Computing Theory

COMP 147 (4 units)

Chapter 1: Regular Languages
Section 1.1: Finite Automata

Course Segments

- Automata and Languages
  - How can we define abstract models of computers?
- Computability Theory
  - What can (or cannot) be computed?
- Complexity Theory
  - What makes some problems computationally difficult?

Computational Models

We'll look at three computational models:

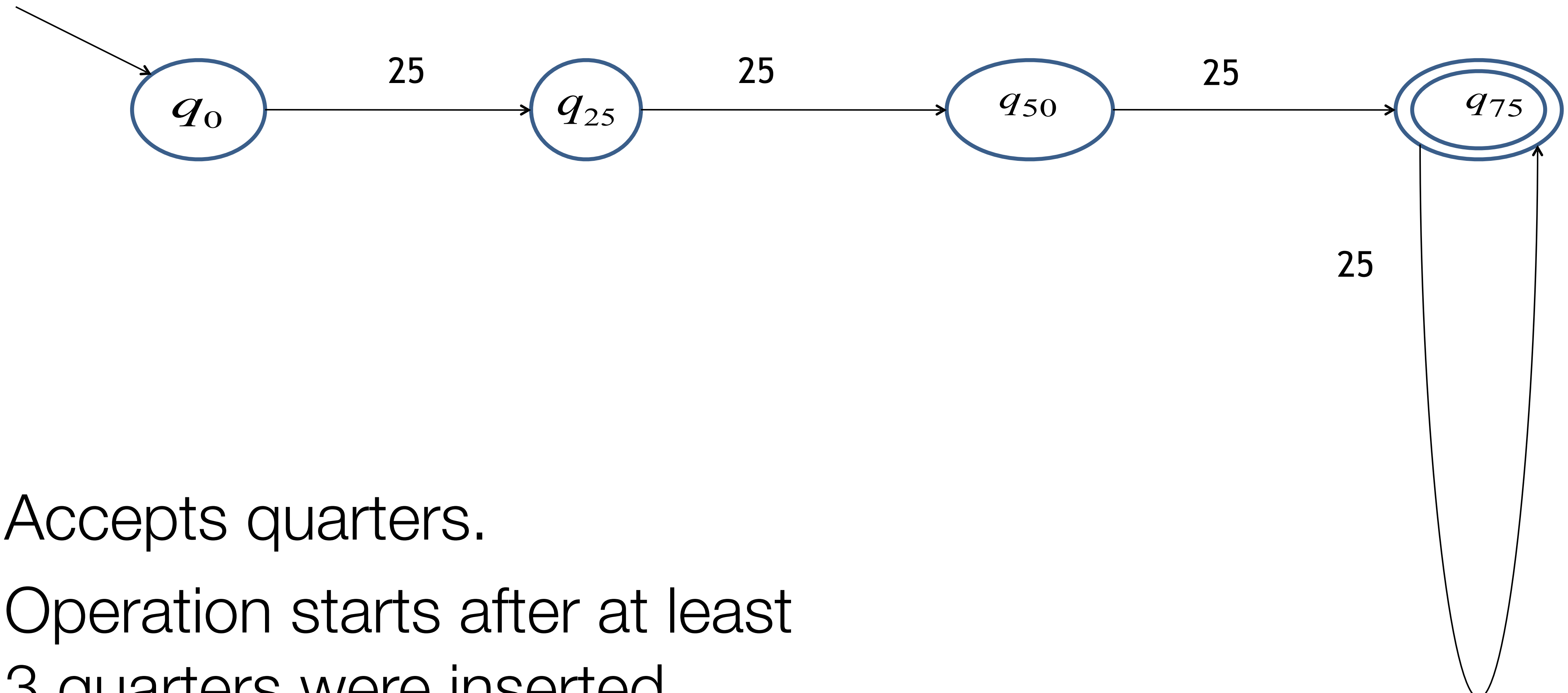
1. Finite Automaton (in short FA) recognize Regular Languages
2. Push Down Automaton (in short PDA) recognize Context Free Languages .
3. Turing Machines (in short TM) recognize Computable Languages .

increasing computational power

FA: Washing Machine Example

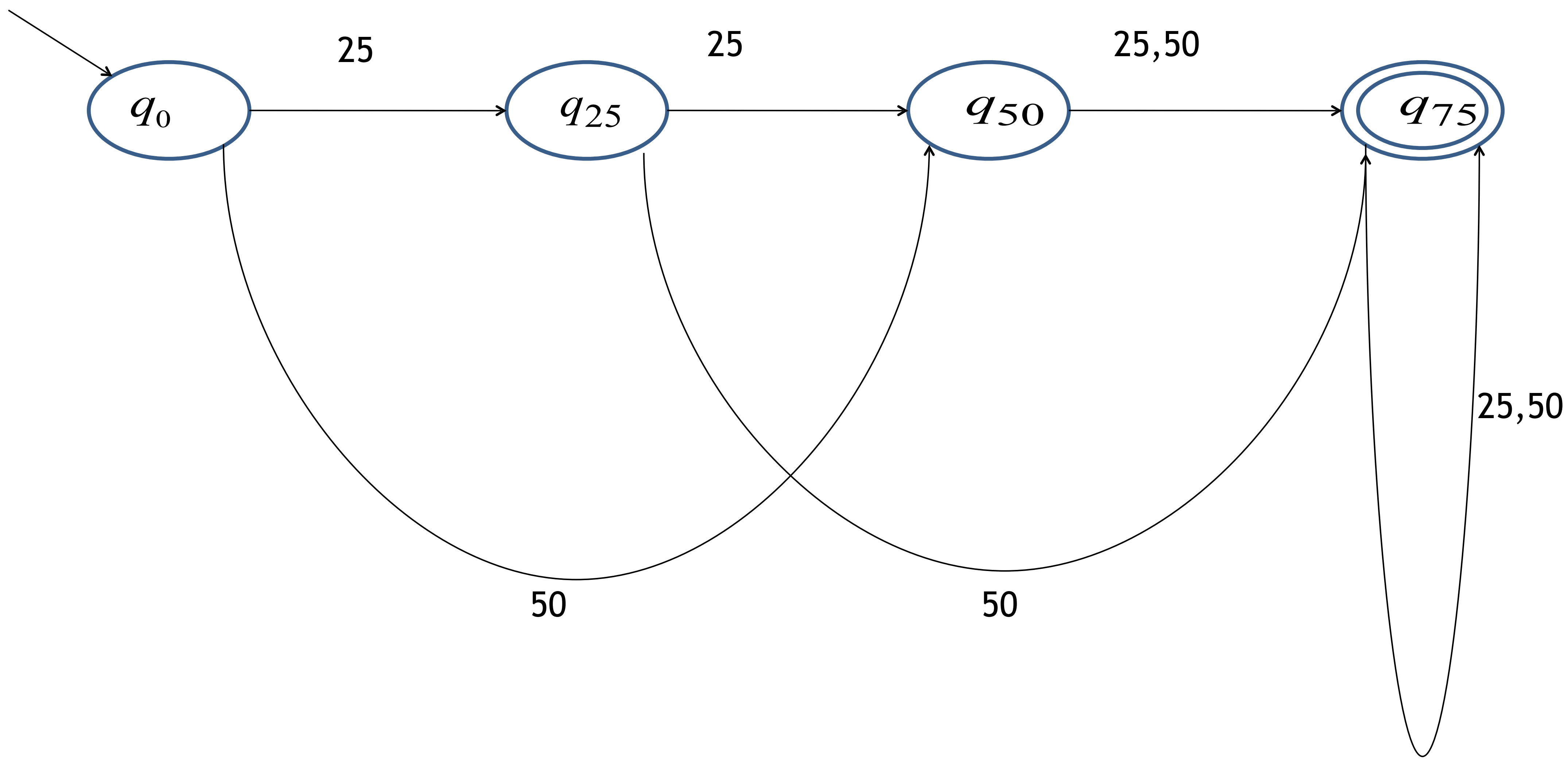
- The control of a washing machine is a very simple example of a finite automaton.
- The most simple washing machine accepts quarters and operation does not start until at least 3 quarters are inserted.

