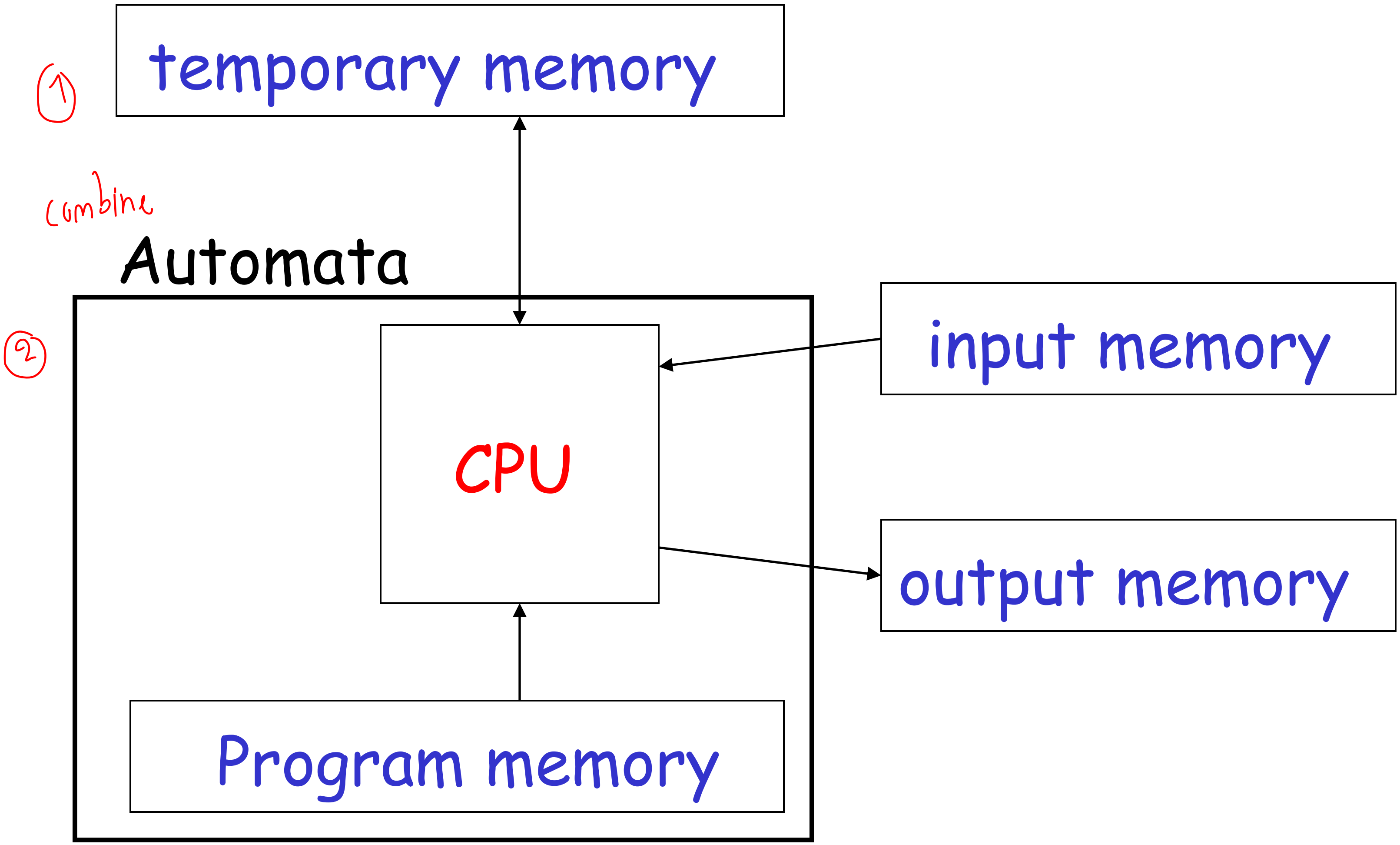
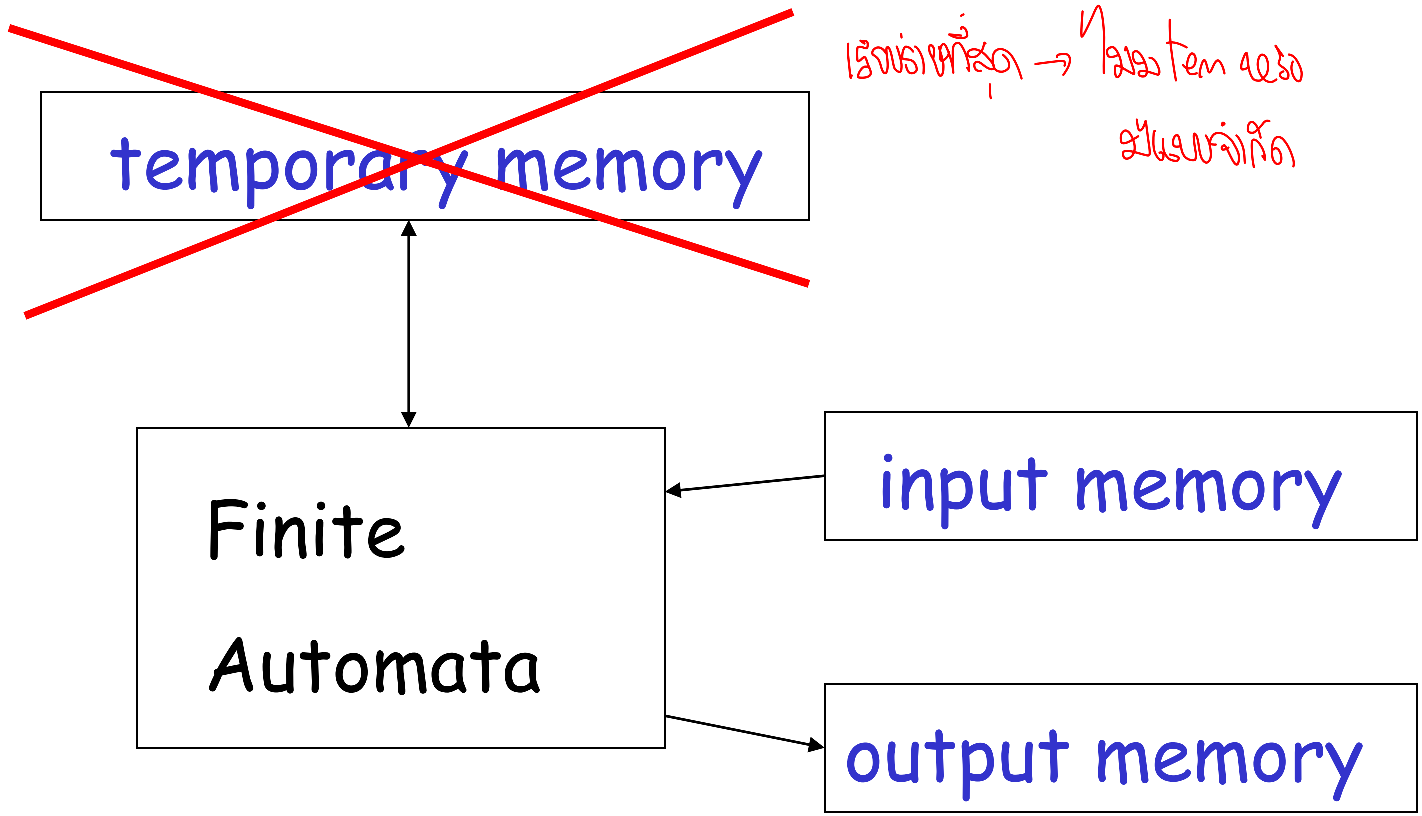
Finite Automata

Automata



temporary memory
1
combine
Automata
2
CPU
input memory
output memory
Program memory

Finite Automata



เรียบง่ายที่สุด → ไม่มี tem memo
มีแบบจำกัด

Example: Vending Machines → เครื่องขายของอัตโนมัติ

(small computing power)

Finite Automata

The simplest form of automata.

Input อยู่ในรูปแบบ tape ที่ยาวไม่จำกัด

String ← ~~abcecef~~ หรือมองเป็น source code ก็ได้

↓

Finite Automata →

Output → finite เช็คถูก ต้องตามที่ กำหนดไว้ไหม

String

↓ ถูกกฎ , ผิดกฎ

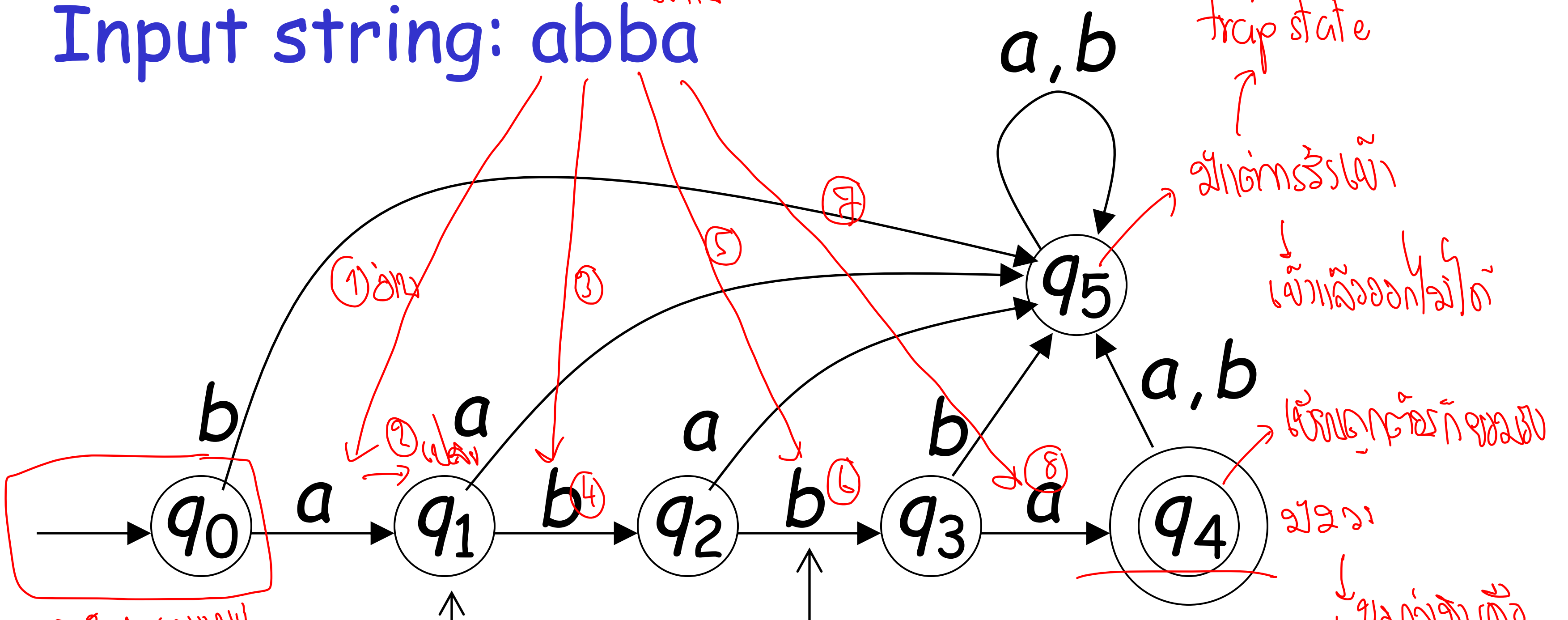
"Accept"
or
"Reject"

ตัวอย่าง

Transition Graph

เปลี่ยนสถานะ

Input string: abba



initial state

เริ่มทำงานทุกครั้งที่มี input ใหม่

state

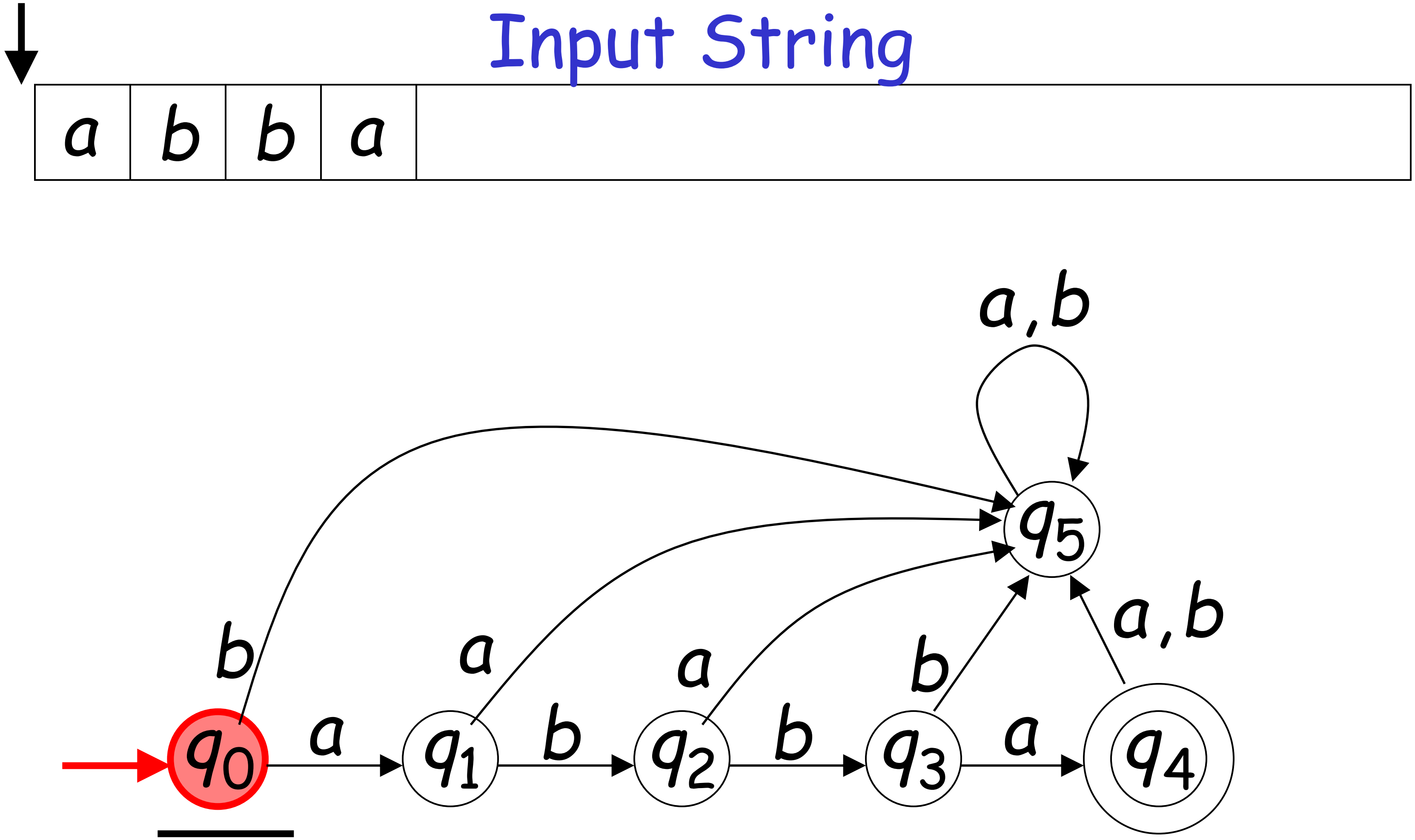
transition

final state

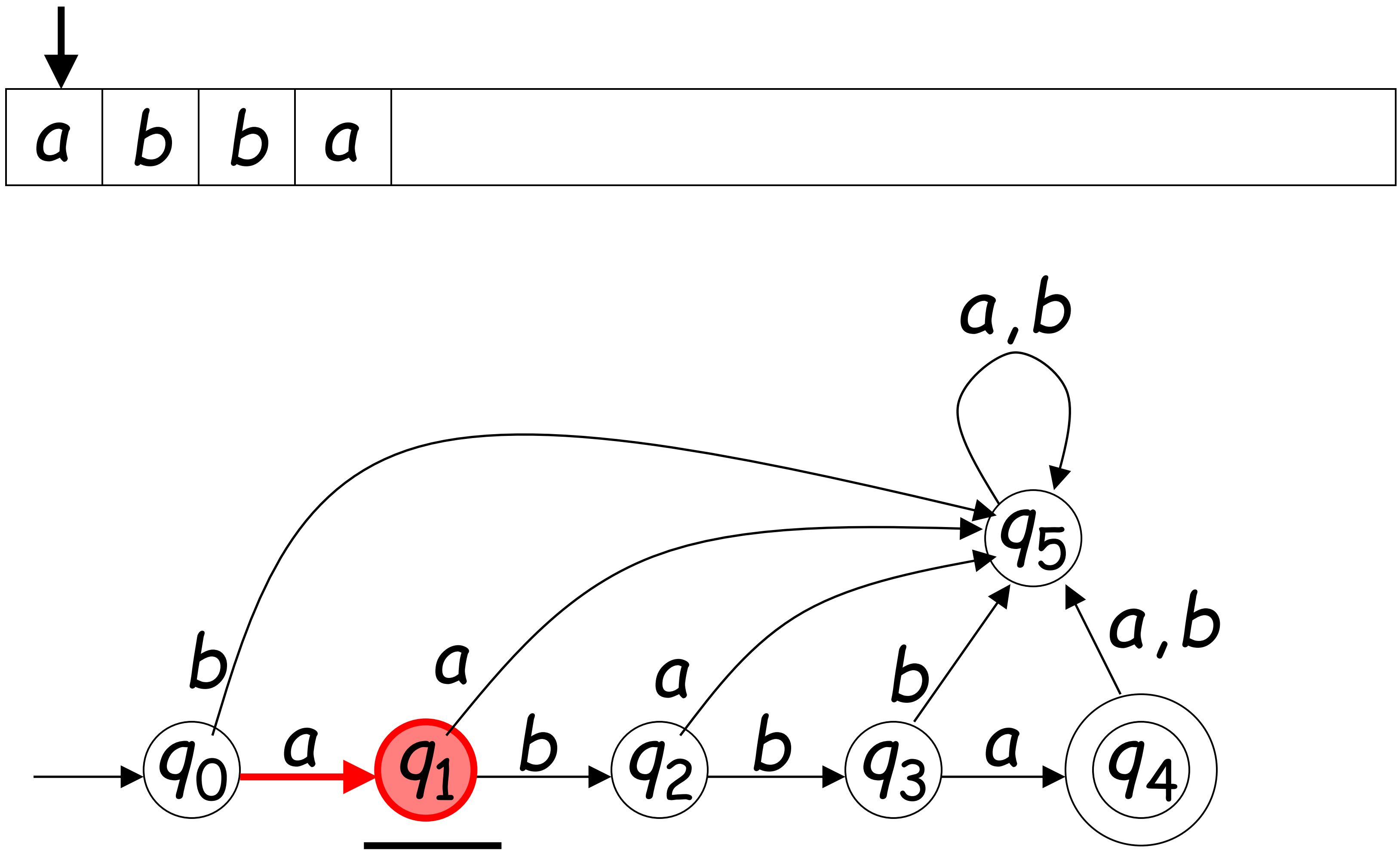
"accept"

