CSE322

Finite Automata

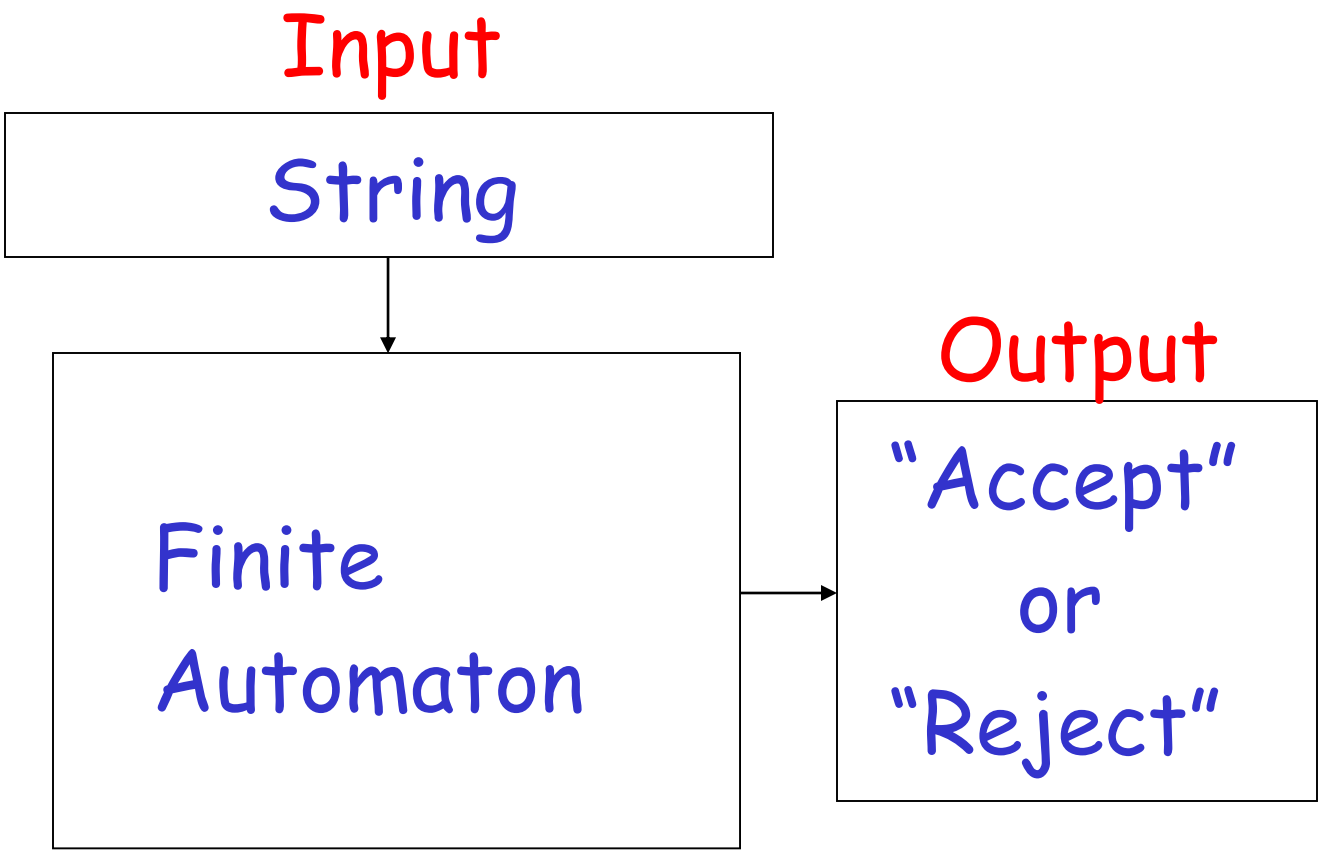
Lecture #2

UNIT I
FINITE AUTOMATA

UNIT-I SYLLABUS

- FINITE AUTOMATA : The Equivalence of Deterministic and Non-deterministic Finite Automata, Definition and Description of a Finite Automaton, Deterministic and Non-deterministic Finite State Machines, Acceptability of a String by a Finite Automaton, Mealy and Moore Machines, Minimization of Finite Automata, Basics of Strings and Alphabets, Transition Graph and Properties of Transition Functions, Regular Languages.

Finite Automaton



Finite Automata

LOVELY PROFESSIONAL UNIVERSITY

- Finite automata are used to recognize patterns.
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

Theory of Automata

- Theory of automata is the study of abstract machines and the computation problems that can be solved using these machines.

- The abstract machine is called the automata.

- The automaton consists of states and transitions. The **State** is represented by **circles**, and the **Transitions** is represented by **arrows**.

- Automata is the kind of machine which takes some string as input and this input goes through a finite number of states and may enter in the final state.

Terminologies used in Automata

- Symbols:
- Symbols are an entity or individual objects, which can be any letter, alphabet or any picture.

- Example:
- 1, a, b, #

- Alphabets:
- Alphabets are a finite set of symbols. It is denoted by Σ.

- Examples:
- Σ = {a, b}
- Σ = {A, B, C, D}
- Σ = {0, 1, 2}
- Σ = {0, 1, ..., 5}
- Σ = {#, β, Δ}

- String:
- It is a finite collection of symbols from the alphabet. The string is denoted by w.
- Example 1:
- If ∑ = {a, b}, various string that can be generated from ∑ are {ab, aa, aaa, bb, bbb, ba, aba.....}.

- A string with zero occurrences of symbols is known as an empty string. It is represented by ε.
- The number of symbols in a string w is called the length of a string. It is denoted by |w|.
- Example 2:
- w = 010
- Length of String |w| = 3

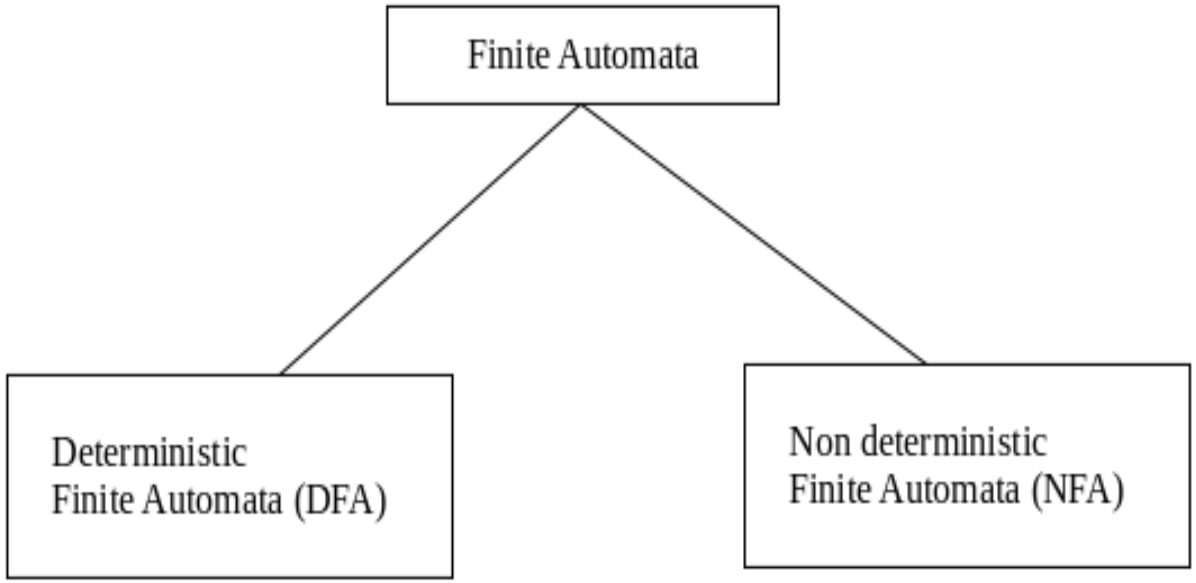
- Language:
- A language is a collection of appropriate string. A language which is formed over Σ can be **Finite** or **Infinite**.

- Example: 1
- L1 = {Set of string of length 2} = {aa, bb, ba, bb} **Finite Language**
- Example: 2
- L2 = {Set of all strings starts with 'a'} = {a, aa, aaa, abb, abbb, ababb} **Infinite Language**

Types of Automata

- There are two types of finite automata:

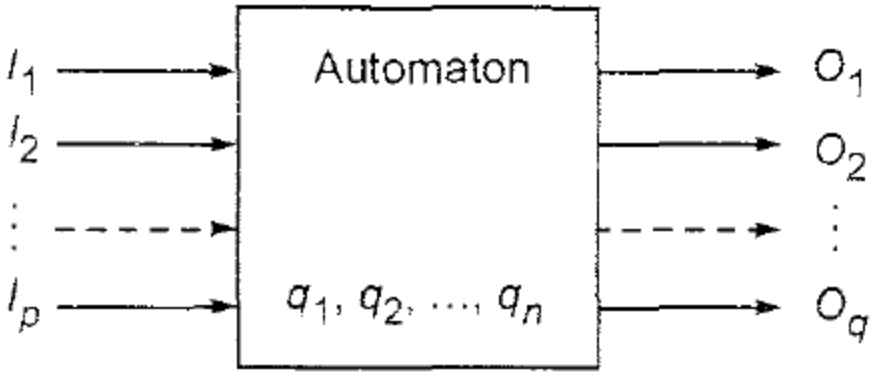
1. DFA(deterministic finite automata)
2. NFA(non-deterministic finite automata)



Automaton:

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.



Finite Automata Model:

- Finite automata can be represented by input tape and finite control.

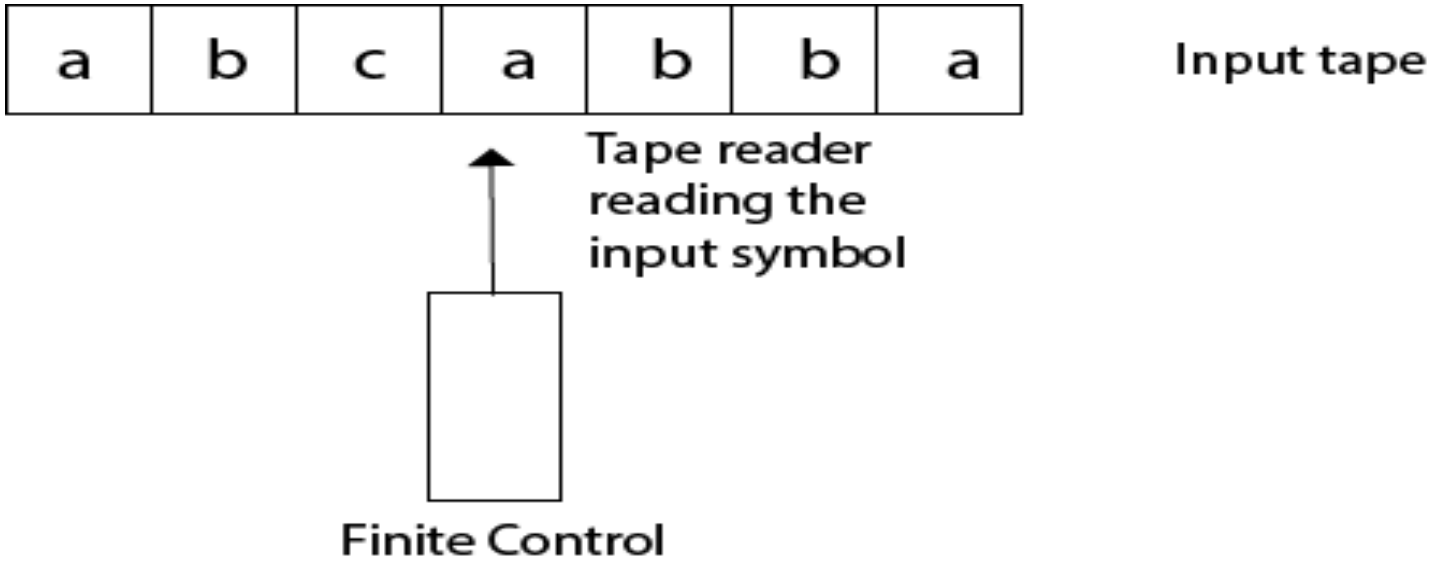


Fig :- Finite automata model

- **Input tape:** It is a linear tape having some number of cells. Each input symbol is placed in each cell.

- **Finite control:** The finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right, and at a time only one input symbol is read.

Formal Definition of Finite Automaton

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

Q : Finite non-empty set of states

Σ : Finite non empty set of input alphabets

δ : (direct) transition function that maps **Q x** Σ **-> Q**

q_0 : initial state

F : set of final states

There are numerous applications of Formal languages and Automata Theory like:

- Text processing, Compilers and Hardware Design
- Motors and Vending machines
- Sensors and Transducers
- Automata Simulators
- And many more ….

Transition Diagram

- A transition diagram or state transition diagram is a directed graph which can be constructed as follows:
- There is a node for each state in Q, which is represented by the circle.
- There is a directed edge from node q to node p labeled a if $\delta(q, a) = p$.
- In the start state, there is an arrow with no source.
- Accepting states or final states are indicating by a double circle.

Notations used in the transition diagram

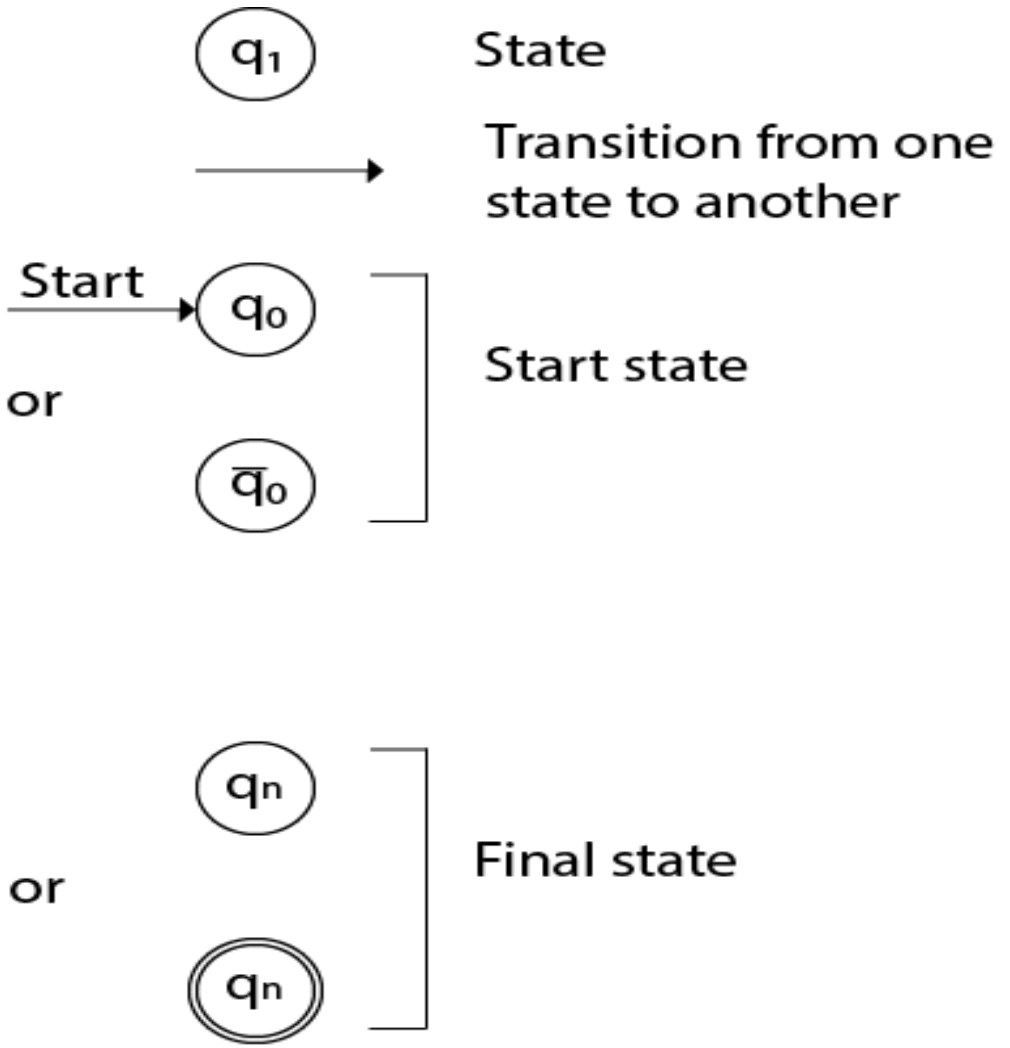
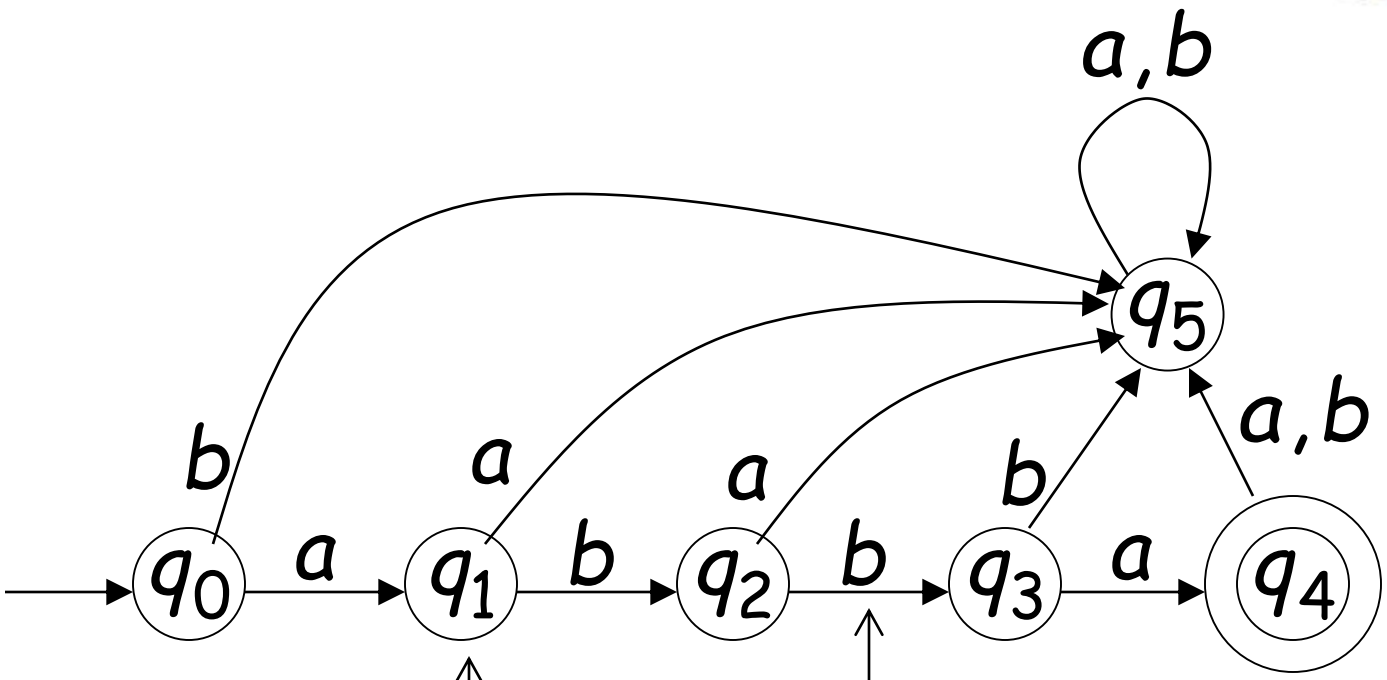


