UNIVERSITAS DIPONEGORO
The Excellent Research University

SI INFORMATIKA
Fakultas Sains dan Matematika

Bermartabat & Bermanfaat

DIKTISAINTEK BERDAMPAK

Bilingual

Formal Languages and Automata Theory
-Teori Bahasa dan Otomata-

Minggu Ke-2

Departemen Informatika, Fakultas Sains dan Matematika
Universitas Diponegoro

Formal Languages and Automata Theory
-Teori Bahasa dan Otomata-

Sub CPMK05-2 (1):

Mampu menjelaskan (C2) konsep simbol, kata, tata bahasa, dan jenis-jenis otomata serta Formal Proof terhadap Regular Language

Bahan Kajian / Materi Pembelajaran:

Teori Finite Automata : konsep simbol, kata dan tata bahasa, dan jenis-jenis otomata

Indikator :

Ketepatan mendefinisikan simbol, kata, dan tata bahasa dalam Finite Automata

Overview/Outline Week-2

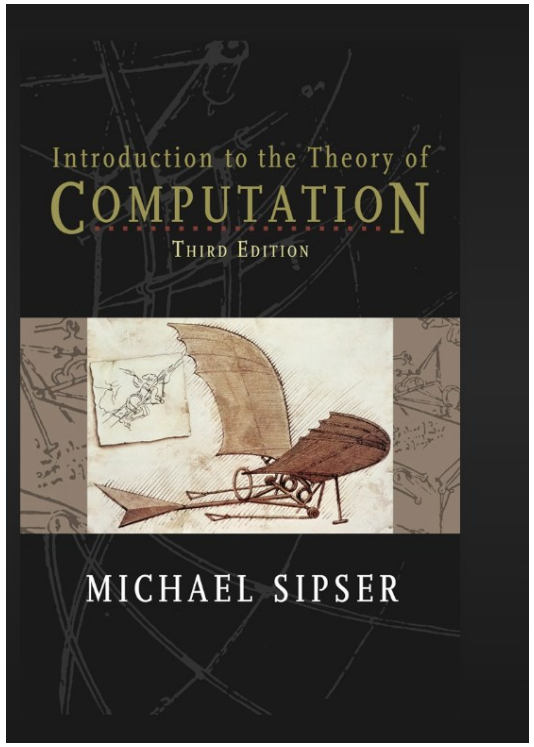
I. Motivation and Review

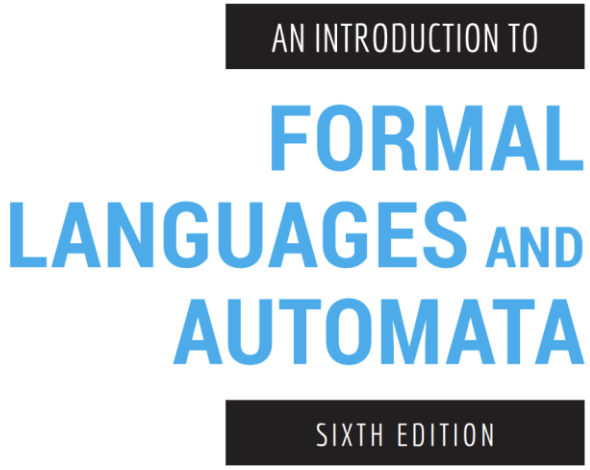
II. Certain Formal and Technical Proof

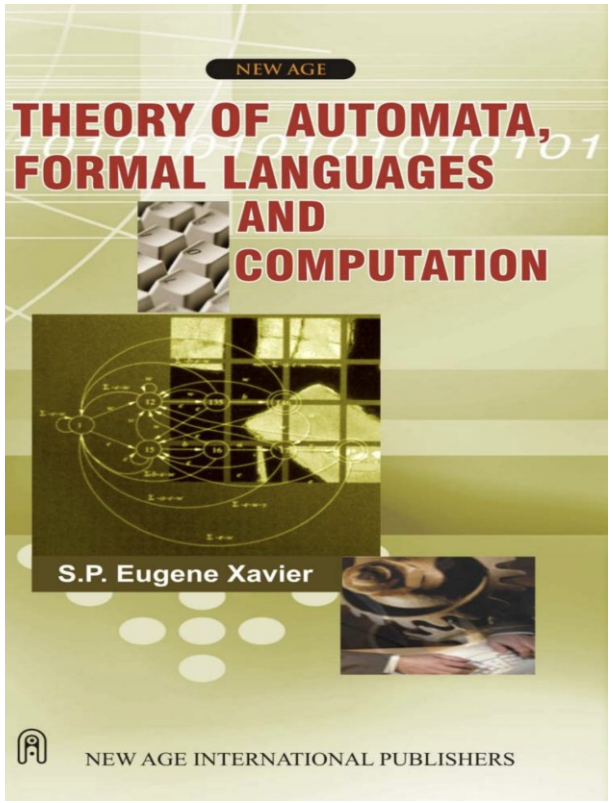
III. DFA Machine (Definition and Example)

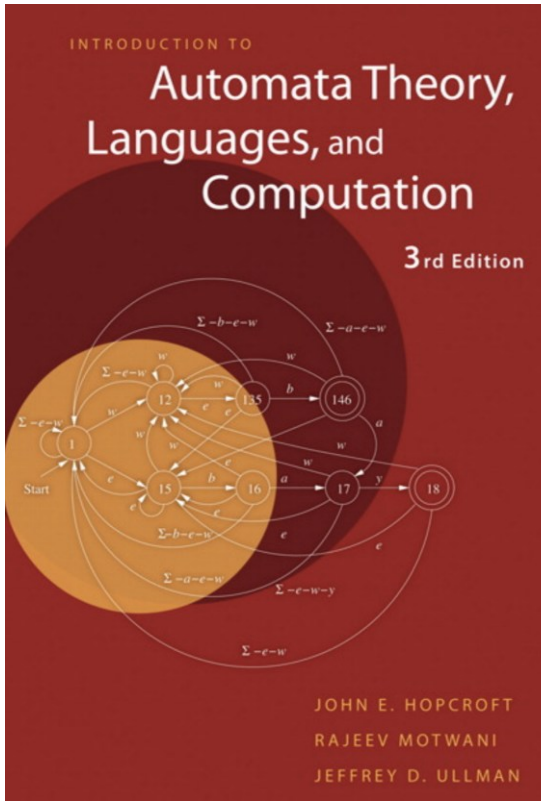
IV. How to construct DFA Machine

References









Silahkan cek di MS Teams, tab 'Files'

I. Motivation and Review

This class is about mathematical models of computation

Motivation/Overview

- At the heart of programs lie algorithms

- To study algorithms we must be able to speak ***mathematically*** about:
  - computational problems
  - computers
  - algorithms

Mathematical Models of Computation

(predated computers as we know them)

Automata and Languages: (1940's)

finite automata, regular languages, context-free languages, pushdown automata, pumping lemmas.

Computability Theory: (1930's-40's)

Turing Machines, decidability, reducibility, the arithmetic hierarchy, the recursion theorem, the Post correspondence problem.

Complexity Theory and Applications: (1960's-70's)

time complexity, classes P and NP, NP-completeness, space complexity PPACE, PSPACE-completeness, the polynomial hierarchy, randomized complexity, classes RP and BPP.

What is a problem?

- Some examples:
  - > given n integers, produce a sorted list
  - > given a graph and nodes s and t, find the shortest path from s to t
  - > given an integer, find its prime factors
- problem associates each input to an output
- input and output are strings over a finite *alphabet* Σ

What is a problem?

- A problem is a function:

$$f:\Sigma^* \rightarrow \Sigma^*$$

- Simple. Can we make it simpler?
- Yes. Decision problems:

$$f:\Sigma^* \rightarrow \{\text{accept}, \text{reject}\}$$

- Does this still capture our notion of problem, or is it too restrictive?

What is a problem?

For most of this course, a problem is a decision problem:

$$f:\Sigma^* \rightarrow \{\text{accept, reject}\}$$

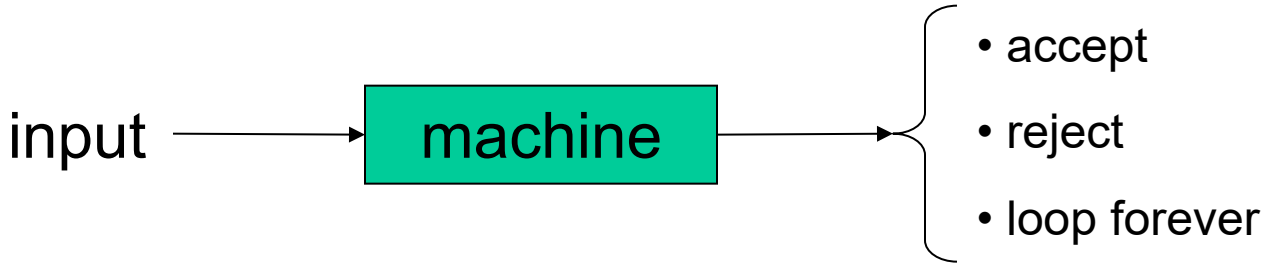
Equivalent notion: language

$$L \subseteq \Sigma^*$$

the set of strings that map to "accept"

Example: L = set of pairs (m,k) for which m has a prime factor p < k

What is computation?



- the set of strings that lead to "accept" is the language recognized by this machine

- if every other string leads to "reject", then this language is decided by the machine

Terminology

finite alphabet Σ : a set of symbols

language $L \subseteq \Sigma^*$: subset of strings over Σ

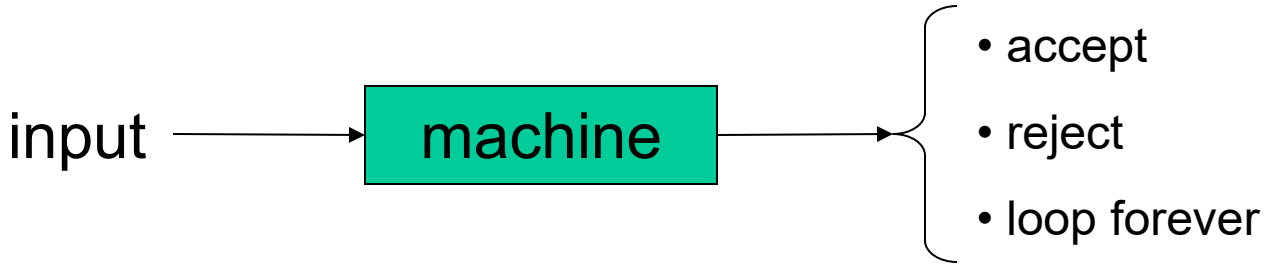
a machine takes an input string and either

- accepts, rejects, or
- loops forever

a machine recognizes the set of strings that lead to accept

a machine decides a language L if it accepts $x \in L$ and rejects $x \notin L$

What goes inside the box?

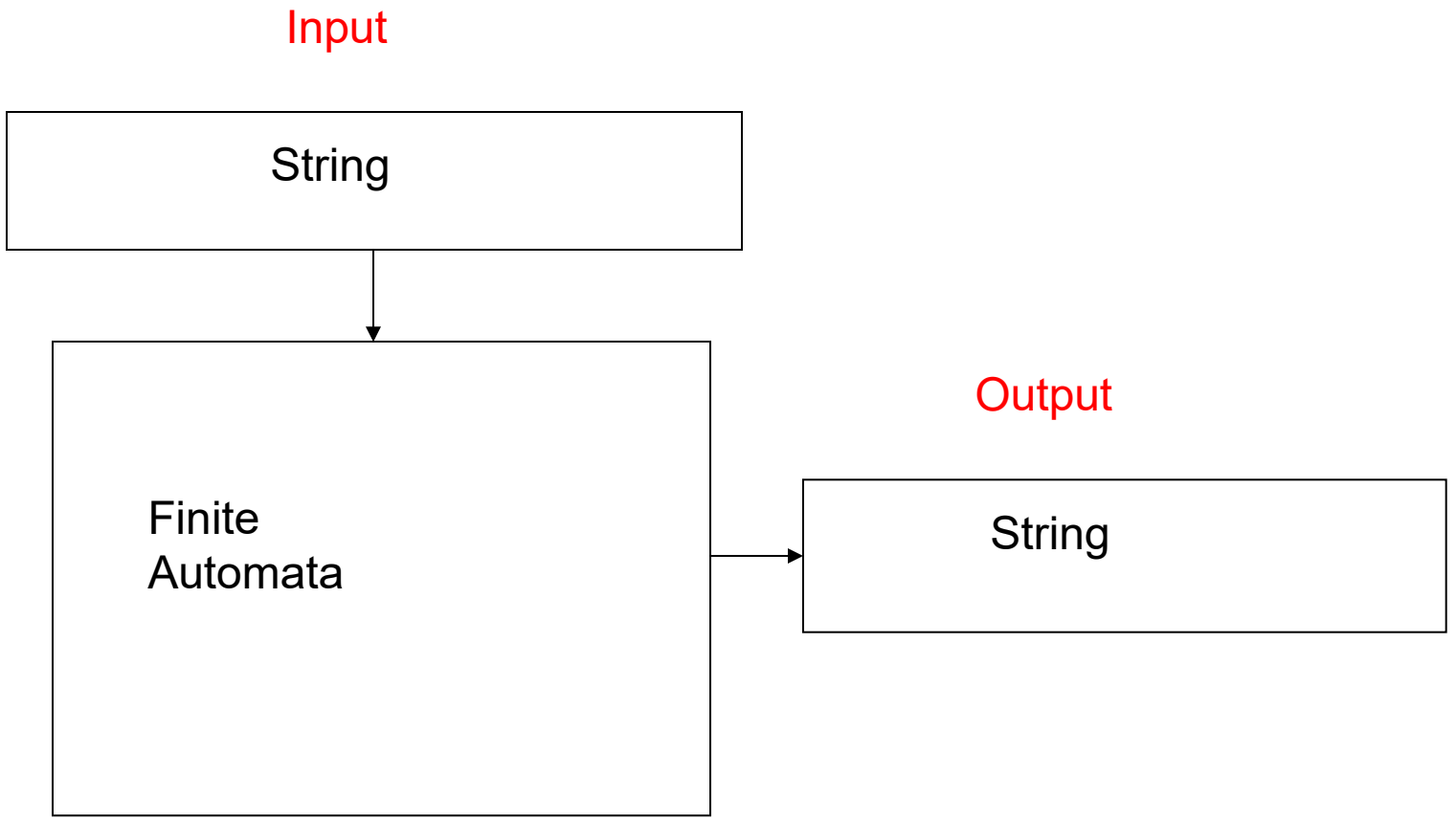


- We want the simplest mathematical formalization of computation possible.
- Strategy:
  - endow box with a feature of computation
  - try to characterize the languages decided
  - identify language we "know" real computers can decide that machine cannot
  - add new feature to overcome limits

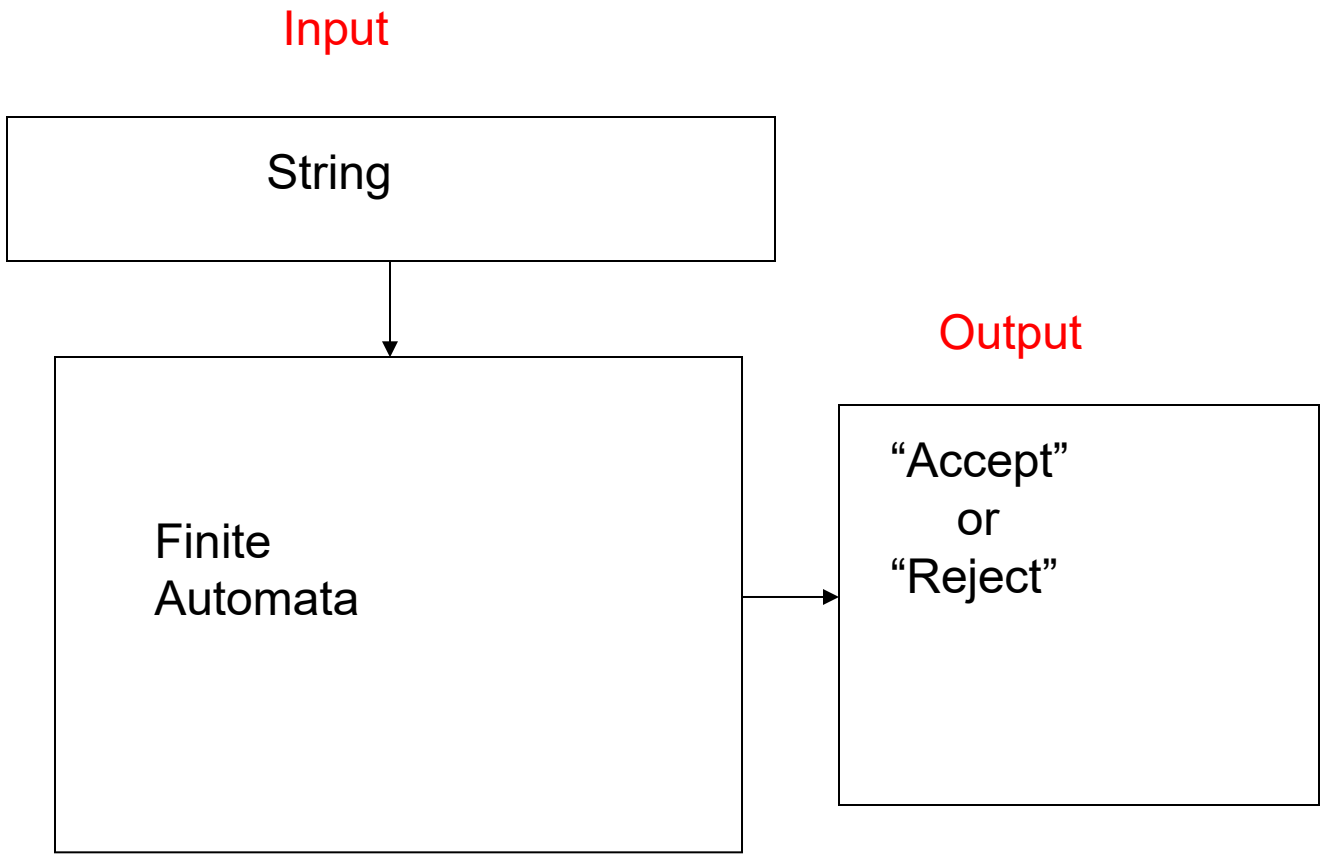
Finite Automata

- simple model of computation
- reads input from left to right, one symbol at a time
- maintains state: information about what seen so far ("memory")
    - >finite automaton has finite # of states: cannot remember more things for longer inputs
- 2 ways to describe: by diagram, or formally

Finite Automata



Finite Accepter



Different Kinds of Automata

Automata are distinguished by the temporary memory

- **Finite Automata**: no temporary memory
- **Pushdown Automata**: stack
- **Turing Machines**: random access memory

Automata vs Bahasa (Language)

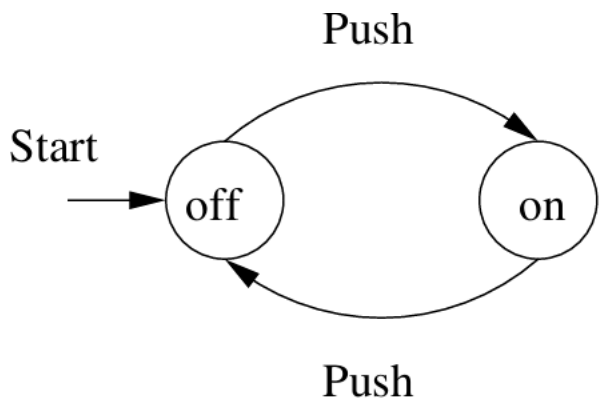
Automata	Bahasa
Finite Automata	Regular Language
Push Down Automata	Context Free Language
Turing Machine	Unrestricted Language

Soal Latihan

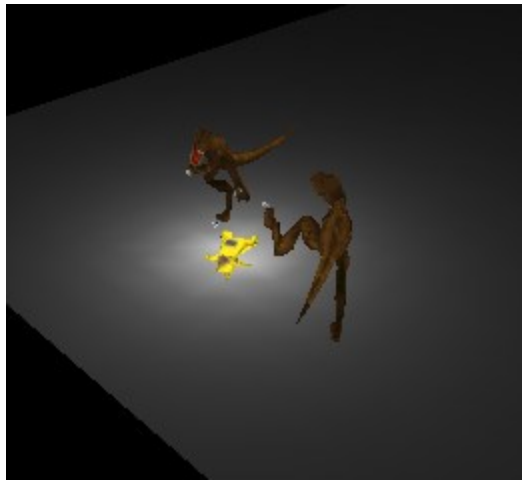
Buatlah skema(flowchart atau pseudocode) untuk masalah berikut:

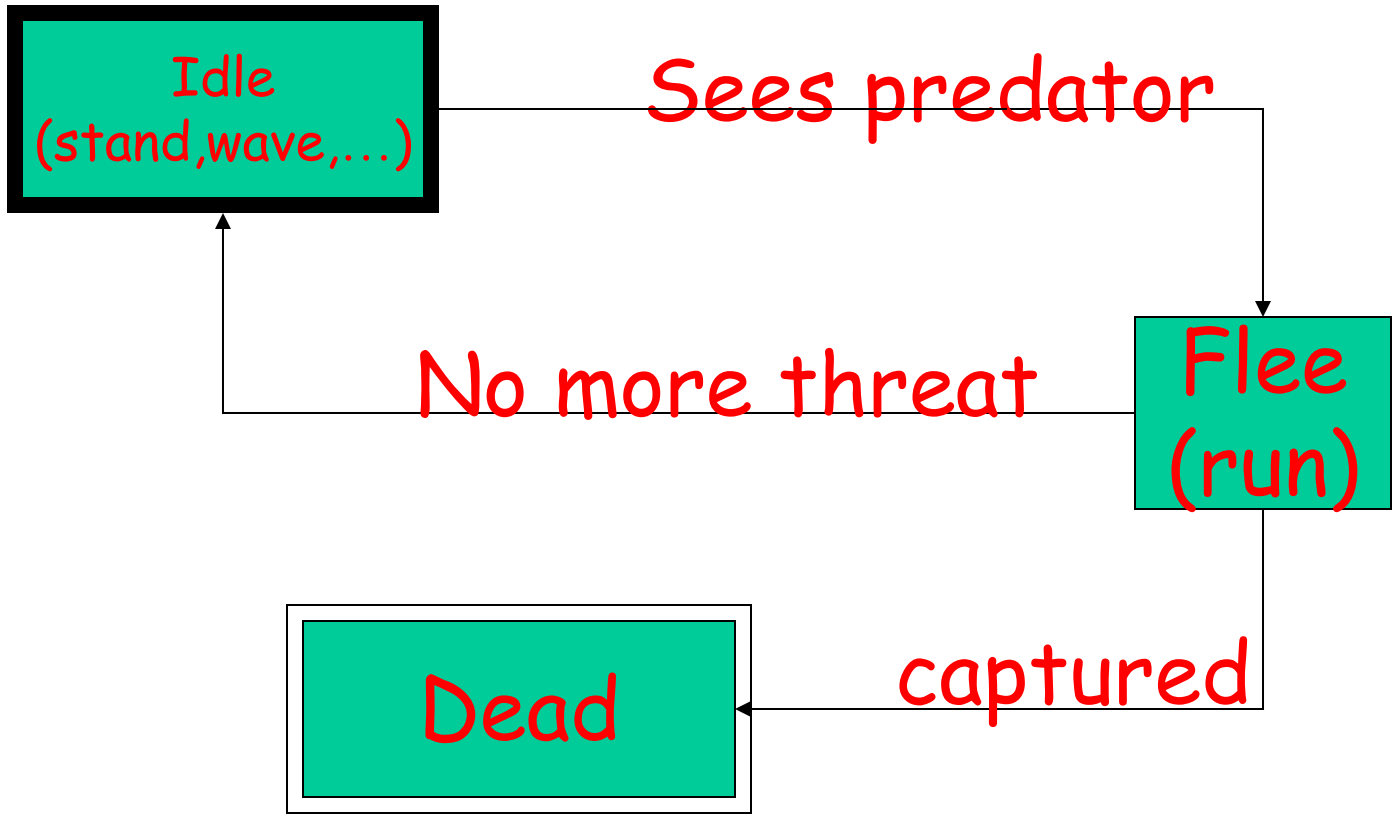
Terdapat mangsa (prey) dan pemangsa (predator). Predator bisa memakan mangsa, atau tidak berhasil memakan mangsa.