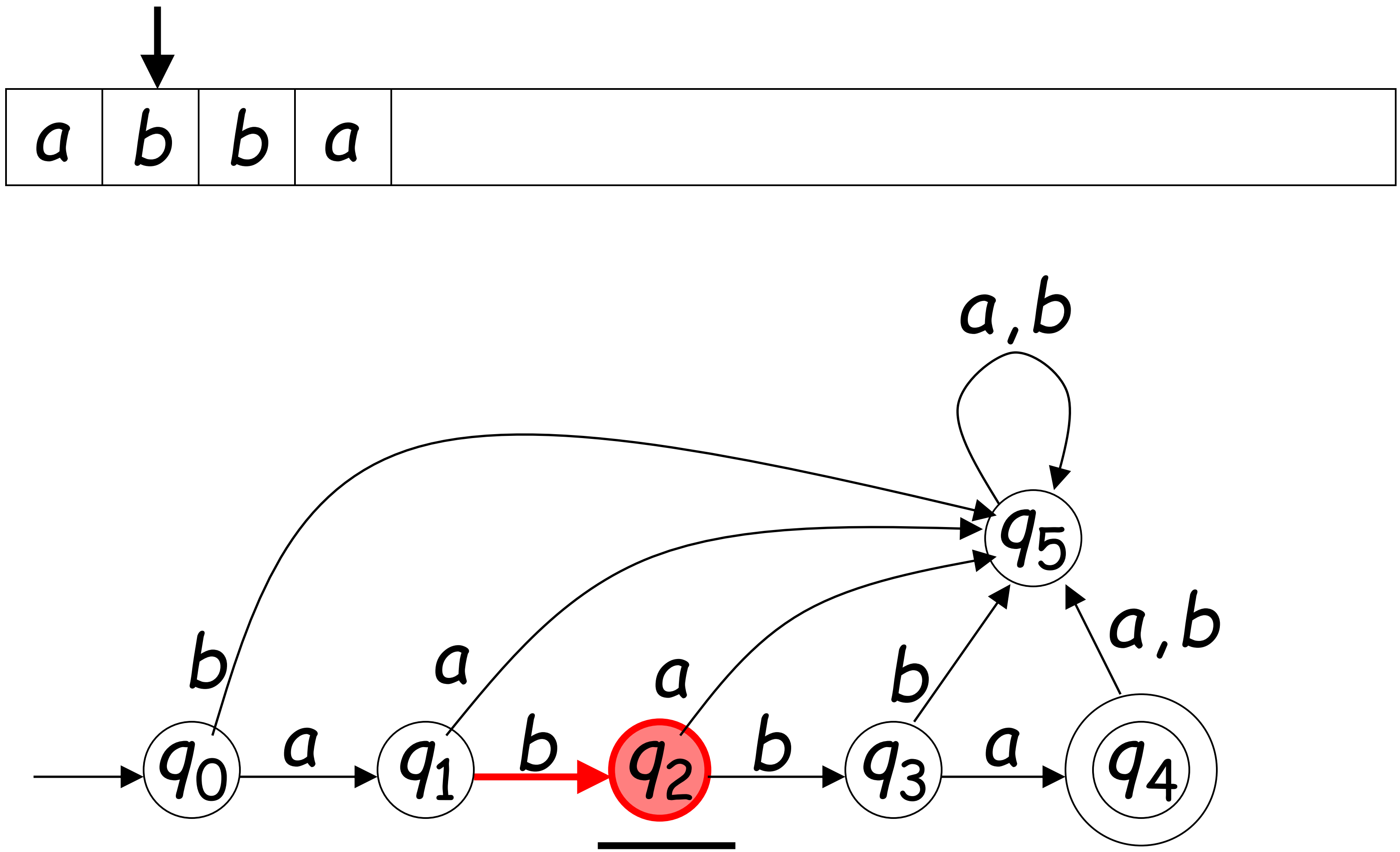
ถ้าจบตรงนี้จะยอมรับ

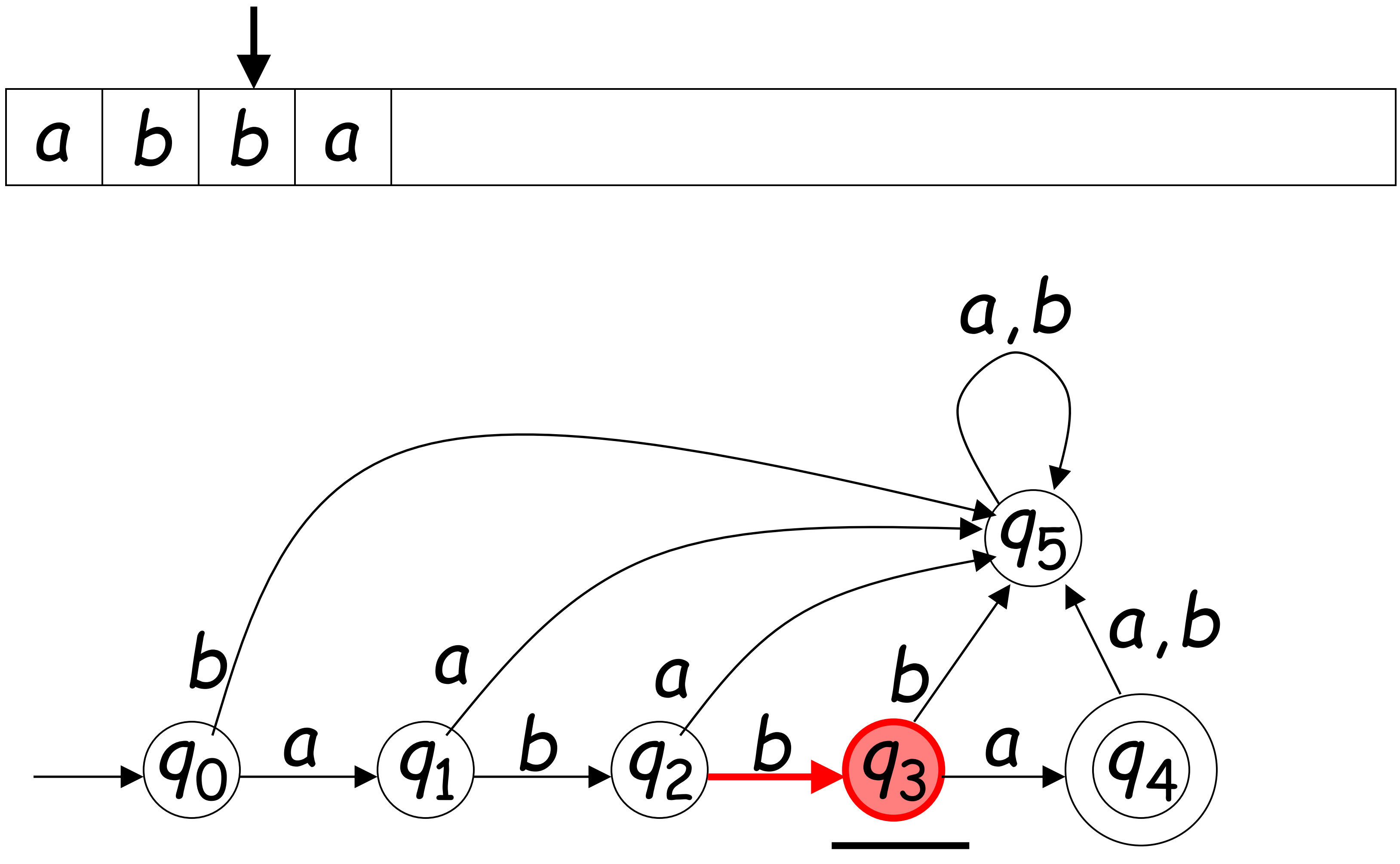
Initial Configuration

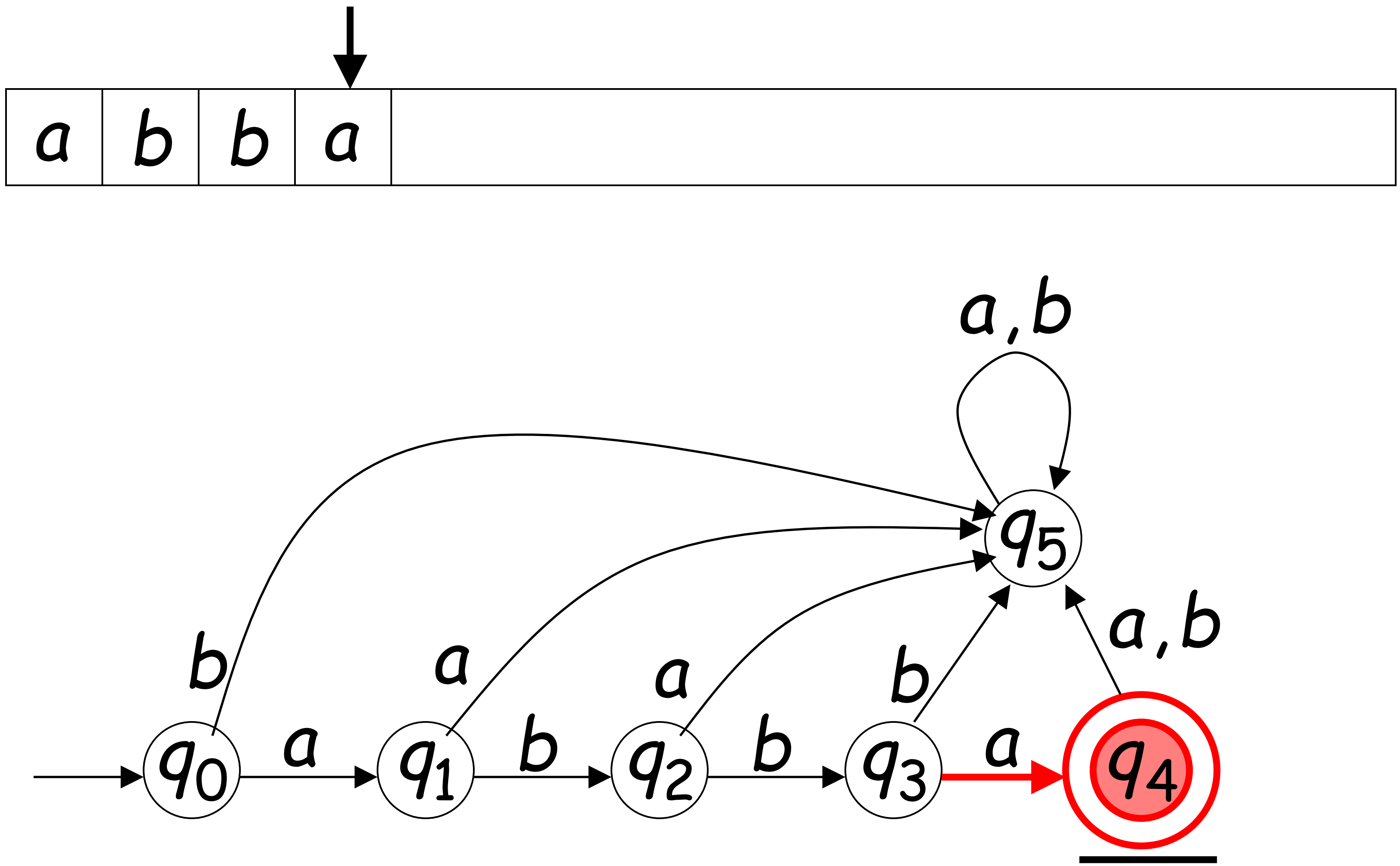


Reading the Input



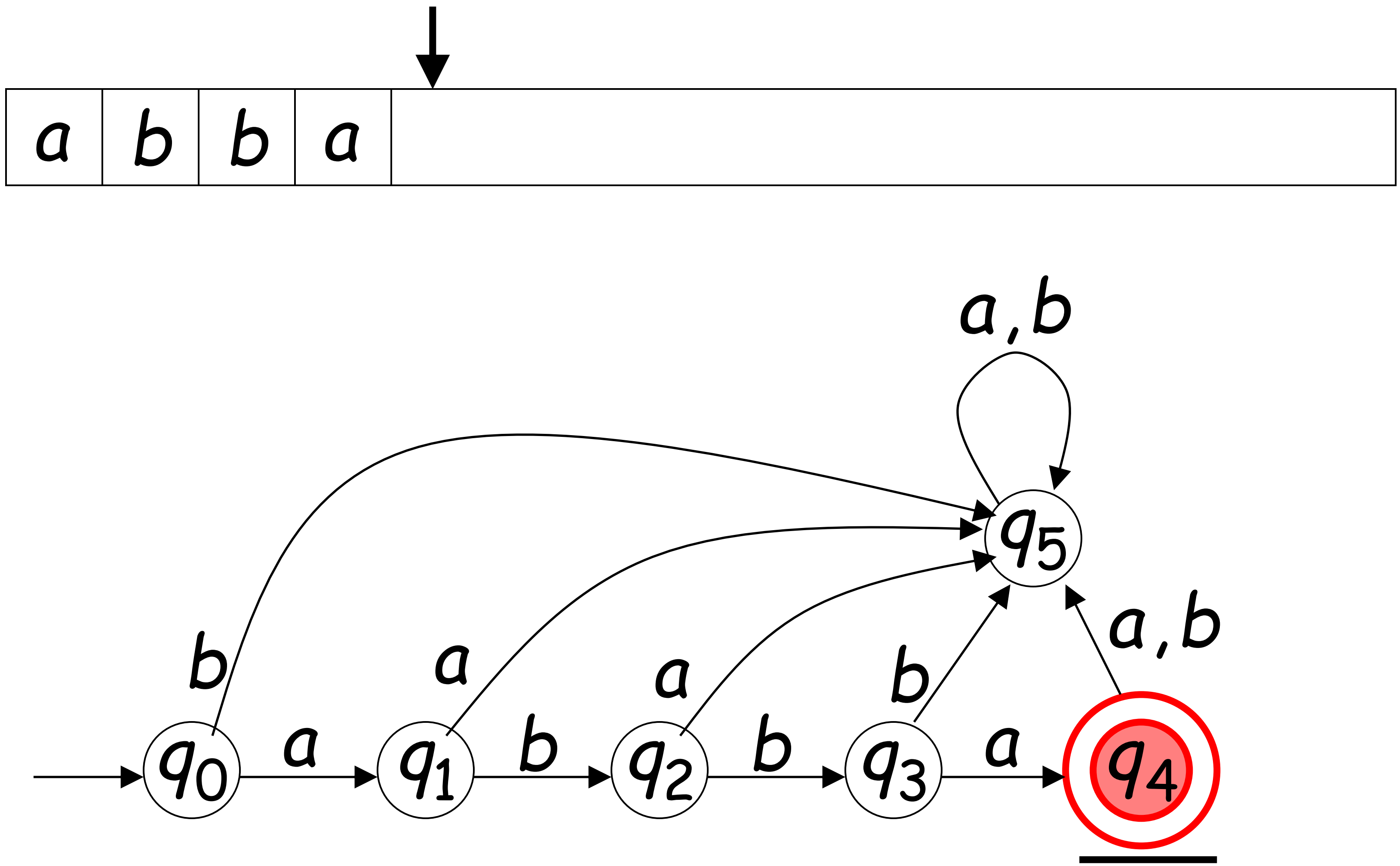

a b b a
a,b
b
a
a
b
a,b
q_0 a q_1 b q_2 b q_3 a q_4
q_5


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



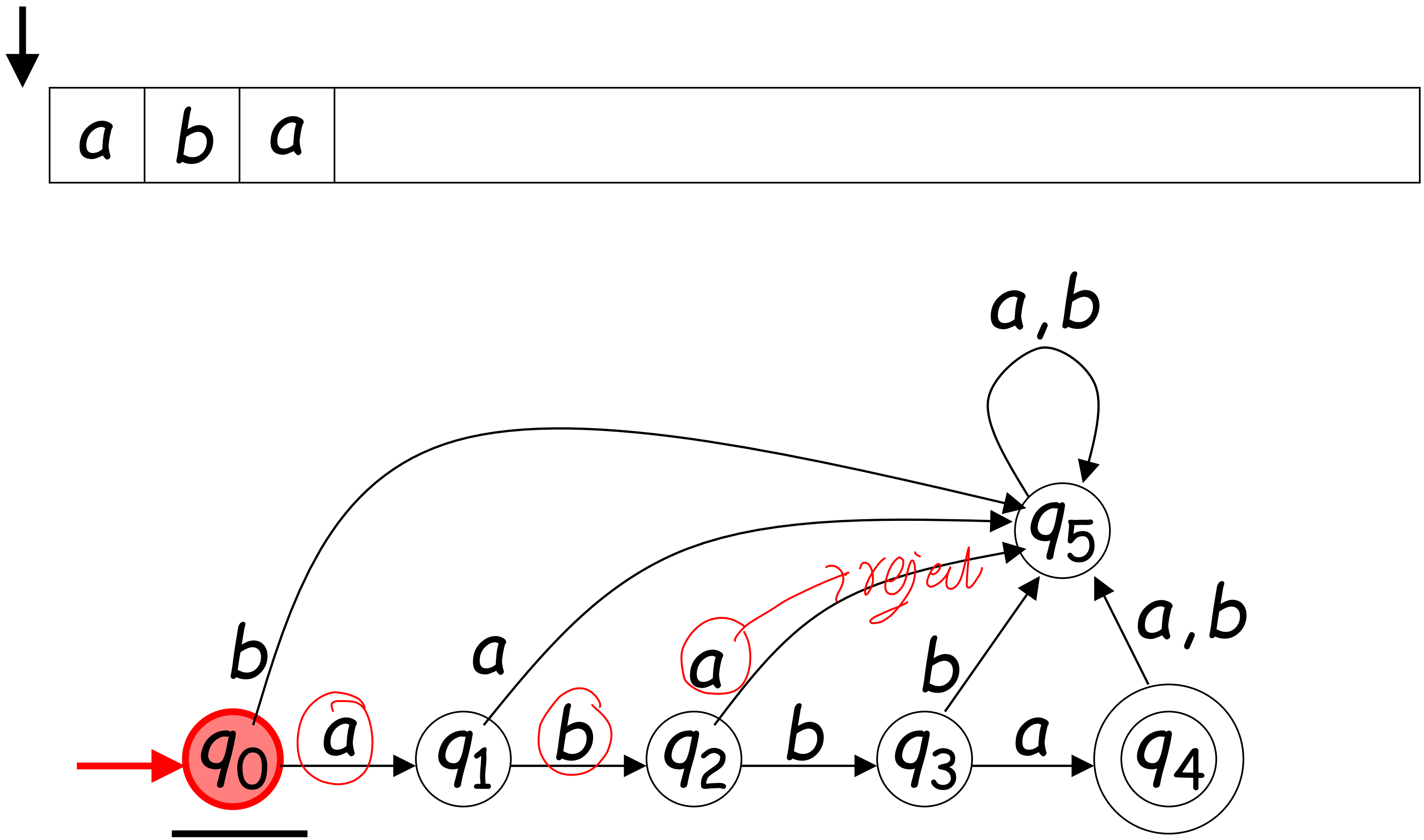
a
b
b
a
a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

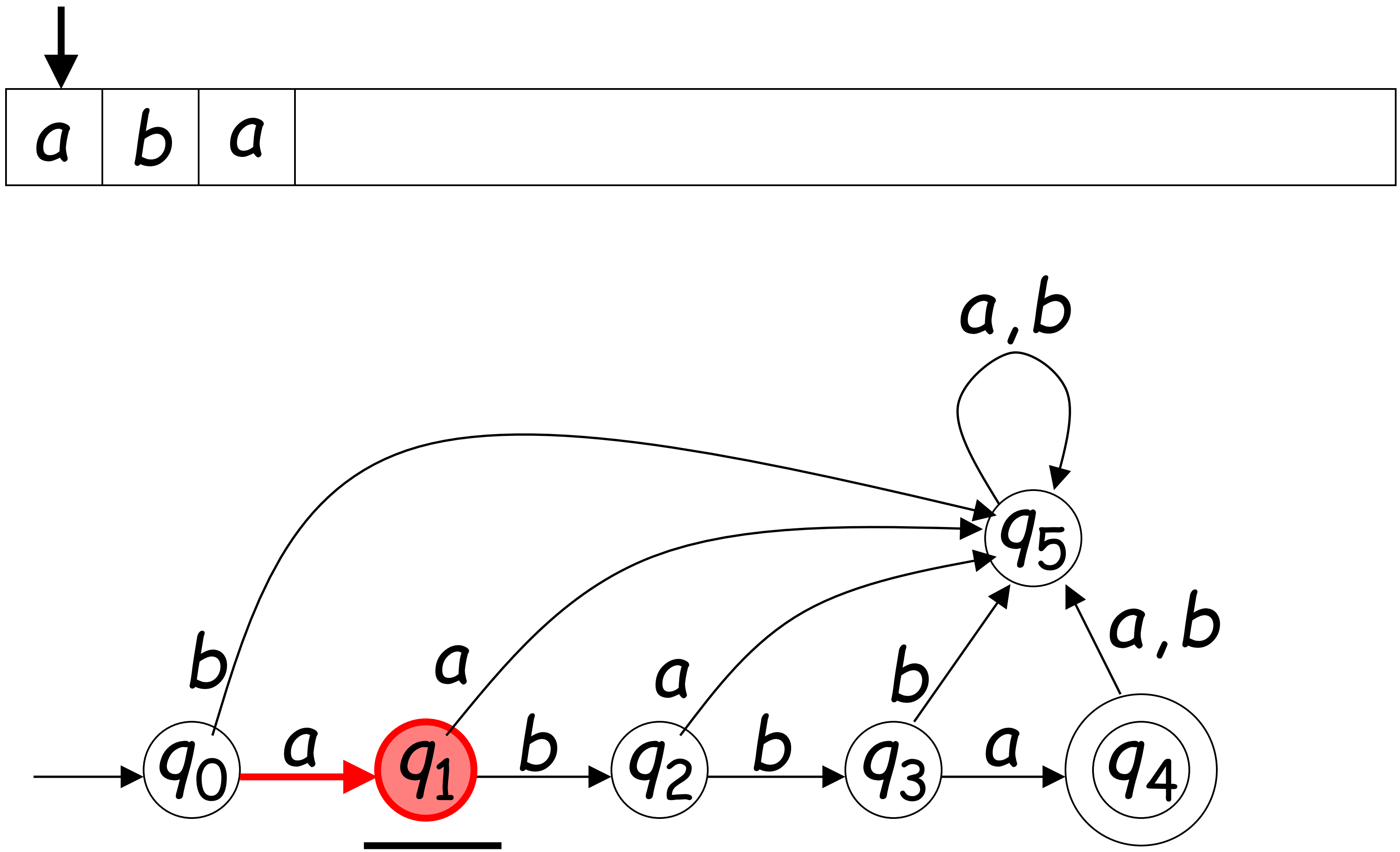
Input finished



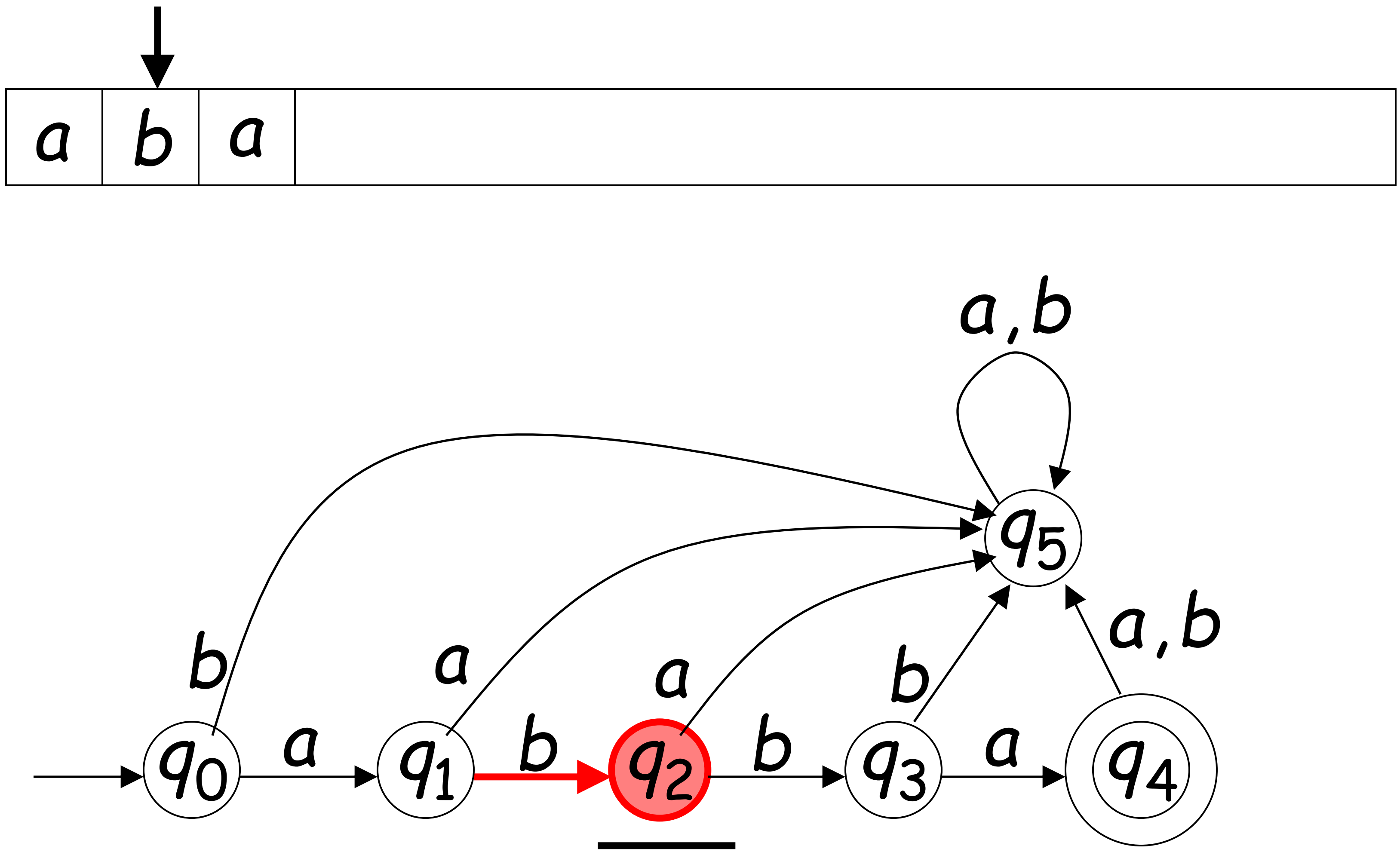
Output: "accept"

