FINITE AUTOMATA

Dr. A. Beulah

AP/CSE

FORMAL LANGUAGE

- A **formal language** consists of words(strings) whose letters(symbols) are taken from an alphabet and are well-formed according to a specific set of rules.
  - Regular Languages
  - Context Free Languages
  - Context Sensitive Languages
  - Recursive Languages
  - Recursive Enumerable Languages

REGULAR LANGUAGES

- A regular language is a language that can be expressed with a
  - **regular expression** or
  - finite automata.

- Alternatively, a *regular language* can be defined as a language recognized by a finite automaton.

CONTEXT FREE LANGUAGES

- It allows richer syntax than Regular Languages.
- Context free grammars can be recognised by computing devices like ***pushdown automata.***
- Pushdown automata is a finite automata with an auxiliary memory in the form of a stack.
- It is immensely used in the design of parsers - another key portion of a compiler

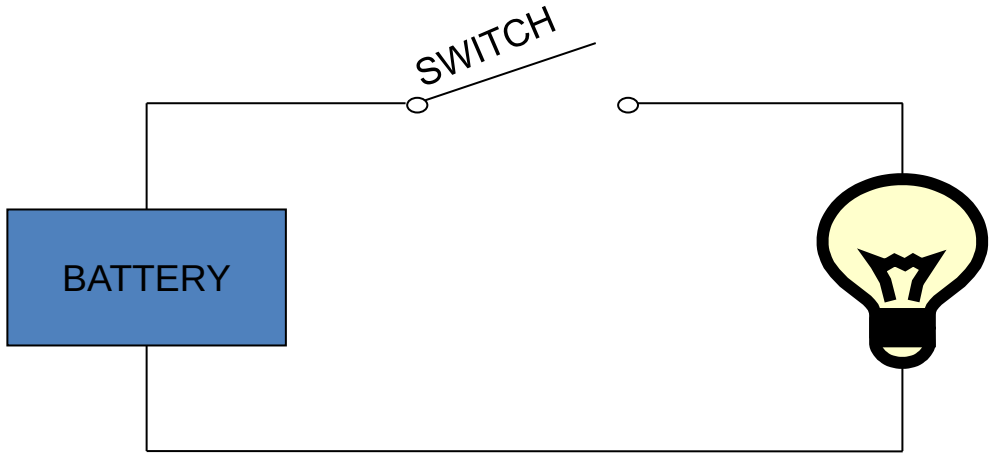
R AND RE LANGUAGES

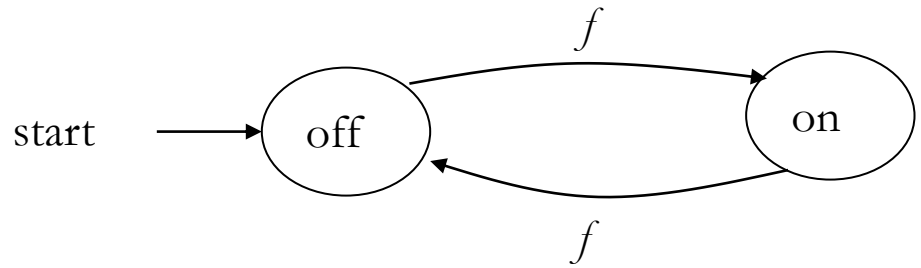
- R and RE Languages can be recognized by Turing machines
- It was invented by British Mathematician *"Alan Turing"*
- It has infinite amount of tape memory accessible in both directions, that is left (or) right.
- It can recognize recursively enumerable languages.
- It simulates digital computer in terms of power.
- If any function is not solvable by Turing machine, it cannot be computed by digital computer.

FINITE AUTOMATA

- FA recognizes regular languages only. It was developed by “Scott & Rabin” in 1950 as a model of a computer with limited memory.

FINITE AUTOMATA CONT…





input: switch

output: light bulb

actions: f for "flip switch"