Model Automata/Formal Language dapat digunakan sebagai recognizer dan generator



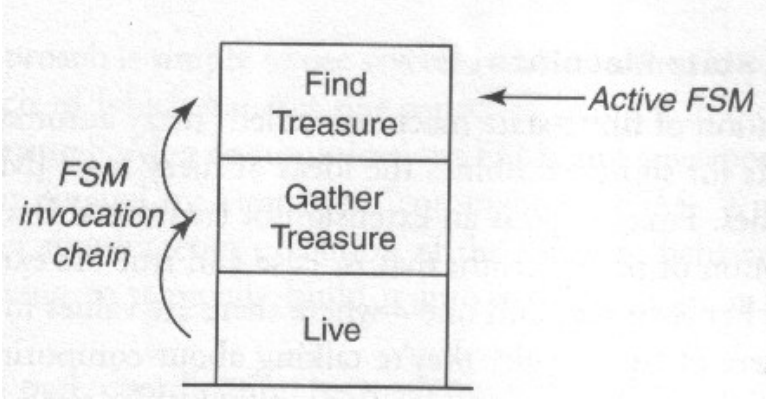
predator vs. prey

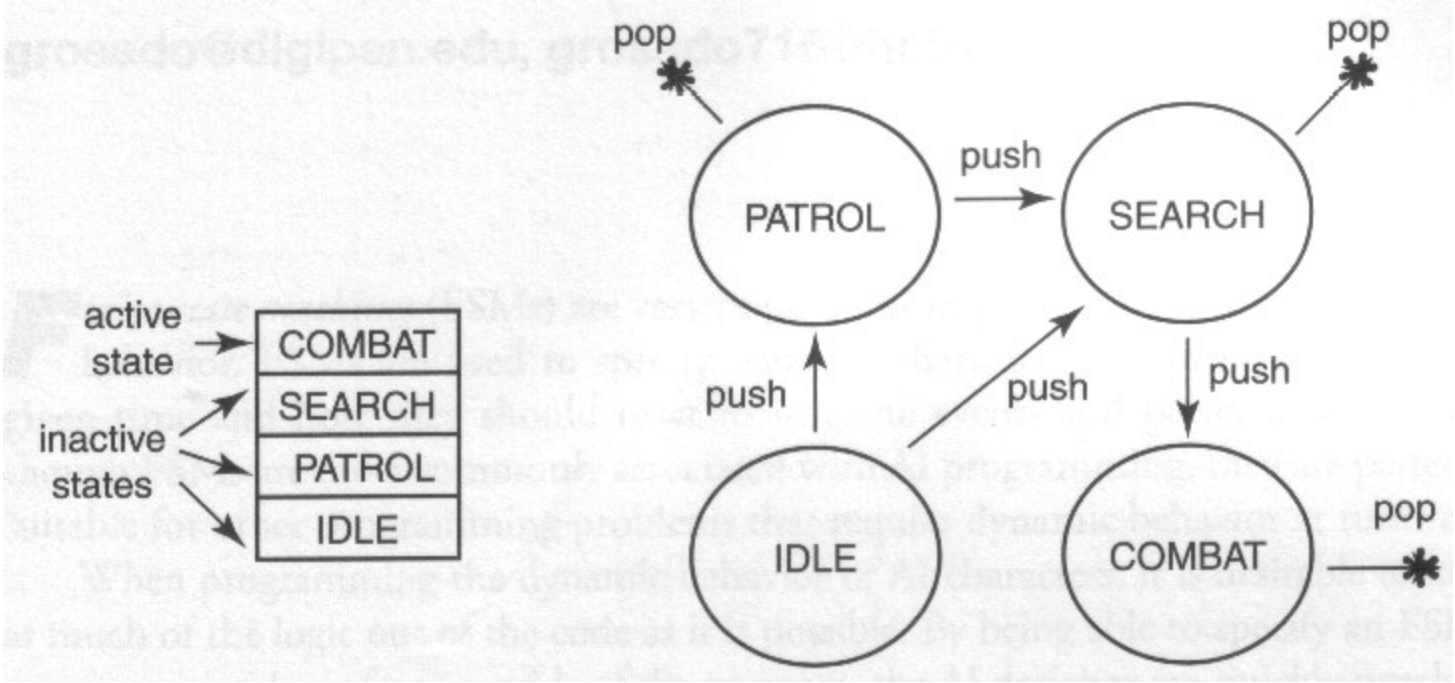




Stack-based FSM

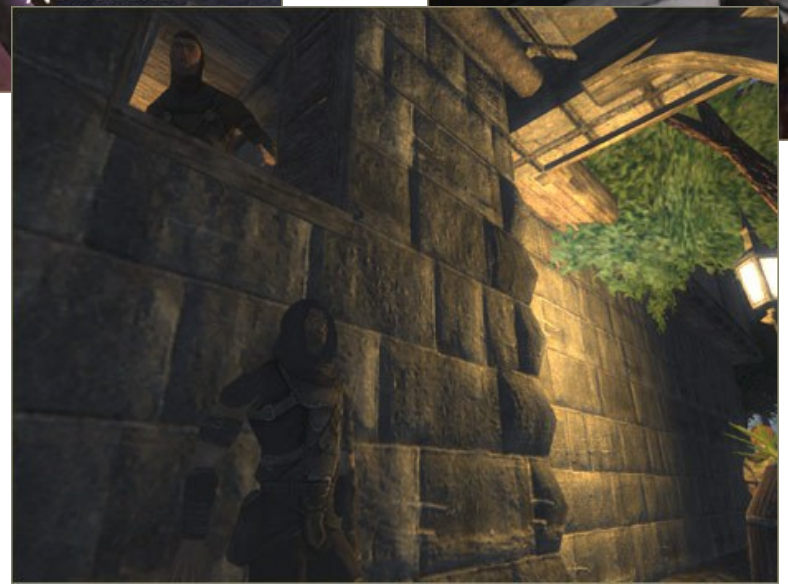
Pushdown automaton (PDA)





History stack: Remember previous state; create characters with a memory …

Stack FSM – Thief 3



Stack allows AI to move back and forth between states.

Leads to more realistic behavior without increasing FSM complexity.

II. Certain Formal and Technical Proof

Quantifiers

Universal Quantifier

Semua bilangan genap adalah kelipatan 2

$\forall x \in Z$, x adalah bilangan Genap$\rightarrow$ x mod2 = 0

Existensial Quantifier

Ada bilangan bulat yang merupakan bilangan prima ganjil

$\exists x \in Z$, x bilangan prima dan ganjil

Kombinasi Quantifier

Setiap bilangan bulat memiliki kuadrat yang tidak negatif

$\forall x \in Z, \exists y \in Z, (y = x2) \wedge (y \geq 0)$

Negasi Quantifier

?

True/False with Quantifier

?

PROOF TECHNIQUES

- Proof by induction

- Proof by contradiction

Proof by Induction

- Inductive basis

  Find $P_1, P_2, \ldots, P_b$ which are true

- Inductive hypothesis

  Let's assume $P_1, P_2, \ldots, P_k$ are true,

  for any k >= b

- Inductive step

  Show that P_{k+1} is true

Proof by Contradiction

We want to prove that a statement P is true

- we assume that P is false
- then we arrive at an incorrect conclusion
- therefore, statement P must be true

Example

Theorem: $\sqrt{2}$ is not rational

Proof:

Assume by contradiction that it is rational

$\sqrt{2}$ = n/m

n and m have no common factors

We will show that this is impossible

$\sqrt{2} = n/m$ ⇒ $2\,m^2 = n^2$

Therefore, n^2 is even ⇒ n is even

$n = 2\,k$

$2\,m^2 = 4k^2$ ⇒ $m^2 = 2k^2$ ⇒ m is even

$m = 2\,p$

Thus, m and n have common factor 2

Contradiction!

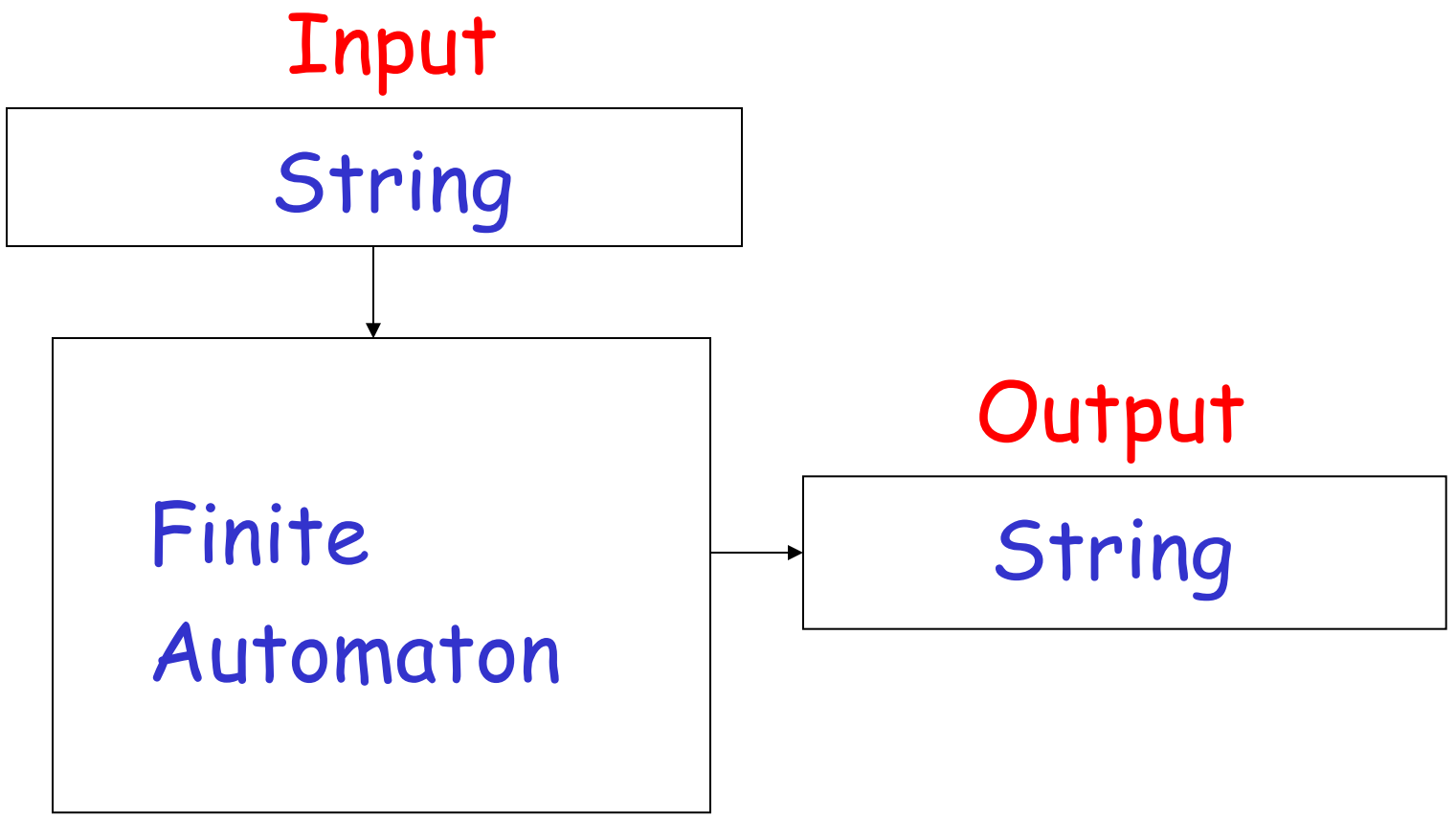
Now you try!

Prove that $1+2+3+\ldots+n = n(n+1)/2$

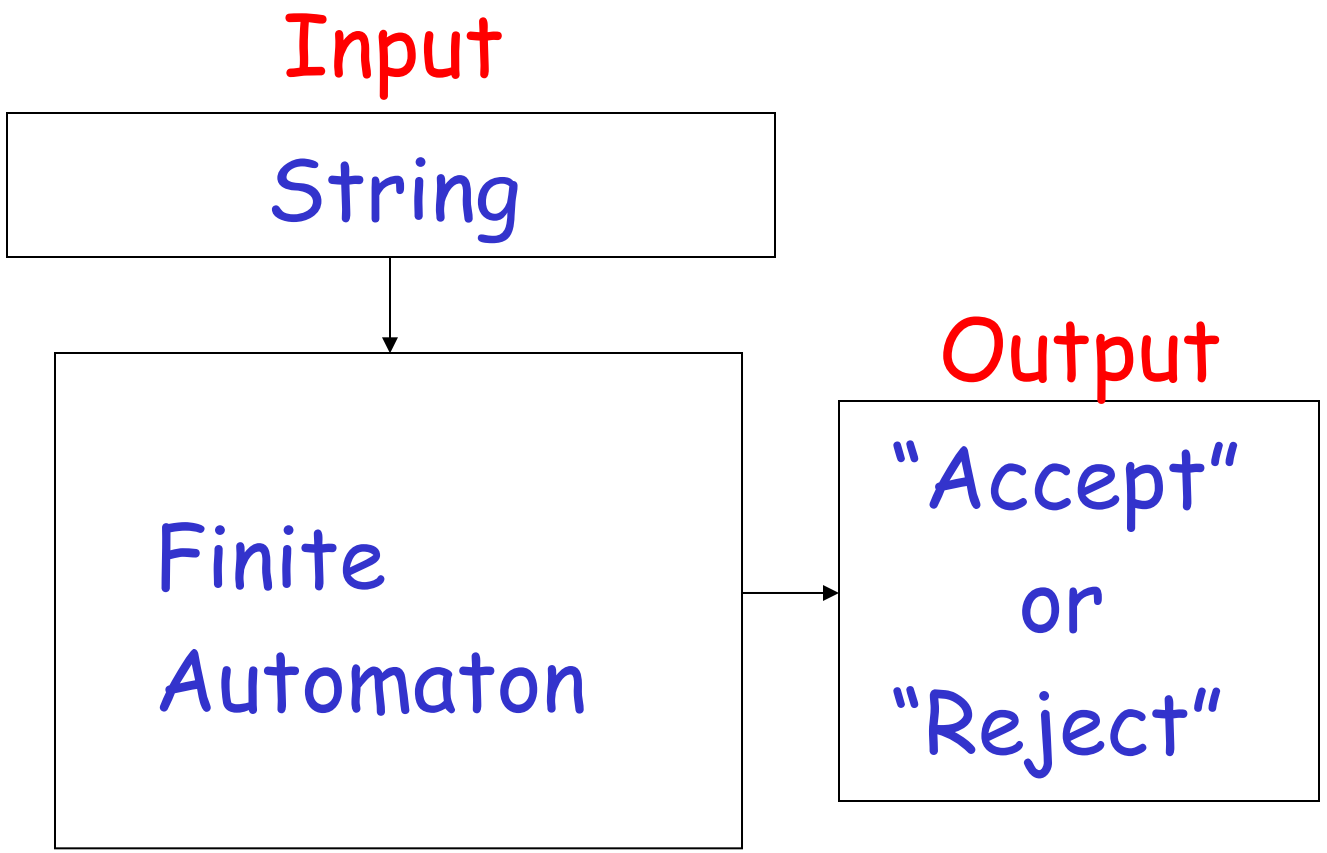
Prove that $\sqrt{3}$ is irrational

III. DFA Machine

Finite Automaton



Finite Accepter



Informal Explanation

- Finite automata are finite collections of states with transition rules that take you from one state to another.
- Original application was sequential switching circuits, where the “state” was the settings of internal bits.
- Today, several kinds of software can be modeled by FA.

Representing FA

- Simplest representation is often a graph.
  - Nodes = states.
  - Arcs indicate state transitions.
  - Labels on arcs tell what causes the transition.

III. DFA Machine

III.A Formal Definition of DFA Machine

Formalities

Deterministic Finite Automata (DFA)