Fig:- Notations

Transition Graph



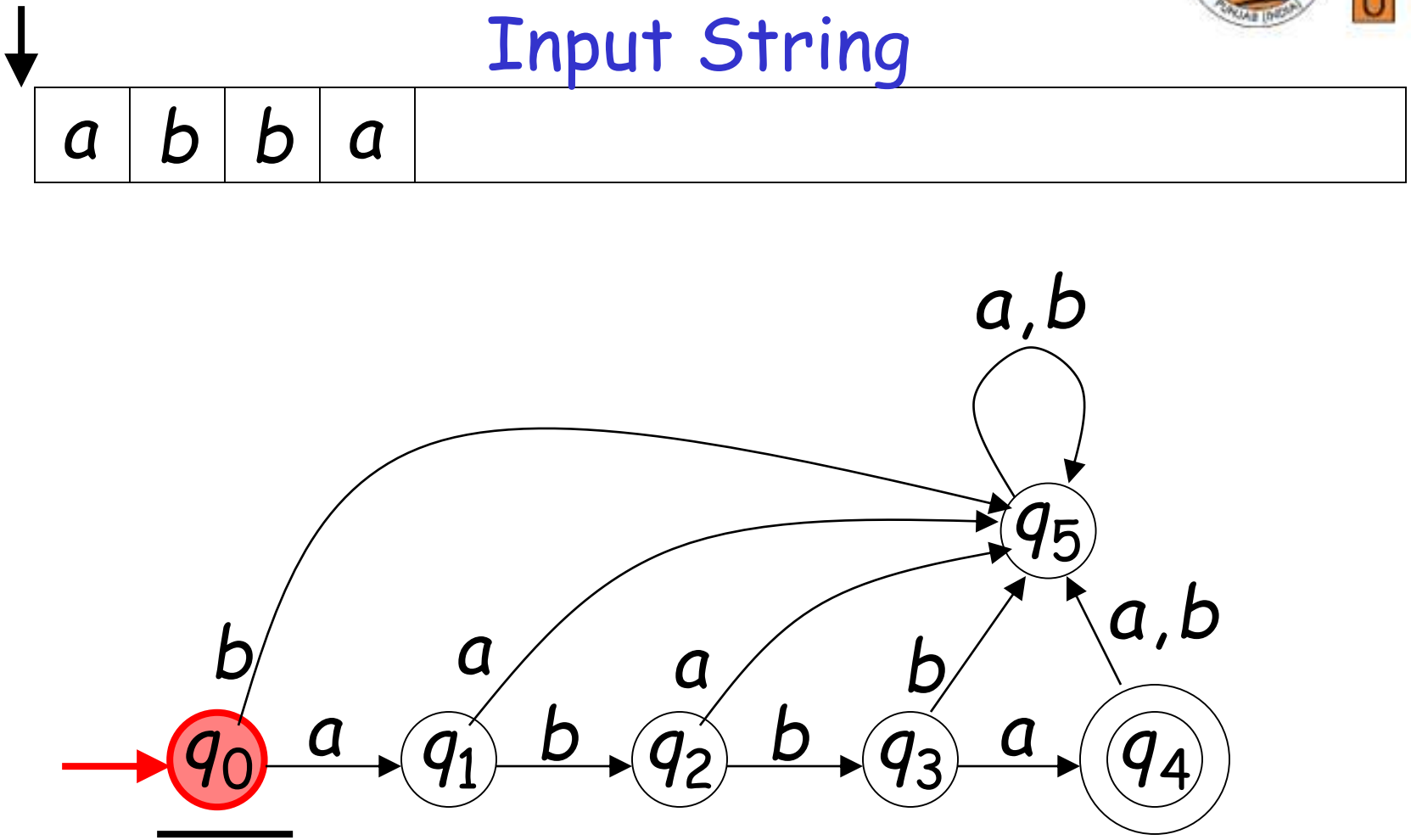
initial state

state

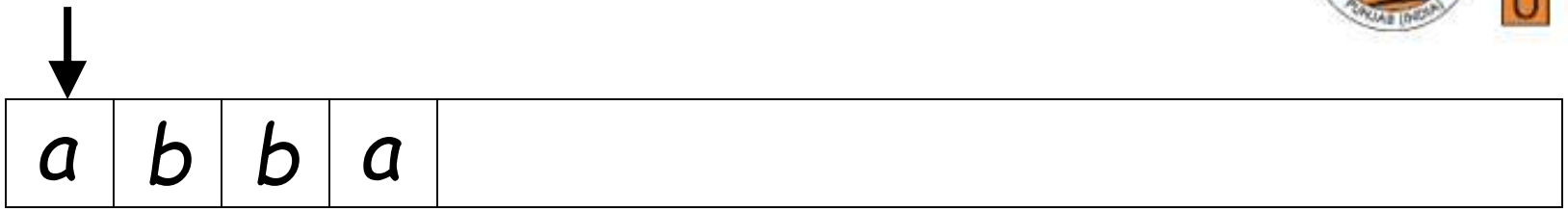
transition

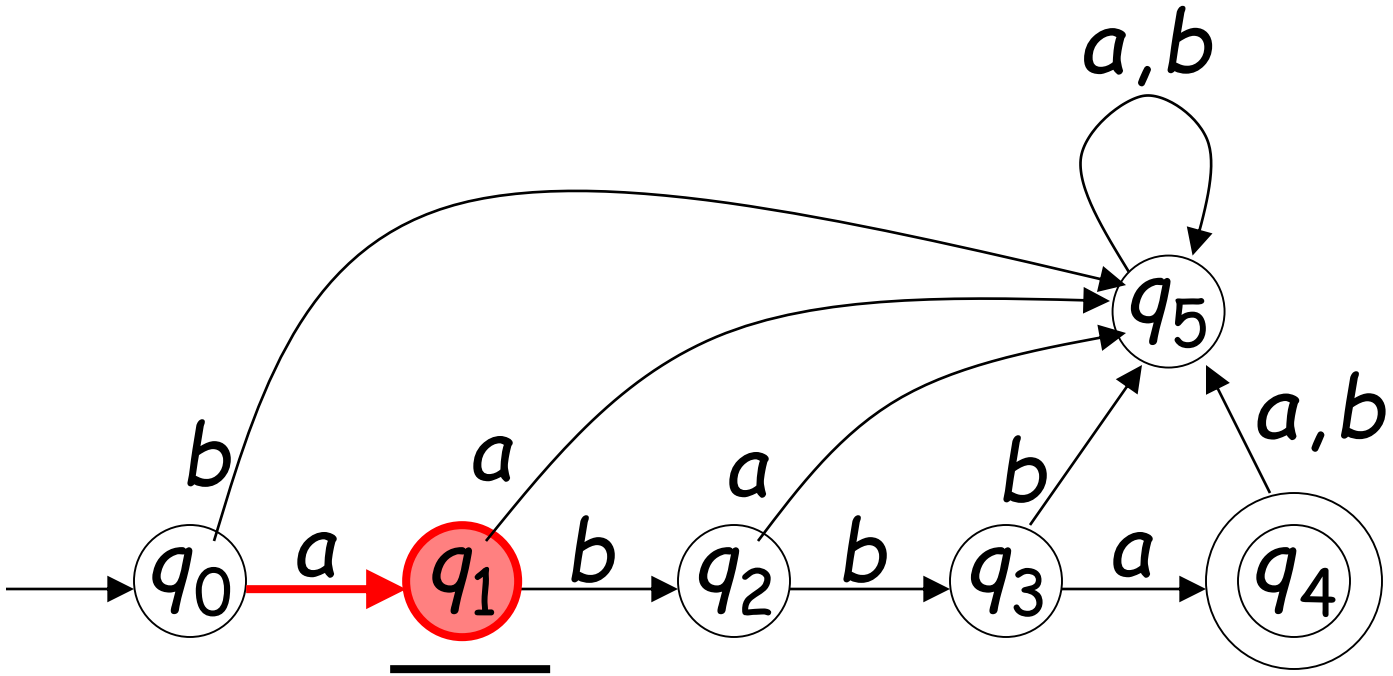
final/accepting state

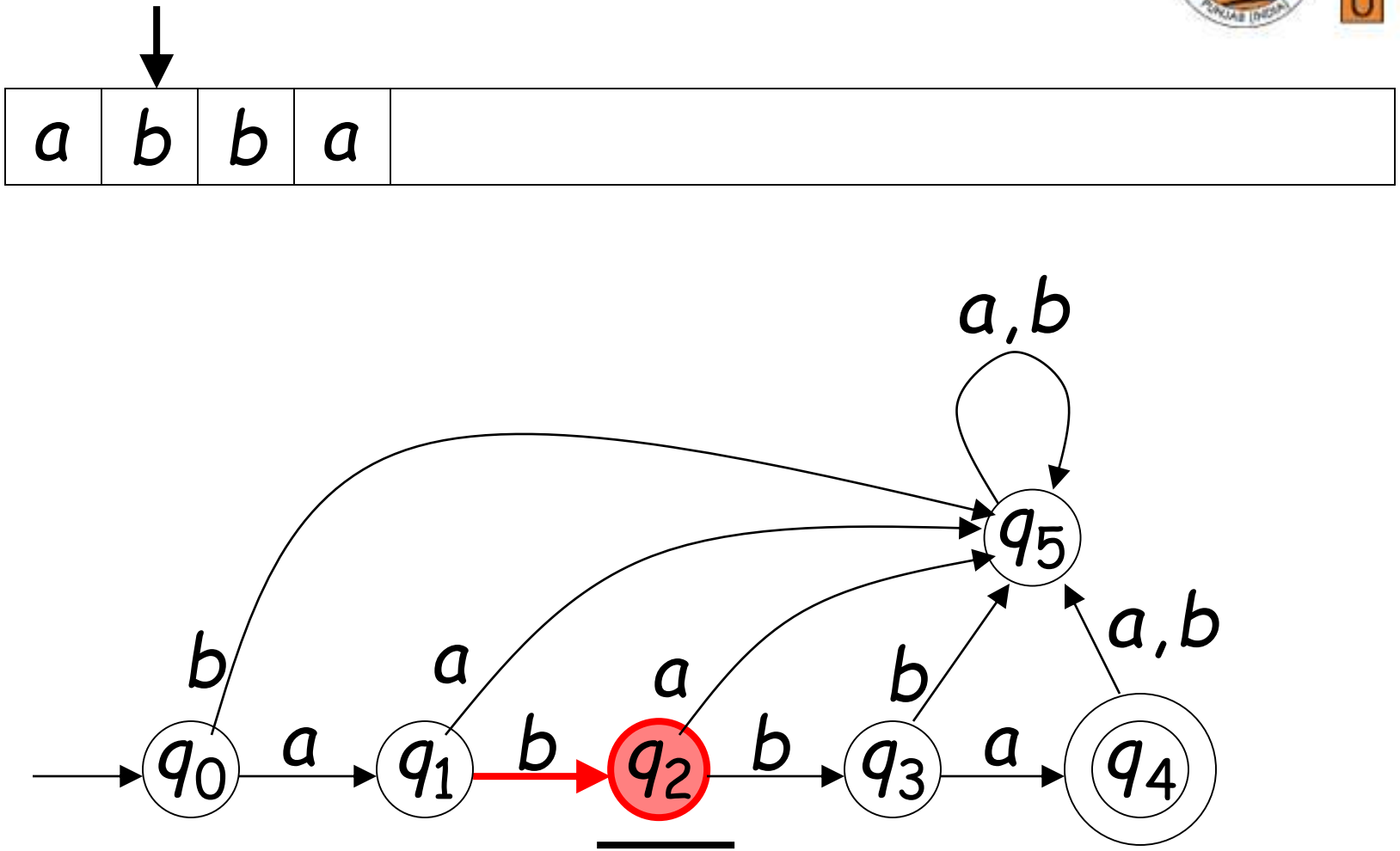
Initial Configuration

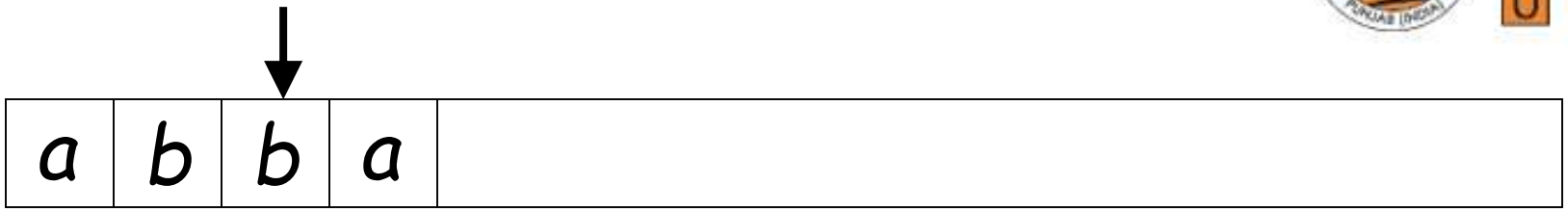


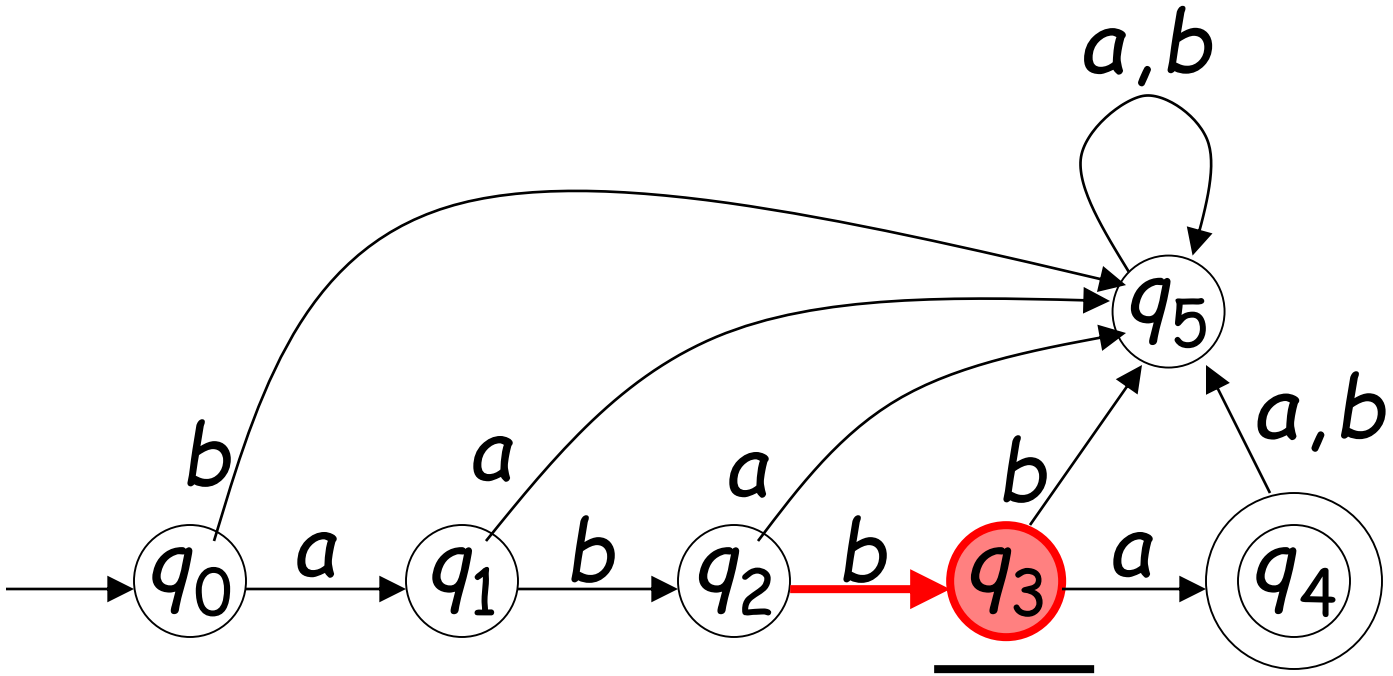
Reading the Input

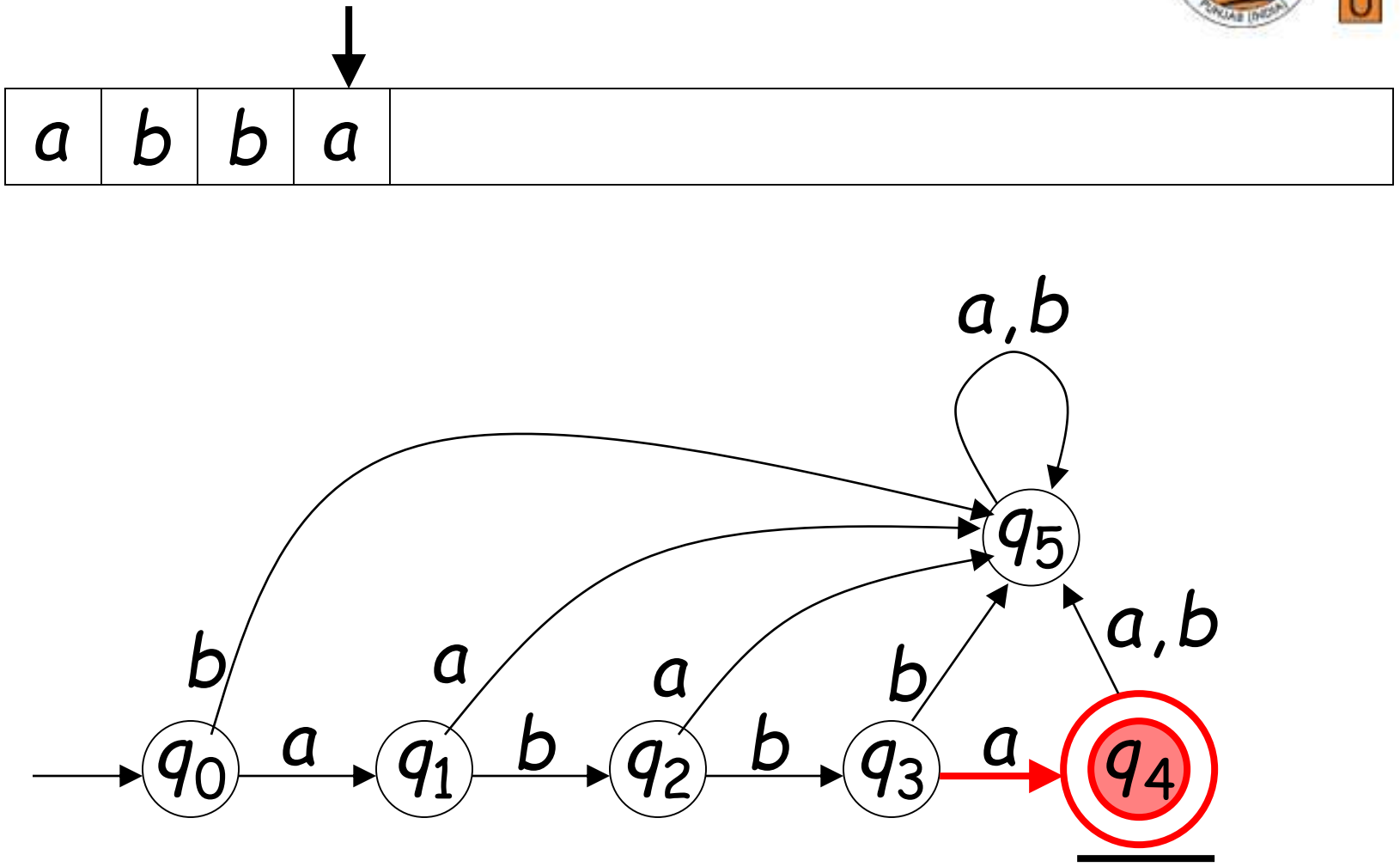




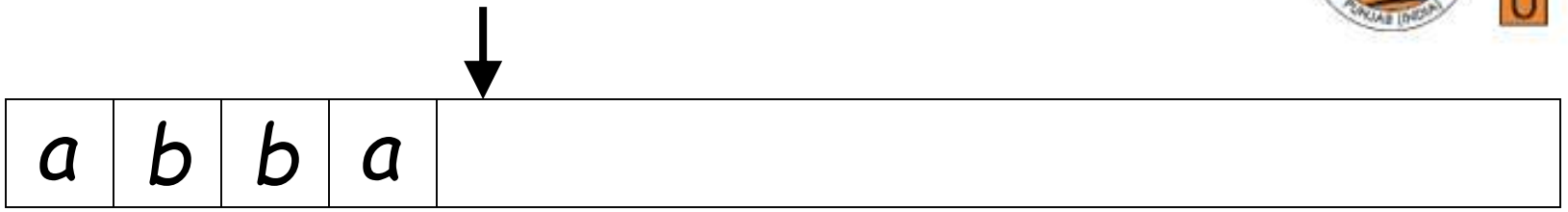

a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

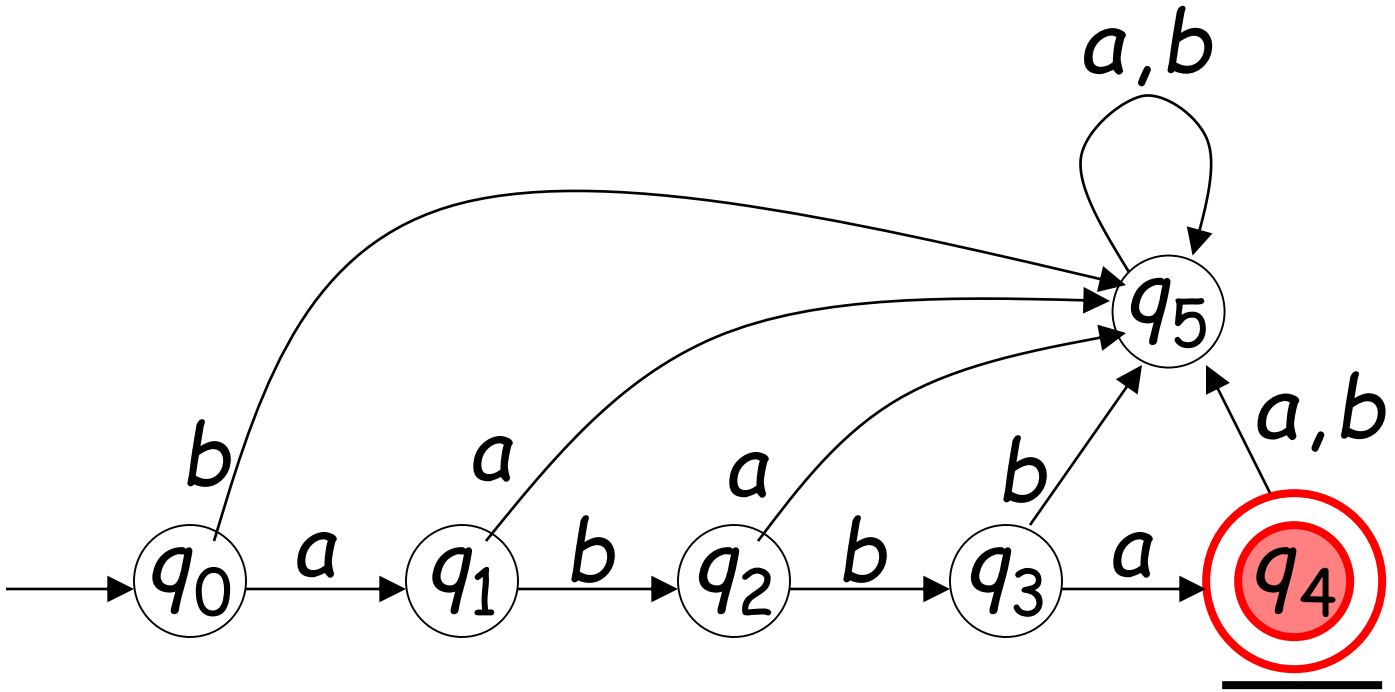

a
b
b
a


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


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