Rejection

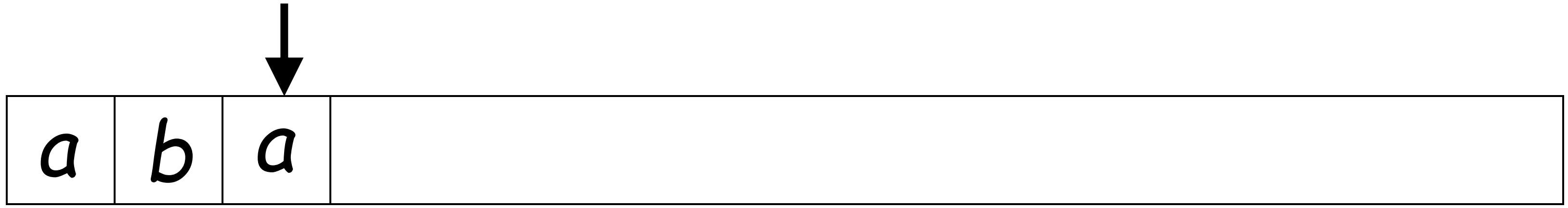


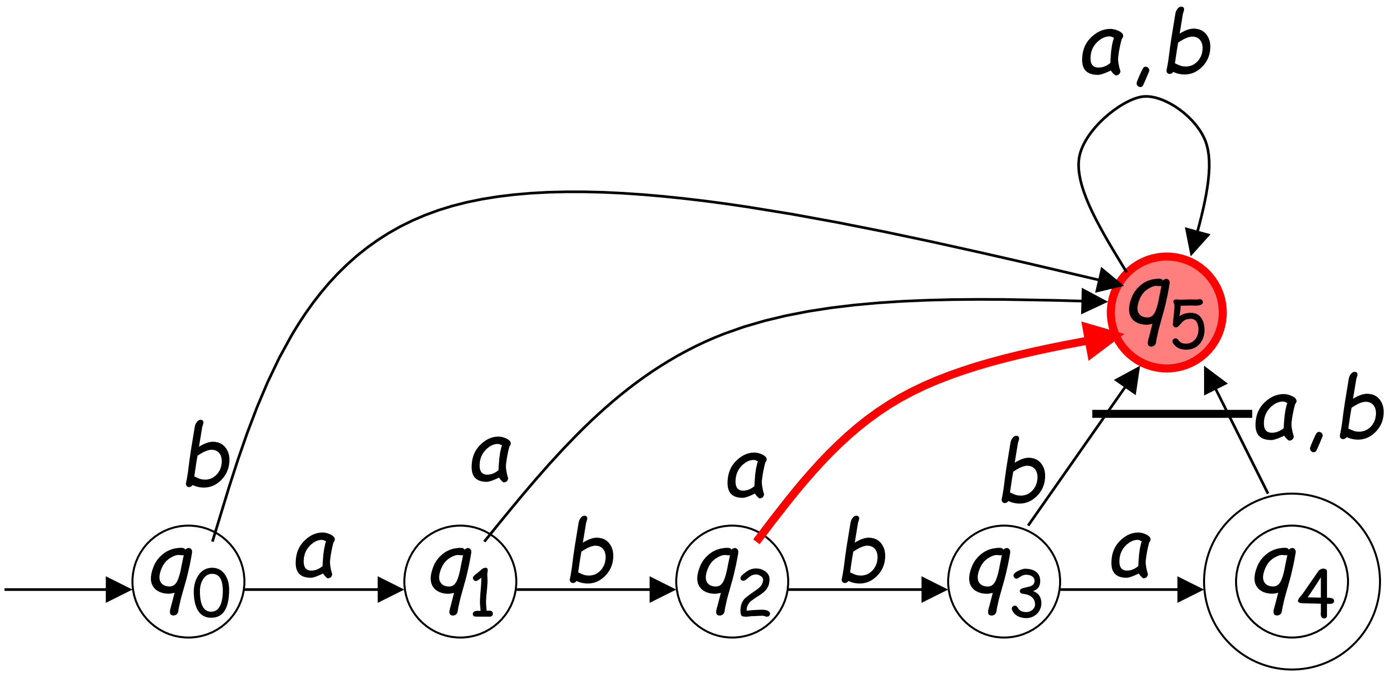


a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

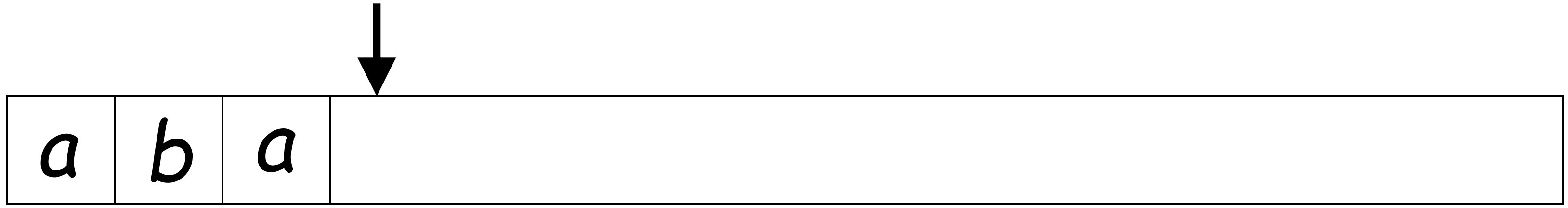


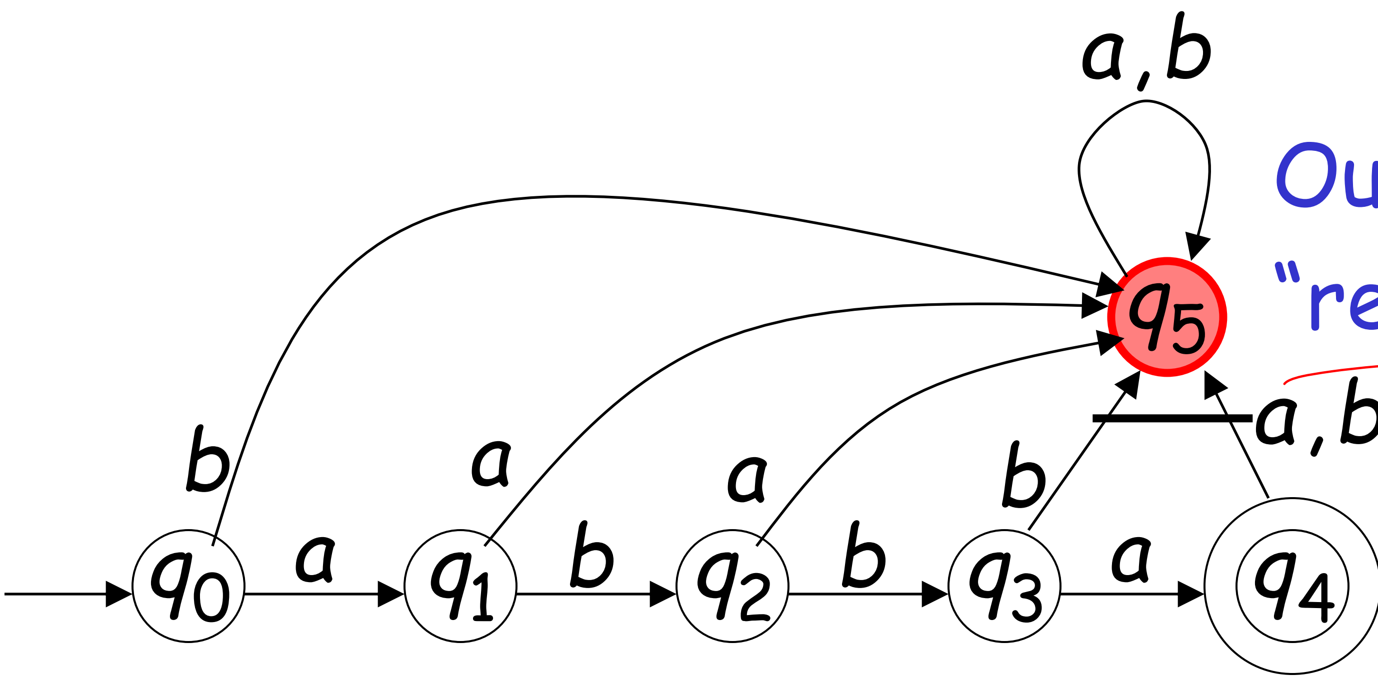
a b a
a,b
q_5
b
a
a
b
a,b
q_0 a q_1 b q_2 b q_3 a q_4


a
b
a


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

Input finished

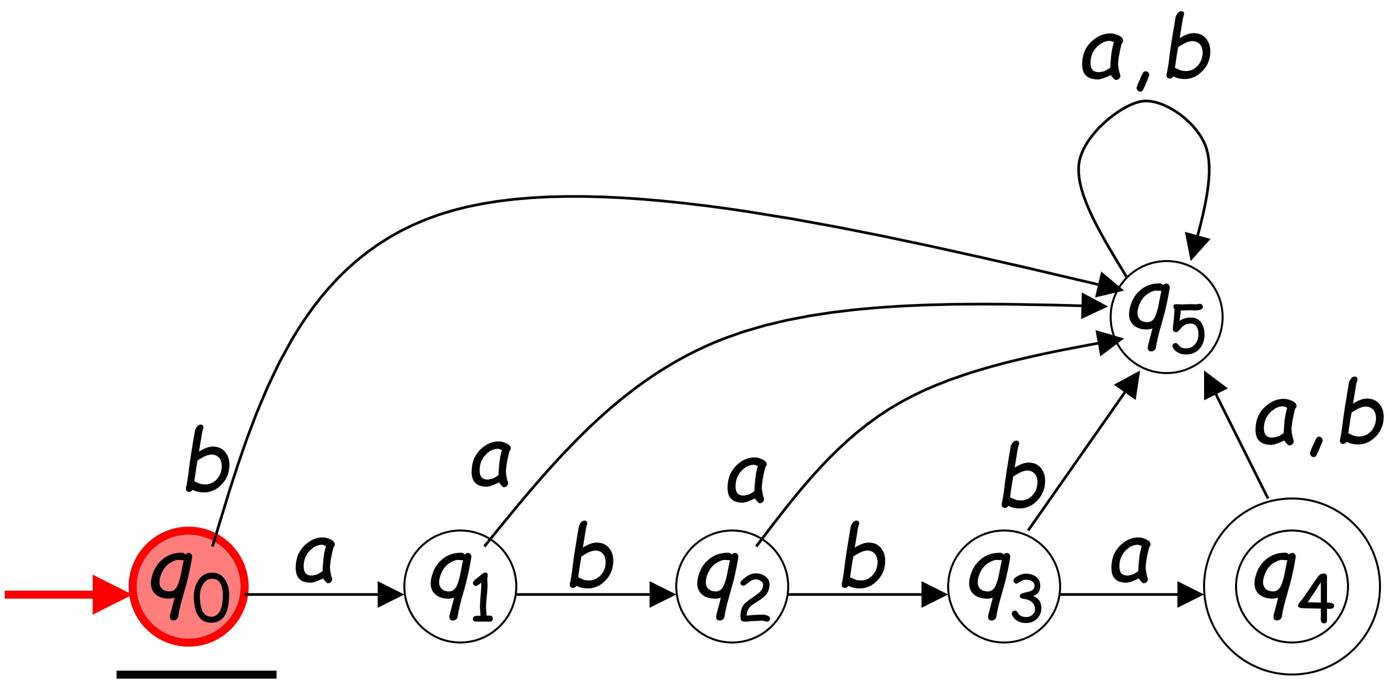




Output:
"reject"

