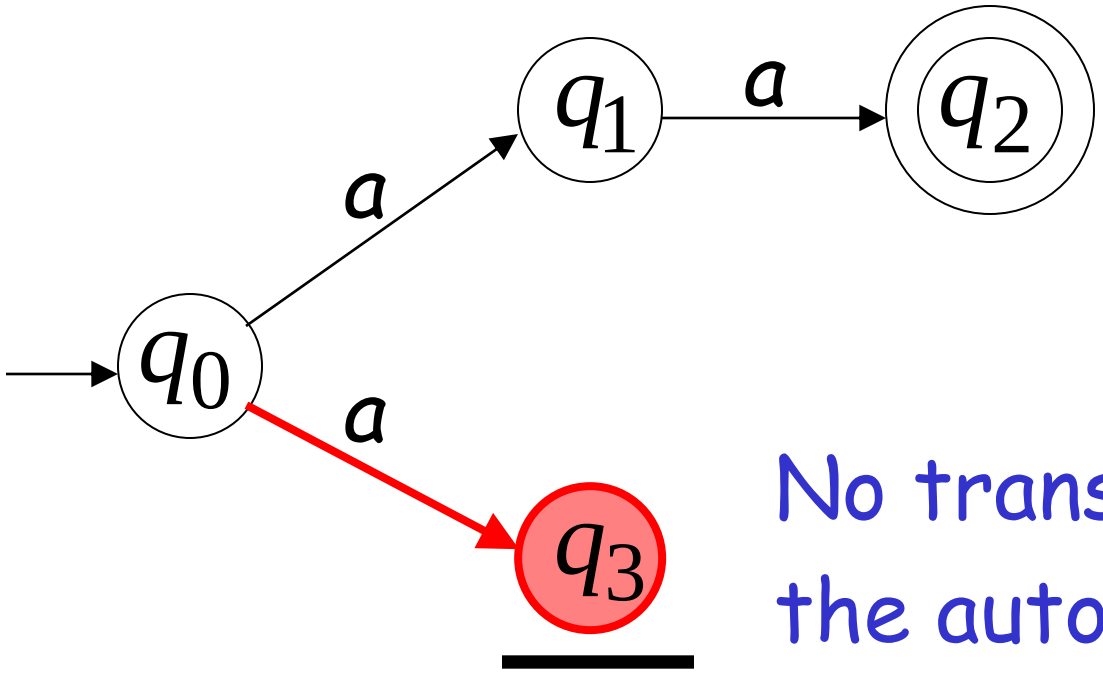
Second Choice





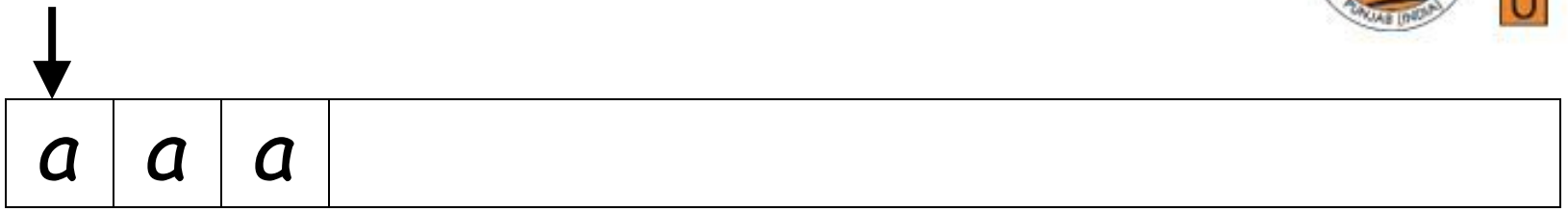
Second Choice



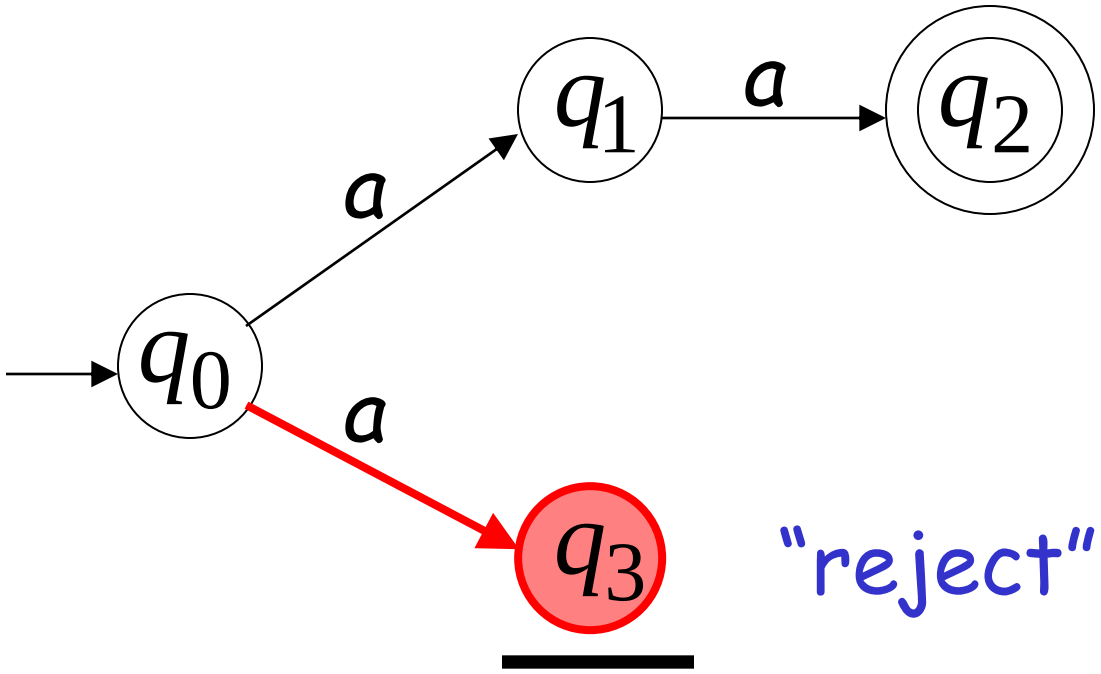


No transition:
the automaton hangs

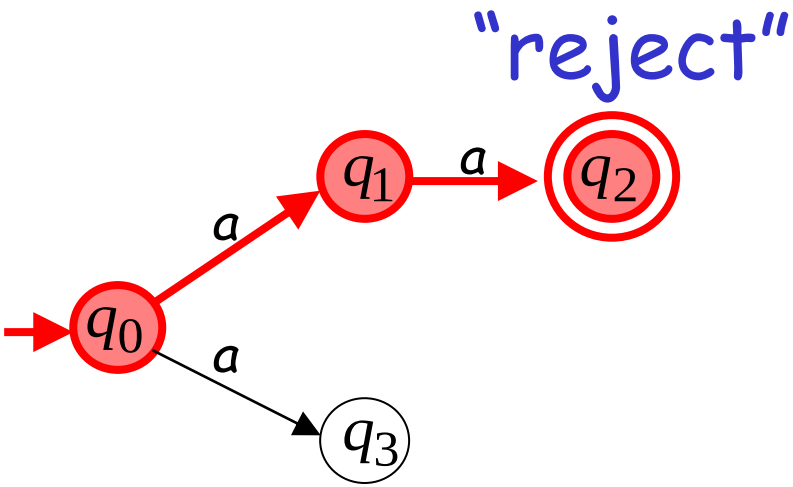
Second Choice

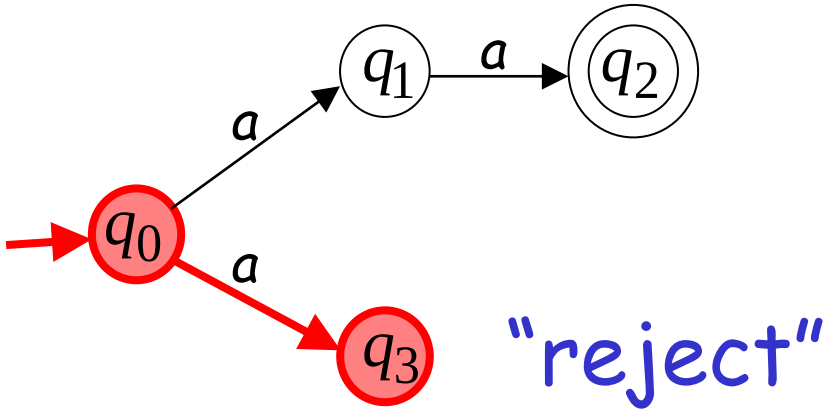


Input cannot be consumed



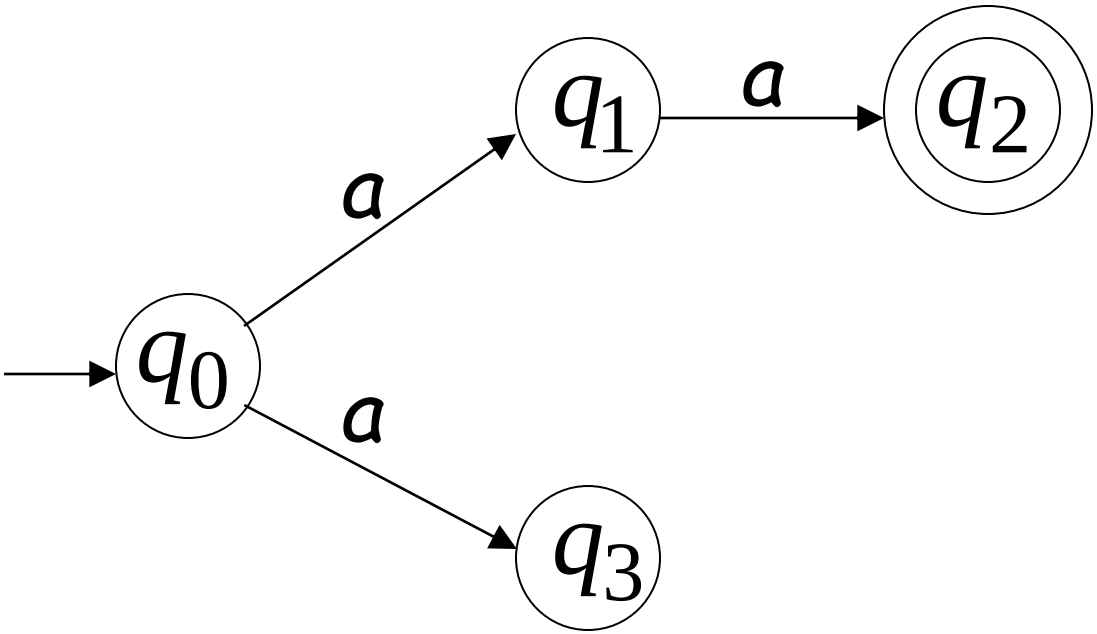
aaa is rejected by the NFA:



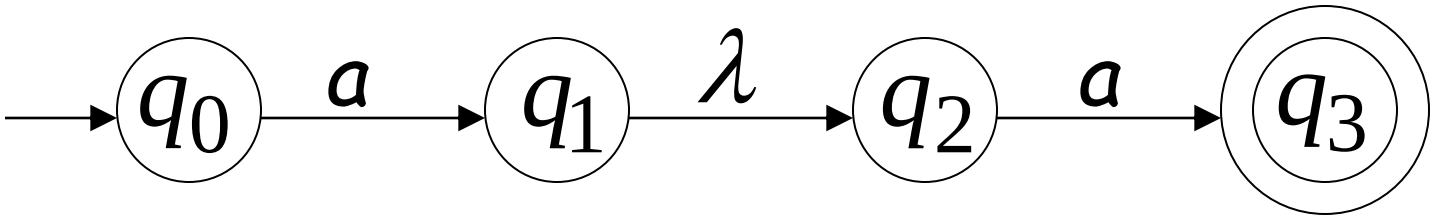


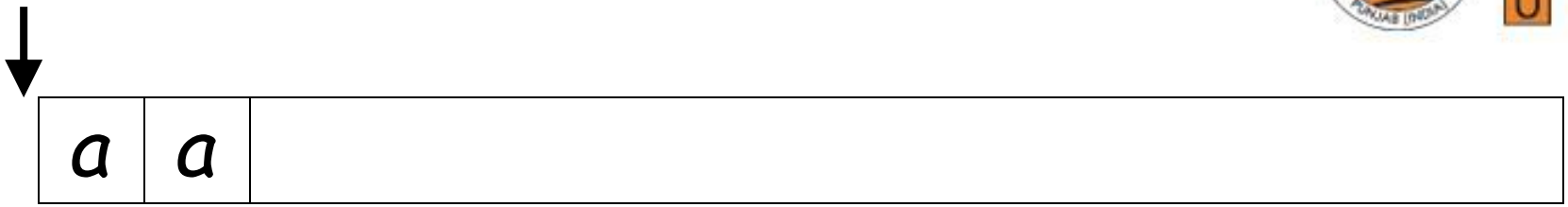
All possible computations lead to rejection

