BLM2502 Theory of Computation

Spring 2017

BLM2502 Theory of Computation

- **Course Outline**
- Week — Content
- 1 — Introduction to Course
- 2 — Computability Theory, Complexity Theory, Automata Theory, Set Theory, Relations, Proofs, Pigeonhole Principle
- **3 — Regular Expressions**
- 4 — Finite Automata
- 5 — Deterministic and Nondeterministic Finite Automata
- 6 — Epsilon Transition, Equivalence of Automata
- 7 — Pumping Theorem
- 8 — April 10 - 14 week is the first midterm week
- 9 — Context Free Grammars
- 10 — Parse Tree, Ambiguity,
- 11 — Pumping Theorem
- 12 — Turing Machines, Recognition and Computation, Church-Turing Hypothesis
- 13 — Turing Machines, Recognition and Computation, Church-Turing Hypothesis
- 14 — May 22 – 27 week is the second midterm week
- 15 — Review
- 16 — Final Exam date will be announced

BLM2502 Theory of Computation

Regular Expressions

- Keywords / Definitions:
  - *Alphabet*: A finite nonempty set of symbols. The members of the alphabet are the *symbols* of the alphabet. We generally use capital Greek letters Σ and Γ to designate alphabets. The following are a few examples of alphabets.

    $\Sigma_1 = \{0,1\}$

    $\Sigma_2 = \{a,b,c,d,e,f,g.h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$

    $\Gamma = \{0, 1, x, y, z\}$
  - A string over an alphabet is a finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas. If $\Sigma_1 = \{0,1\}$, then 01101 is a string over Σ_1. If $\Sigma_2 = \{a, b, c, \ldots, z\}$, then abracadabra is a string over Σ_2.

BLM2502 Theory of Computation

Regular Expressions

- Keywords / Definitions:
  - If w is a string over Σ, the *length* of w, written Iwl, is the number of symbols that it contains. The string of length zero is called the *empty string* and is written as ε (or λ). The empty string plays the role similiar to 0 in a number system.
  - If w has length n, we can write

    $w = w_1 w_2 \ldots w_n$ where each $w_i \in \Sigma$.

    The reverse of w, written as w^R, is the string obtained by writing w in the opposite order (i.e., $w_n w_{n-i} \ldots w_1$).

    String z is a *substring* of w if z appears consecutively within w. For example, *cad* is a substring of *abracadabra*

BLM2502 Theory of Computation

Regular Expressions

- Keywords / Definitions:
  - If we have string x of length m and string y of length n, the *concatenation* of x and y, written xy, is the string obtained by appending y to the end of x, as in $x_1 \ldots x_m y_1 \ldots y_n$. To concatenate a string with itself many times we use the superscript notation:

    $x^3 = xxx$,

    $x^n = xx\ldots.x$ (n times)

    To indicate all possible recurrencies, a special superscript (in fact a special operator) is used:

    * (kleene star)
  - *Language* : A set of strings (finite or infinite?) over an alphabet. Languages are used to describe computation problems.

BLM2502 Theory of Computation

Regular Expressions

- Keywords / Definitions:
  - *Regular Languages*: A subset of all languages. There is no way to define regular set. However, it is possible to say whether a set is regular or not.
  - Best way is using *Finite Automata*. If some finite automata recognizes the language, then the language is regular.
  - Regular (set) operations are used to build up regular sets:
  - Union, concatenation, and kleene star operations are as follows.
    - *Union*: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
    - *Concatenation*: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
    - *Star*: $A^* = \{x_1 x_2 \ldots x_k \mid k > 0 \text{ and each } x_i \in A\}$.

BLM2502 Theory of Computation

Regular Expressions

- Keywords / Definitions:
  - Class of regular languages is closed under **union**
  - Class of regular languages is closed under **intersection**
  - Class of regular languages is closed under **complement**
  - Class of regular languages is closed under **concenation**
  - Class of regular languages is closed under (kleene) **star**

BLM2502 Theory of Computation

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

Strings: a, ab, abba, aaabbbaabab, …
u = ab,
v = bbbaaa,
w = abba …

BLM2502 Theory of Computation

Decimal numbers

Alphabet: $\Sigma = \{0, 1, 2, 3, 4, \ldots, 9\}$

Strings: 102345, 567463386, …

Binary numbers

Alphabet: $\Sigma = \{0, 1\}$

Strings: 0, 1, 0000, 01001011, …

…

BLM2502 Theory of Computation

Unary numbers

Alphabet: $\Sigma = \{1\}$

Strings: 1, 11, 11111, …

Decimal equivalent: 1, 2, 5

BLM2502 Theory of Computation

- Examples on String Operations

  $w = a_1a_2\ldots a_n$, $v = b_1b_2\ldots b_m$ $a,b \in \{x, y\}$

  eg, w = xyyxy, v = xxxyyy

  Concenation:

  $wv = a_1a_2\ldots a_nb_1b_2\ldots b_n$

  eg, xyyxyxxxyyy

  Reverse:

  $w^R = a_na_{n-1}\ldots a_1\, c$

  eg, yxyyx

BLM2502 Theory of Computation

- Examples on String Operations

  $w = a_1a_2\ldots a_n$, $v = b_1b_2\ldots b_m$ $a,b \in \{x, y\}$

  Length:

  $|w| = n$, $|v| = m$

  eg, $|yxyyx| = 5$; $|xxxyyy| = 6$

  $|wv| = |w| + |v|$

  eg, $|yxyyxxxxyyy| = 11$ (recall example above)

  Empty String:

  $|\varepsilon| = 0$

BLM2502 Theory of Computation

» Examples on String Operations

$w = a_1a_2\ldots a_n$, $v = b_1b_2\ldots b_m$ $a,b \in \{x, y\}$

Substring:

$a_1, a_2, \ldots, a_n$ (each symbol of the string are its substrings)

$a_1, a_1a_2, a_1a_2a_3\ldots$ (these are prefix substrings)

$a_2, a_2a_3, a_2a_3a_4, \ldots$

$a_n, a_{n-1}a_n, a_{n-2}a_{n-1}a_n, \ldots$(these are suffix substrings)

eg, x, xy, xyy, xyyx, xyyxy are prefixes of w

y, xy, yxy, yyxy, xyyxy are suffixes of w

BLM2502 Theory of Computation

» Examples on String Operations

Special cases:

ε is prefix and suffix of each string

any string is prefix and suffix of itself

String: abba

Prefix	Suffix
ε	abba
a	bba
ab	ba
abb	a
abba	ε

Observe that:
εw = wε = w

BLM2502 Theory of Computation

- Examples on String Operations

  $w = a_1a_2\ldots a_n$, $v = b_1b_2\ldots b_m$ $a,b \in \{x, y\}$

  Self Concenation:

  $ww = w^2$, $www = w^3$, etc

  $w^0 = \varepsilon$

  eg, w = abba;

  $(abba)^0 = \varepsilon$

  $(abba)^1 = abba$

  $(abba)^2 = abbaabba$

  $(abba)^3 = abbaabbaabba$

  Kleene Star:

  $w^* = \{w^0, w^1, w^2, w^3\}$

BLM2502 Theory of Computation

- The * operation:

  The set of all possible strings from the alphabet

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

  $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

- The + operation:

  The set of all possible strings from the alphabet excluding ε

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

  $\Sigma^* = \Sigma^* - \{\varepsilon\} = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

BLM2502 Theory of Computation

- Language:

A Language over an alphabet $\Sigma = \{a, b\}$

is a subset of $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Examples:

$L_1 = \{\varepsilon\}$

$L_2 = \{a, b, aa, ab, ba, bb\}$

$L_3 = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$

$L_4 = \{a^n b^n, n \geq 0\} = \{\varepsilon, ab, aabb, aaabbbb, \ldots\}$

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

PRIME_NUMBERS $= \{x \mid x \in \Sigma^* \text{ and } x \text{ is prime}\}$

$= \{2, 3, 5, 7, 11, \ldots\}$

EVEN_NUMBERS $= \{x \mid x \in \Sigma^* \text{ and } x \text{ is even}\} = \{0, 2, 4, \ldots\}$

ODD_NUMBERS $= \{x \mid x \in \Sigma^* \text{ and } x \text{ is odd}\} = \{1, 3, 5, \ldots\}$

BLM2502 Theory of Computation

- Unary Addition

  Alphabet $\Sigma = \{1, +, =\}$

  ADDITION = $\{x + y = z: x = 1^m, y=1^n, z = 1^k; k = m + n\}$

  $111 + 11 = 11111 \in$ ADDITION

  $111 + 111 = 1111 \notin$ ADDITION

- Squaring

  Alphabet $\Sigma = \{1, \#\}$

  SQUARES= $\{x \# y : x = 1^m, y=1^n, n = m^2\}$

  $111 \# 111111111 \in$ SQUARES

  $1111 \# 11111111 \notin$ SQUARES

BLM2502 Theory of Computation

» Operations On Languages

Set Operations: Regular sets are closed under union, intersection negation and complement.