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

Q : set of states

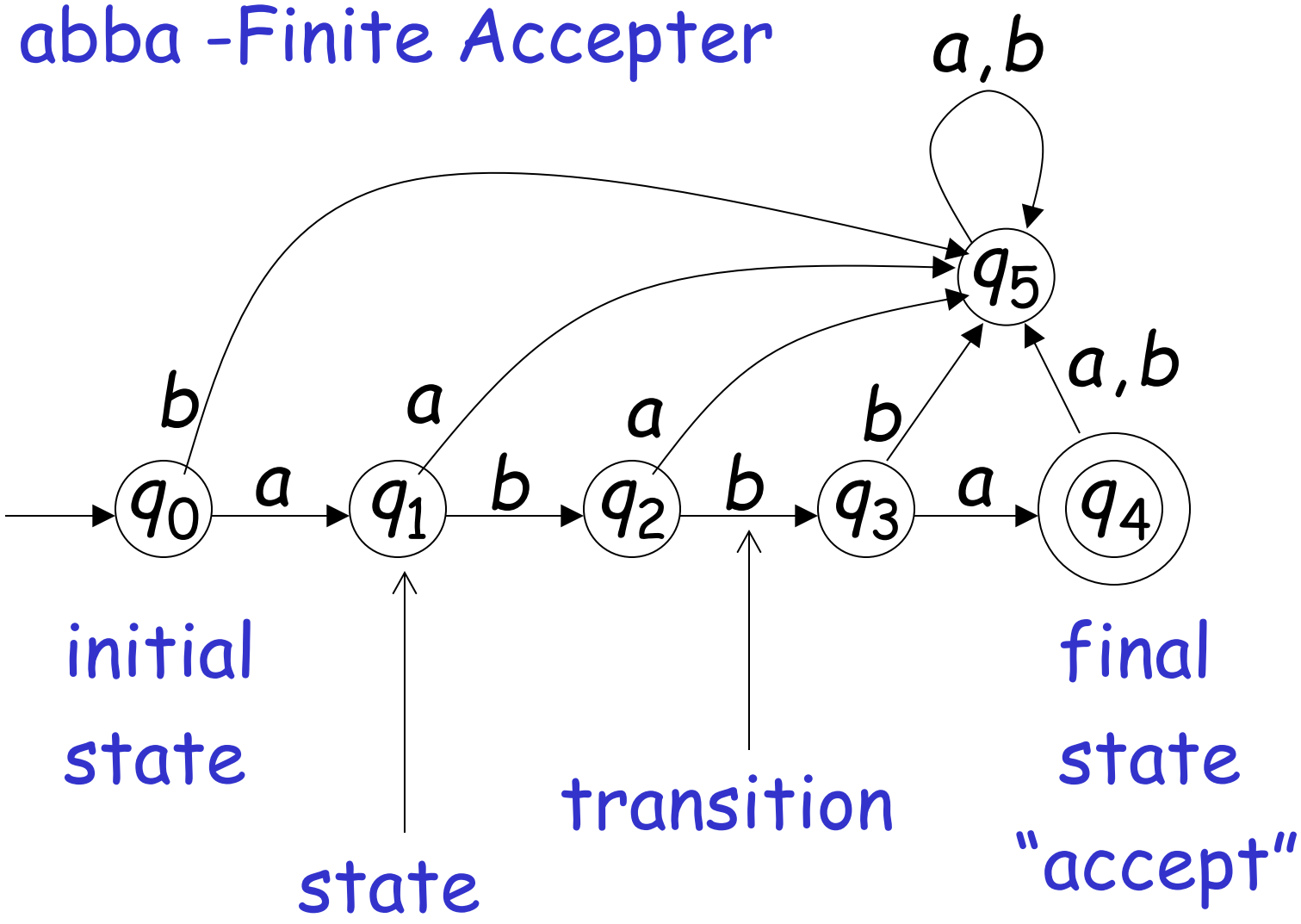
Σ : input alphabet

δ : transition function

q_0 : initial state

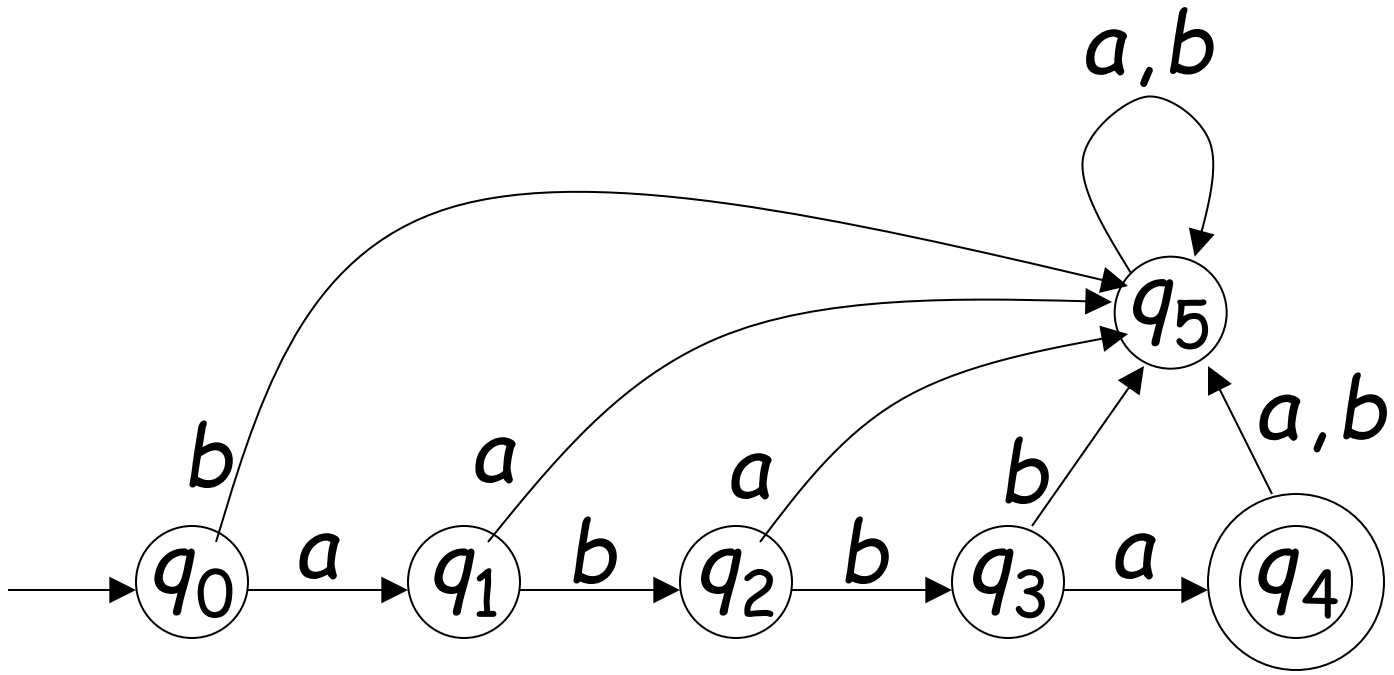
F : set of final states

Transition Graph



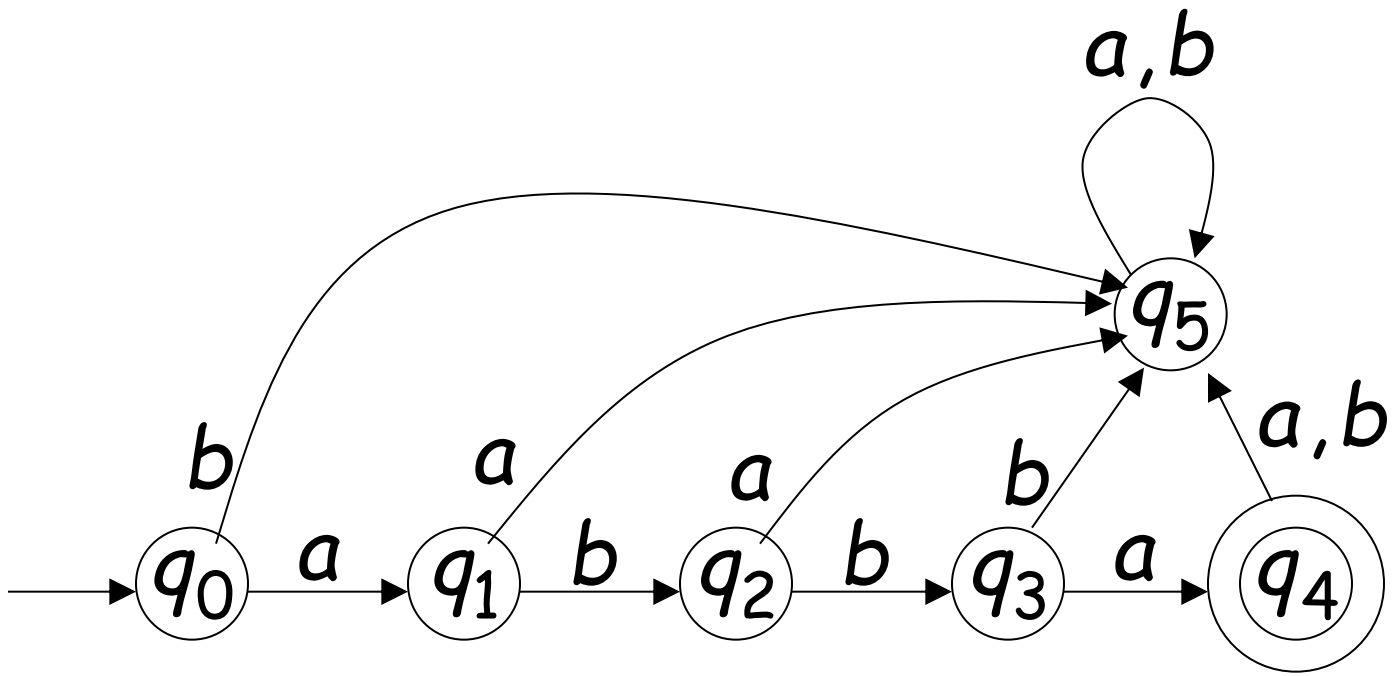
Set of States Q

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

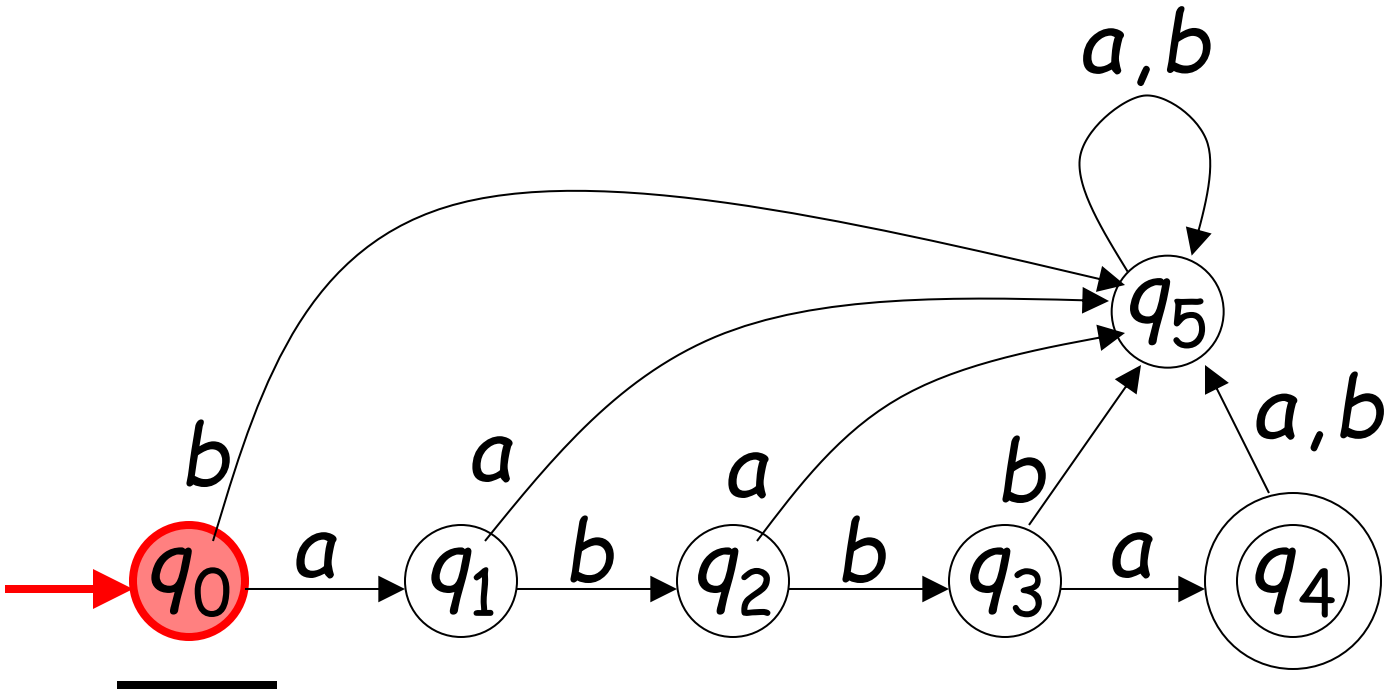


Input Alphabet Σ

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

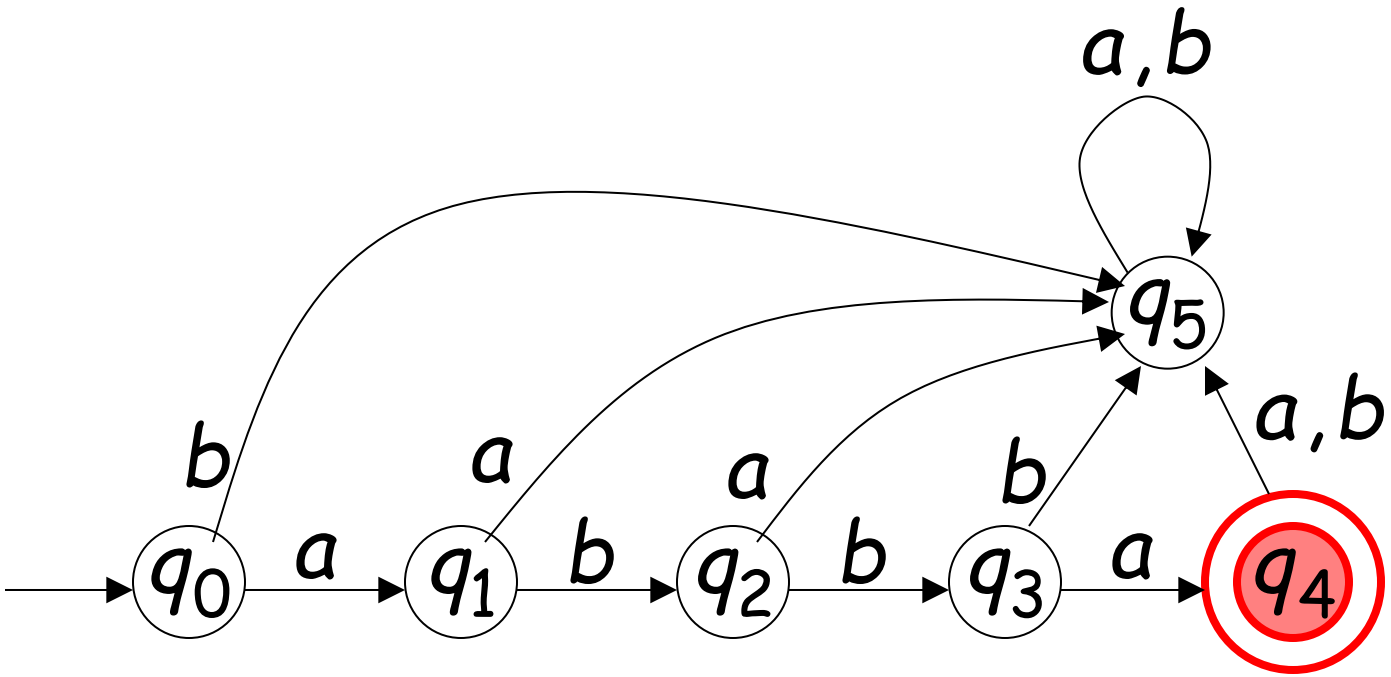


Initial State q_0



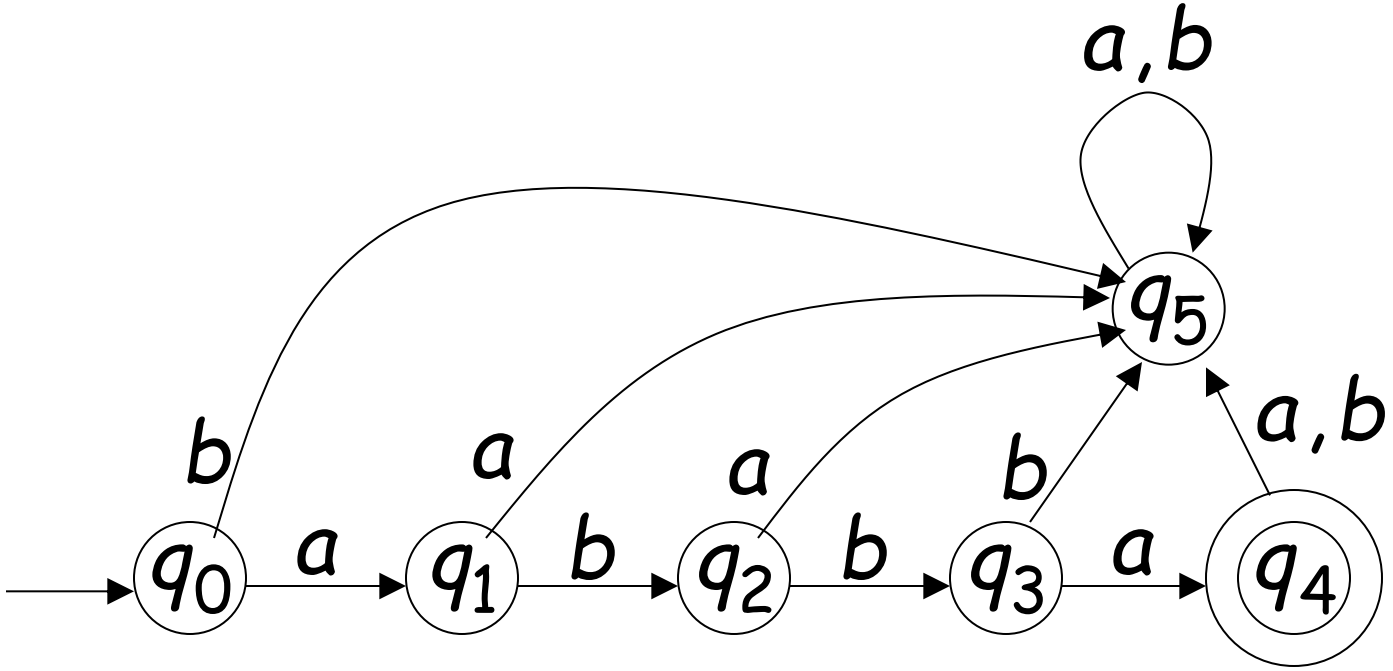
Set of Final States F

$$F = \{q_4\}$$

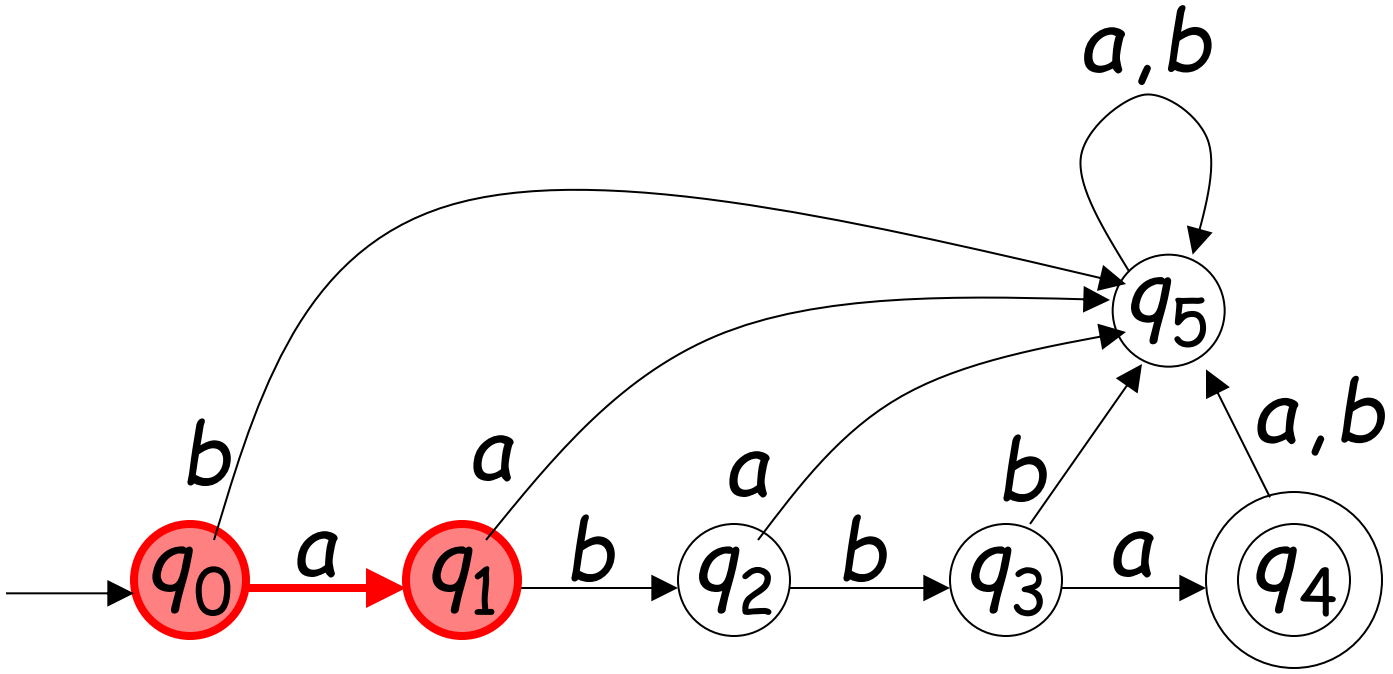


Transition Function δ

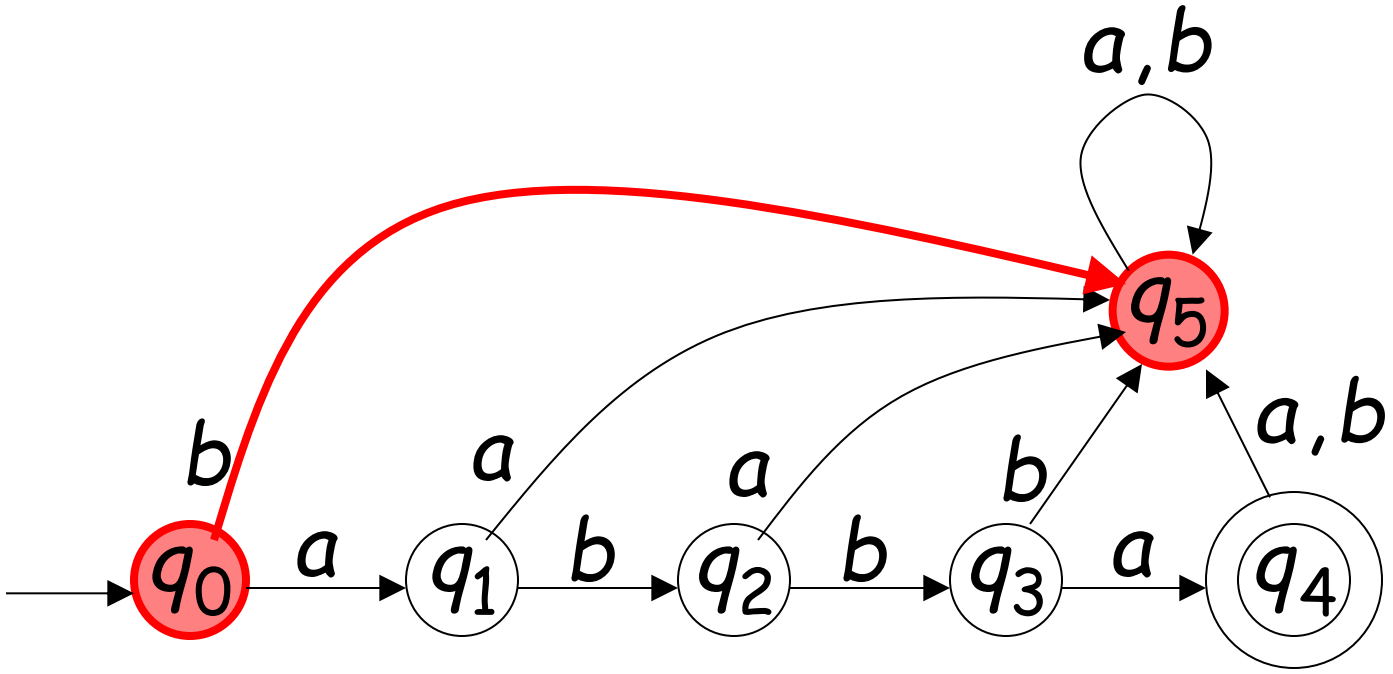
$$\delta : Q \times \Sigma \to Q$$



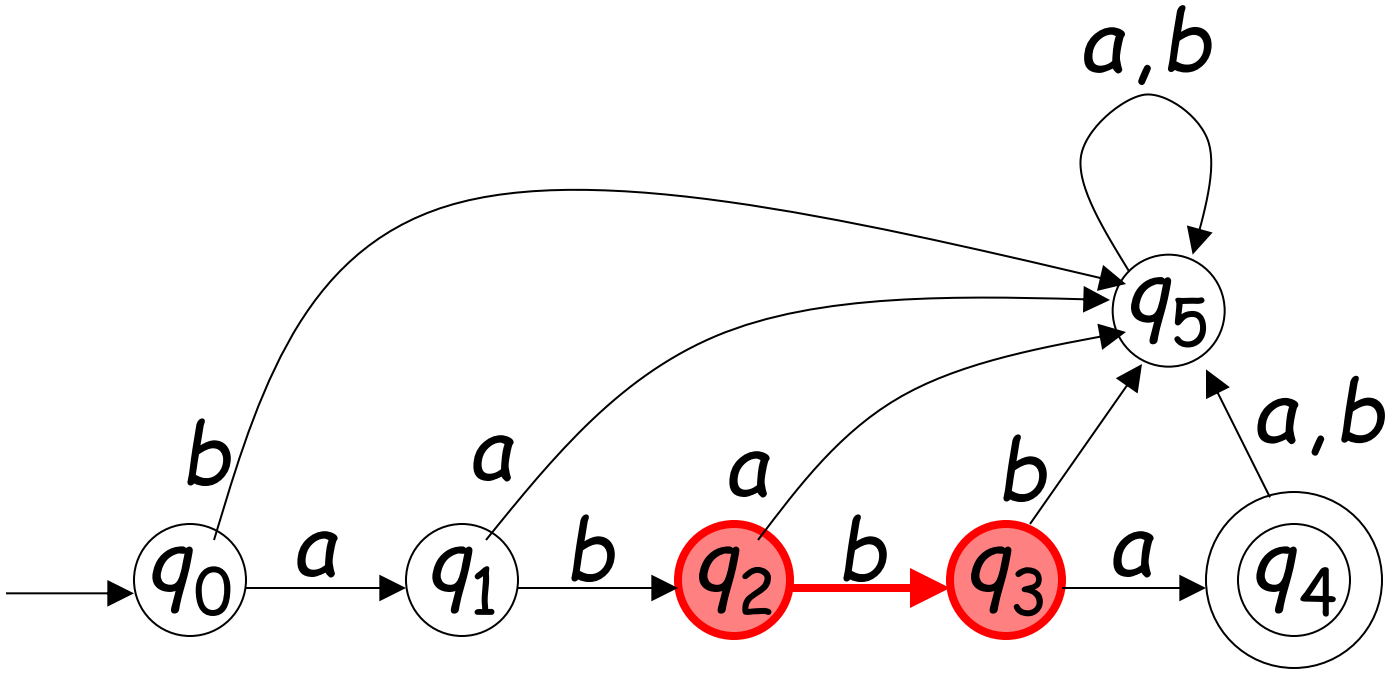
$$\delta(q_0, a) = q_1$$



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

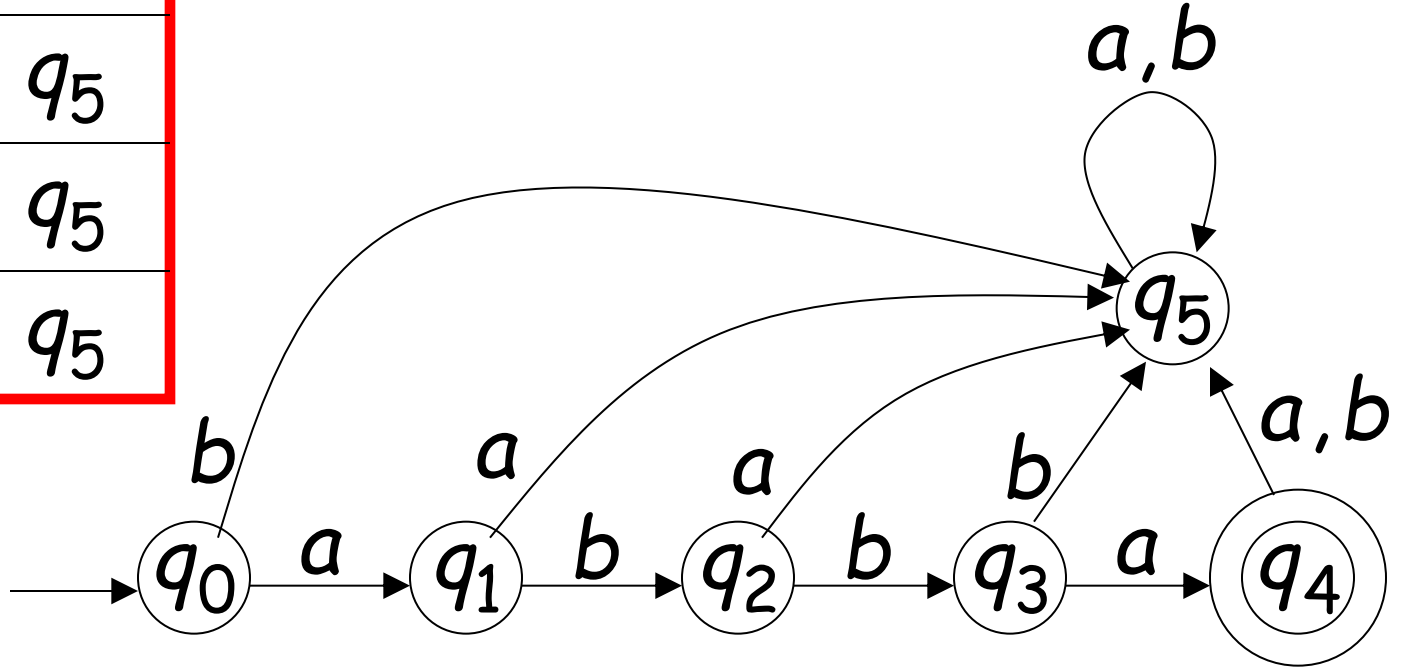


$$\delta(q_2, b) = q_3$$



Transition Function δ

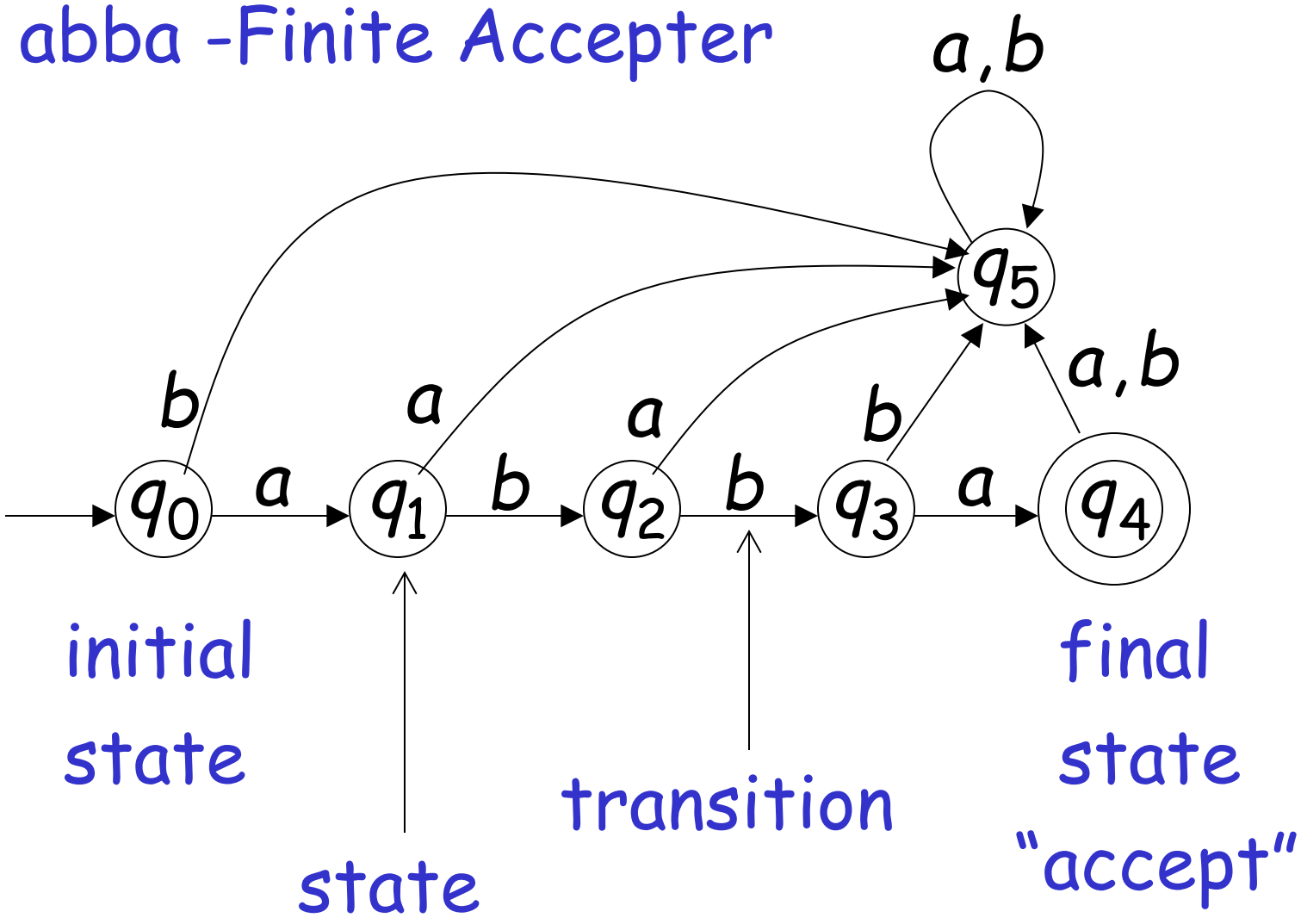
δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5



III. DFA Machine

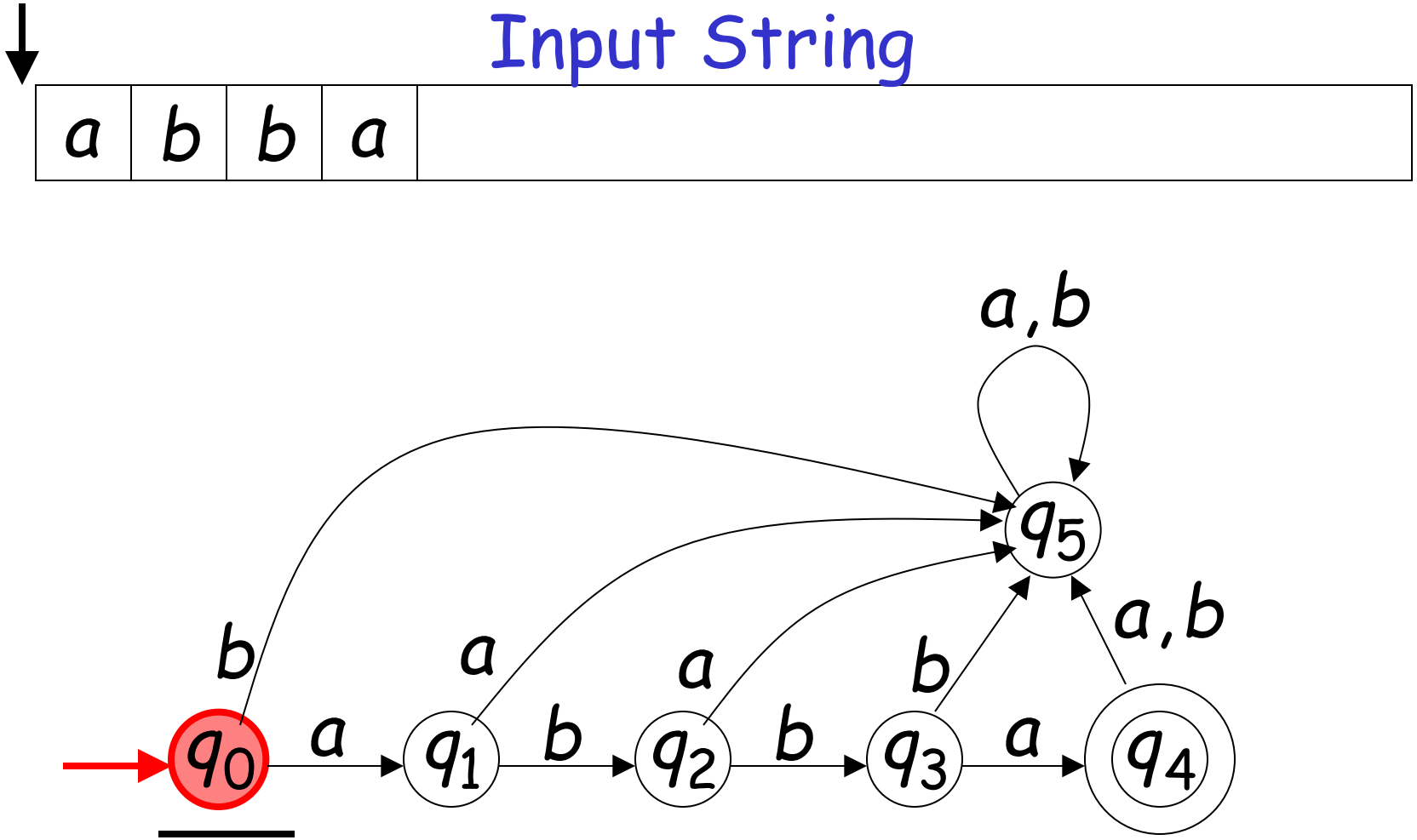
III.B How a DFA Processes String

Transition Graph

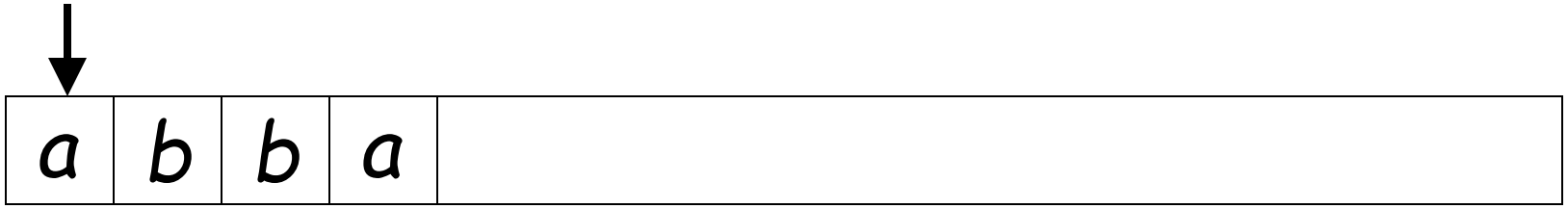


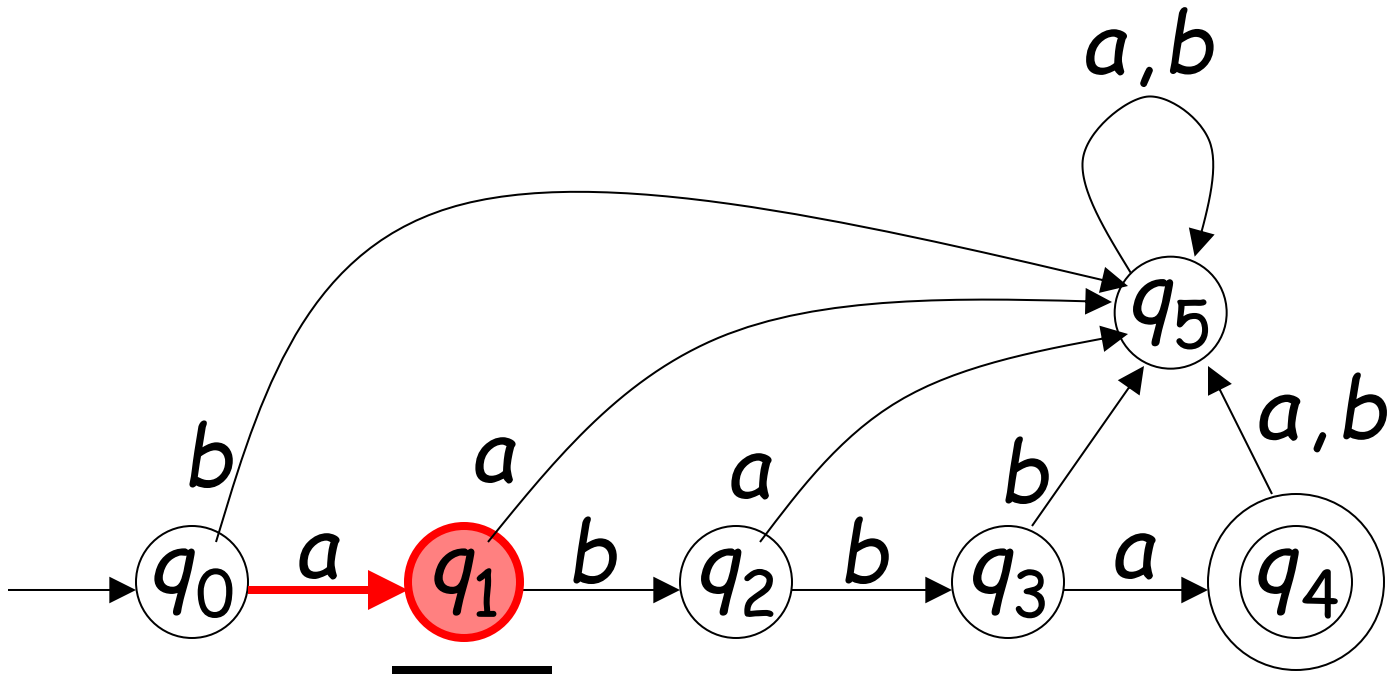
Notes:

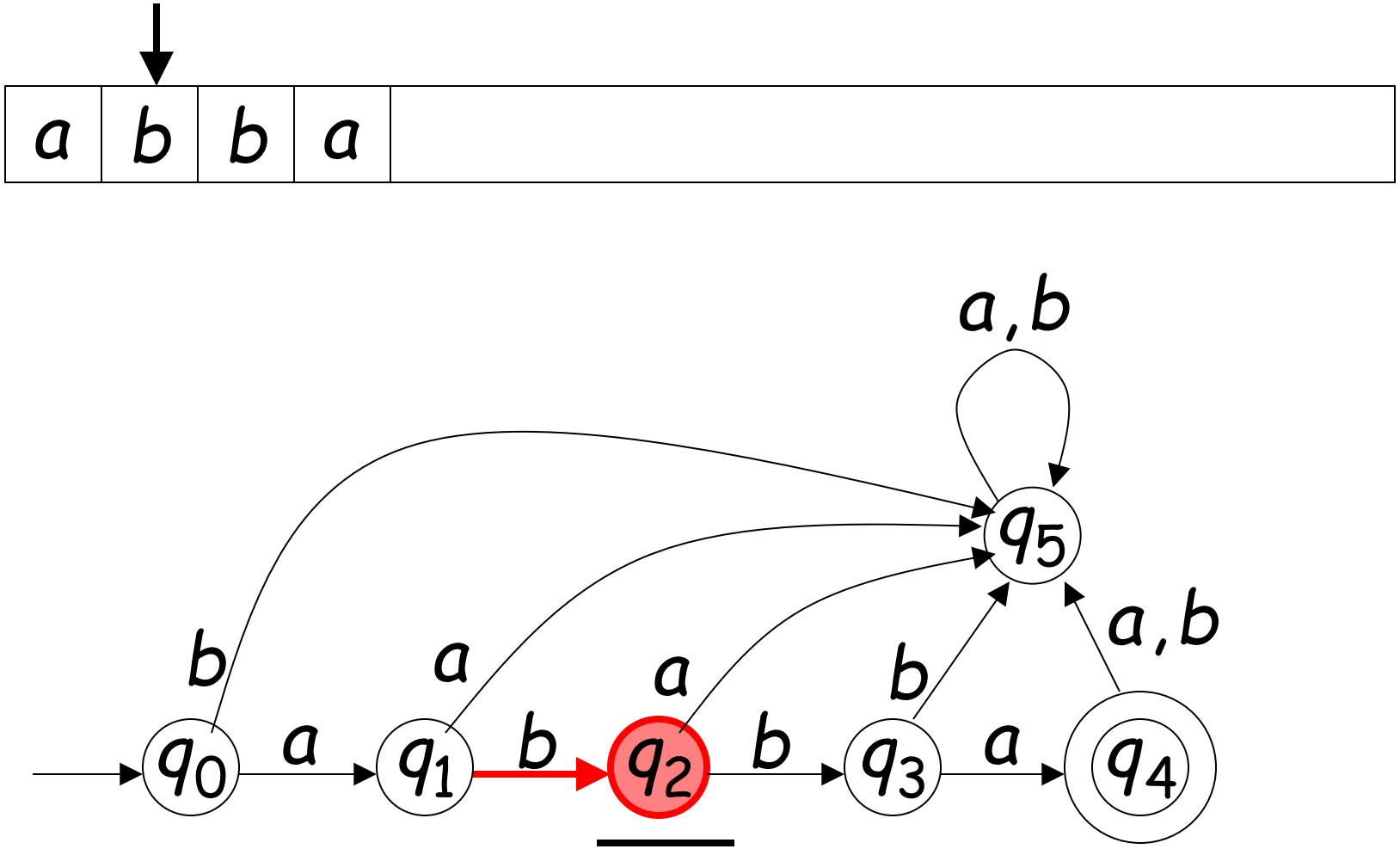
Initial Configuration

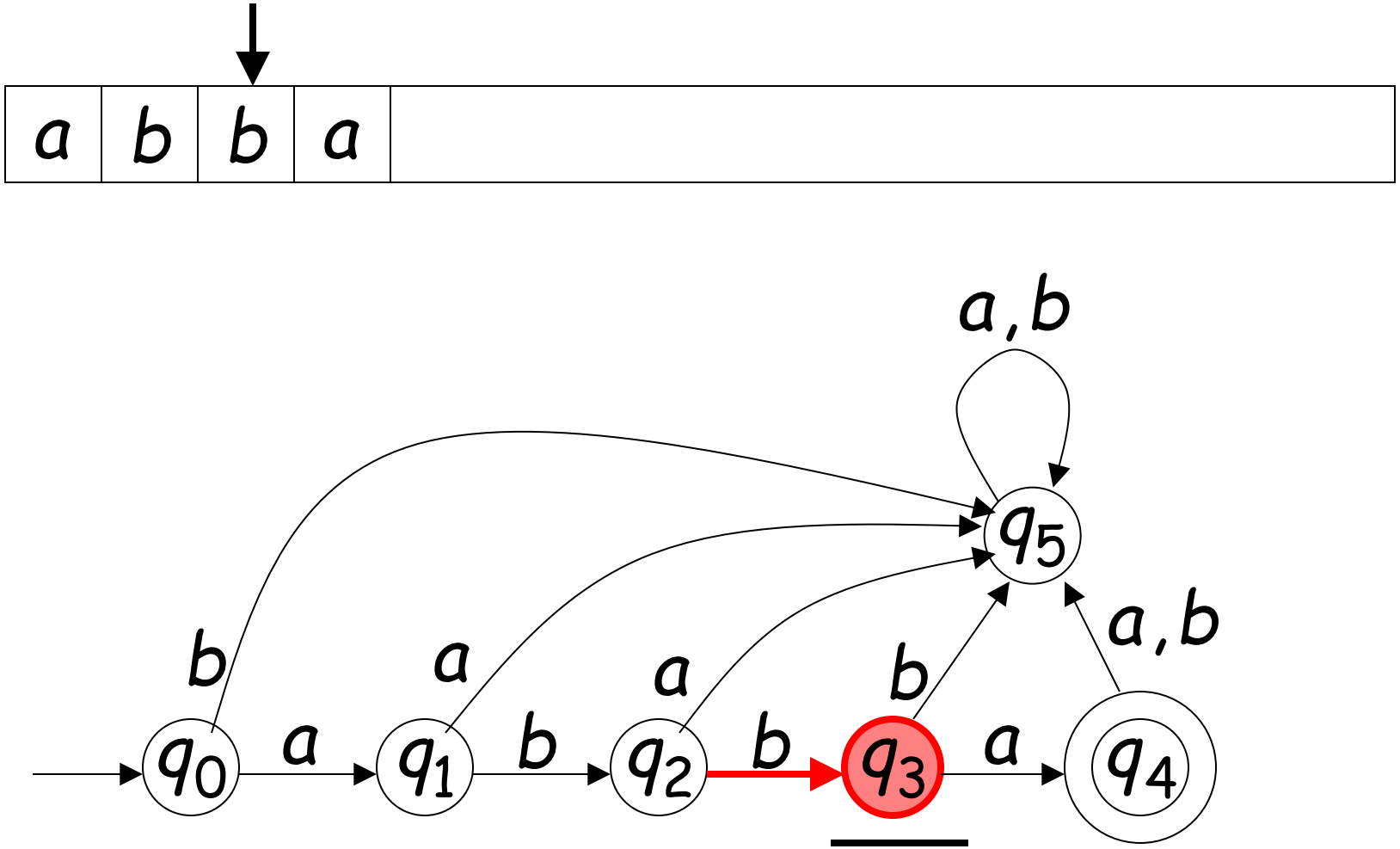


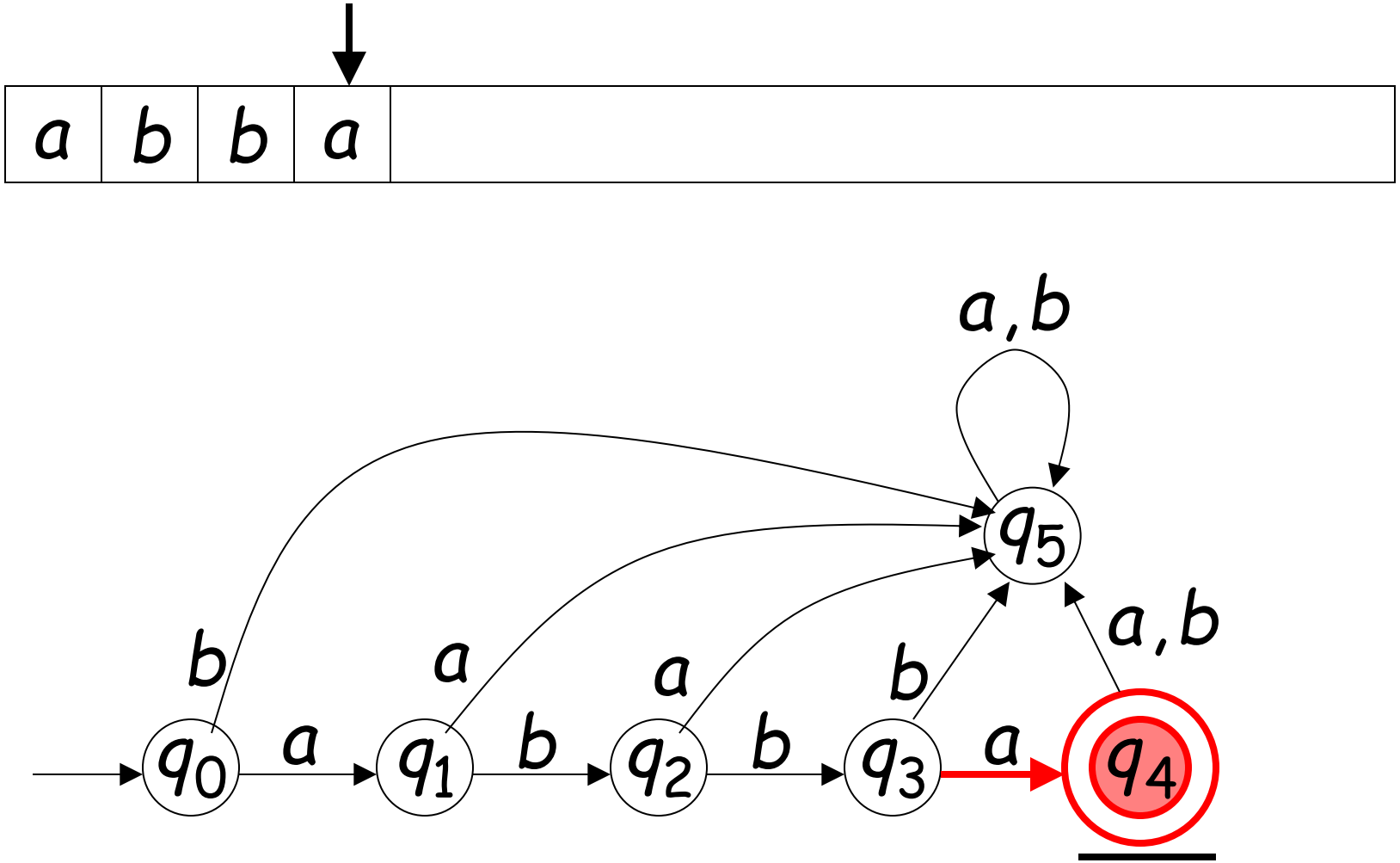
Reading the Input



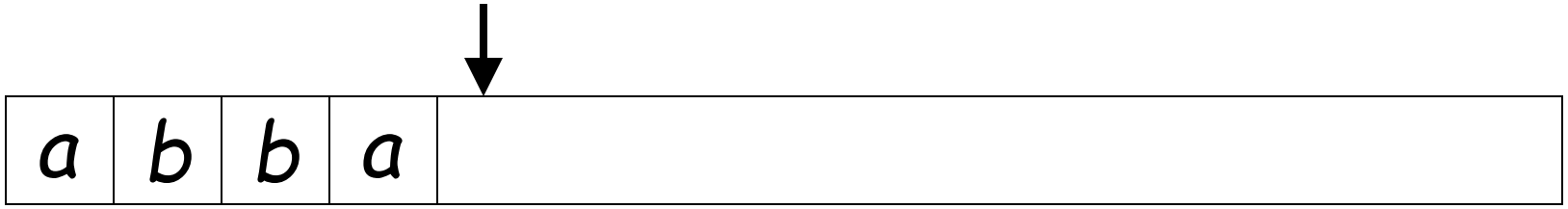


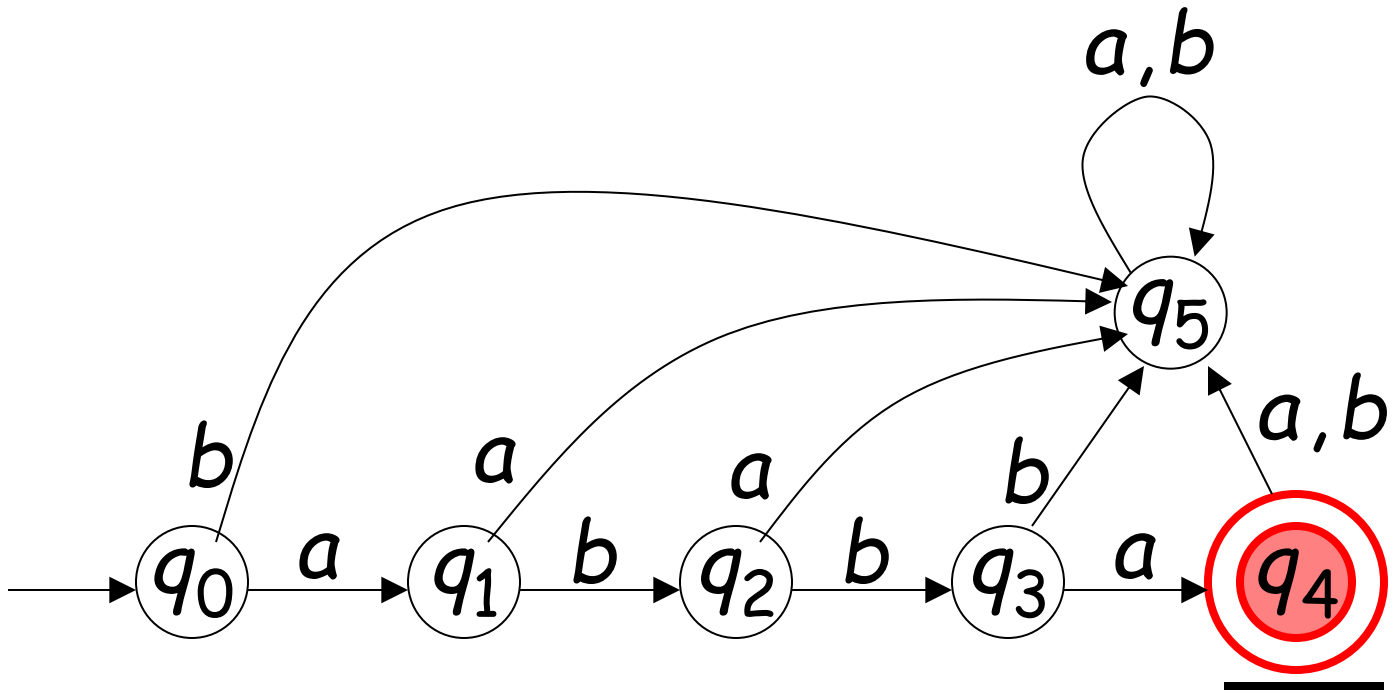

a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


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

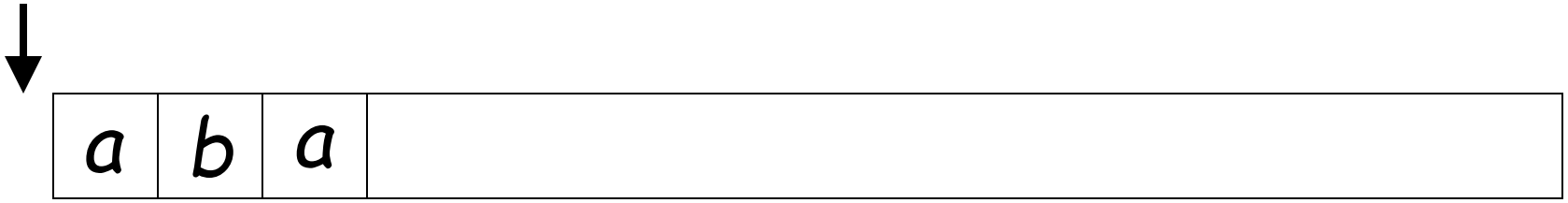
Input finished

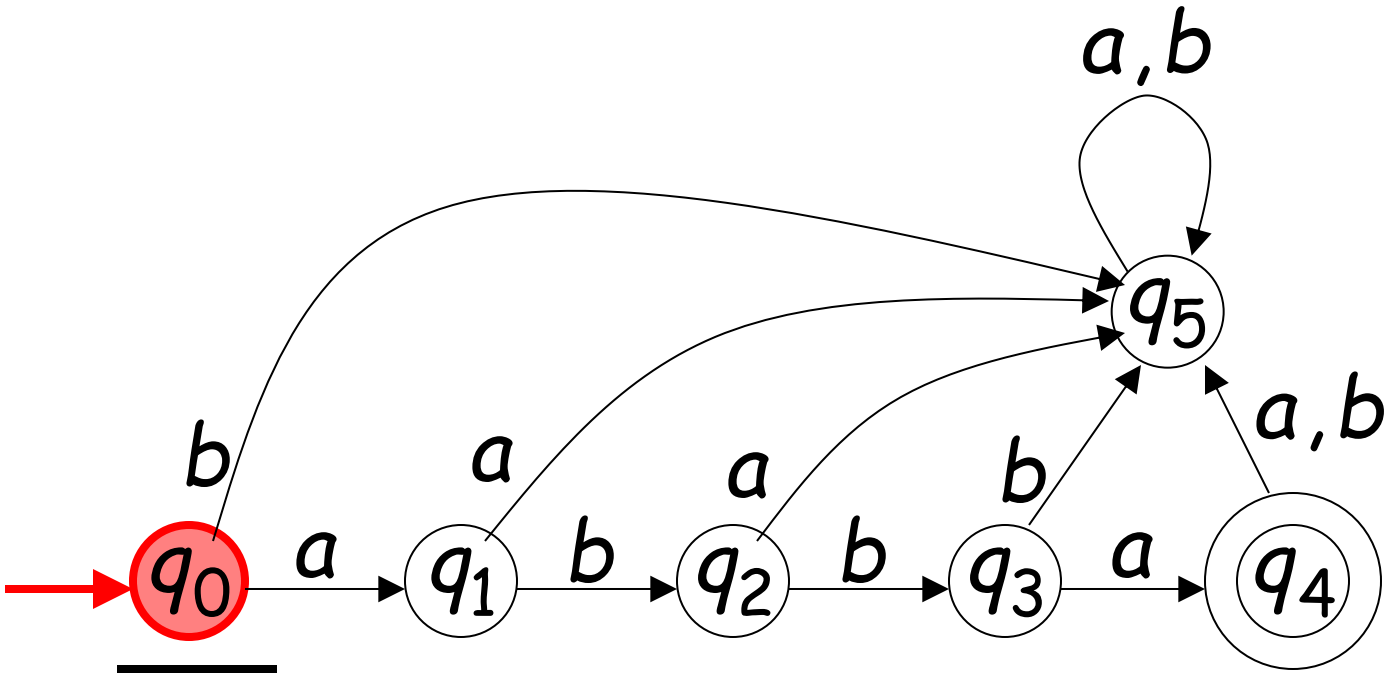


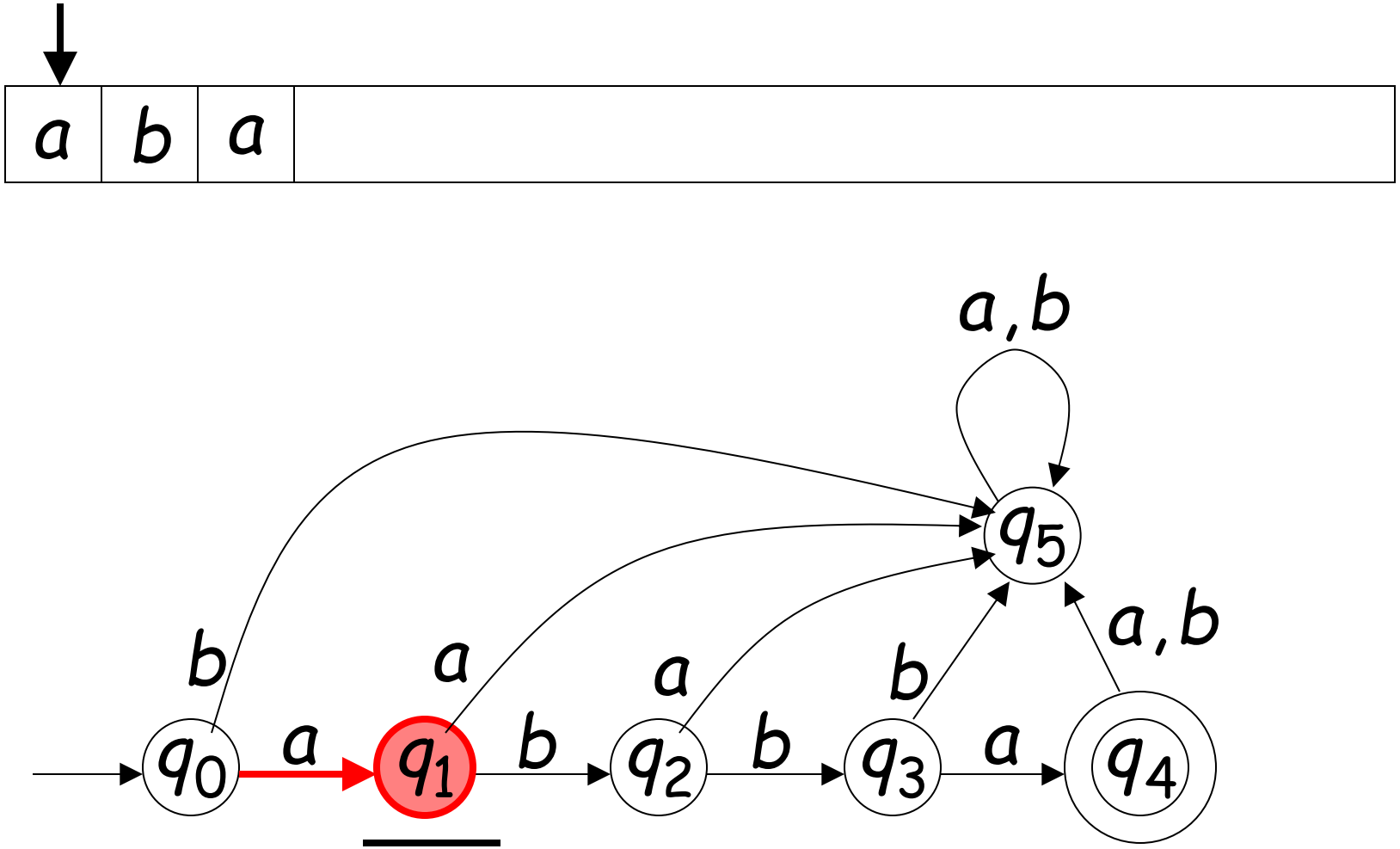


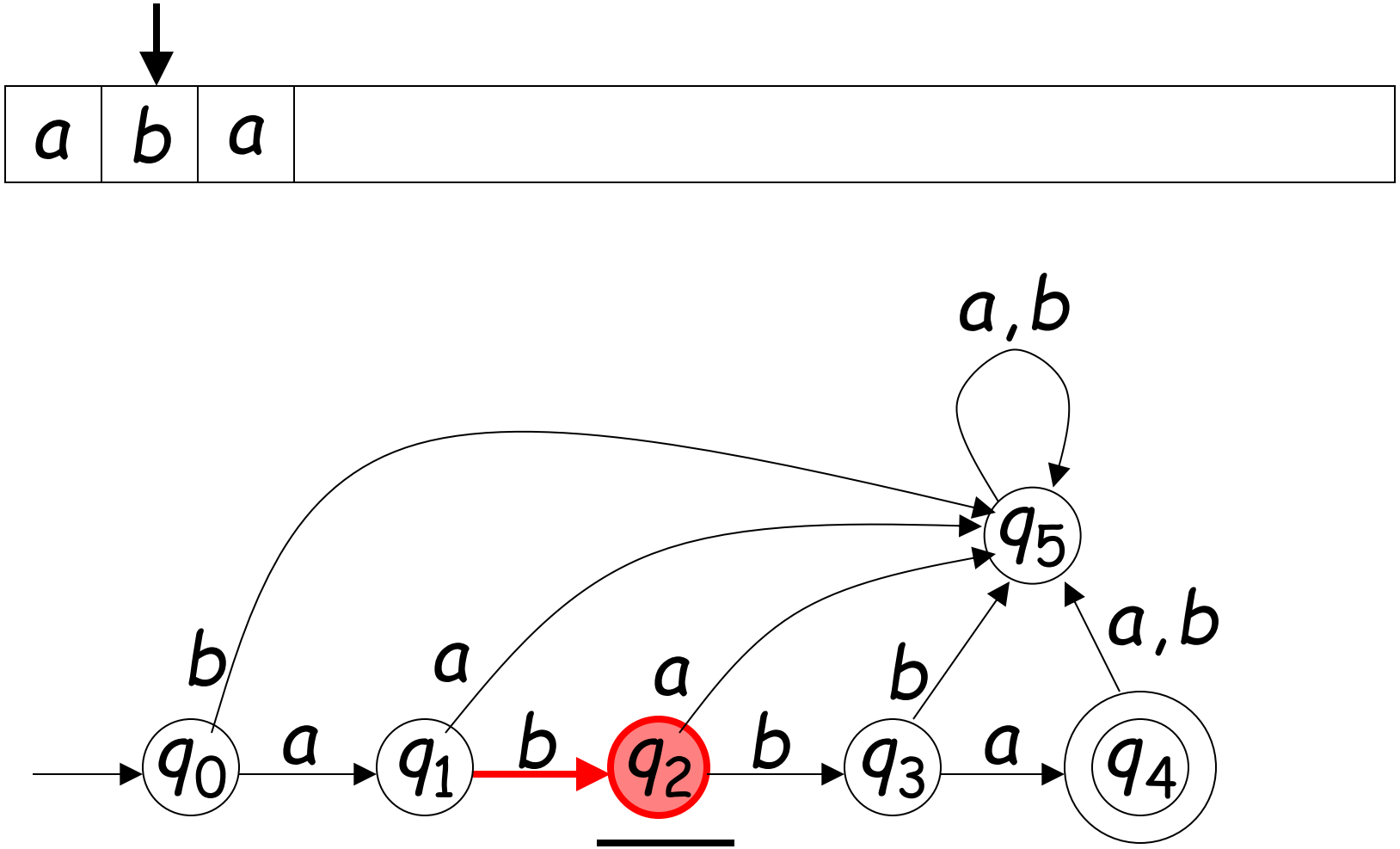
Output: "accept"