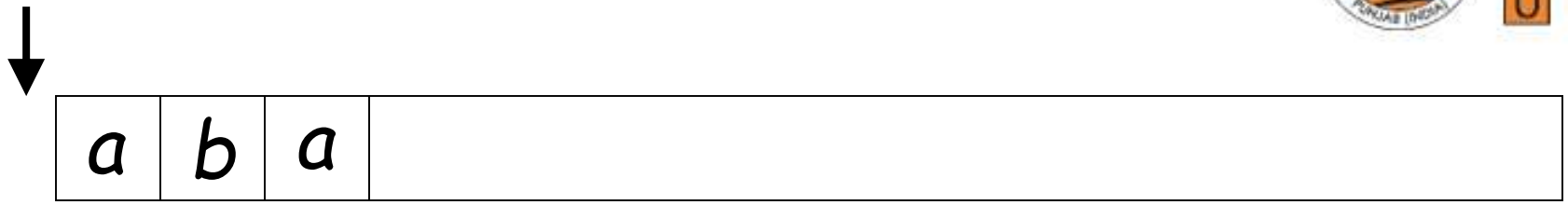
Input finished

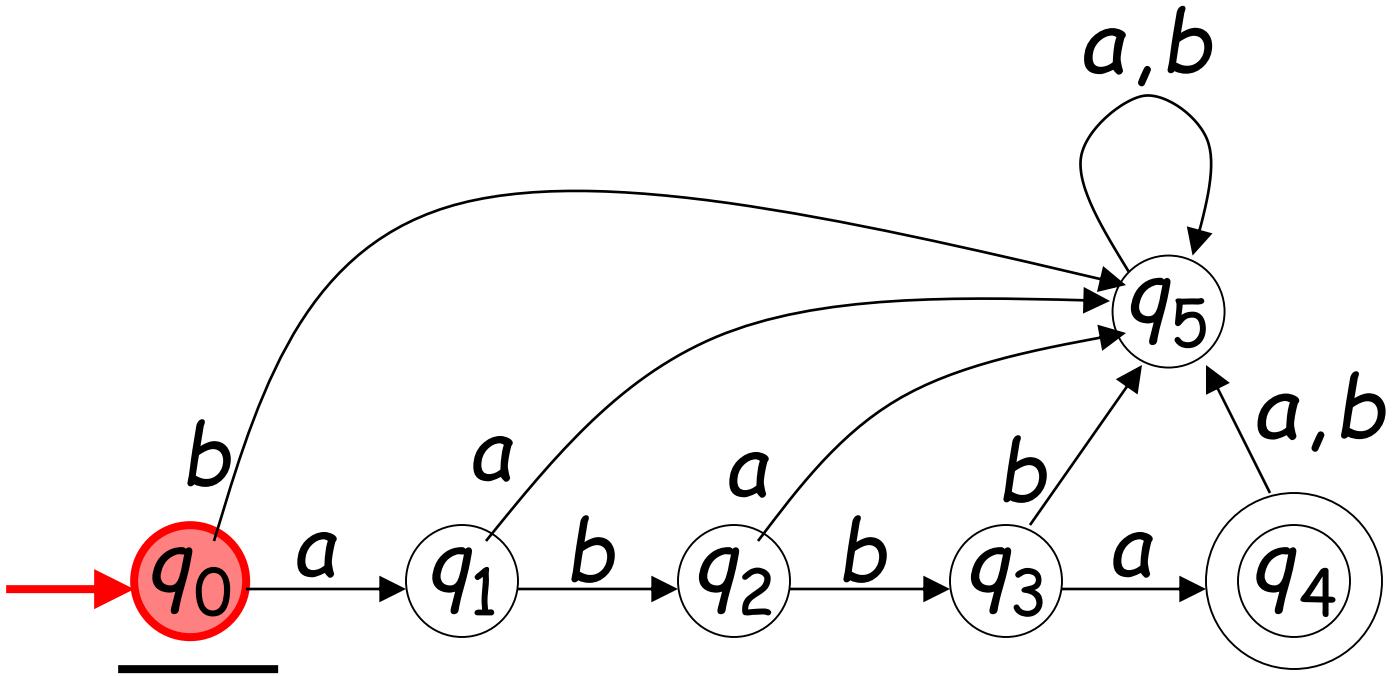


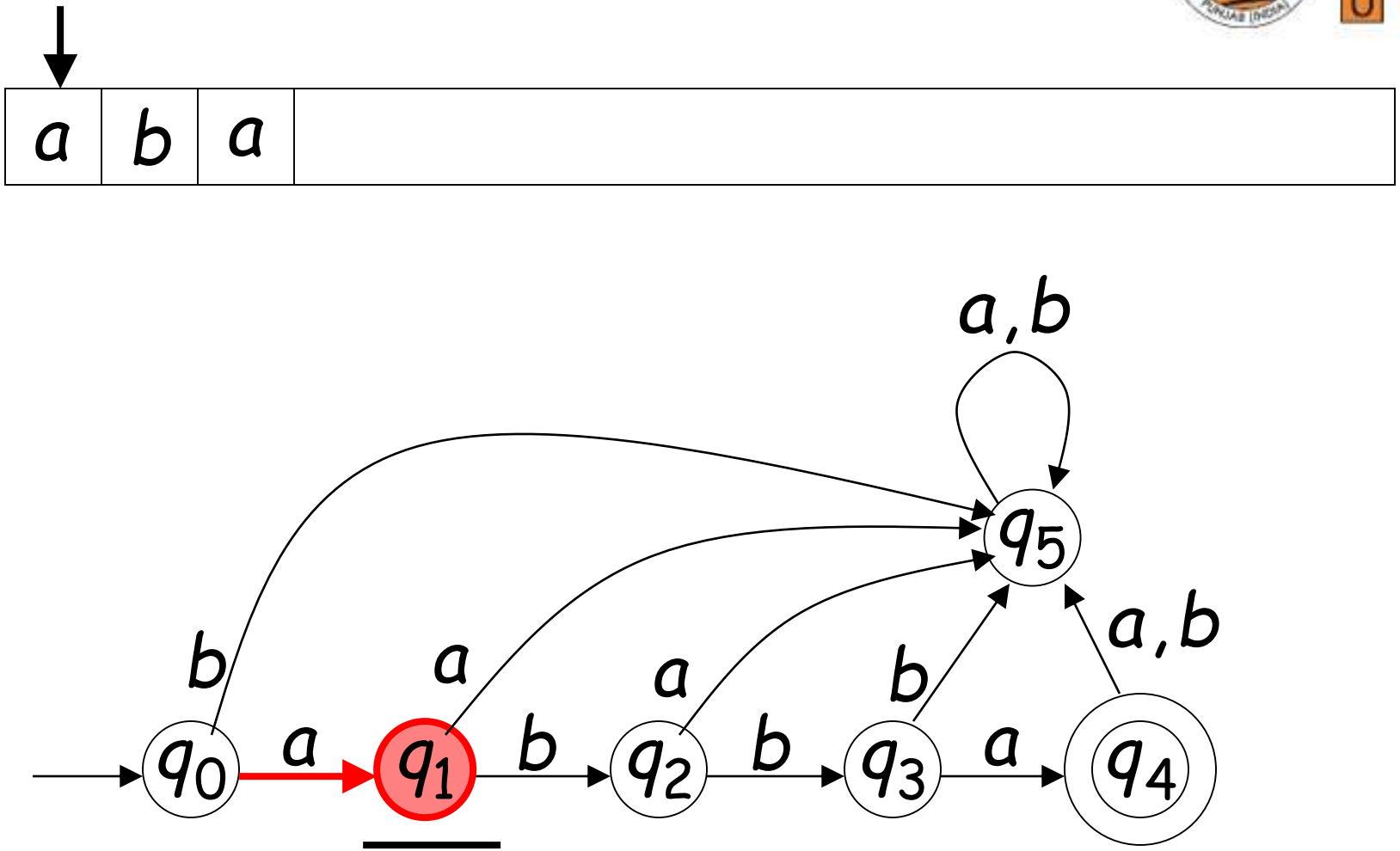


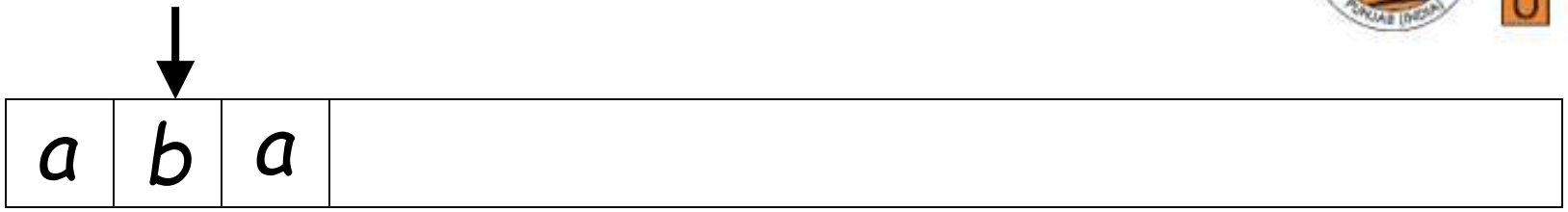
accept

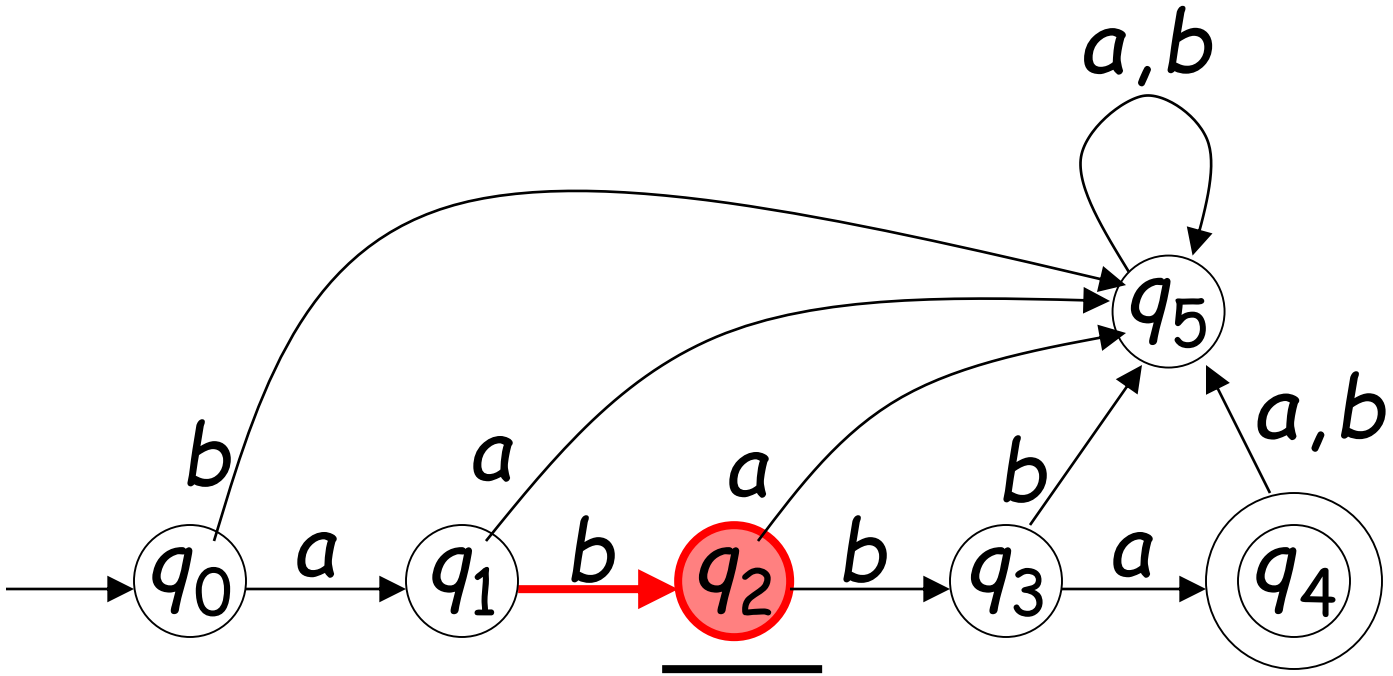
Rejection

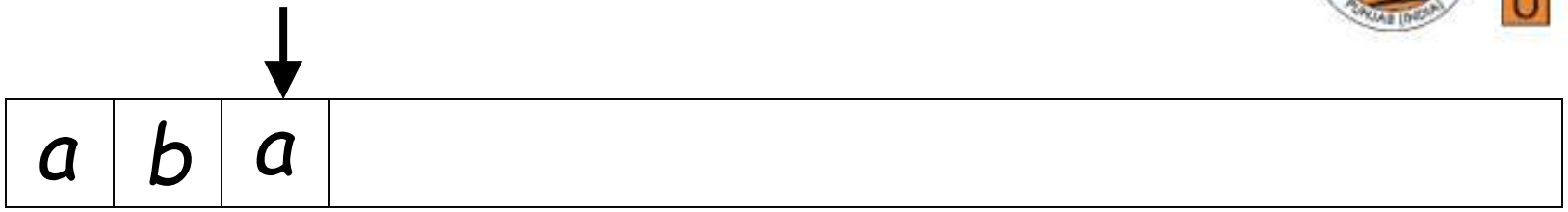


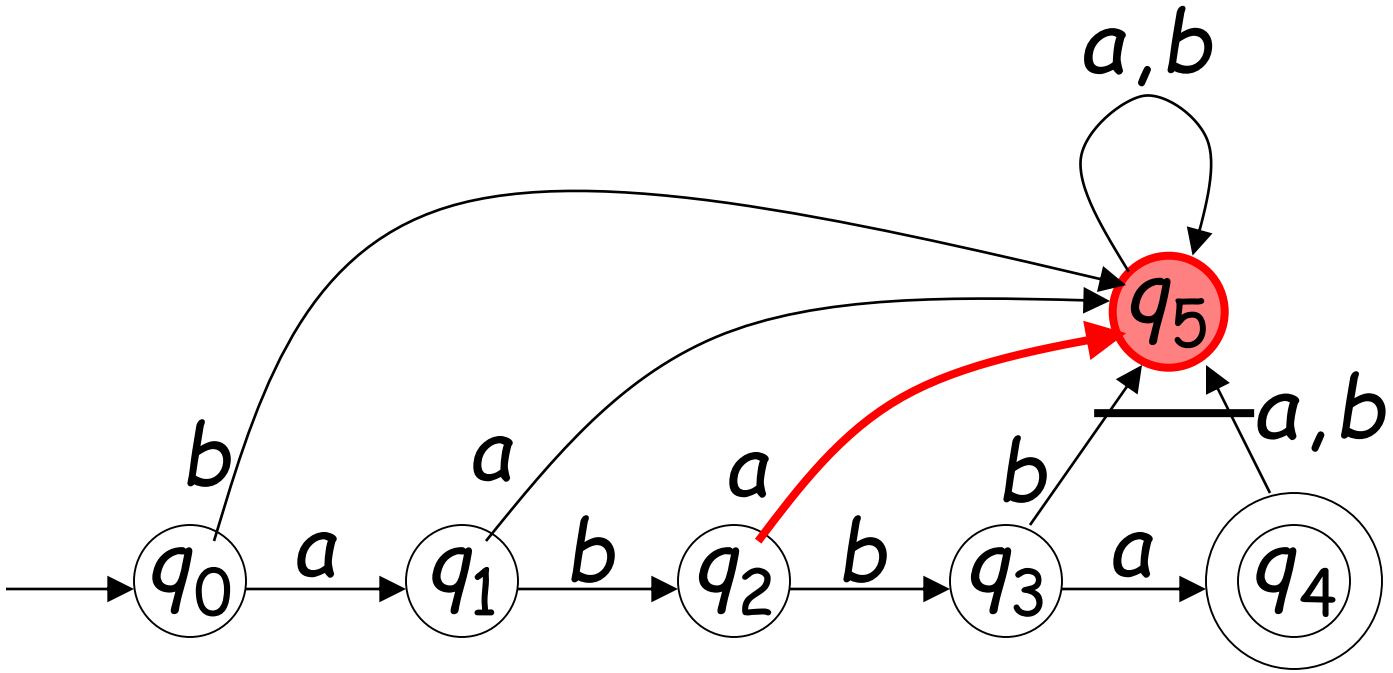



a b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

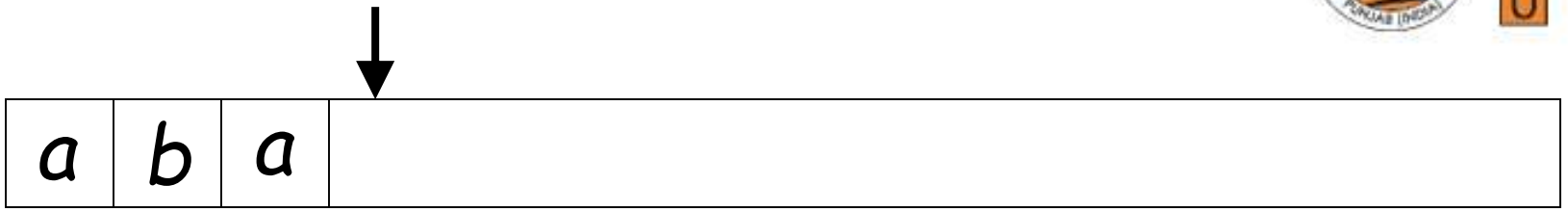

a
b
a

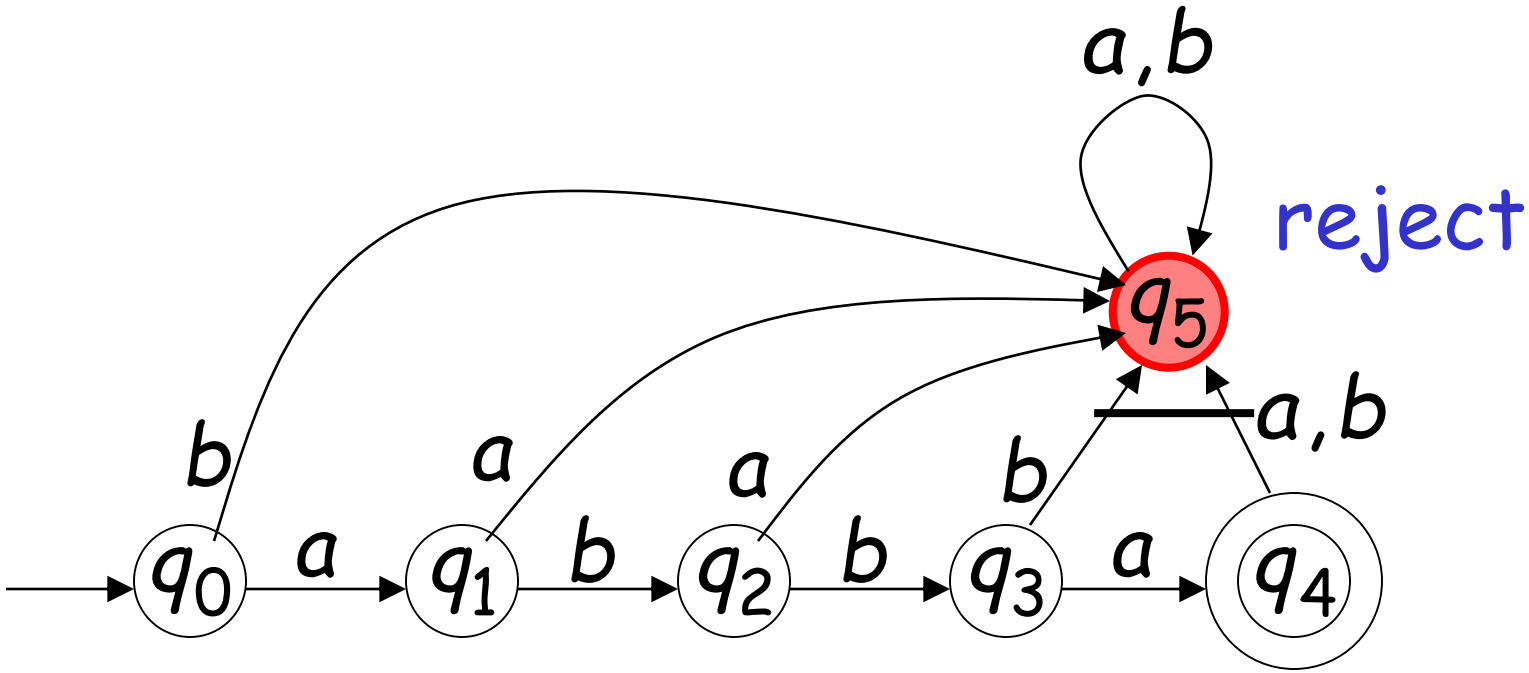

a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


a
b
a

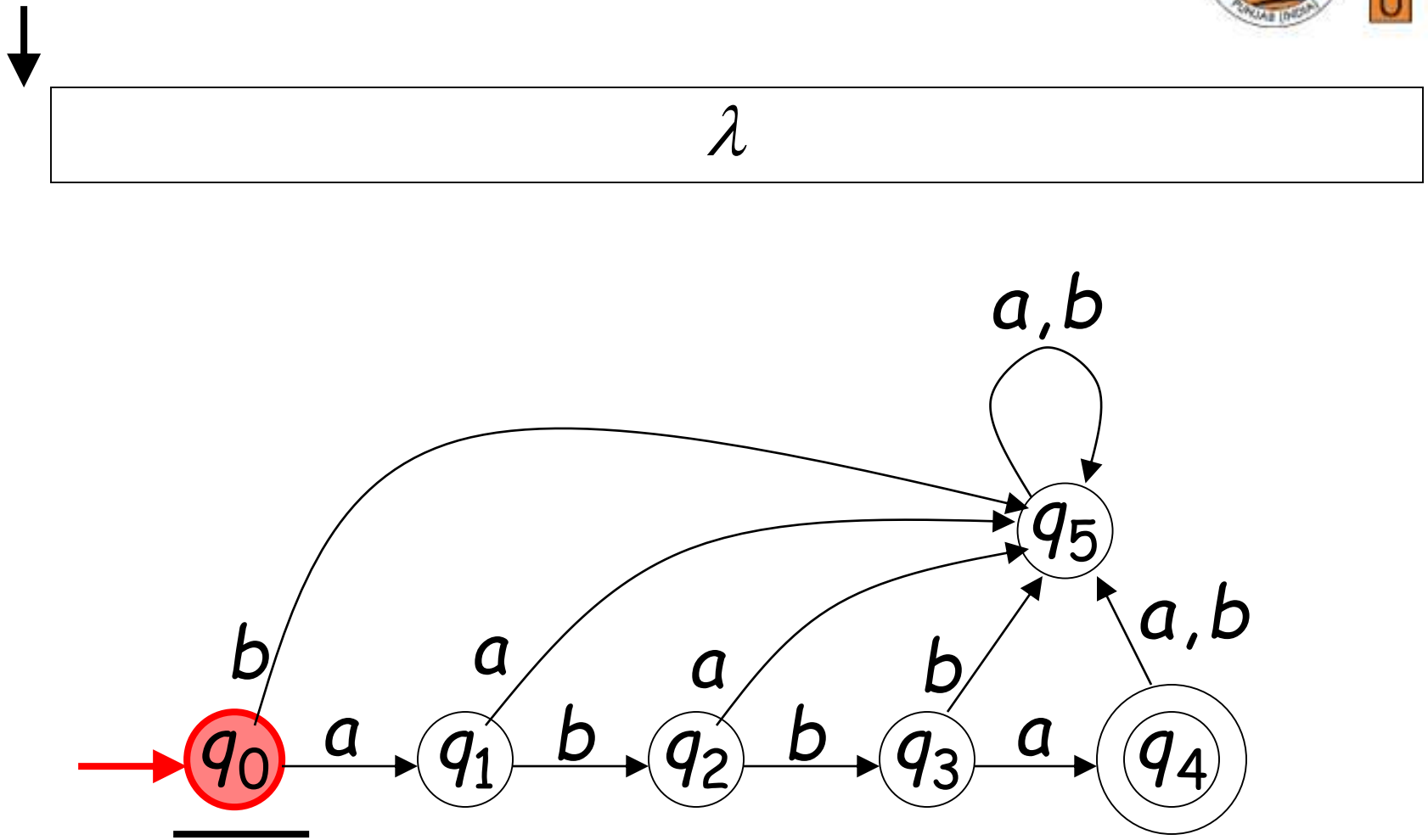

a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