$A = \{a, ab, aaaa\}$

$B = \{ab, bb\}$

$A \cup B = \{a, ab, bb, aaaa\}$

$A \cap B = \{ab\}$

$A - B = \{a, bb, aaaa\}$

$\bar{A} = \Sigma^* - A = \overline{\{a, ab, aaaa\}} = \{\varepsilon, aa, ba, bb, aba, \ldots\}$

Note that :

$\emptyset = \{\} \neq \{\varepsilon\}$

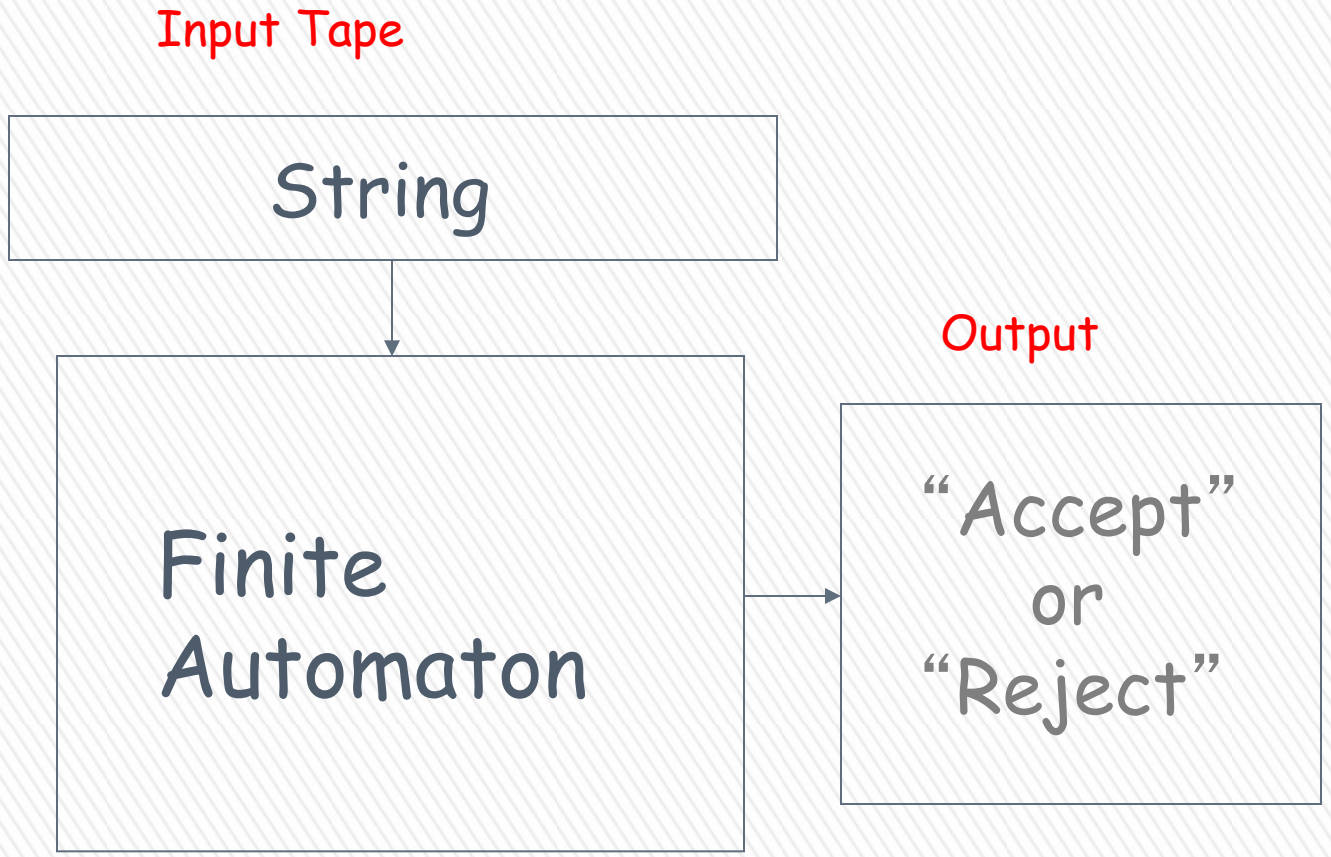
$|\emptyset| = |\{\}| = 0 \neq |\{\varepsilon\}|$

$|\{\varepsilon\}| = 1$ * This is set size

$|\varepsilon| = 0$ * This is string length

Finite Automata

»


Input Tape
String
Finite Automaton
Output
"Accept" or "Reject"

BLM2502 Theory of Computation

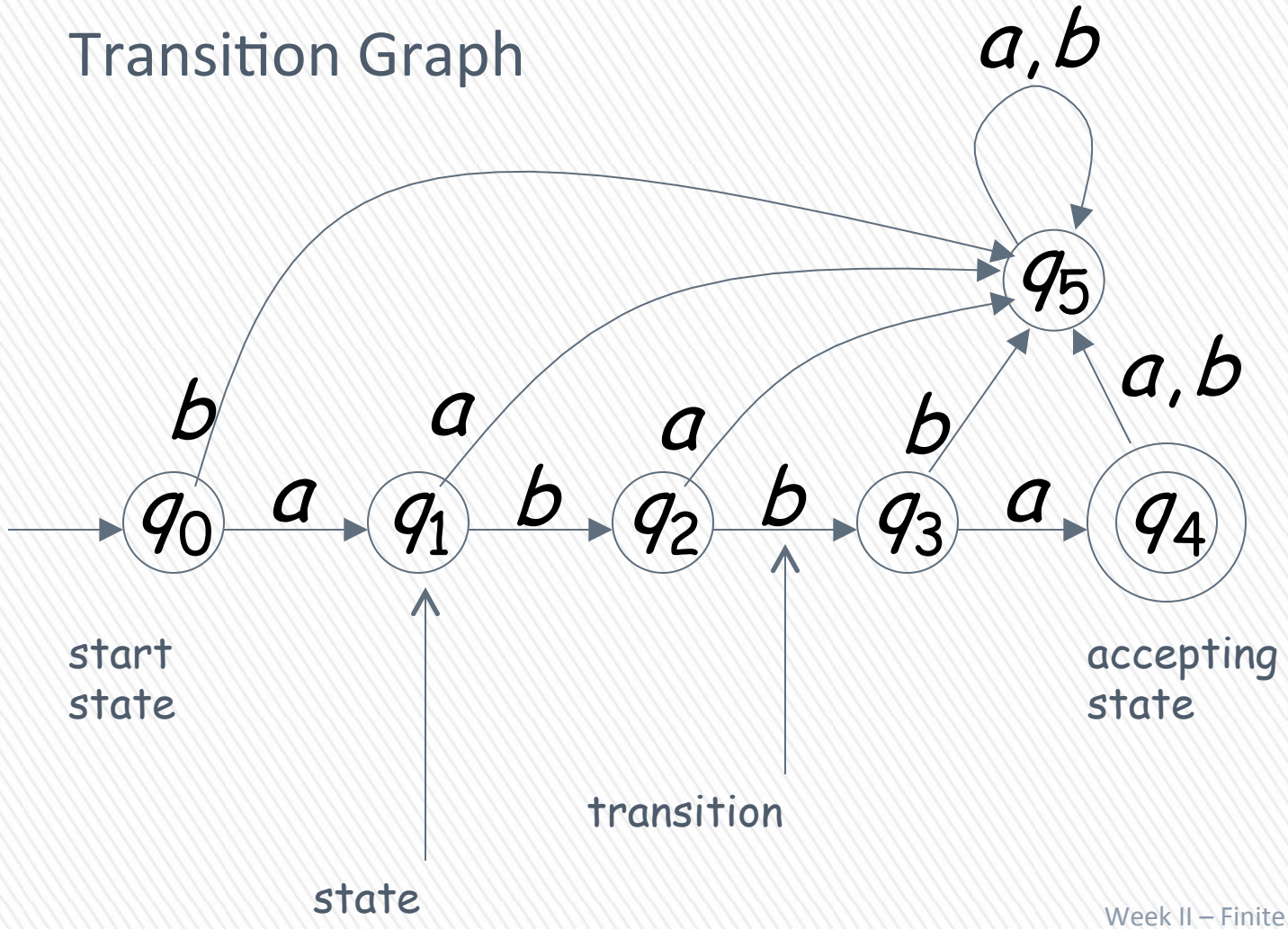
Finite Automata

» 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. δ: $Q \times \Sigma \rightarrow Q$ is the ***transition function,***
4. $q_0 \in Q$ is the ***start state,*** and
5. $F \subseteq Q$ is the ***set of accept states.***