FA: Washing Machine Example



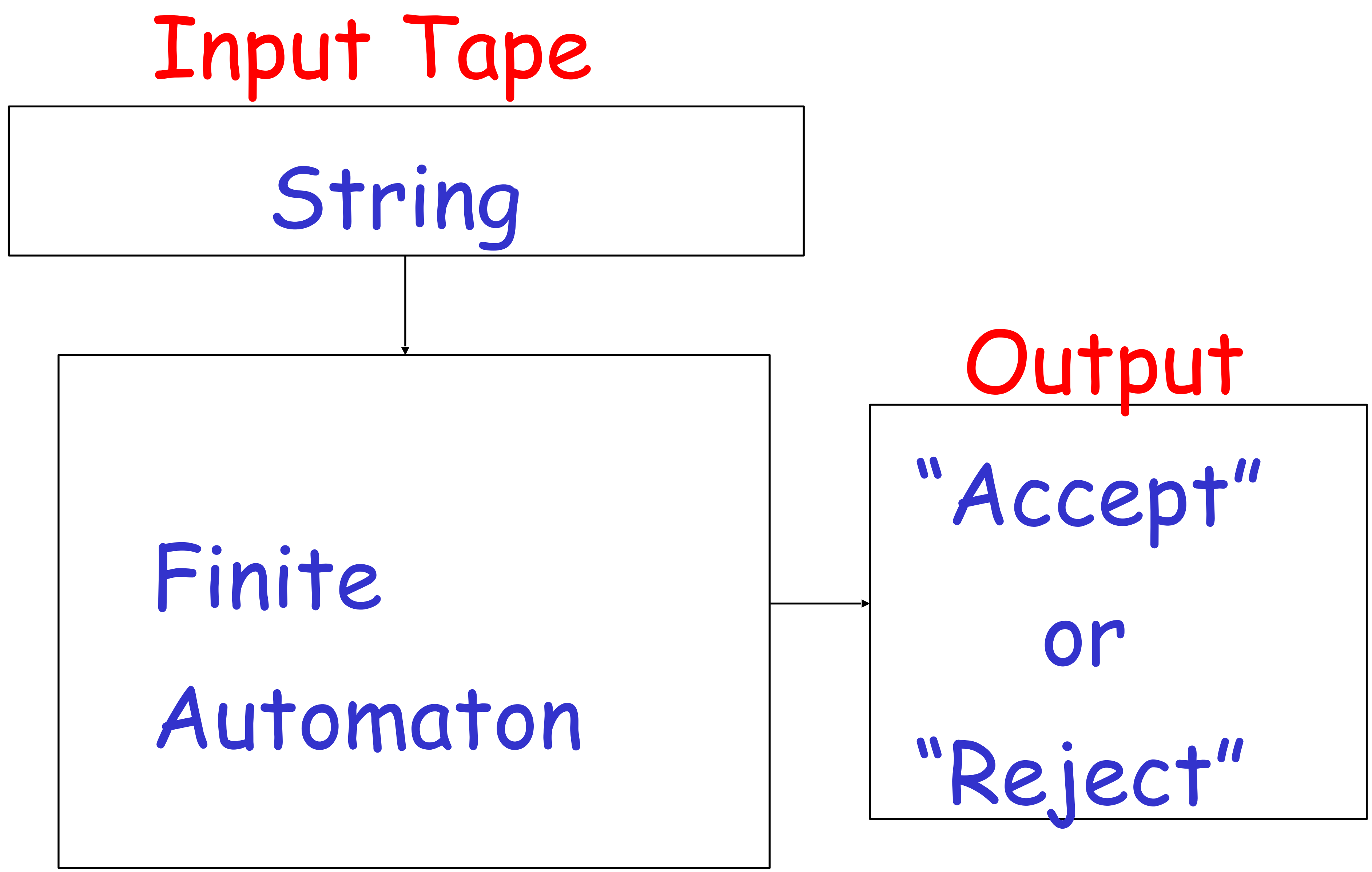
- Accepts quarters.
- Operation starts after at least 3 quarters were inserted.
- Accepted strings: 25,25,25; 25,25,25,25; …

FA: Washing Machine Example

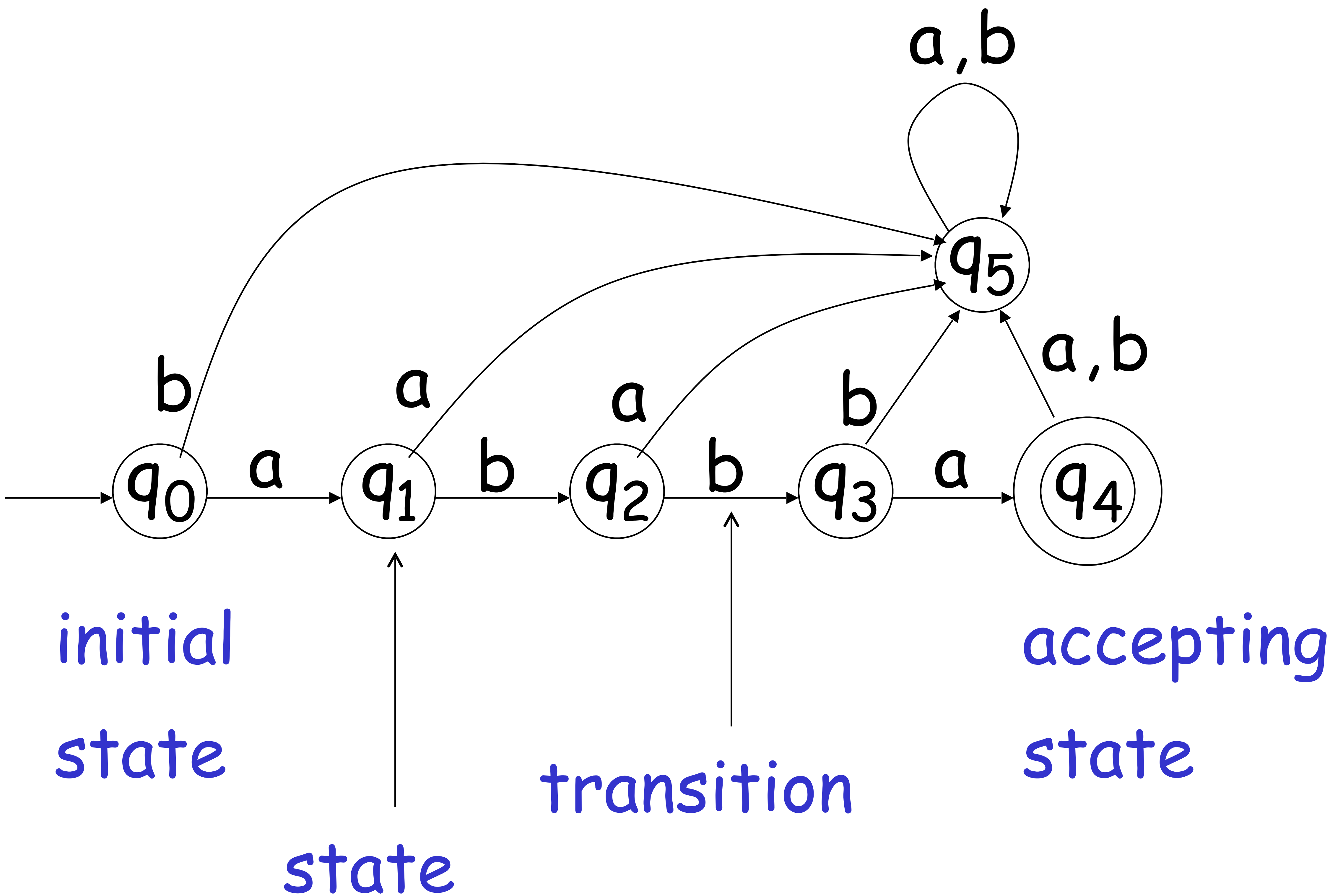


- A second washing machine also accepts half-dollar coins.
- Accepted strings: 25,25,25; 25,50; …

Finite Automaton (FA)