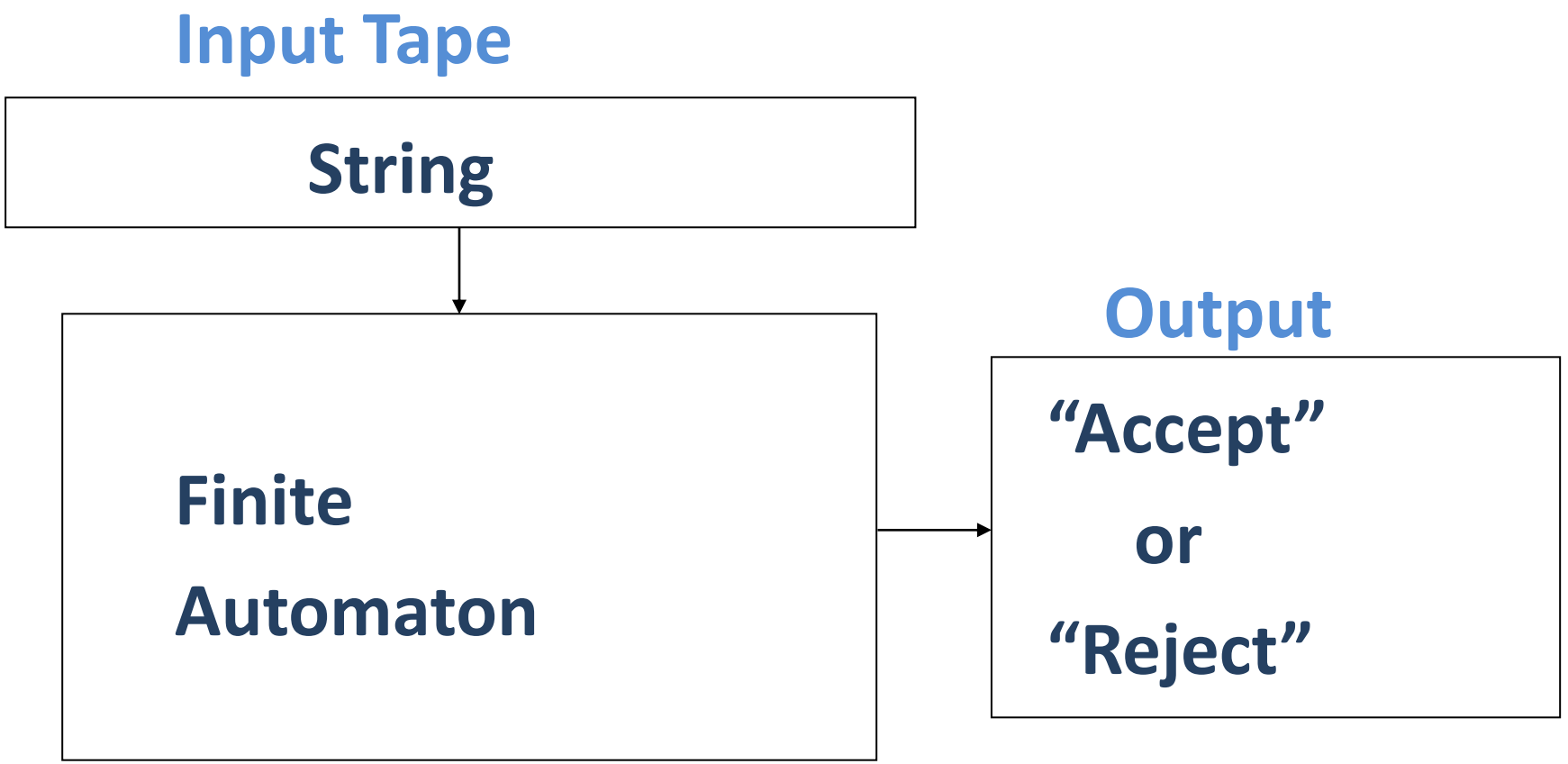
states: on, off

bulb is on if and only if there was an odd number of flips

FINITE AUTOMATA CONT…

- The FA or Finite State Machine(FSM) is a mathematical model of a system, with
  - discrete inputs and outputs
  - a finite number of states
  - a set of transitions from state to state that occurs over input symbols from alphabet Σ