Transition Graph

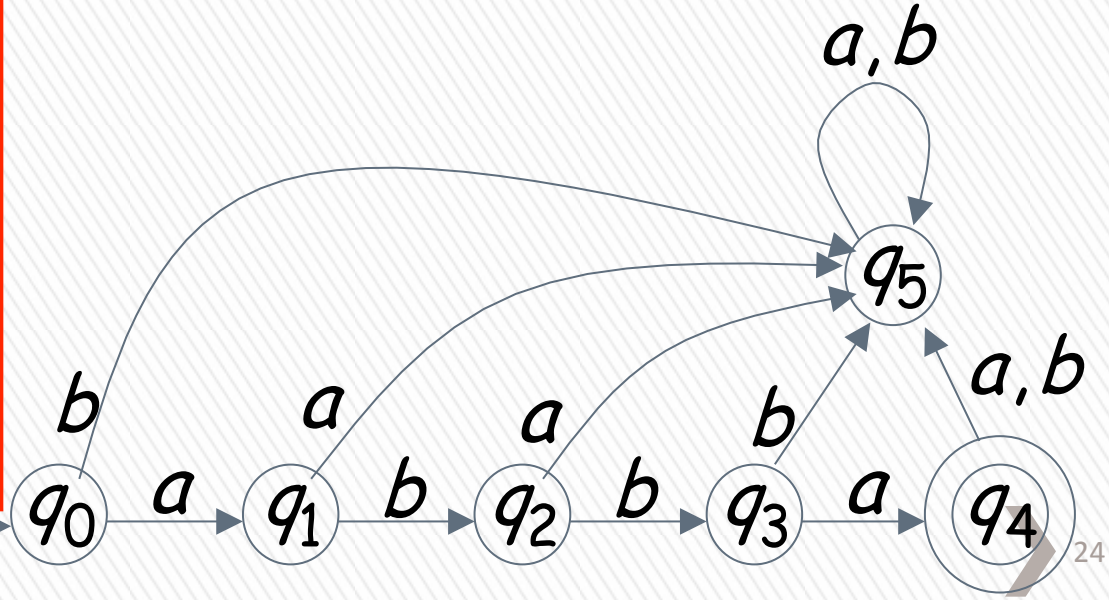


BLM2502 Theory of Computation

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



BLM2502 Theory of Computation

- You can think of the transition function as being the "program" of the finite automaton M. This function tells us what M can do in "one step":
  - Let r be a state of Q and let a be a symbol of the alphabet Σ. If the finite automaton M is in state r and reads the symbol a, then it switches from state r to state $\delta(r, a)$. (In fact, $\delta(r, a)$ may be equal to r.)
- Example:
  - A = {w : w is a binary string containing an odd number of 1s}.

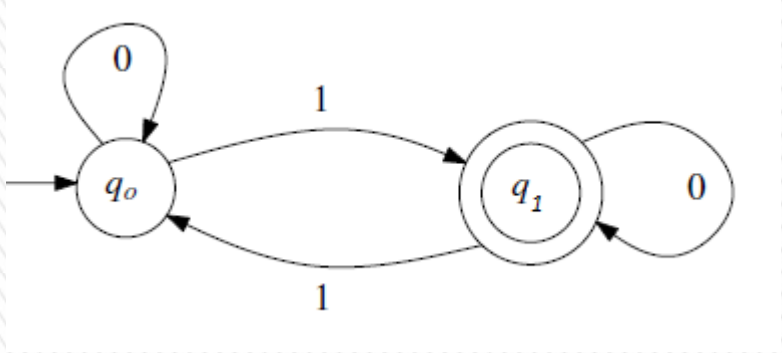
BLM2502 Theory of Computation

Design:

- The finite automaton reads the input string w from left to right and keeps track of the number of 1s it has seen. After having read the entire string w, it checks whether this number is odd (in which case w is accepted) or even (in which case w is rejected).
- Using this approach, the finite automaton needs a state for every integer $i \geq 0$, indicating that the number of 1s read so far is equal to i.

BLM2502 Theory of Computation

- Design – Continued
- However, this is design is not feasible since FA have finite number of states.
- ???
- A better, and correct approach, is to keep track of whether the number of 1s read so far is even or odd.



BLM2502 Theory of Computation

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

$Q = \{q_0, q_1\}$

$\Sigma = \{0, 1\}$ (trivial from the problem)

$q_0 \in Q$

$F = \{q_1\}$

δ:

$(q_0, 0) \rightarrow q_0$

$(q_0, 1) \rightarrow q_1$

$(q_1, 0) \rightarrow q_1$

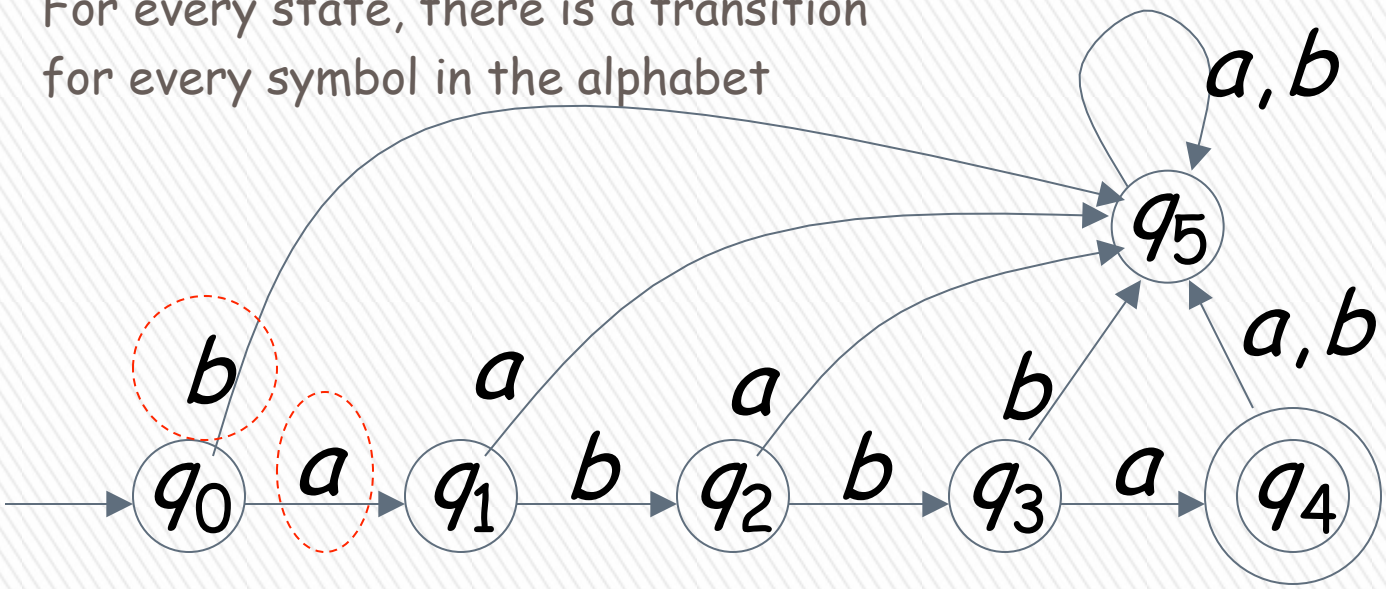
$(q_1, 1) \rightarrow q_0$

δ:	0	1
q_0	q_0	q_1
q_1	q_1	q_0

BLM2502 Theory of Computation

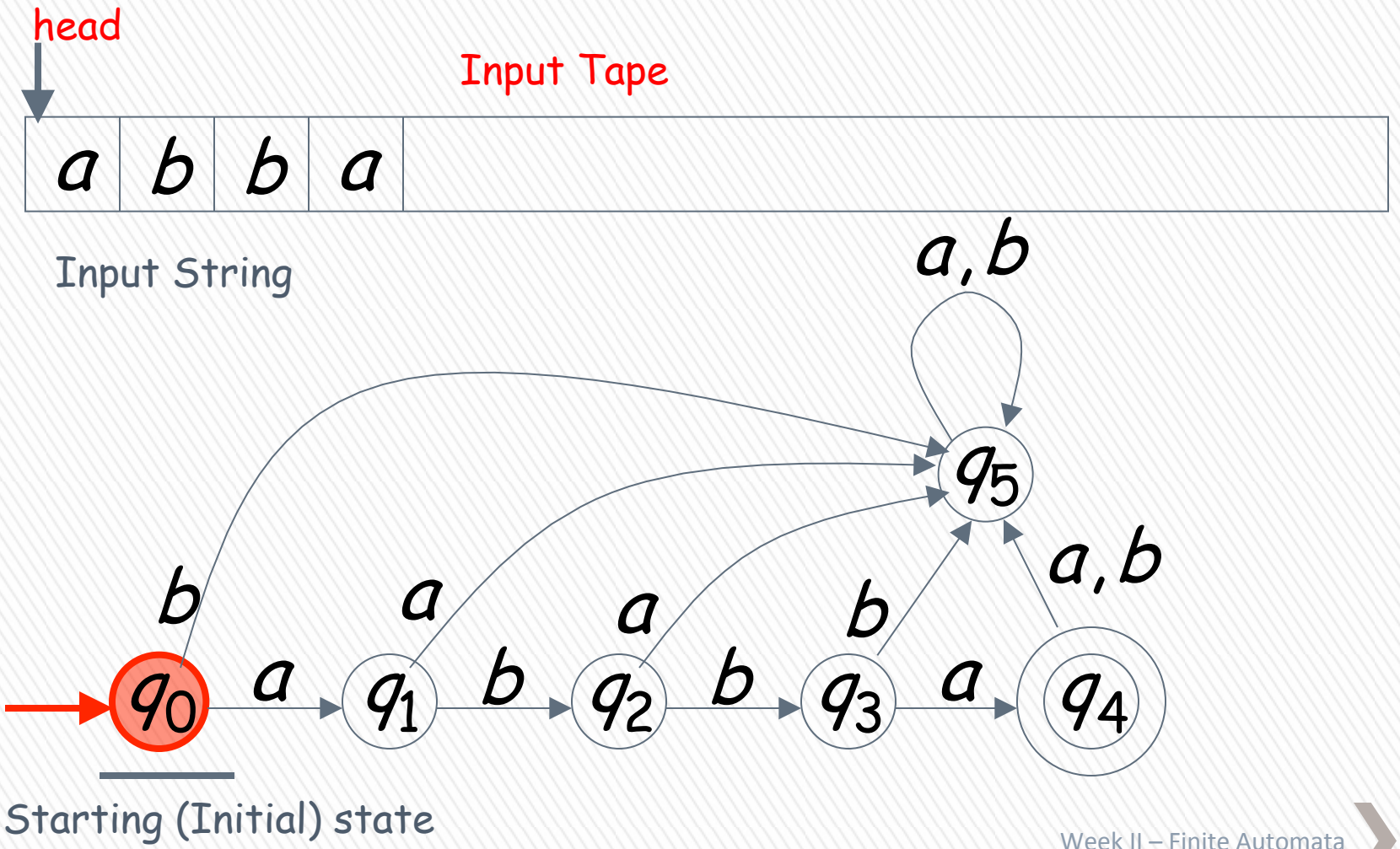
» DFA - Deterministic Finite Automata

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