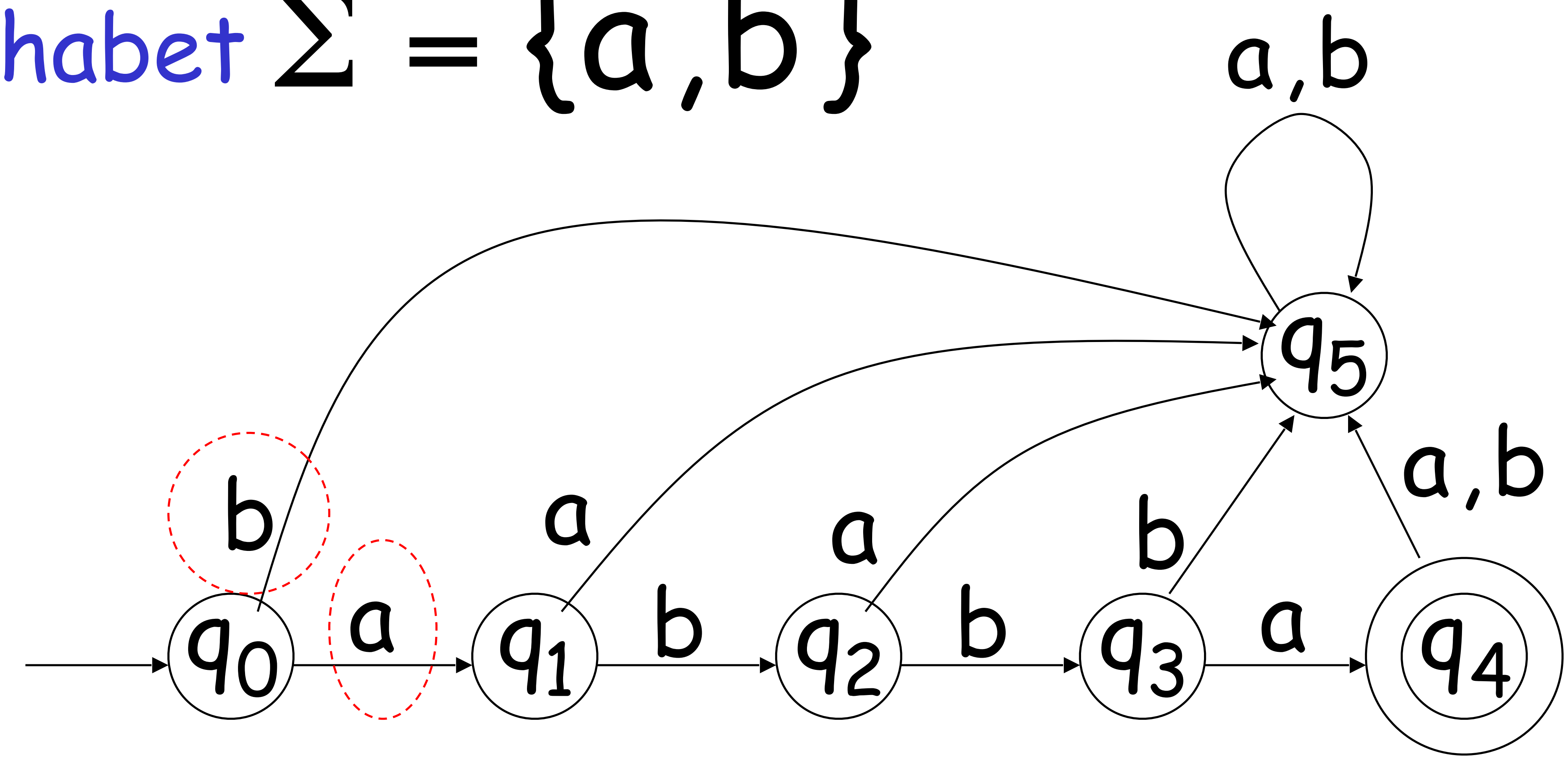


State Diagram



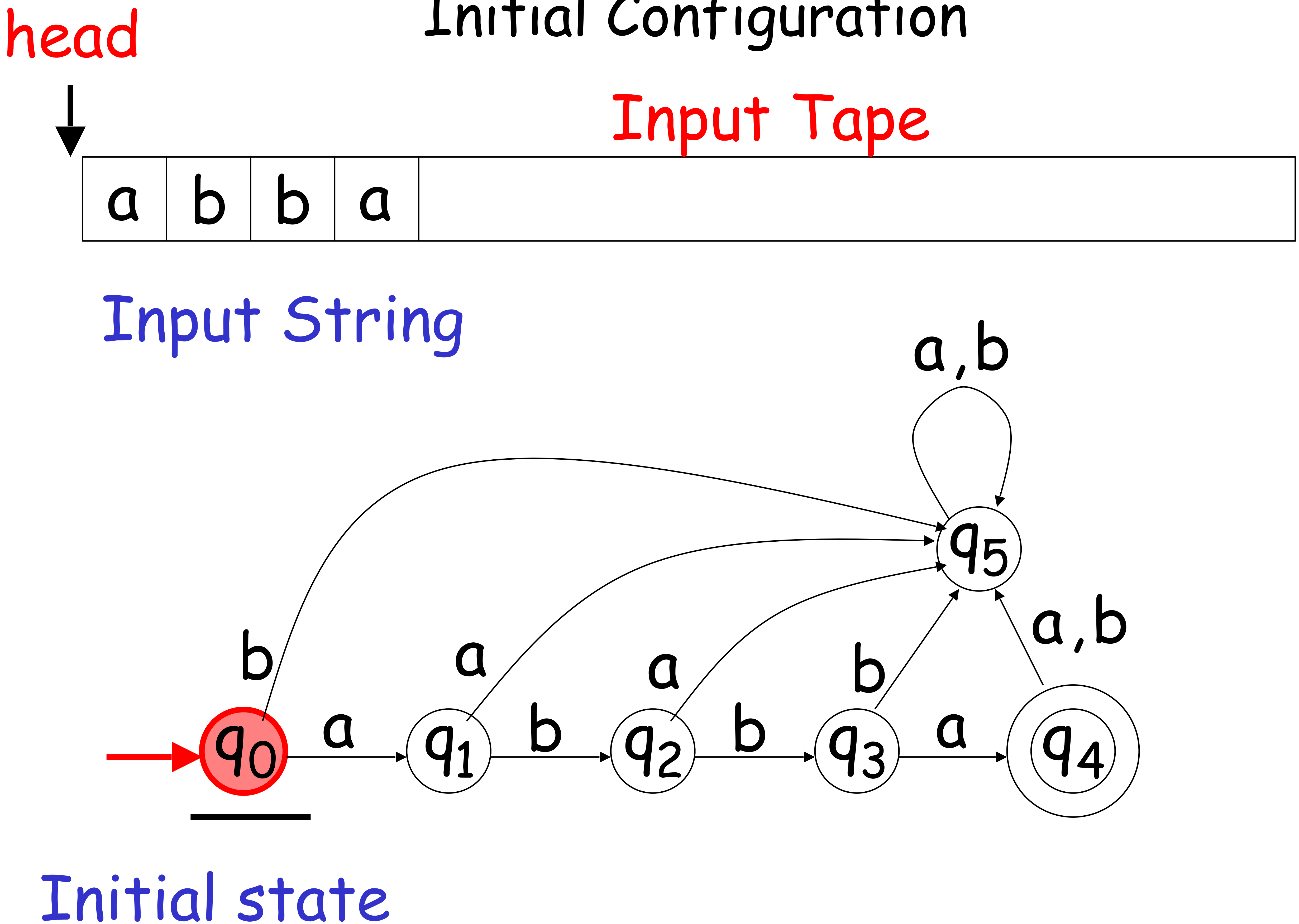
Deterministic Finite Automaton (DFA)