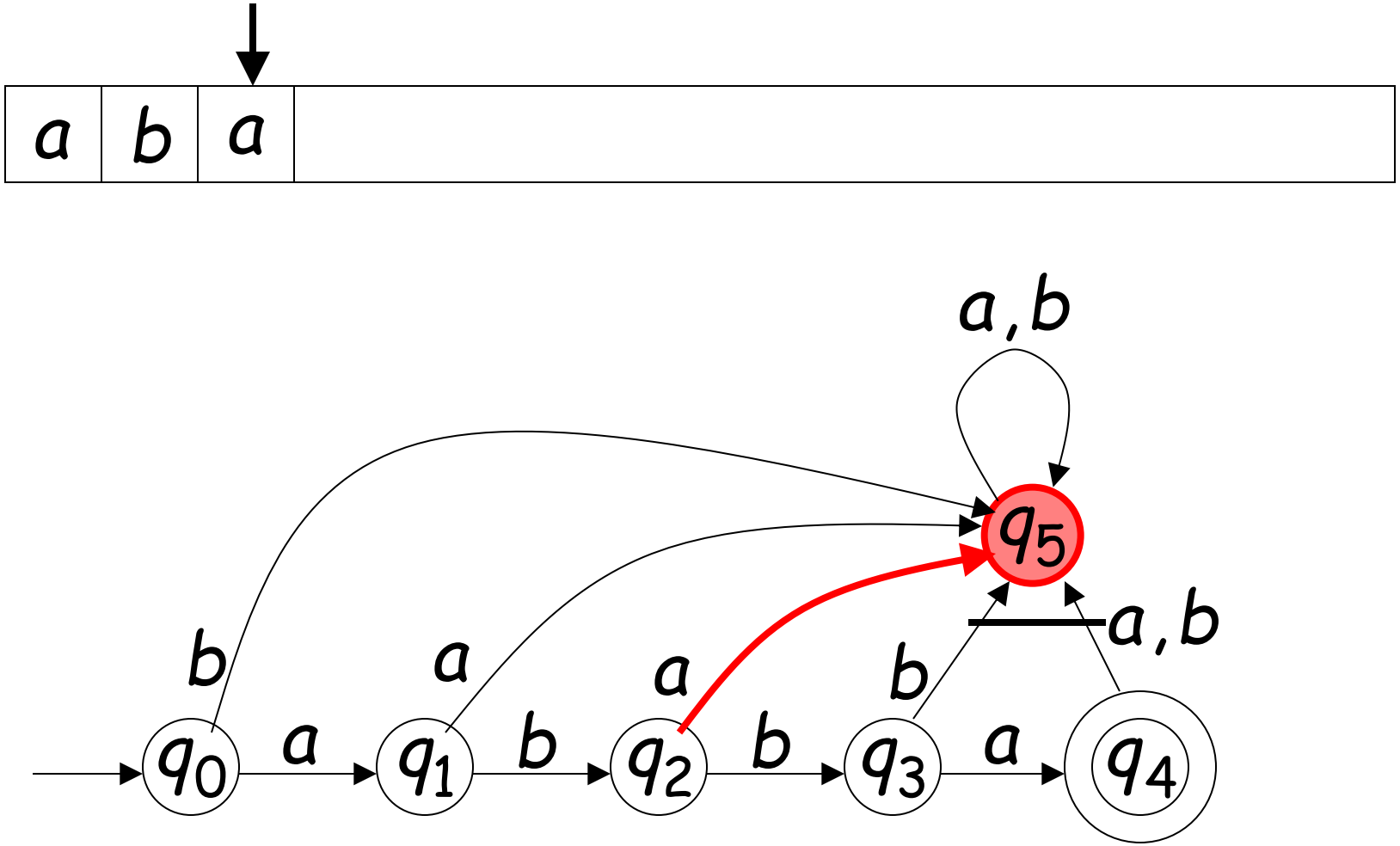
Rejection



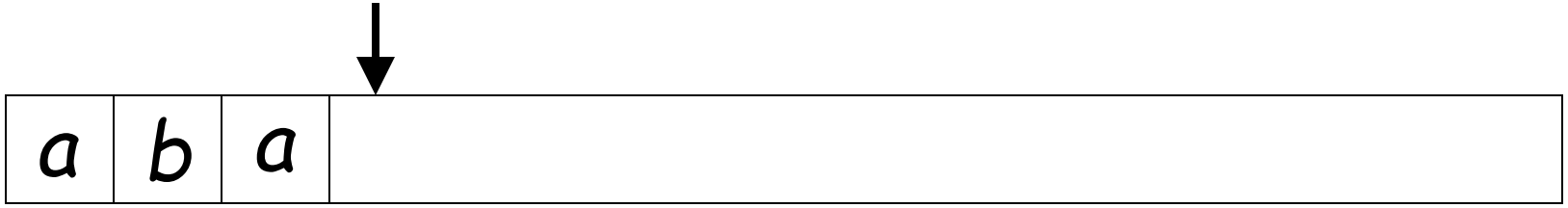


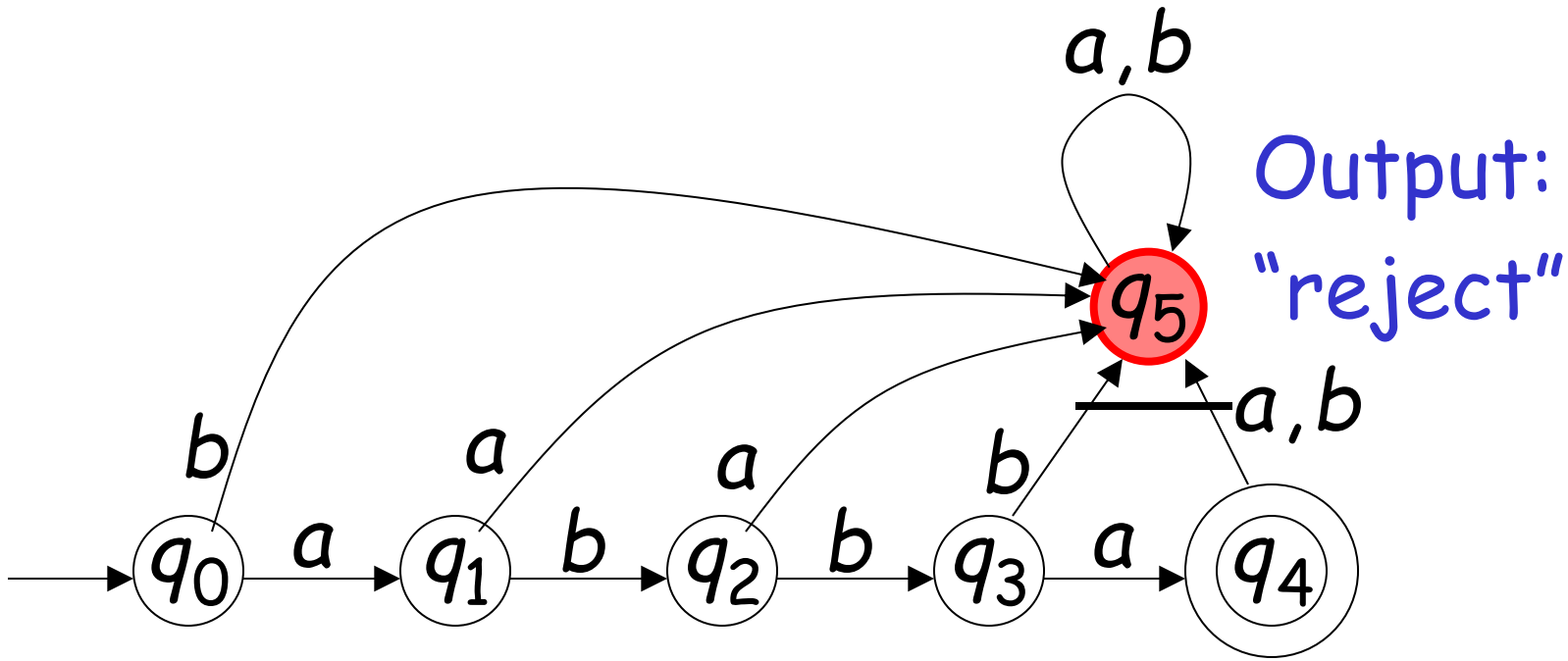

a b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


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

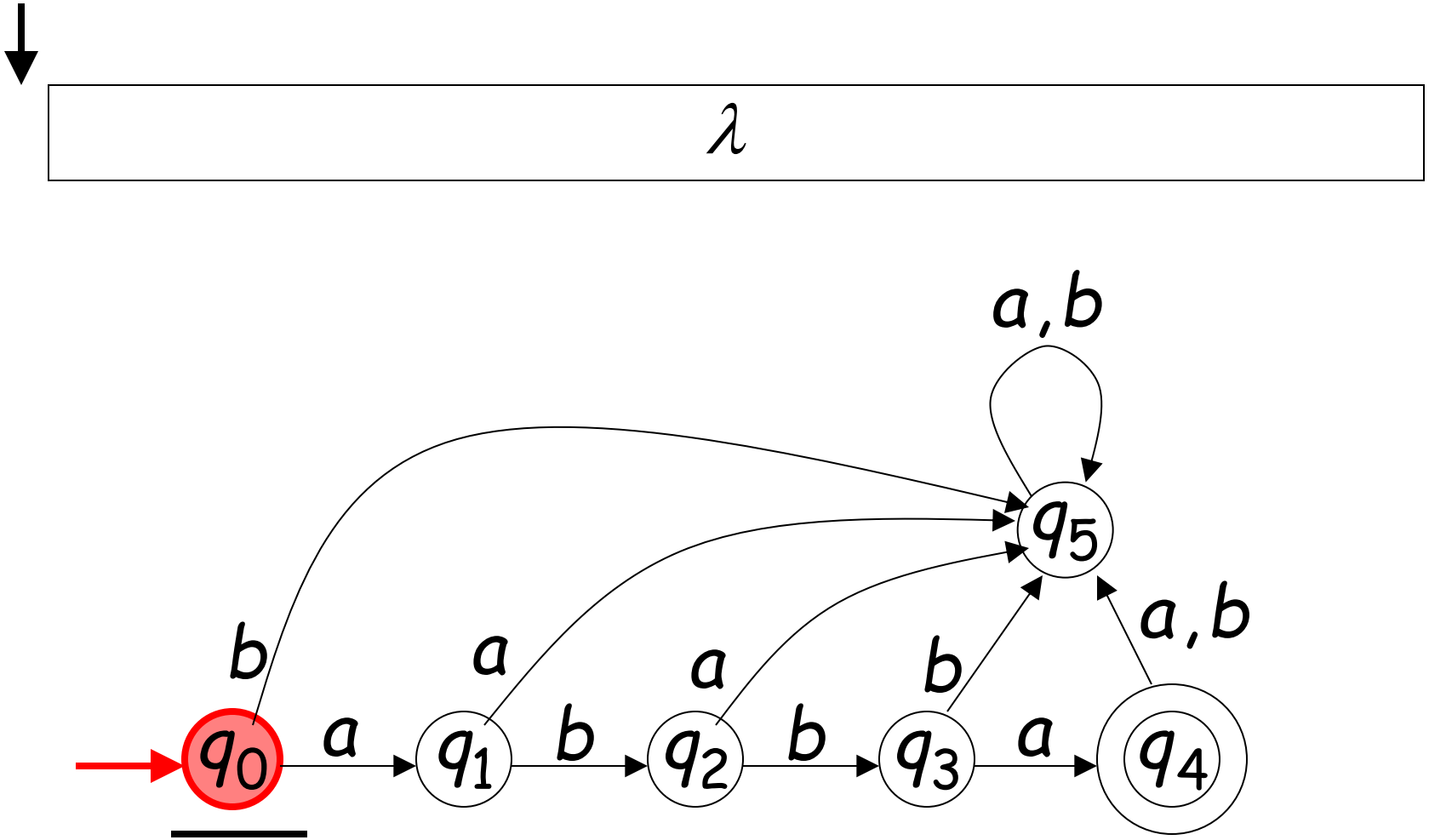

a b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

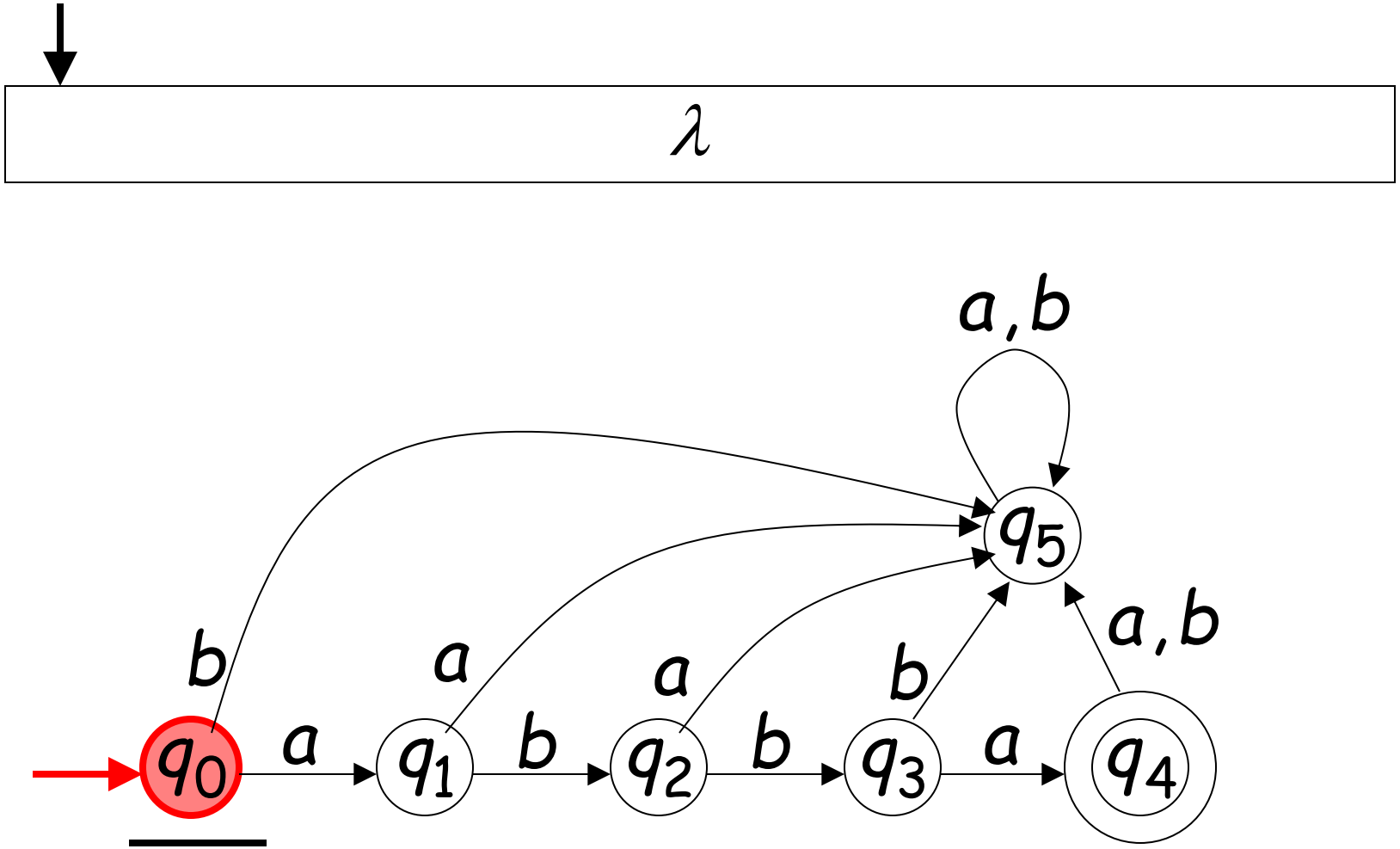
Input finished





Another Rejection

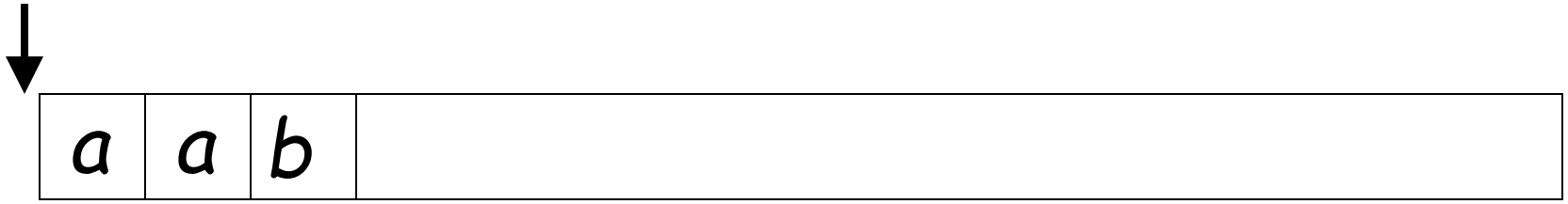


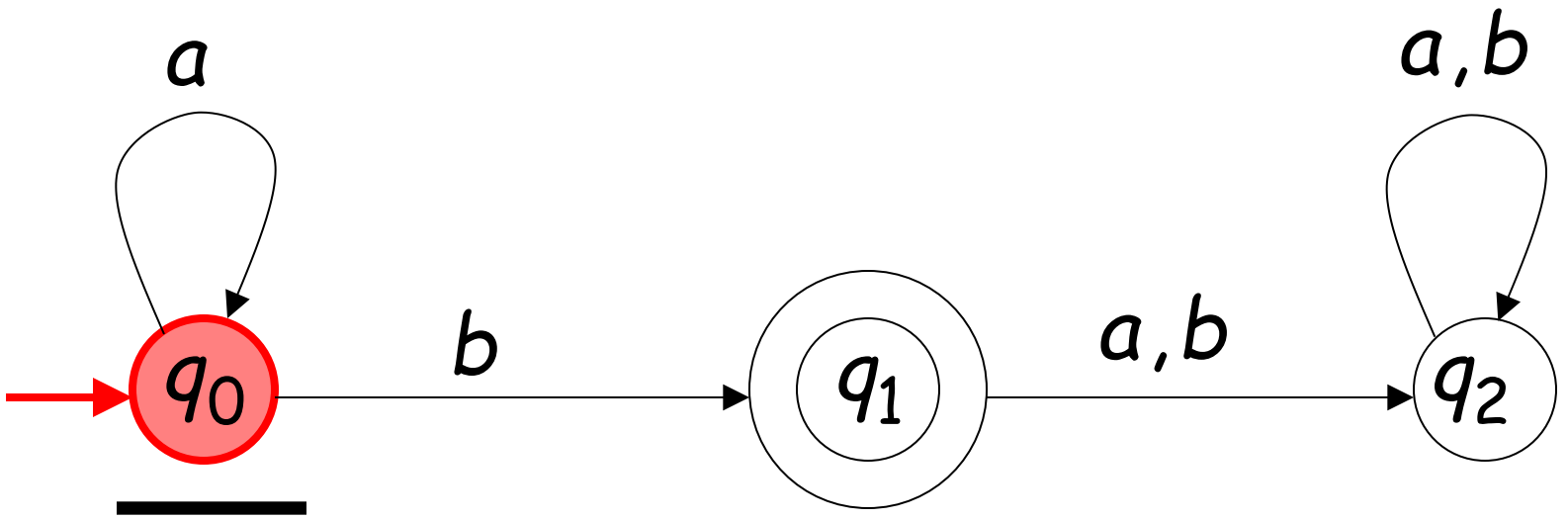


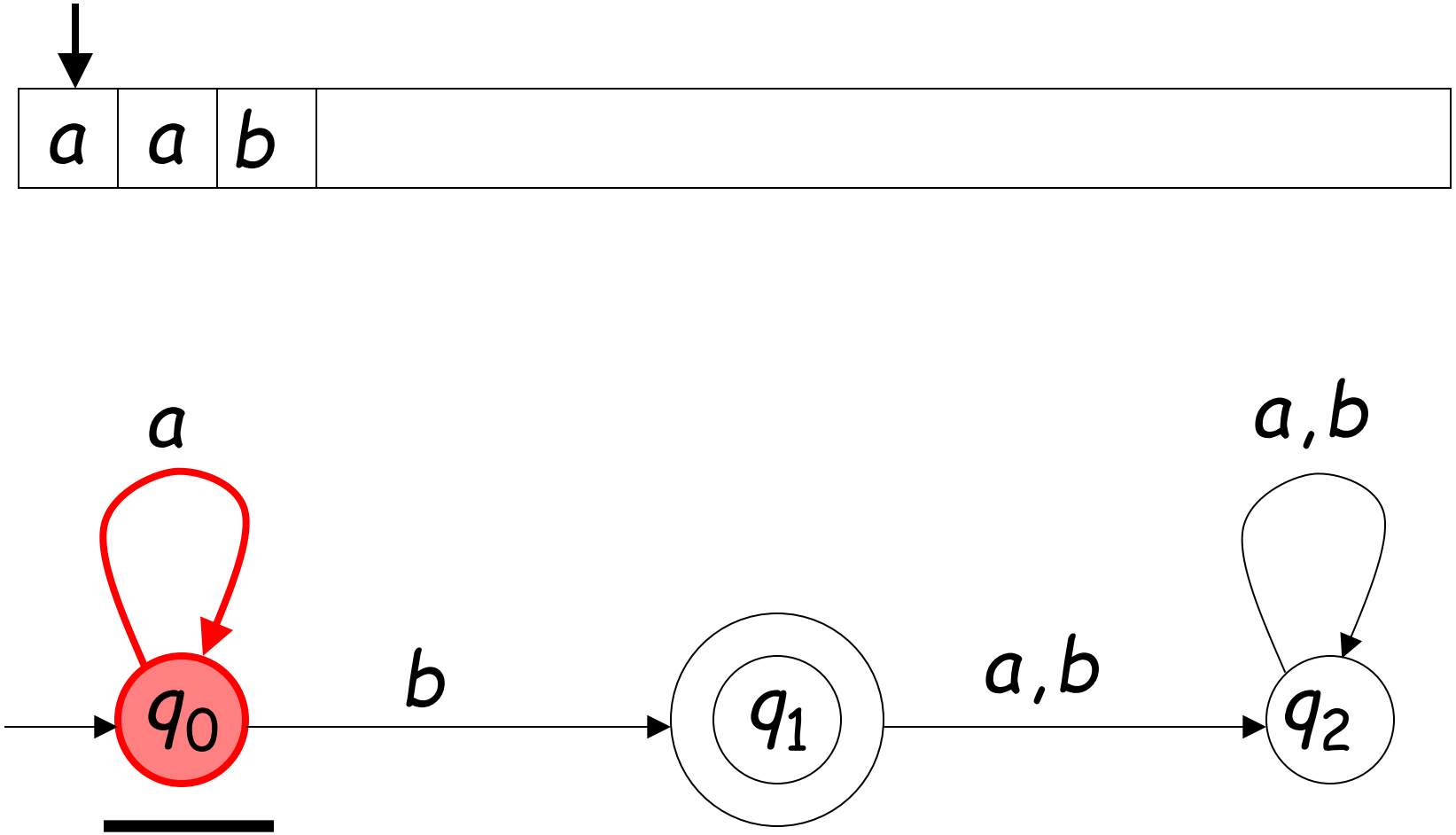
Output:

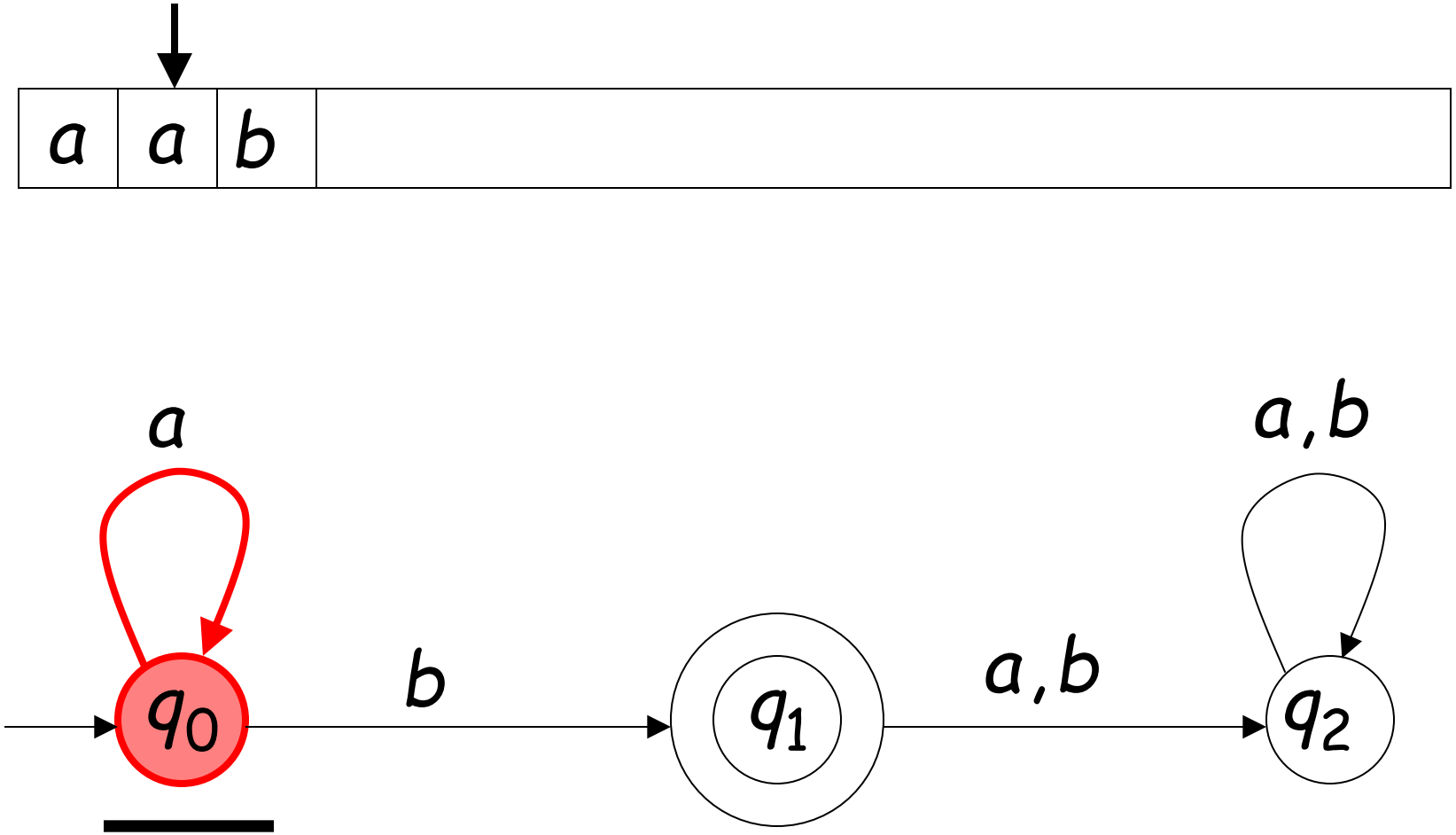
"reject"