FINITE AUTOMATA CONT…



REPRESENTATION OF FA

- Set of Transition Functions
- Transition Table
- Transition Diagram

REPRESENTATION OF FA

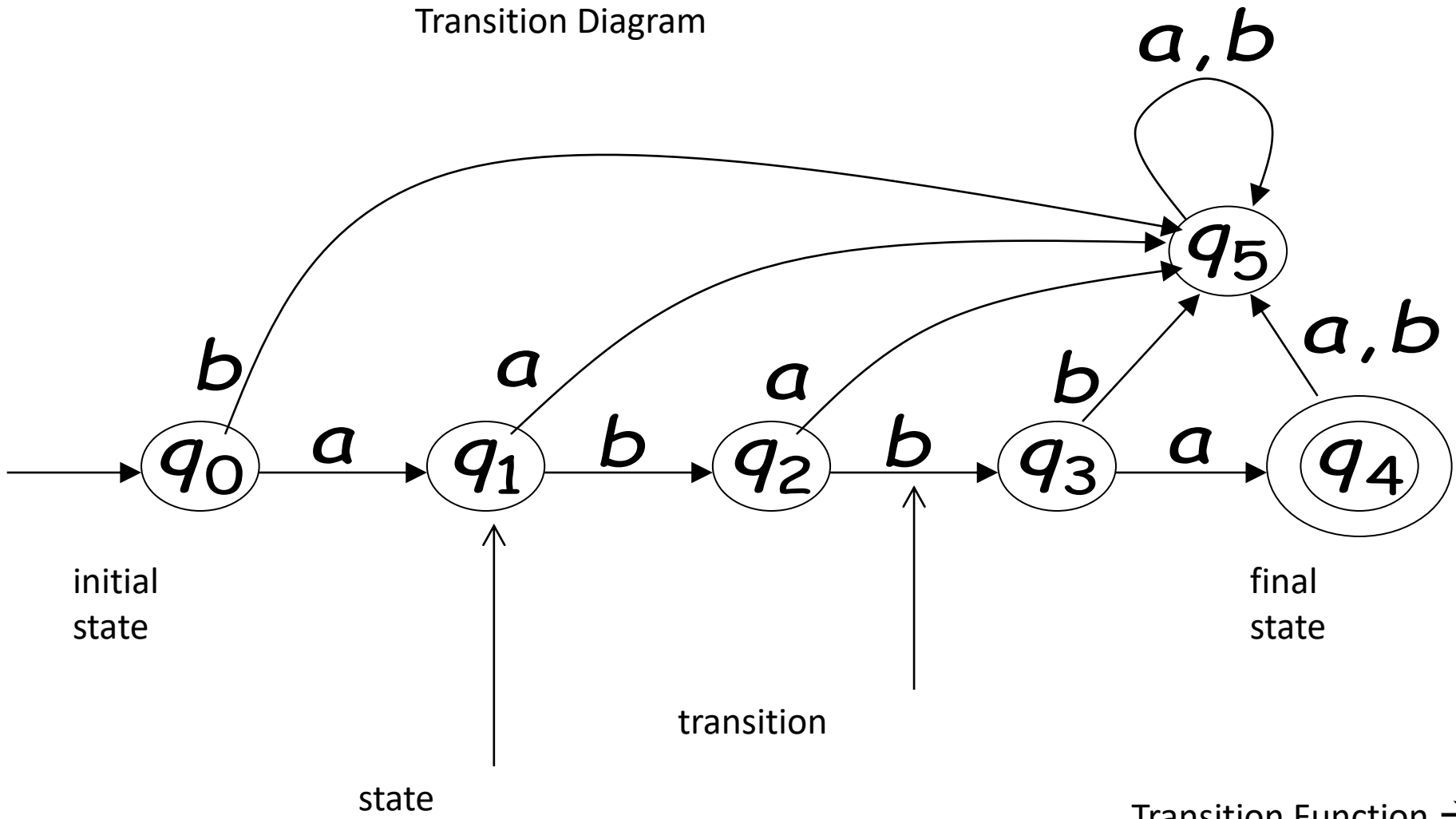
Input symbols

states	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

Transition Table

REPRESENTATION OF FA

Transition Diagram



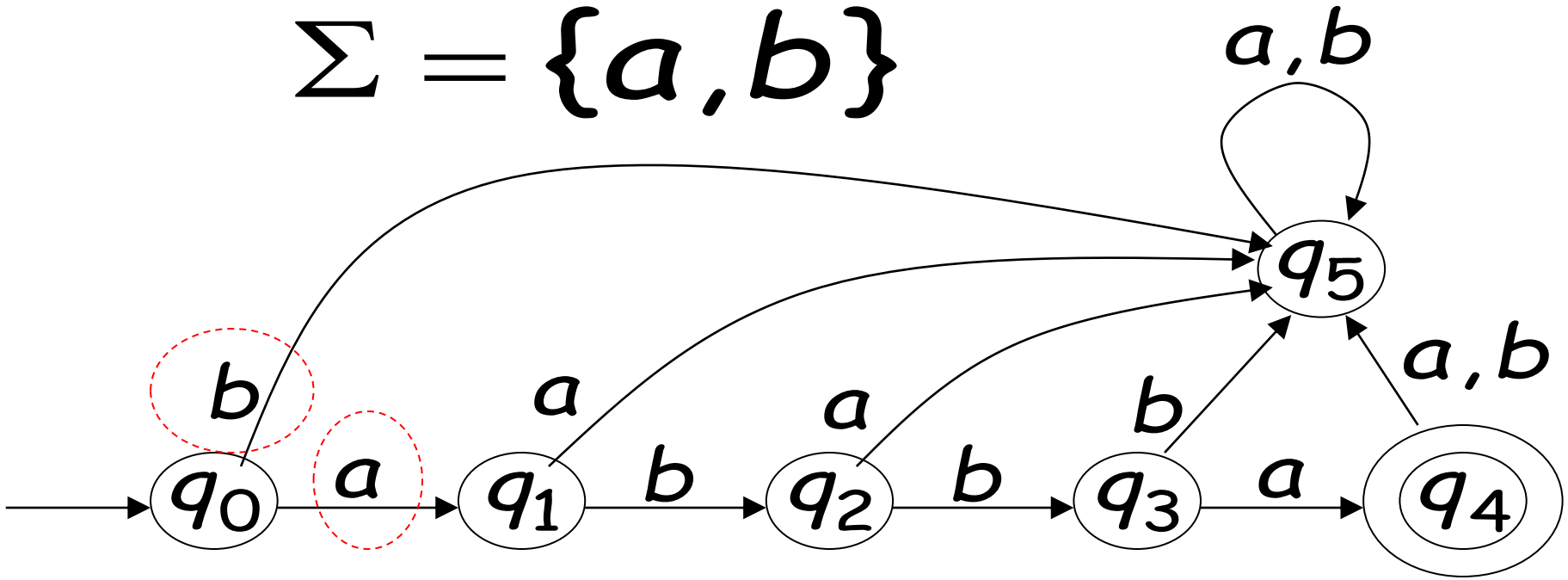
Transition Function → Inclass

HOW FA WORKS ?

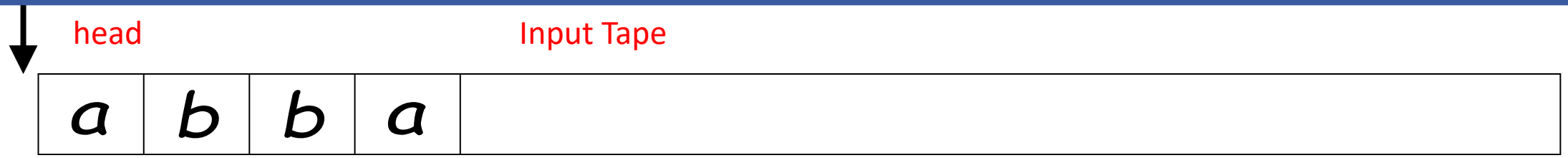
- Initialization :