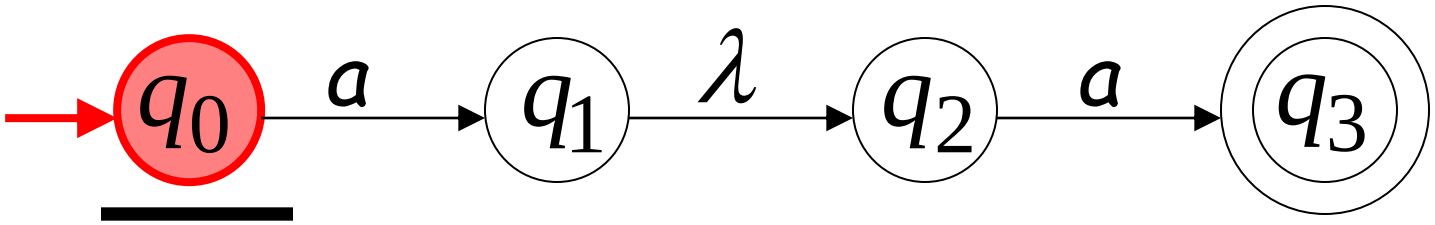
Language accepted: $L = \{aa\}$

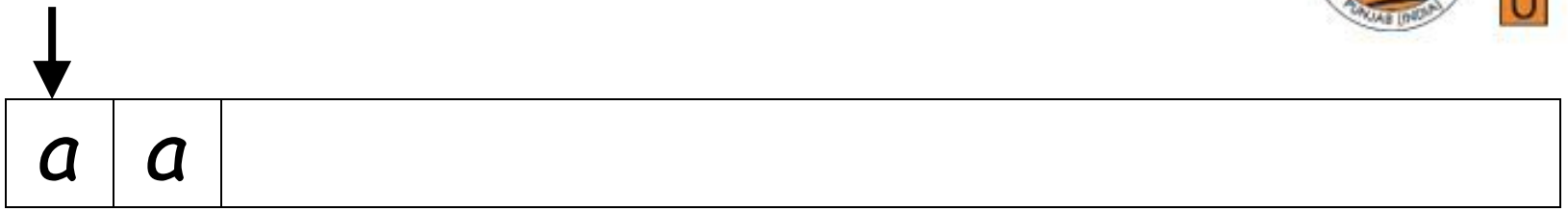


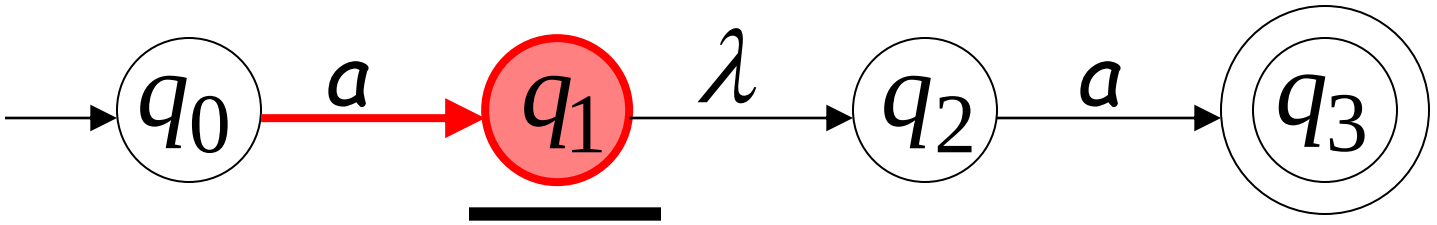
Lambda Transitions



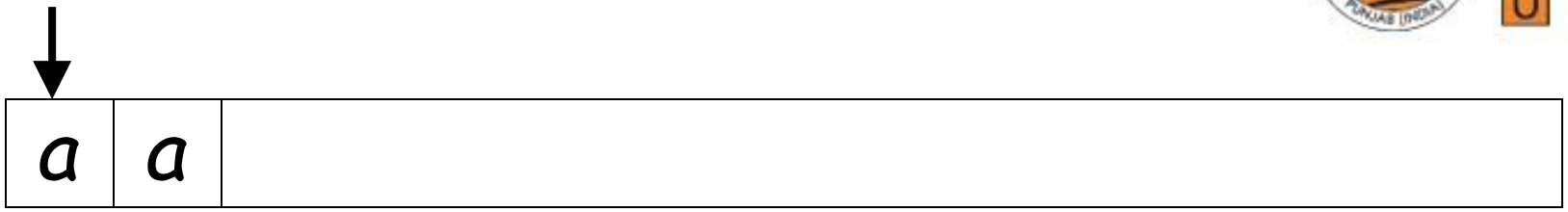

a
a

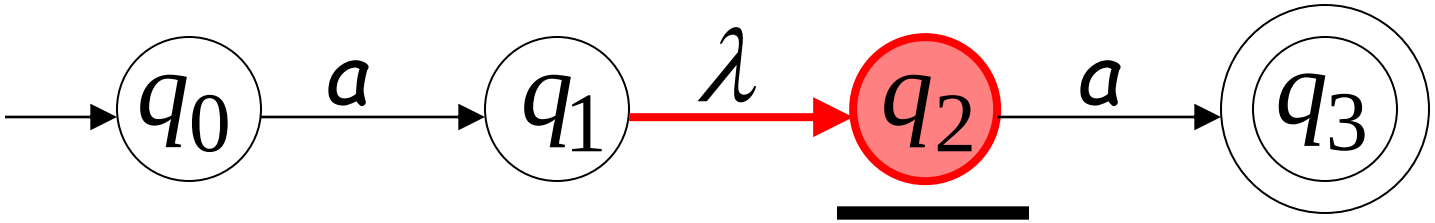

q_0
a
q_1
λ
q_2
a
q_3

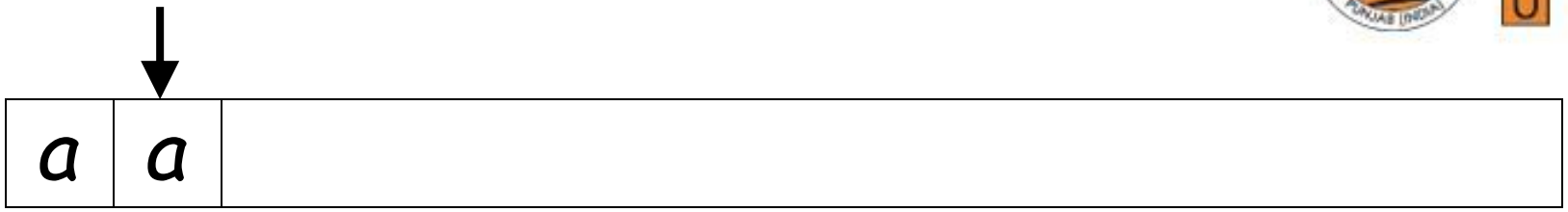

a
a

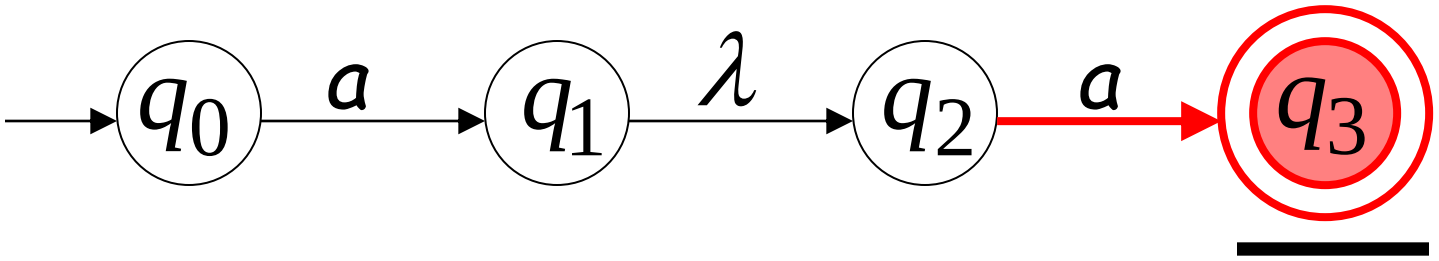

q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

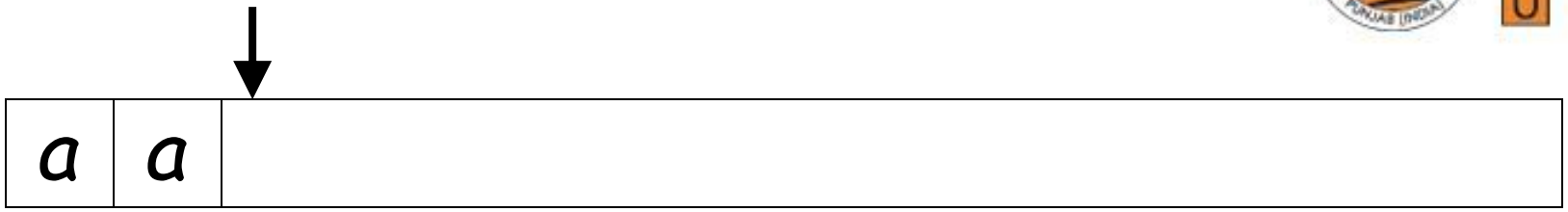




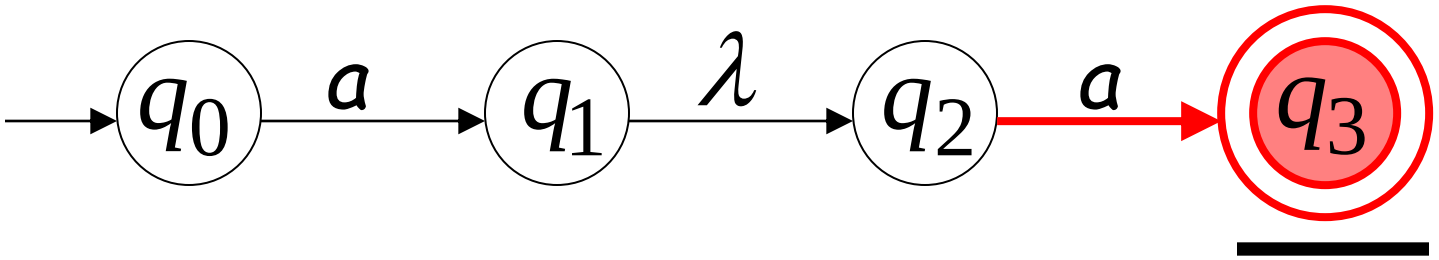

a
a


q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

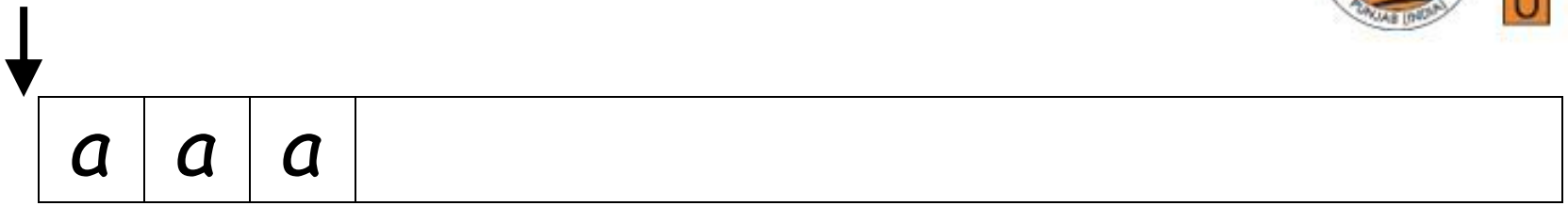


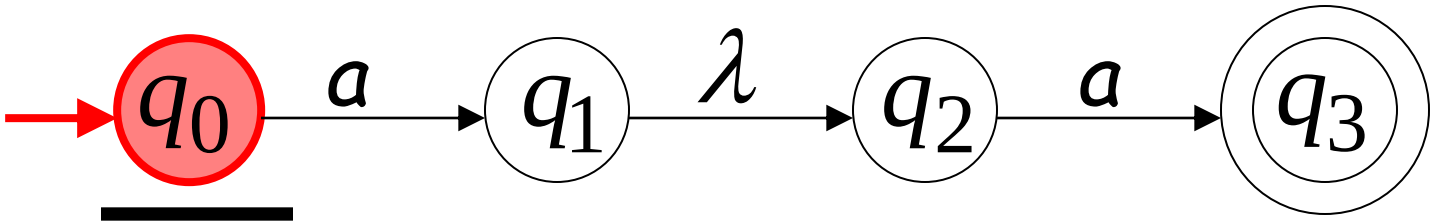
"accept"

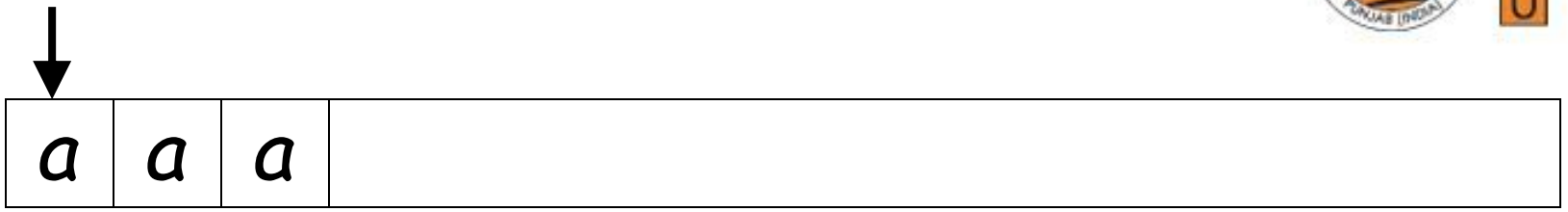


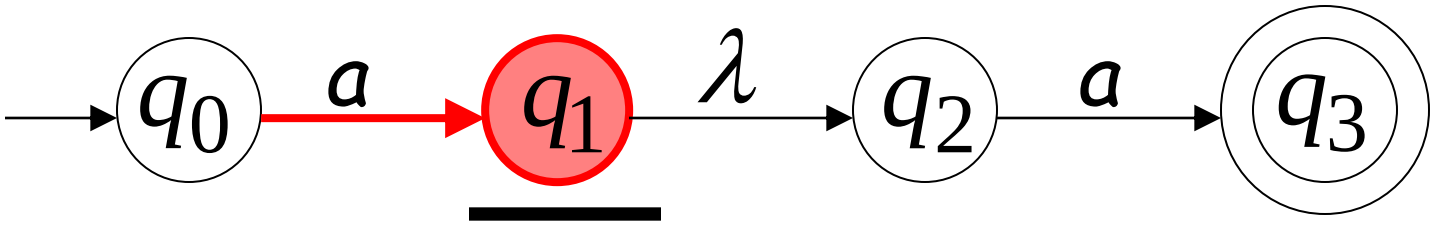
String aa is accepted

Rejection Example

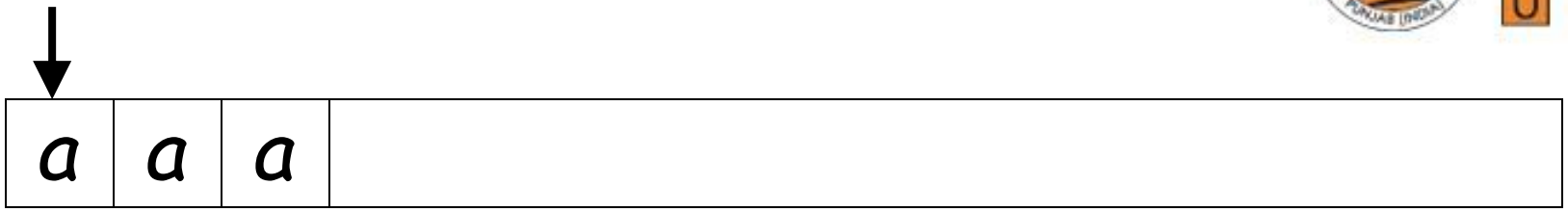


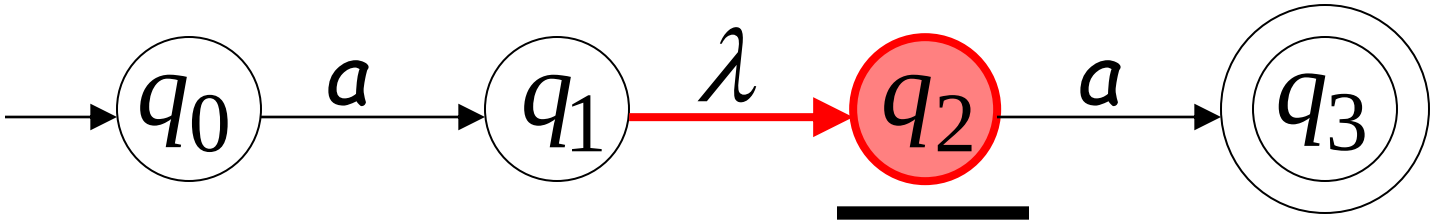


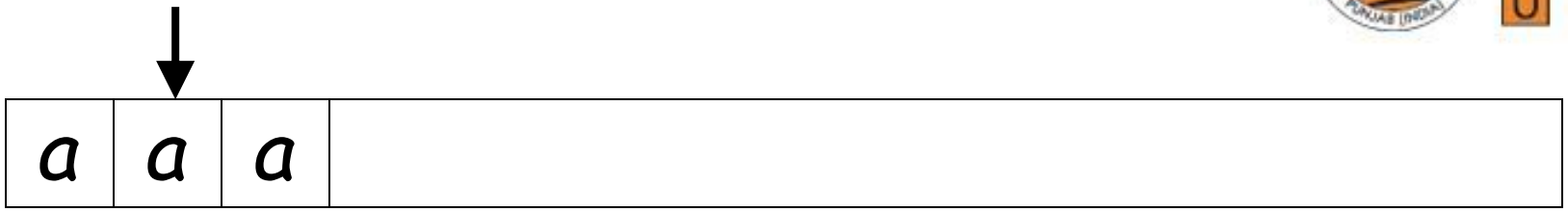

a
a
a

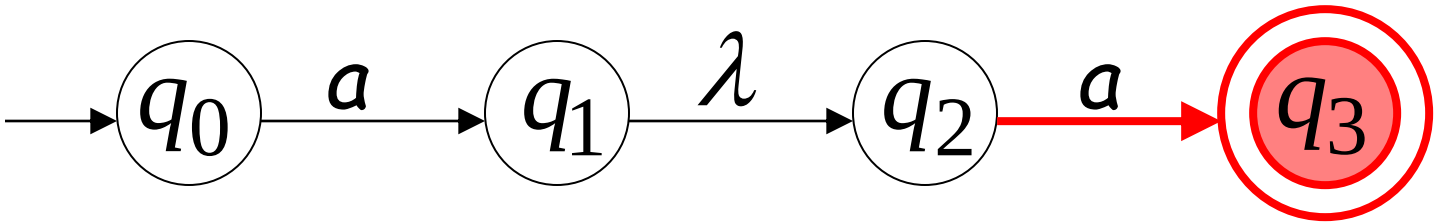

q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)



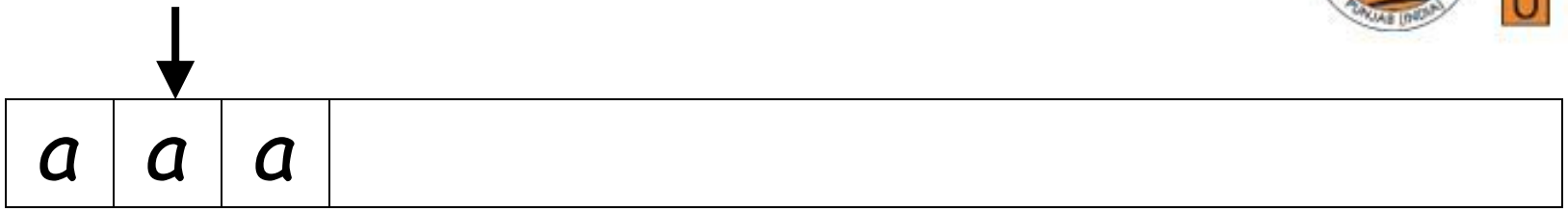




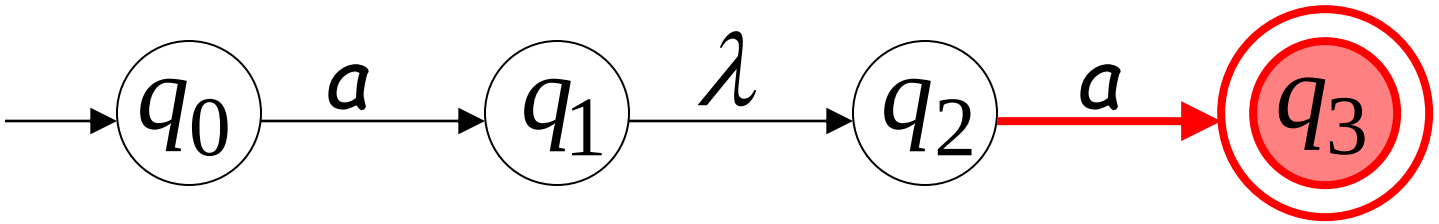


No transition:
the automaton hangs

Input cannot be consumed

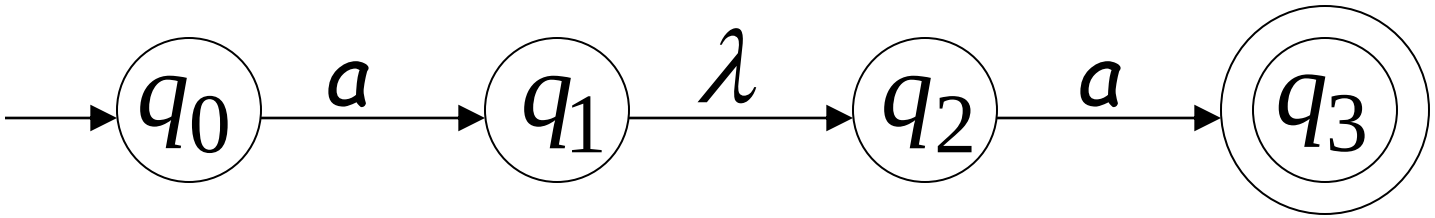


"reject"