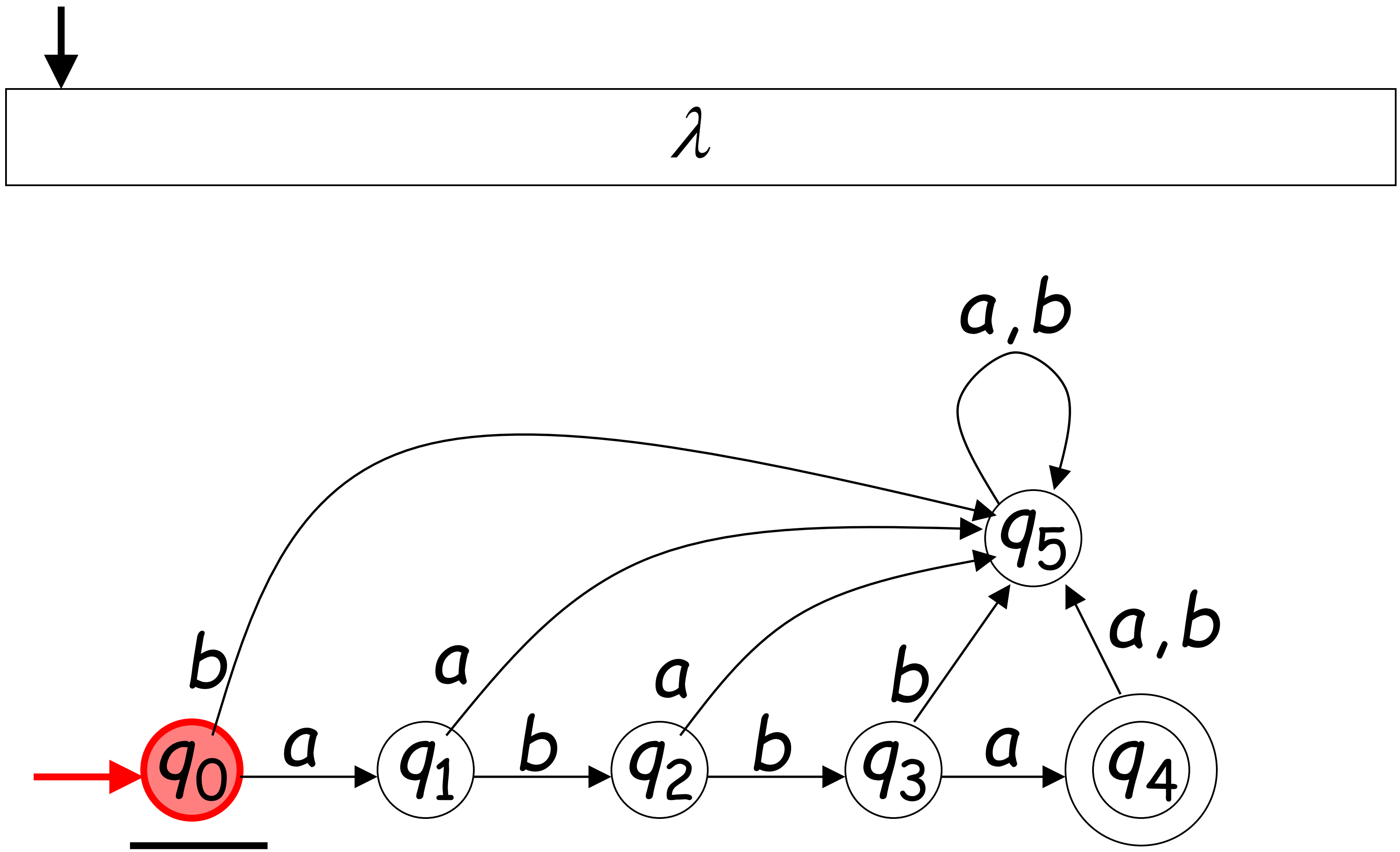
Another Rejection

λ empty string



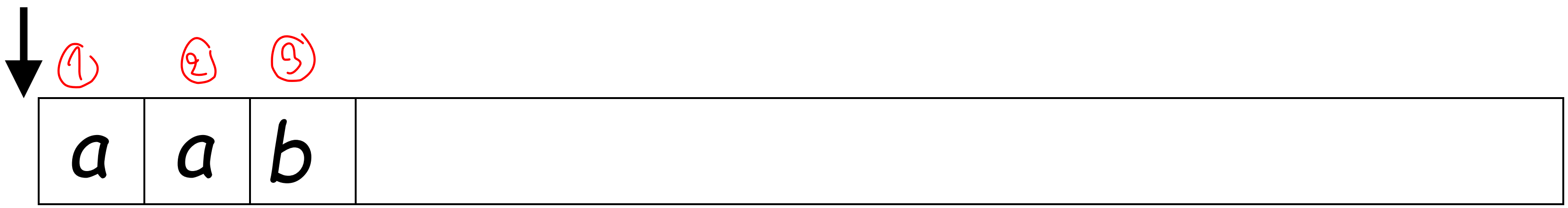
output
reject

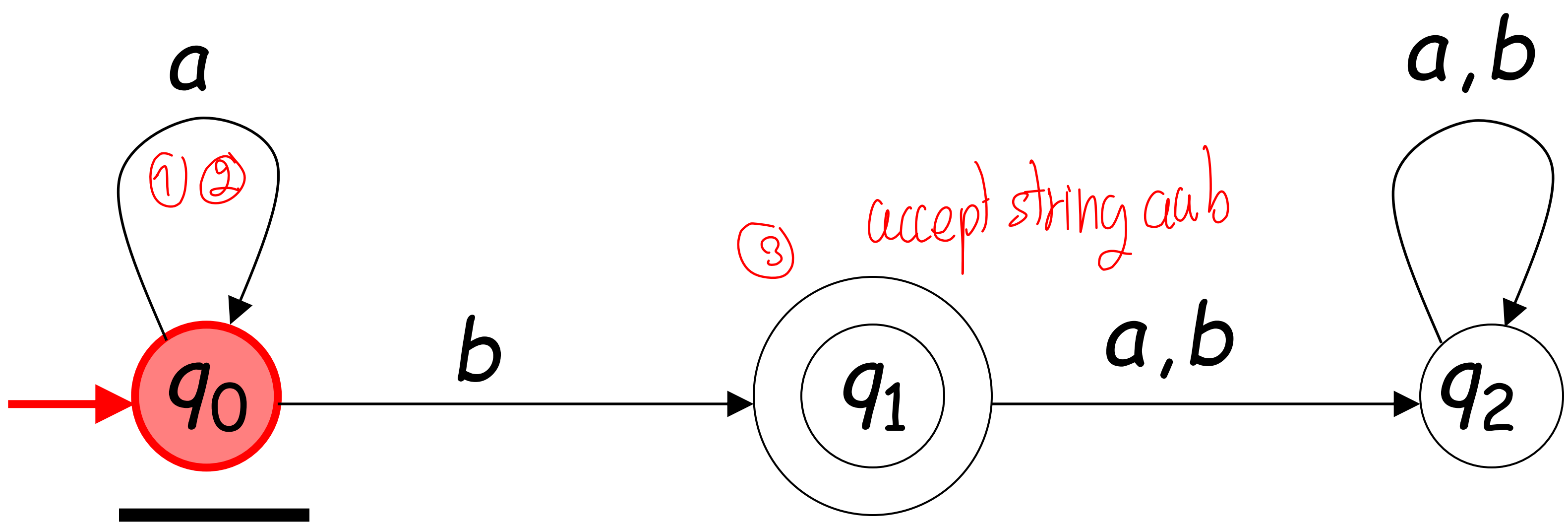
เงื่อนไข match condition

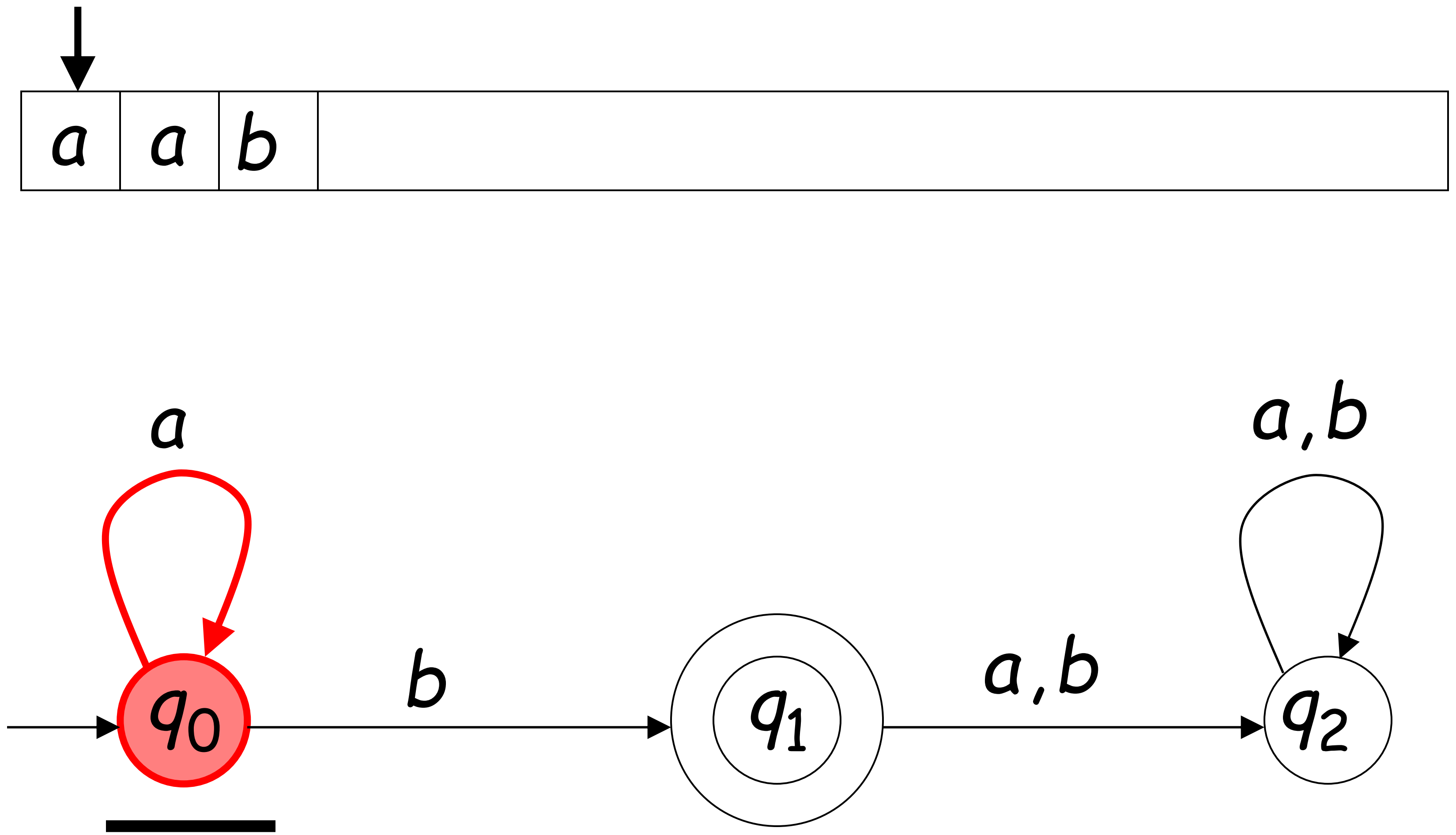


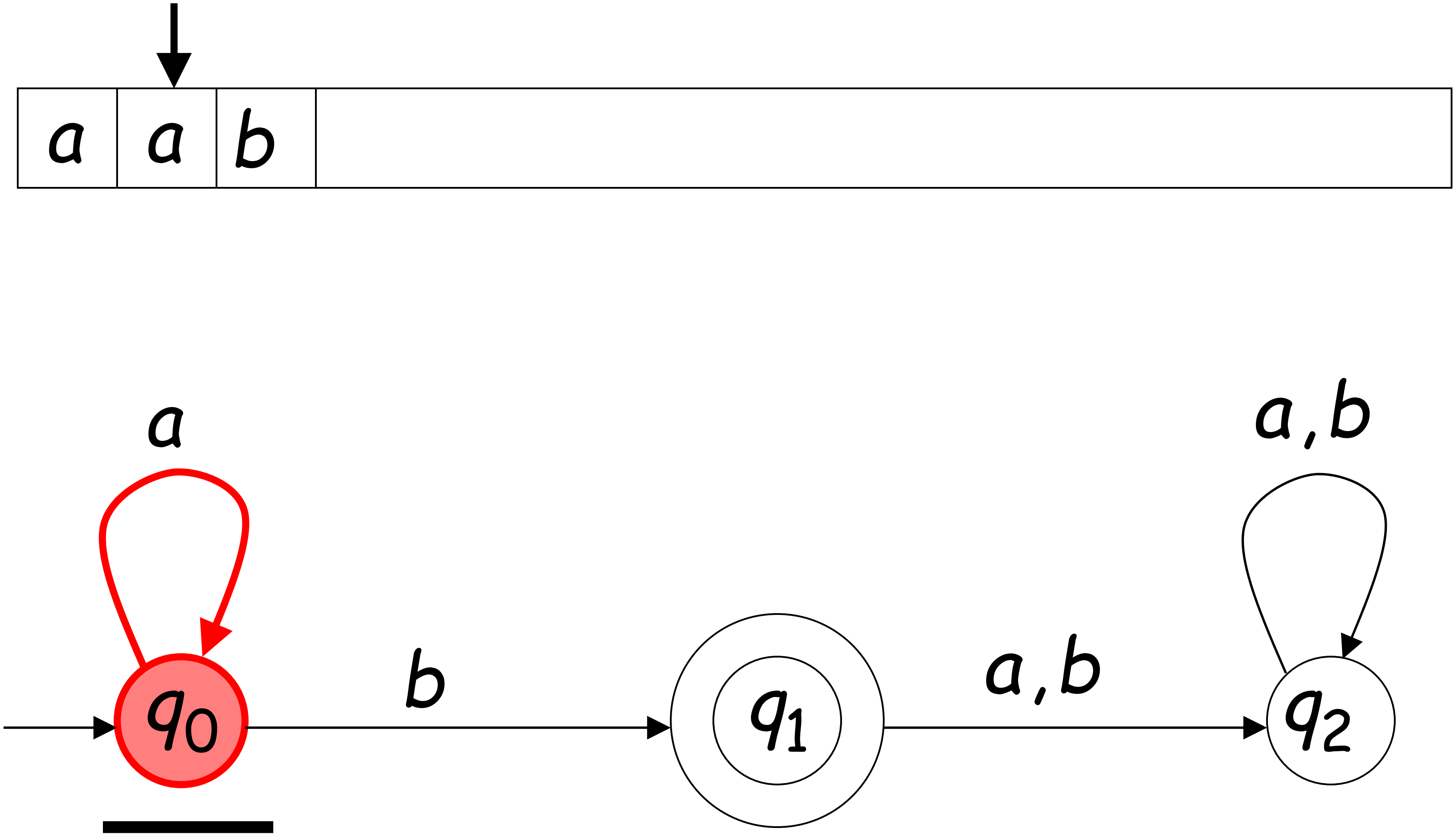
Output:
"reject"

Another Example

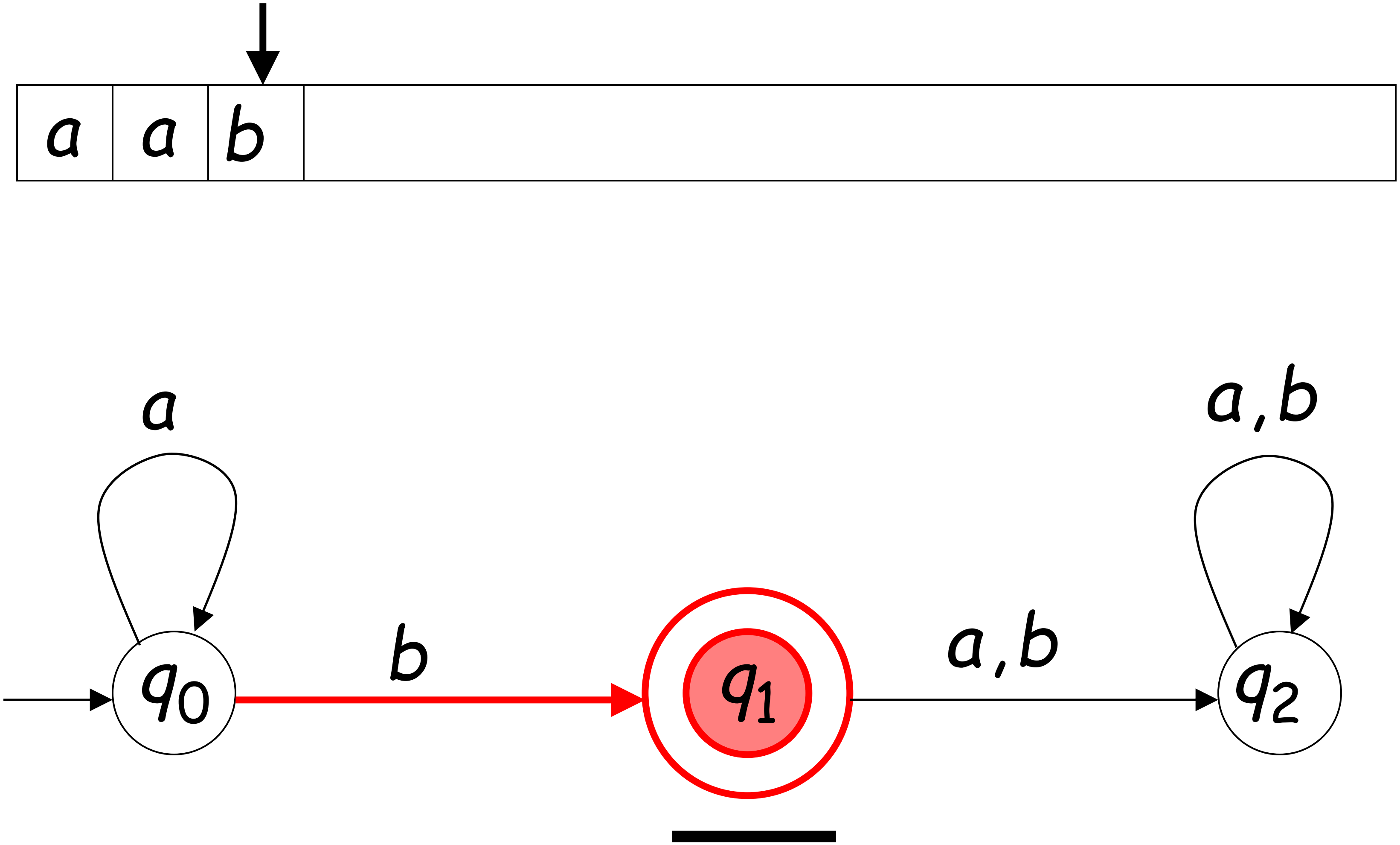





a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

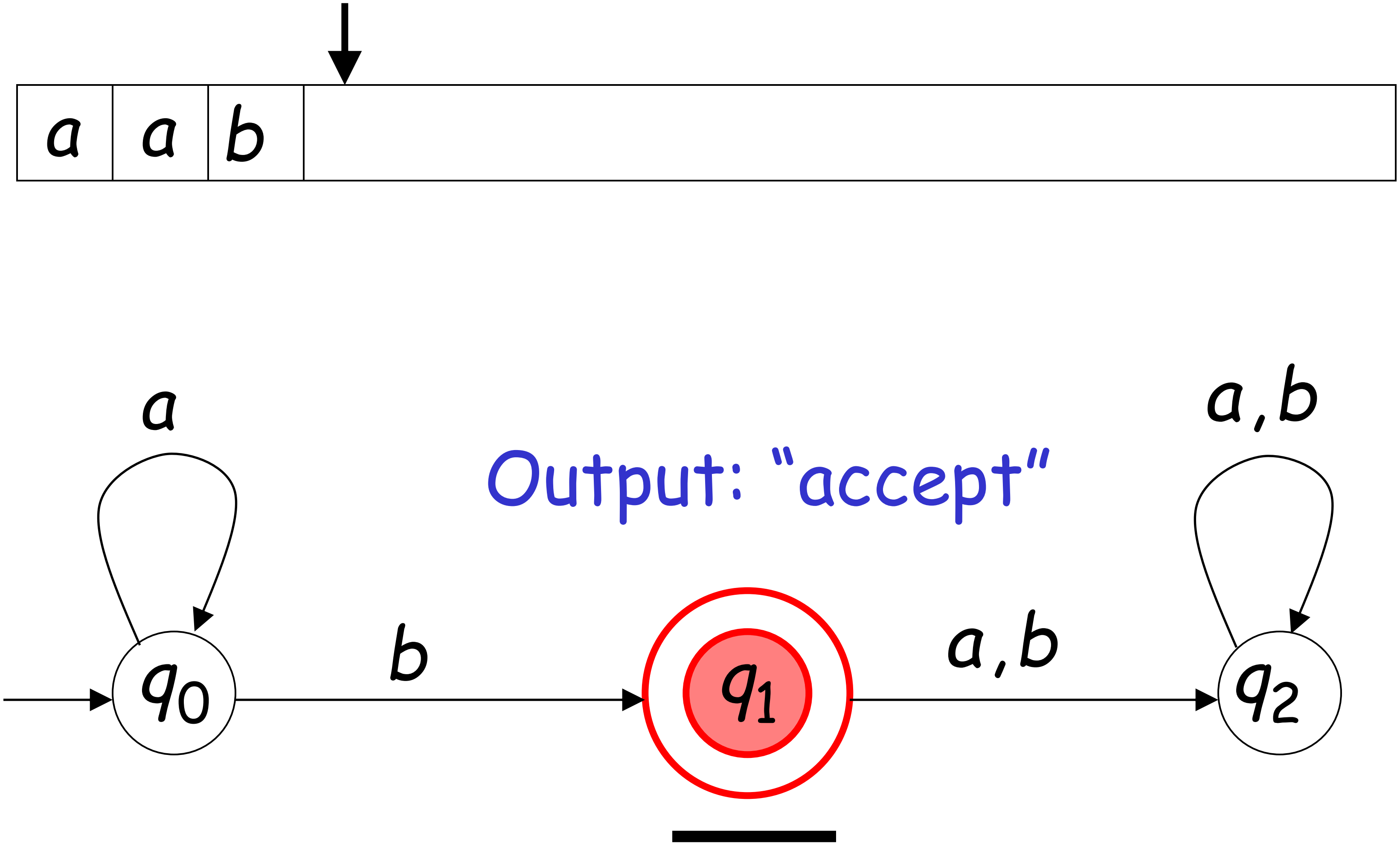


a
a
b
a
b
a,b
a,b
q_0
q_1
q_2



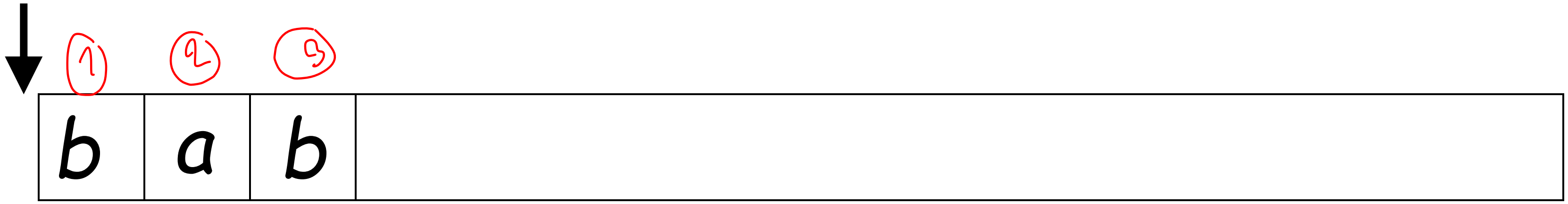
a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

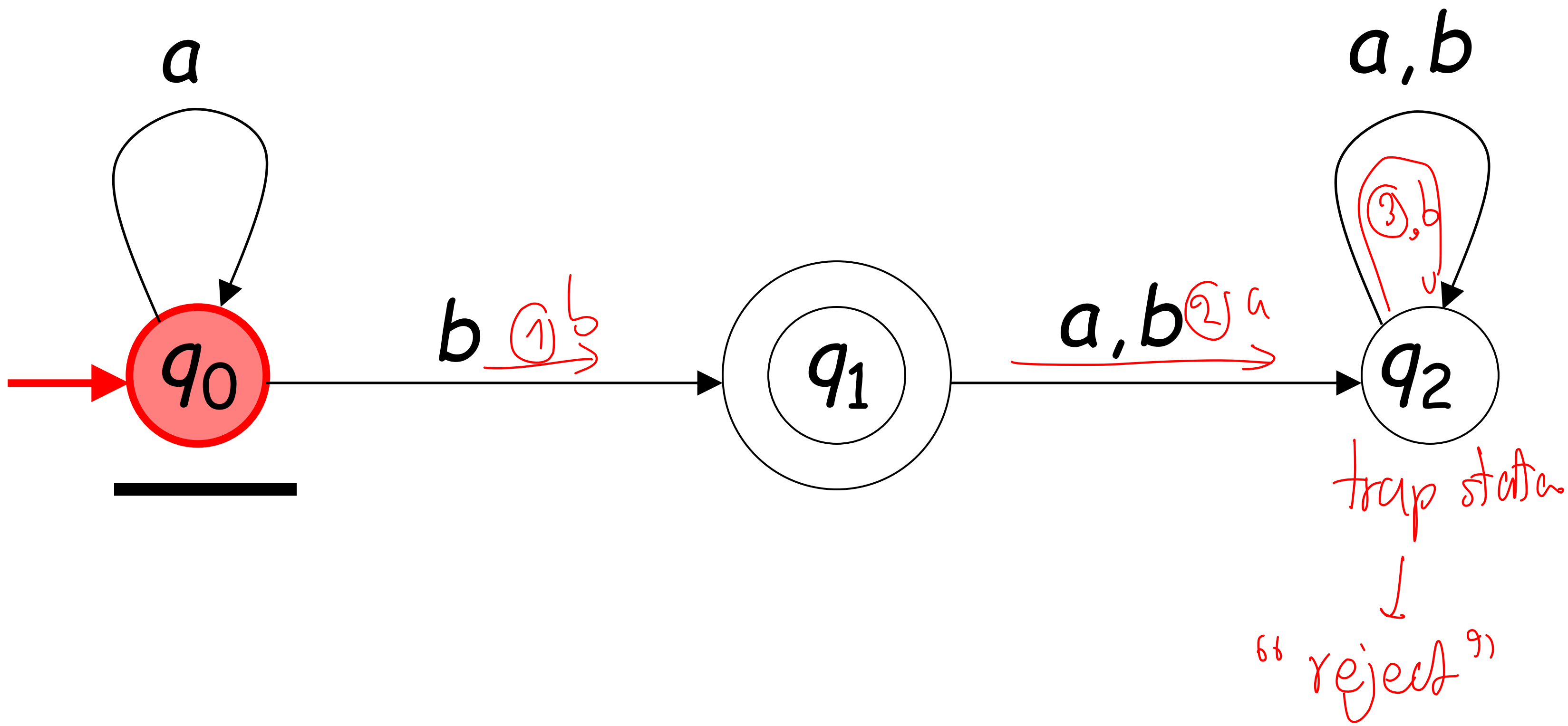
Input finished

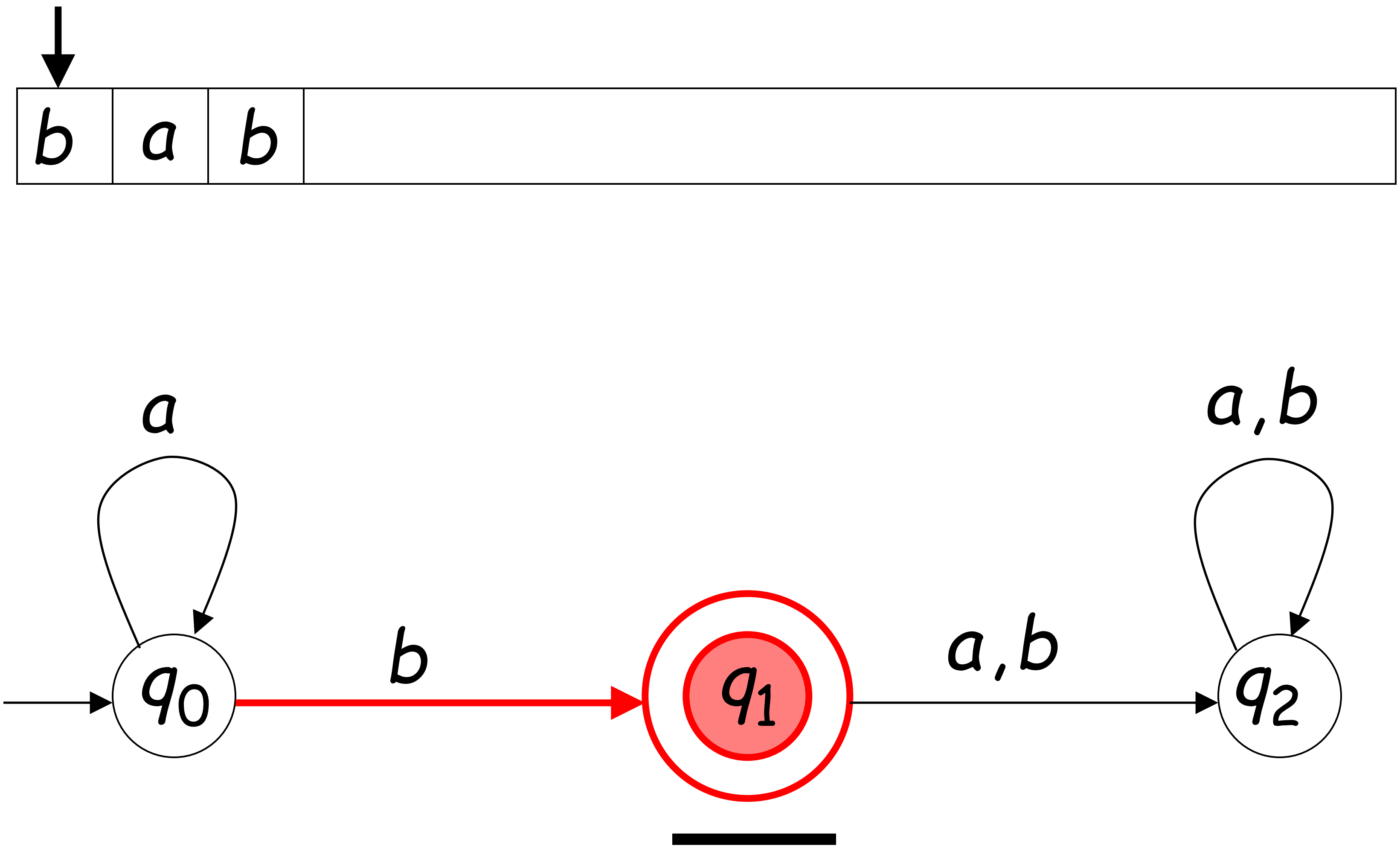


Output: "accept"