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

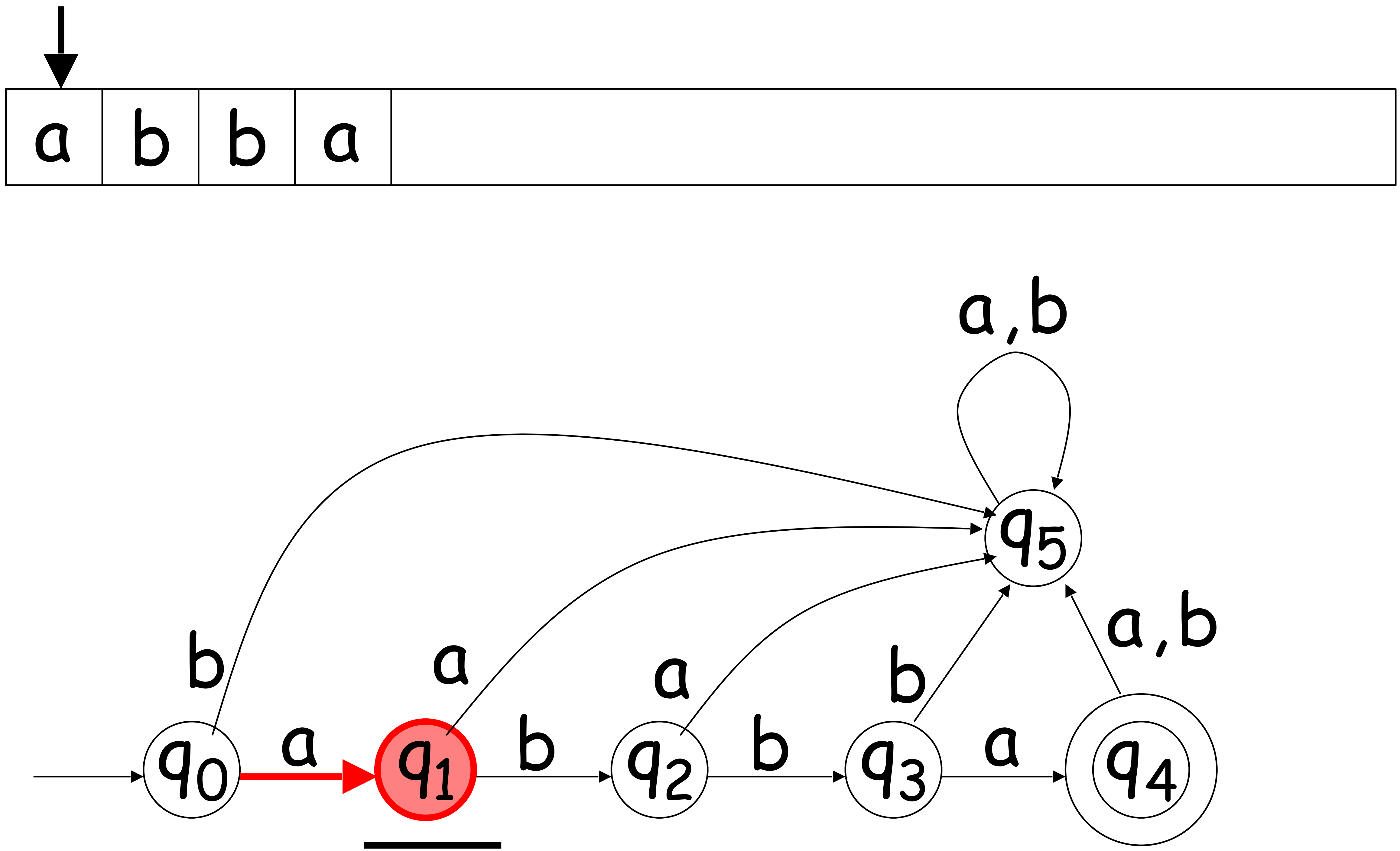


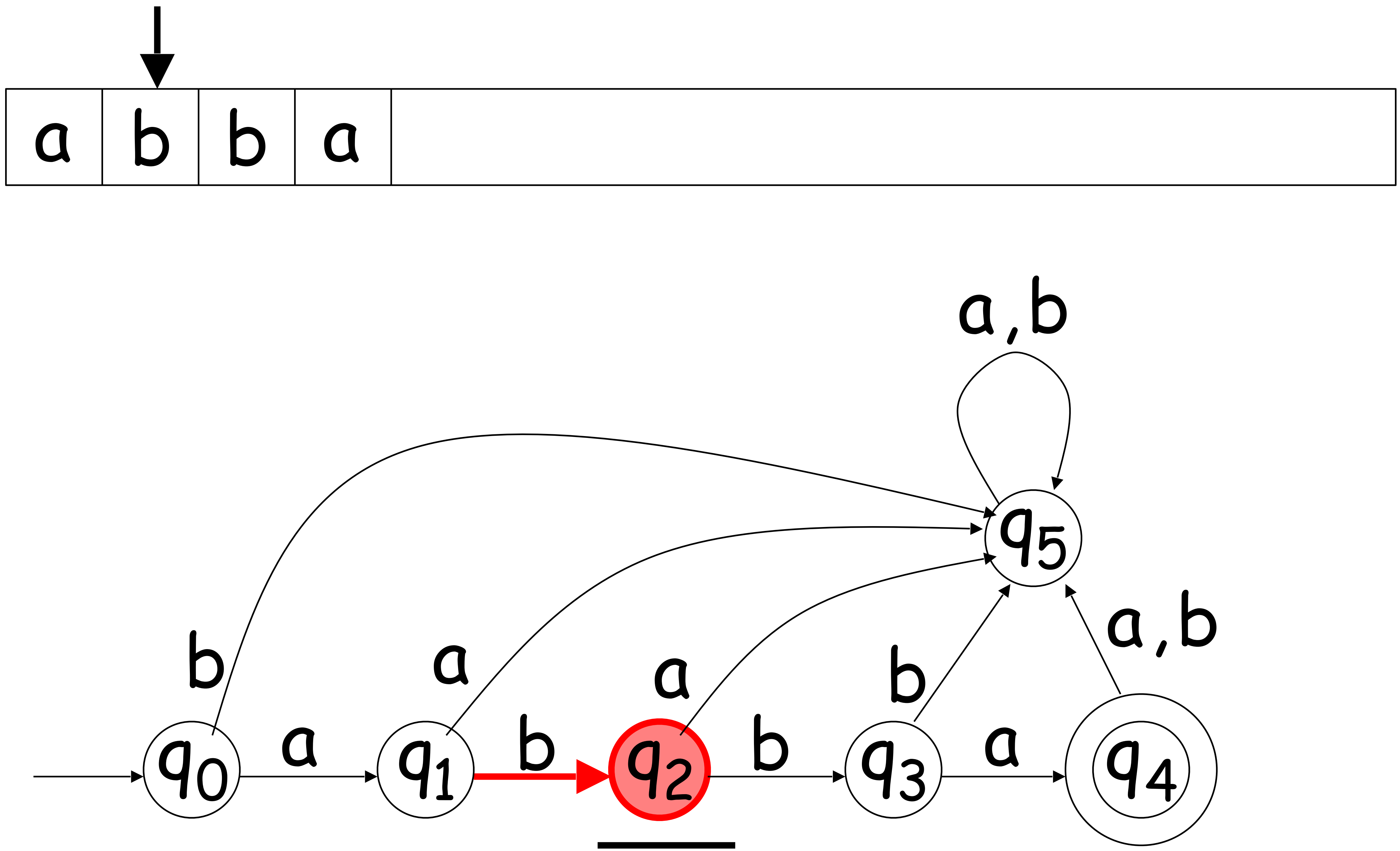
For every state, there is a transition
for every symbol in the alphabet