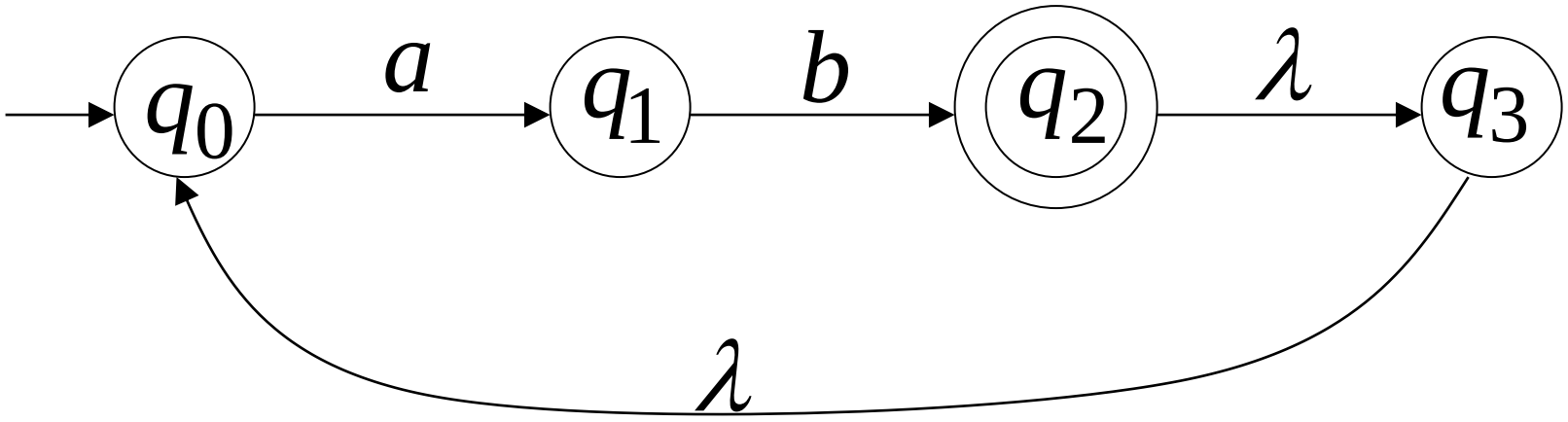


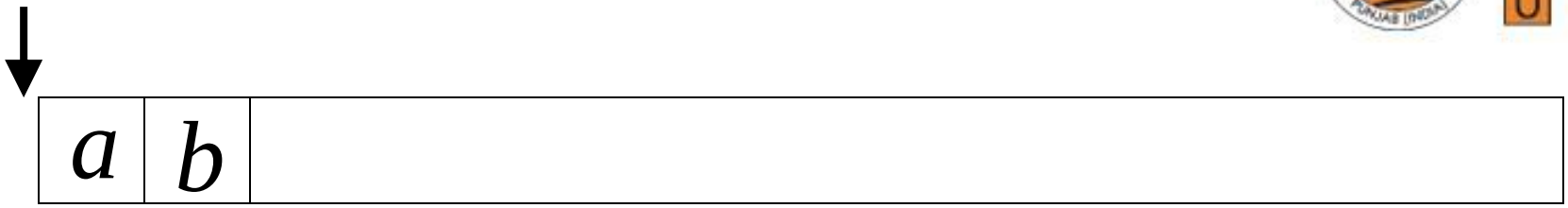
String aaa is rejected

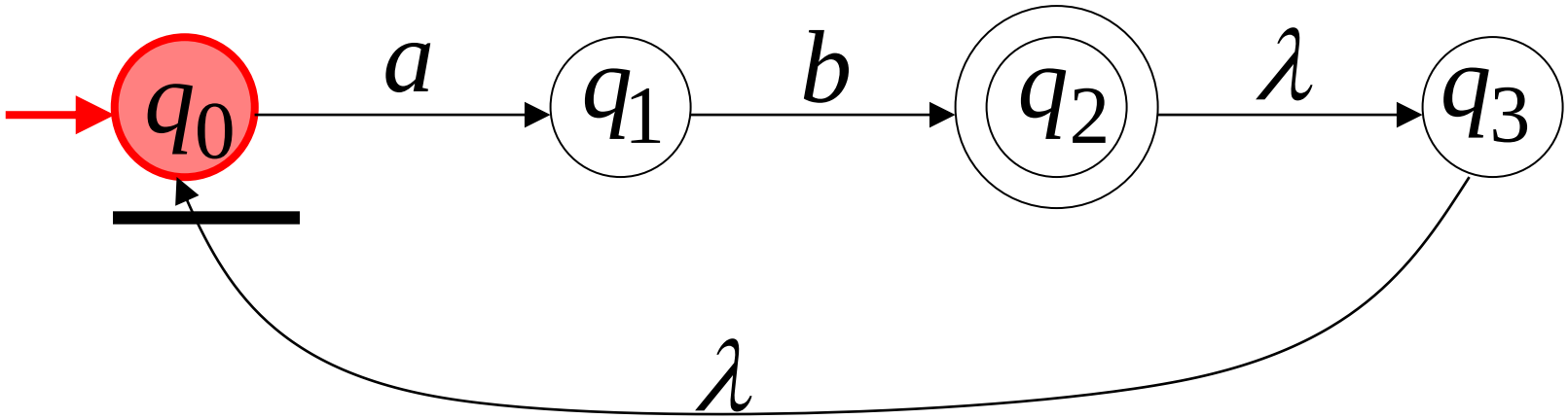
Language accepted: $L = \{aa\}$

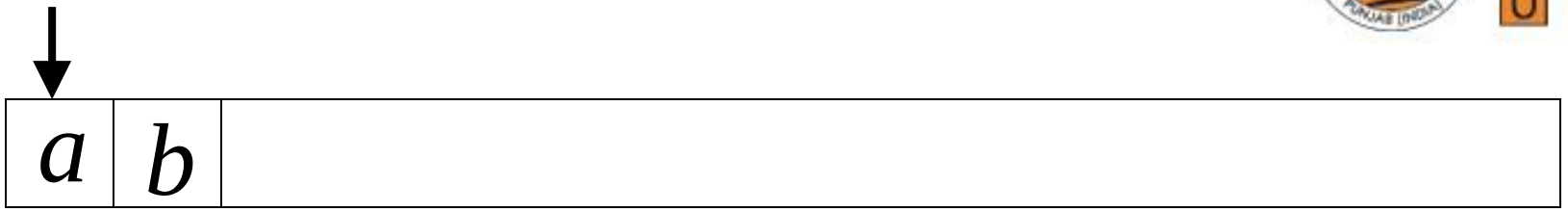


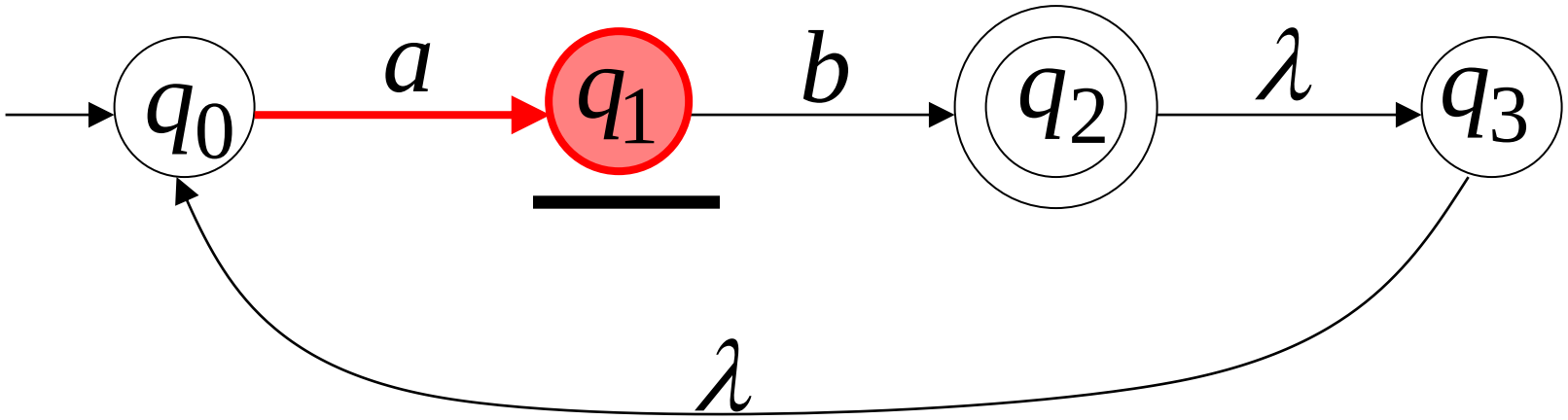
Another NFA Example

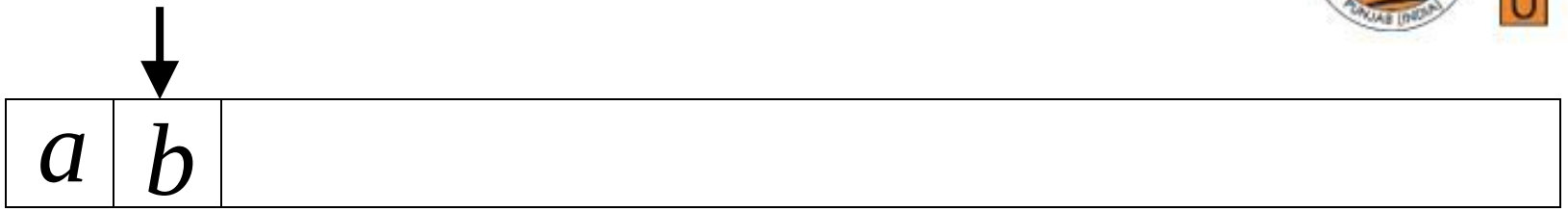


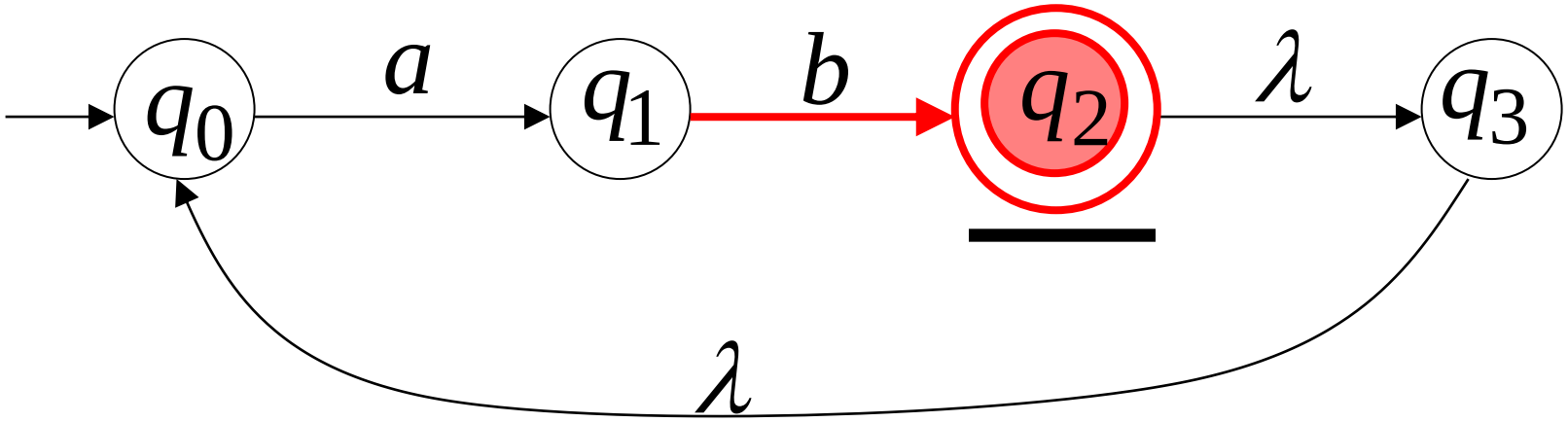

a
b

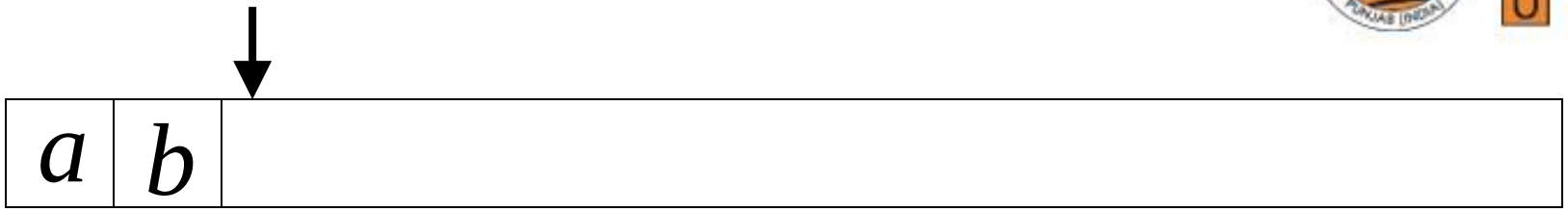

q_0
a
q_1
b
q_2
λ
q_3
λ

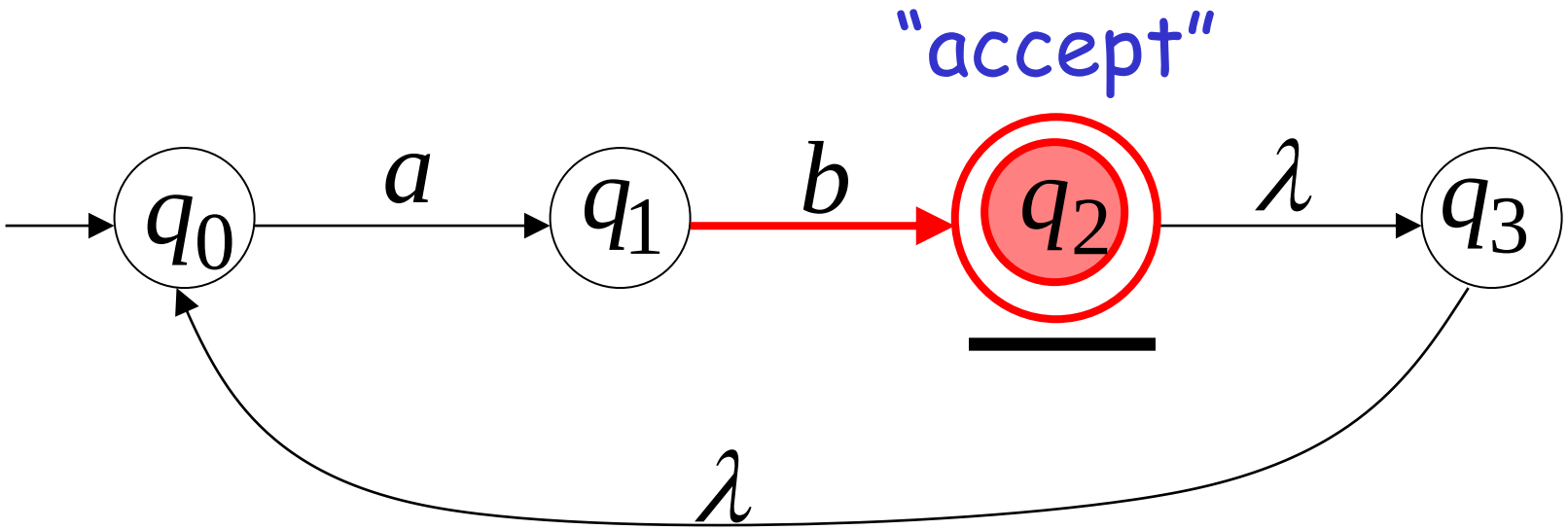

a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

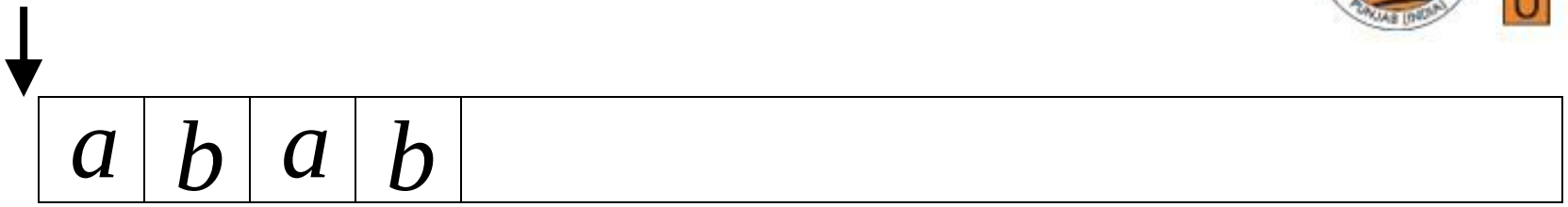

a
b

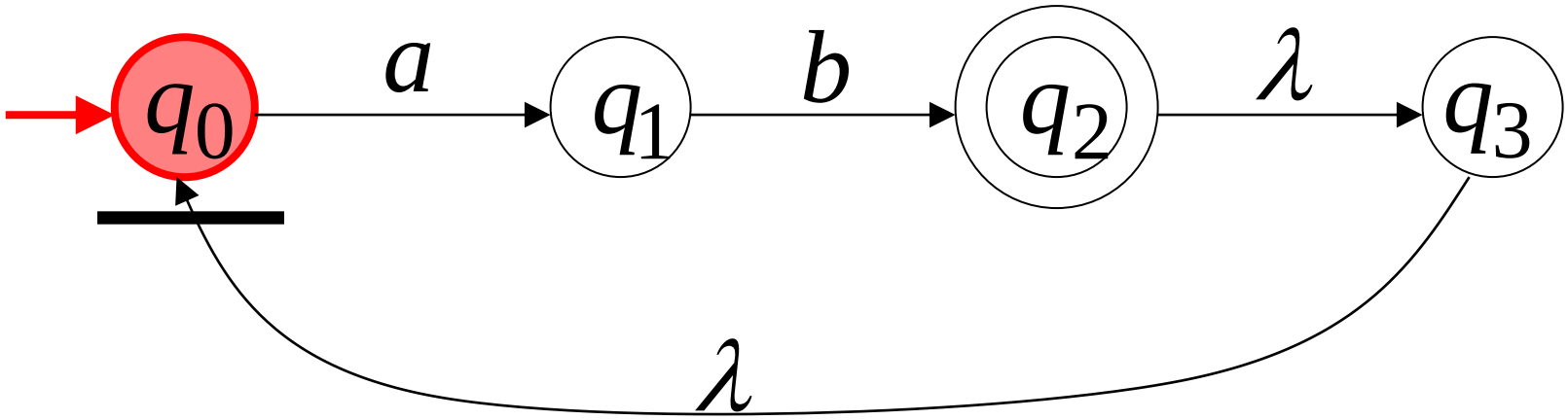

q_0
a
q_1
b
q_2
λ
q_3
λ

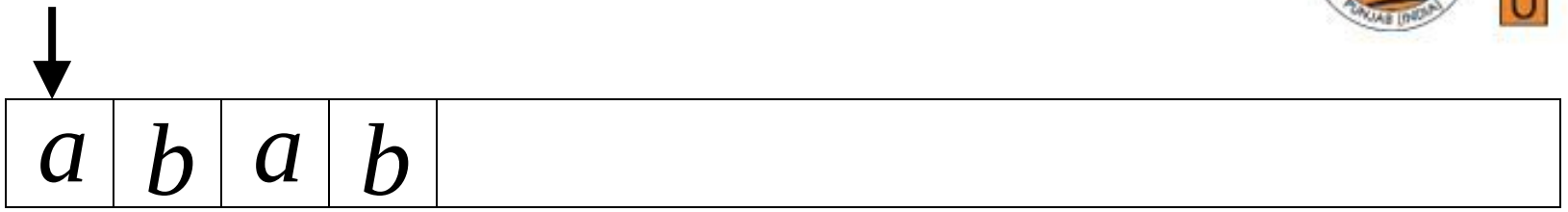