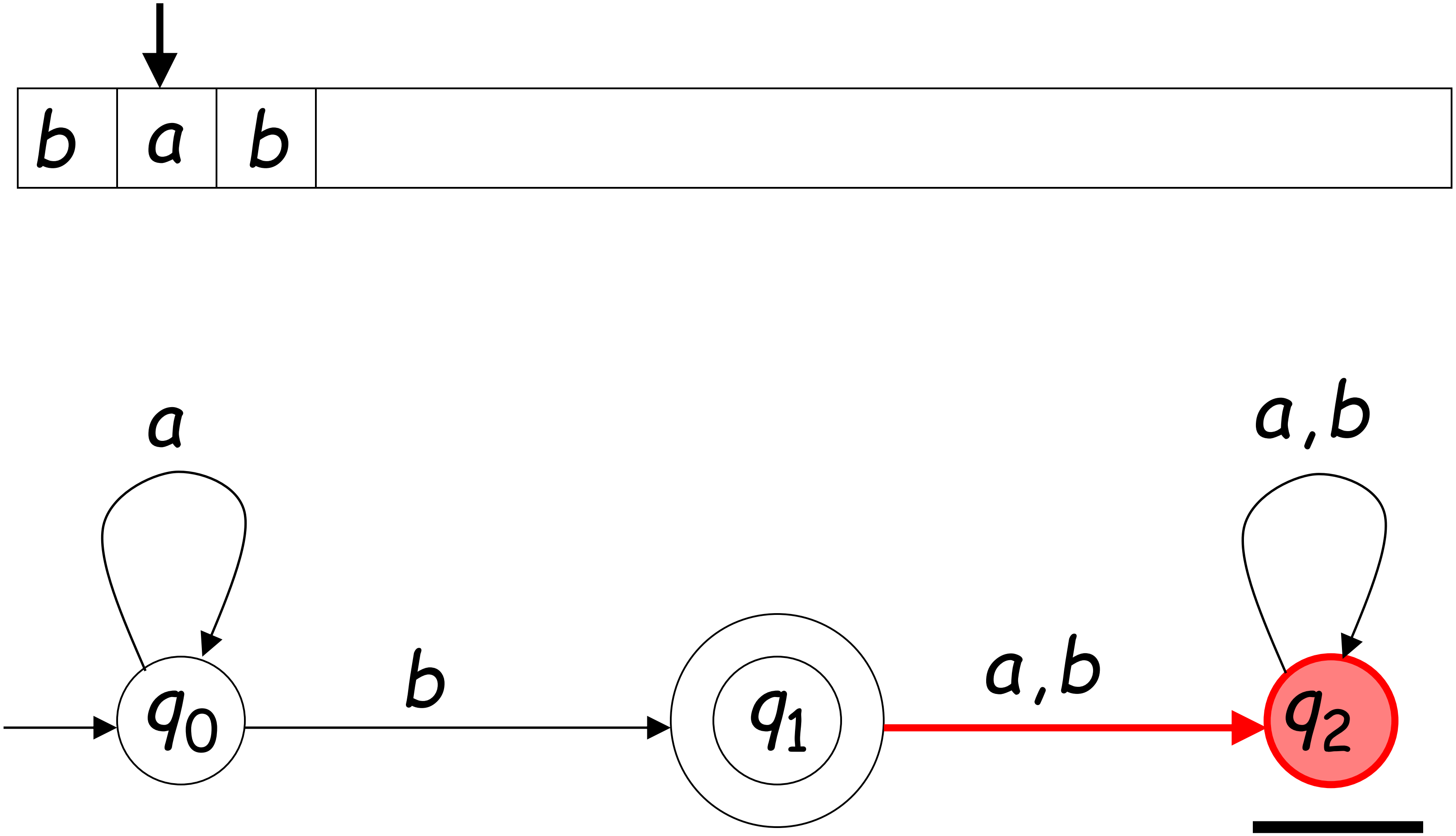
Rejection

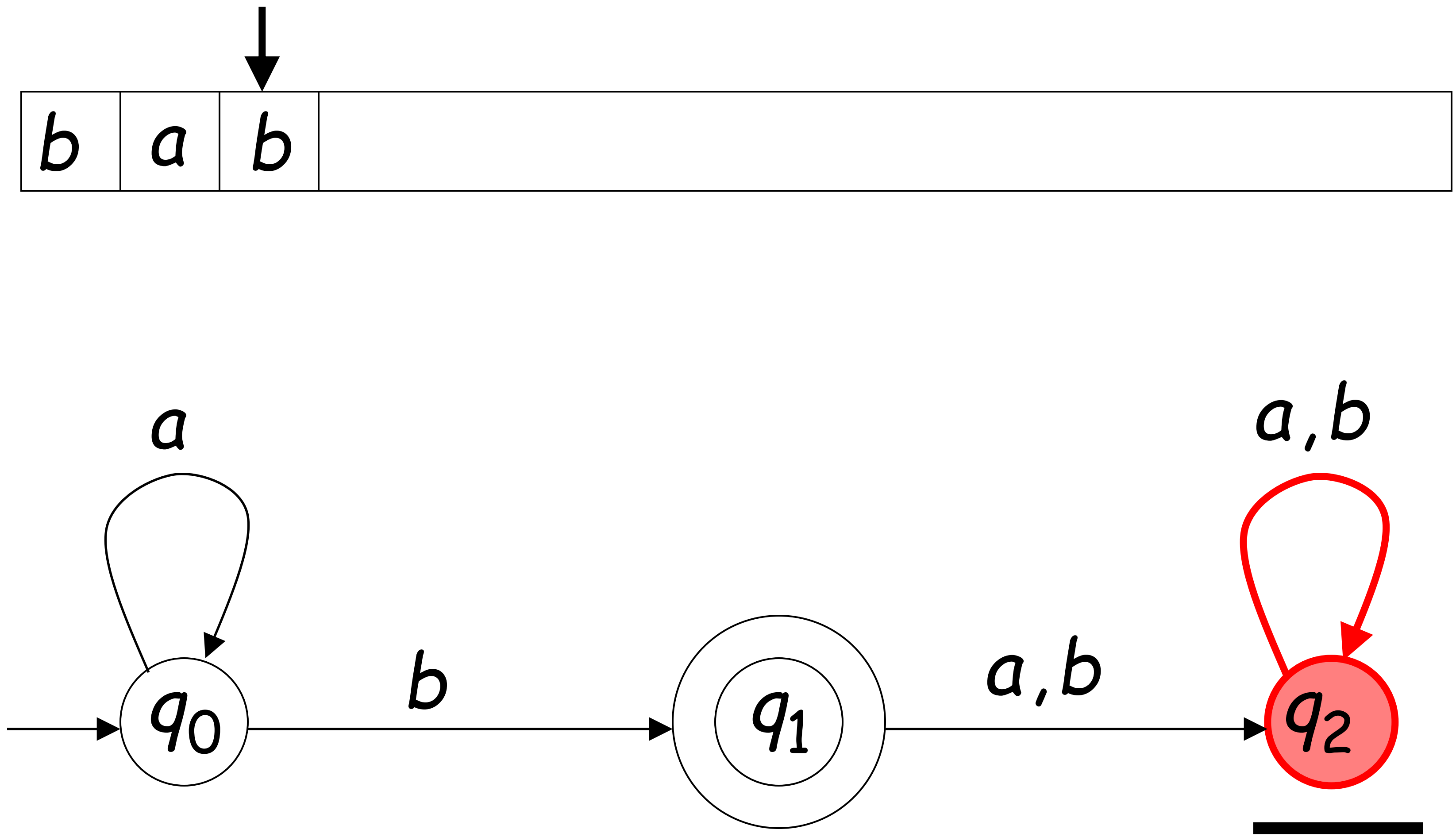




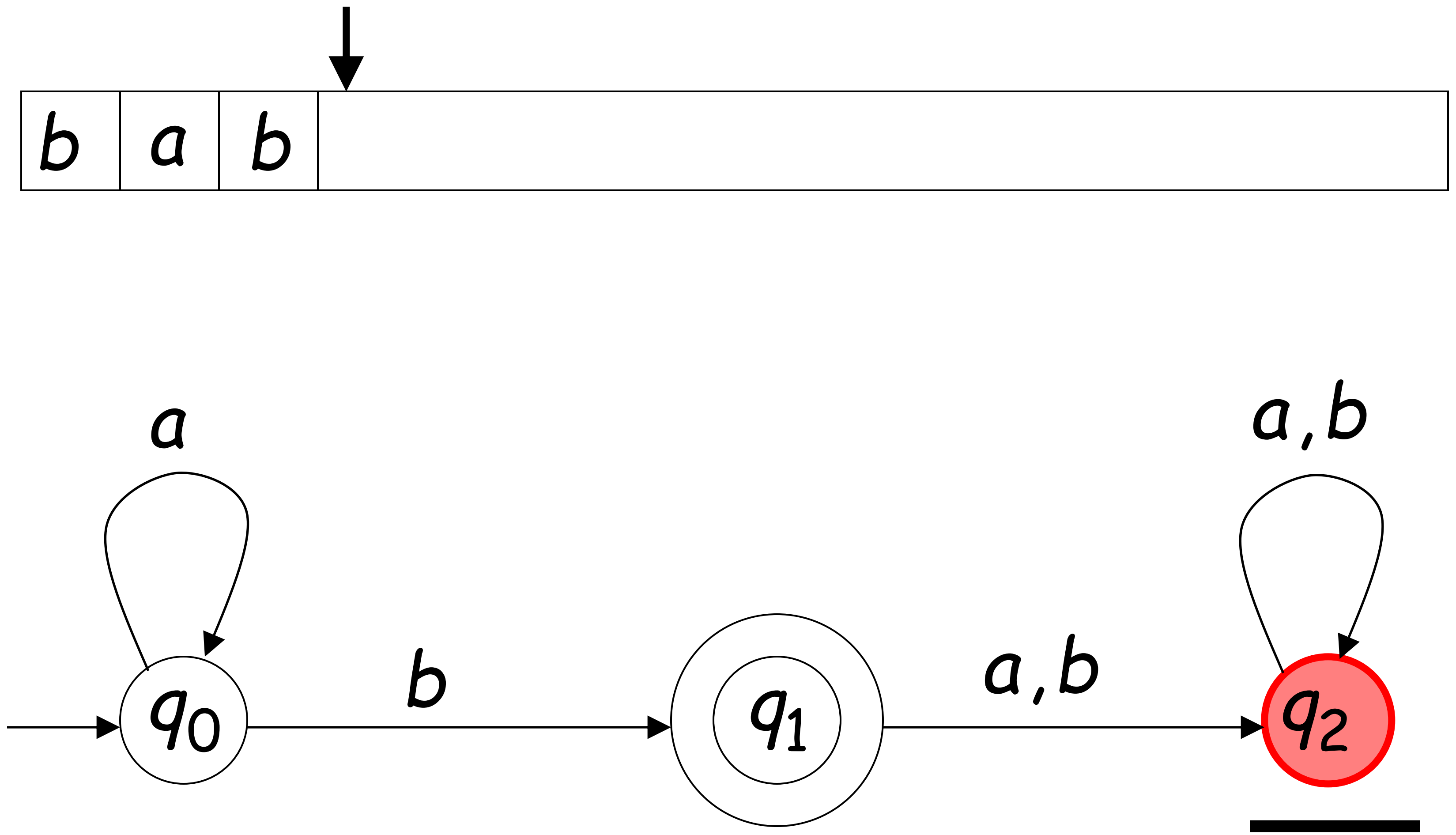


b
a
b
a
a,b
b
a,b
q_0
q_1
q_2


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


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

Input finished



Output: “reject”

Formalities

การแทน finite automata แทนเป็นสมการ

Deterministic Finite Accepter (DFA)

machine → สมัยก่อน com is called "machine"