  - Reader start reading from the leftmost symbol. Finite control is in start state.
- Single step :
  - Reader reads current symbol then, reader moves to the next symbol to the right. And Control enters a new state
- No current symbol :
  - All symbols have been read then, if control is in final state, the input string is accepted. Otherwise, the input string is not accepted.

HOW FA WORKS ?

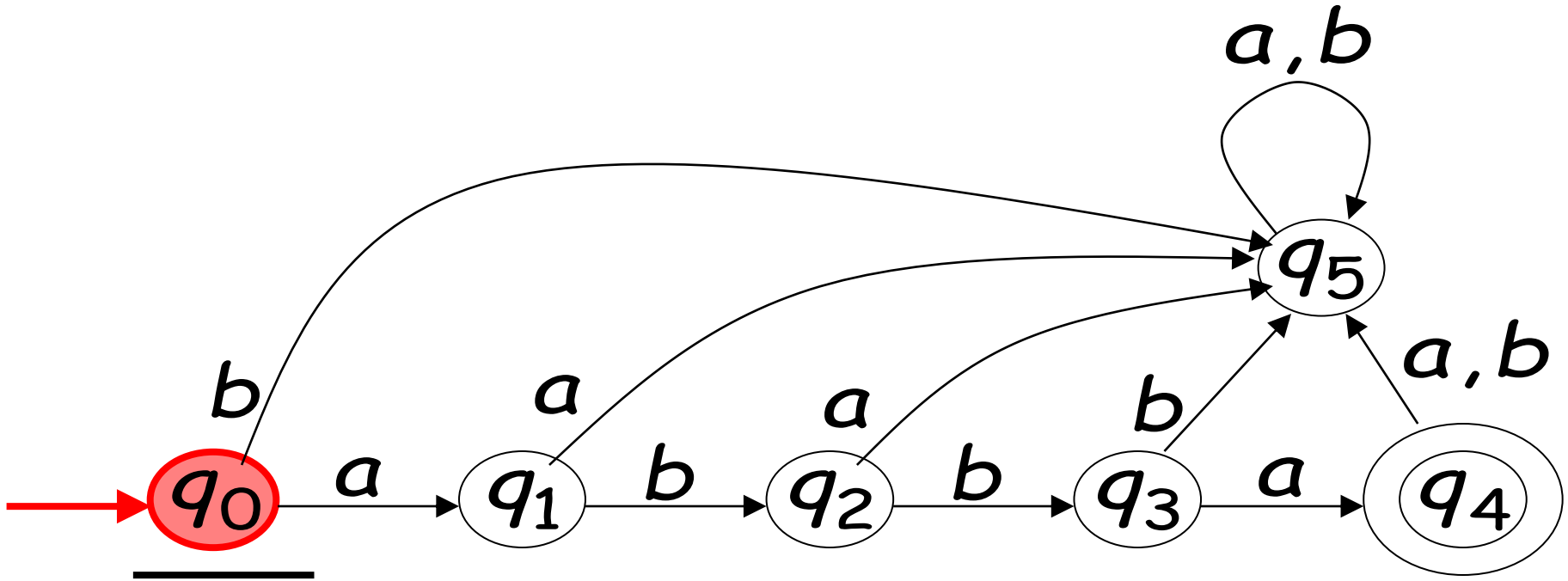
$$\Sigma = \{a, b\}$$



HOW FA WORKS ?