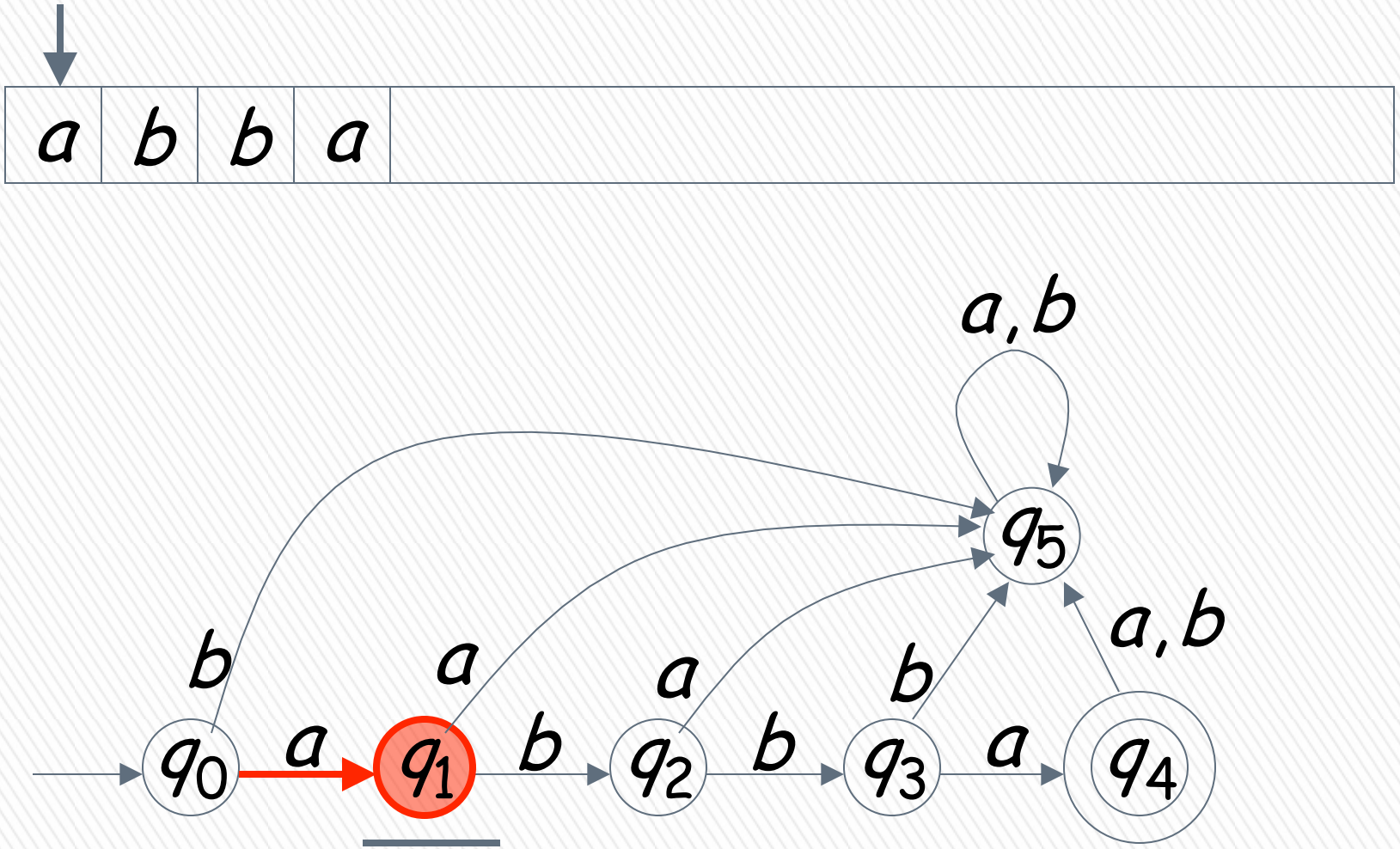


Alphabet Σ = {a, b}

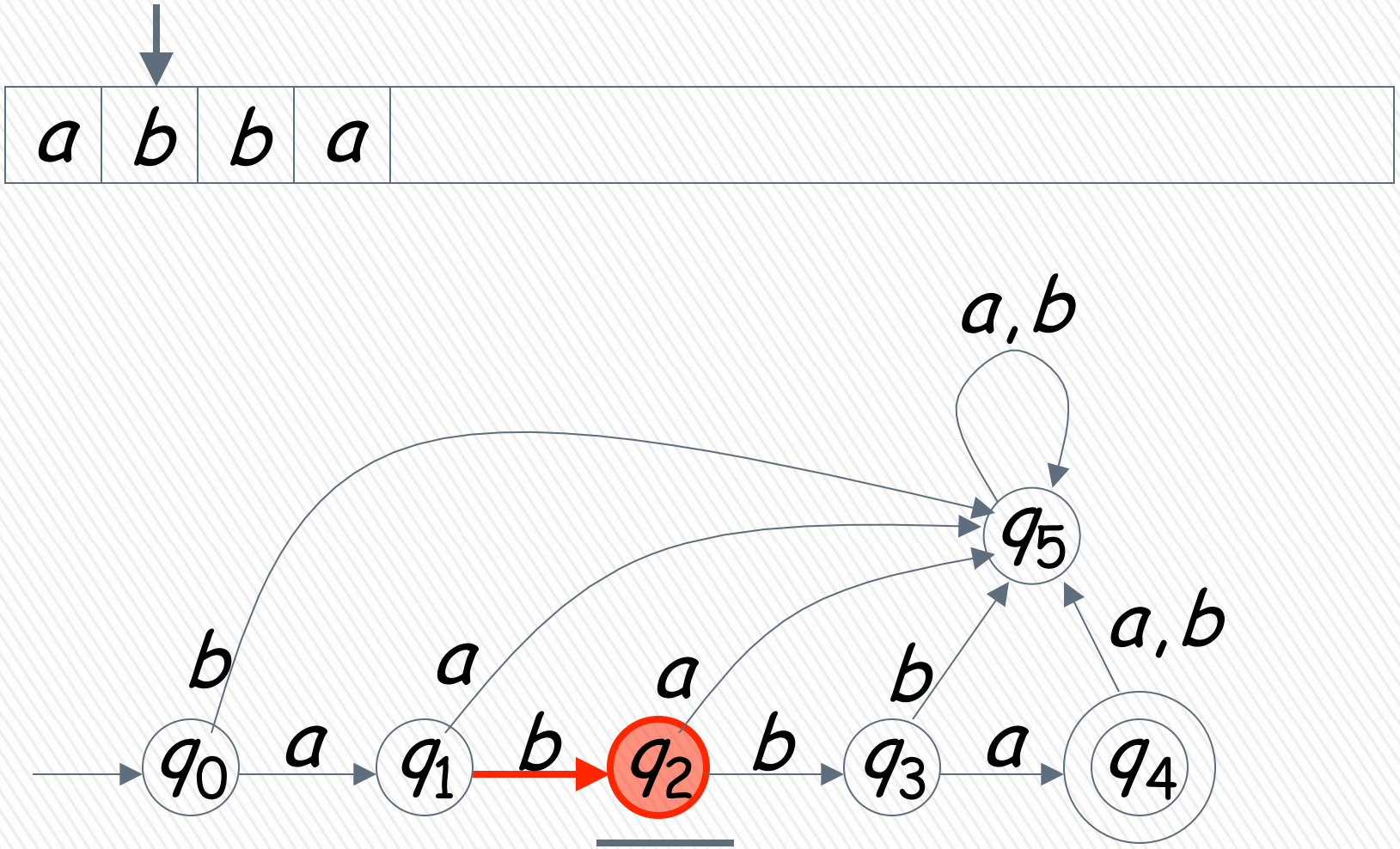
BLM2502 Theory of Computation



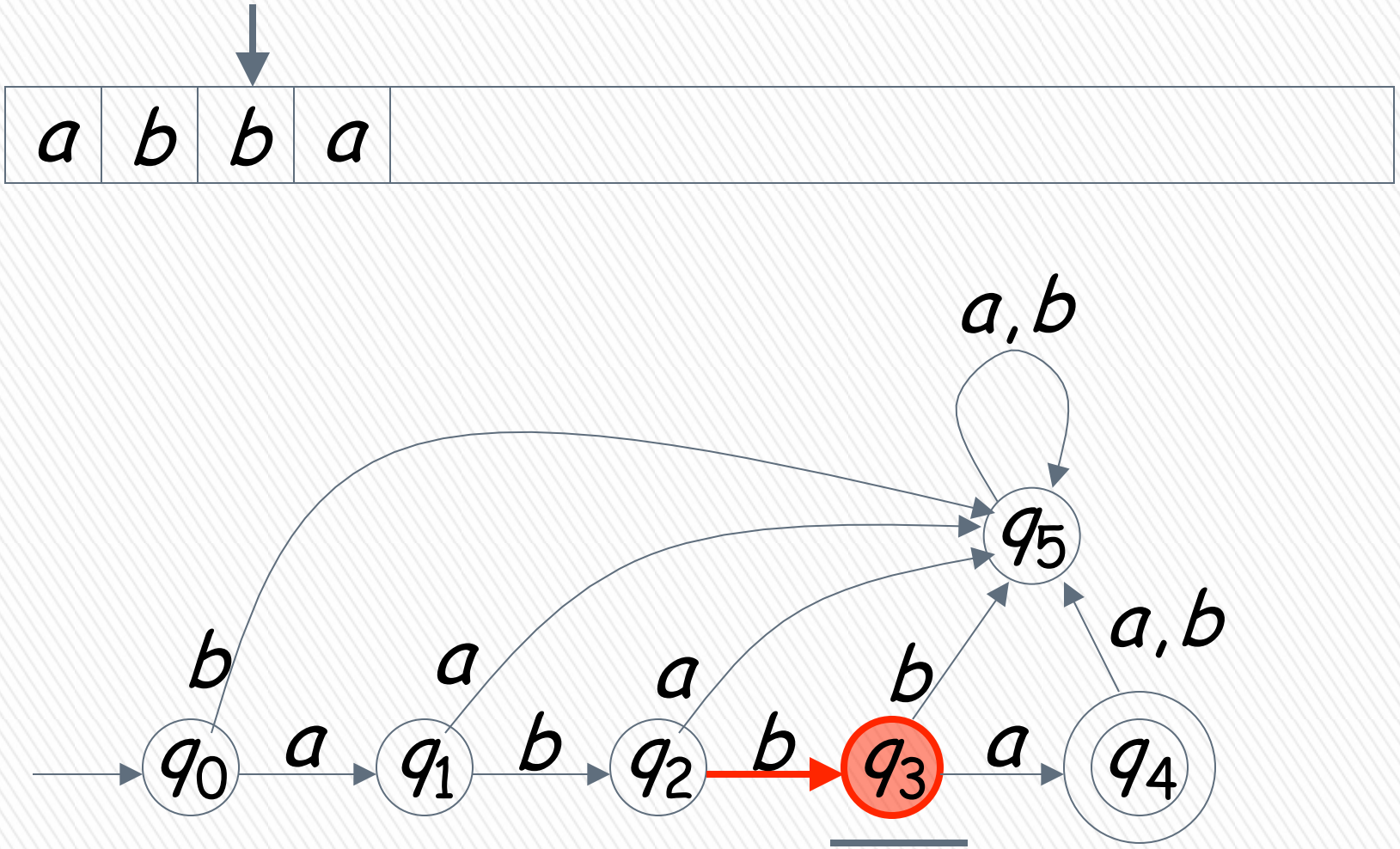
BLM2502 Theory of Computation



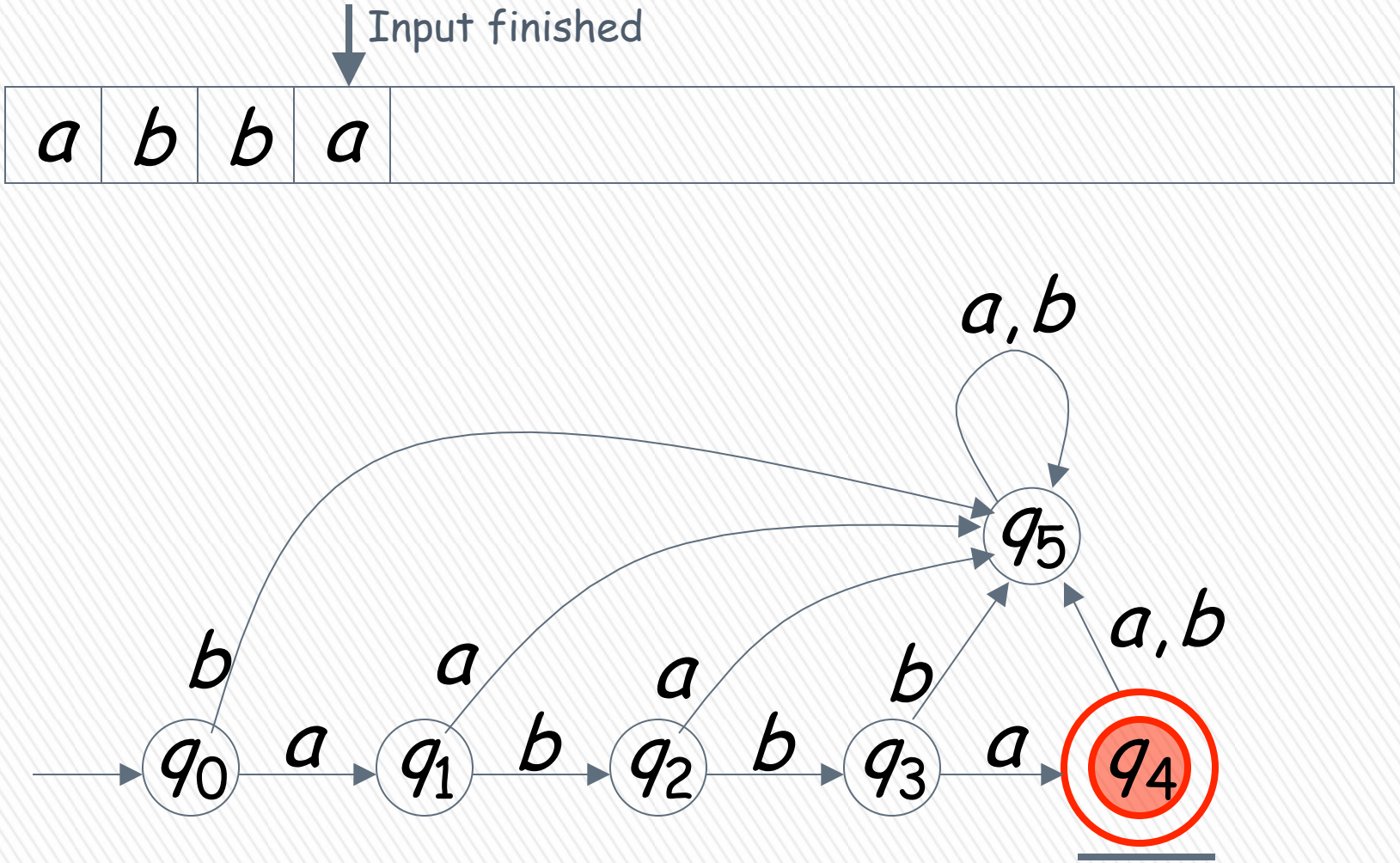
BLM2502 Theory of Computation

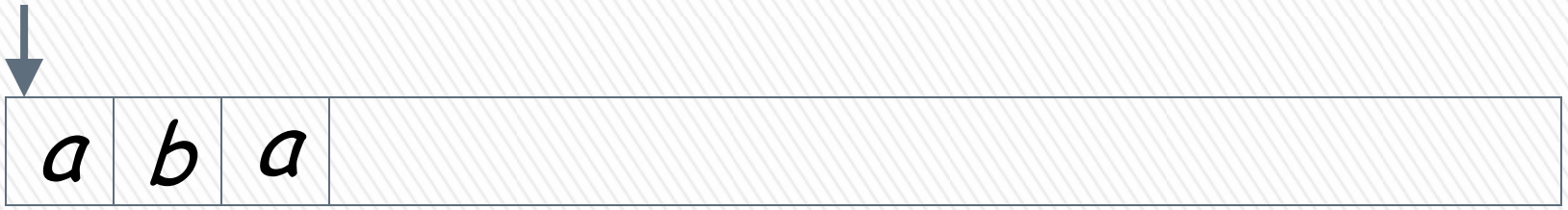