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

สัญลักษณ์

Q : set of states เดี๋ยวอธิบายต่อได้เลย

Σ : input alphabet

δ : transition function

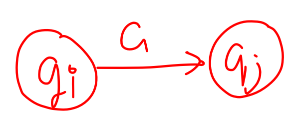
q_0 : initial state

F : set of final states

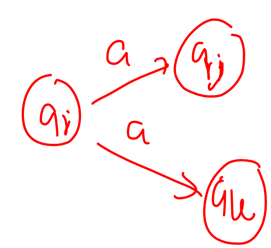
FA

compose of

DFA — Deterministic — คาดเดาได้



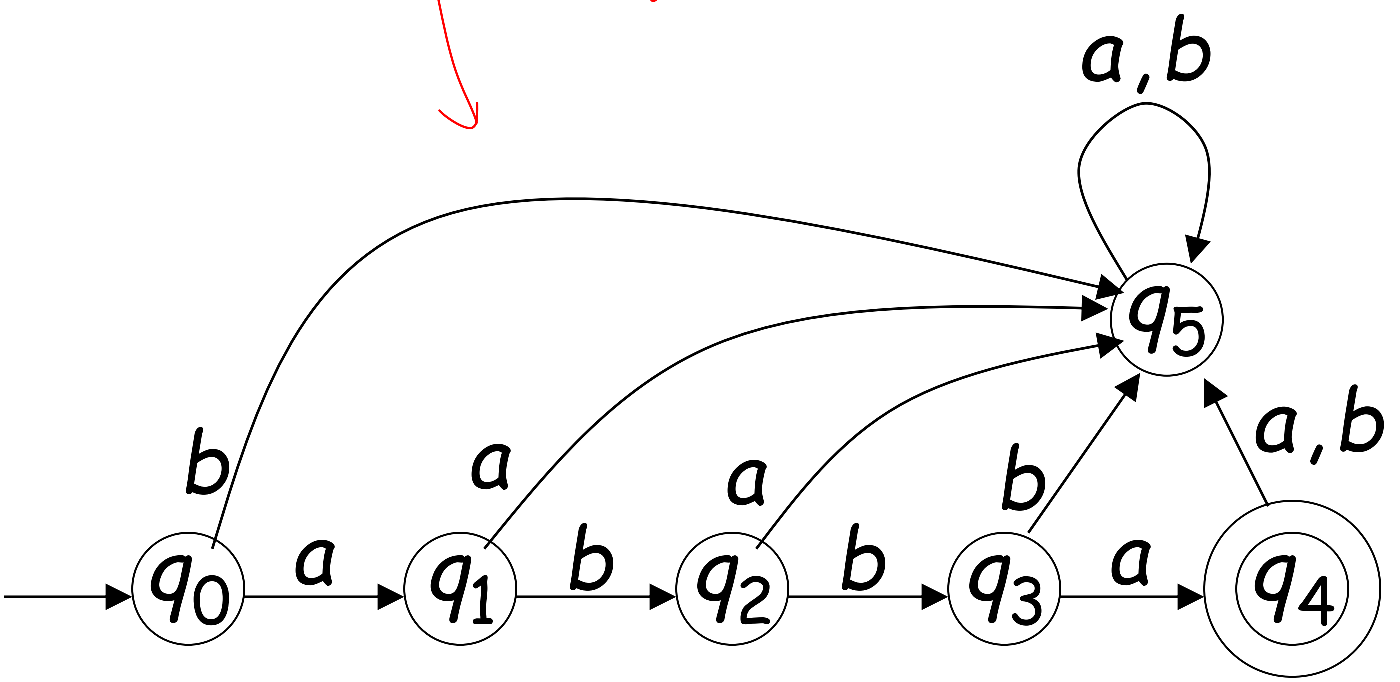
NFA — Non-Deterministic — คาดเดาไม่ได้



Input Alphabet Σ

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

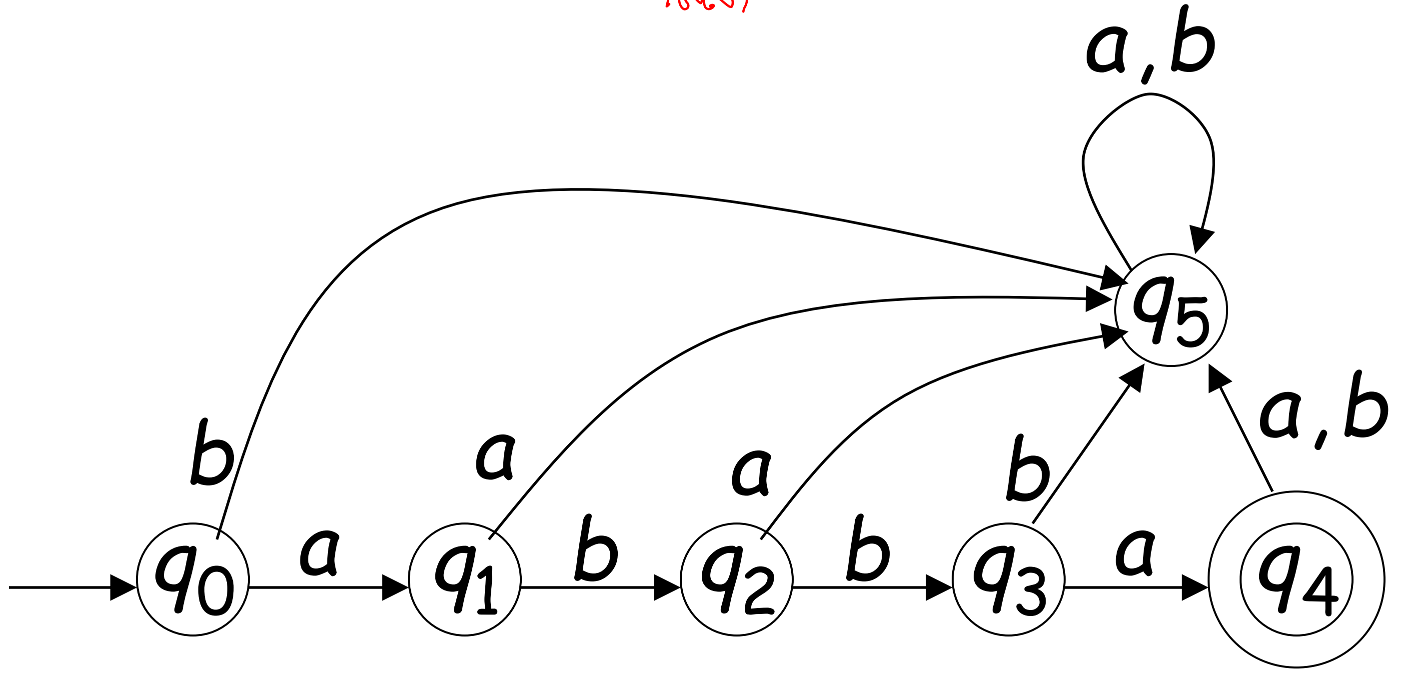
have only {a, b}



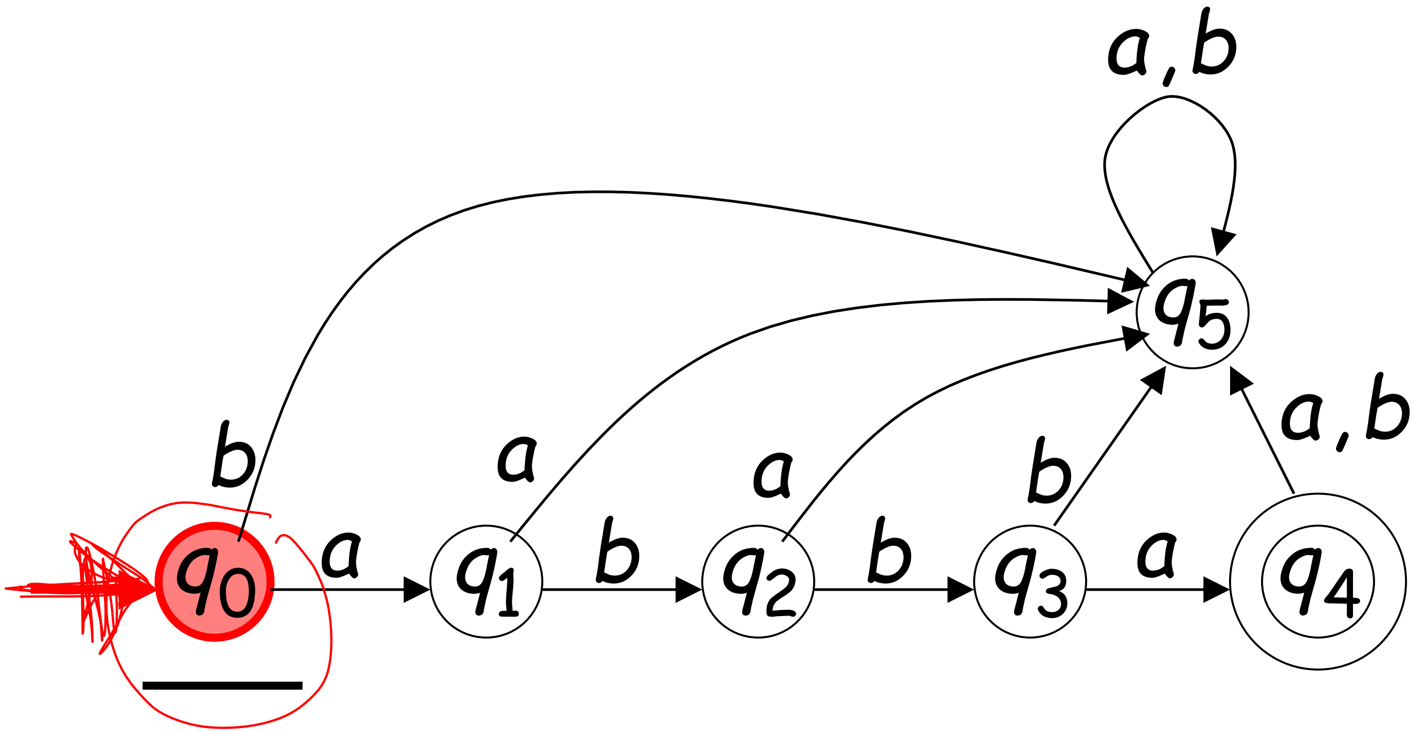
Set of States Q

internal state

$$Q = \underbrace{\{q_0, q_1, q_2, q_3, q_4, q_5\}}_{\text{ๆสัลร}}$$



Initial State q_0

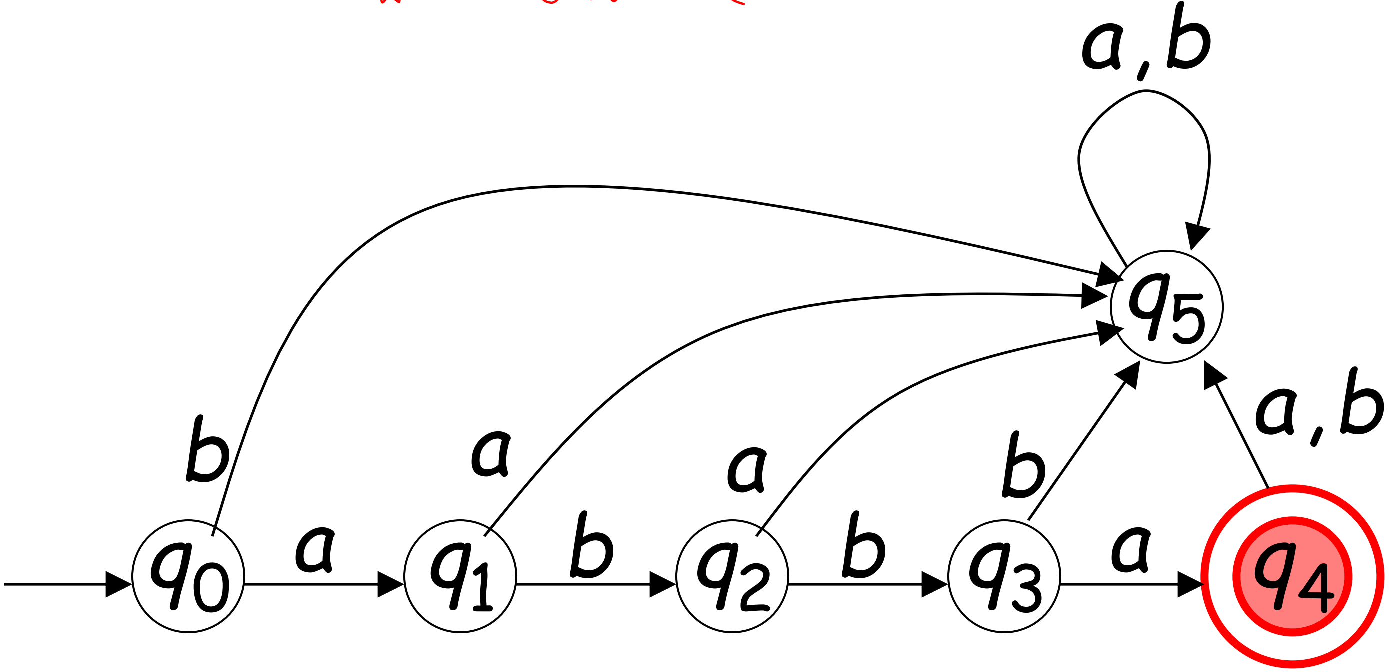


ไม่เป็น set มีแค่ตัวเดียวเท่านั้น

Set of Final States F

$$F = \{q_4\}$$

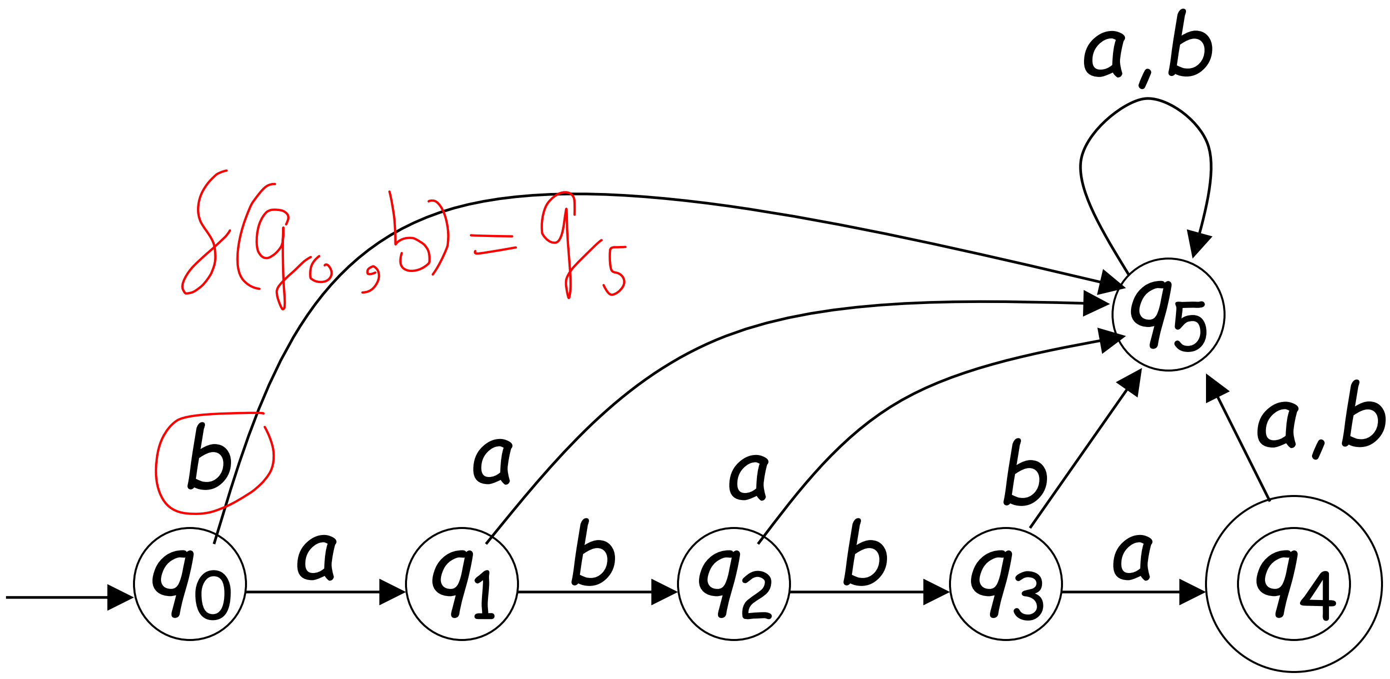
มี final อันเดียว (case นี้นะ)



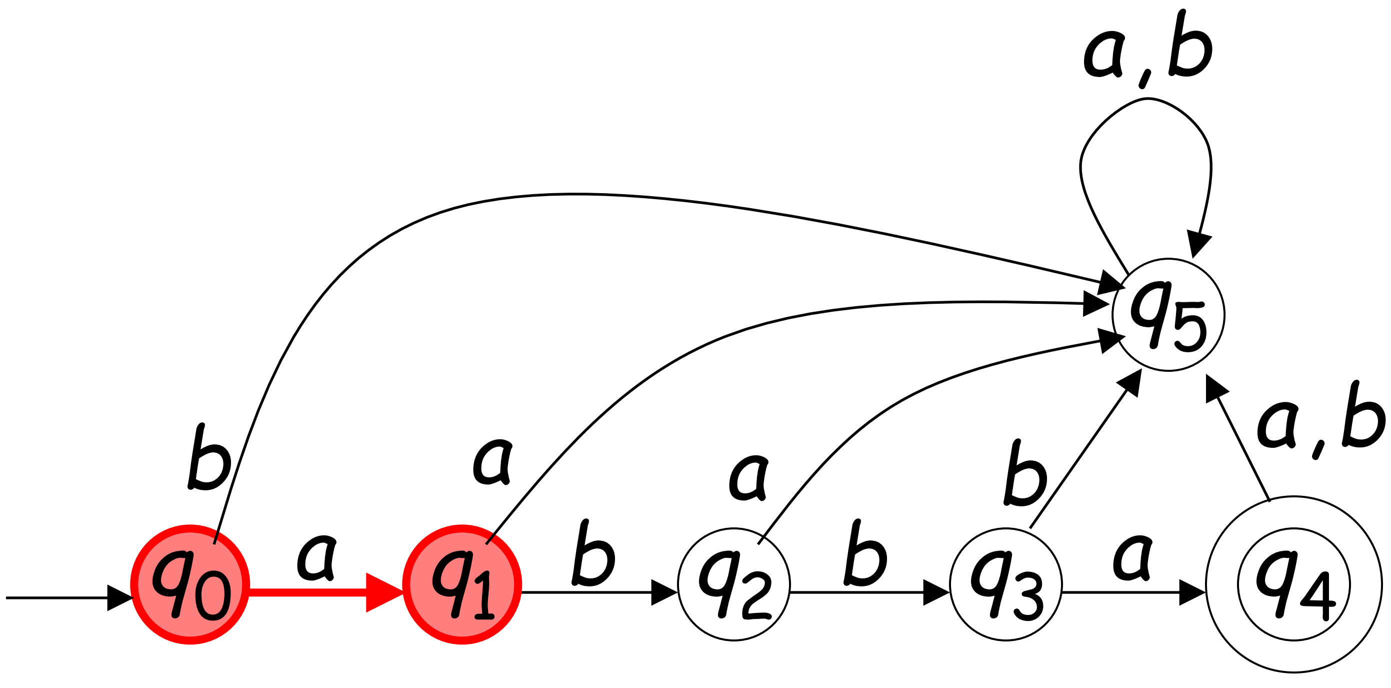
Transition Function δ

$$\delta : \underline{Q} \times \underline{\underline{\Sigma}} \rightarrow \underline{\underline{Q}}$$