Initial Configuration

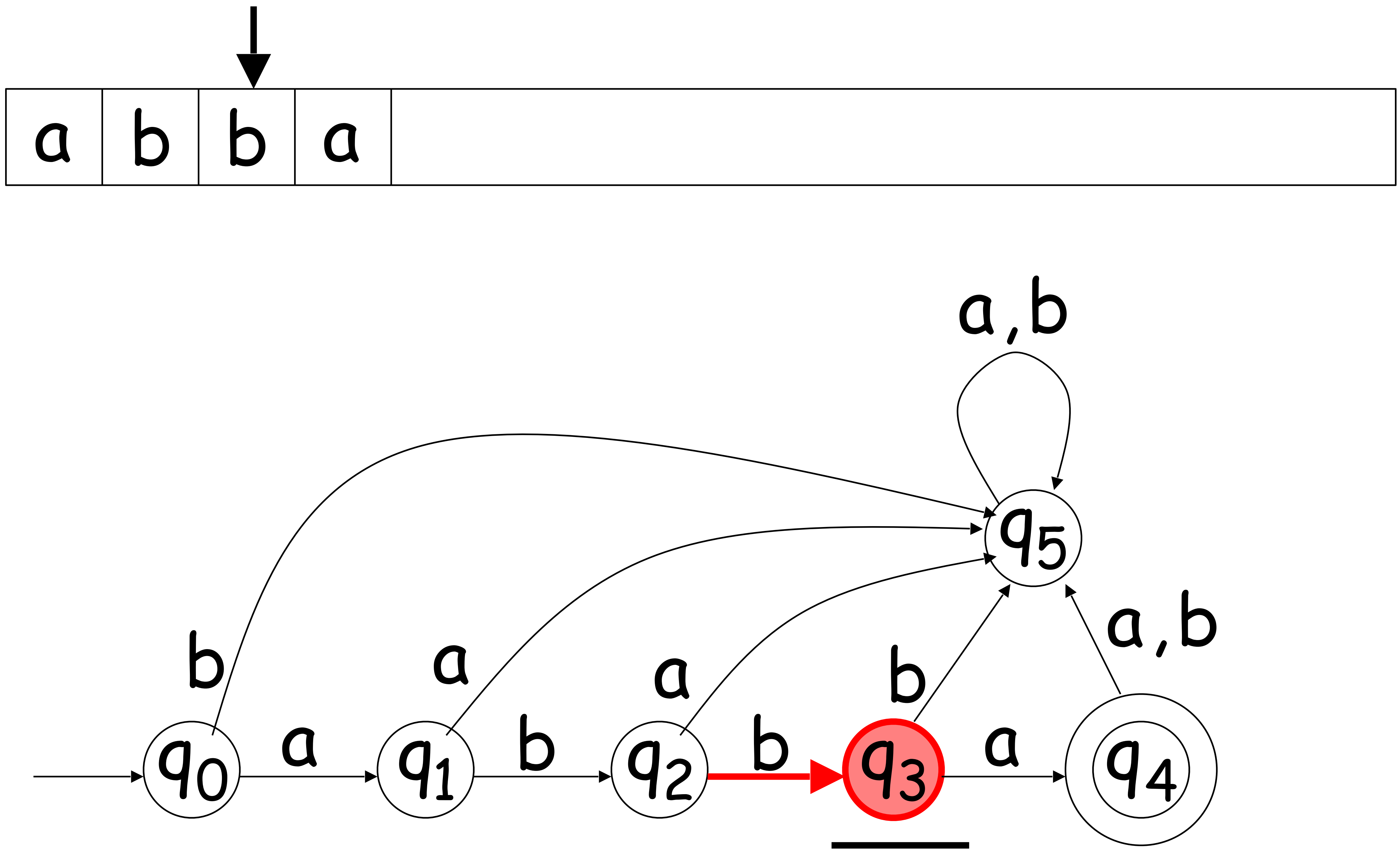


Scanning the Input



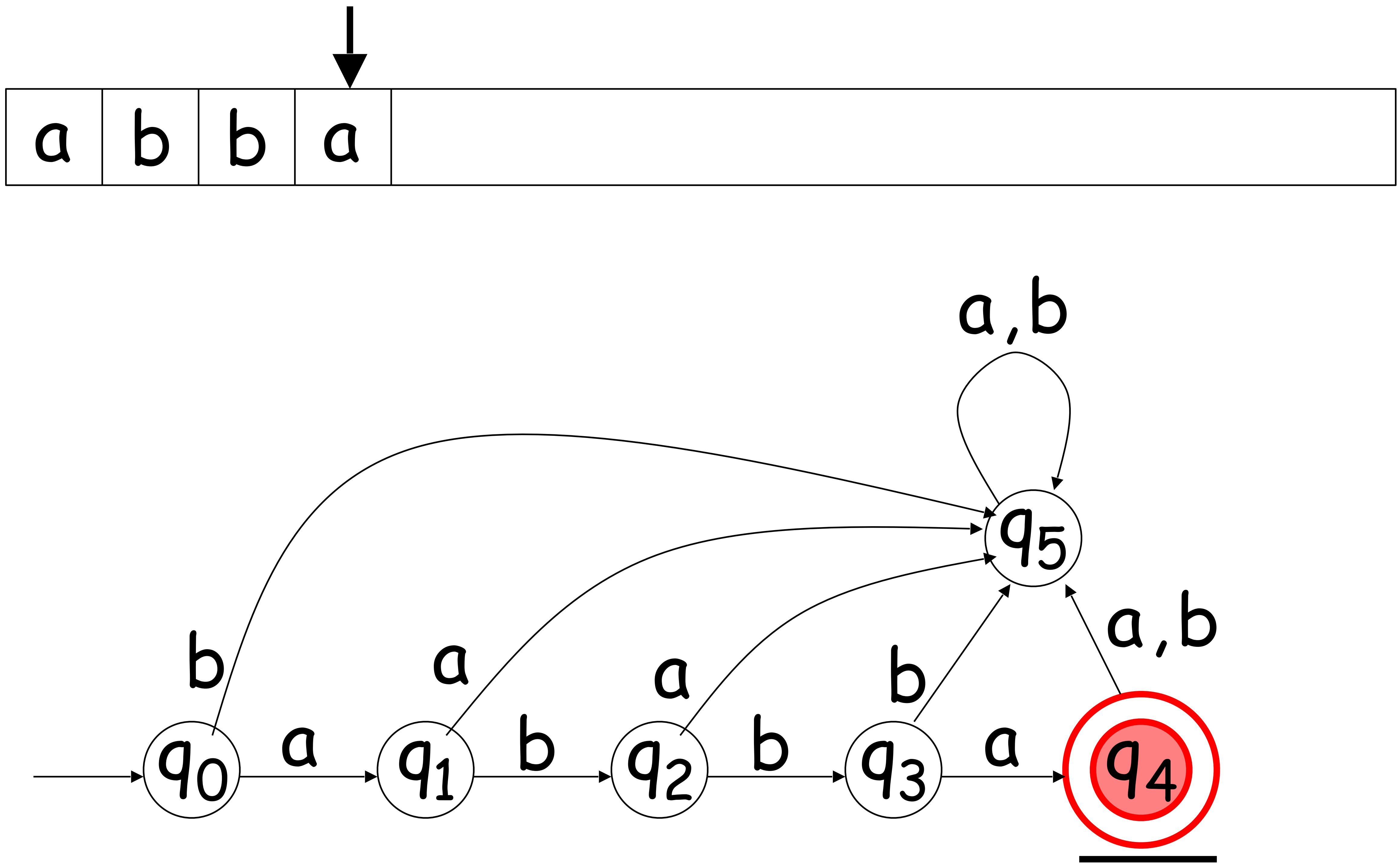


a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4



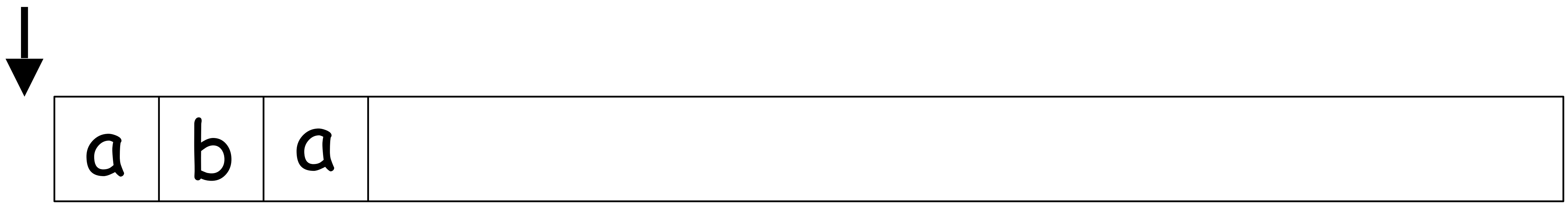
a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