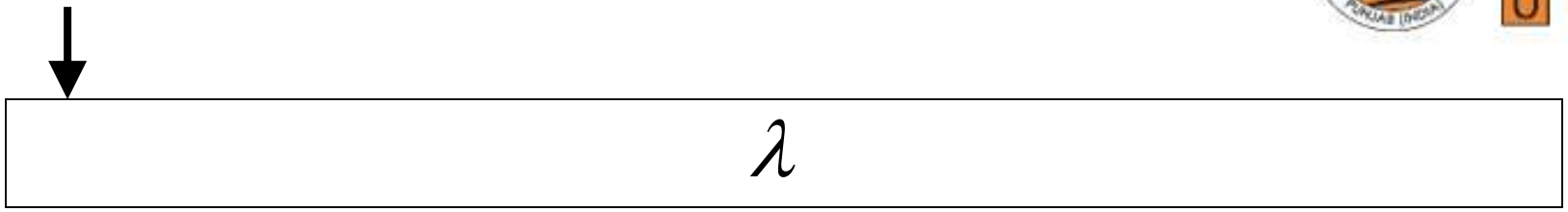
Input finished

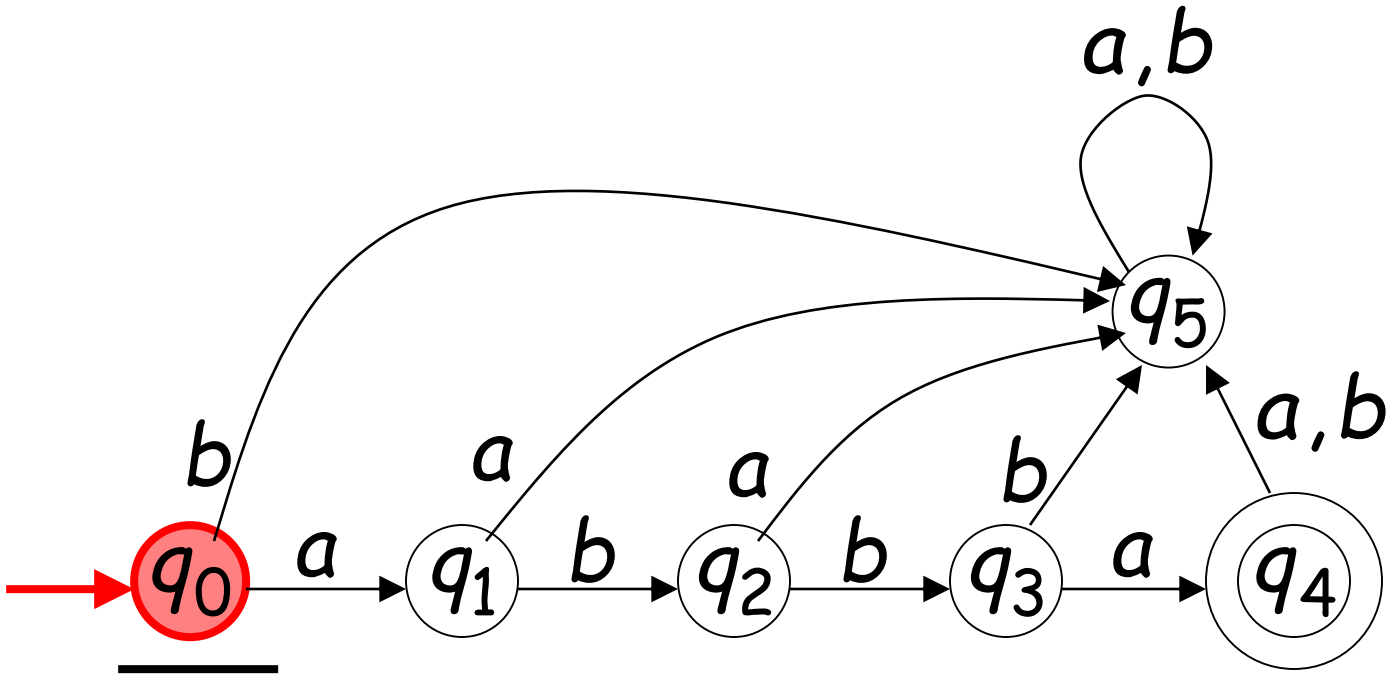




Another Rejection

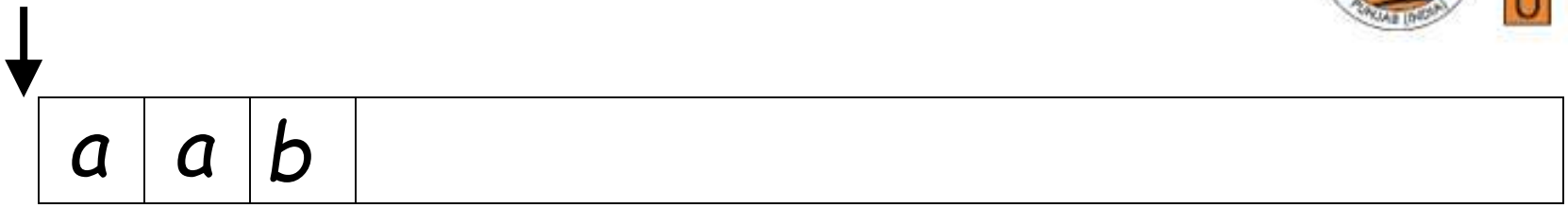


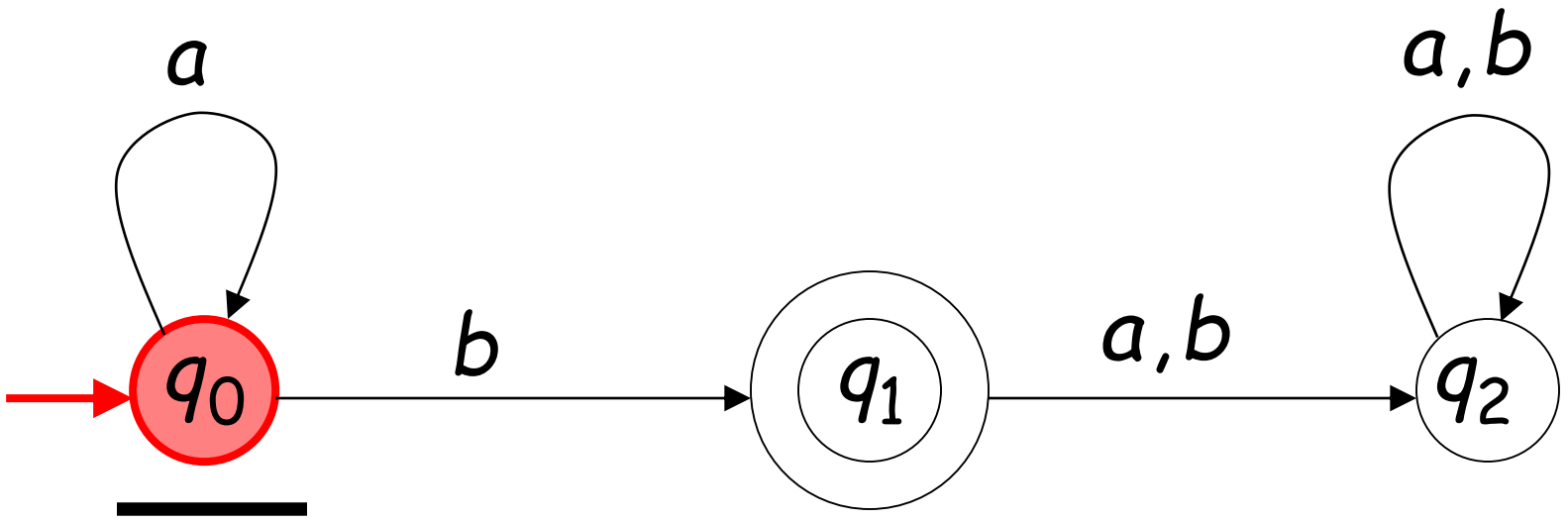


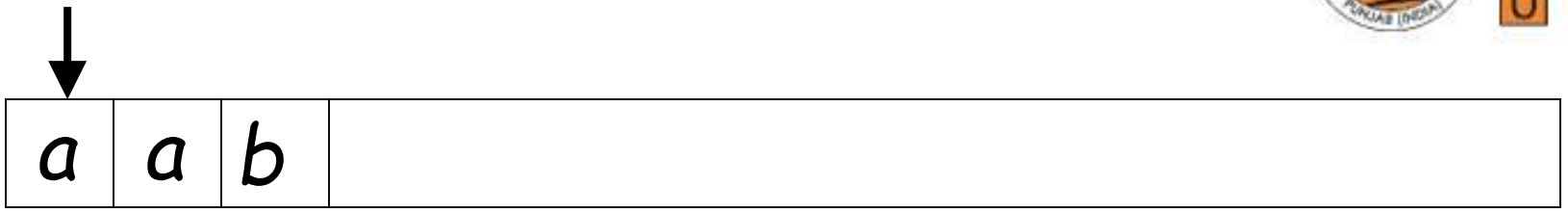


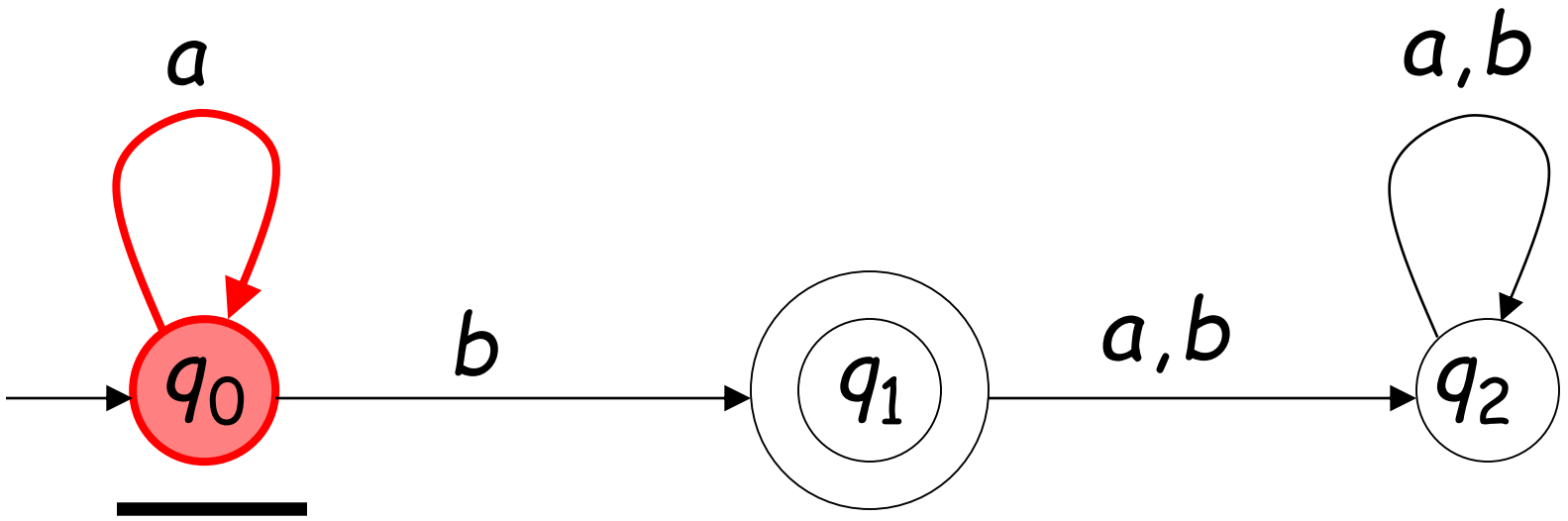
reject

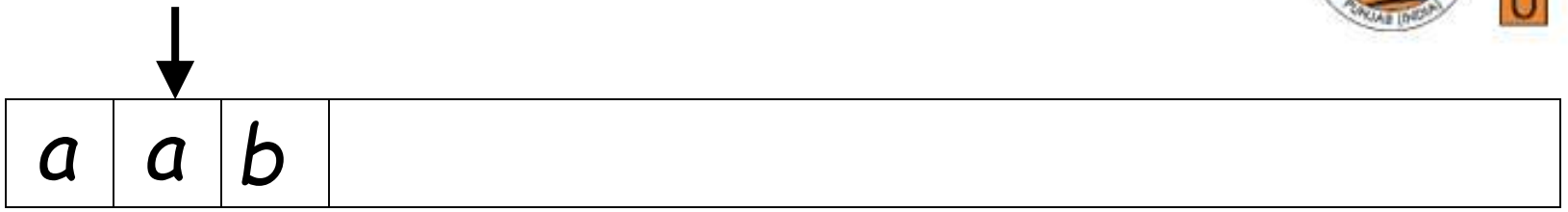
Another Example

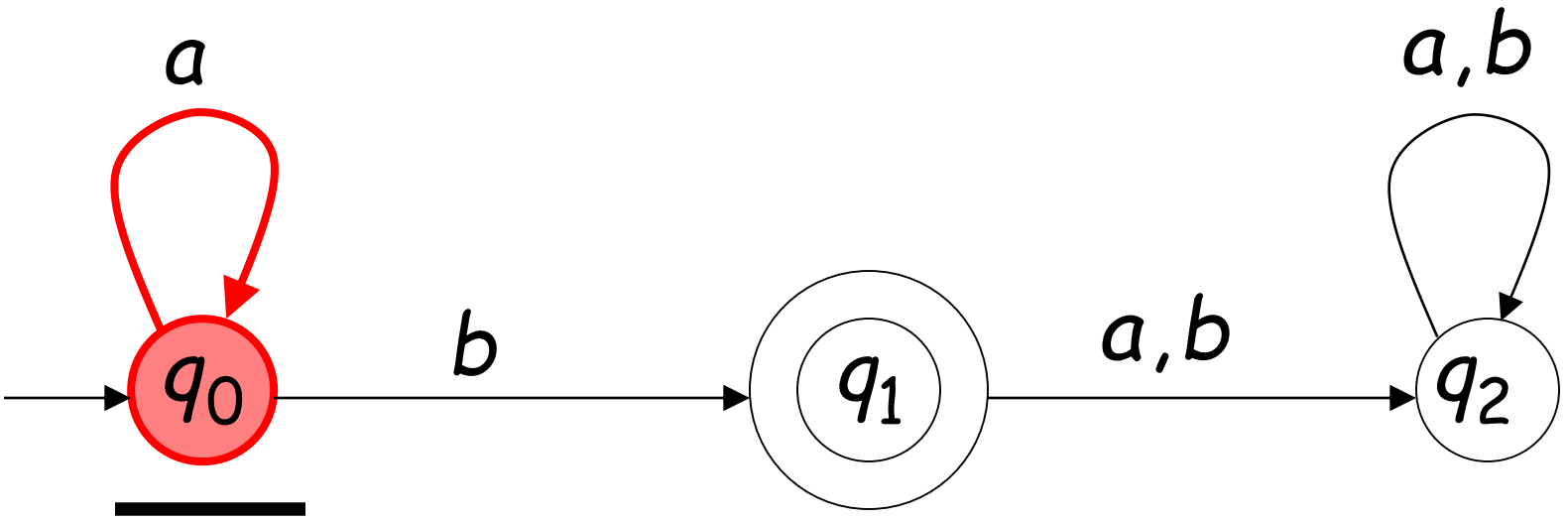


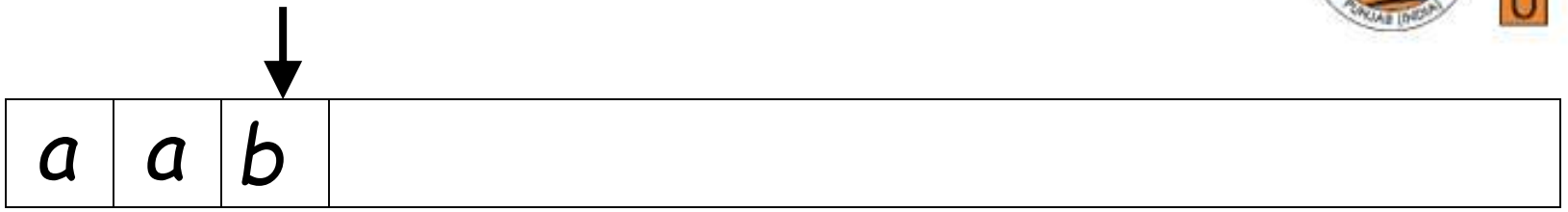


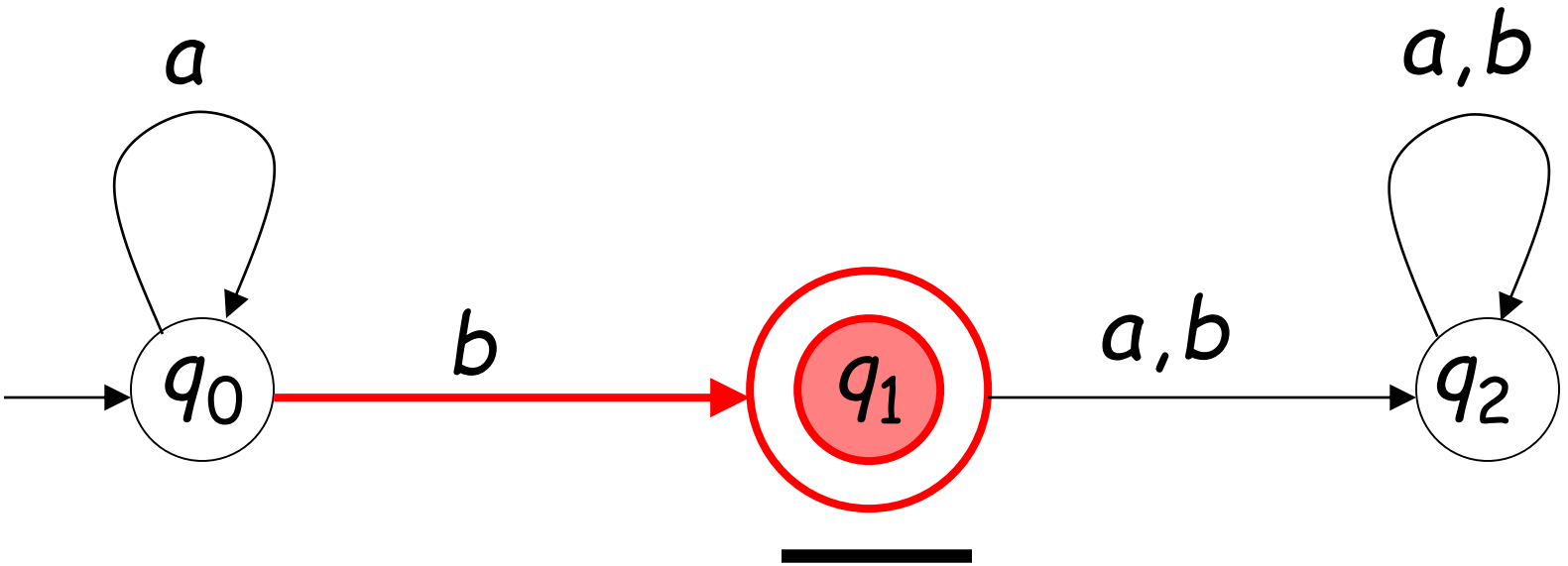

a
a
b


a
a,b
q0
b
q1
a,b
q2

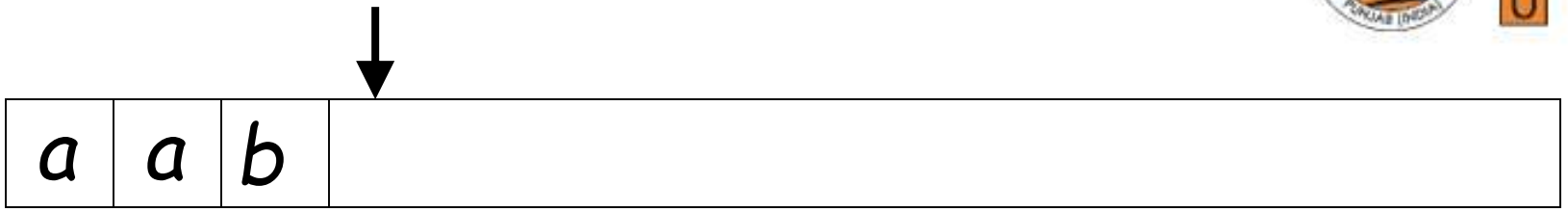

a
a
b

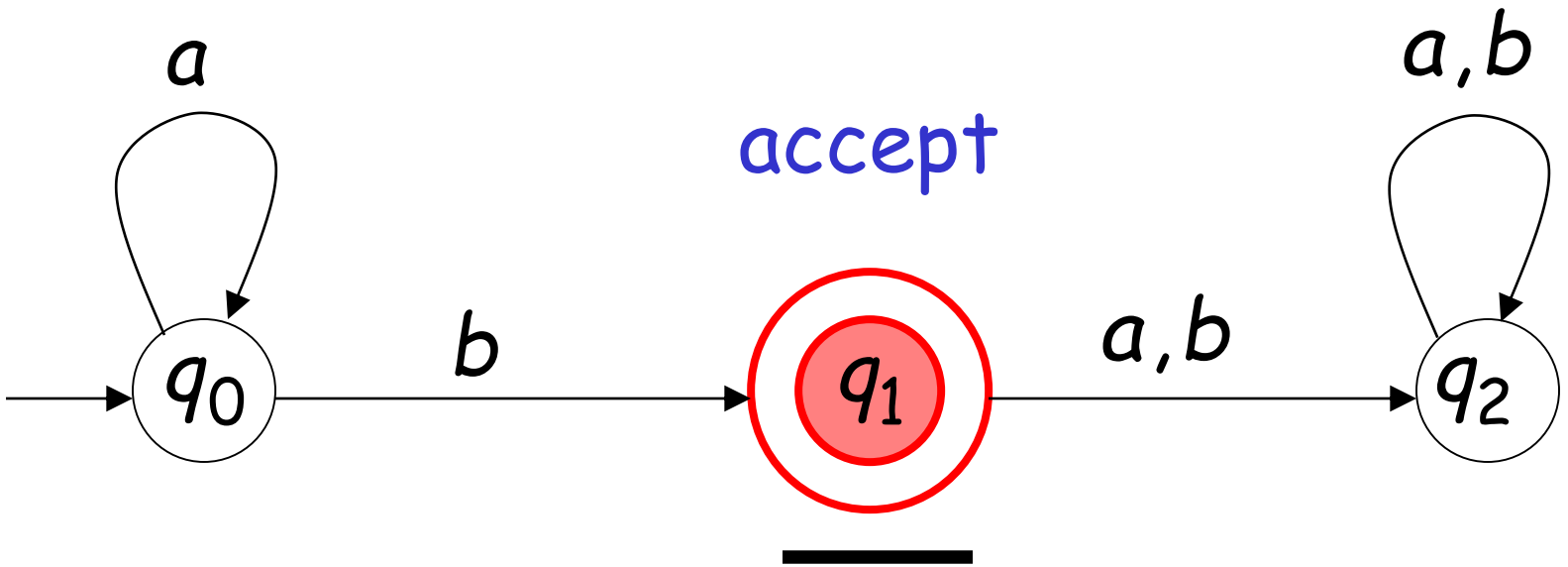

a
a, b
q0
b
q1
a, b
q2


a
a
b

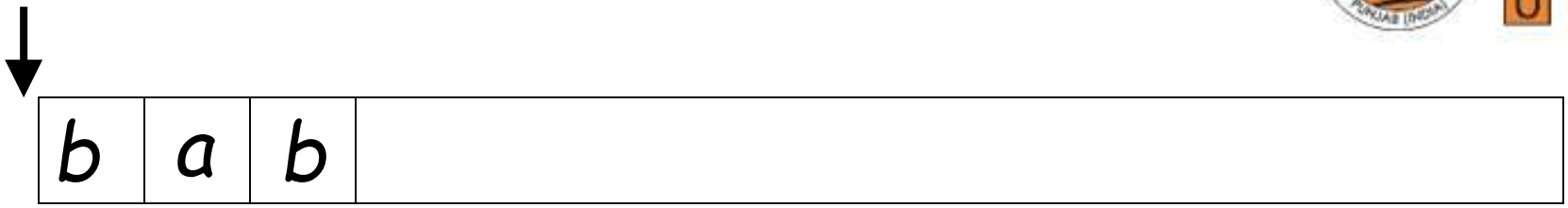