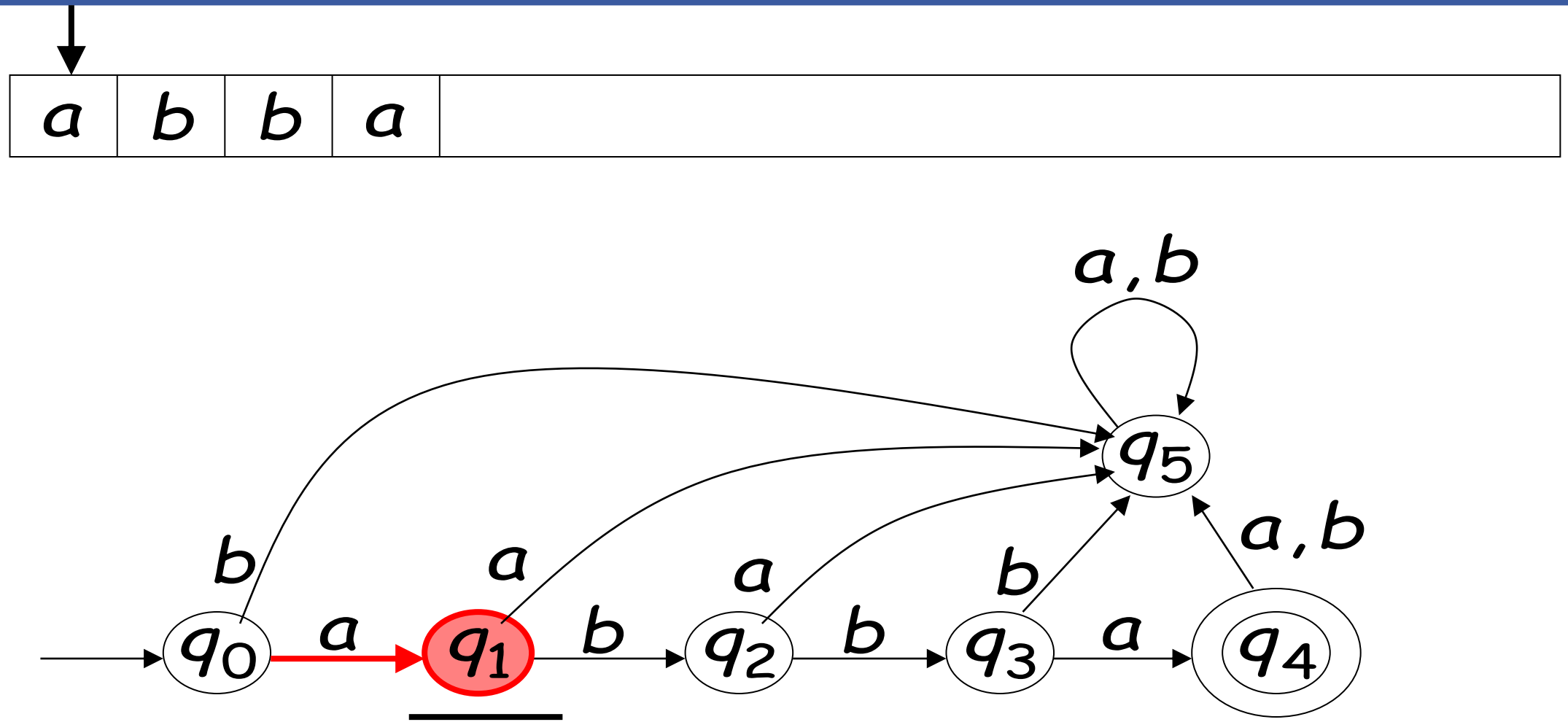


Input String

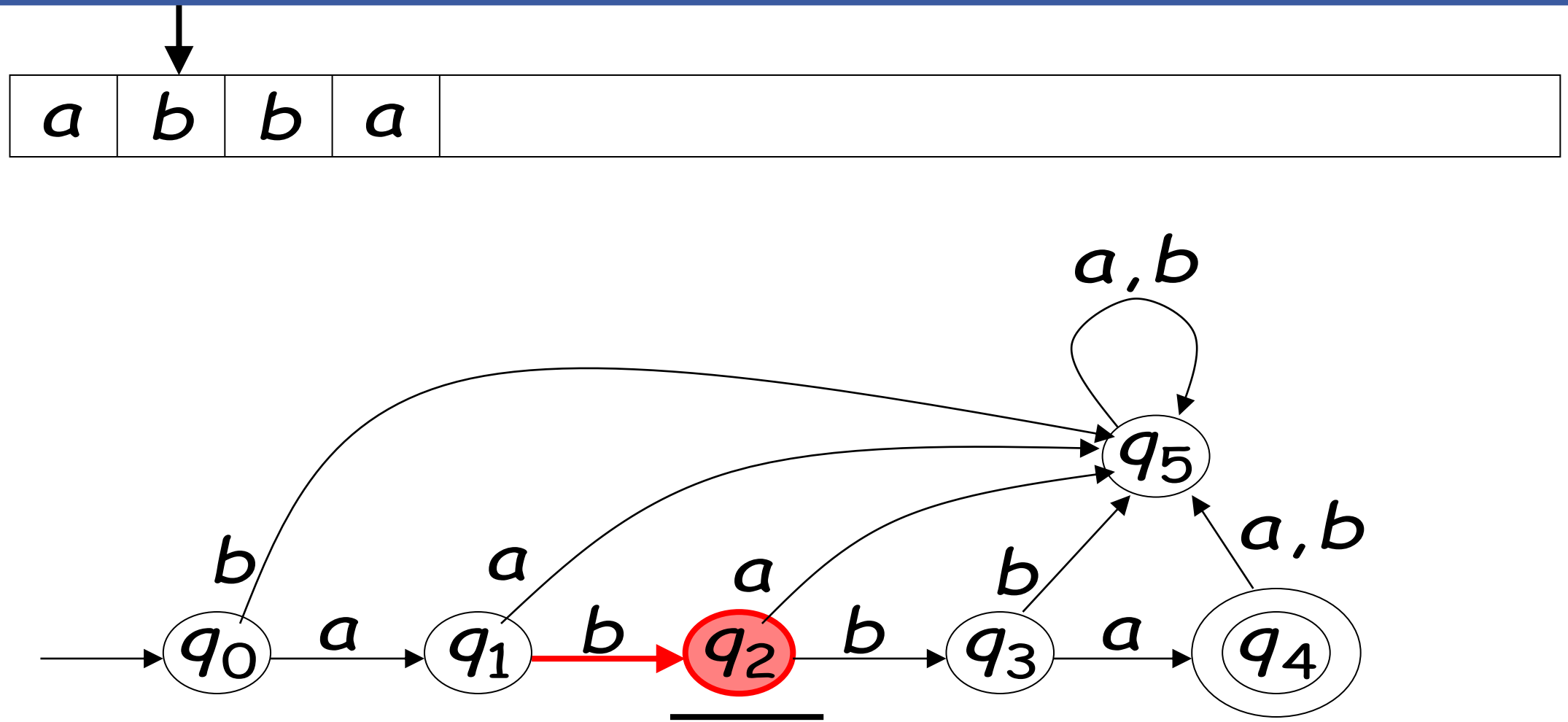


Initial state

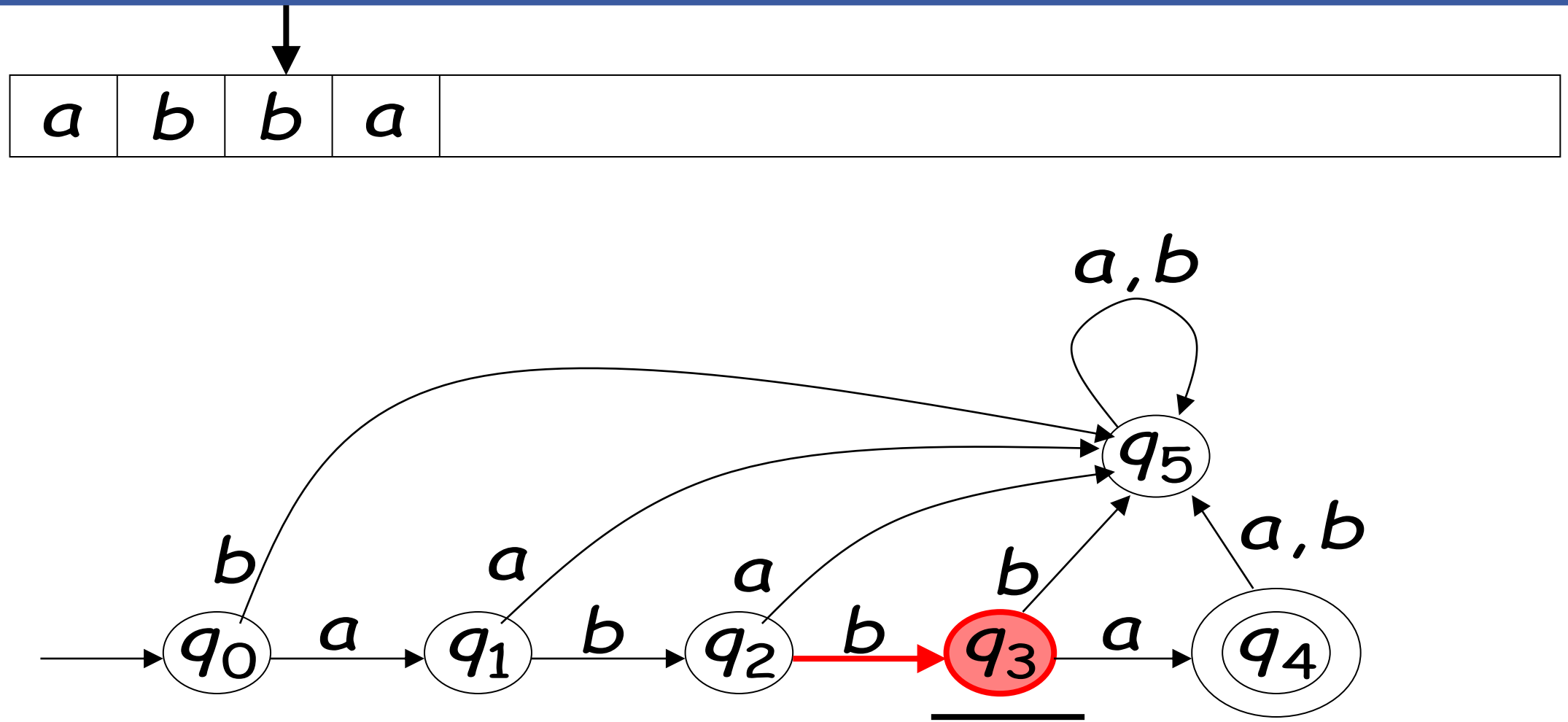
HOW FA WORKS ?