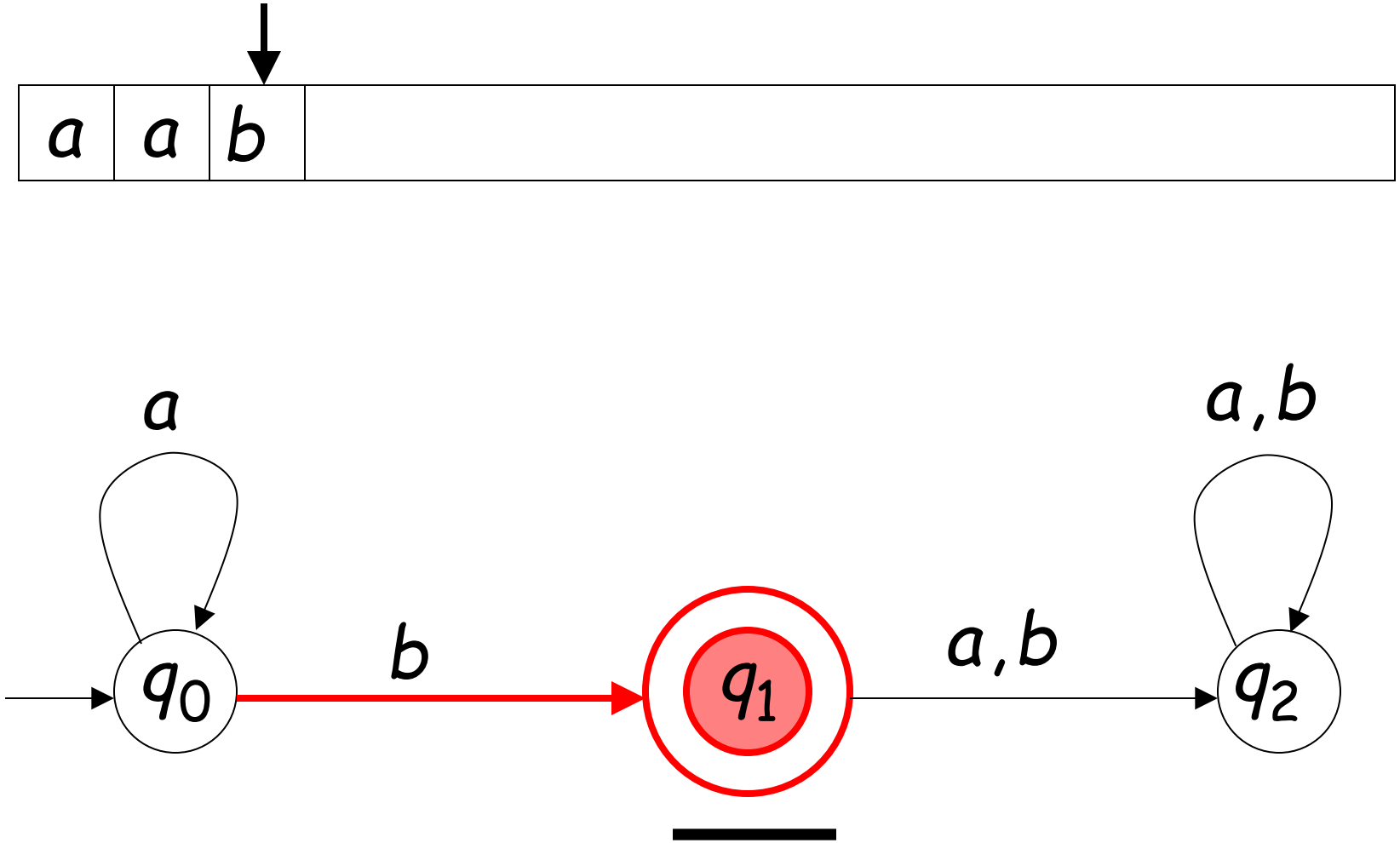
Another Example



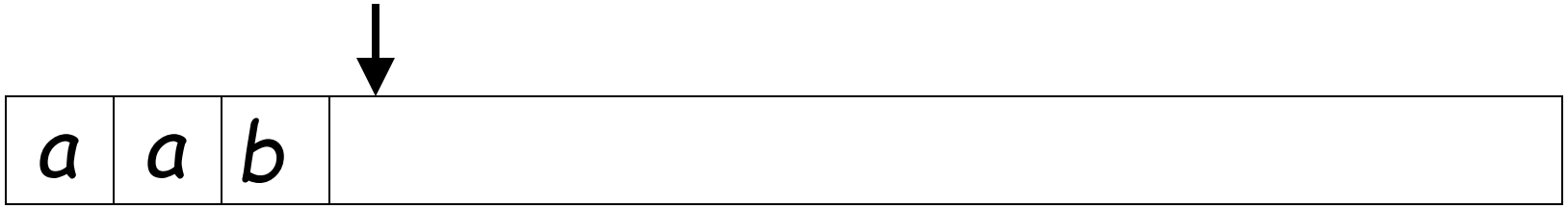


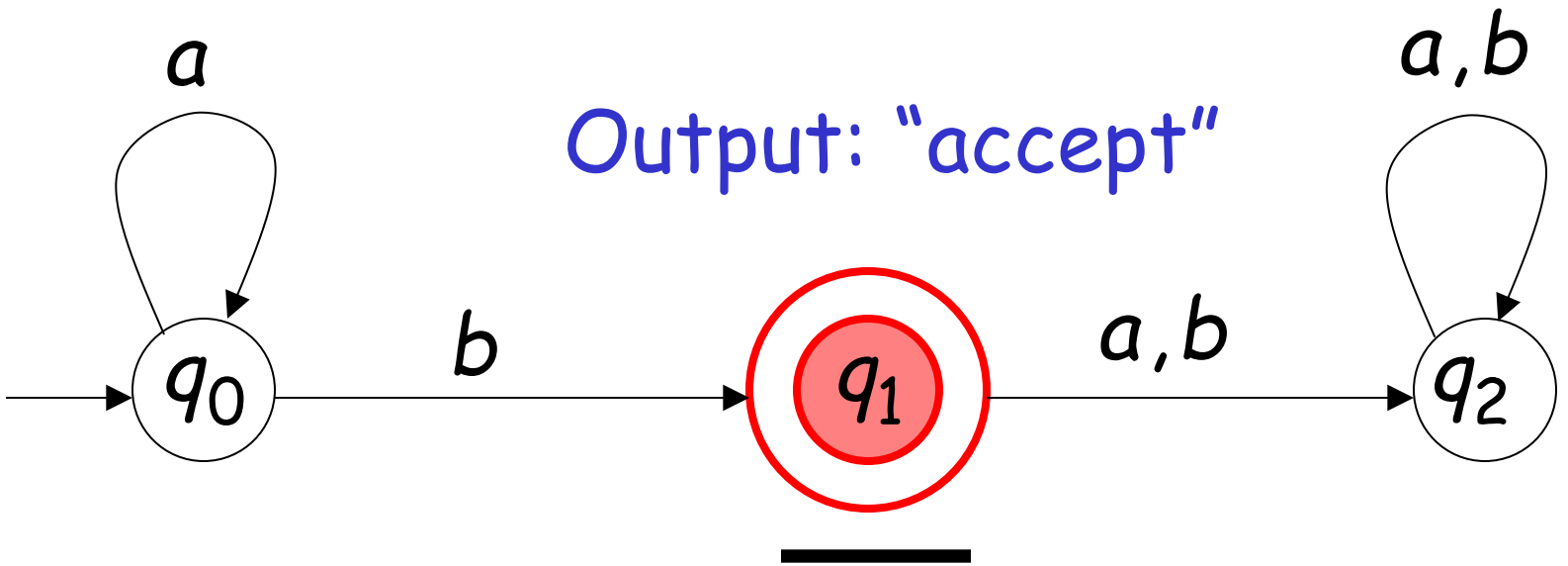

a
a
b
a
a, b
b
a, b
q_0
q_1
q_2


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

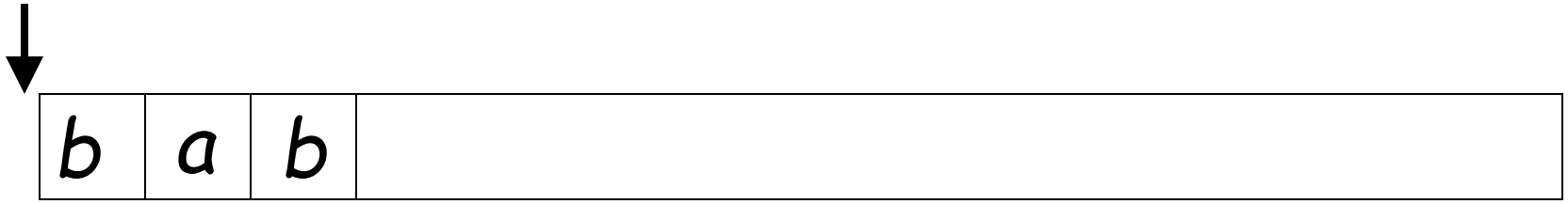

a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

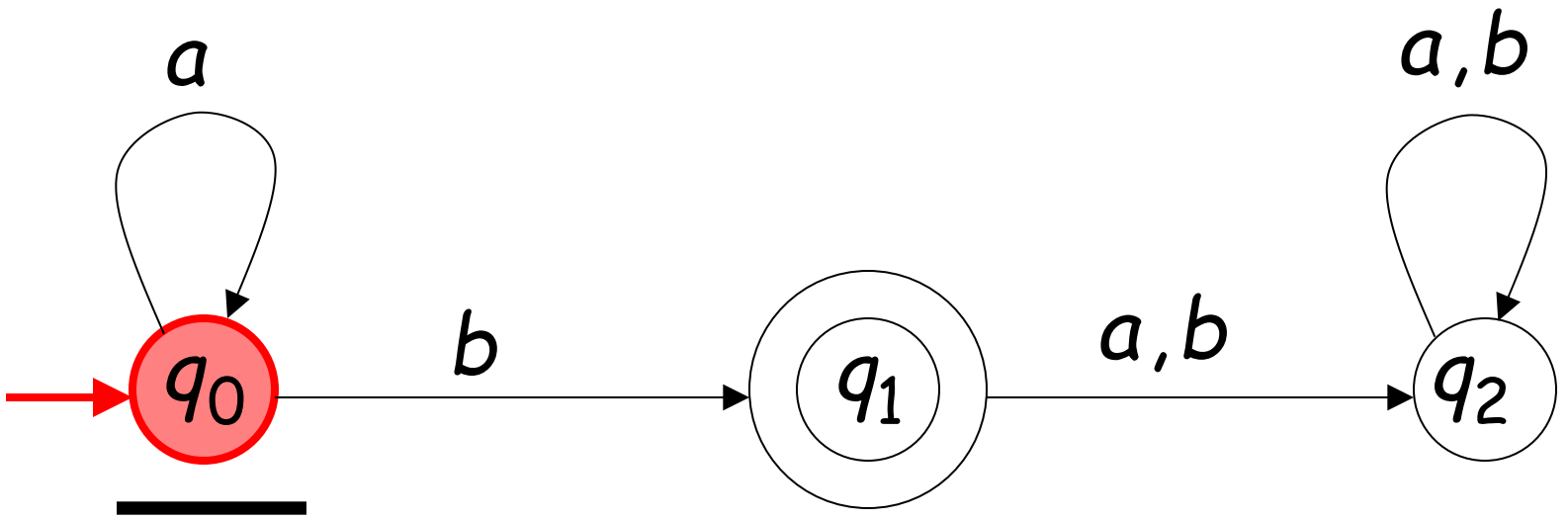
Input finished

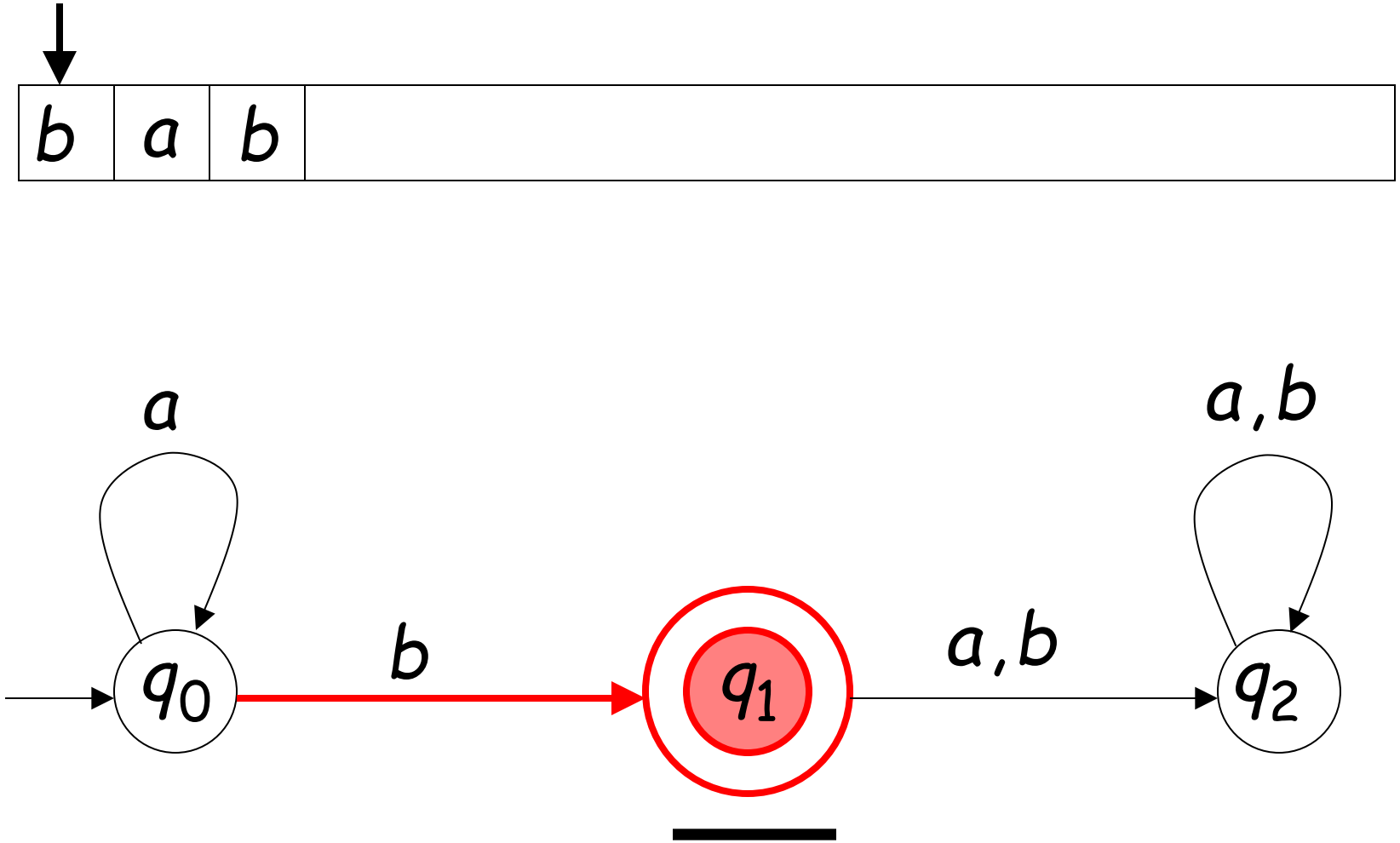


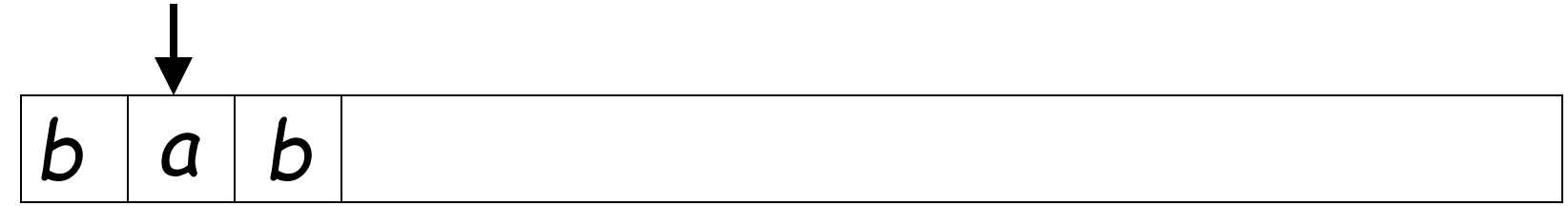


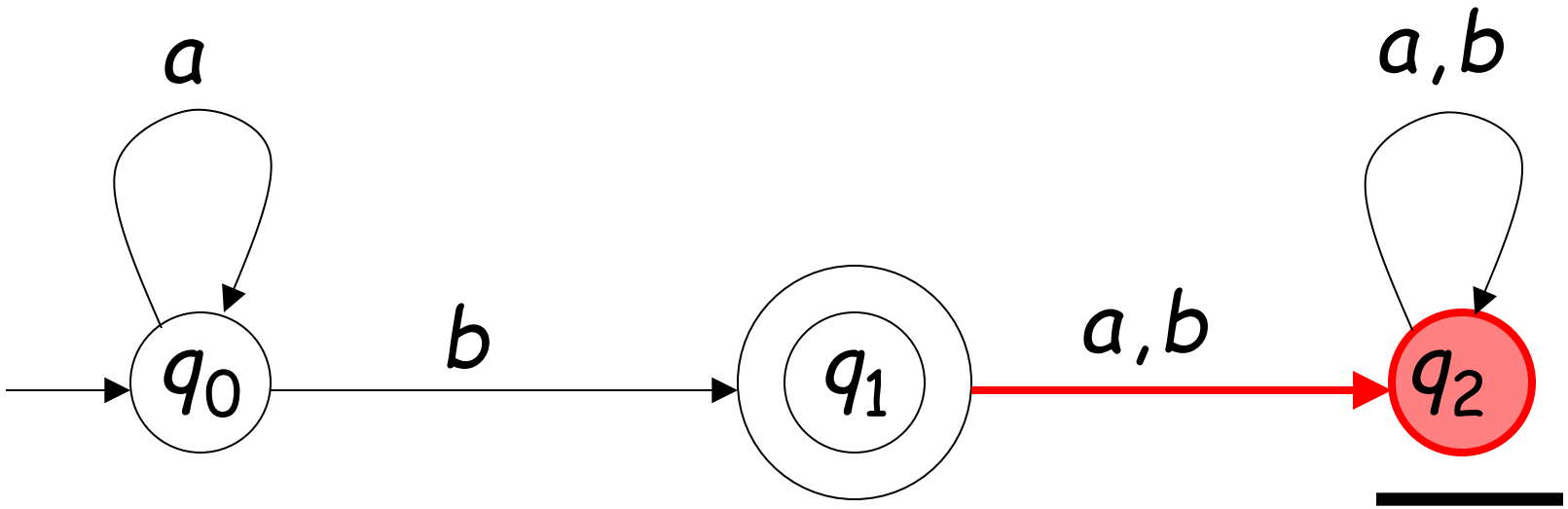
Rejection

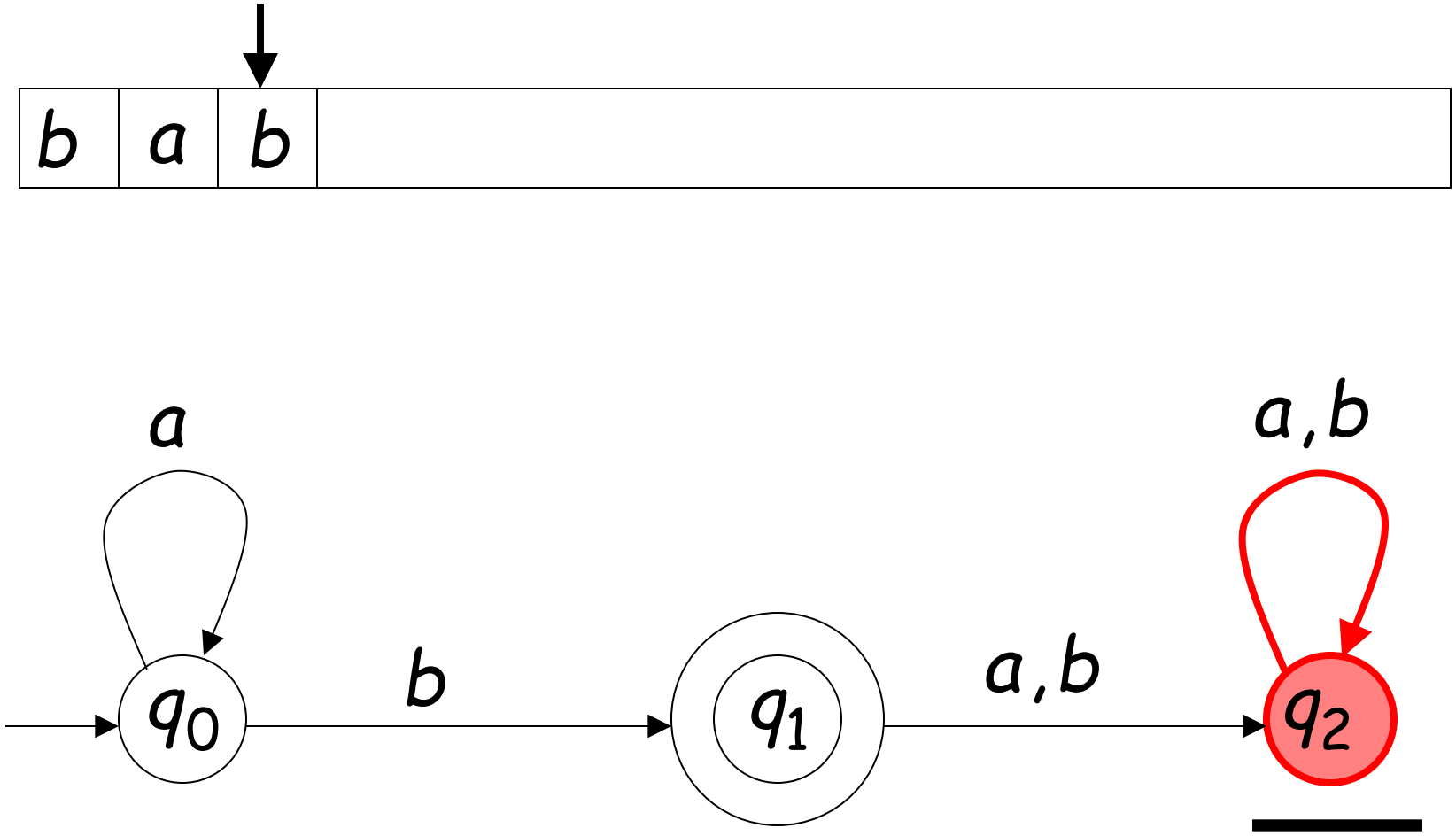




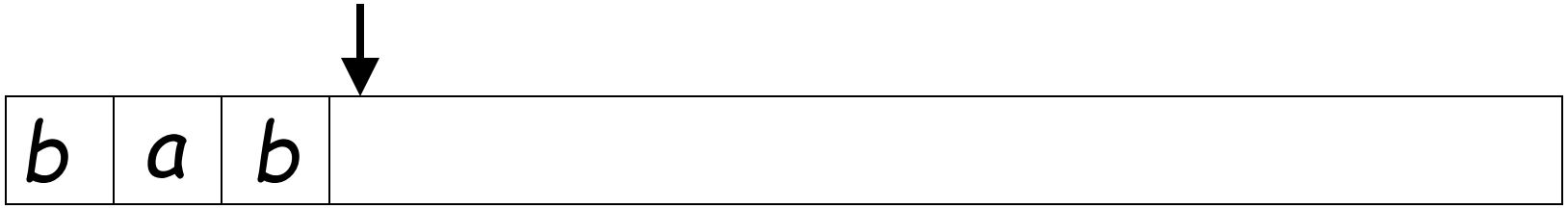

b
a
b
a
a,b
a,b
b
q_0
q_1
q_2

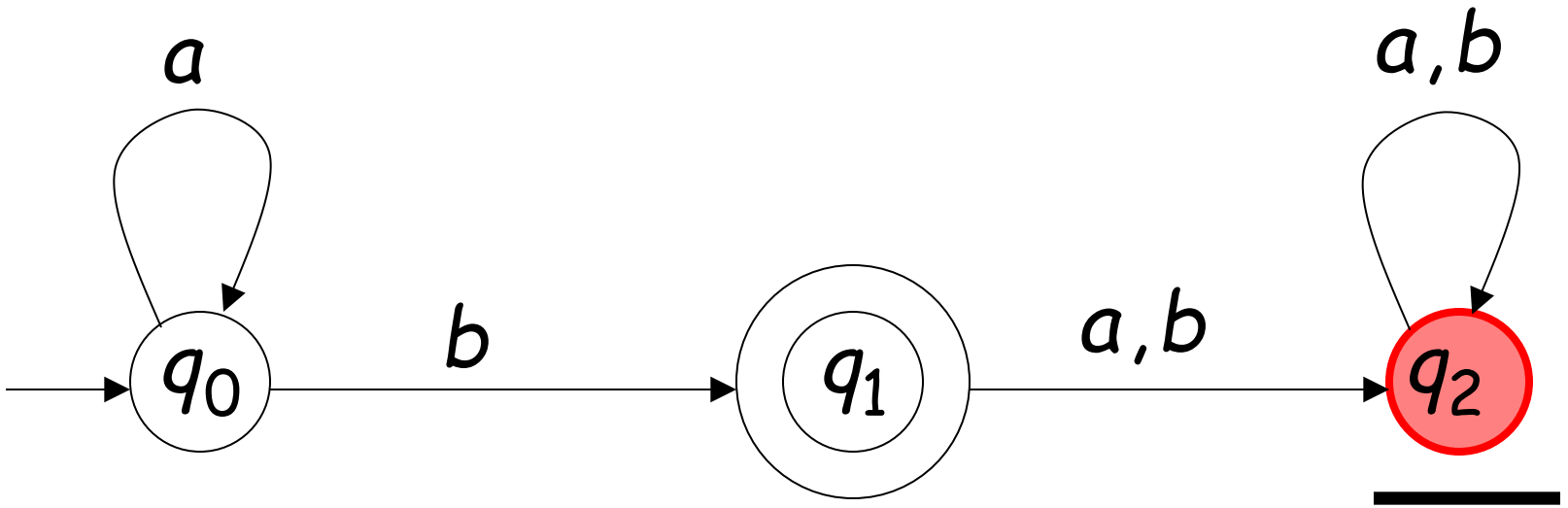

b
a
b


a
a,b
b
a,b
q_0
q_1
q_2


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

Input finished

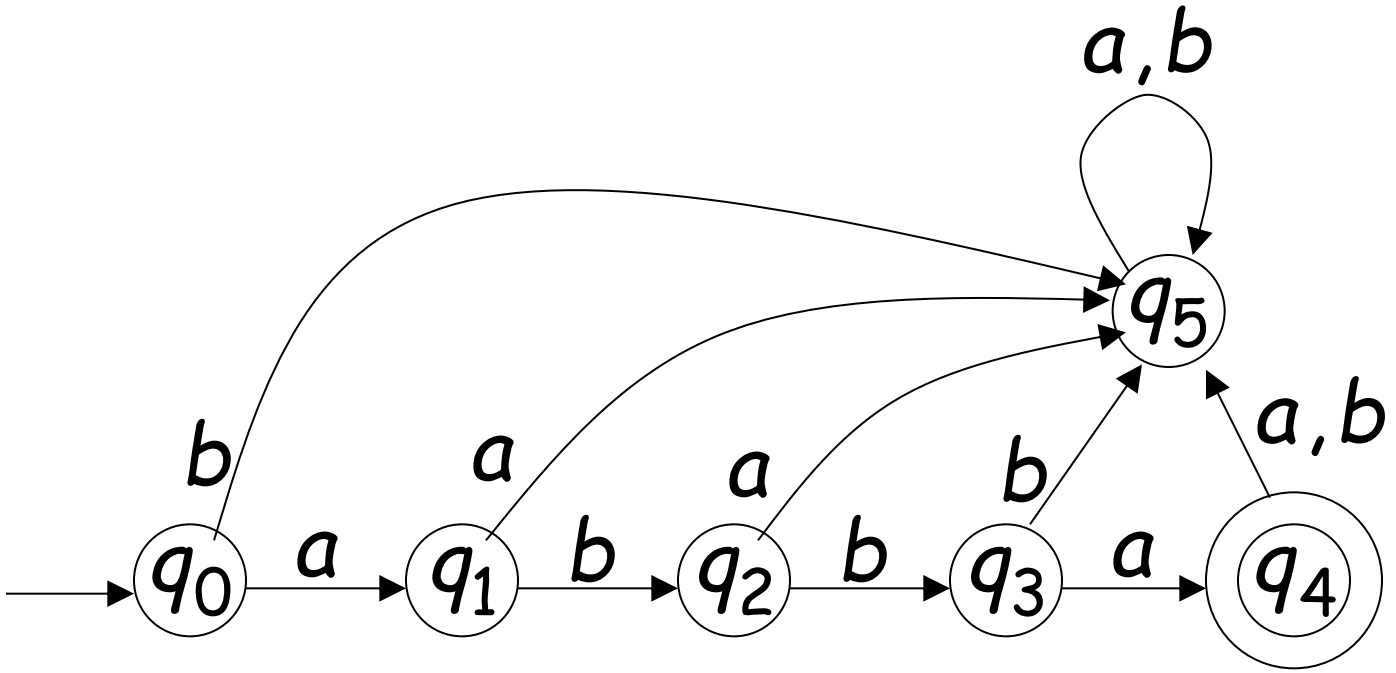




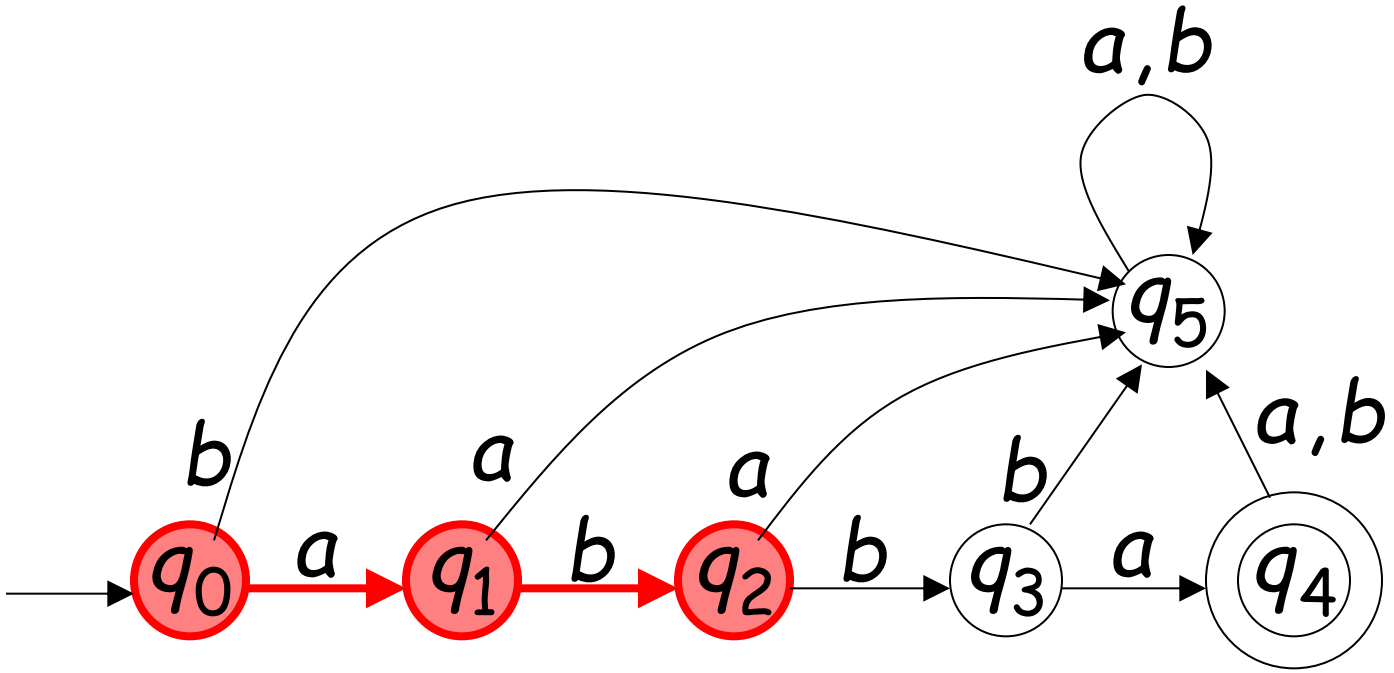
Output: "reject"