a
a,b
b
a,b
q0
q1
q2

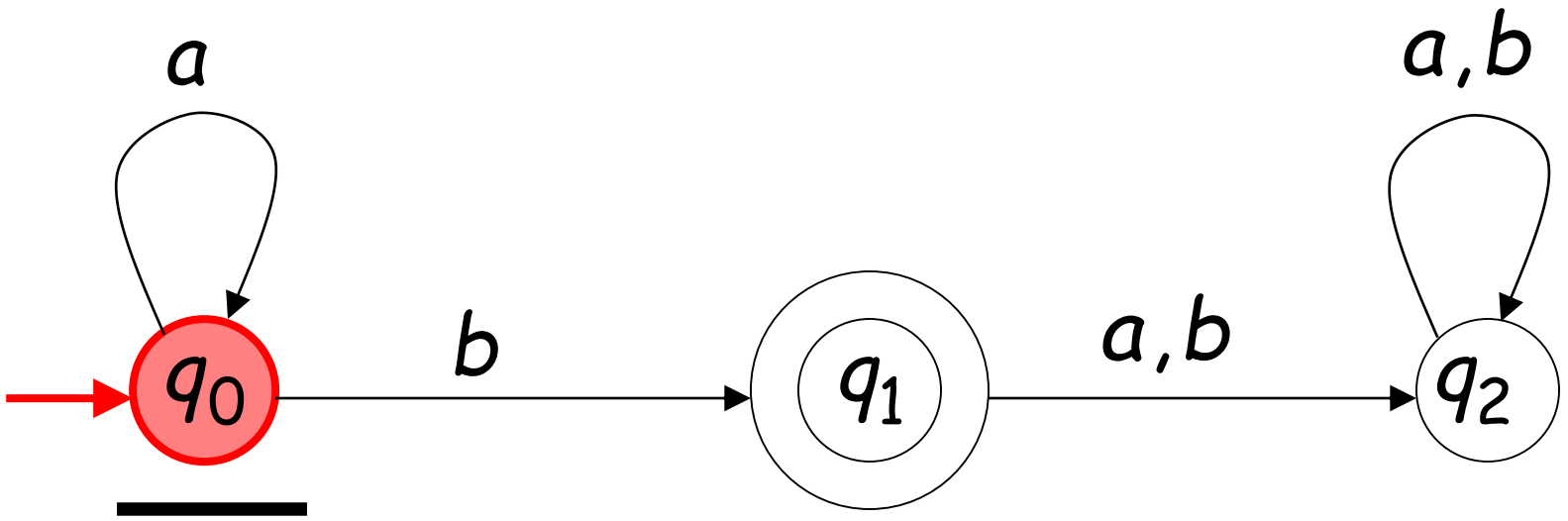
Input finished

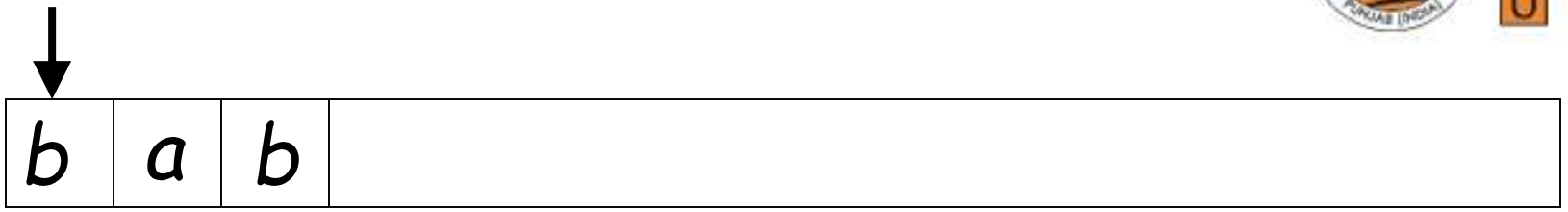


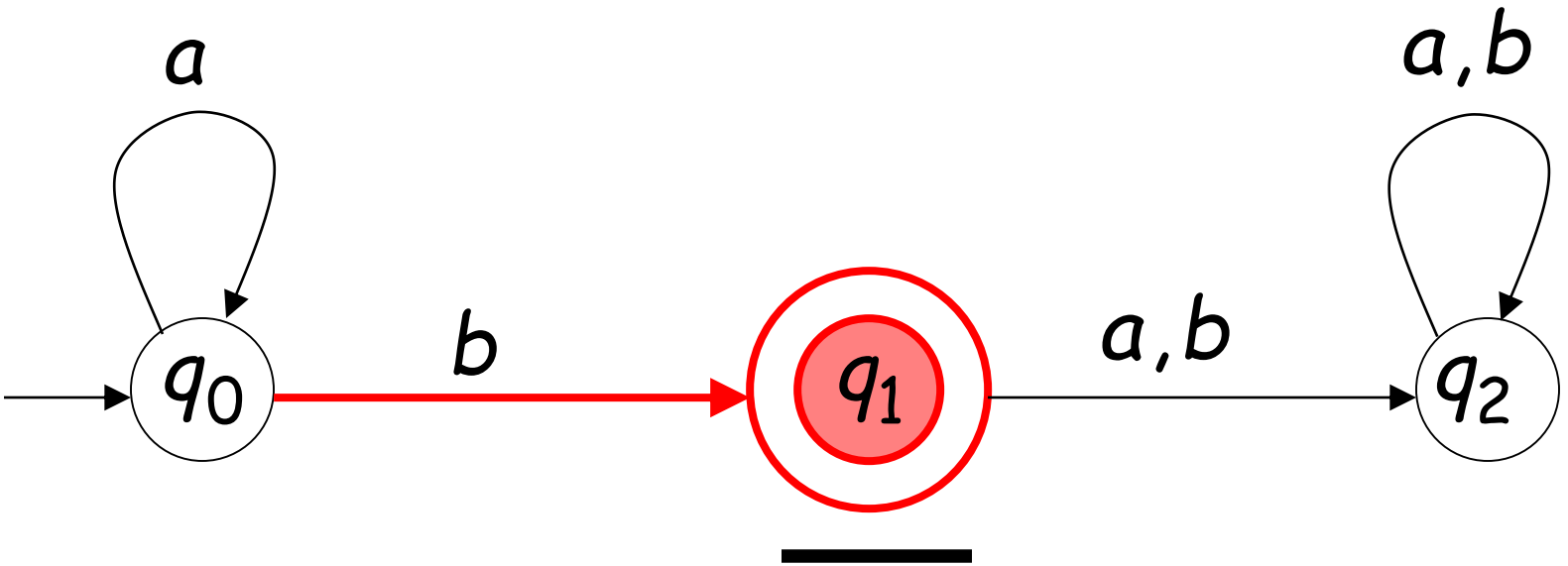


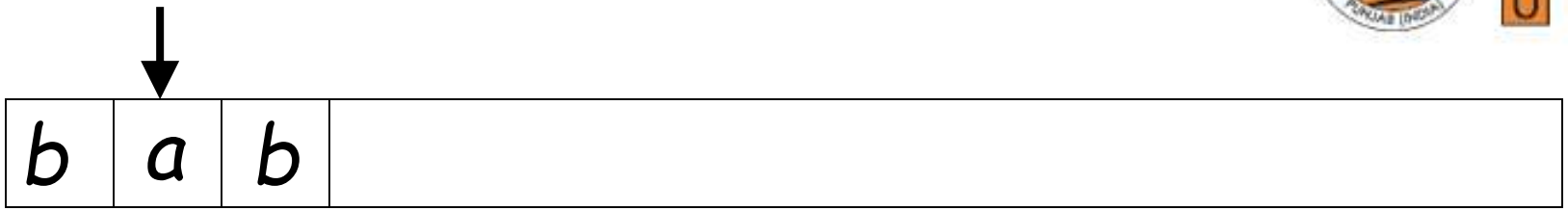
Rejection Example

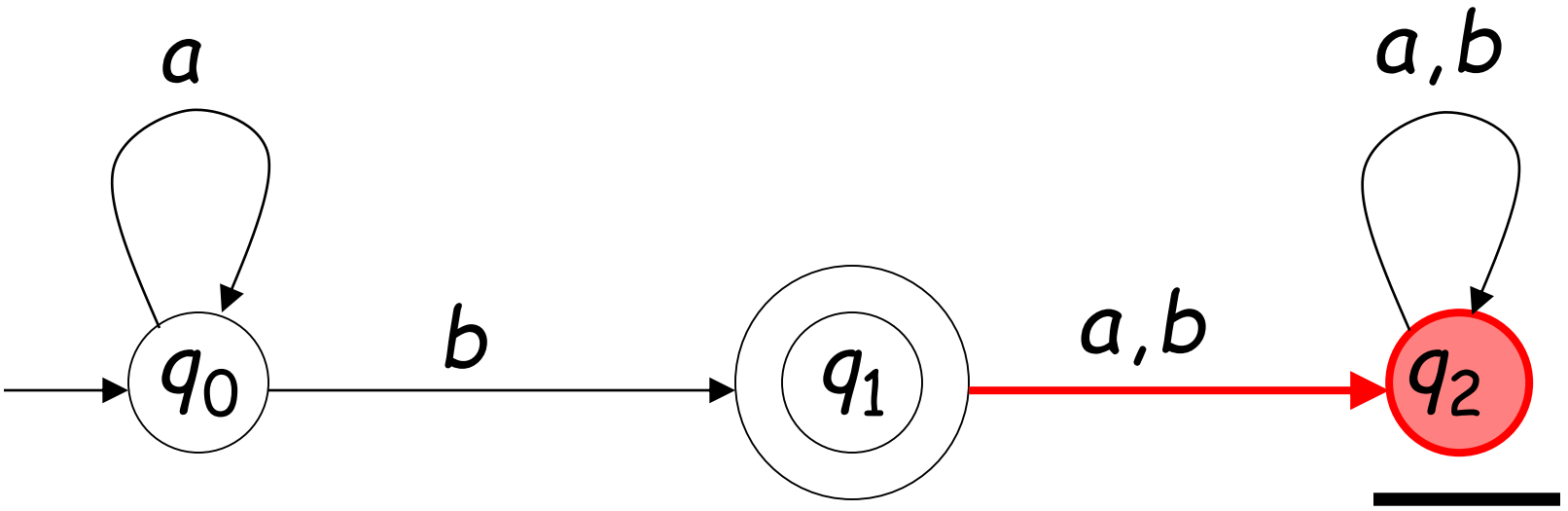


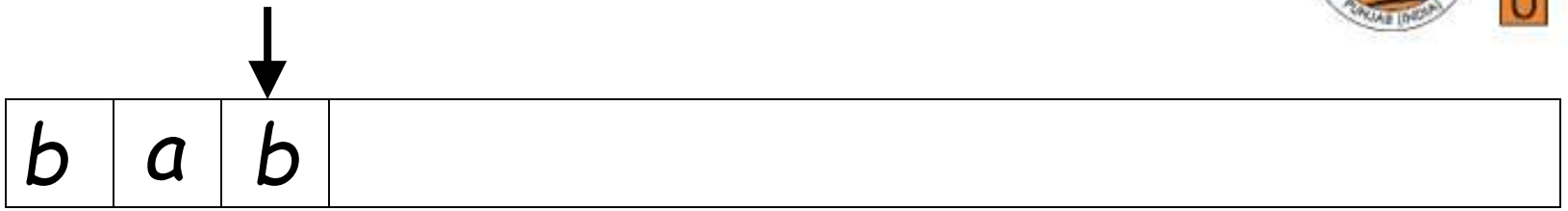


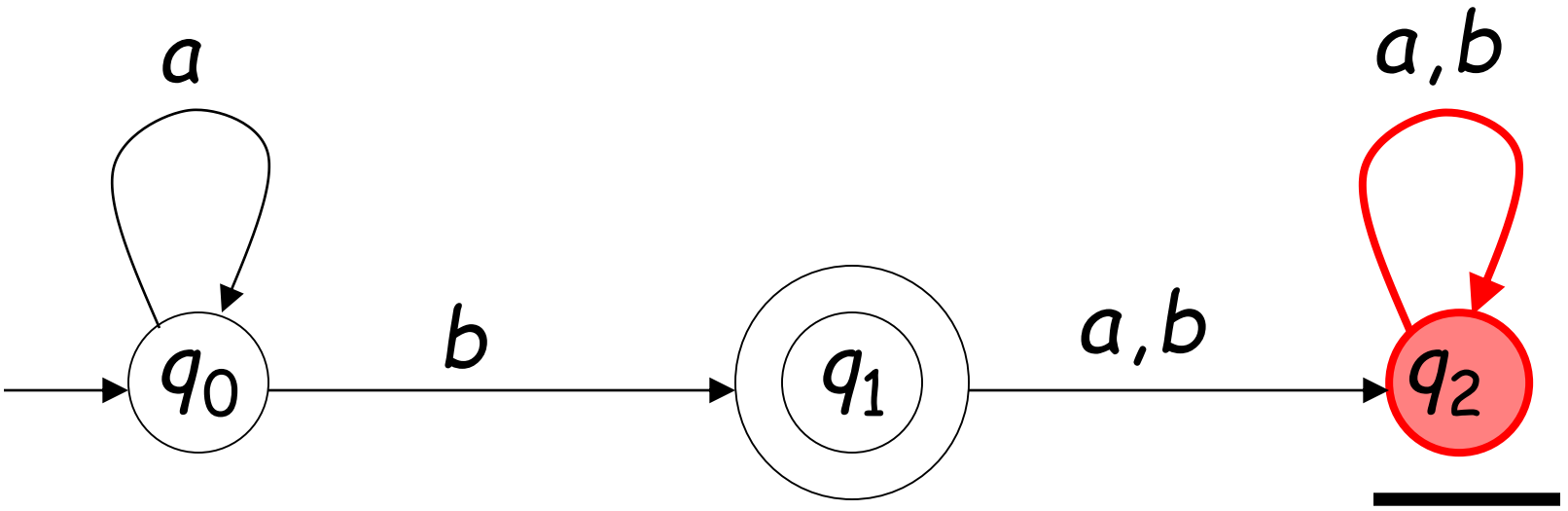

b
a
b


a
a, b
b
a, b
q0
q1
q2

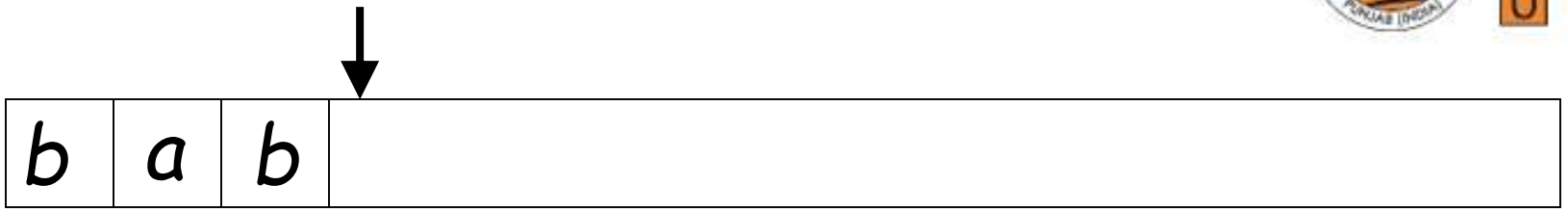

b
a
b

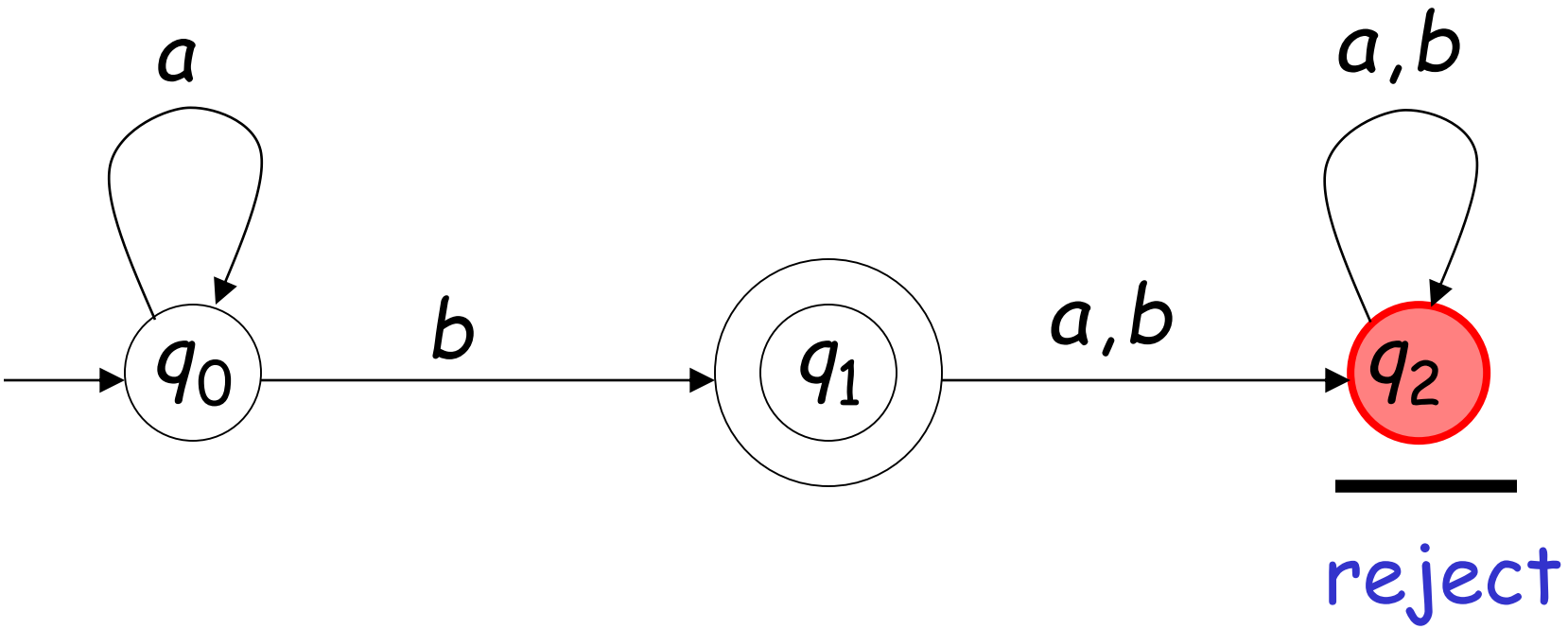

a
a, b
b
a, b
q0
q1
q2


b
a
b


a
a, b
b
a, b
q_0
q_1
q_2

Input finished





Languages Accepted by FAs

FA M

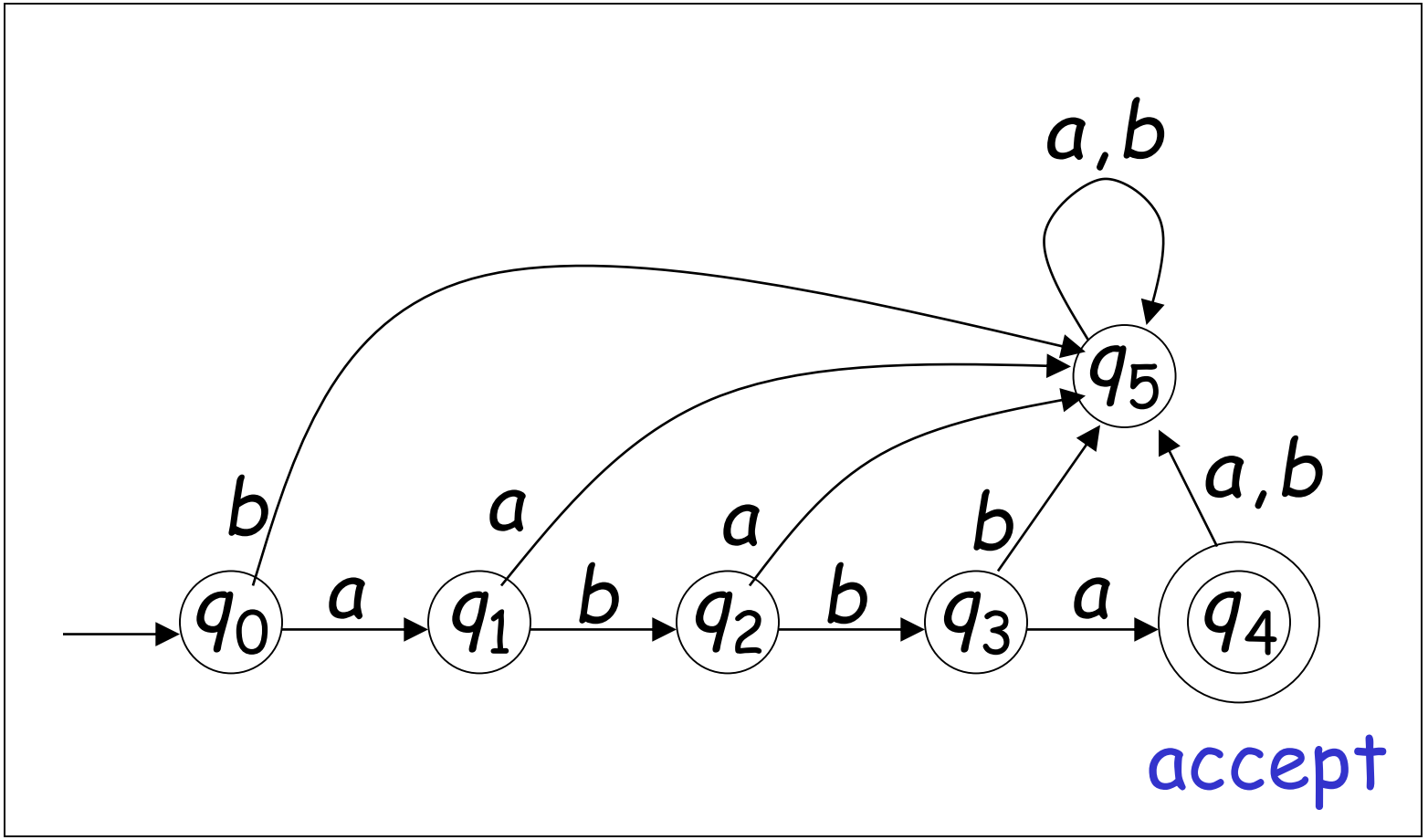
Definition:

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

$L(M)$ = { strings that bring M to an accepting state}

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

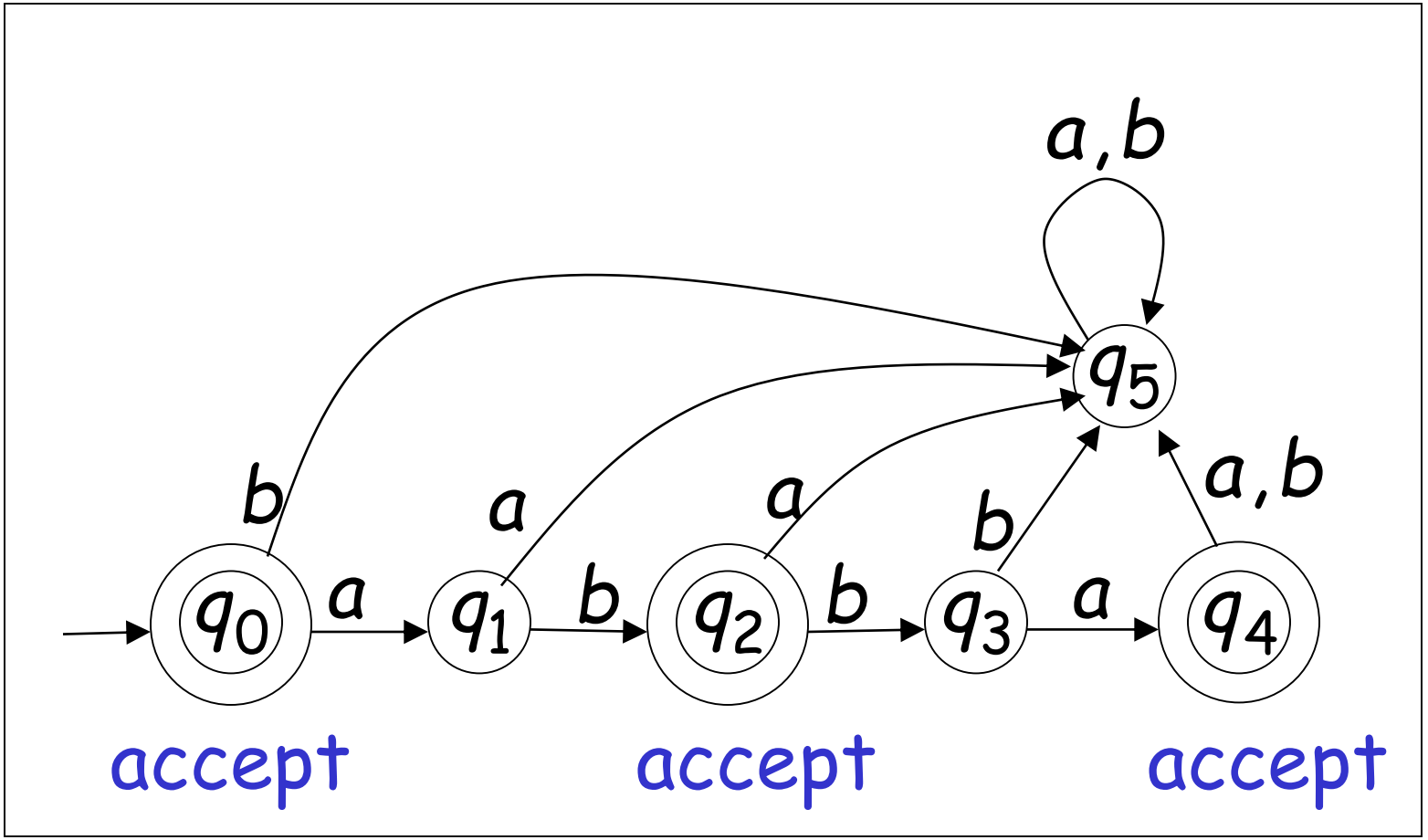
M



Example

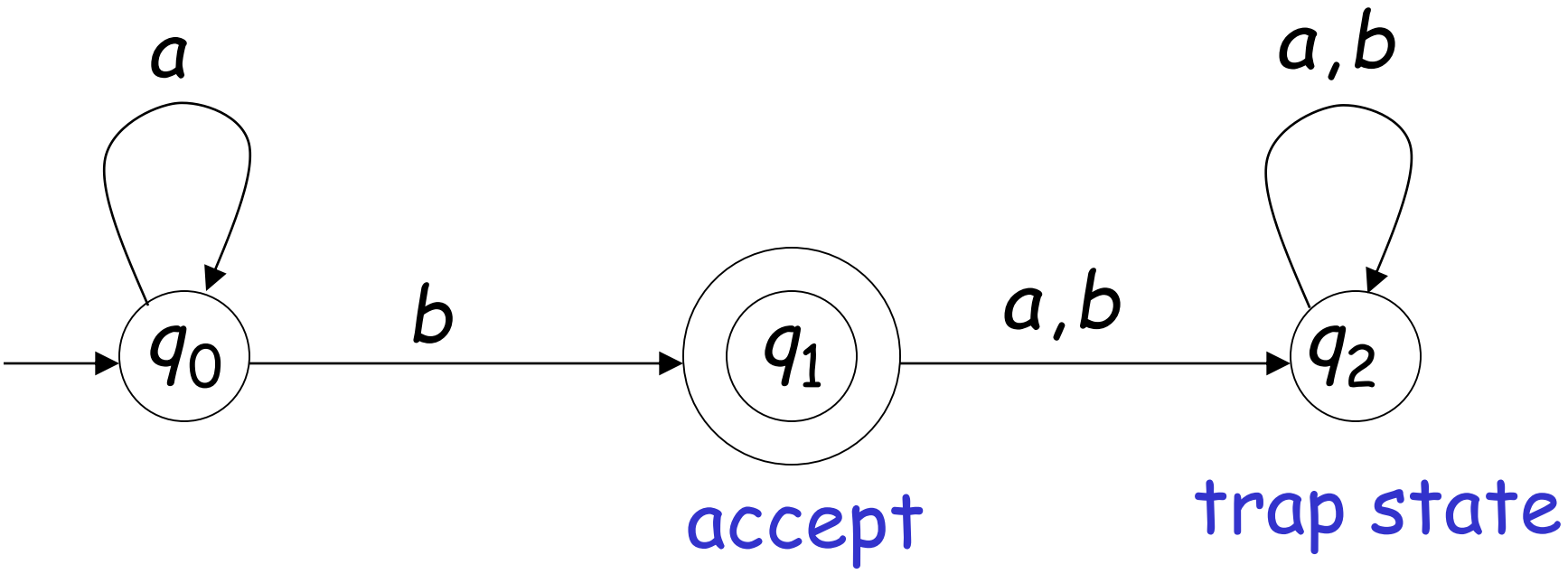
$L(M) = \{\lambda, ab, abba\}$

M



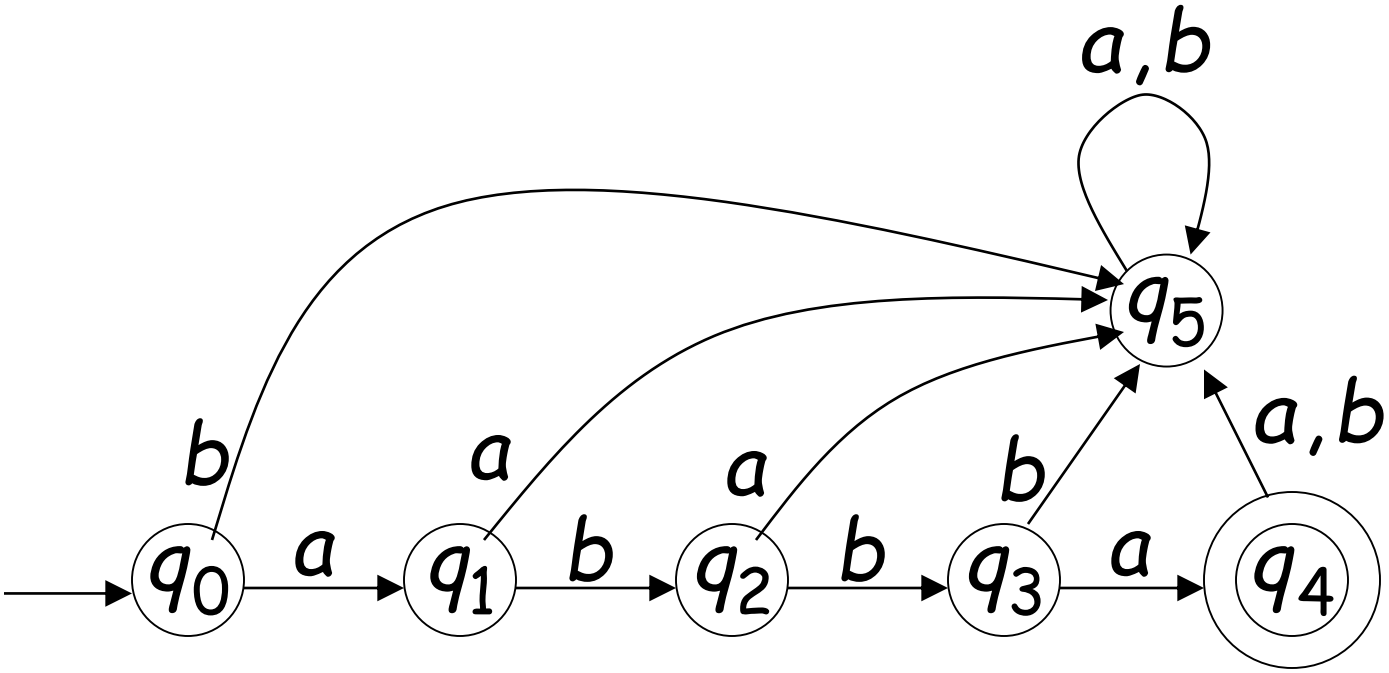
Example

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

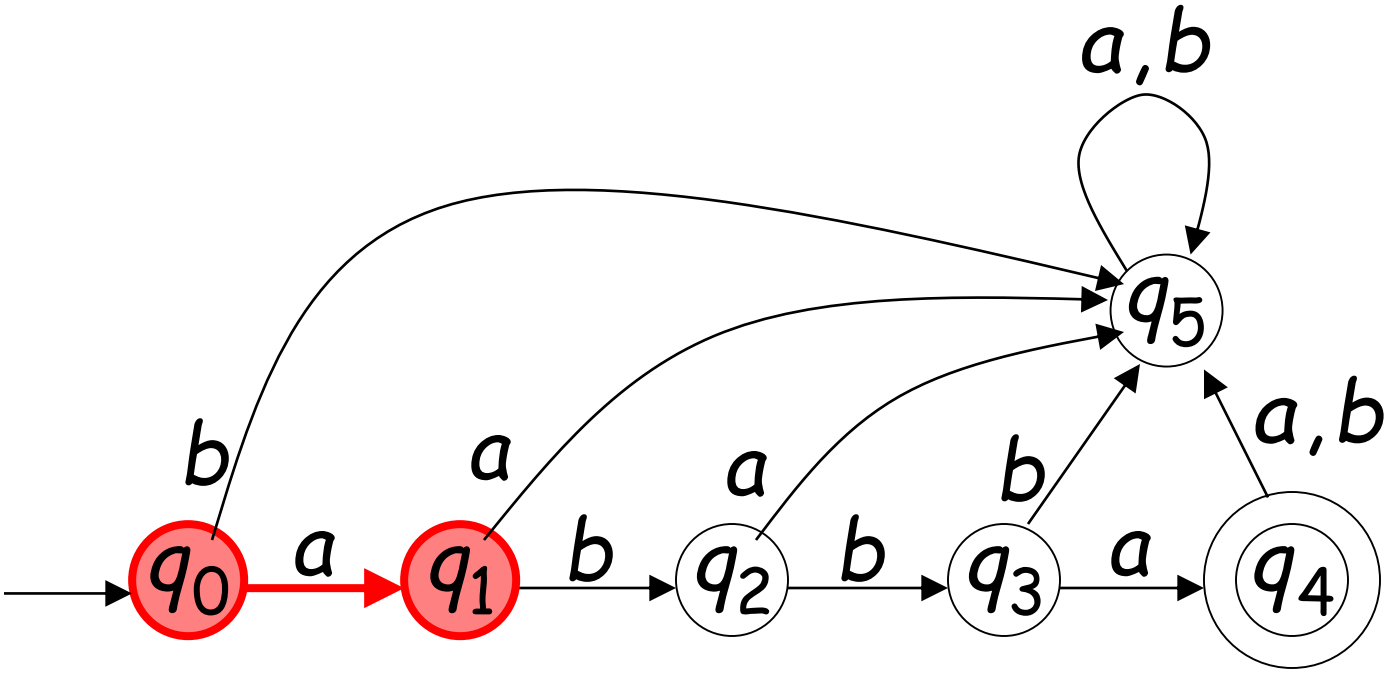


Transition Function δ

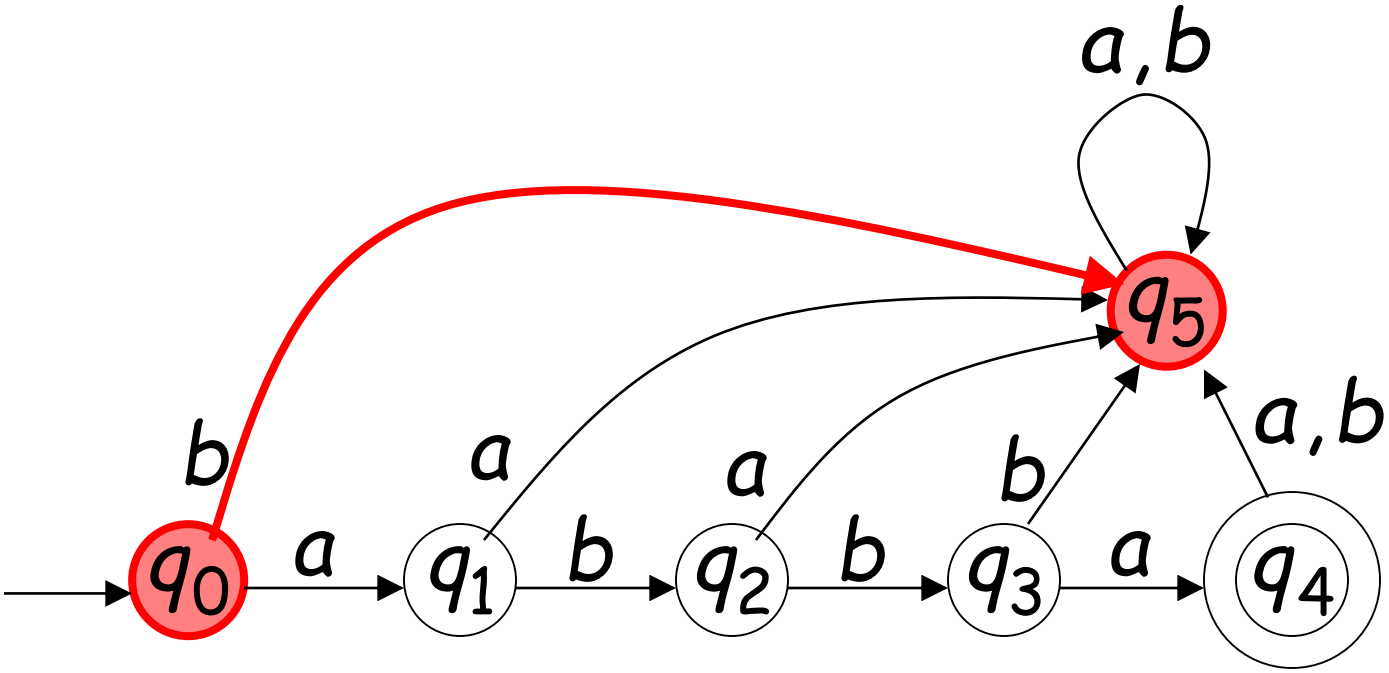
$$\delta : Q \times \Sigma \rightarrow Q$$