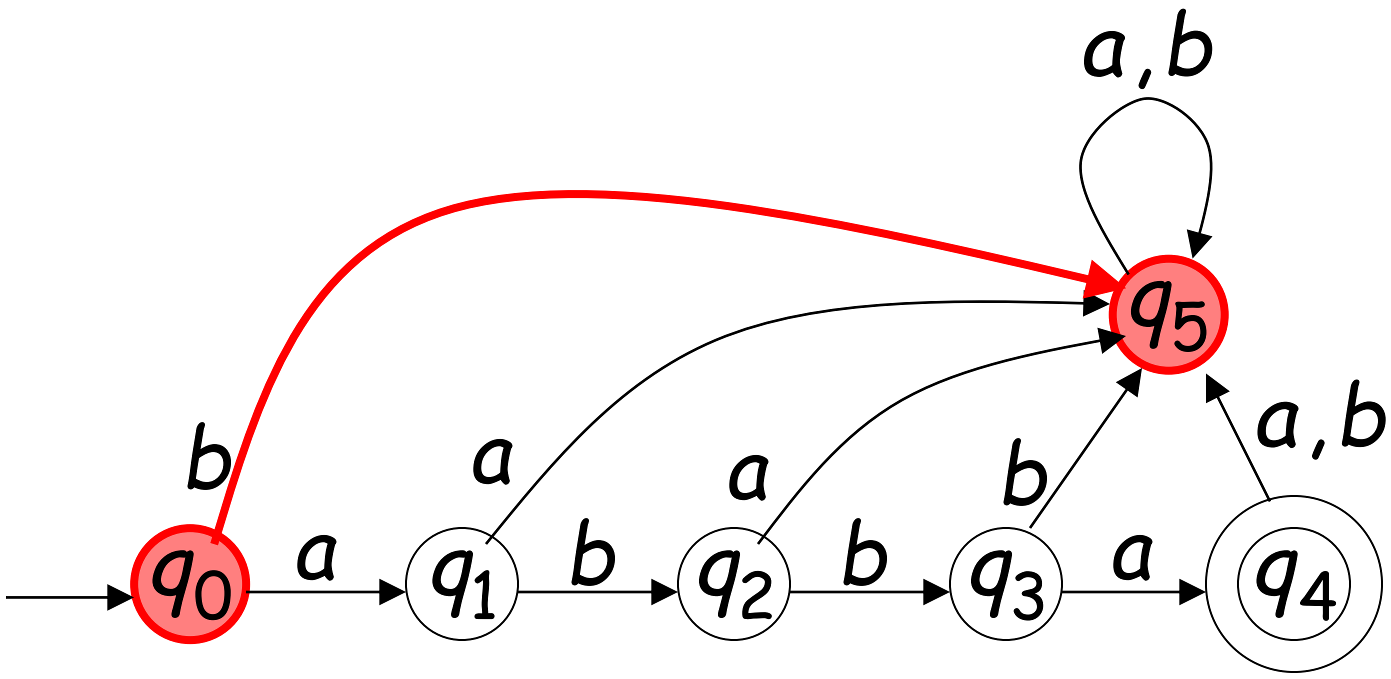
state   input ch   next state



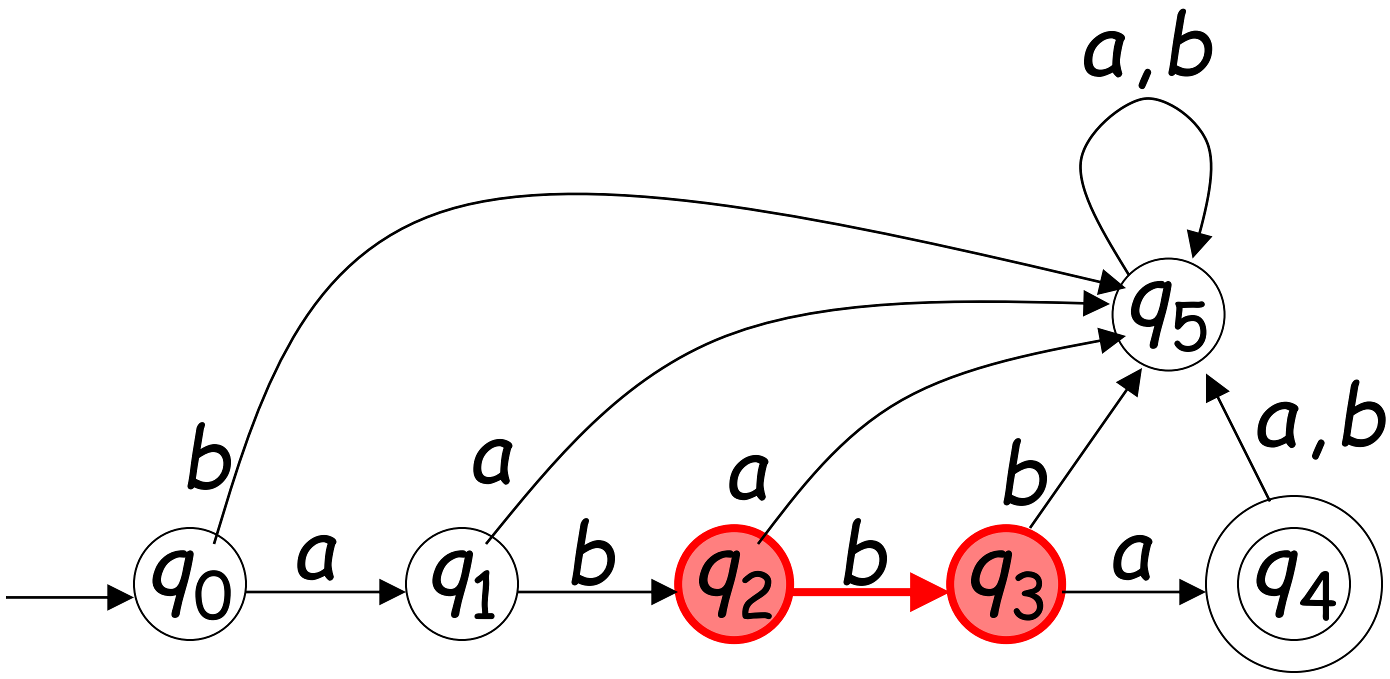
$$\delta(q_0, a) = q_1$$



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



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



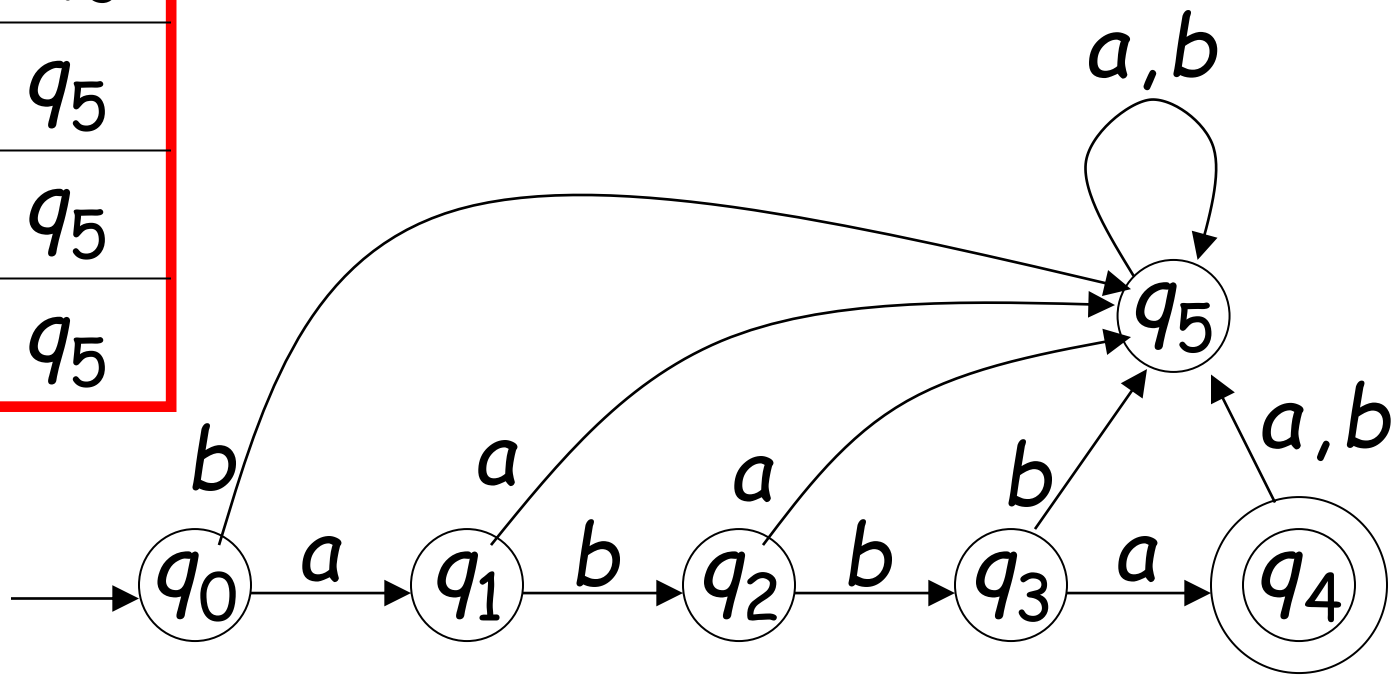
input

Transition Function δ

input

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

เขียนเป็นตารางเถอะใจยม



Extended Transition Function δ^*

*

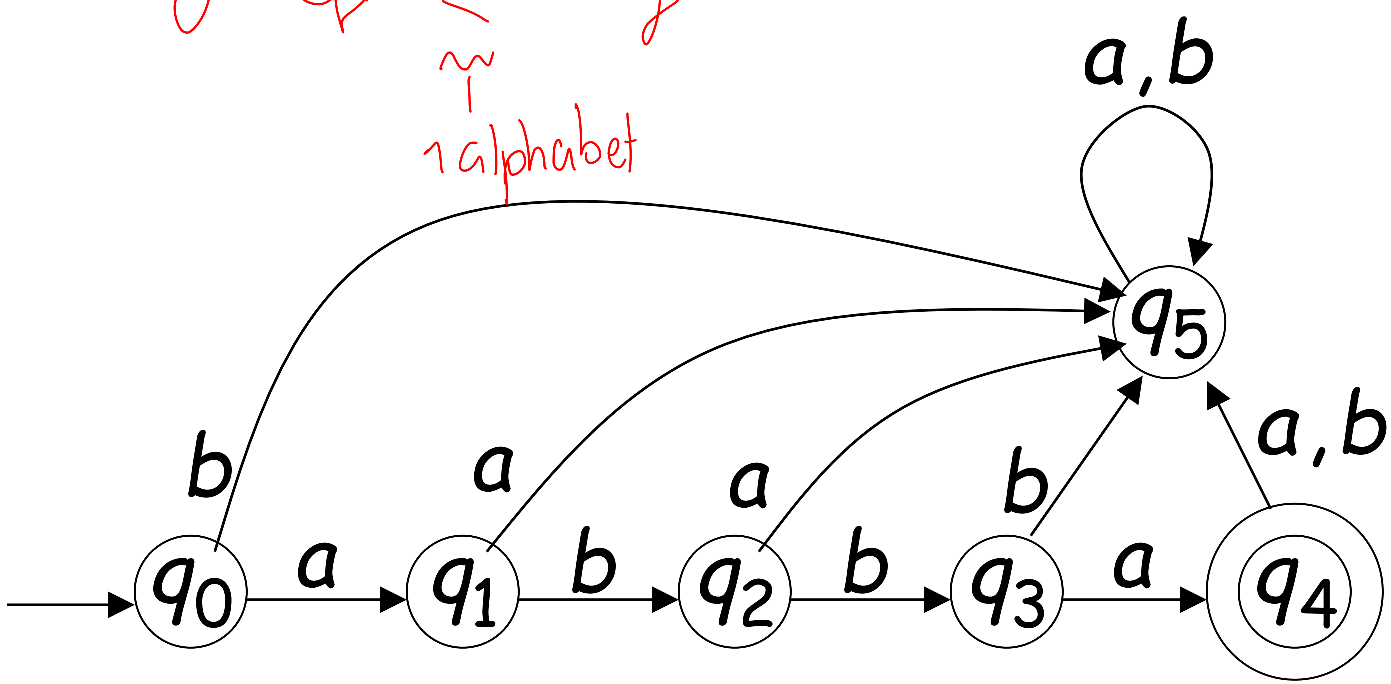
delta

$$\delta^*: Q \times \boxed{\Sigma^*} \to Q$$

string → many alphabet

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

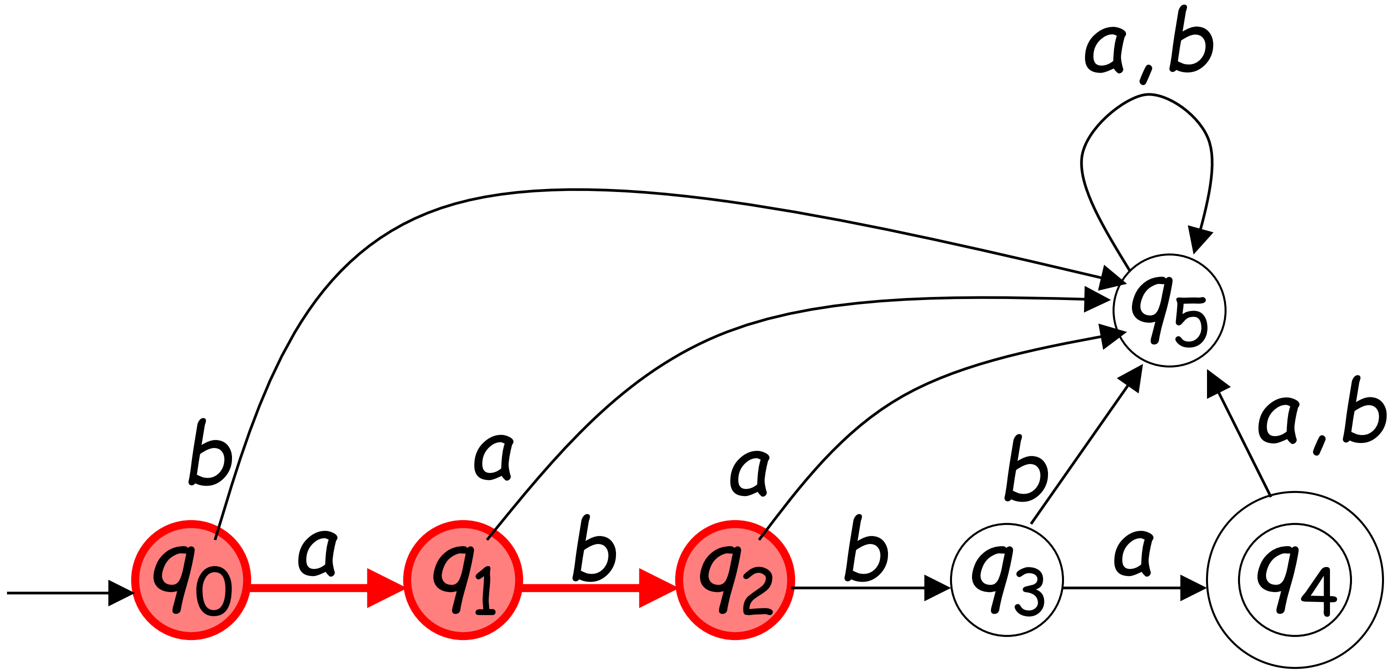
1 alphabet



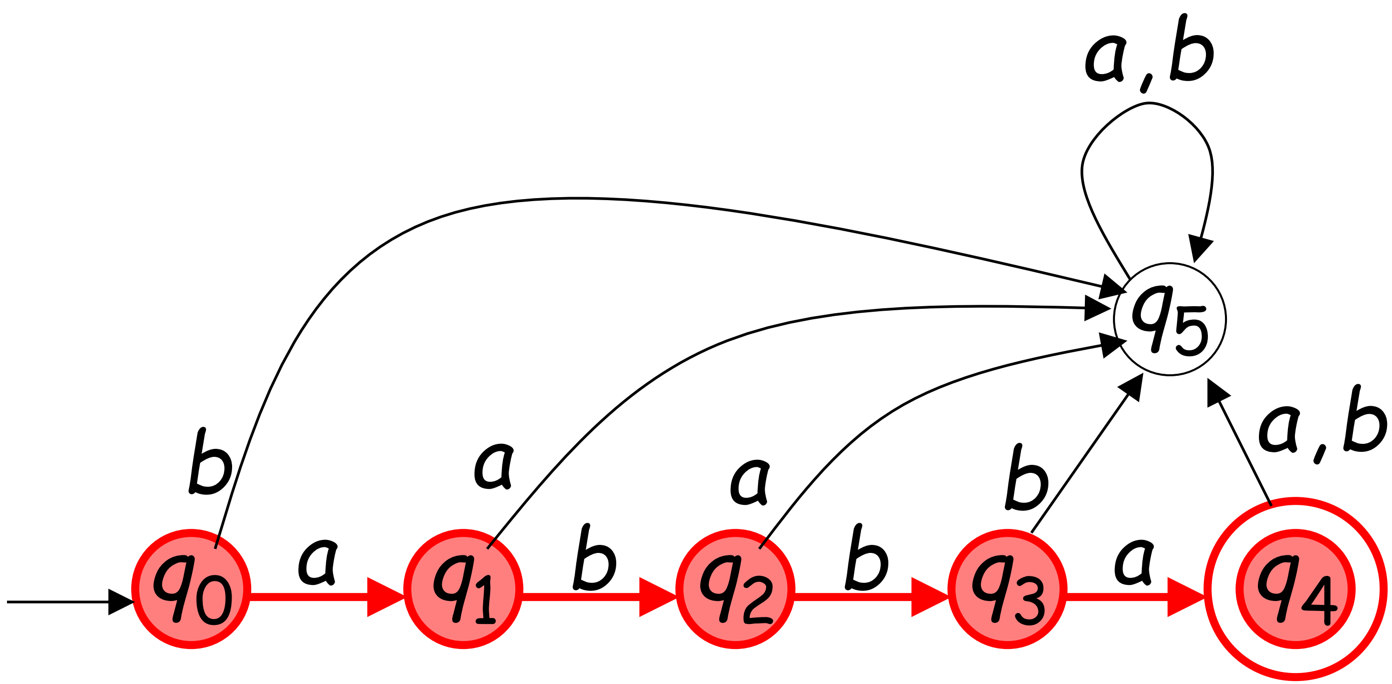
$$\delta^*(q_0, ab) = q_2$$

เหมือนรวม transition หลายๆอัน มาเหลืออันเดียว

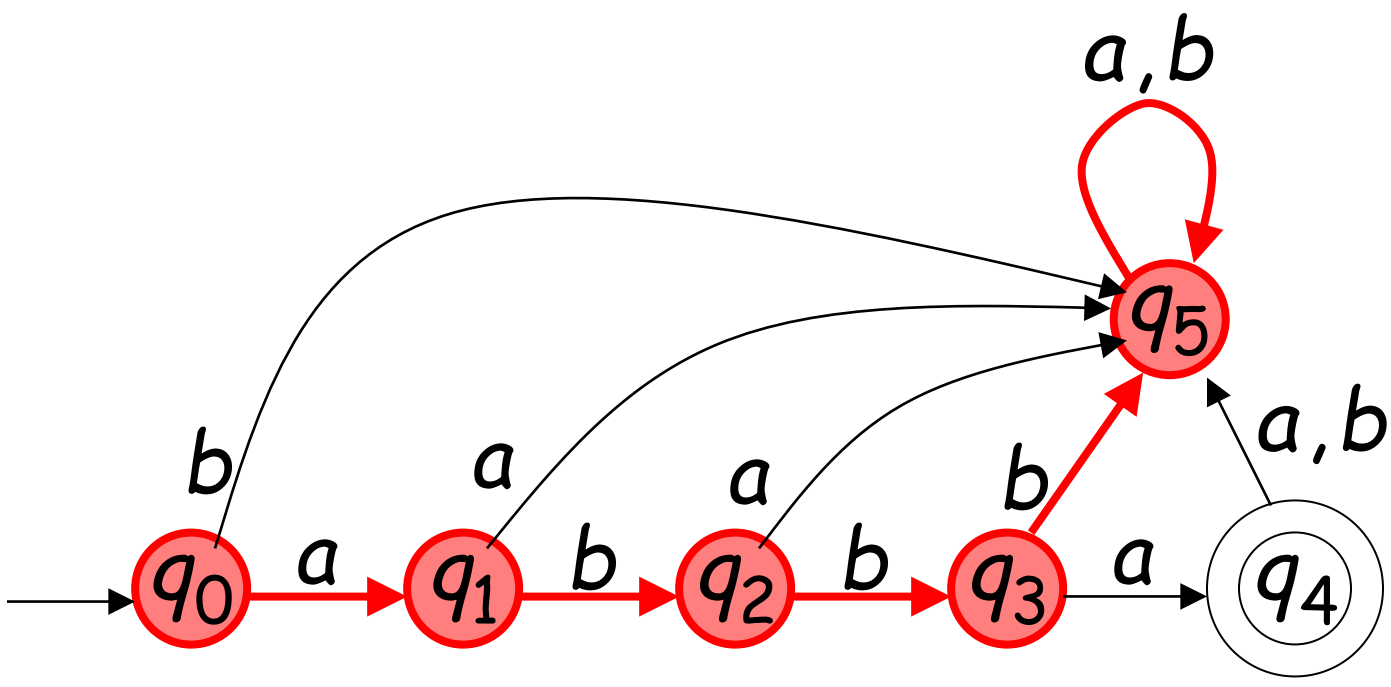
มองไม่เห็นตัวที่ 1 เลย → มองเห็นแค่ต้นทาง → ปลายทาง



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



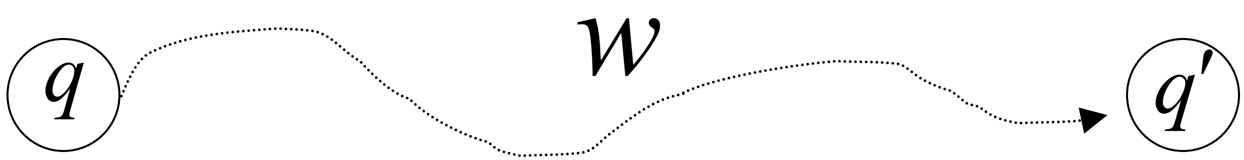
$$\delta^*(q_0, abbbaa) = q_5$$



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

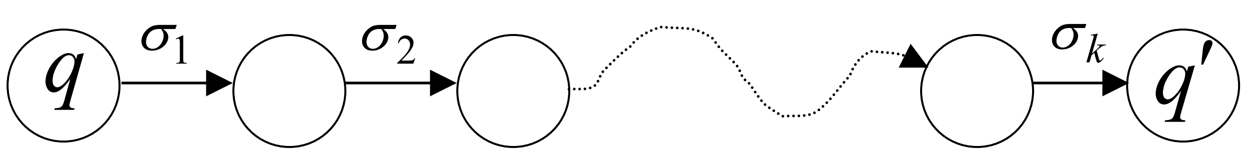
สามารถเดินได้โดยใช้ input w

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



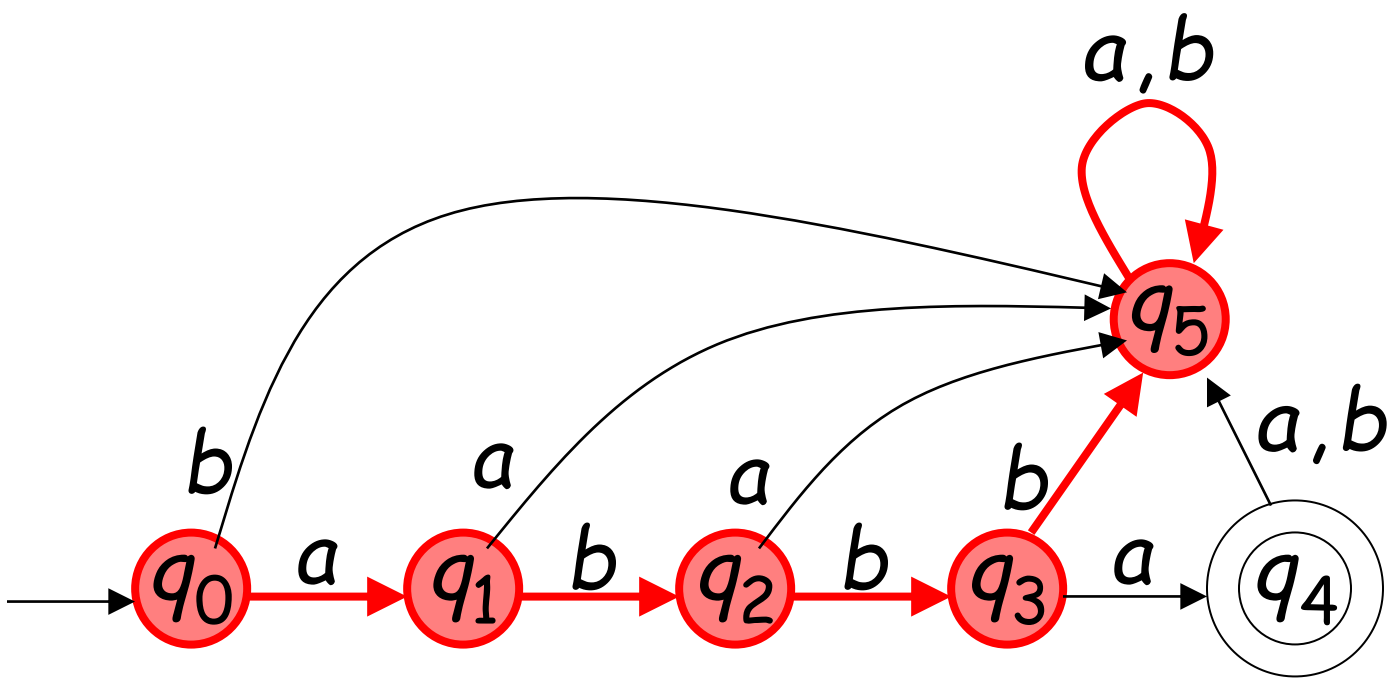
โดย w คือ string

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



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

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



Languages Accepted by DFAs

Take DFA M

ยอมรับภาษาโดย DFA

Definition:

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

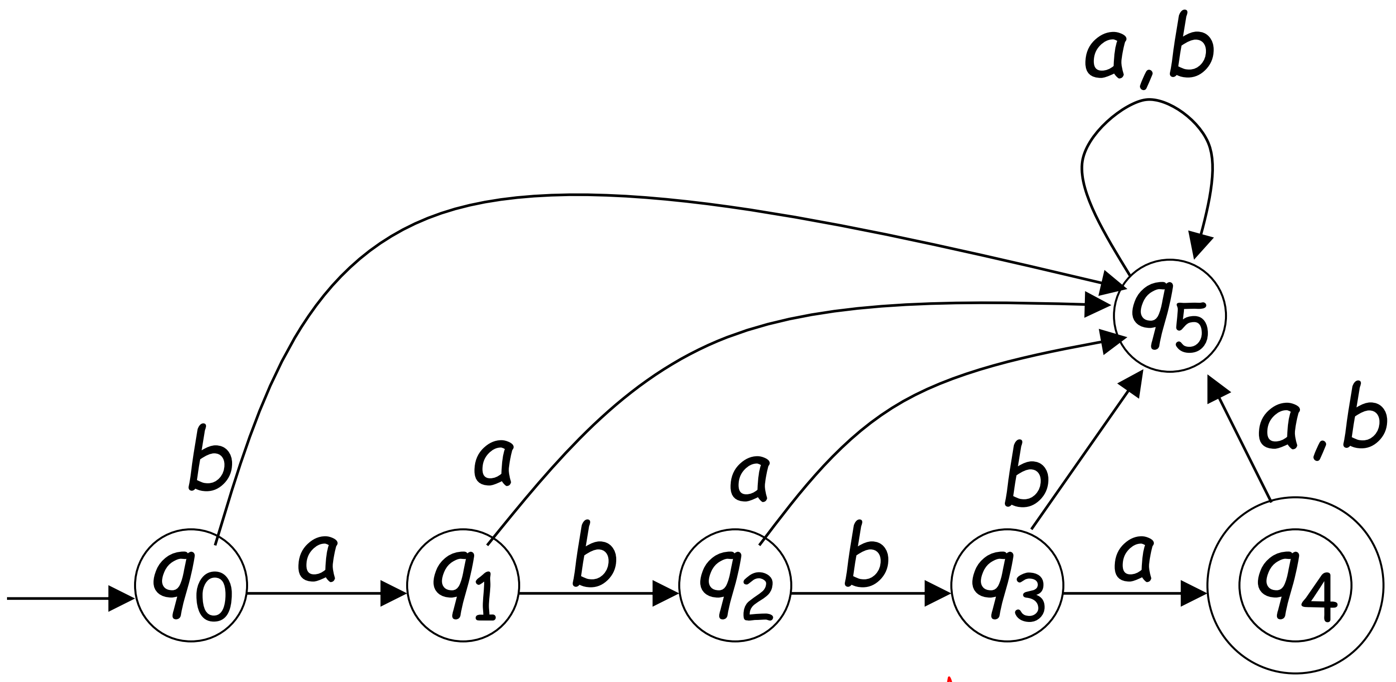
set ของ string ที่ขับเคลื่อนให้ DFA M เดินเข้าสู่ final state ได้