Extended Transition Function δ^*

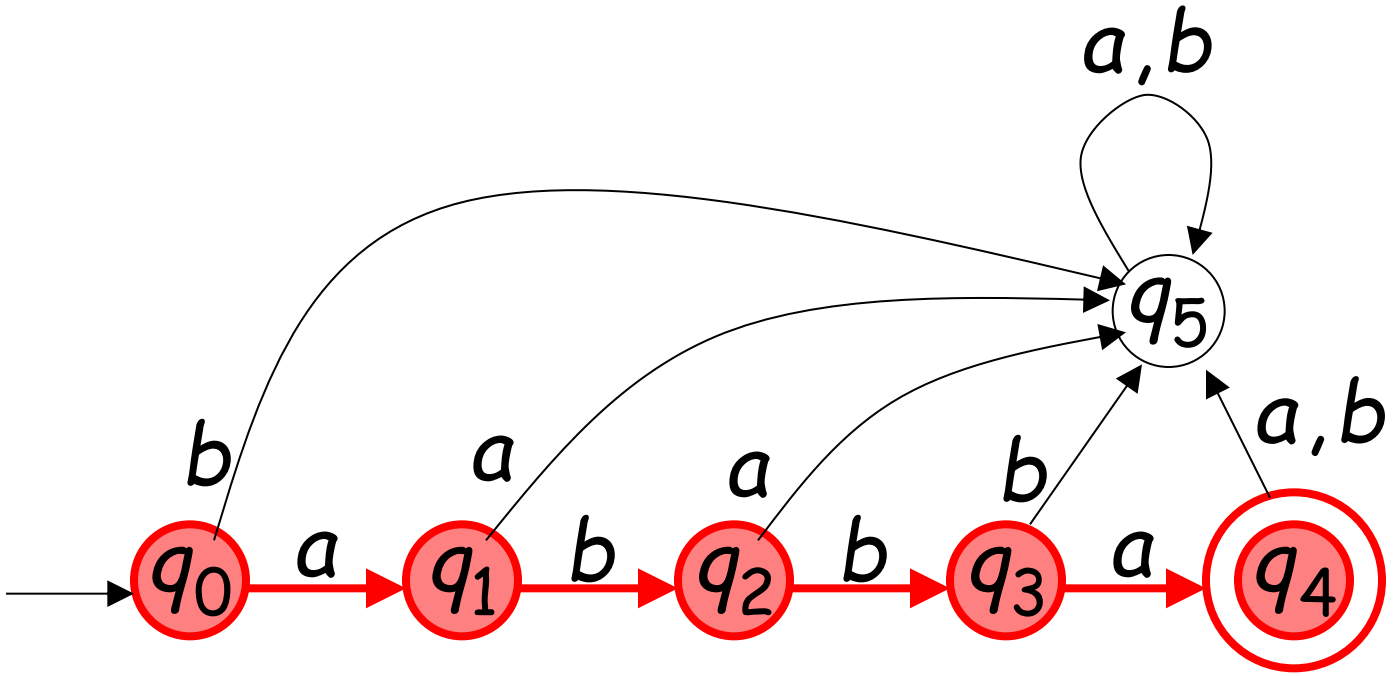
$$\delta^* : Q \times \Sigma^* \to Q$$



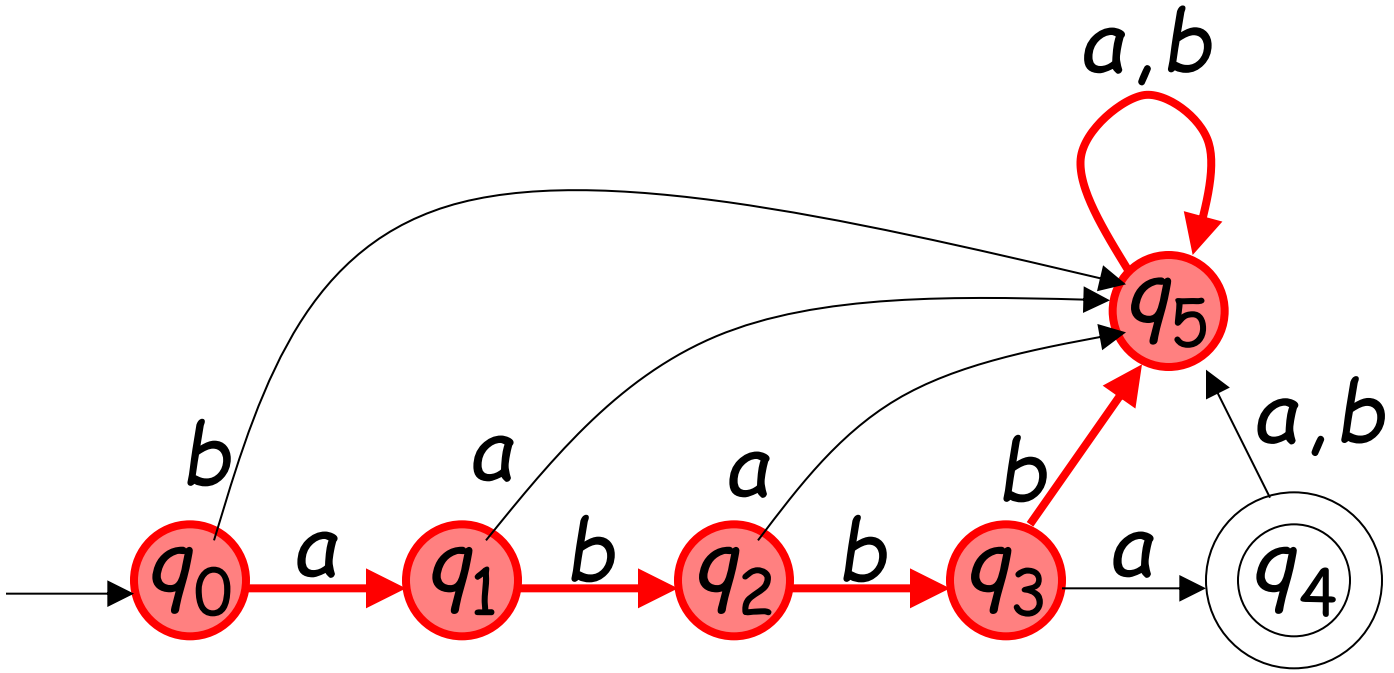
$$\delta^*(q_0, ab) = q_2$$



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

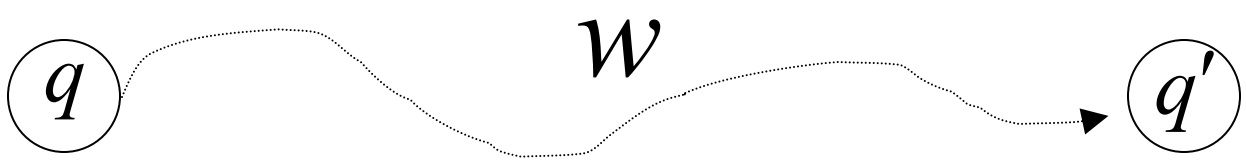


$$\delta^*(q_0, abbbaa) = q_5$$

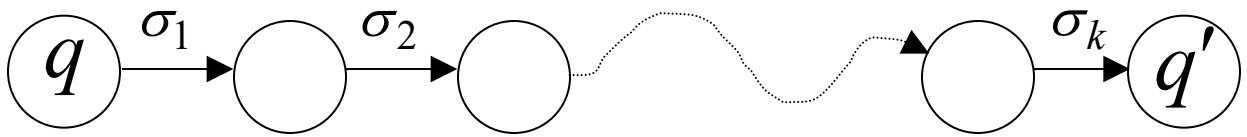


Observation: There is a walk from q to q' with label w

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

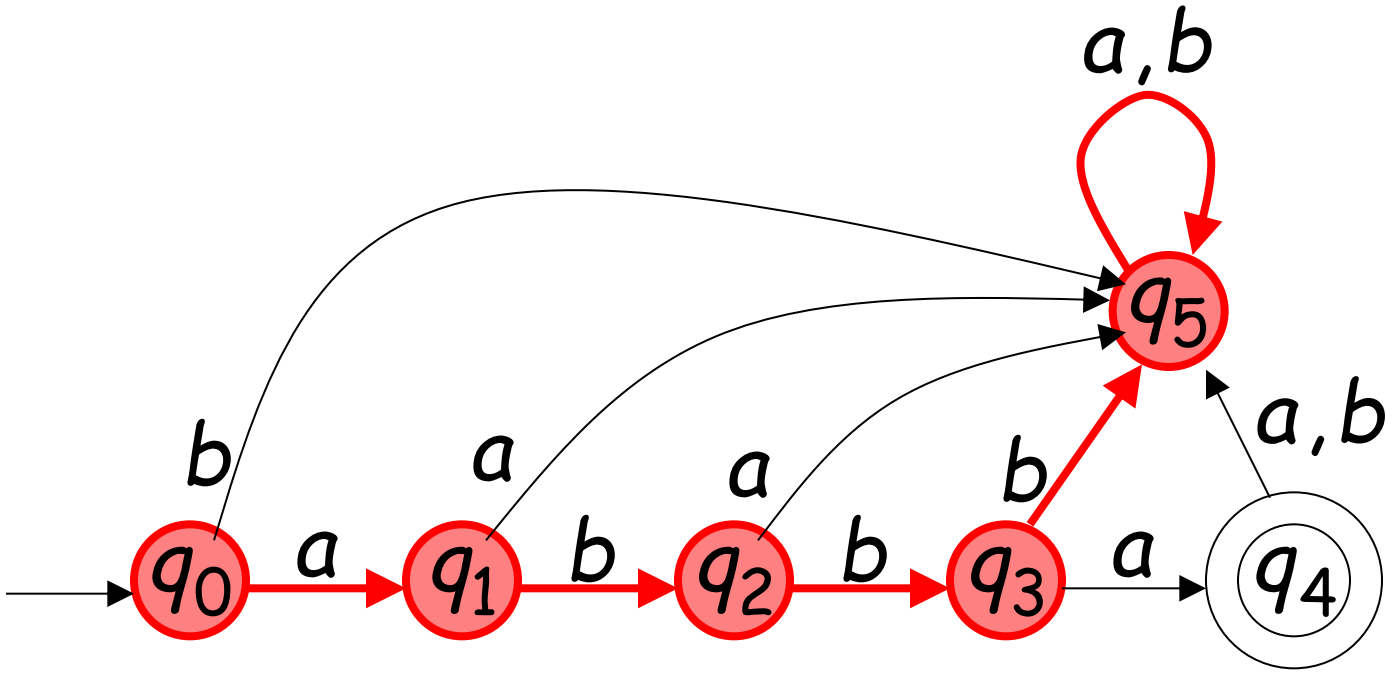


$$w=\sigma_1\sigma_2\cdots\sigma_k$$



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

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



Recursive Definition

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

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

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

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

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

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

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

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

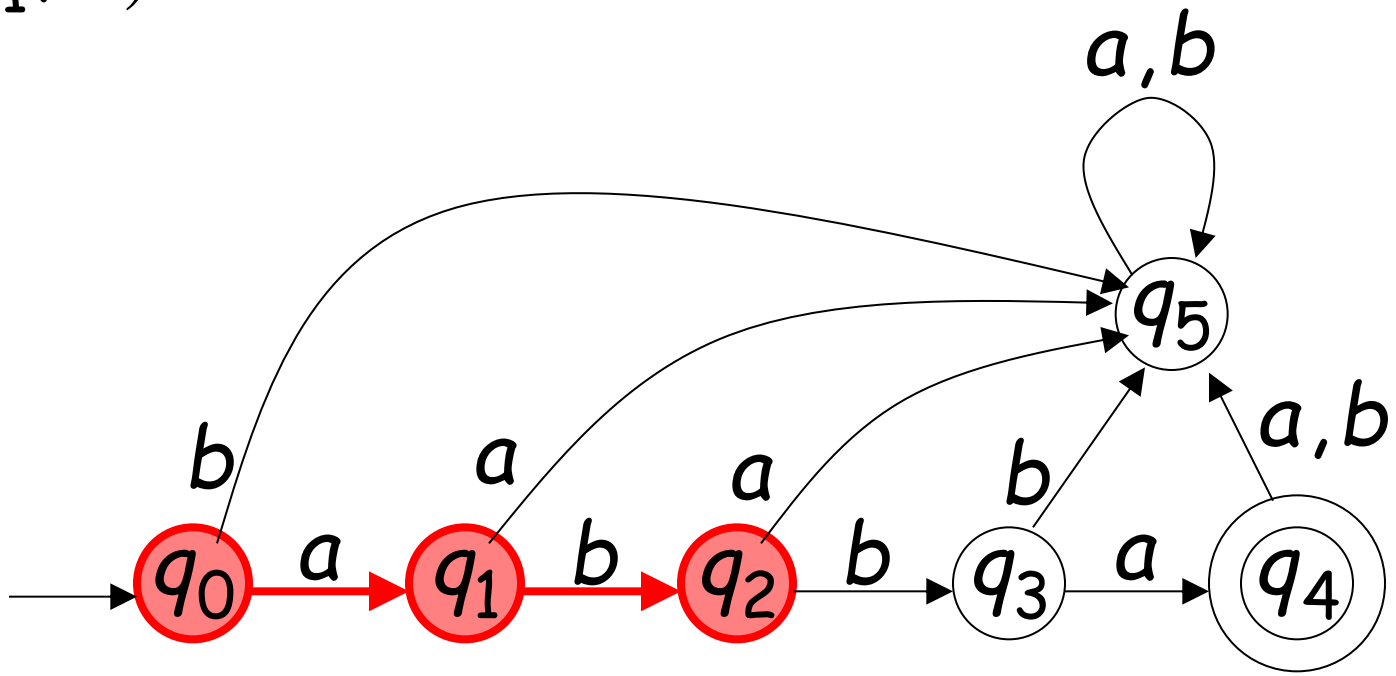
$$\delta(\delta^*(q_0, a), b) =$$

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

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

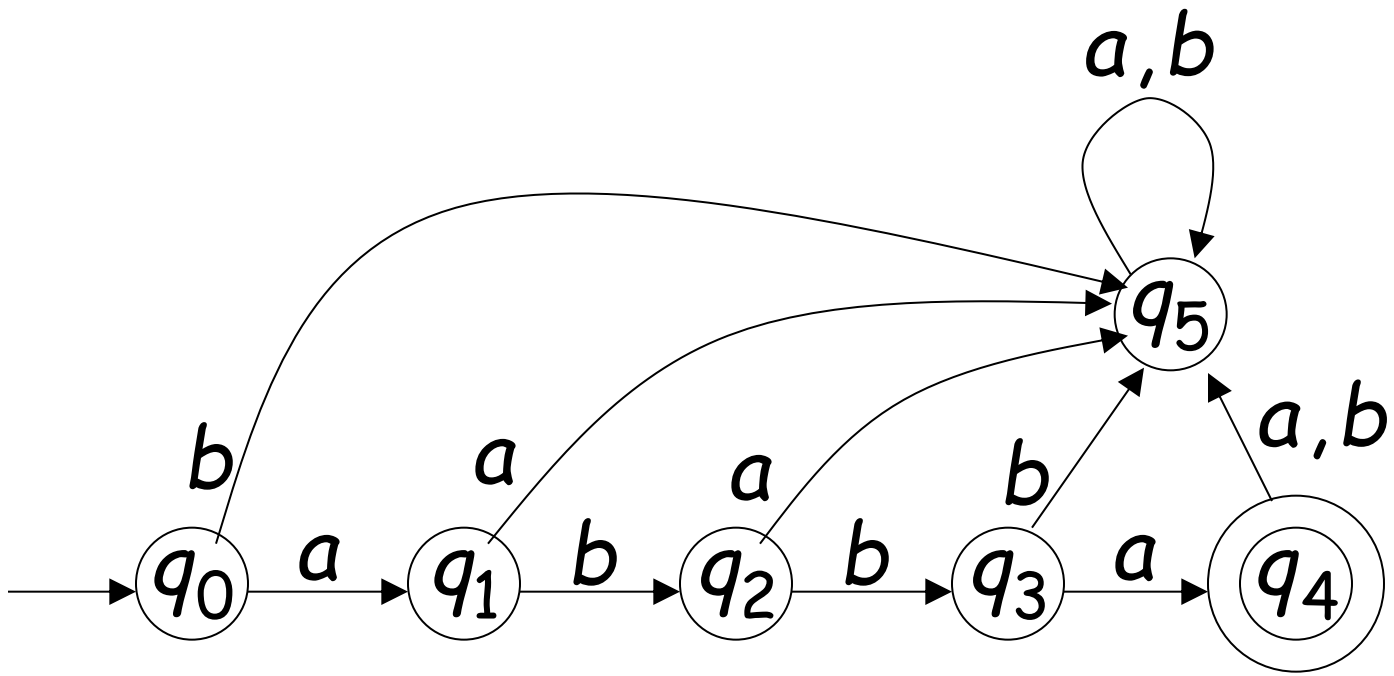
$$\delta(q_1, b) =$$

$$q_2$$



Input Alphabet Σ

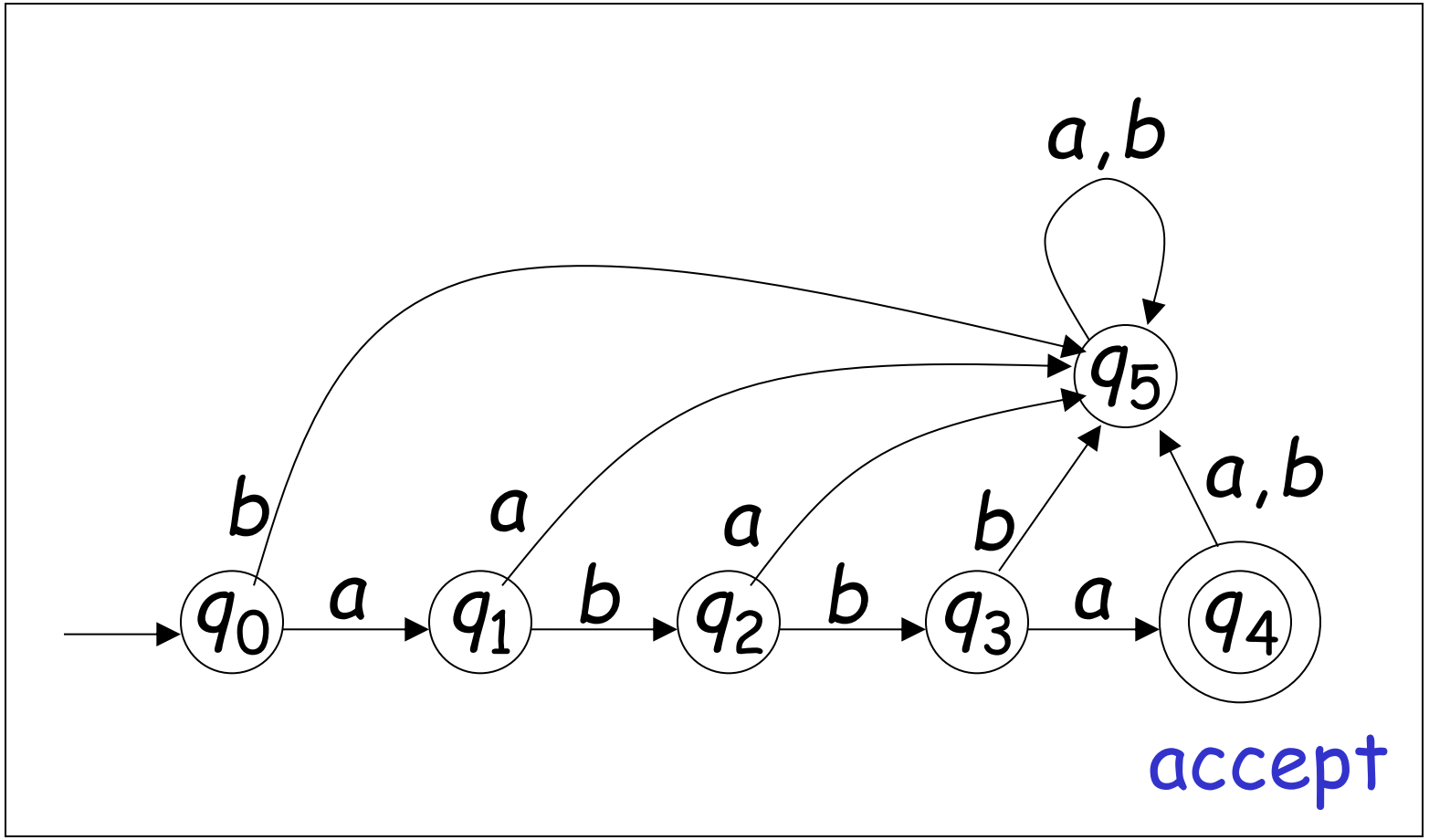
$$\Sigma = \{a, b\}$$



Example

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

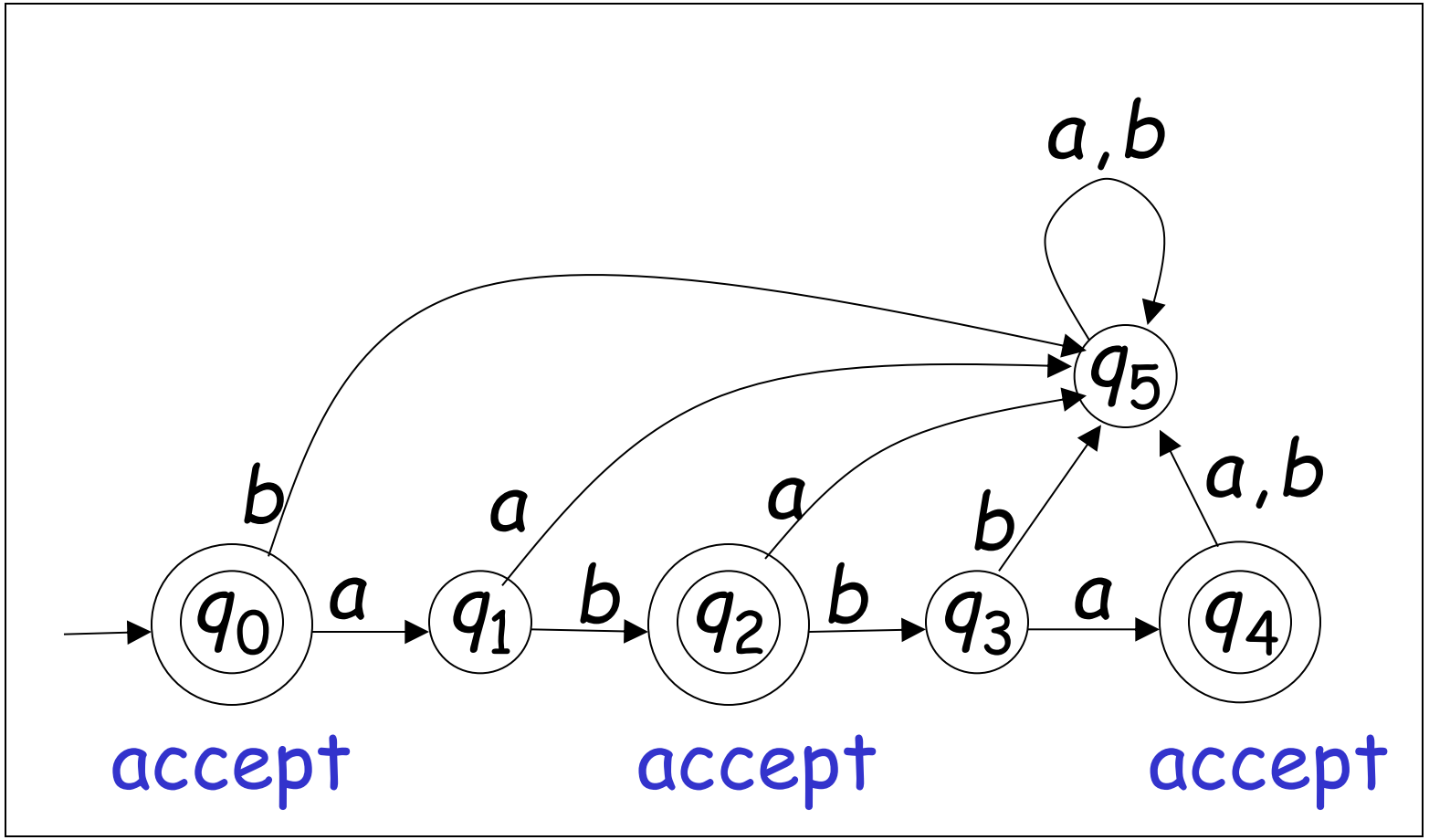
M



Another Example

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

M

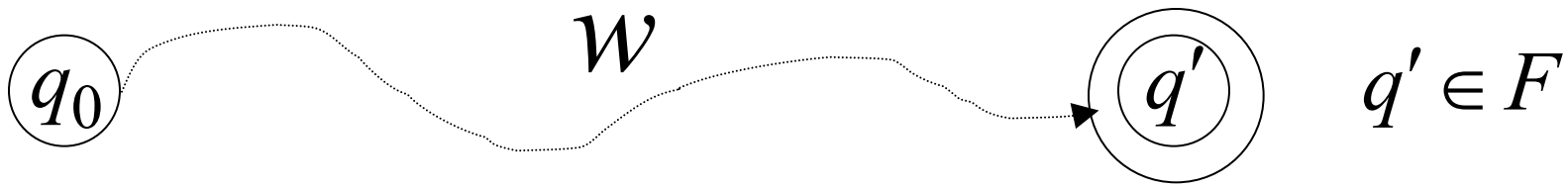


Formally

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

Language accepted by M :

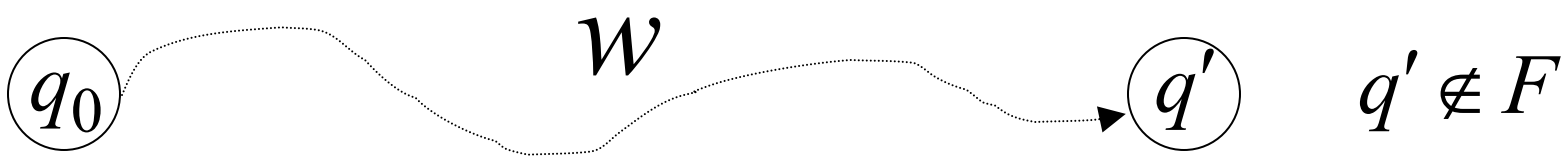
$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$



Observation

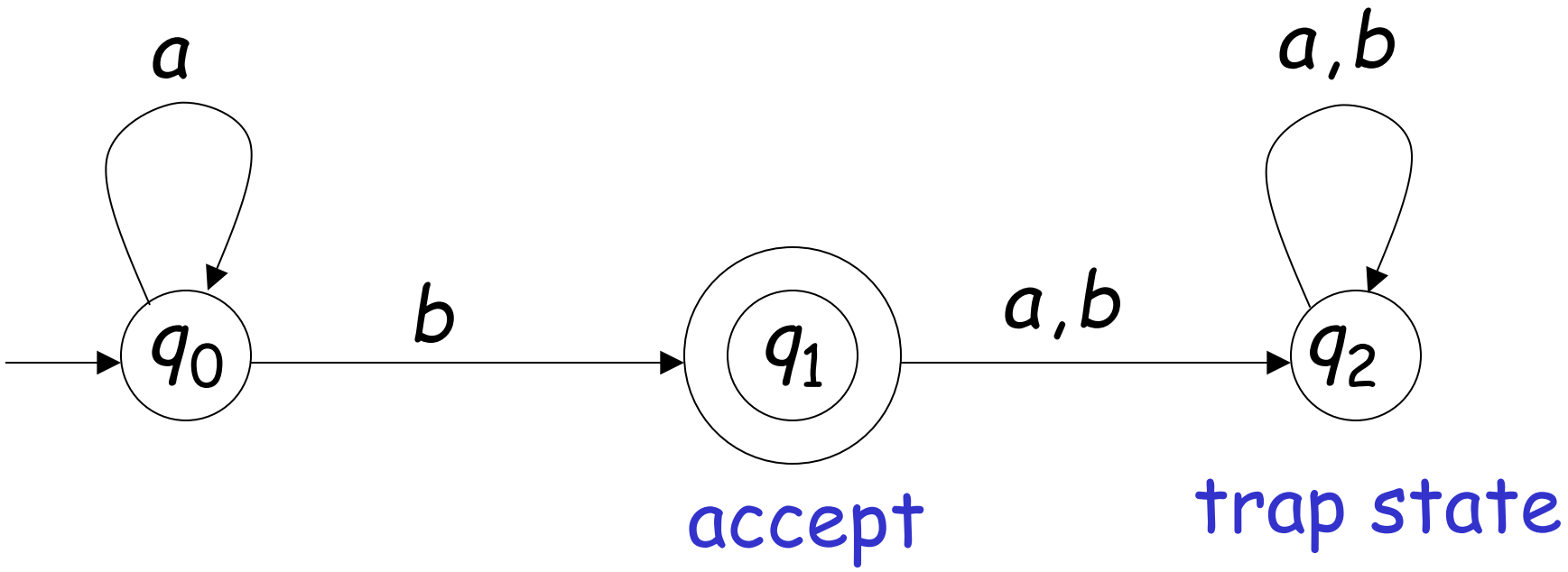
Language rejected by : M

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

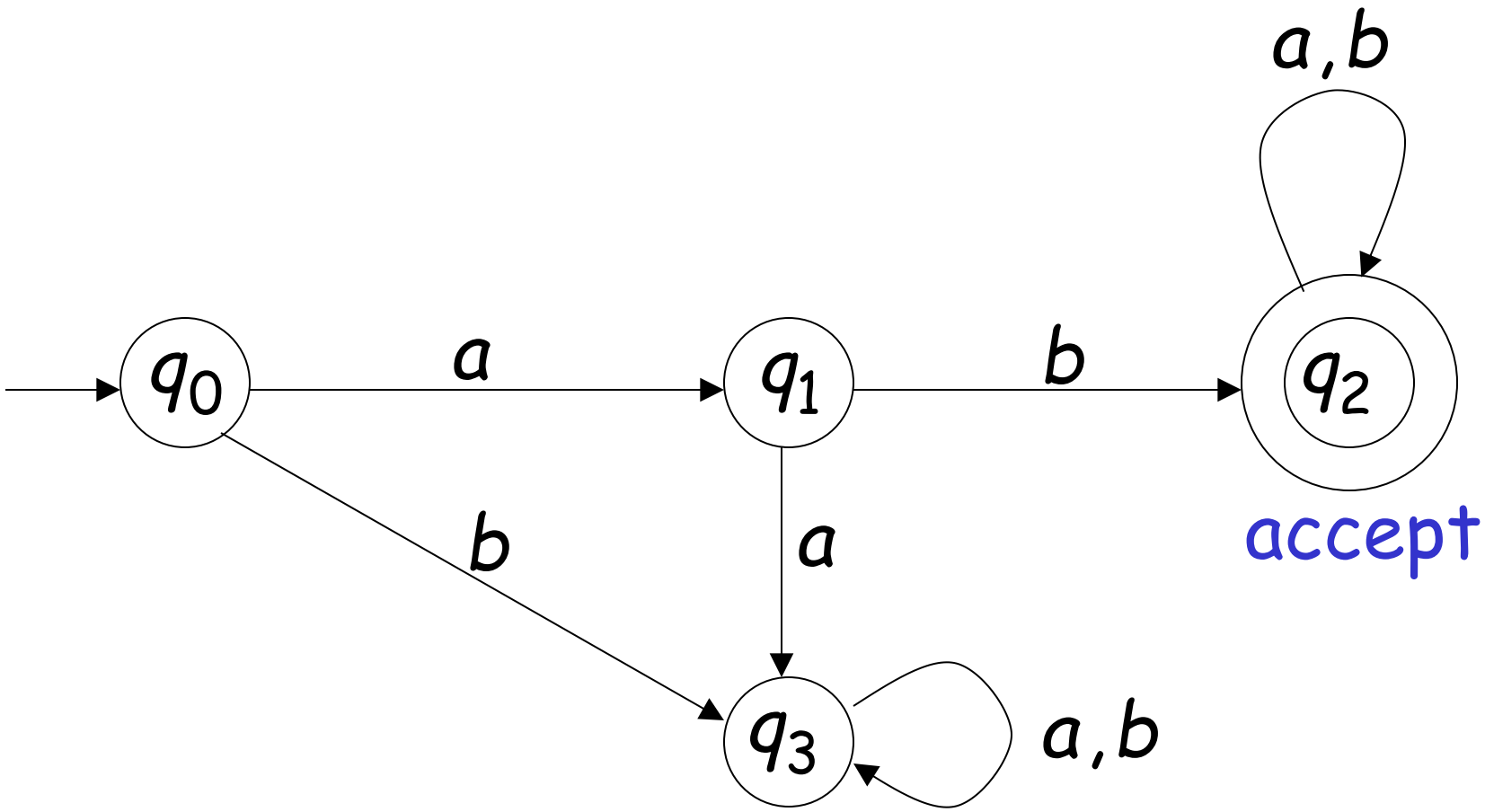


More Examples

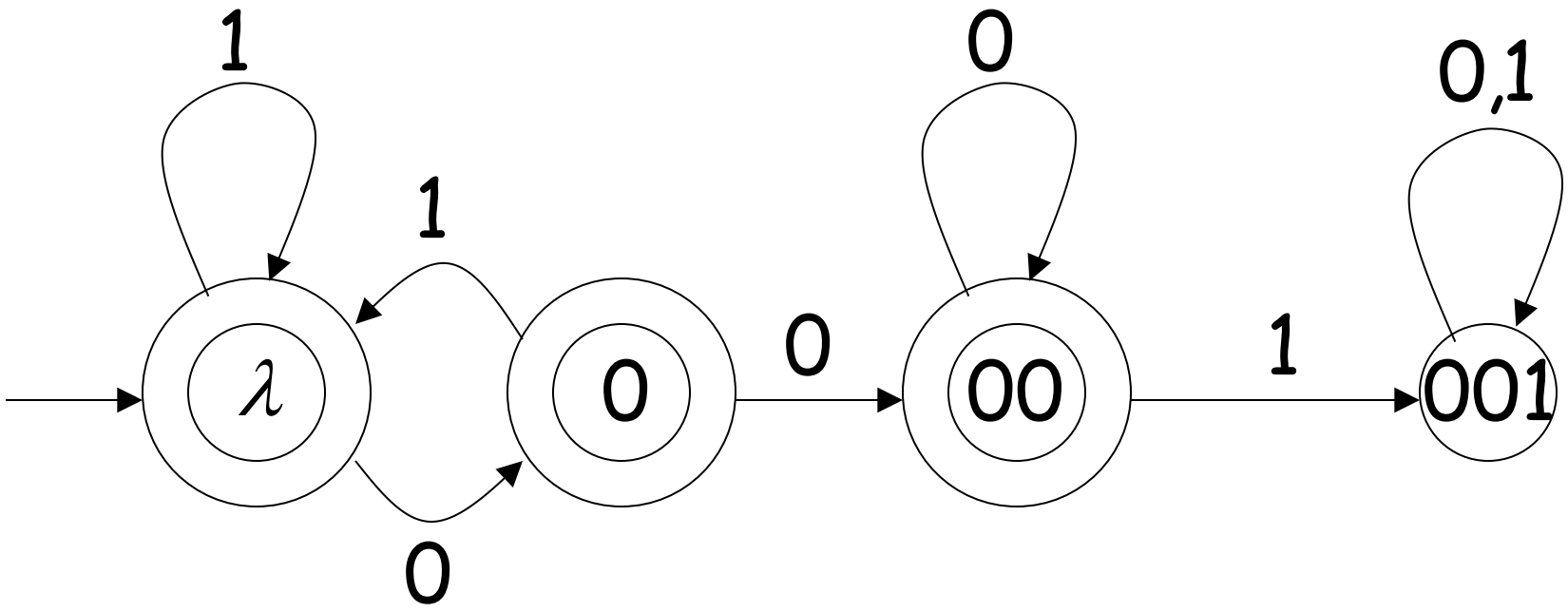
$$L(M) = \{a^n b : n \geq 0\}$$



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



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



IV. How to construct DFA machine

3-steps Method

Test Set:

- Write the set members both for strings that are accepted and for strings that are rejected.

Designing the automaton

- Determine the number of states, then the transitions

Use the predetermined test set to test the constructed automaton

3-steps Method

- Test Set:

  Tuliskan anggota himpunan baik untuk string yang diterima maupun string yang ditolak

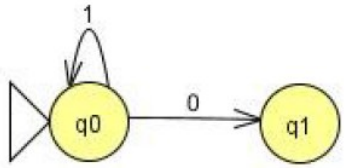
- Merancang automata

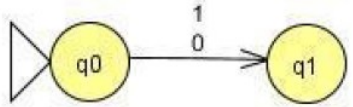
  Tentukan jumlah status, kemudian transisinya.

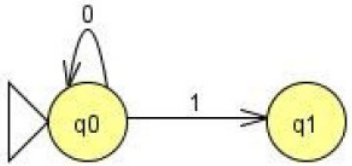
- Gunakan test set yang telah ditetapkan untuk menguji automata yang dibangun

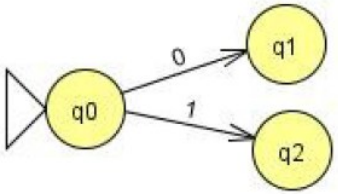
Merancang Automata

- State dapat diwakili dengan kondisi string yang telah dibaca. Setiap state memiliki arti/interpretasi yang bisa dijelaskan dengan sebuah comment pada program
  - State 0: sudah membaca empty string
  - Teruskan ke state berikutnya yaitu dari state 0 dengan satu simbol input
- State 0 dan state 1







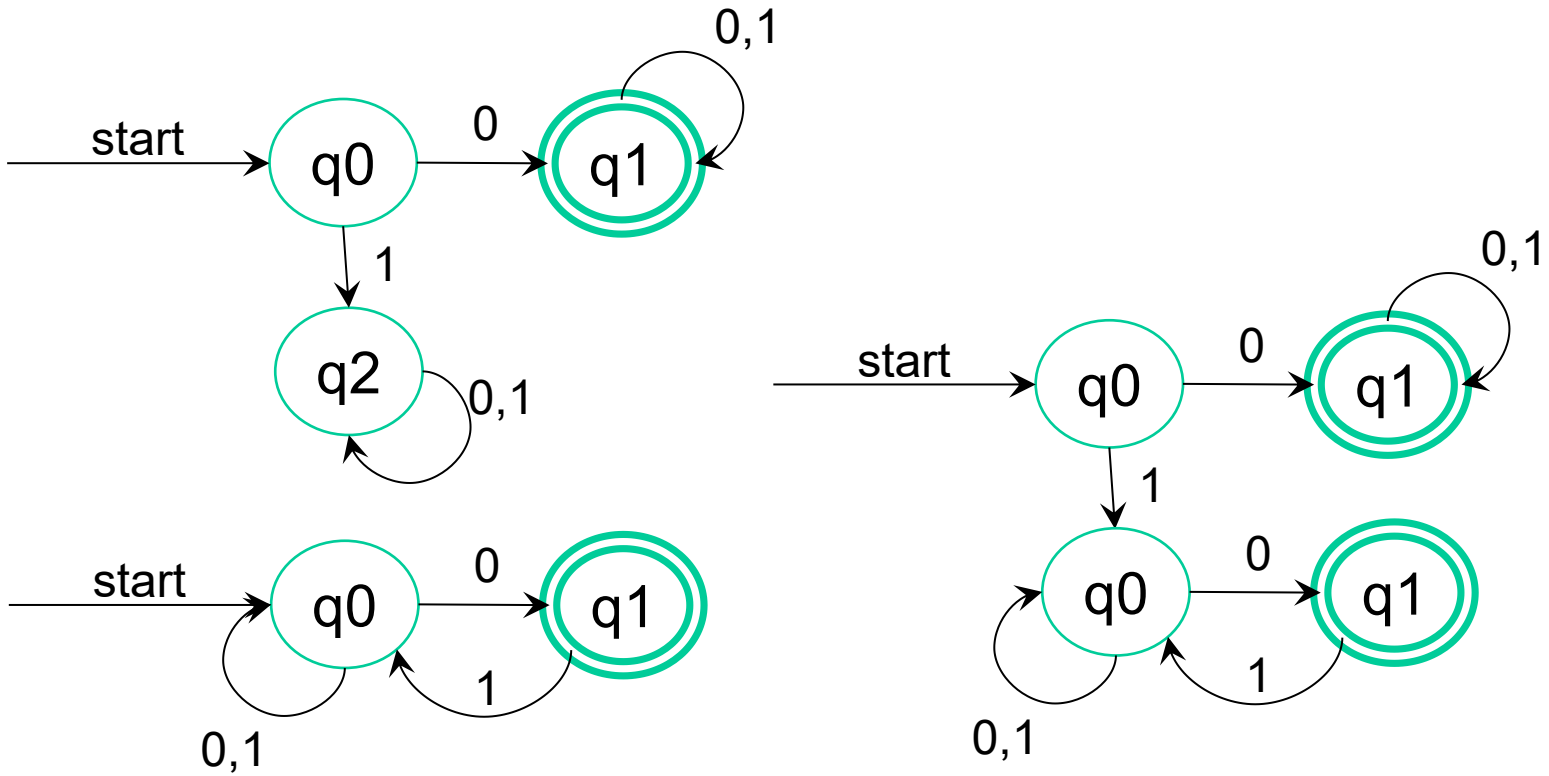


Aturan Khusus

- **Jika language mengandung "or" dari dua sub language**
  - Buat 2 automata, @satu untuk ksetiap sub language
  - Gabungkan kedua automata dengan operasi "or" (cabang) dariawal
- **Jika language mengandung "not"**
  - Buat automata untuk language yang positif, tidak mengandung "not"
  - Ubah semua accepting state menjadi not-accepting dan juga sebaliknya
- **Jika language mengandung "and" dari dua sub language**
  - Buat 2 automata, @satu untuk setiap sub language
  - Gabungkan kedua automata dengan operasi "and" (concatenation)
  - Jika language mengandung "reverse"
  - Buat automata yang tanpa reverse
  - Balikkan semua arah pada busur
  - Ubah start state menjadi final state
  - Tambahkan initial state menuju semua accepting state

Contoh

Buat DFA yang menerima string yang diawali dengan simbol 0 atau string yang diakhiri dengan simbol 0



Exercise

Example: **Let** L = { w | w is a multiple of 4 when interpreted as a binary integer and is read form right to left}.