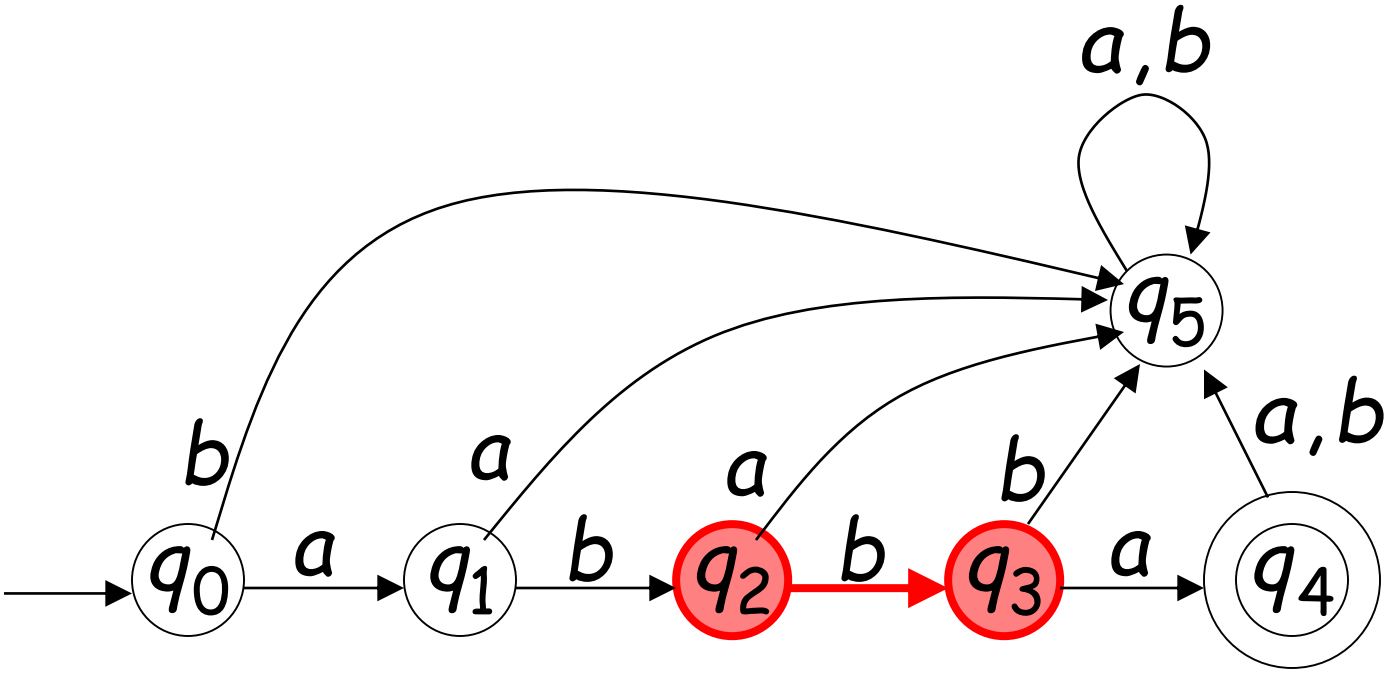
$$\delta(q_0, a) = q_1$$



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

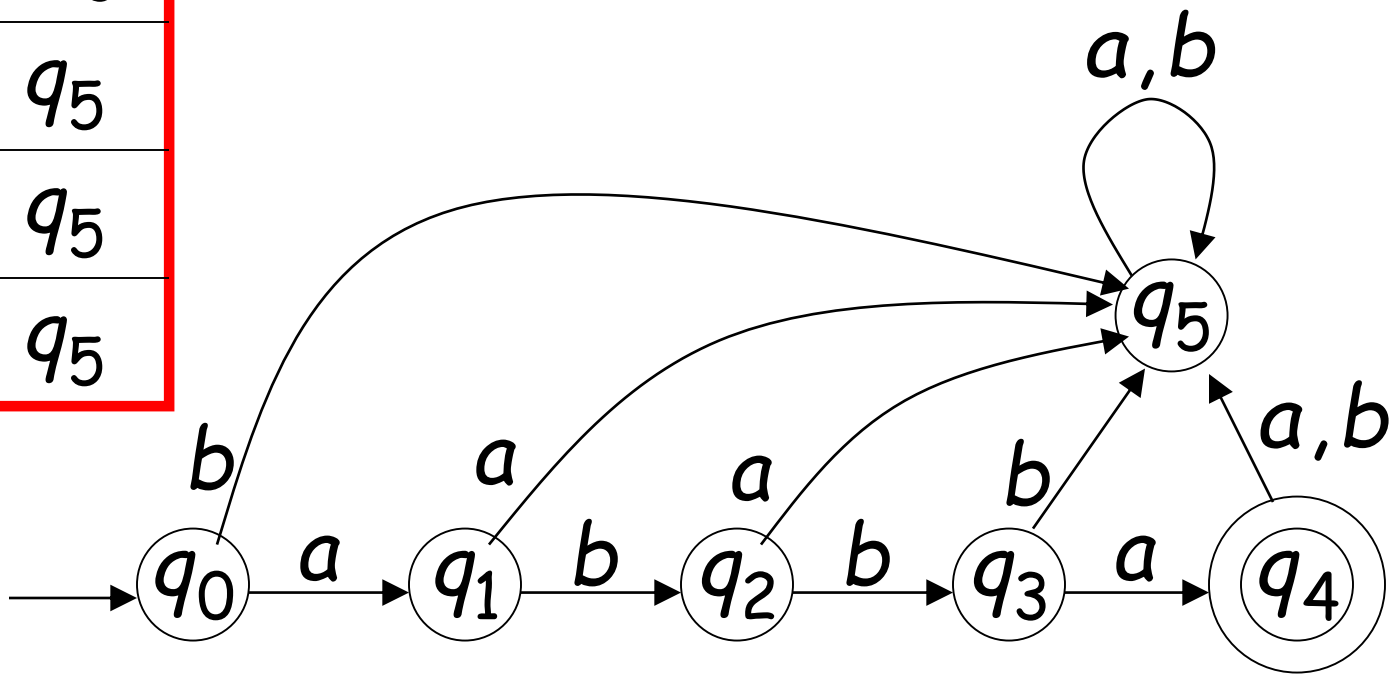


$$\delta(q_2, b) = q_3$$



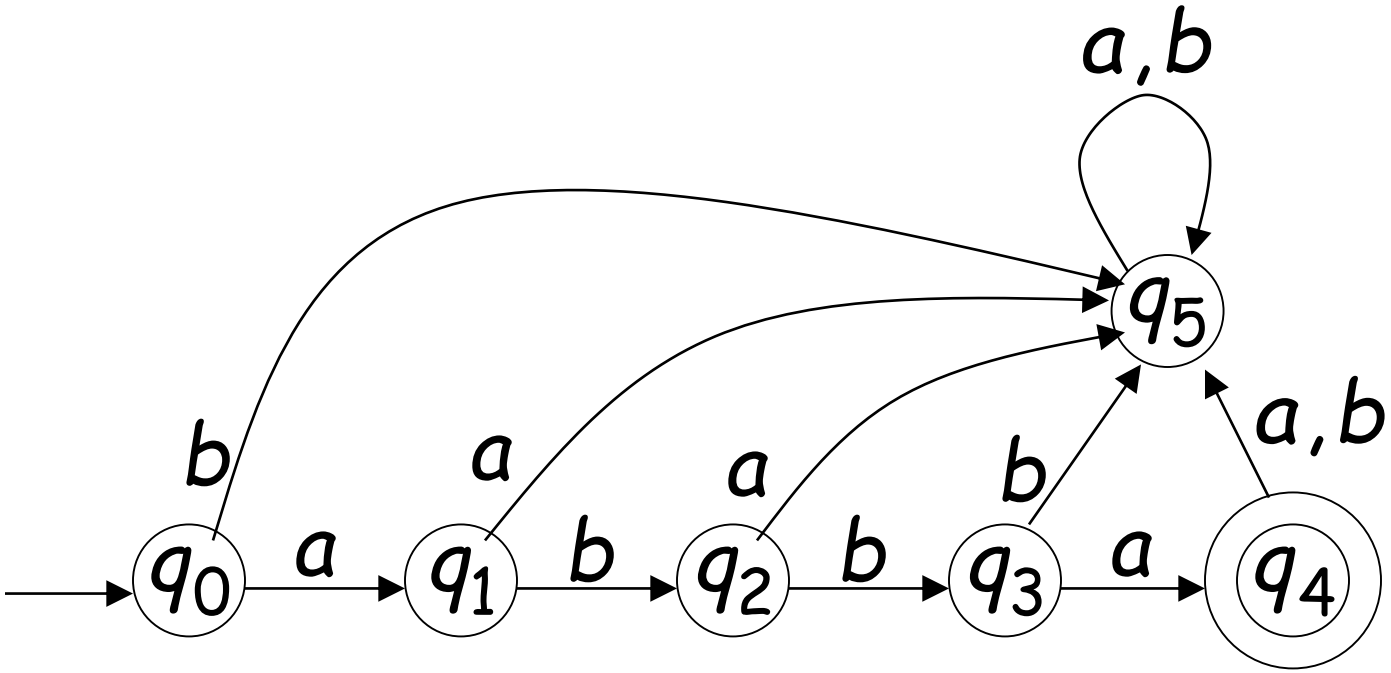
Transition Function δ

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

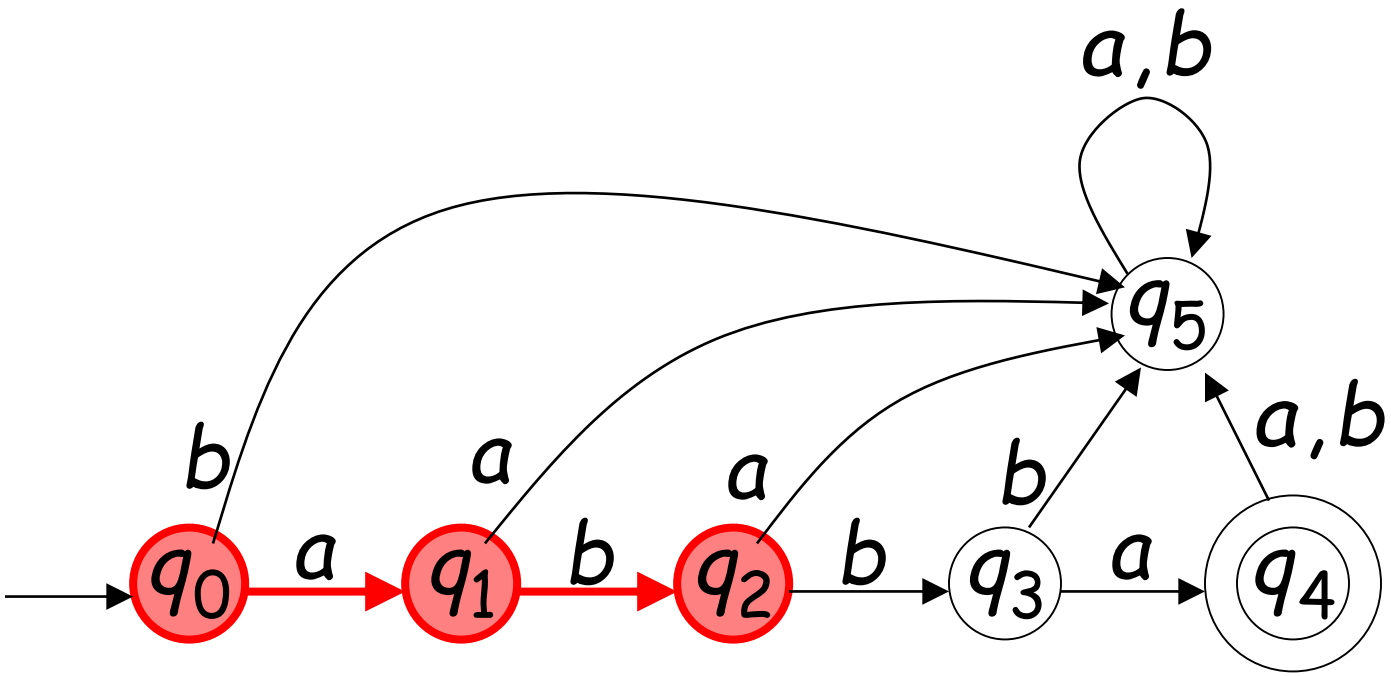


Extended Transition Function δ^*

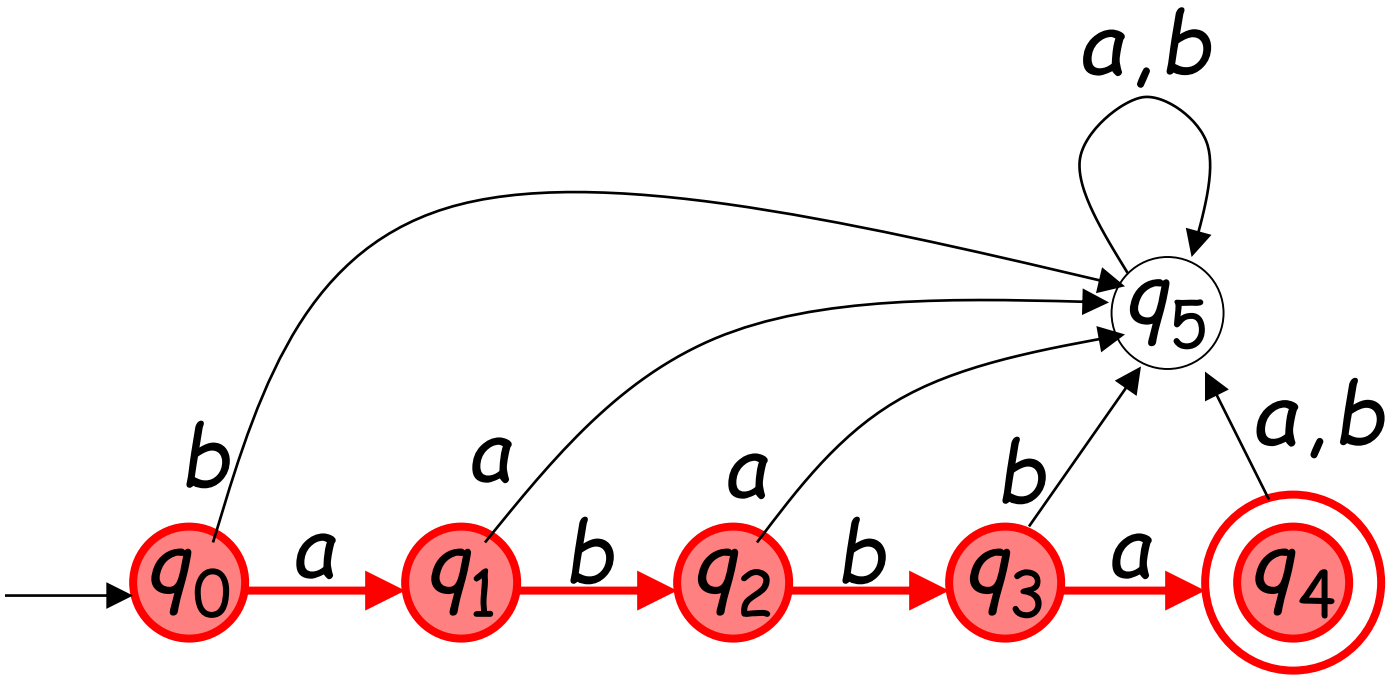
$$\delta^* : Q \times \Sigma^* \rightarrow Q$$



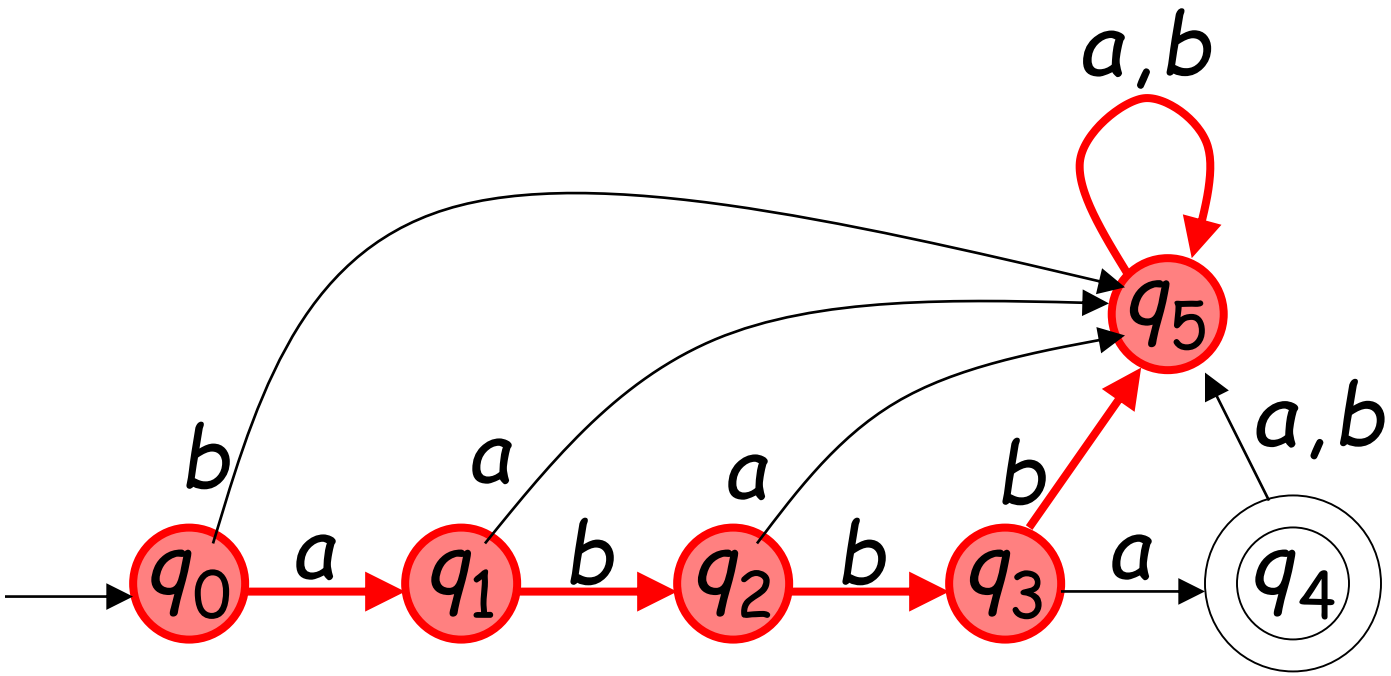
$$\delta^*(q_0, ab) = q_2$$



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



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



Recursive Definition

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

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

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

$\delta(q_1,\sigma) = q'$

$\Rightarrow \delta^*(q,w\sigma) = \delta(q_1,\sigma)$

$\delta^*(q,w) = q_1$

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

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,

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

The symbol ↓ indicates that the current input symbol is being processed by the machine.

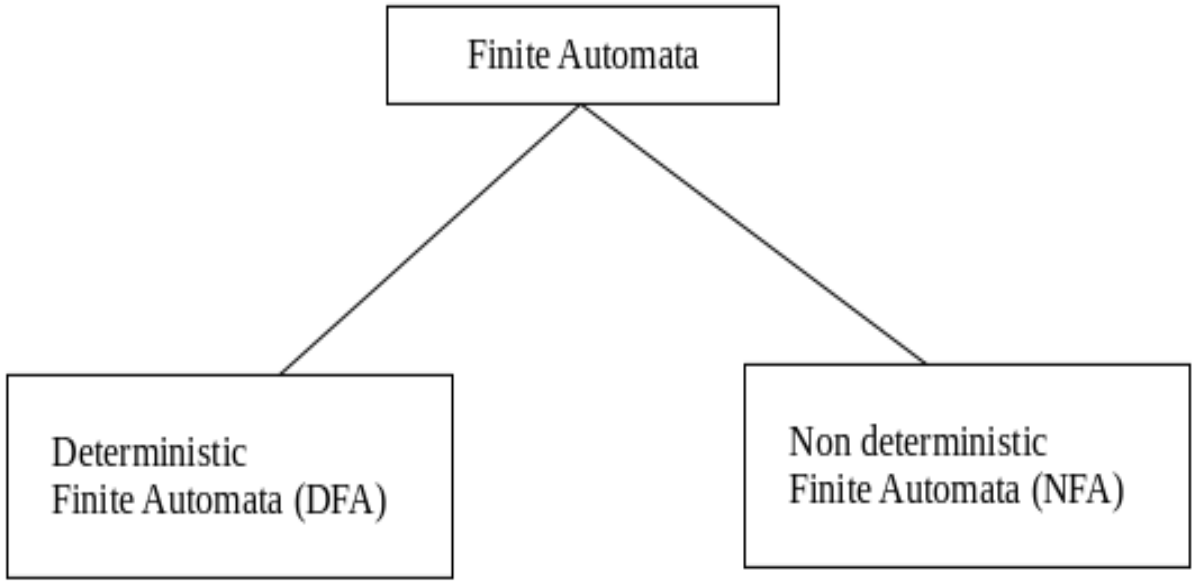
LOVELY
PROFESSIONAL
UNIVERSITY

FINITE AUTOMATA

Types of Automata

- There are two types of finite automata:

1. DFA(deterministic finite automata)
2. NFA(non-deterministic finite automata)



Deterministic Finite Automata(DFA)

- The finite automata are called deterministic finite automata if the m/c is read an i/p string one symbol at a time.
- Deterministic refers to the uniqueness of the computation.
- In the DFA, there is only one path for specific i/p from the current state to the next state.
- DFA does not accept the null move i.e. DFA cannot change state without any i/p character.

- DFA can contain multiple final states.

- It is used in lexical Analysis in compiler.

- Acceptance of languages: Reaching to final state
- DFA to accept zero or more “a”
- L={a,aa,aaa, aaaa,……}

Formal Definition of DFA

- A DFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the alphabet.
- **δ** is the transition function where $\delta: Q \times \sum \rightarrow Q$
- **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
- **F** is a set of final state/states of Q ($F \subseteq Q$).

Graphical Representation of a DFA

- A DFA is represented by digraphs called **state diagram**.
- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

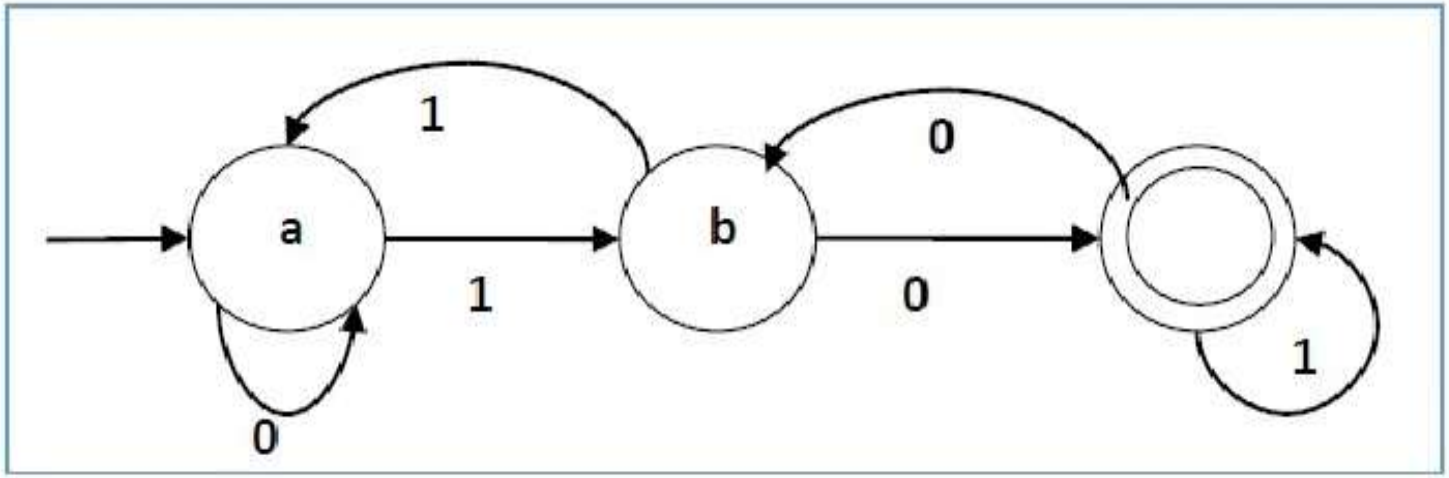
LOVELY PROFESSIONAL UNIVERSITY

- Let a deterministic finite automaton be →
- Q = {a, b, c},
- ∑ = {0, 1},
- q_0 = {a},
- F = {c}, and

- No of States = | Min String |+1

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows –



LOVELY
PROFESSIONAL
UNIVERSITY

Practice Questions on DFA

1. Construct DFA accepting all strings over {0,1} that starts with 1 and ends with 0.

2. Construct DFA accepting all strings over {a,b} ending with ‘ab’.

- 3.Construct DFA accepting all strings over {a,b} ending with ‘abb’.

LOVELY PROFESSIONAL UNIVERSITY

Non Deterministic Finite Automata(NDFA)

- The FA are called NFA, when there exist many paths for specific i/p from the current state to the next state.
- It is easy to construct NFA than DFA for a given regular language.
- Every NFA is not DFA, but each NFA can be translated into DFA.

- NFA is defined in the same way as DFA but with two exceptions:

1. It contain multiple next state

2. It contains ∈ transitions.

Formal Definition of an NDFA

- An NDFA can be represented by a 5-tuple $(Q, \sum, \delta, q_0, F)$ where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the alphabets.
- **δ** is the transition function where $\delta: Q \times \sum \rightarrow 2^Q$
- (Here the power set of Q (2^Q) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states)
- **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
- **F** is a set of final state/states of Q ($F \subseteq Q$).

Graphical Representation of an NDFA: (same as DFA)

- An NDFA is represented by digraphs called **state diagram.**
- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

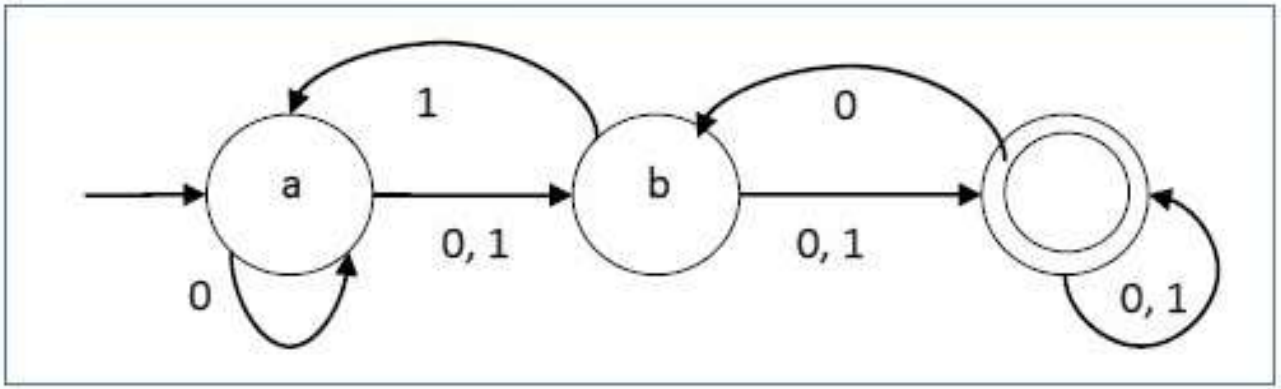
LOVELY PROFESSIONAL UNIVERSITY

Example

- Let a non-deterministic finite automaton be →
- Q = {a, b, c}
- ∑ = {0, 1}
- q_0 = {a}
- F = {c}

Present State	Next State for Input 0	Next State for Input 1
a	a, b	b
b	c	a, c
c	b, c	c

Its graphical representation would be as follows −



DFA vs NDFA

LOVELY PROFESSIONAL UNIVERSITY

DFA	NDFA
The transition from a state is to a single particular next state for each input symbol. Hence it is called *deterministic*.	The transition from a state can be to multiple next states for each input symbol. Hence it is called *non-deterministic*.
Empty string transitions are not seen in DFA.	NDFA permits empty string transitions.
Backtracking is allowed in DFA	In NDFA, backtracking is not always possible.
Requires more space.	Requires less space.
A string is accepted by a DFA, if it transits to a final state.	A string is accepted by a NDFA, if at least one of all possible transitions ends in a final state.

- Practice Questions on NFA

Question 1: Design NFA for given transition table.

State	0	1
-> q0	q0, q1	q0, q2
q1	q3	∈
q2	q2, q3	q3
* q3	q3	q3

Question 2: Construct NFA , L={Set of all strings that starts with 0} with ∑ = {0, 1}.

Question 3: Construct NFA with ∑ = {0, 1} that accepts all string with 01.

Question 4: Construct NFA with ∑ = {0, 1} that accepts all string of length atleast 2.

Question 5: Design an NFA with ∑ = {0, 1} that accepts all strings ending with 01.

Question 6: Construct NFA with ∑ = {0, 1} in which double '1' is followed by double '0'.

Question 7: Construct NFA in which all the string contain a substring 1110.

Question 8: Design an NFA with ∑ = {0, 1} that accepts all string in which the third symbol from the right end is always 0.

Equivalence of DFA and NFA: Conversion from NFA to DFA

- In NFA, when a specific input is given to the current state, the machine goes to multiple states.

- It can have zero, one or more than one move on a given input symbol.

- On the other hand, in DFA, when a specific input is given to the current state, the machine goes to only one state.
-
- DFA has only one move on a given input symbol.

- Let, $M = (Q, \sum, \delta, q0, F)$ is an NFA which accepts the language L(M).

- There should be equivalent DFA denoted by $M' = (Q', \sum', q0', \delta', F')$ such that $L(M) = L(M')$.

Steps for converting NFA to DFA:

- **Step 1:** Initially Q' = φ
- **Step 2:** Add q0 of NFA to Q'. Then find the transitions from this start state.
- **Step 3:** In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.
- **Step 4:** In DFA, the final state will be all the states which contain F(final states of NFA)

Finite automata: Mealy and Moore Machines

LOVELY PROFESSIONAL UNIVERSITY

- Finite automata may have outputs corresponding to each transition.

- There are two types of finite state machines that generate output –

- Mealy Machine
- Moore machine

Mealy Machine

LOVELY PROFESSIONAL UNIVERSITY

- A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

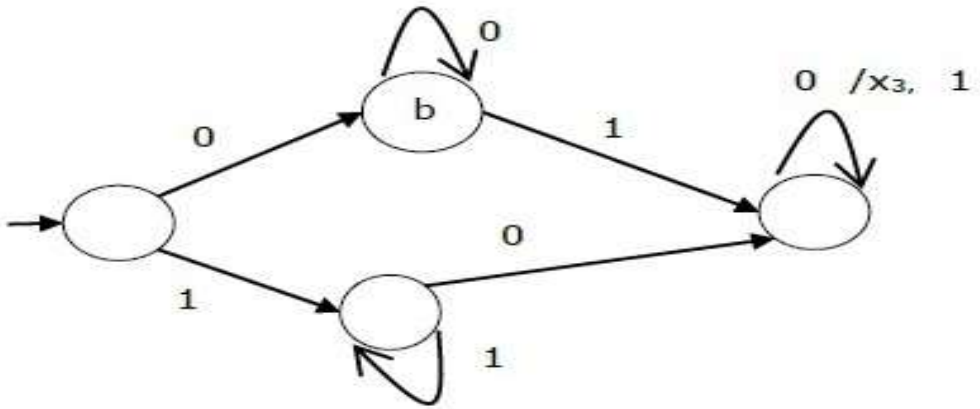
- It can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where δ: Q × ∑ → Q
- **X** is the output transition function where X: Q × ∑ → O
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).

Example

The state table of a Mealy Machine is shown below –

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2

The state diagram of the above Mealy Machine is –



Practice Questions

LOVELY PROFESSIONAL UNIVERSITY

1. Construct Mealy m/c that produce 1's complement of any binary i/p string.

2. Construct a mealy m/c that points 'a' whenever the sequence '01' is encountered in any i/p binary string.

3. Construct a mealy m/c accepting language consisting of strings where {a,b} and the string should end with either aa or bb.

Moore Machine

LOVELY PROFESSIONAL UNIVERSITY

- Moore machine is an FSM whose outputs depend on only the present state.