a
b

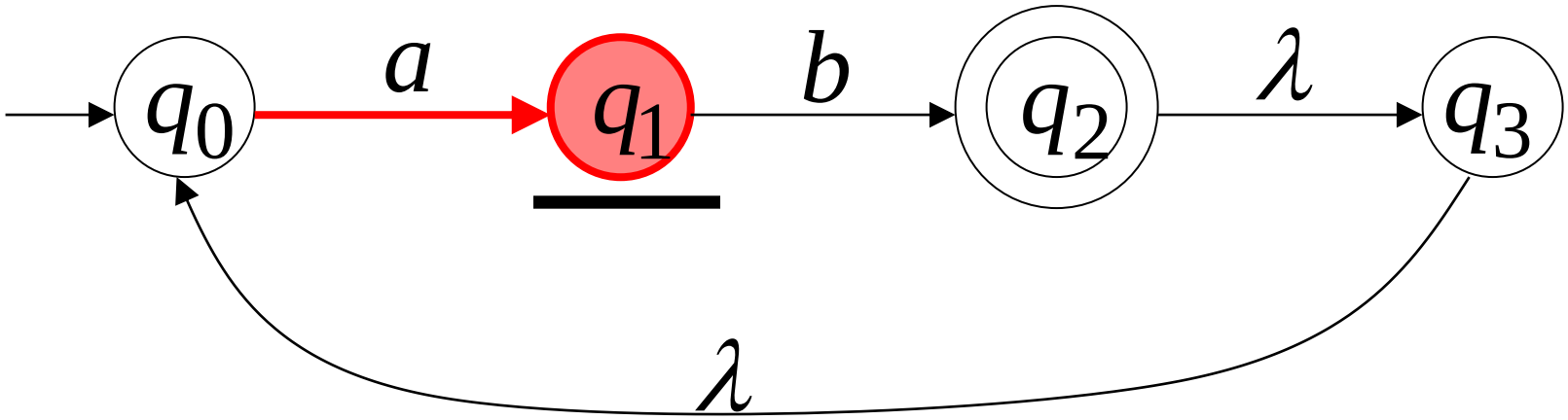

"accept"
q_0
a
q_1
b
q_2
λ
q_3
λ

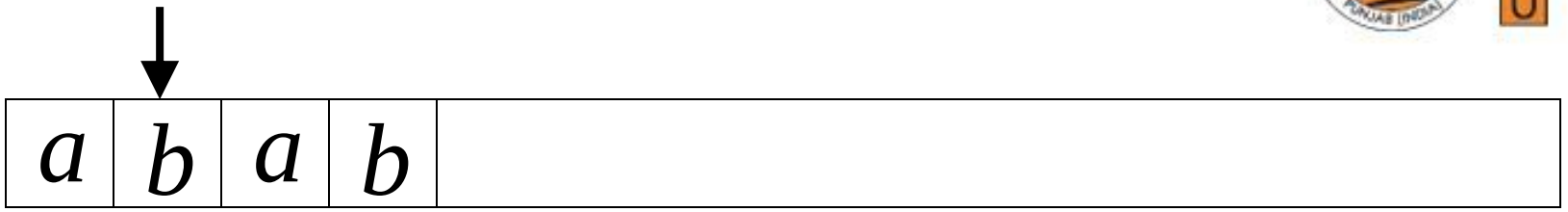
Another String

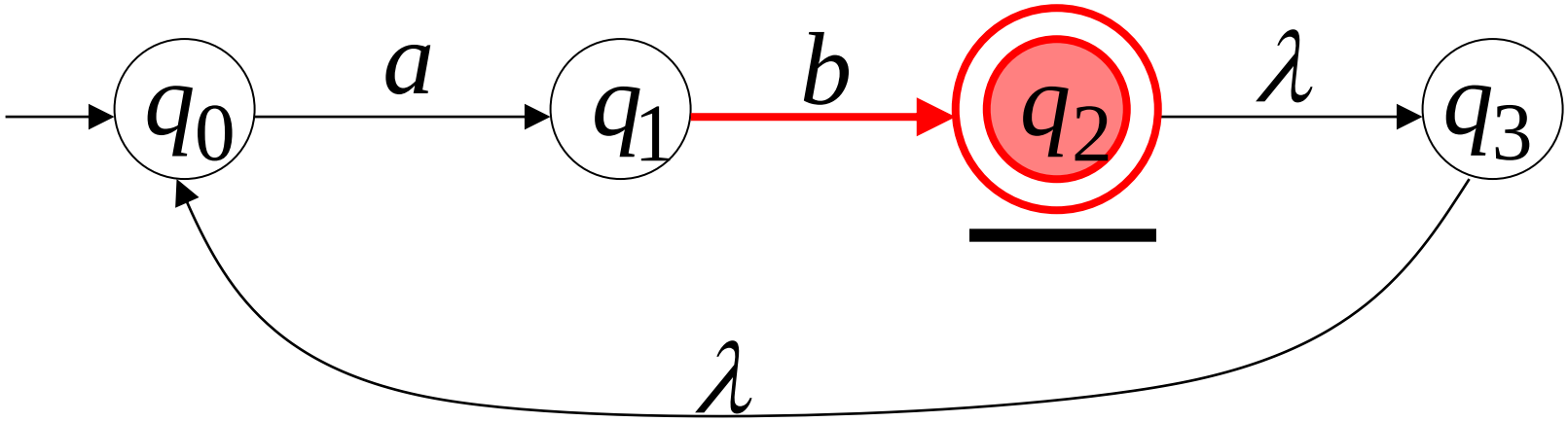


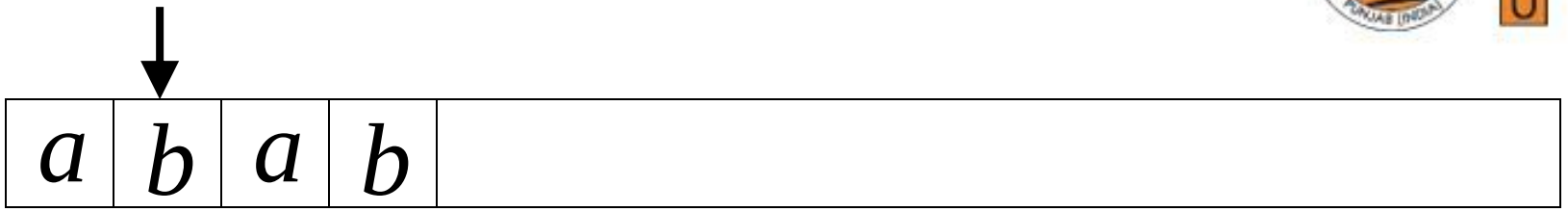


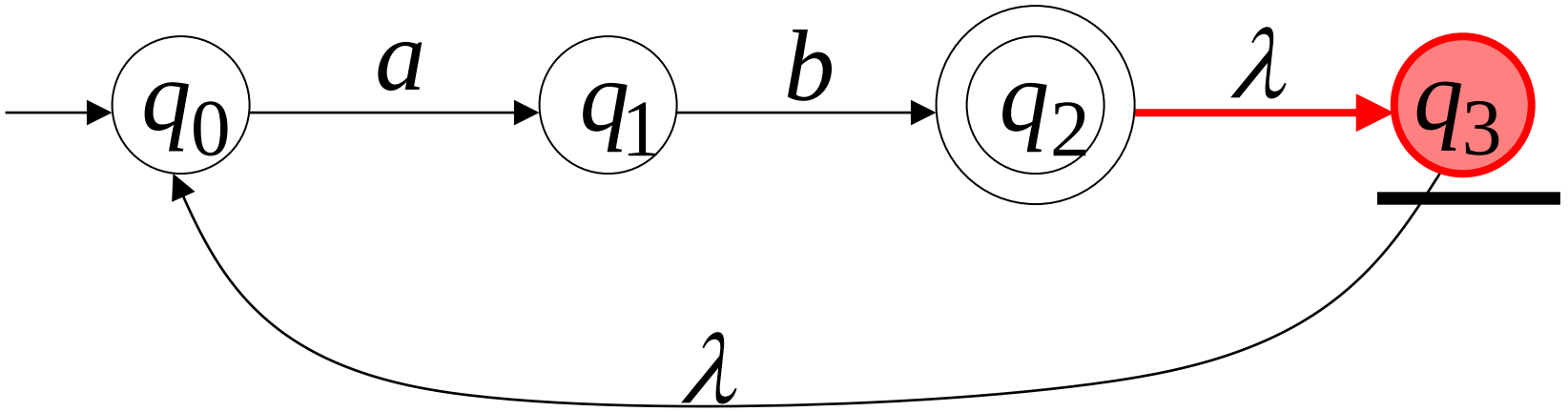

a
b
a
b

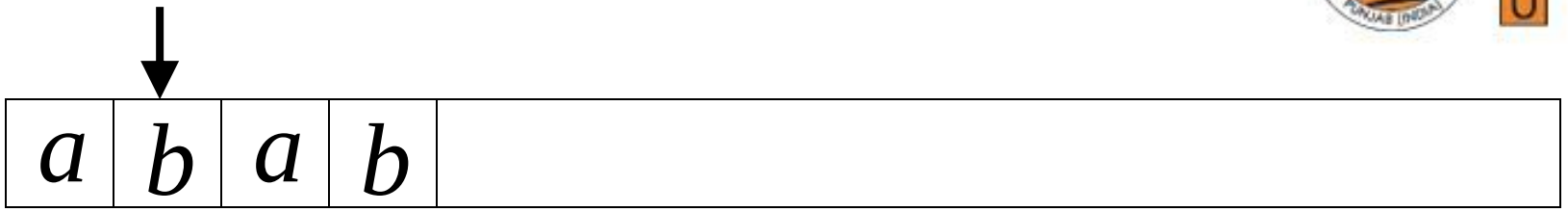

q_0
a
q_1
b
q_2
λ
q_3
λ

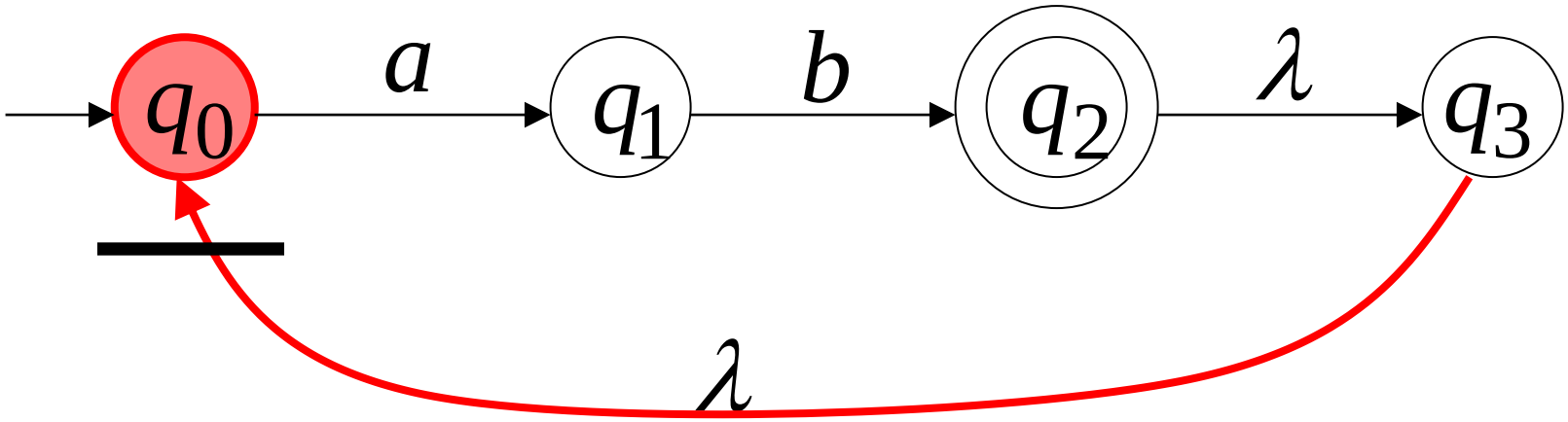

a
b
a
b

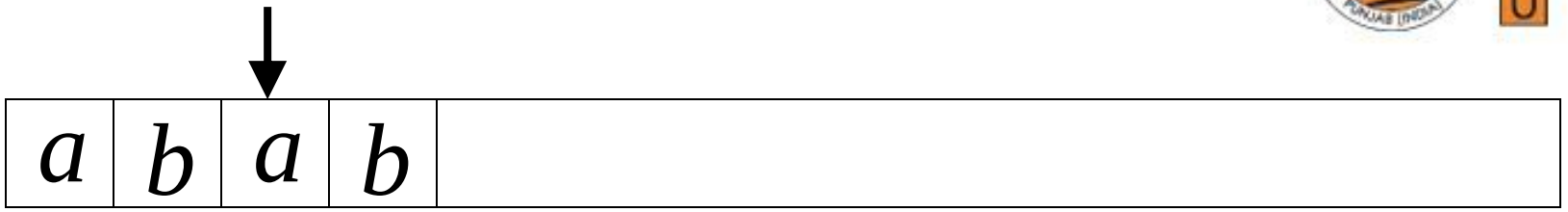

q_0
a
q_1
b
q_2
λ
q_3
λ

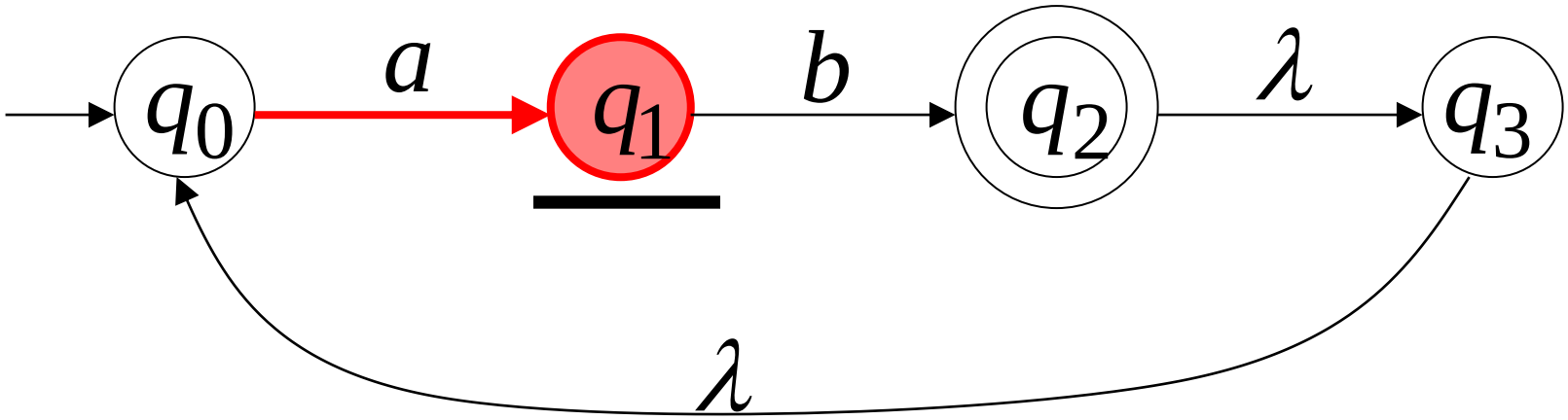

a
b
a
b

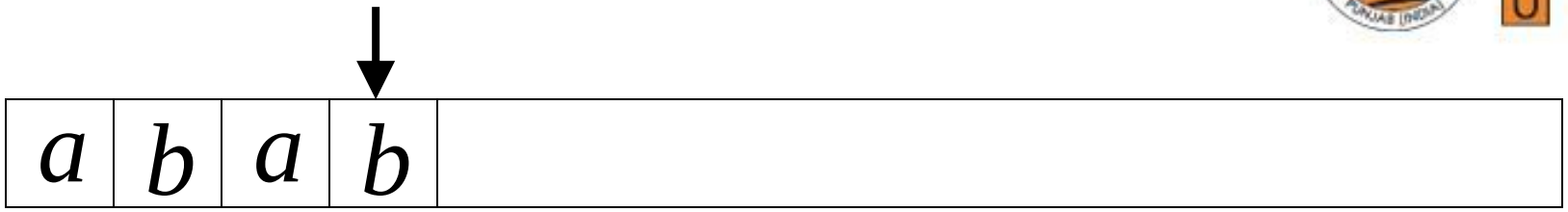

q_0
a
q_1
b
q_2
λ
q_3
λ

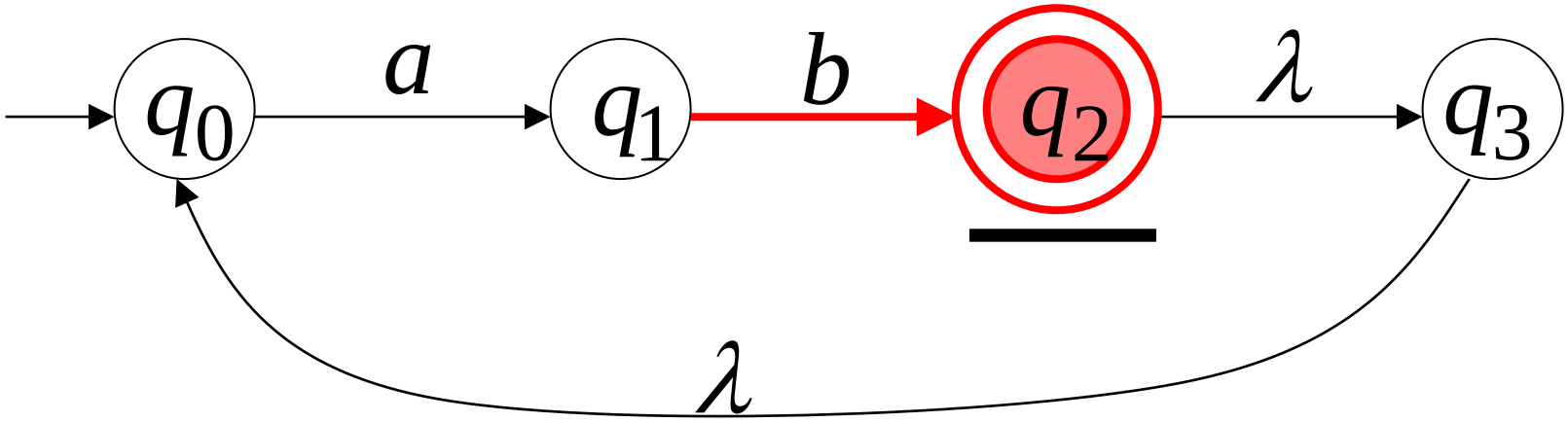

a
b
a
b

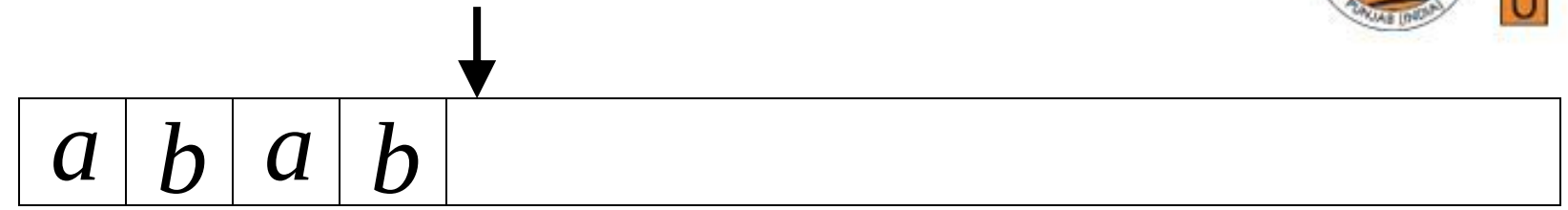

q_0
a
q_1
b
q_2
λ
q_3
λ