BLM2502 Theory of Computation

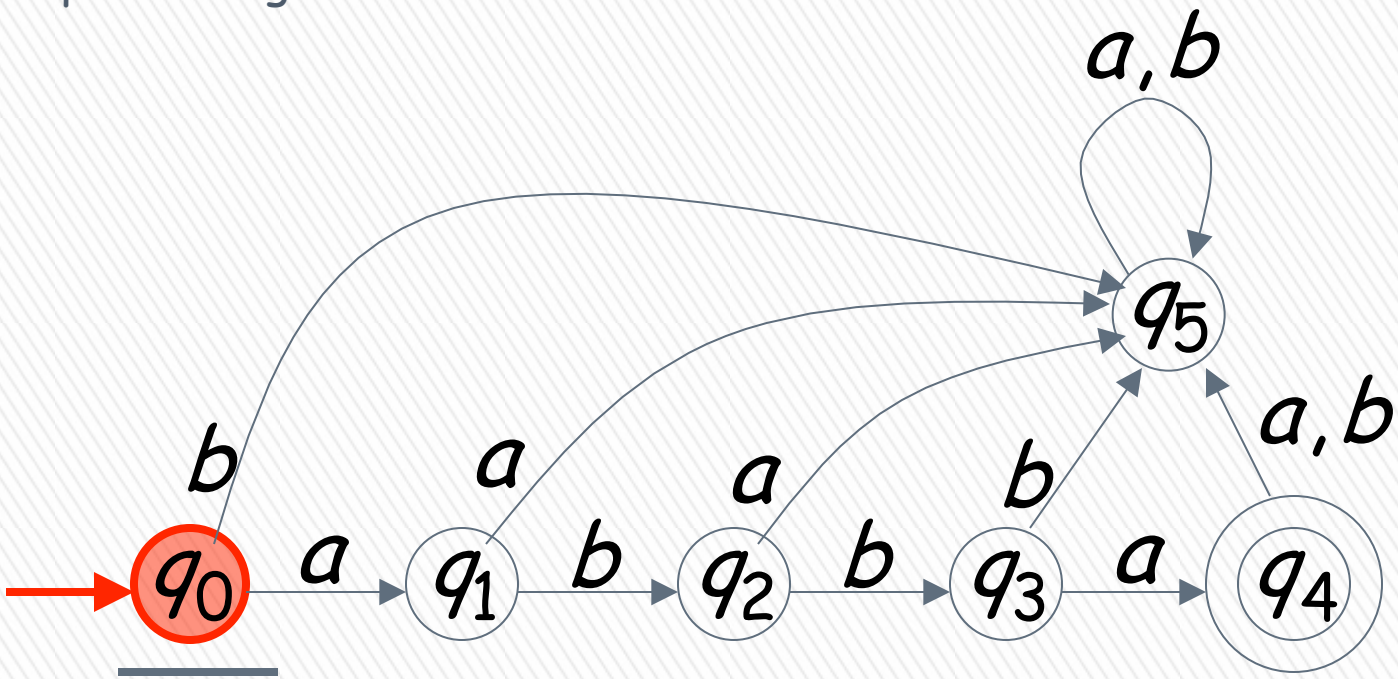


BLM2502 Theory of Computation



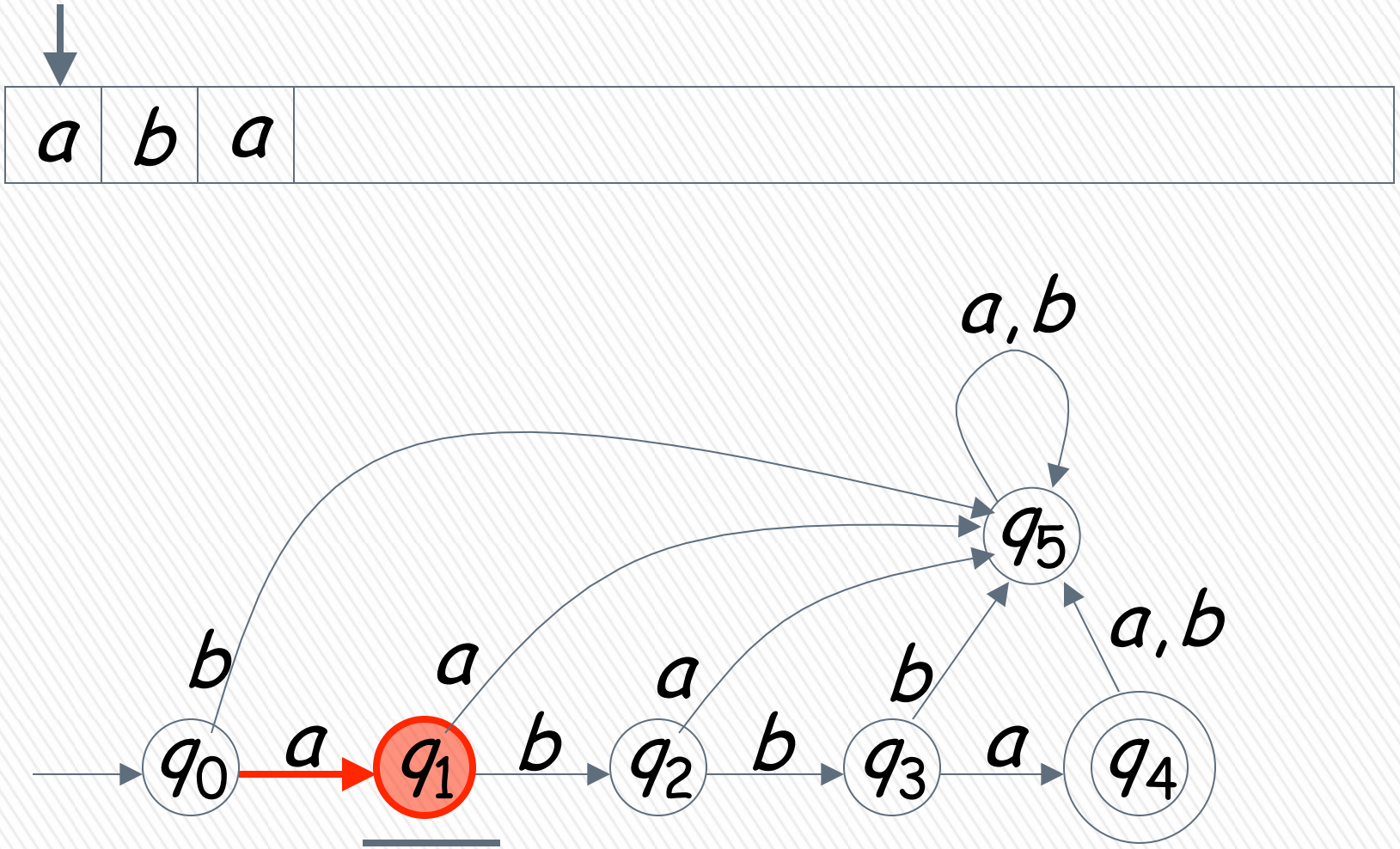


Input String

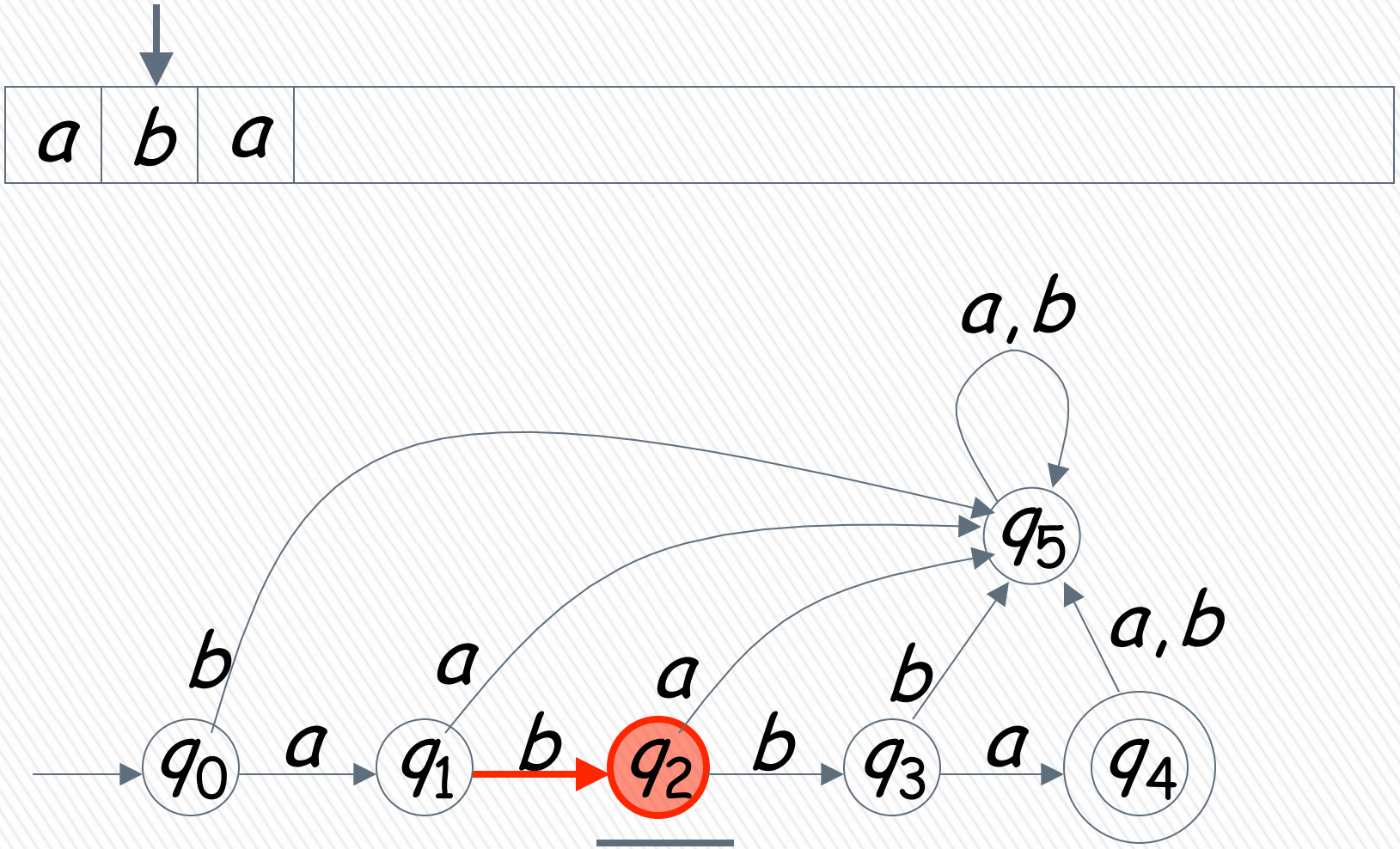


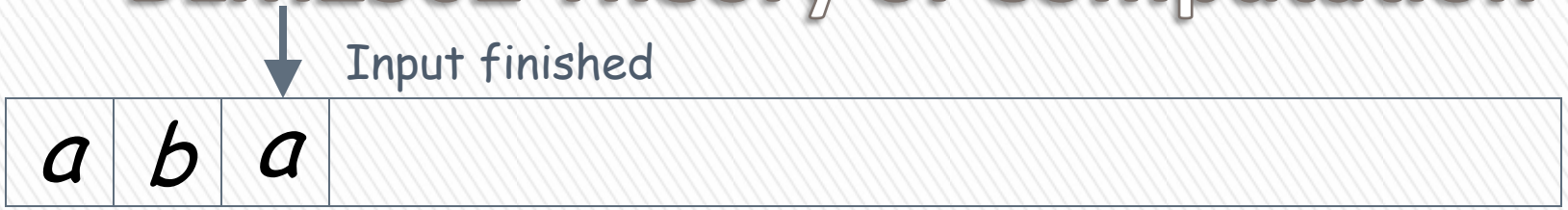
A Rejection Case

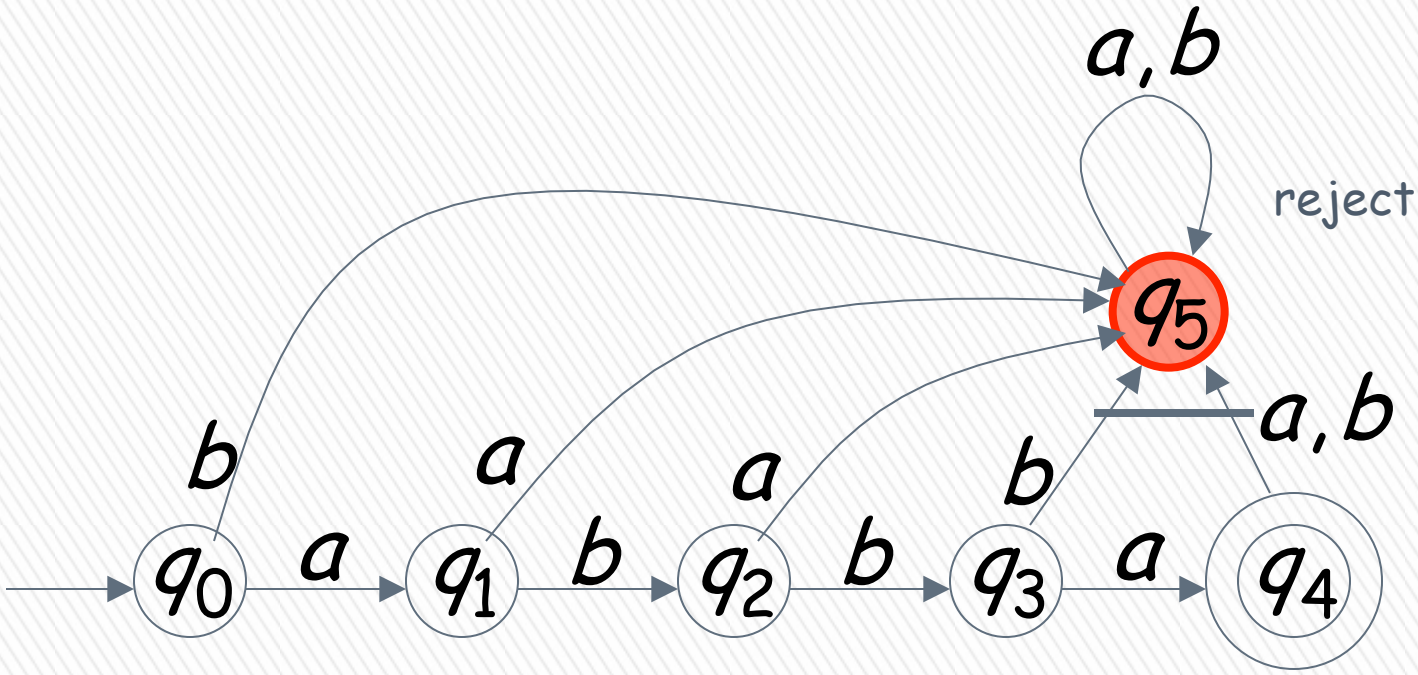