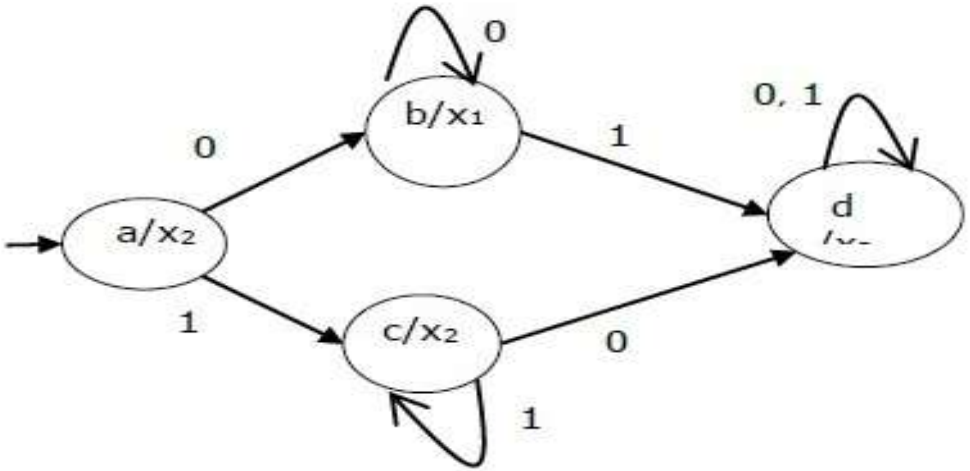
- A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where δ: Q × ∑ → Q
- **X** is the output transition function where X: Q → O
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).

Example

The state table of a Moore Machine is shown below –

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x_2
b	b	d	x_1
c	c	d	x_2
d	d	d	x_3

The state diagram of the above Moore Machine is –



Practice Questions

LOVELY PROFESSIONAL UNIVERSITY

1. Construct Moore m/c that produce 1's complement of any binary i/p string.

2. Design a moore m/c for a binary i/p sequence such that if it has a substring 101, the m/c o/p is A, if the i/p has substring 110, its o/p is B, otherwise o/p is C.

3. Design moore m/c, where i/p alphabet is ∑={a,b}, and the o/p alphabet is O={0,1}. Run the following i/p sequence and find the respective o/p:

(i) aabab (ii) abbb (iii) ababb

State transition table is given as:

States	a	b	o/p
Q0	Q1	Q2	0
Q1	Q2	Q3	0
Q2	Q3	Q4	1
Q3	Q4	Q4	0
Q4	Q0	Q0	0

Conversion from Mealy machine to Moore Machine

- In Moore machine, the output is associated with every state, and in Mealy machine, the output is given along the edge with input symbol.

- To convert Moore machine to Mealy machine, state output symbols are distributed to input symbol paths.

- But while converting the Mealy machine to Moore machine, we will create a separate state for every new output symbol and according to incoming and outgoing edges are distributed.

- **Step 1:** For each state(Qi), calculate the number of different outputs that are available in the transition table of the Mealy machine.

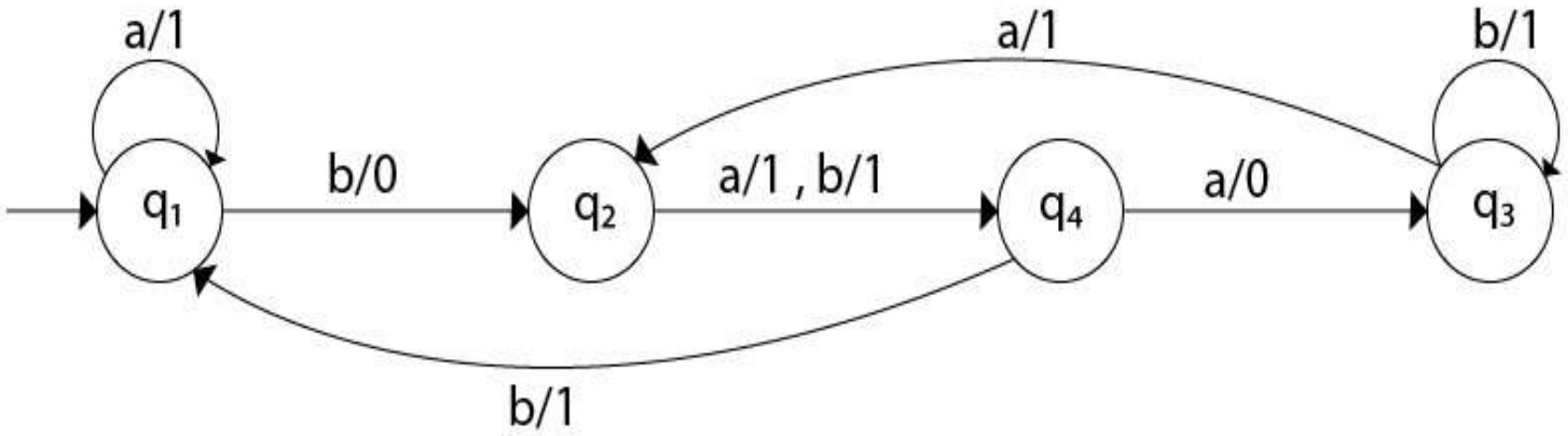
- **Step 2:** Copy state Qi, if all the outputs of Qi are the same. Break qi into n states as Qin, if it has n distinct outputs where n = 0, 1, 2....

- **Step 3:** If the output of initial state is 0, insert a new initial state at the starting which gives 1 output.

Practice Questions

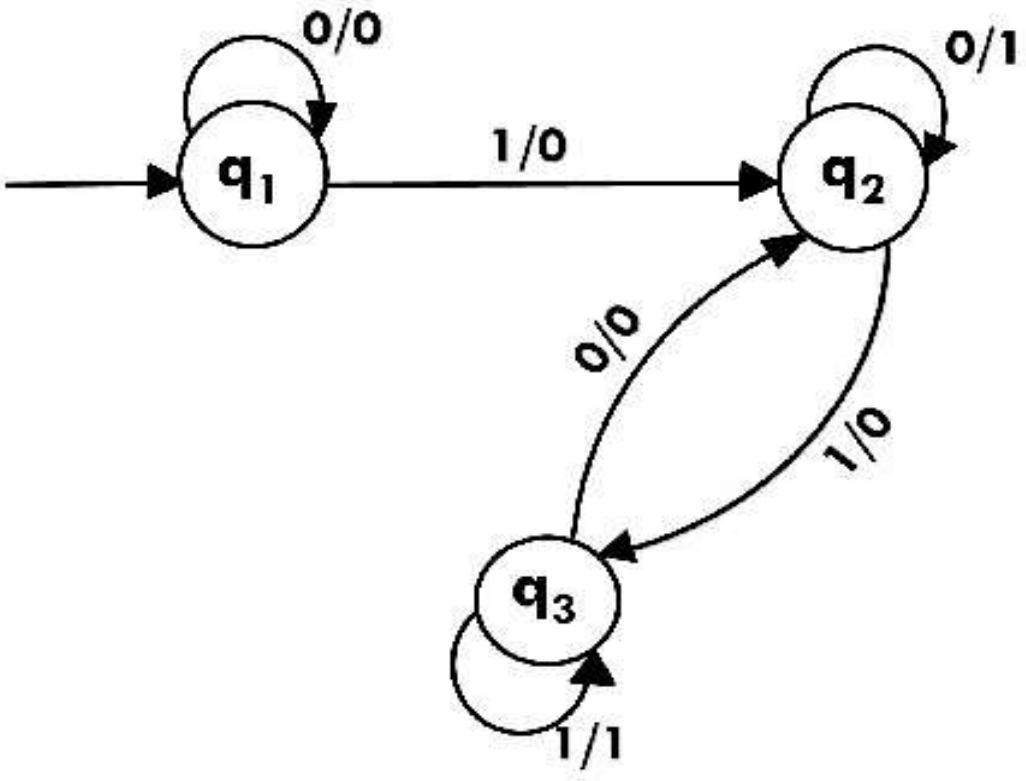
Que:1

Convert the following Mealy machine into equivalent Moore machine.



- Ques:2

Convert the following Mealy machine into equivalent Moore machine.



Conversion from Moore machine to Mealy Machine

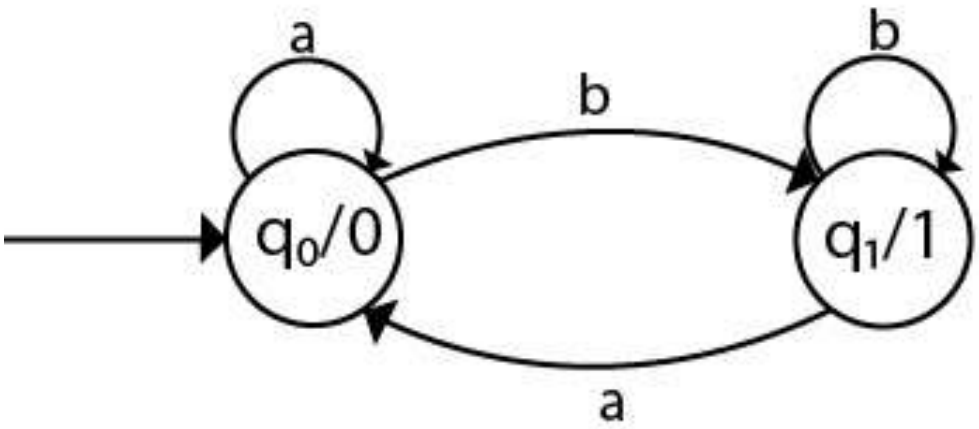
- We cannot directly convert Moore machine to its equivalent Mealy machine because the length of the Moore machine is one longer than the Mealy machine for the given input.

- To convert Moore machine to Mealy machine, state output symbols are distributed into input symbol paths.

Practice Questions

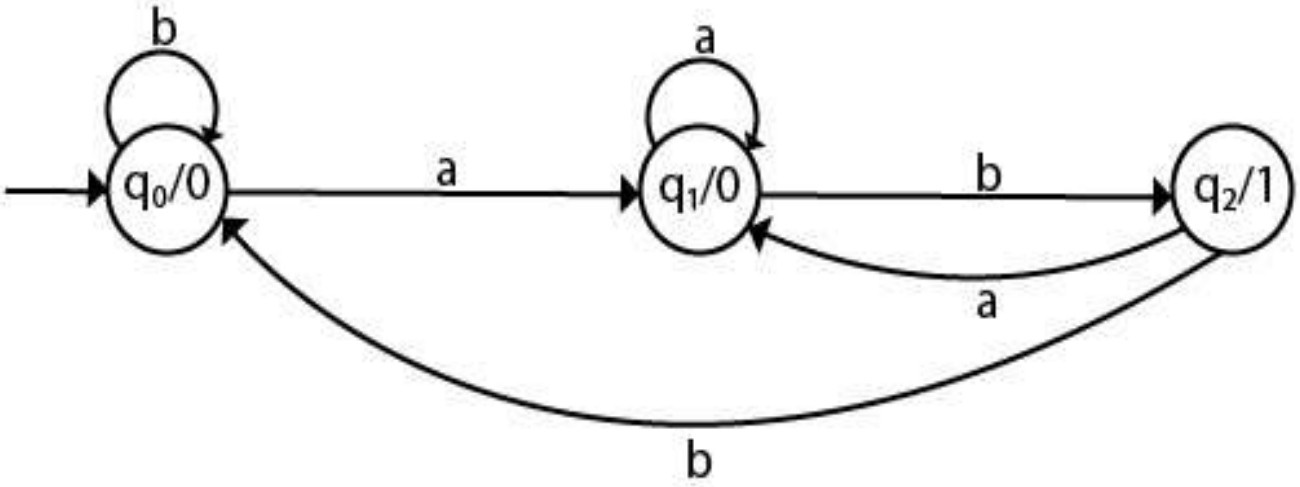
- Ques:1

Convert the following Moore machine into its equivalent Mealy machine.



- Ques:2

Convert the given Moore machine into its equivalent Mealy machine.



- Ques:3

Convert the given Moore machine into its equivalent Mealy machine.

Q	a	b	Output(λ)
q0	q0	q1	0
q1	q2	q0	1
q2	q1	q2	2

- Ques:4 The moore m/c counts the occurrences of sequence ‘abb’ in any i/p binary strings over {a,b}. Convert it into its equivalent mealy m/c.

LOVELY PROFESSIONAL UNIVERSITY

Minimization of DFA Equivalence Theorem

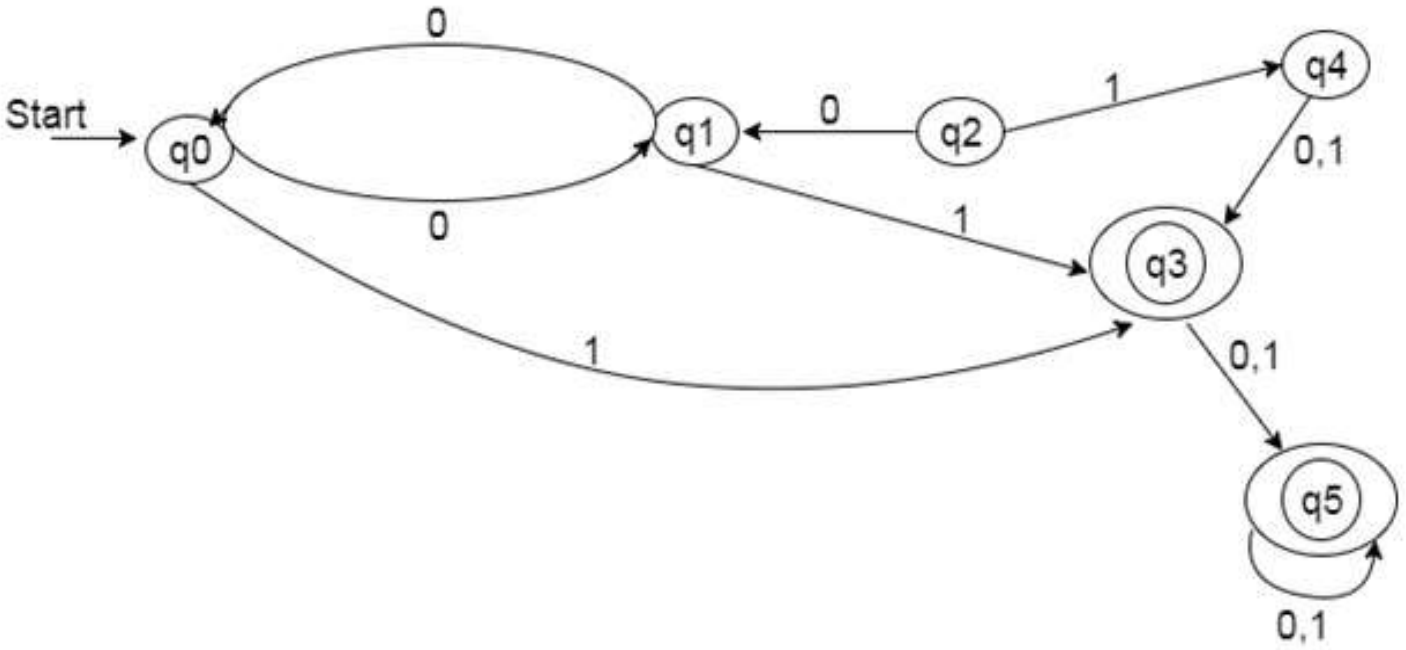
- Minimization of DFA means reducing the number of states from given FA.

- **Step 1:** Remove all the states that are unreachable from the initial state via any set of the transition of DFA.
- **Step 2:** Draw the transition table for all pair of states.
- **Step 3:** Now split the transition table into two tables T1 and T2. T1 contains all final states, and T2 contains non-final states.

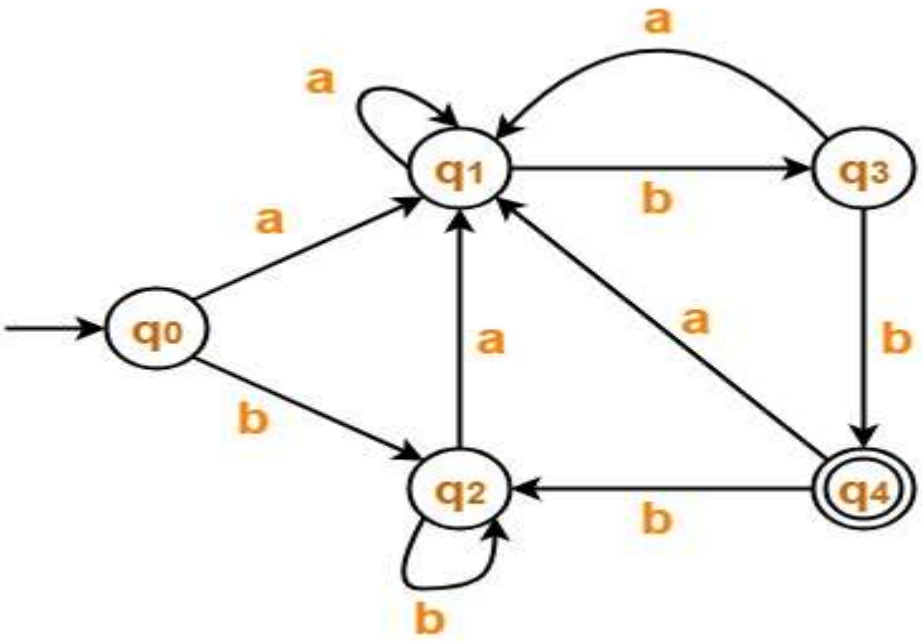
- **Step 4:** Find similar rows from T1 such that:
    1. $\delta(q, a) = p$
    2. $\delta(r, a) = p$
- That means, find the two states which have the same value of a and b and remove one of them.
- **Step 5:** Repeat step 3 until we find no similar rows available in the transition table T1.
- **Step 6:** Repeat step 3 and step 4 for table T2 also.
- **Step 7:** Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Practice Questions

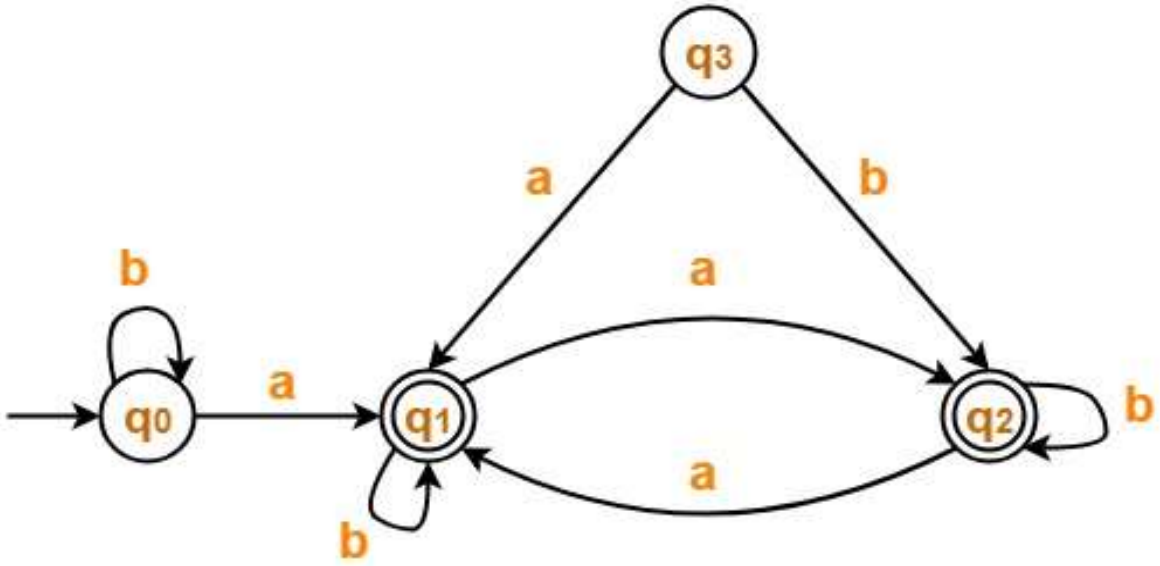
- Ques:1 Minimize the given DFA:



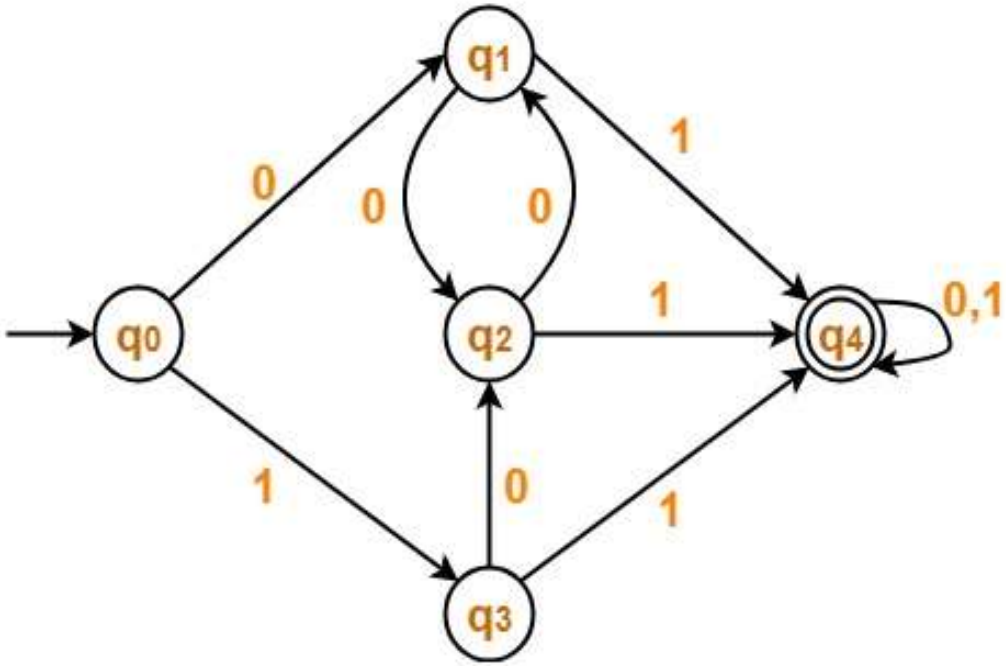
- Ques:2 Minimize the given DFA:



- Ques:3 Minimize the given DFA:



- Ques:4 Minimize the given DFA:



Minimization of DFA
Table Filling Method
(MyHill Nerode Theorem)

- Suppose there is a DFA D < Q, Σ, q0, δ, F > which recognizes a language L. Then the minimized DFA D < Q', Σ, q0, δ', F' > can be constructed for language L as:

Step 1: We will divide Q (set of states) into two sets. One set will contain all final states and other set will contain non-final states. This partition is called P_0.

Step 2: Initialize k = 1

- **Step 3:** Find P_k by partitioning the different sets of P_{k-1}. In each set of P_{k-1}, we will take all possible pair of states. If two states of a set are distinguishable, we will split the sets into different sets in P_k.

  Step 4: Stop when $P_k = P_{k-1}$ (No change in partition)

  Step 5: All states of one set are merged into one. No. of states in minimized DFA will be equal to no. of sets in P_k.

- **How to find whether two states in partition P_k are distinguishable ?**

  Two states (qi, qj) are distinguishable in partition P_k if for any input symbol a, δ (qi, a) and δ (qj, a) are in different sets in partition P_{k-1}.

POLLING QUESTIONS

1. Number of states require to accept string ends with 10.

   A) 3
   B) 2
   C) 1
   D) can’t be represented.