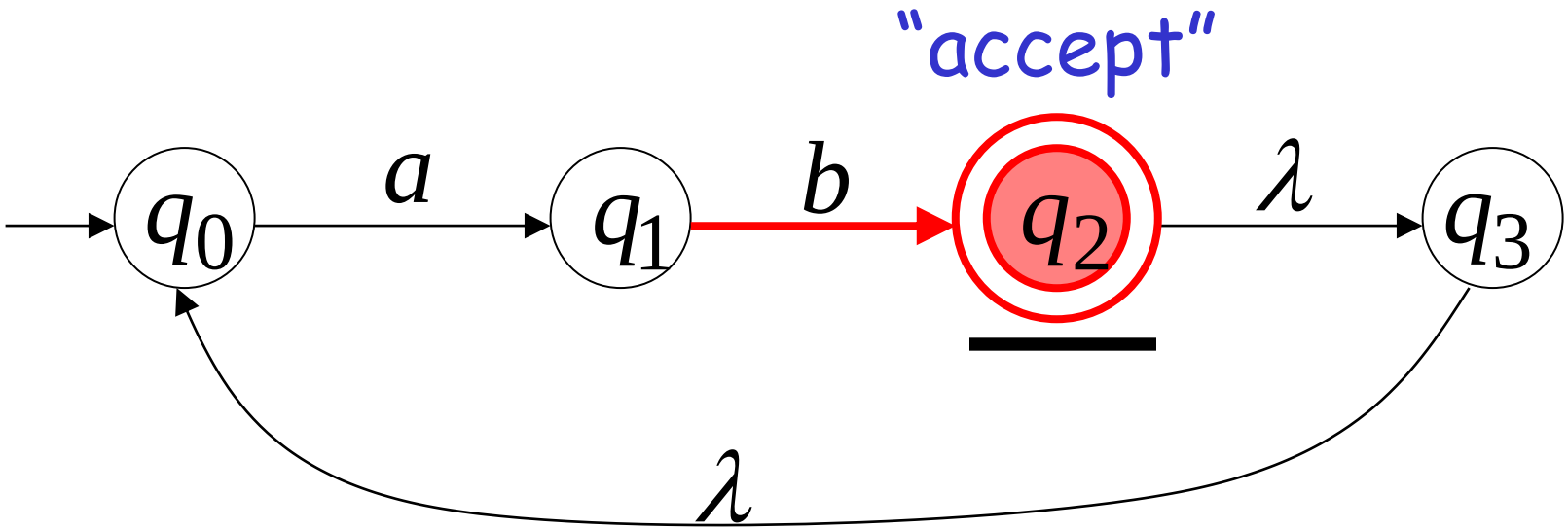

a
b
a
b


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

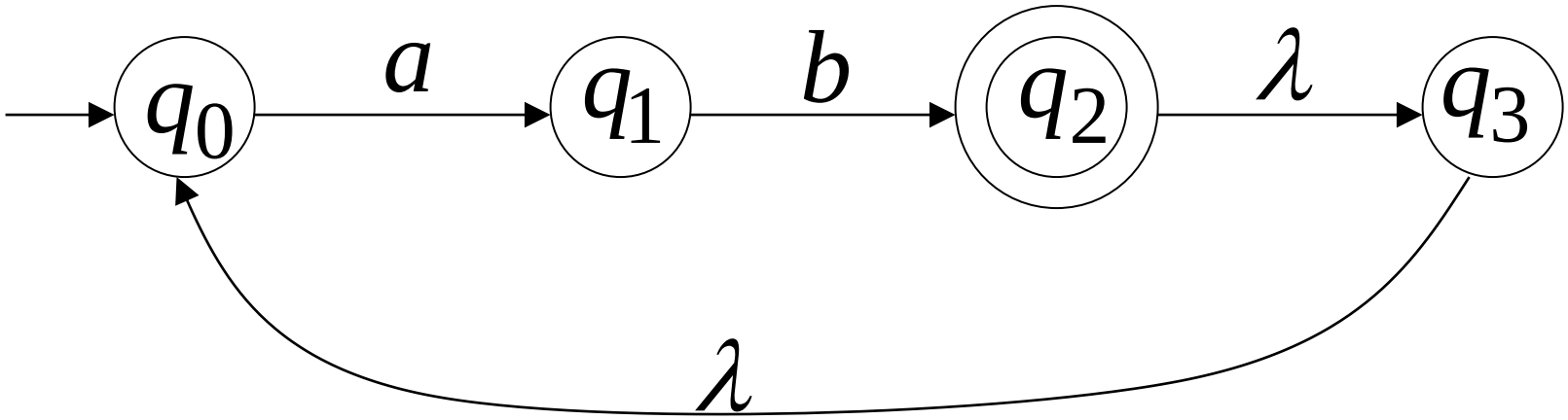

a
b
a
b


"accept"
q_0
q_1
q_2
q_3
a
b
λ
λ

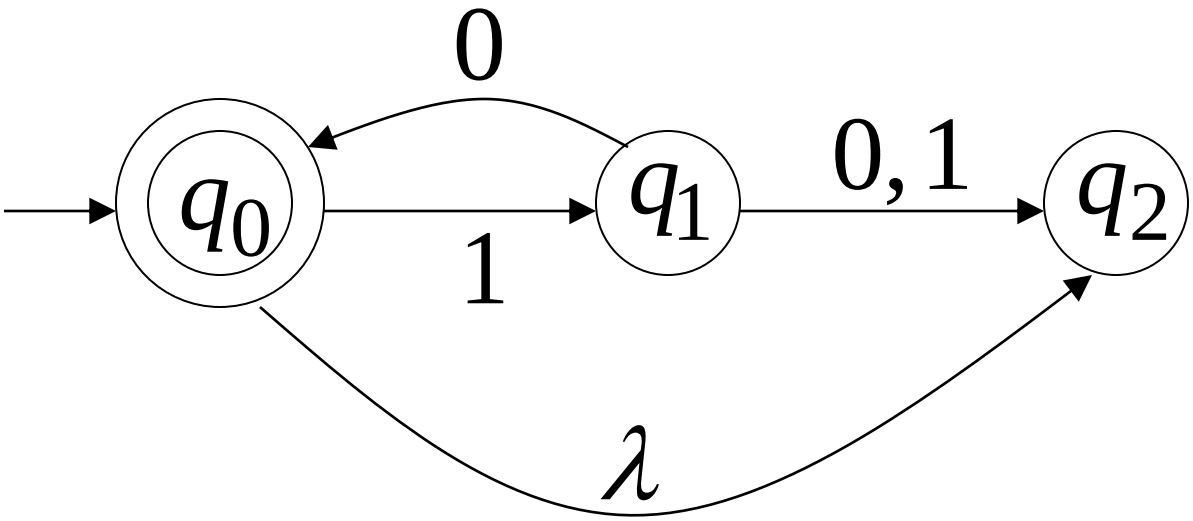
Language accepted

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

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



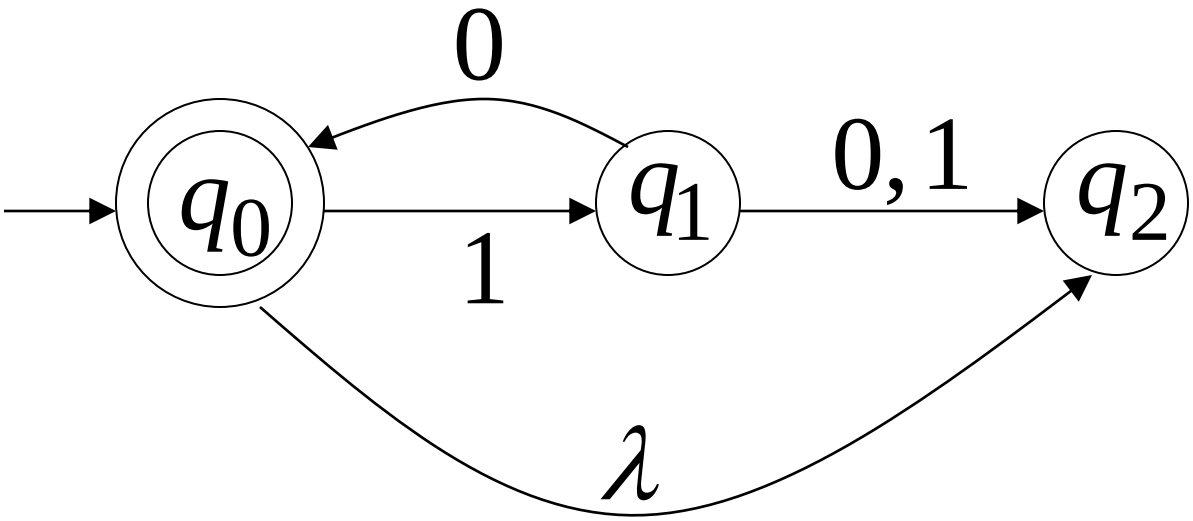
Another NFA Example



Language accepted

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

$$= \{10\}*$$



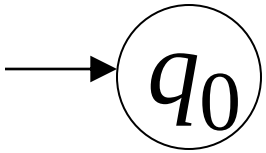
(redundant state)

Remarks:

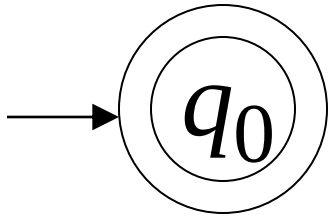
- The λ symbol never appears on the input tape

- Simple automata:

M_1



M_2

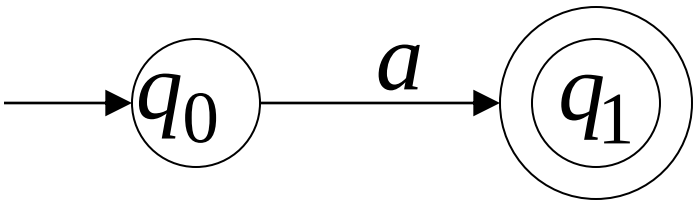


$L(M_1) = \{\}$

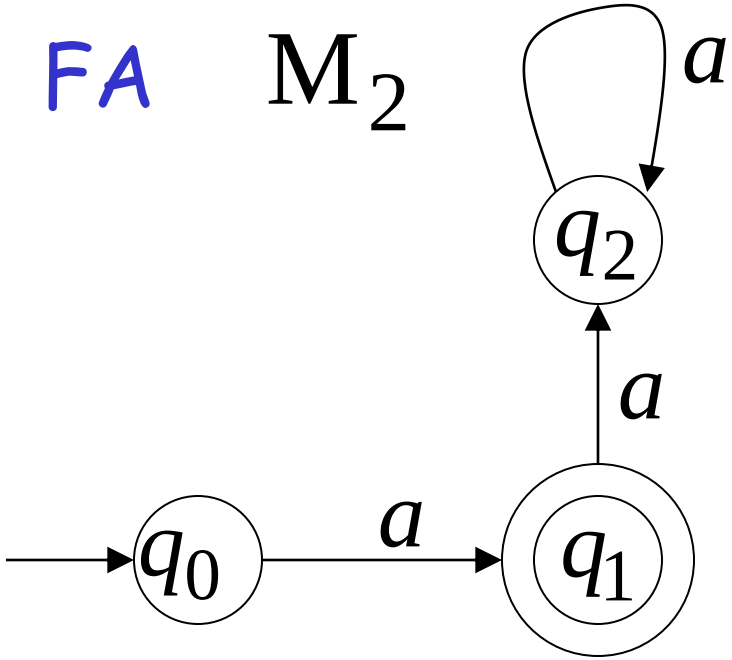
$L(M_2) = \{\lambda\}$

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

NFA M_1



FA 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. $\{q_0, q_1, q_2\}$

Σ: Input aplhabet, i.e. $\{a, b\}$

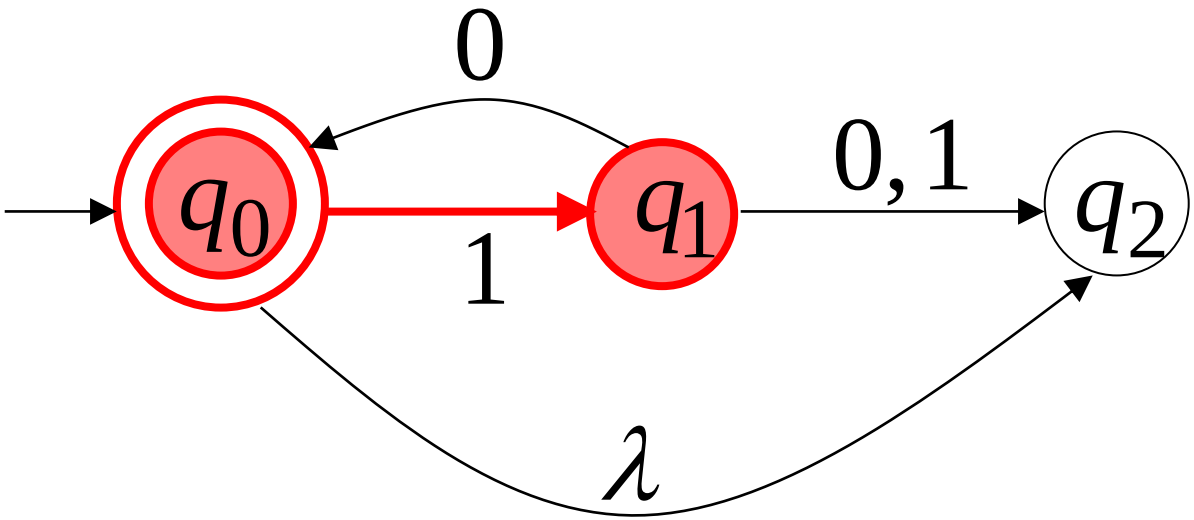
δ: Transition function

q_0: Initial state

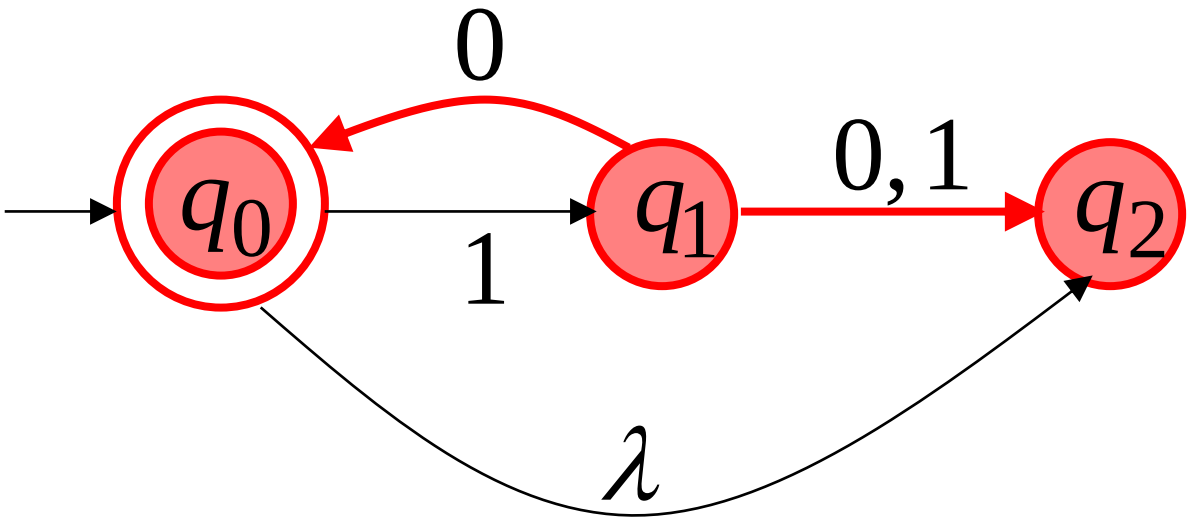
F: Accepting states

Transition Function δ

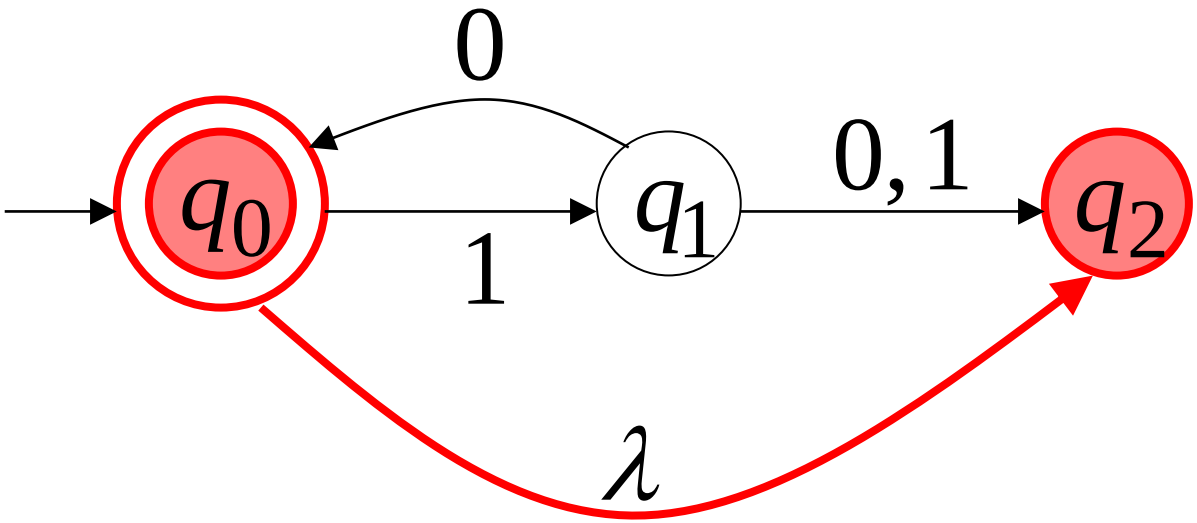
$$\delta(q_0, 1) = \{q_1\}$$



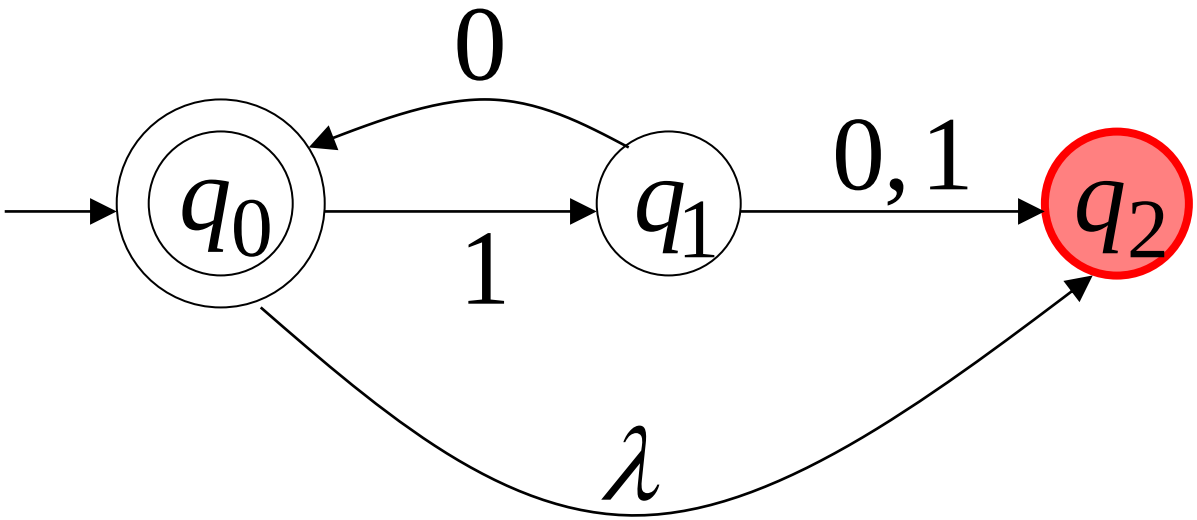
$$\delta(q_1, 0) = \{q_0, q_2\}$$



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

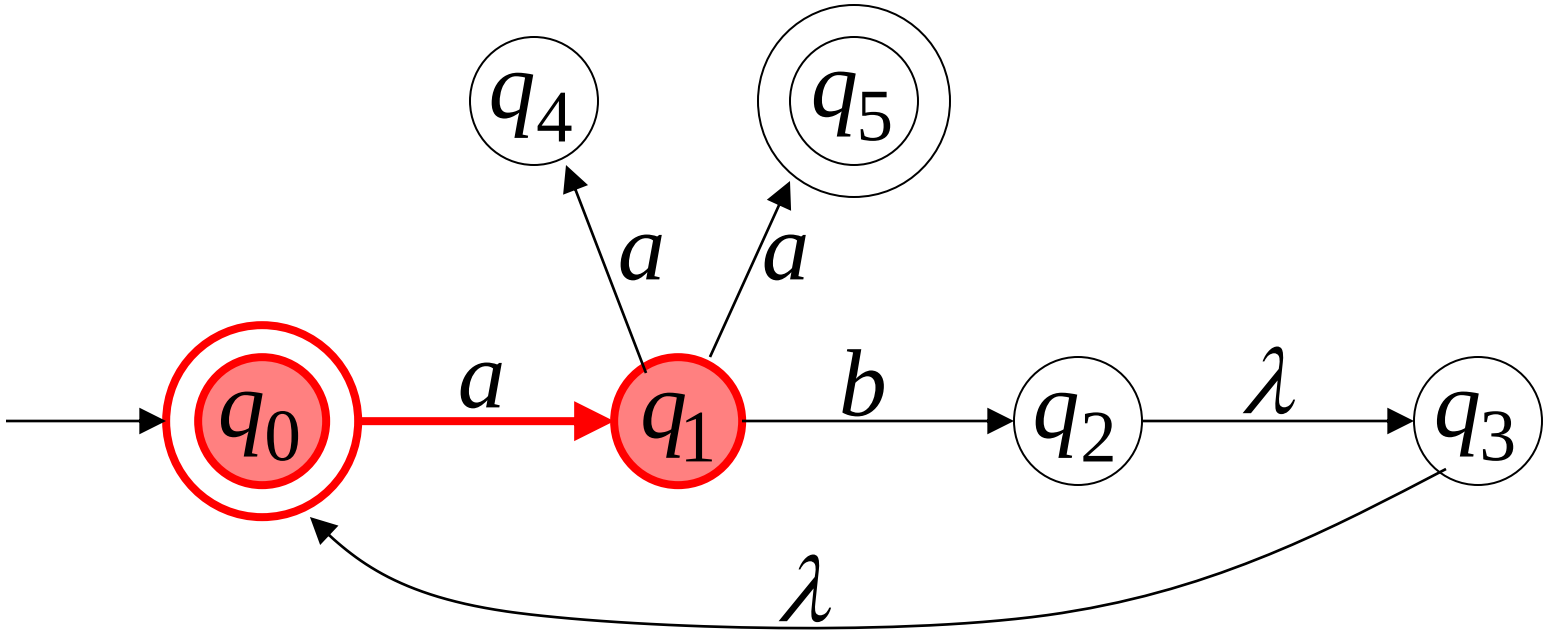


$$\delta(q_2, 1) = \varnothing$$

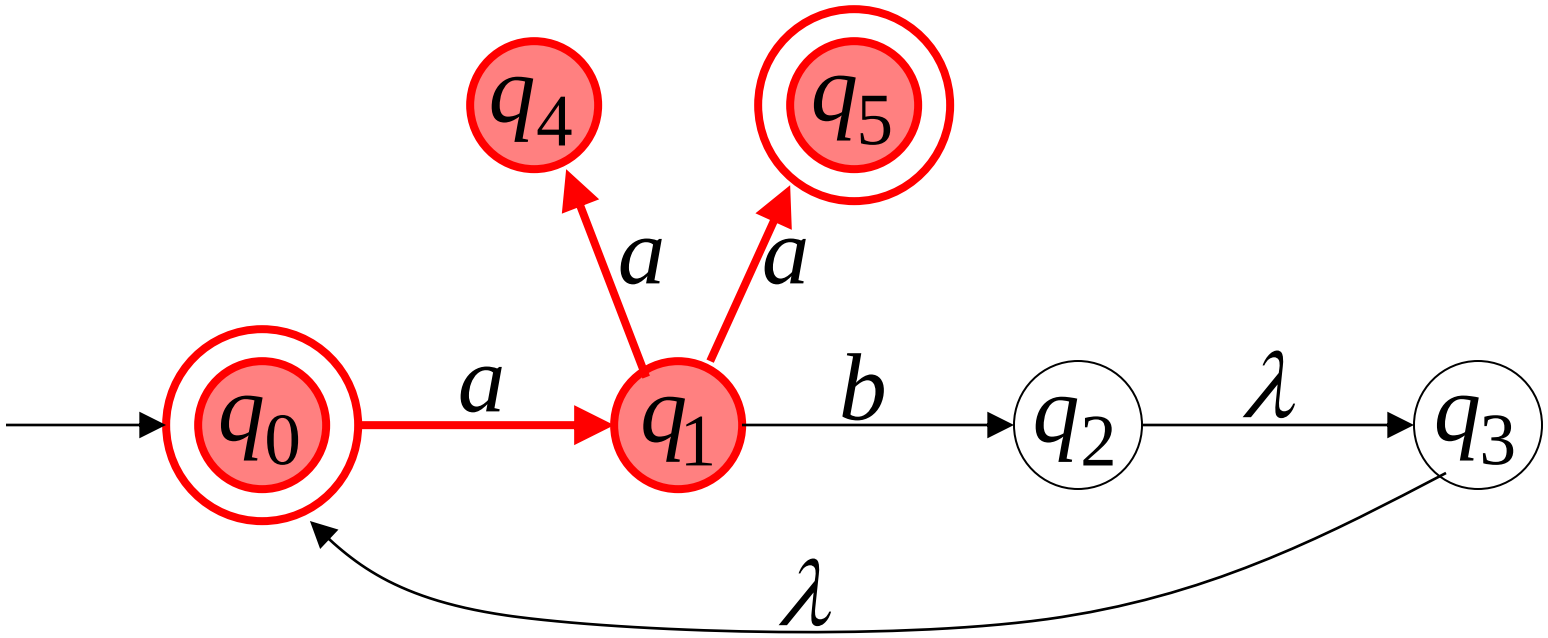


Extended Transition Function δ^*

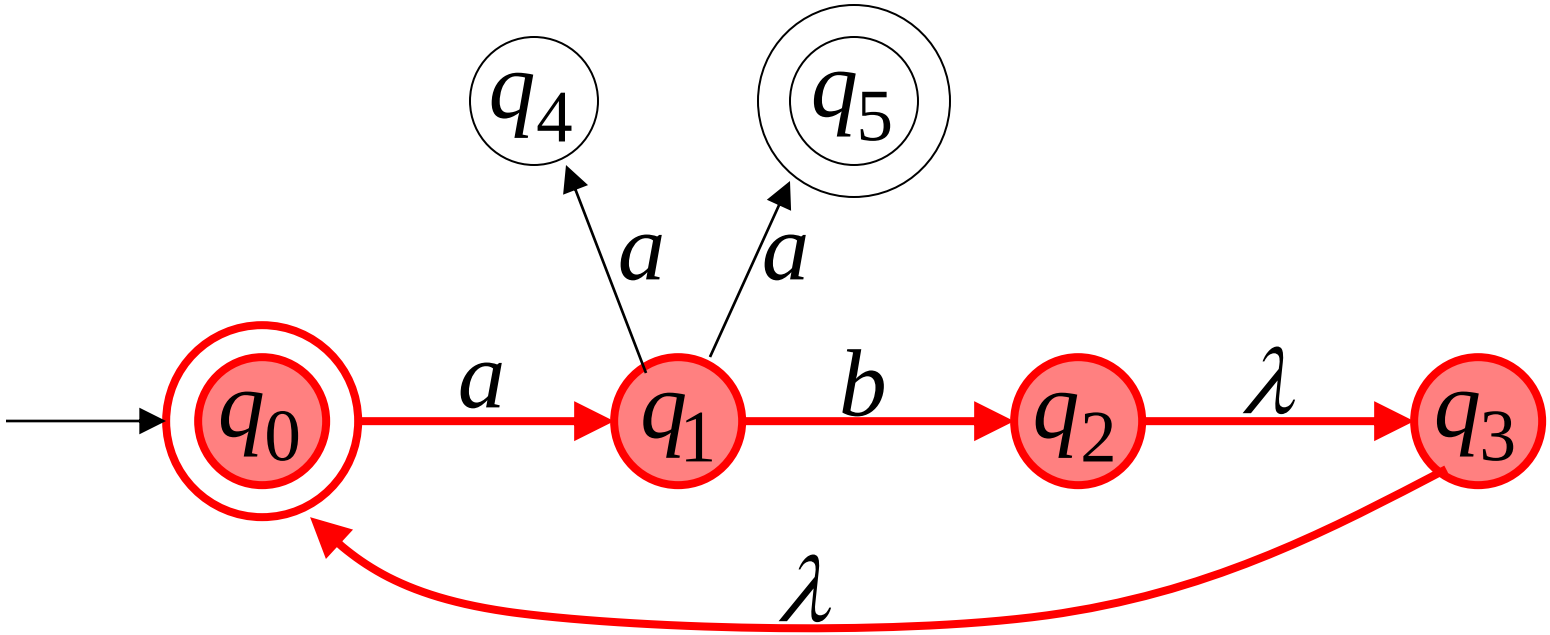
$$\delta^*(q_0, a) = \{q_1\}$$



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

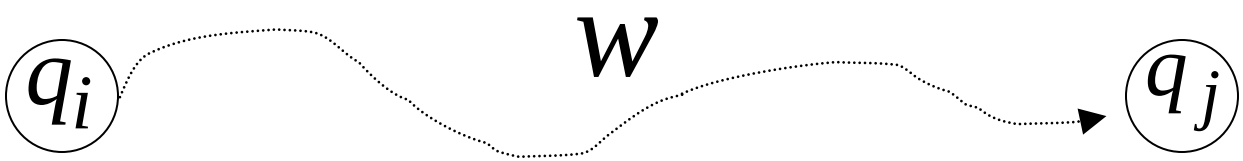