Input finished

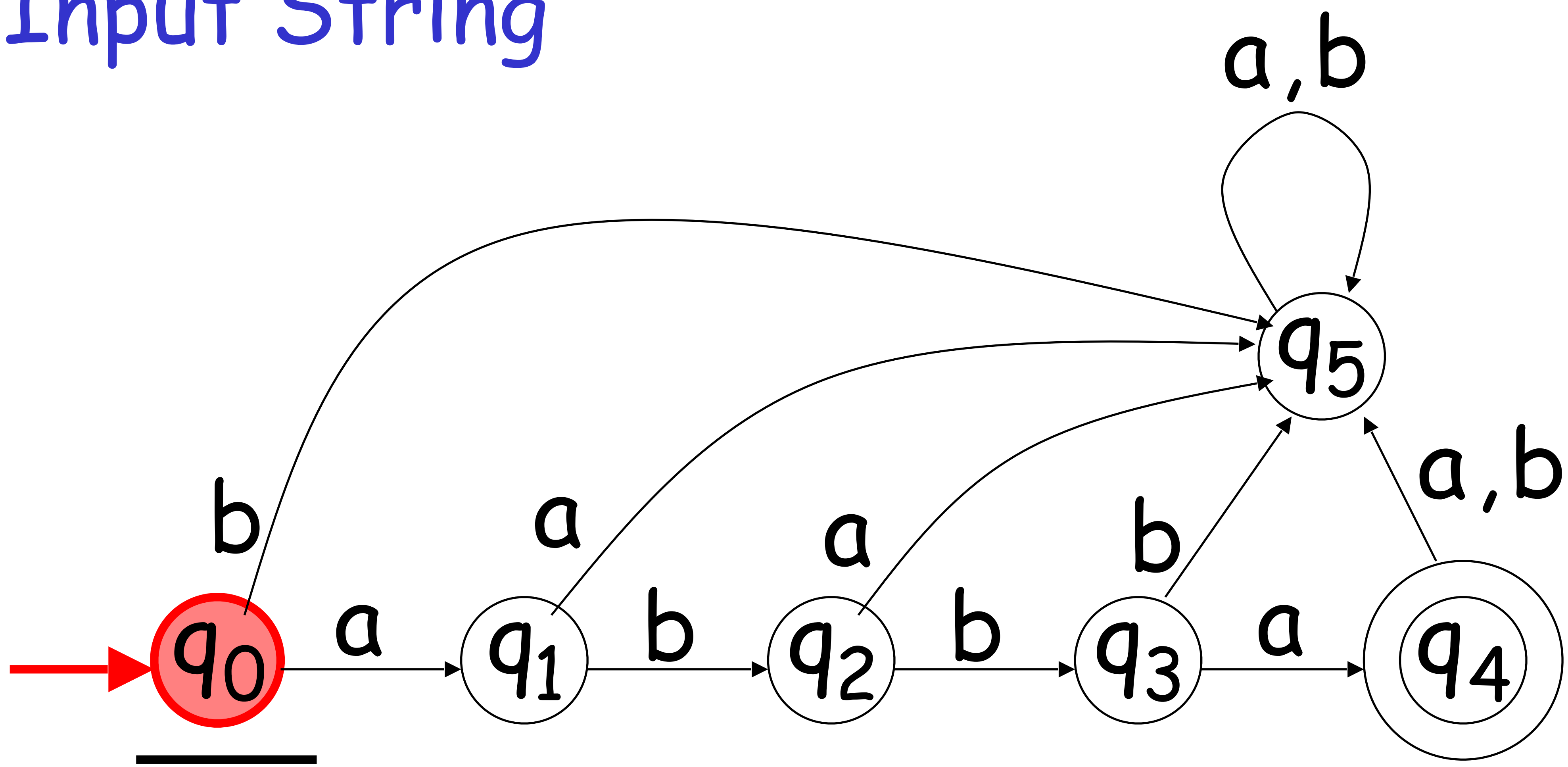


accept

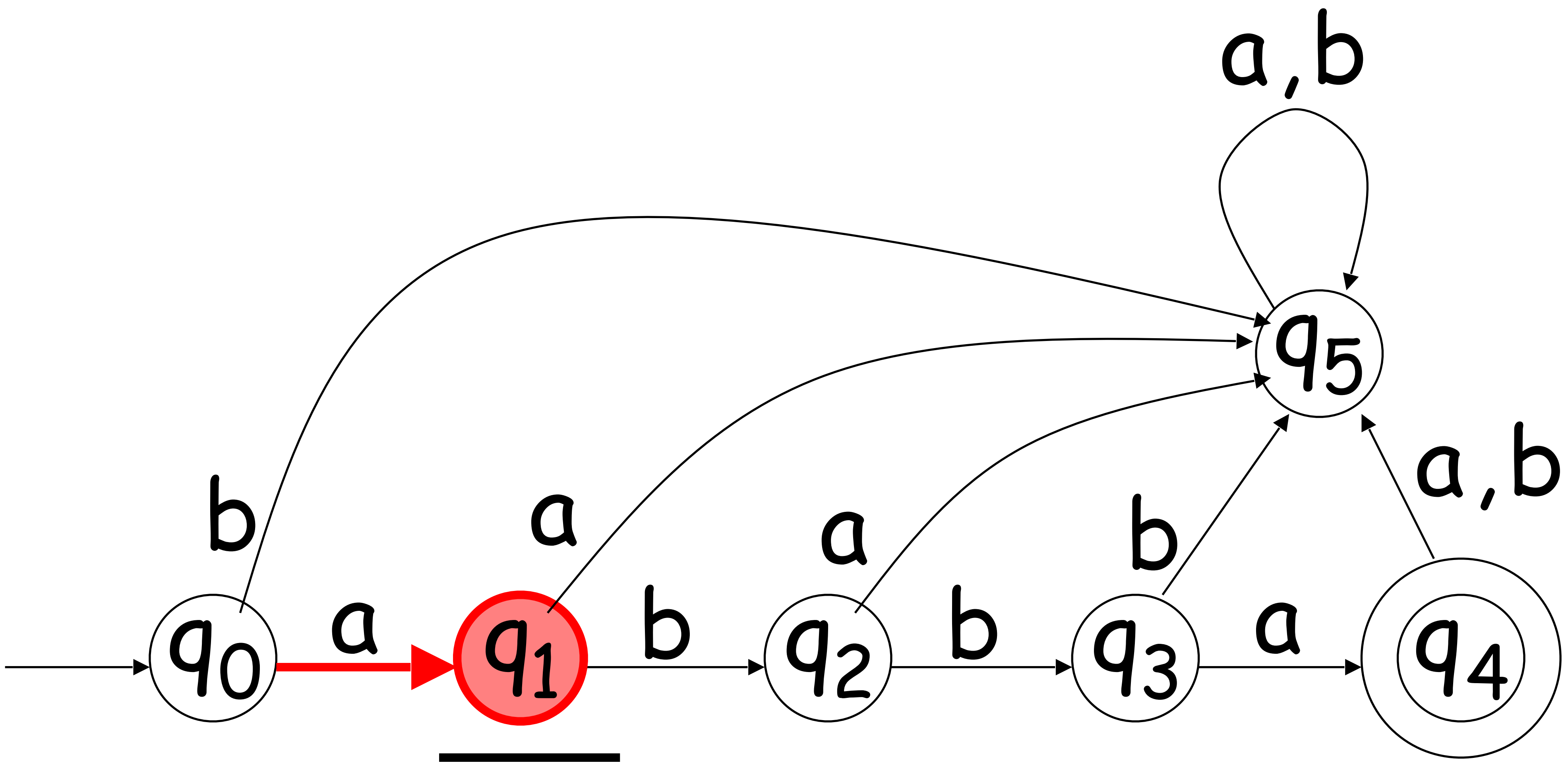
A Rejection Case

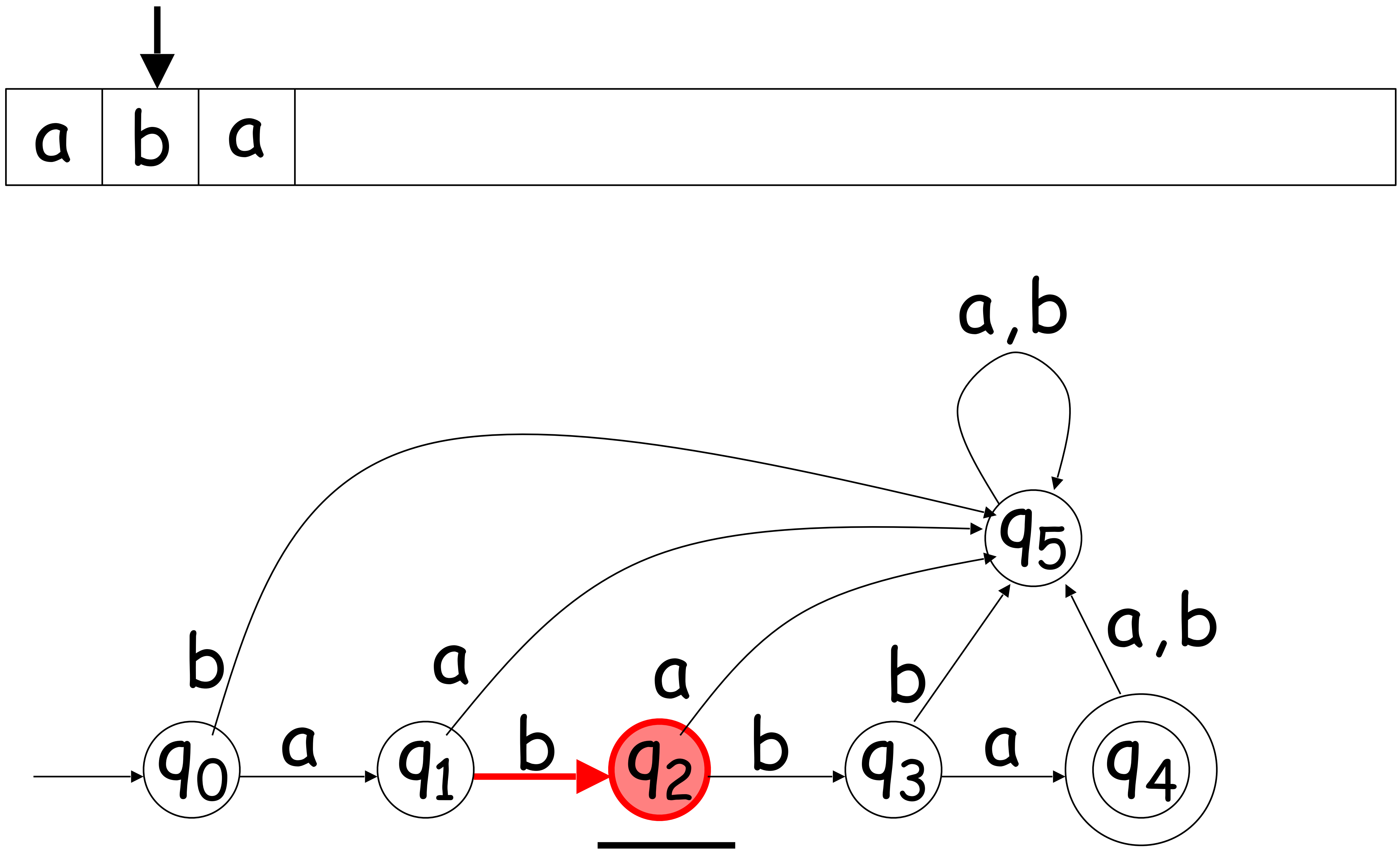


Input String

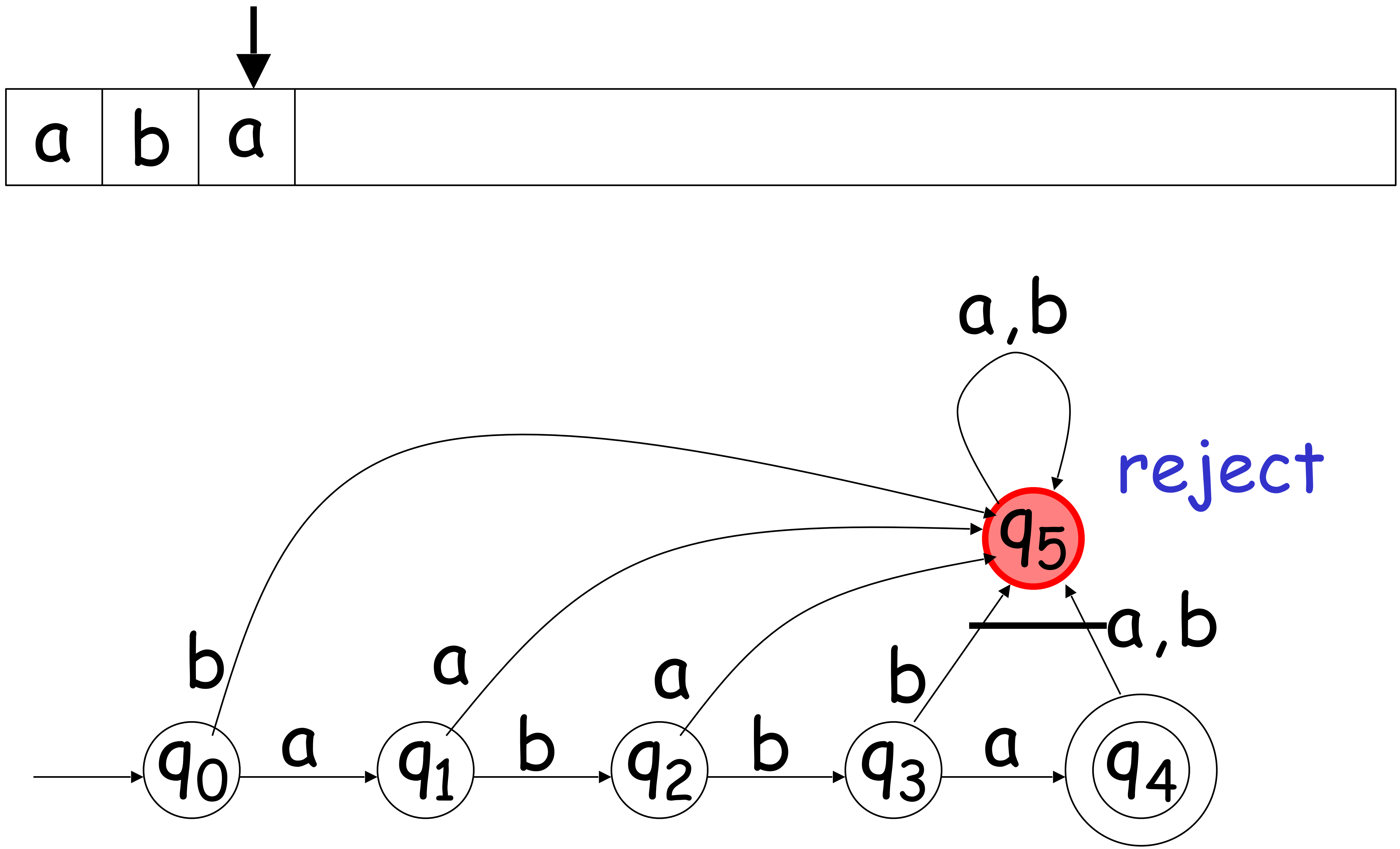



a
b
a


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


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

Input finished

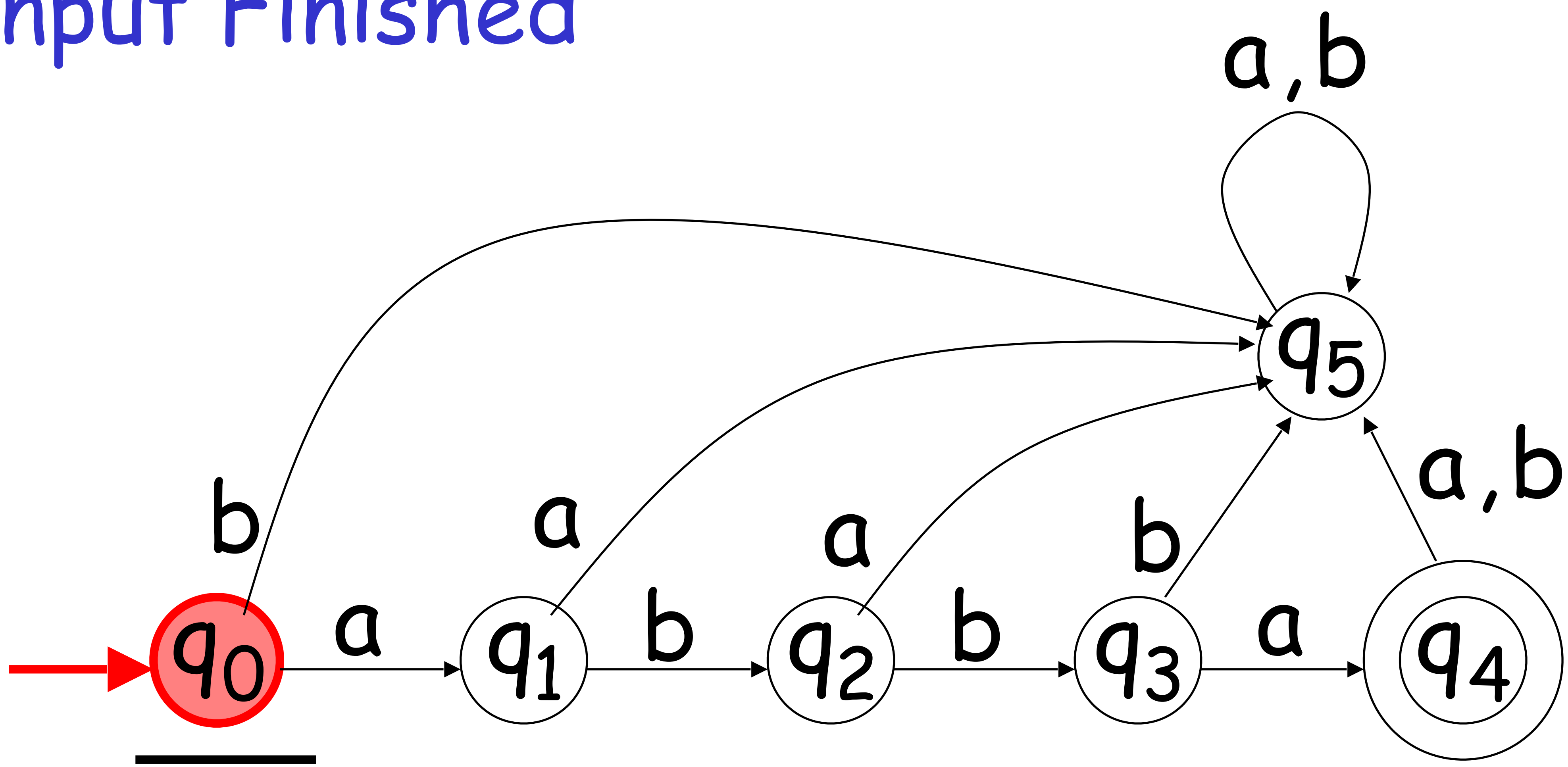


Another Rejection Case

Tape is empty

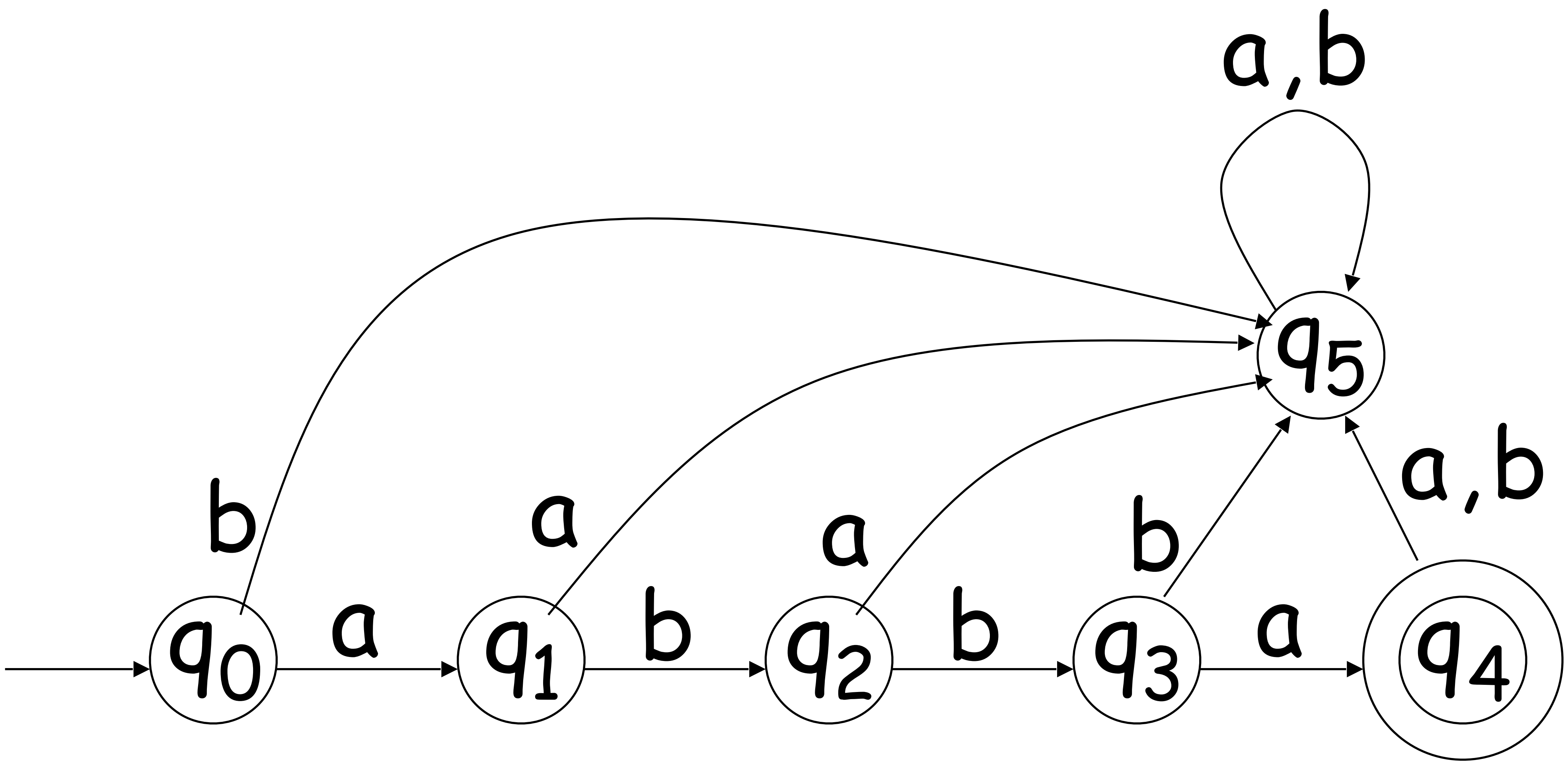
ϵ

Input Finished

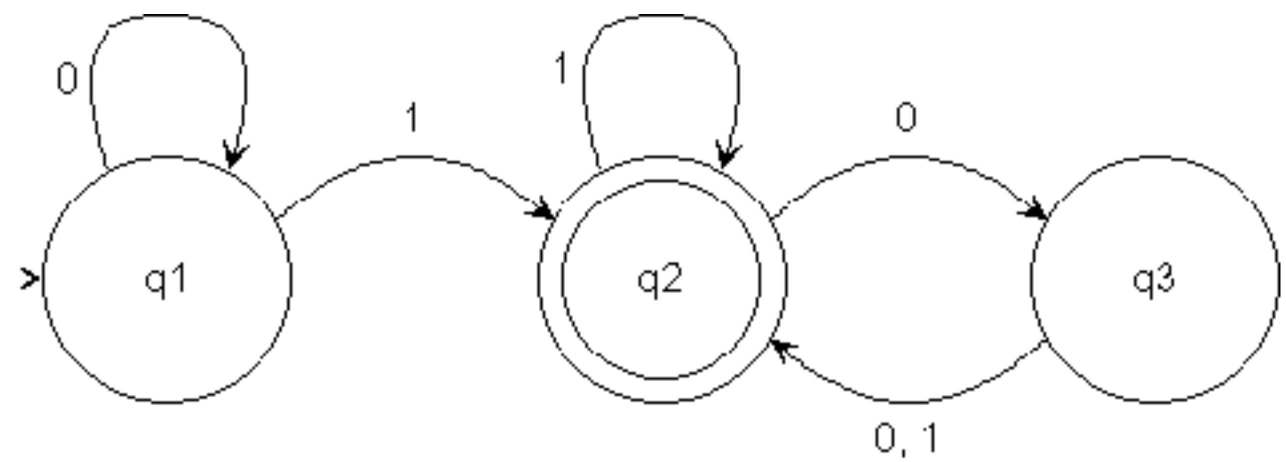


reject

Language Accepted: $L = \{abba\}$



FA: Second Example



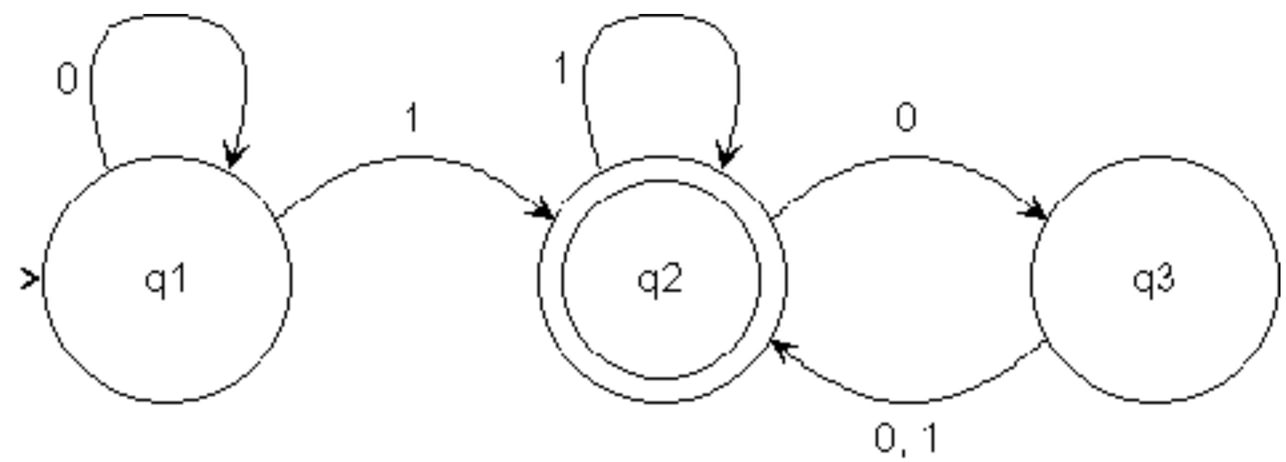
Does it accept the following strings:

0101 

01110 

0100 

FA: Second Example



- States: q1, q2, q3
- Start State: q1
- Final State: q2
- Alphabet $\Sigma = \{0, 1\}$
- Transition function:

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

$$\delta(q_1, 1) = q_2$$

$$\dots$$

Formal Definition

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

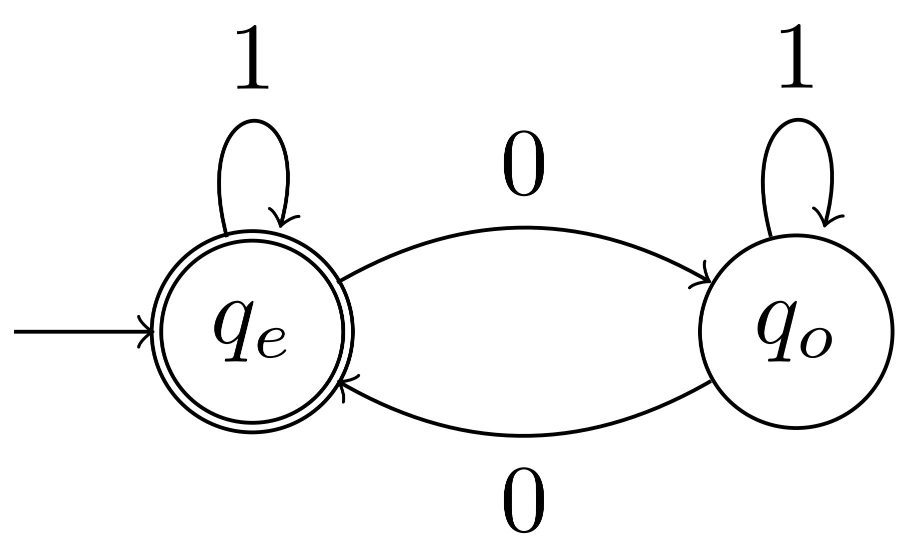
1. Q is a finite set called the ***states***.
2. Σ is a finite set called the ***alphabet.***
3. $\delta : Q \times \Sigma \rightarrow Q$ is the ***transition function.***
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the set of ***accept states***.

Designing FA

- We would like to design a DFA for the following languages (examples on board)
  - L1 = { w | w has even number of 0’s}
  - L2 = {w | w has even number of 0’s and 1’s}
  - L3 ={ w | w start with 00}
  - L4 = { w | w divisible by 4}

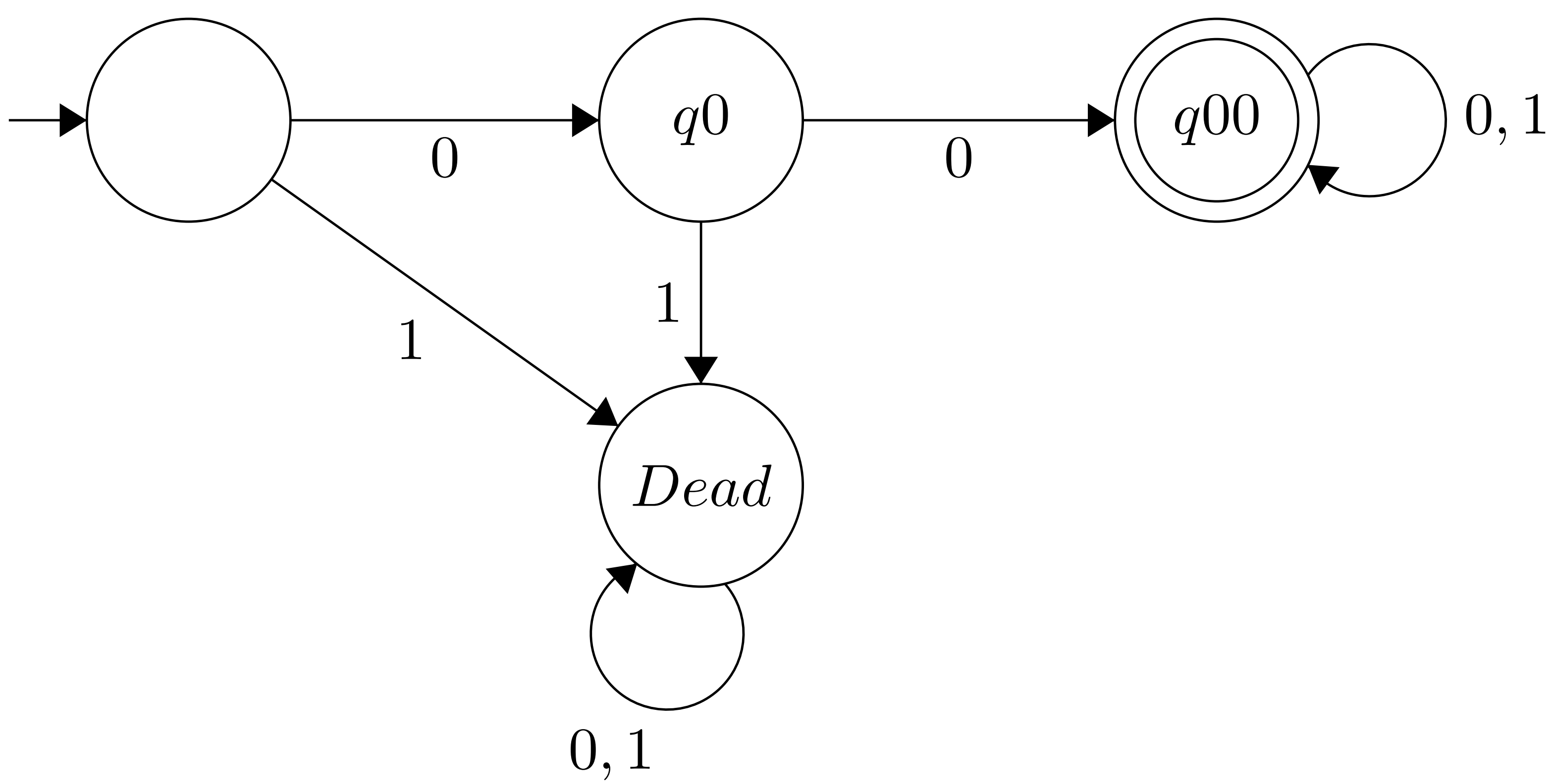
Example

- L = { w | w has even number of 0's}



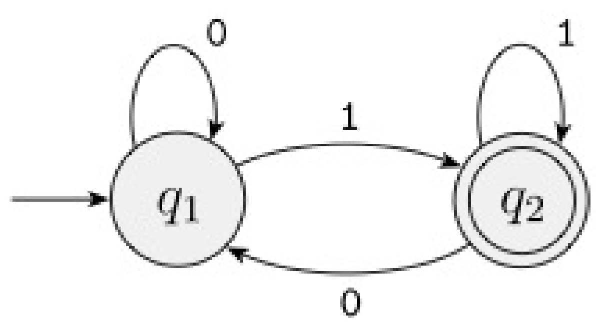
Example

- L = { w | w starts with 00}



Example formal definition

- L = { w | w ends with a 1}



- Formal description

$$(\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$$

function δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2.

Regular Language

- Definition: A ***language*** is a set of strings over some alphabet.

- The language of an FA, M, designated $L(M)$, is the set of strings that M **accepts**

- If L is recognized by some finite automaton, then L is

  a **regular language.**

Questions

Q1: How do you prove that a language L is regular?

A1: By presenting an FA, *M*, such that $L(M) = L_a$

Q2: Why is this important?

A2: It defines a class of problems that can be solved by a computational device with bounded memory.

Q3: How do you prove that a language L is not regular?

A3: This is more difficult! We'll answer this later.