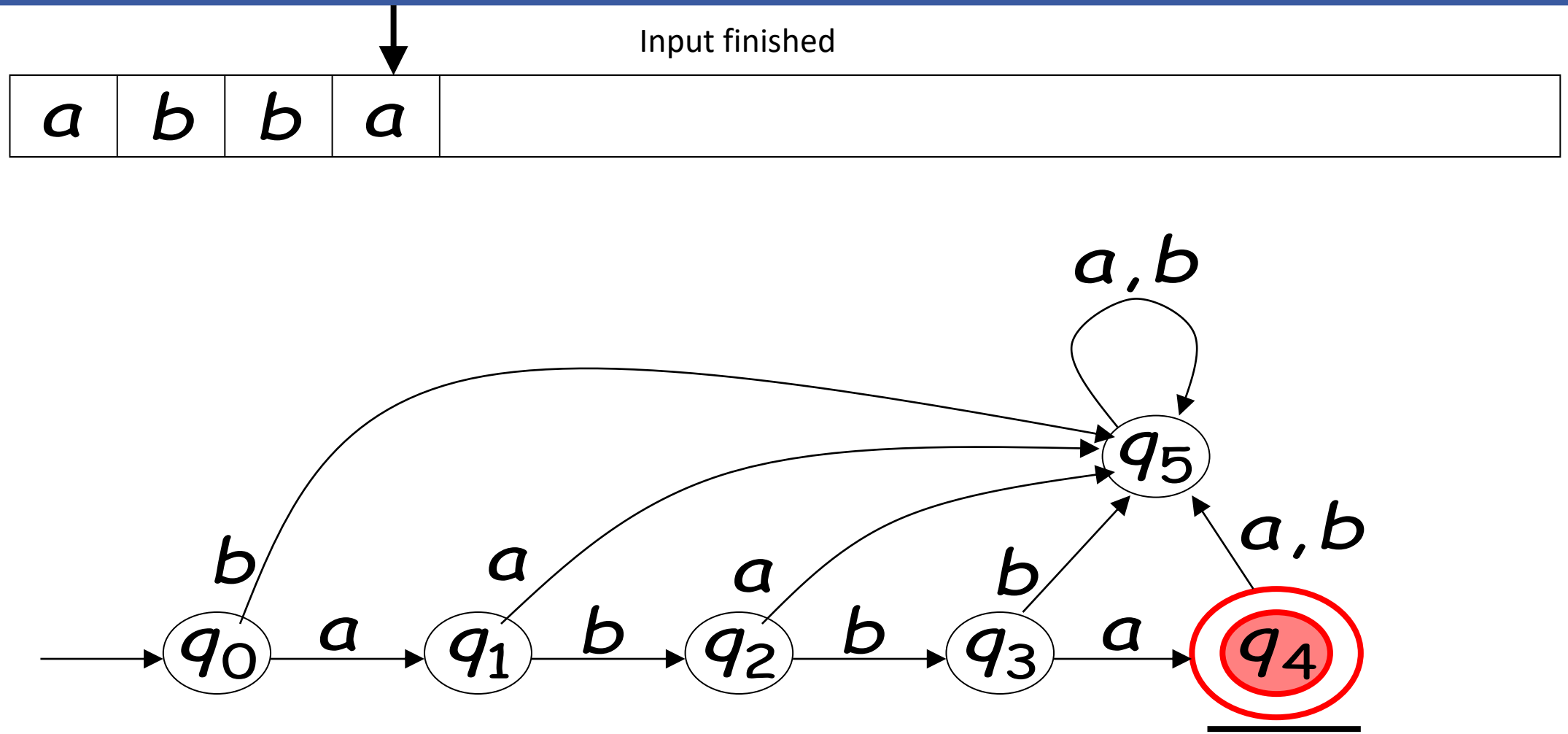
HOW FA WORKS ?



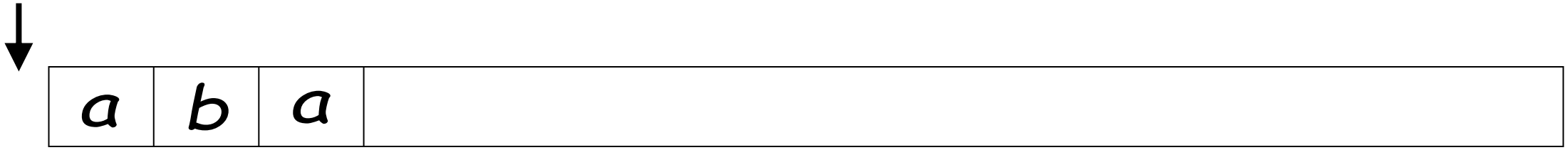
HOW FA WORKS ?



HOW FA WORKS ?

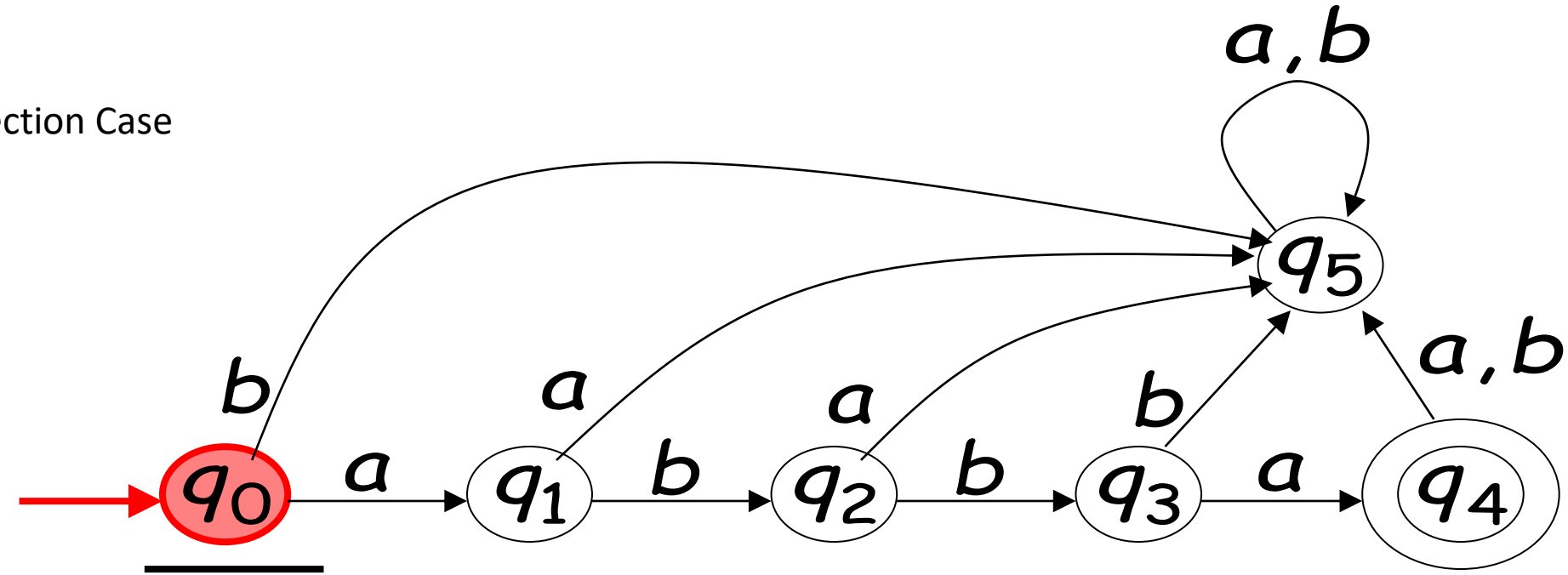


HOW FA WORKS ?