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

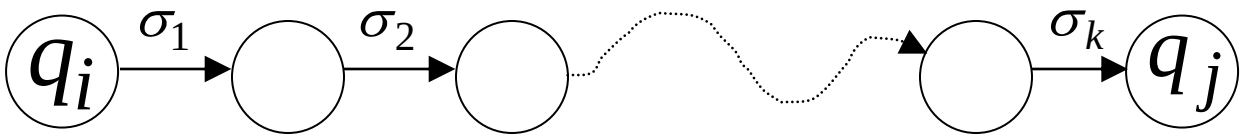


Formally

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

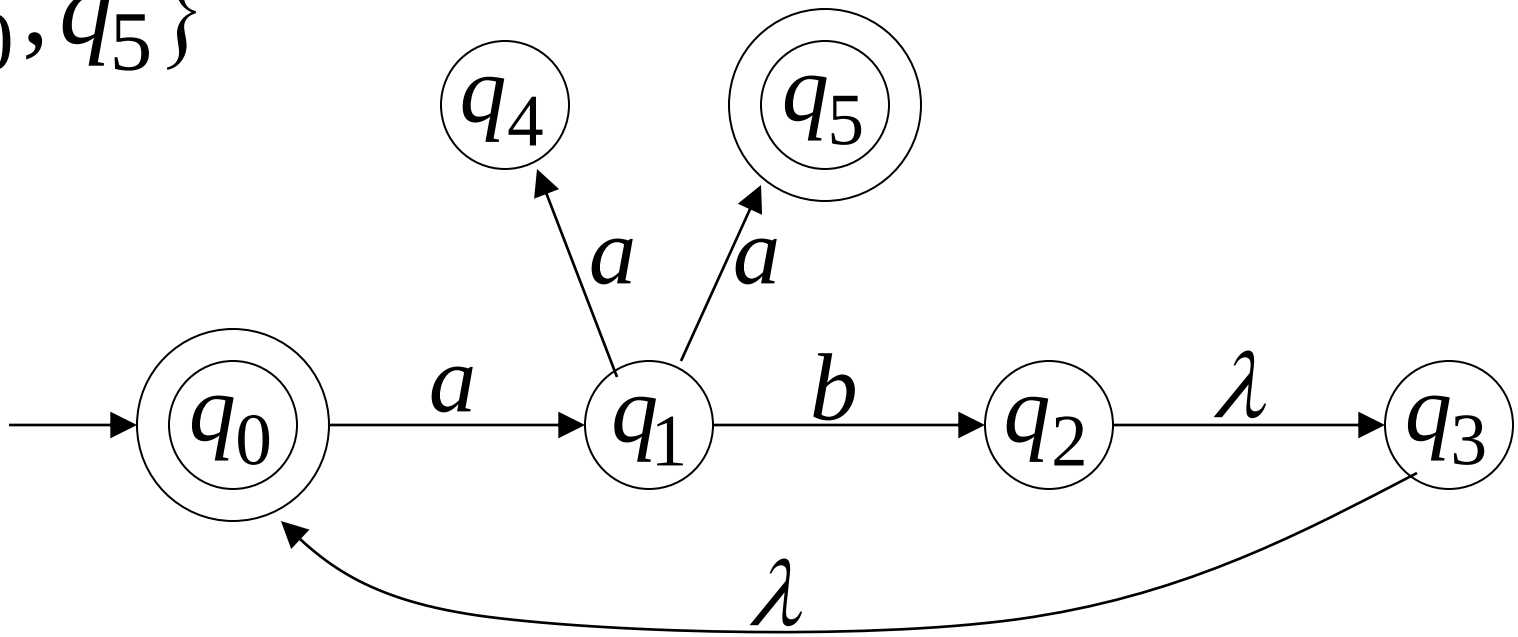


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



The Language of an NFA

$F = \{q_0, q_5\}$

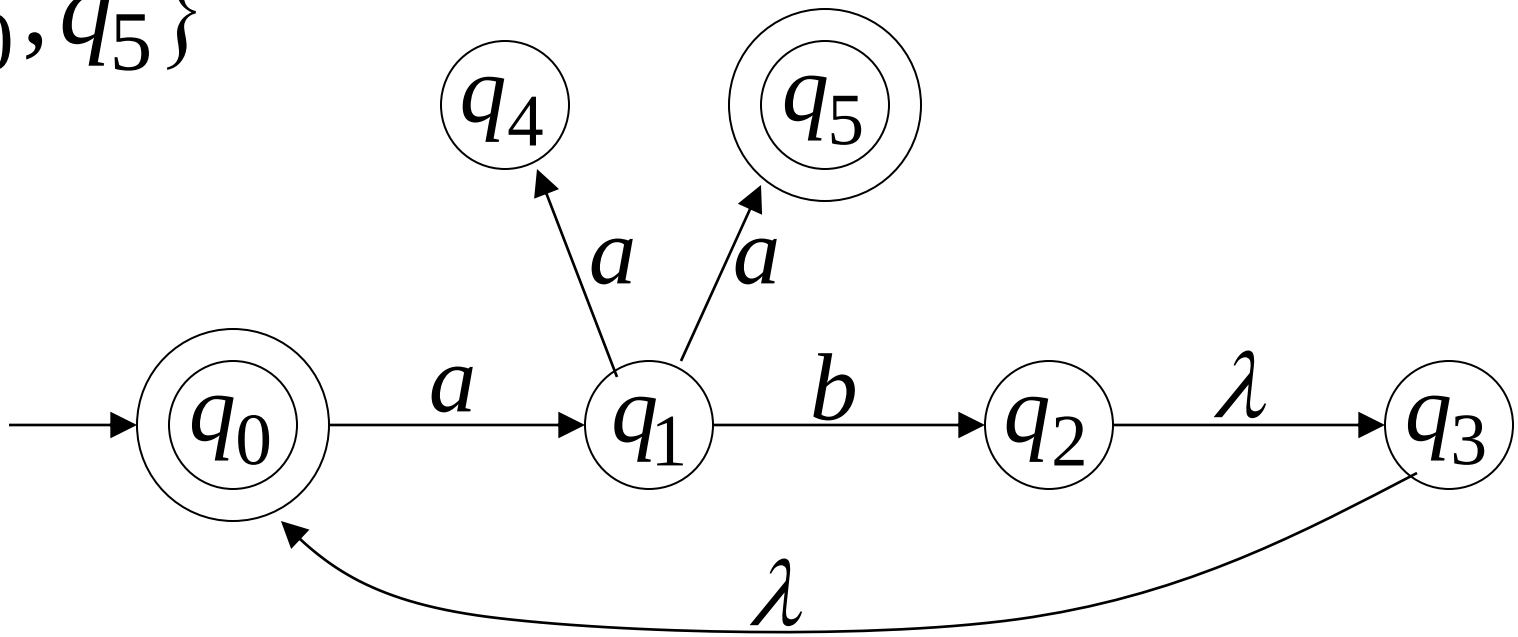


$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\}$

$\searrow \in F$

$aa \in L(M)$

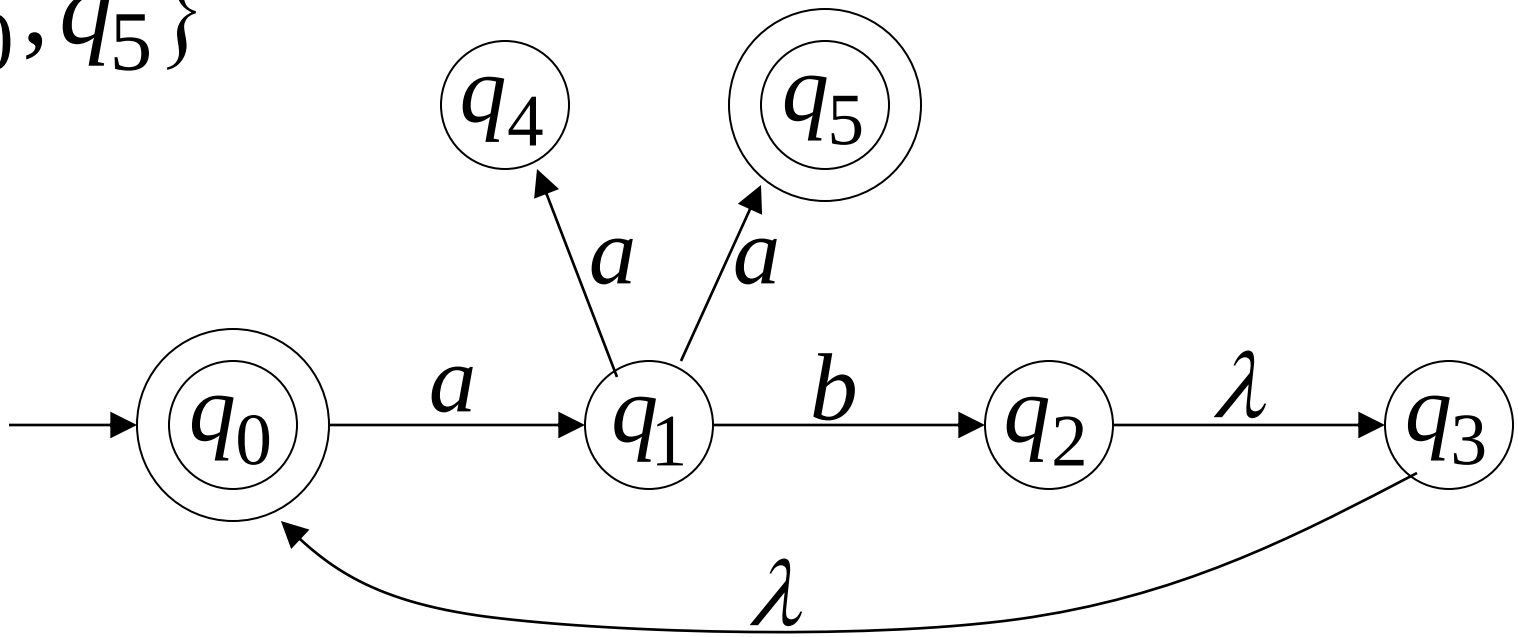
$F = \{q_0, q_5\}$



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

$\searrow \in F$

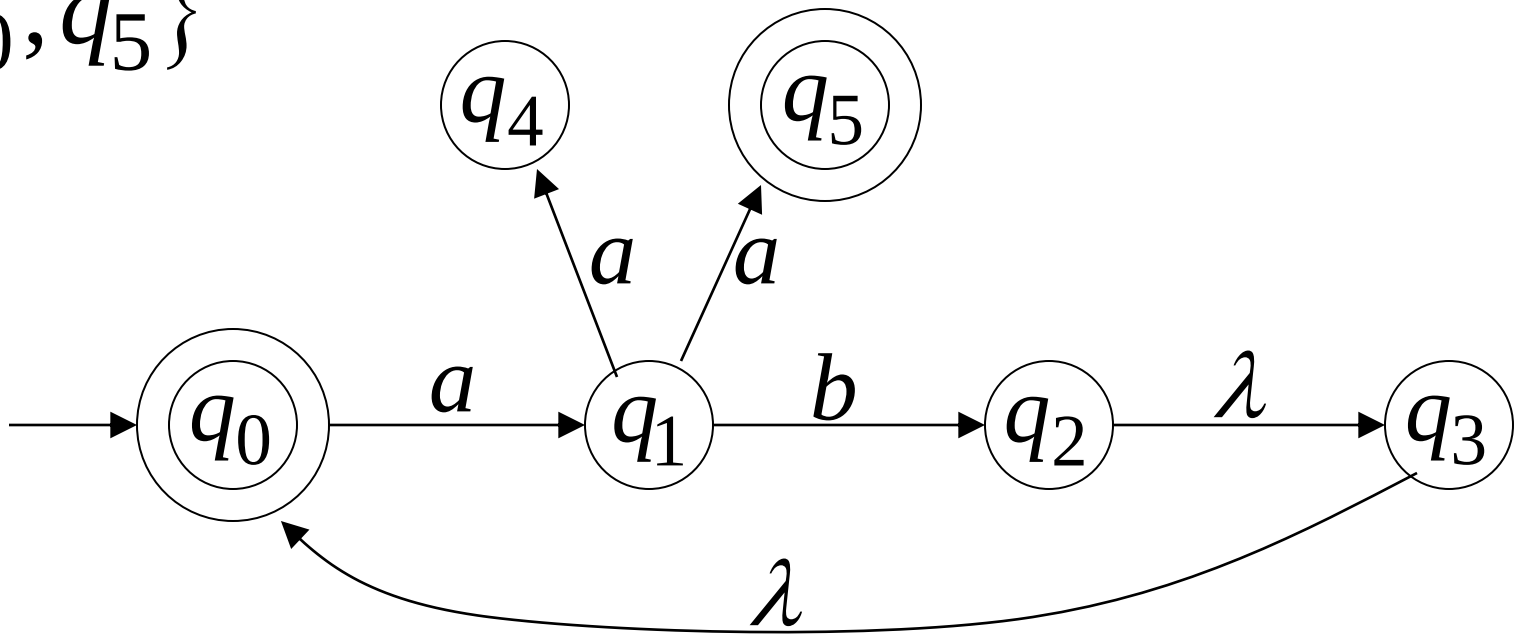
$F = \{q_0, q_5\}$



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

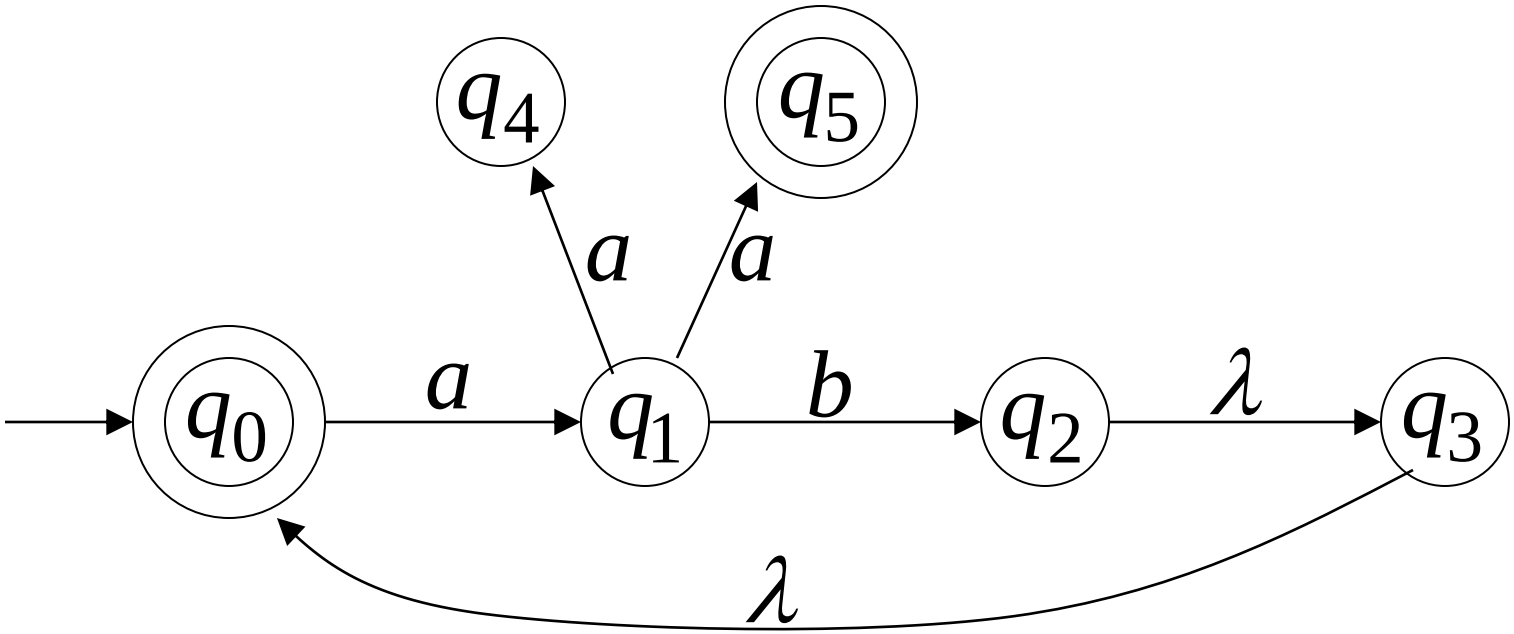
$\searrow \in F$

$F = \{q_0, q_5\}$



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

$\searrow \notin F$



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

3.7 THE EQUIVALENCE OF DFA AND NDFA

We naturally try to find the relation between DFA and NDFA. Intuitively we now feel that:

(i) A DFA can simulate the behaviour of NDFA by increasing the number of states. (In other words, a DFA $(Q, \Sigma, \delta, q_0, F)$ can be viewed as an NDFA $(Q, \Sigma, \delta', q_0, F)$ by defining $\delta'(q, a) = \{\delta(q, a)\}$.)

(ii) Any NDFA is a more general machine without being more powerful.

Construct a deterministic finite automaton equivalent to

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}), \delta, q_0, \{q_3\})$$

where δ is given by Table 3.6.

TABLE 3.6 State Table for Example 3.8

State/Σ	a	b
$\rightarrow q_0$	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)		q_2

Find a deterministic acceptor equivalent to

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

where δ is as given by Table 3.4.

TABLE 3.4 State Table for Example 3.7

State/Σ	a	b
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_0	q_1
(q_2)		q_0, q_1

TABLE 3.2 State Table for Example 3.6

State/Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	q_0, q_1

3.8 MEALY AND MOORE MODELS

3.8.1 FINITE AUTOMATA WITH OUTPUTS

The finite automata which we considered in the earlier sections have binary output. i.e. either they accept the string or they do not accept the string. This acceptability was decided on the basis of reachability of the final state by the initial state. Now, we remove this restriction and consider the model where the outputs can be chosen from some other alphabet. The value of the output function $Z(t)$ in the most general case is a function of the present state $q(t)$ and the present input $x(t)$, i.e.

$$Z(t) = \lambda(q(t), x(t))$$

where λ is called the output function. This generalized model is usually called the *Mealy machine*. If the output function $Z(t)$ depends only on the present state and is independent of the current input, the output function may be written as

$$Z(t) = \lambda(q(t))$$

This restricted model is called the *Moore machine*. It is more convenient to use Moore machine in automata theory. We now give the most general definitions of these machines.

Definition 3.8 A Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

(i) Q is a finite set of states;
(ii) Σ is the input alphabet;
(iii) Δ is the output alphabet;
(iv) δ is the transition function $\Sigma \times Q$ into Q;
(v) λ is the output function mapping Q into Δ; and
(vi) q_0 is the initial state.

Definition 3.9 A Mealy machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where all the symbols except λ have the same meaning as in the Moore machine. λ is the output function mapping $\Sigma \times Q$ into Δ.

TABLE 3.8 A Moore Machine

Present state	*Next state* δ		*Output*
	$a = 0$	$a = 1$	λ
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

For the input string 0111, the transition of states is given by $q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$. The output string is 00010. For the input string Λ, the output is $\lambda(q_0) = 0$.

Transition Table 3.9 describes a Mealy machine.

TABLE 3.9 A Mealy Machine

Present state	*Next state*			
	$a = 0$		$a = 1$	
	state	*output*	*state*	*output*
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_4	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Note: For the input string 0011, the transition of states is given by $q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$, and the output string is 0100. In the case of a Mealy machine, we get an output only on the application of an input symbol. So for the input string Λ, the output is only Λ. It may be observed that in the case of a Moore machine, we get $\lambda(q_0)$ for the input string Λ.

Consider the Mealy machine described by the transition table given by Table 3.10. Construct a Moore machine which is equivalent to the Mealy machine.

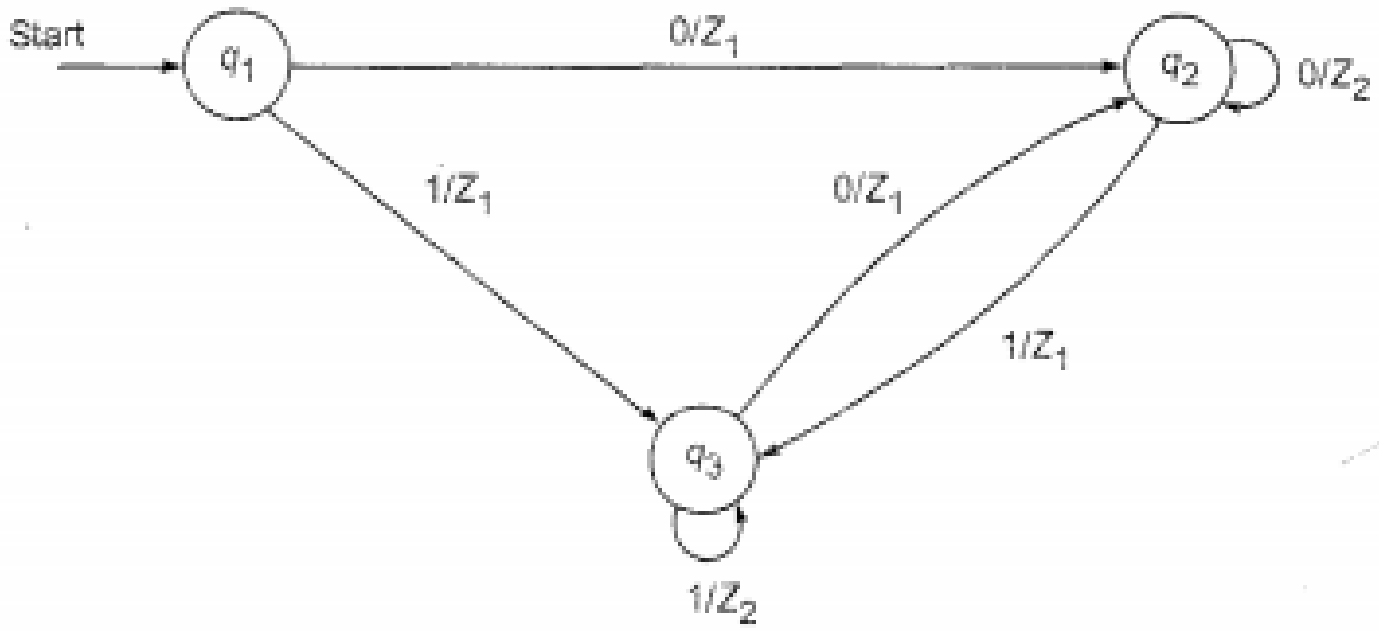
TABLE 3.10 Mealy Machine of Example 3.9

Present state	*Next state*			
	Input a = 0		*Input a = 1*	
	state	*output*	*state*	*output*
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Construct a Mealy Machine which is equivalent to the Moore machine given by Table 3.14.

TABLE 3.14 Moore Machine of Example 3.10

Present state	*Next state*		*Output*
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0


Start
q_1
$0/Z_1$
q_2
$0/Z_2$
$1/Z_1$
$0/Z_1$
$1/Z_1$
q_3
$1/Z_2$

3.9.1 Construction of Minimum Automaton

Step 1 (Construction of π_0). By definition of 0-equivalence, $\pi_0 = \{ Q_1^0, Q_2^0 \}$ where Q_1^0 is the set of all final states and $Q_2^0 = Q - Q_1^0$.

Step 2 (Construction of π_{k+1} from π_k). Let Q_i^k be any subset in π_k. If q_1 and q_2 are in Q_i^k, they are $(k + 1)$-equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k-equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are in the same equivalence class in π_k for every $a \in \Sigma$. If so, q_1 and q_2 are $(k + 1)$-equivalent. In this way, Q_i^k is further divided into $(k + 1)$-equivalence classes. Repeat this for every Q_i^k in π_k to get all the elements of π_{k+1}.

Step 3 Construct π_n for $n = 1, 2, \ldots$ until $\pi_n = \pi_{n+1}$.

Step 4 (Construction of minimum automaton). For the required minimum state automaton, the states are the equivalence classes obtained in step 3, i.e. the elements of π_n. The state table is obtained by replacing a state q by the corresponding equivalence class $[q]$.

Remark In the above construction, the crucial part is the construction of equivalence classes; for, after getting the equivalence classes, the table for minimum automaton is obtained by replacing states by the corresponding equivalence classes. The number of equivalence classes is less than or equal to $| Q |$. Consider an equivalence class $[q_1] = \{q_1, q_2, \ldots, q_k\}$. If q_1 is reached while processing $w_1 w_2 \in T(M)$ with $\delta(q_0, w_1) = q_1$, then $\delta(q_1, w_2) \in F$. So, $\delta(q_i, w_2) \in F$ for $i = 2, \ldots, k$. Thus we see that q_i, $i = 2, \ldots, k$ is reached on processing some $w \in T(M)$ iff q_1 is reached on processing w, i.e. q_1 of $[q_1]$ can play the role of $q_2, \ldots, q_k$. The above argument explains why we replace a state by the corresponding equivalence class.

Note: The construction of π_0, π_1, π_2, etc. is easy when the transition table is given. $\pi_0 = \{Q_1^0, Q_2^0\}$, where $Q_1^0 = F$ and $Q_2^0 = Q - F$. The subsets in π_1 are obtained by further partitioning the subsets of π_0. If $q_1, q_2 \in Q_1^0$, consider the states in each a-column, where $a \in \Sigma$ corresponding to q_1 and q_2. If they are in the same subset of π_0, q_1 and q_2 are 1-equivalent. If the states under some a-column are in different subsets of π_0, then q_1 and q_2 are not 1-equivalent. In general, $(k + 1)$-equivalent states are obtained by applying the above method for q_1 and q_2 in Q_i^k.

Construct a minimum state automaton equivalent to the finite automaton described by Fig. 3.12.

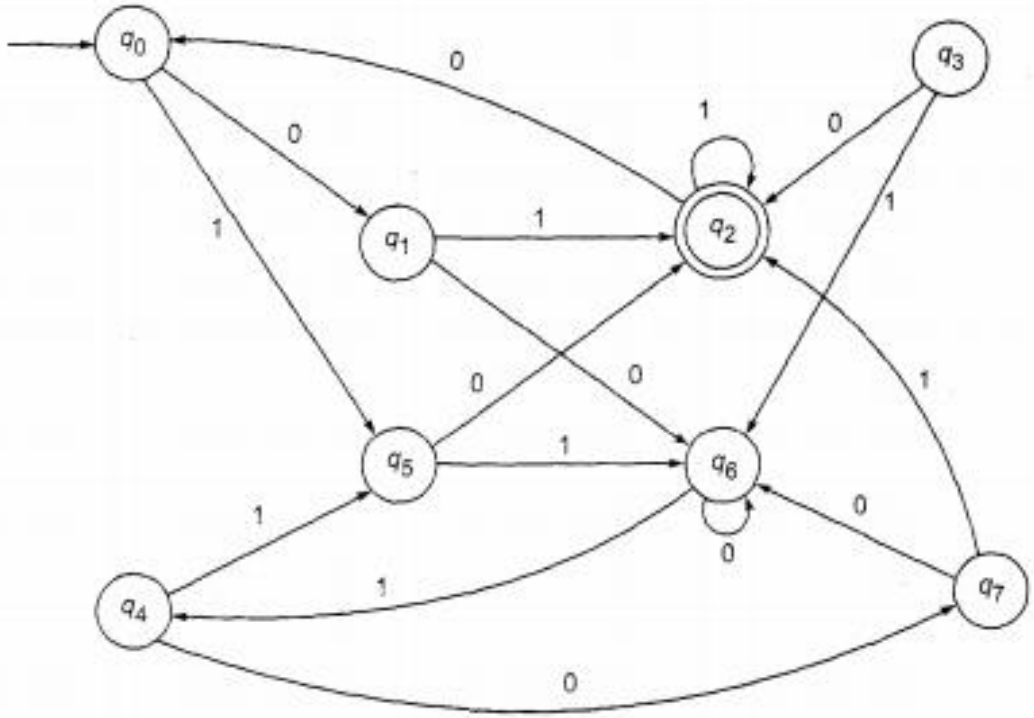


Fig. 3.12 Finite automaton of Example 3.13.

Construct the minimum state automaton equivalent to the transition diagram given by Fig. 3.14.

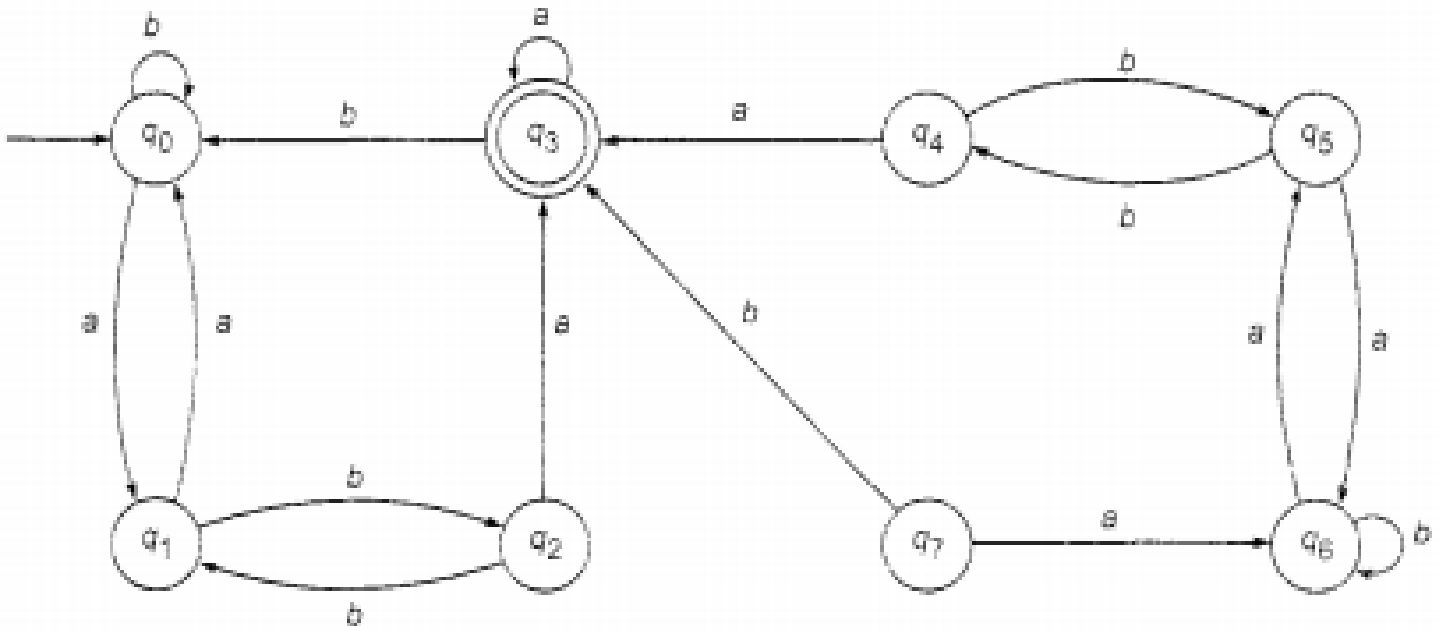


Fig. 3.14 Finite automaton of Example 3.14.