$L(M)$ = { strings that drive M to a final state}

Example

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

M

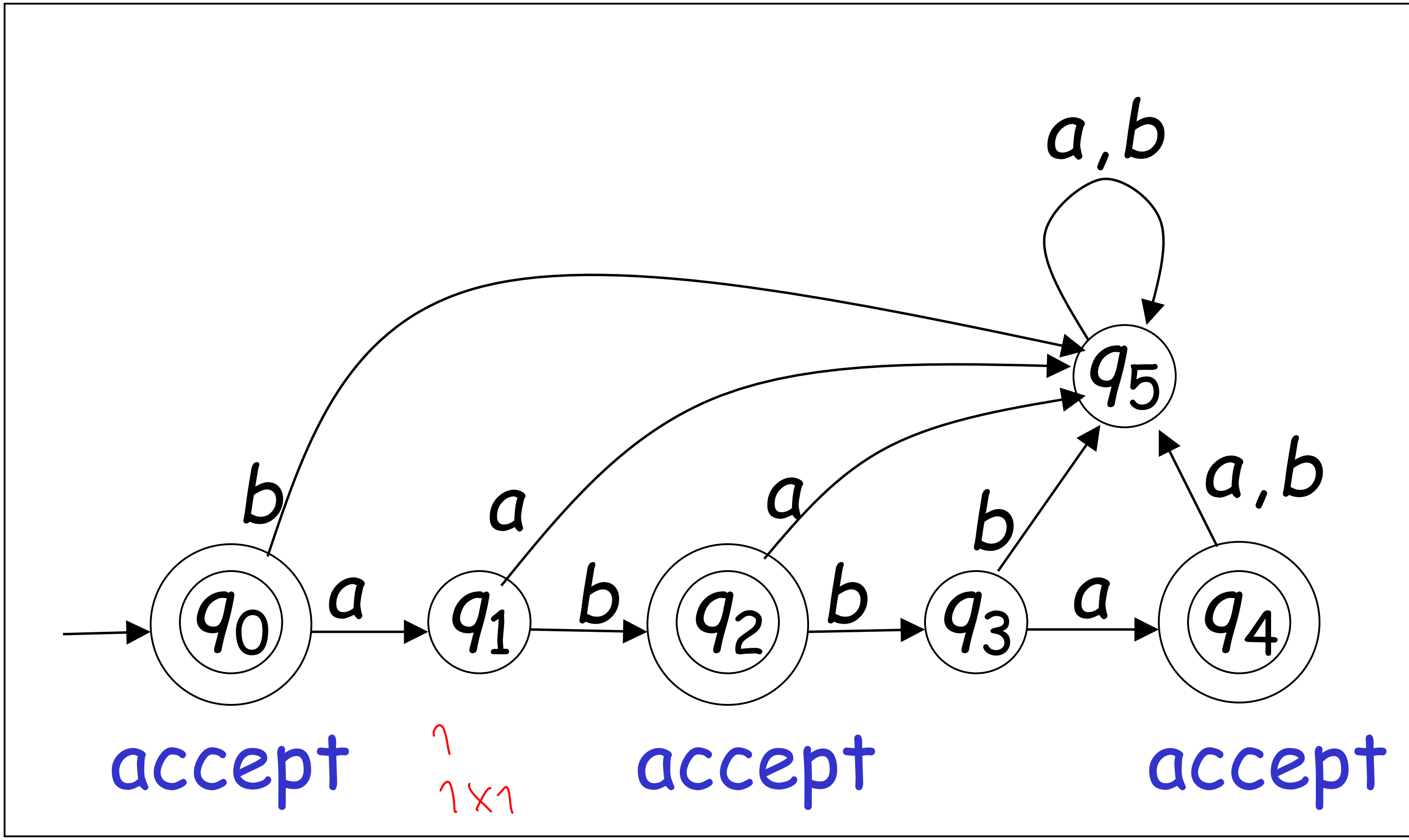


ม 1 x 1 x 1 x 1 only เท่านั้น

Another Example

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

M



1

1x1

1x1x1x1

= 3 ตัว

Formally

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

Language accepted by M :

extended transition

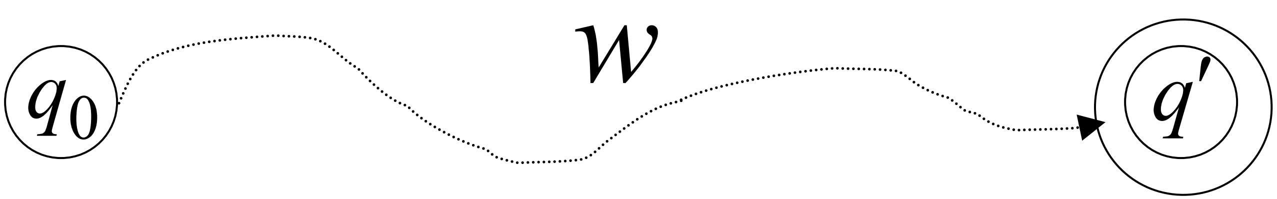
คือ universe ใหญ่ๆแบบเบื้องๆ

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

string w

โดยที่

เมื่อเอา w ป้อนเข้าไปใน q_0
จะได้ state ออกมา
อยู่ใน set ของ F



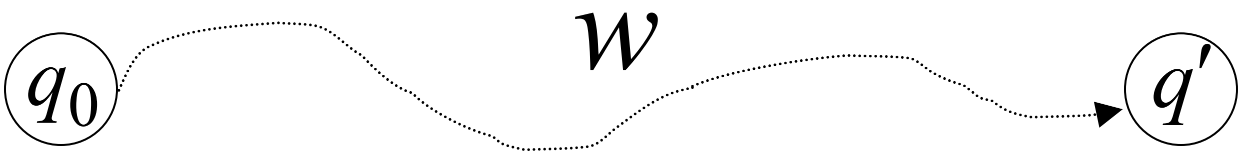
$q' \in F$

Observation

Language rejected by M :

ใส่ผงไปแล้วแตกขึ้นเอง

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



$q' \notin F$

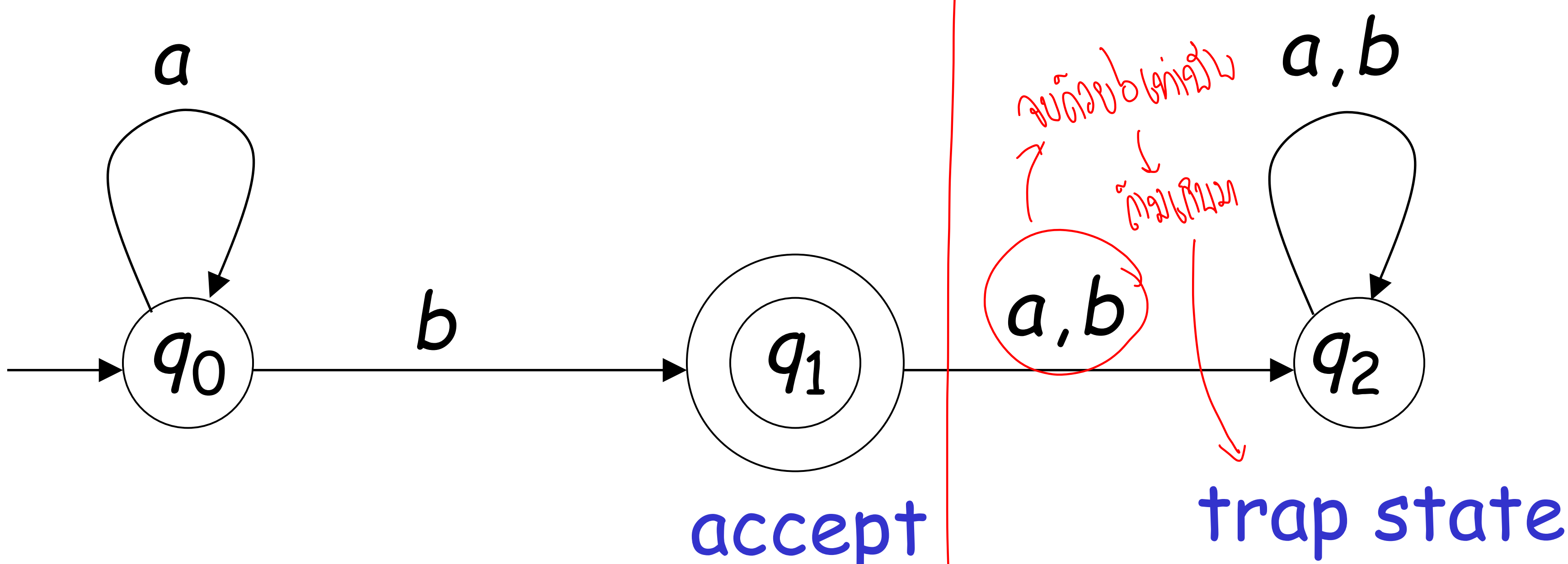
More Examples

โจทย์มี 2 การคือให้ language มาแล้วเขียน DFA
อีกการคือให้ DFA มาแล้วให้หา language

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

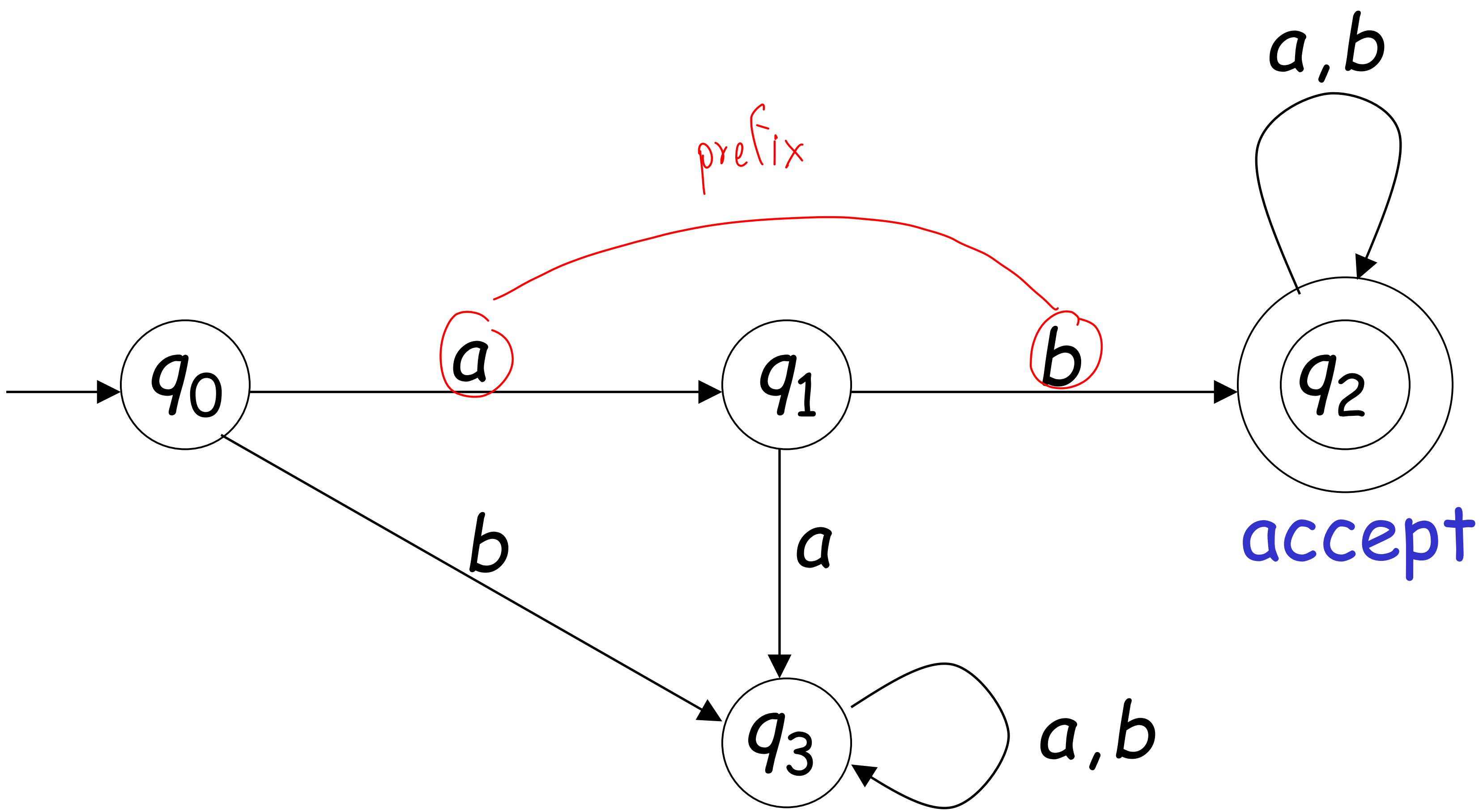
b
ab
aab
aaab
aaaab

detect error



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

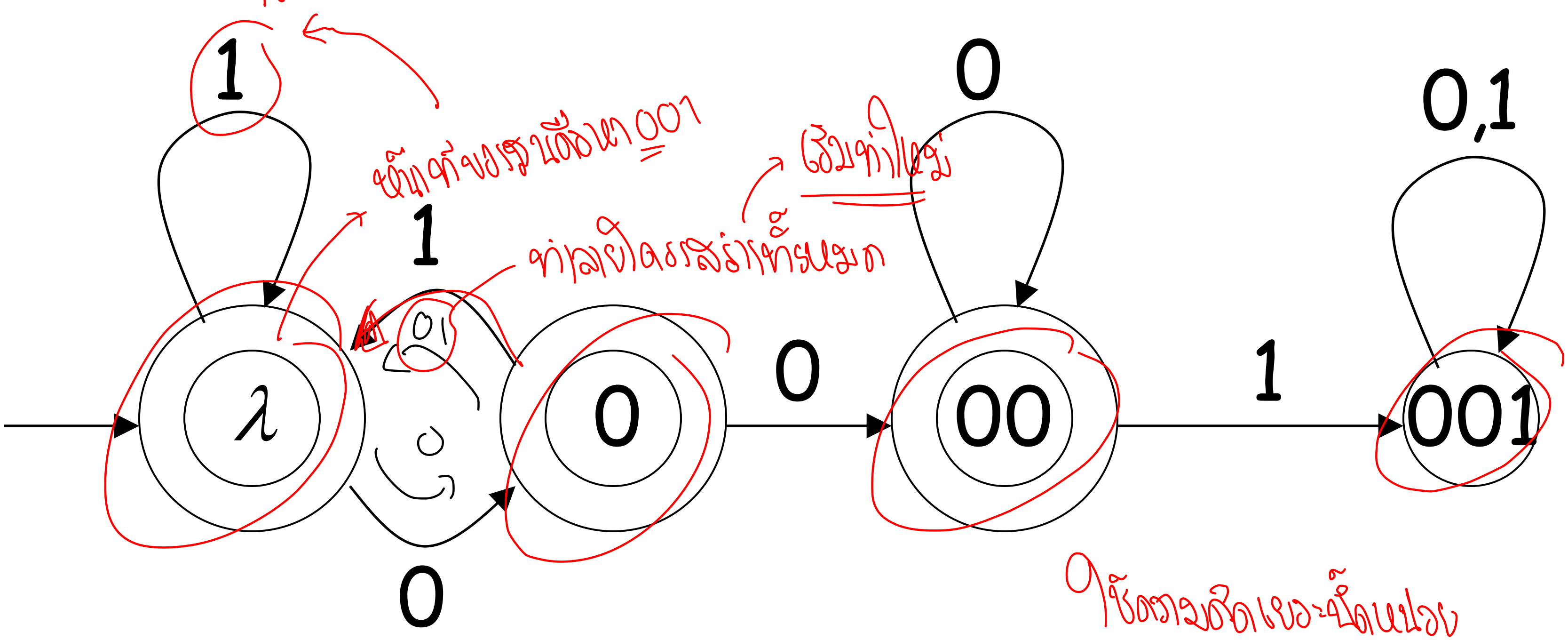
ต้องไปตาม set นี้ตนของ



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

ทำ trap state
↓
ที่เหลือทำ final

เขียน ○ เพื่อ detect แต่ state 001 เท่านั้นที่ไม่เป็น final



Regular Languages

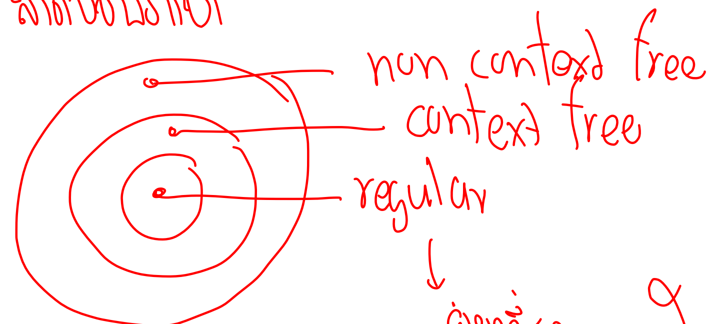
ชื่อของภาษา

A language L is regular if there is a DFA M such that $L = L(M)$

ลำดับชั้นภาษา

ภาษา L จะเป็นภาษา regular ถ้าเรา

สามารถสร้าง DFA ขึ้นมารองรับมันได้



ง่ายที่สุด ภาษาไม่ต้องการหน่วยความจำ

All regular languages form a language family

Examples of regular languages:

$\{abba\}$ $\{\lambda, ab, abba\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab }

{ all strings without substring 001 }

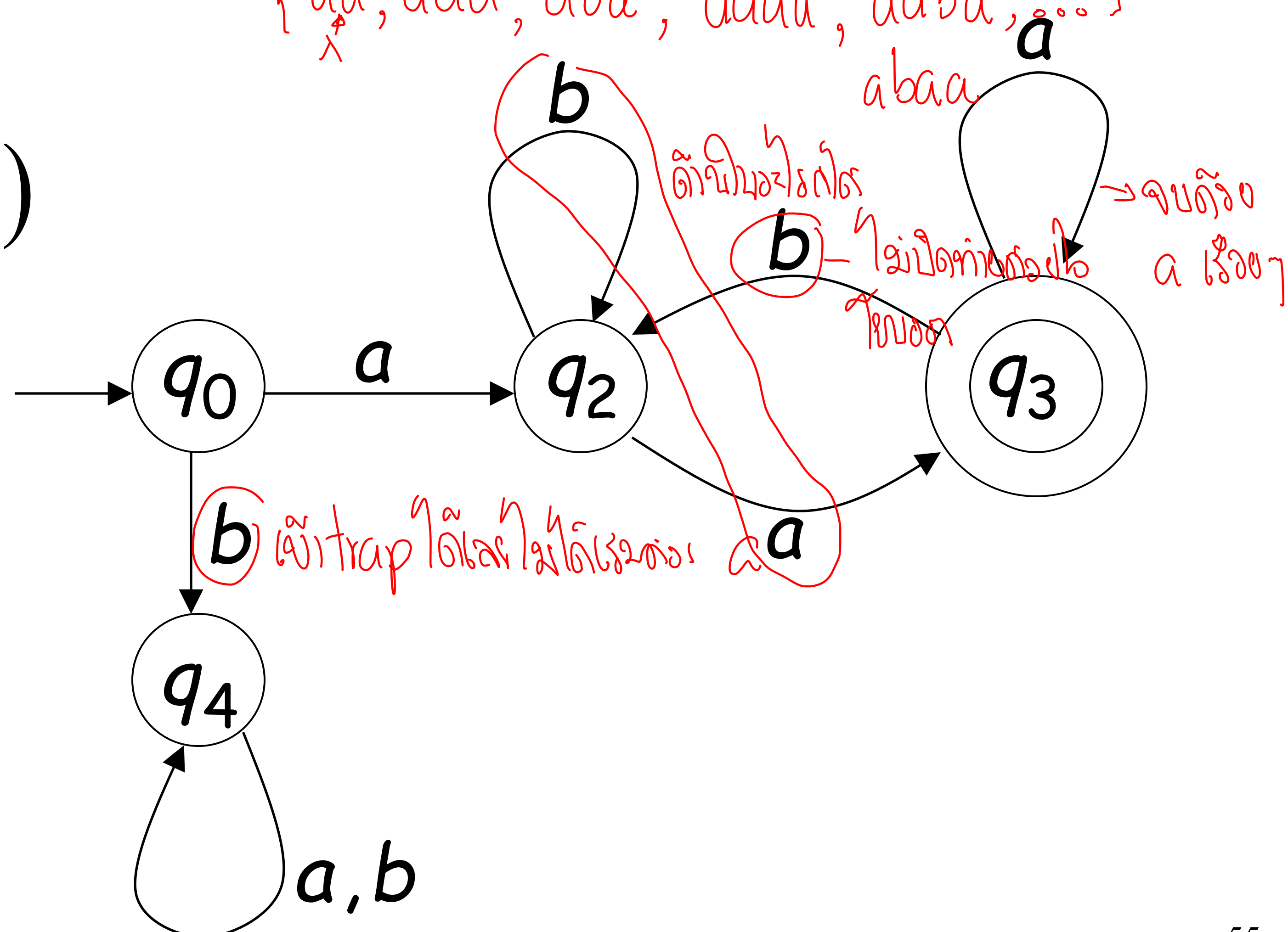
There exist automata that accept these Languages (see previous slides).

Another Example

The language $L = \{awa : w \in \{a,b\}^*\}$ is regular:

$\{aa, aaa, aba, aaaa, aaba, \ldots\}$

$$L = L(M)$$



There exist languages which are not Regular:

λ , ab , $aabb$, $aaabbb$

Example: $L = \{a^n b^n : n \geq 0\}$

ไม่ใช่ regular → เนื่องจาก DFA มี limit memory

↳ จำไม่ได้ว่ากี่ตัวแล้ว

There is no DFA that accepts such a language

(we will prove this later in the class)