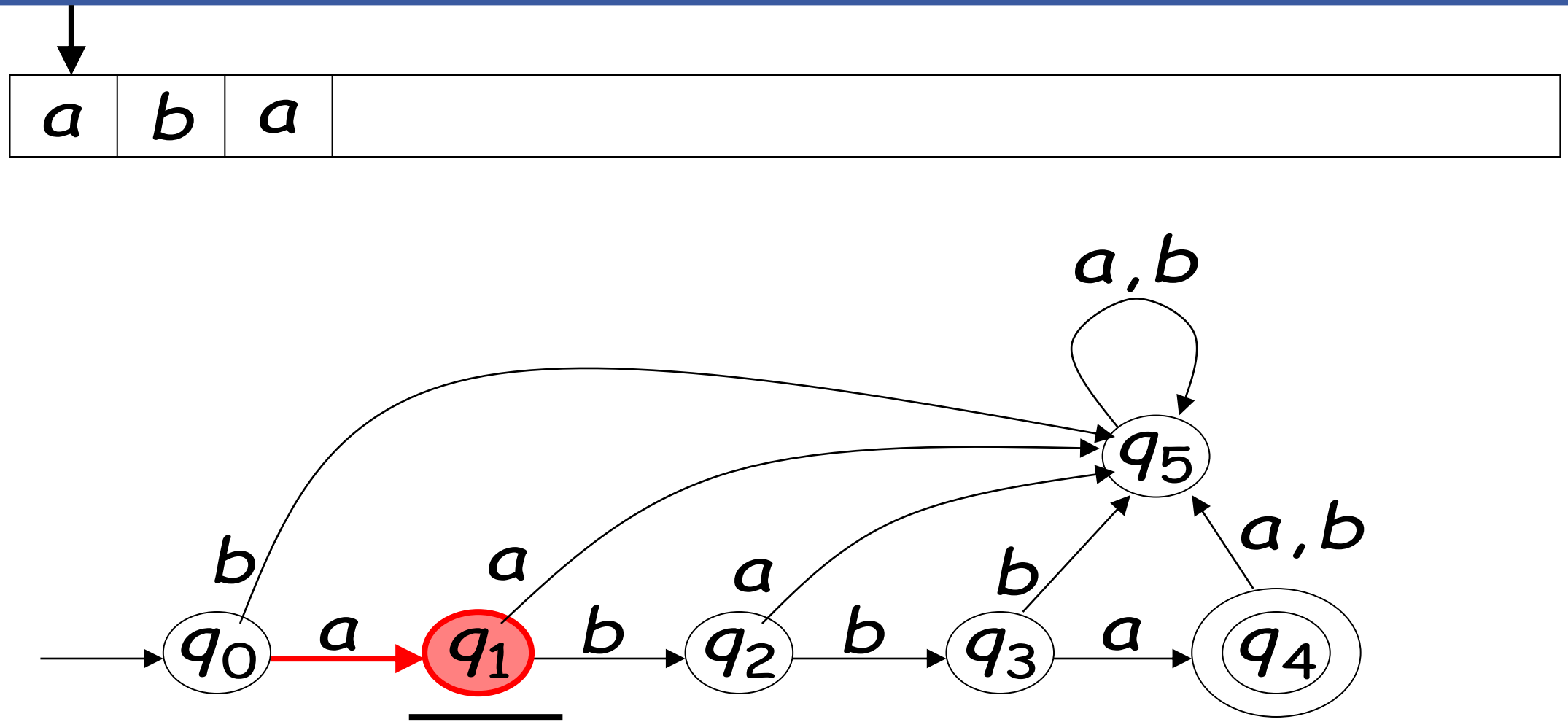


Input String

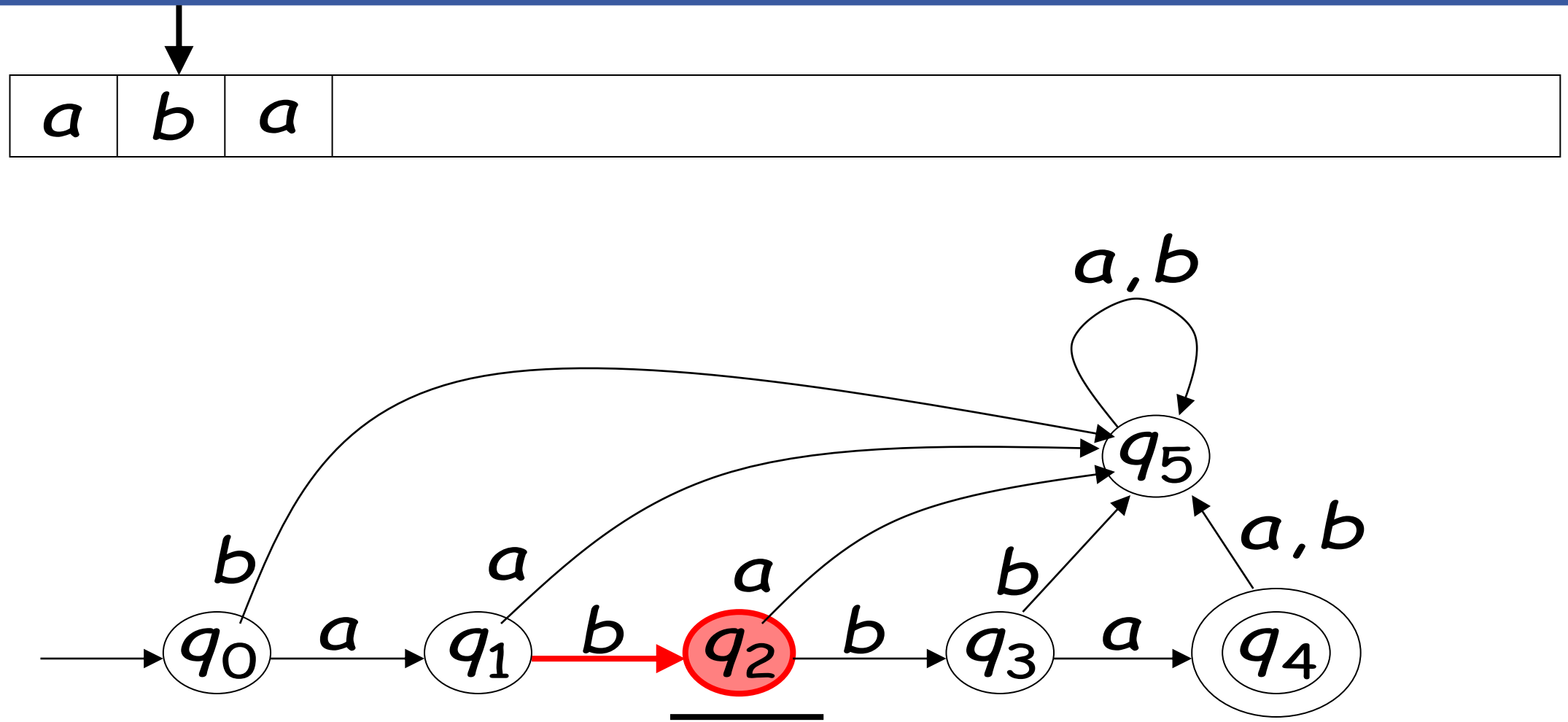
A Rejection Case



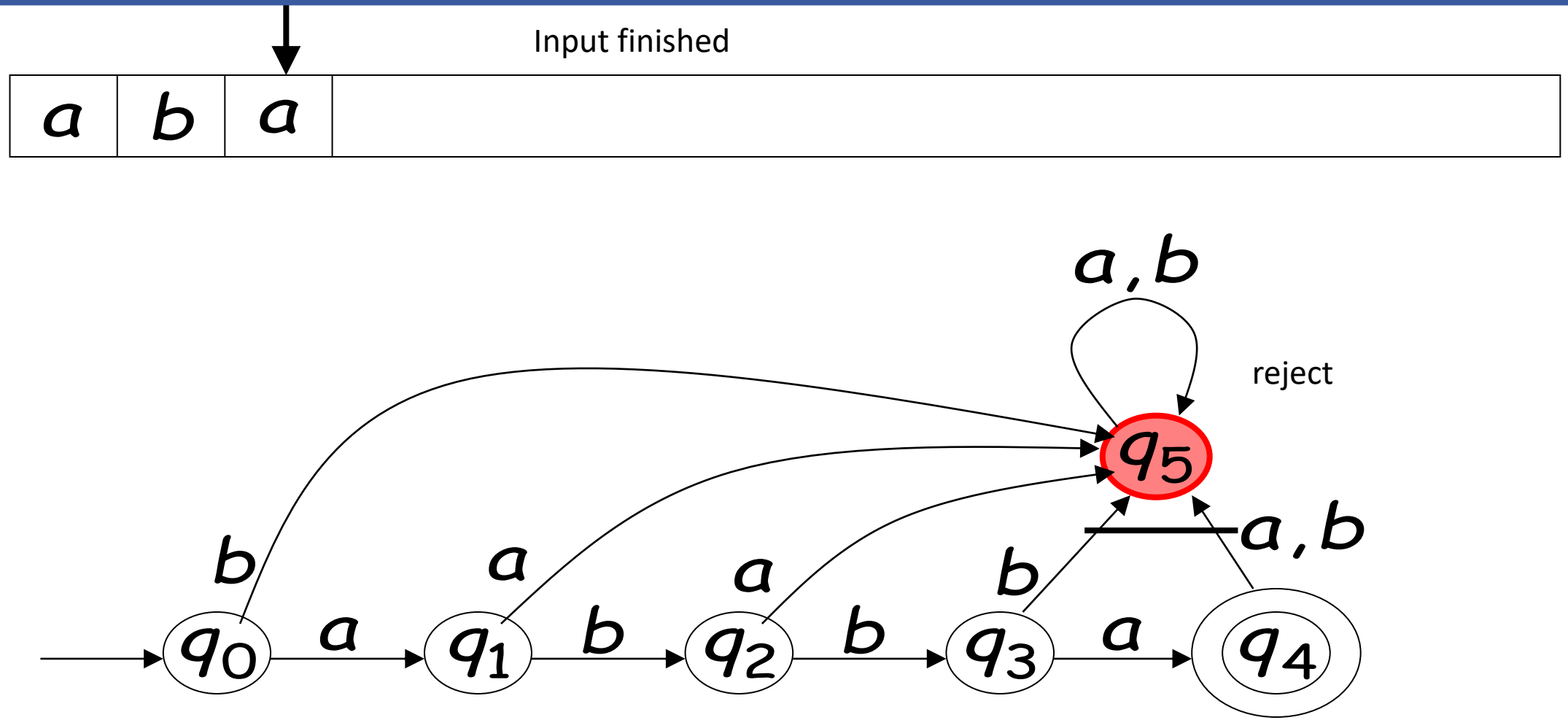
HOW FA WORKS ?



HOW FA WORKS ?



HOW FA WORKS ?



LANGUAGE OF A FA

—

APPLICATIONS OF FA

- It is an useful tool in the design of Lexical analyzer - a part of compiler that groups characters into tokens, indivisible units such as variable name and keyword.
- Text editor
- Pattern matching
- File searching program
- Text processing (searching an occurrence of one string in a file)

LIMITATIONS

- It can recognise only simple languages (regular)
- FA can be designed only for decision making problems.

REFERENCE

- Hopcroft J.E., Motwani R. and Ullman J.D, “Introduction to Automata Theory, Languages and Computations”, Second Edition, Pearson Education, 2008