BLM2502 Theory of Computation



BLM2502 Theory of Computation

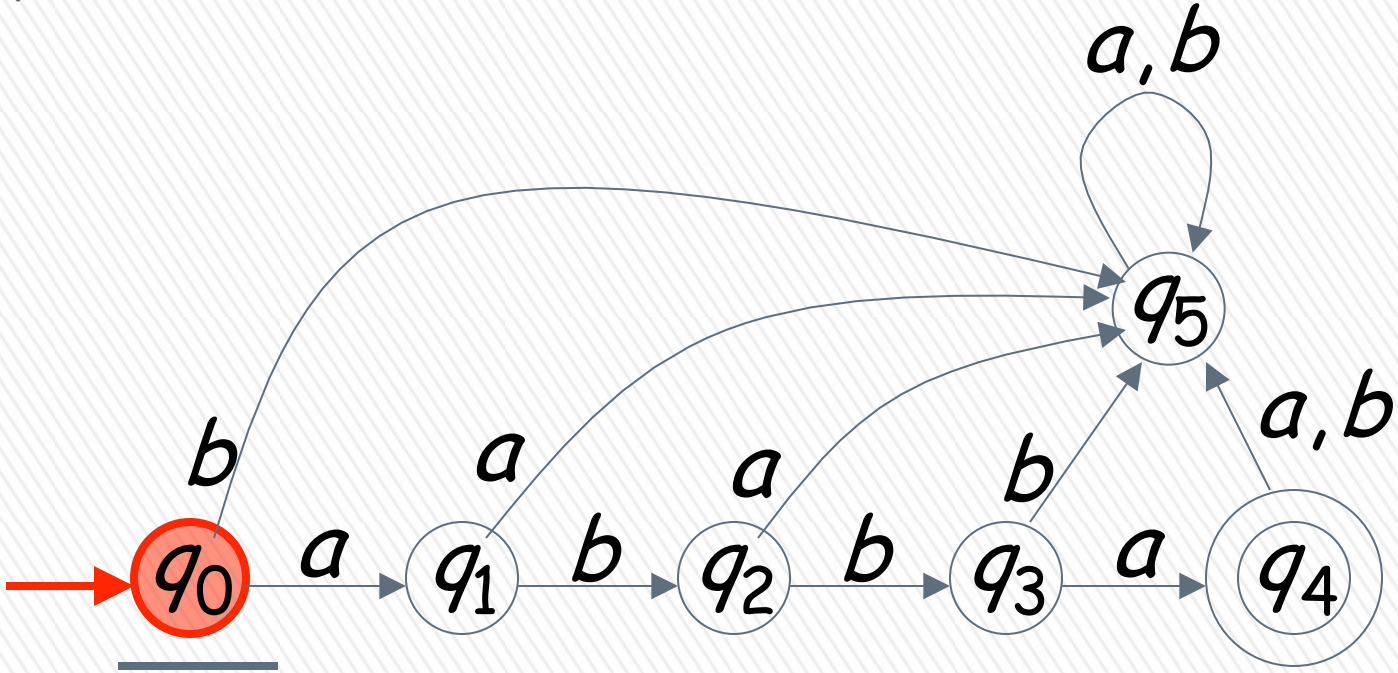



Input finished
a
b
a


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

Tape is empty

Input Finished



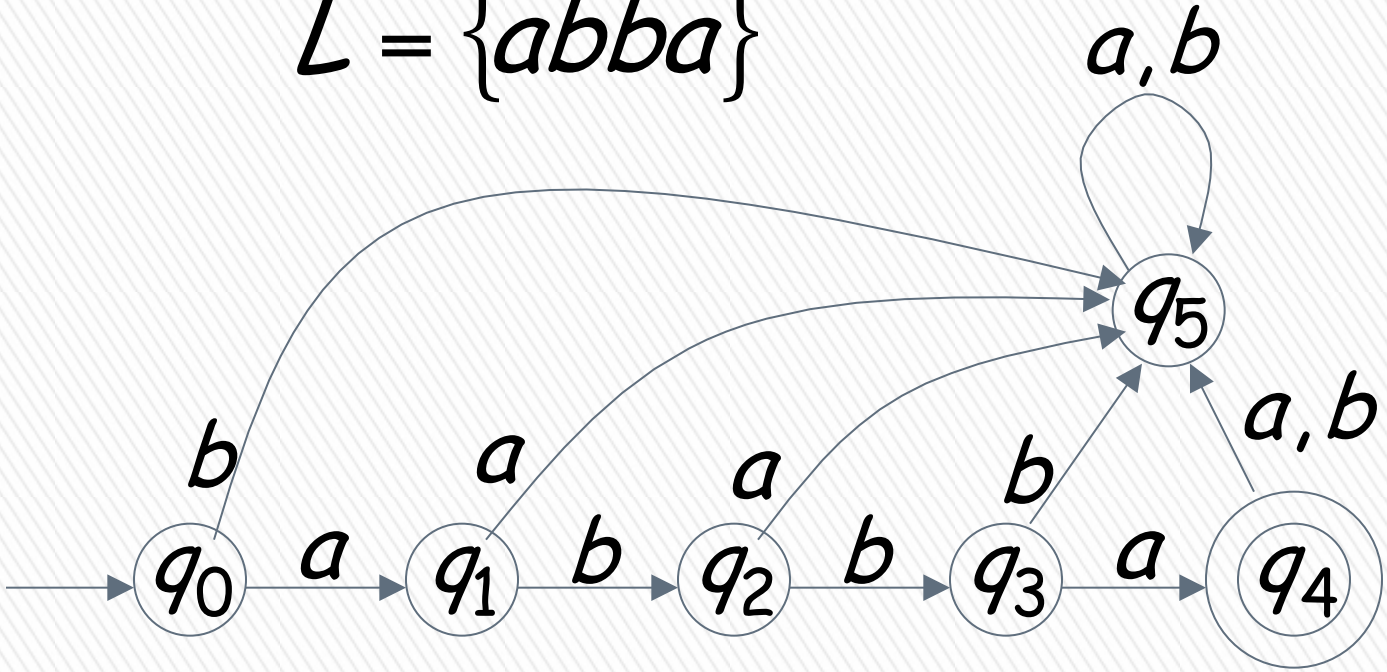
reject

Another Rejection Case

BLM2502 Theory of Computation

Language Accepted:

$$L = \{abba\}$$



BLM2502 Theory of Computation

To accept a string:

- all symbols of the input string is scanned

and

- the last state is accepting

To reject a string:

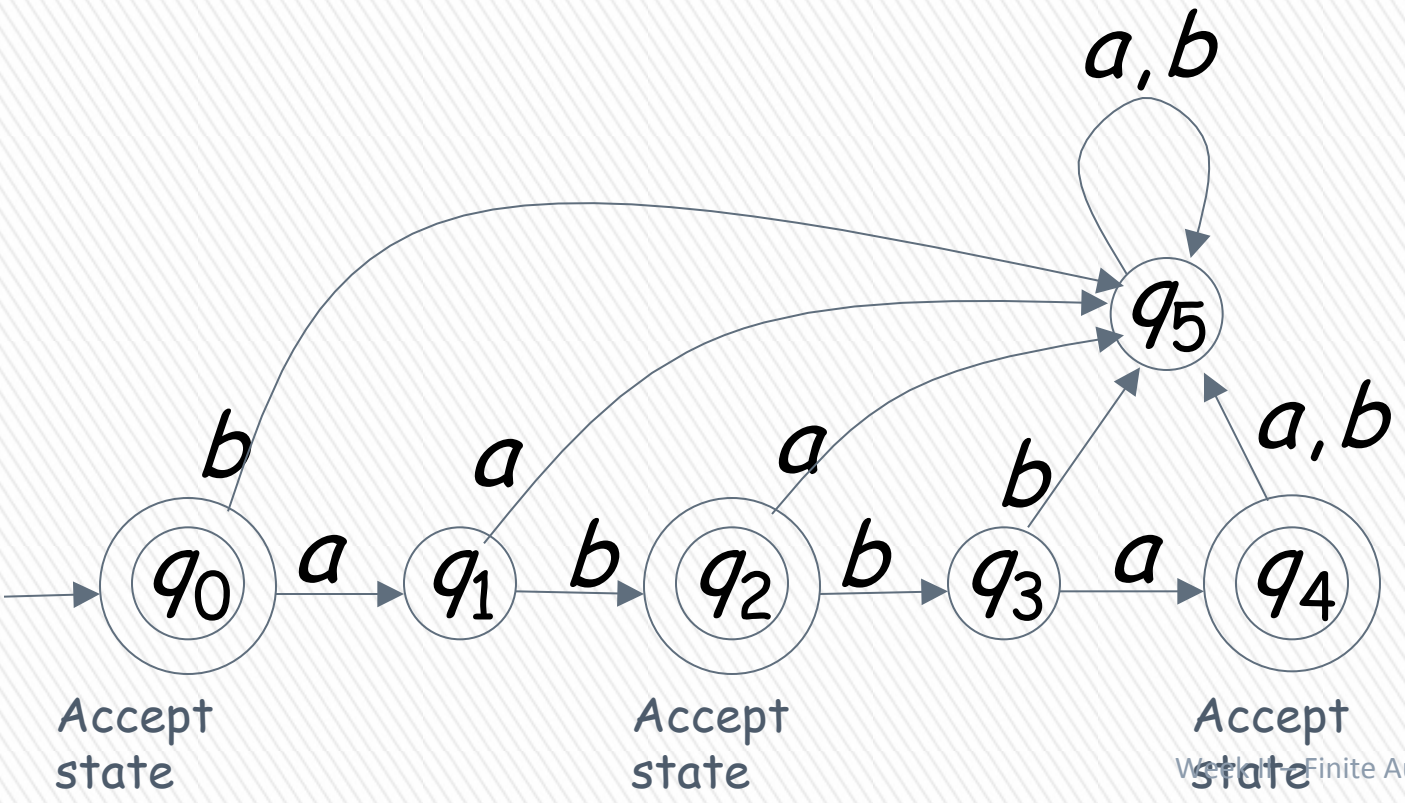
- all symbols the input string is scanned

and

- the last state is non-accepting

BLM2502 Theory of Computation

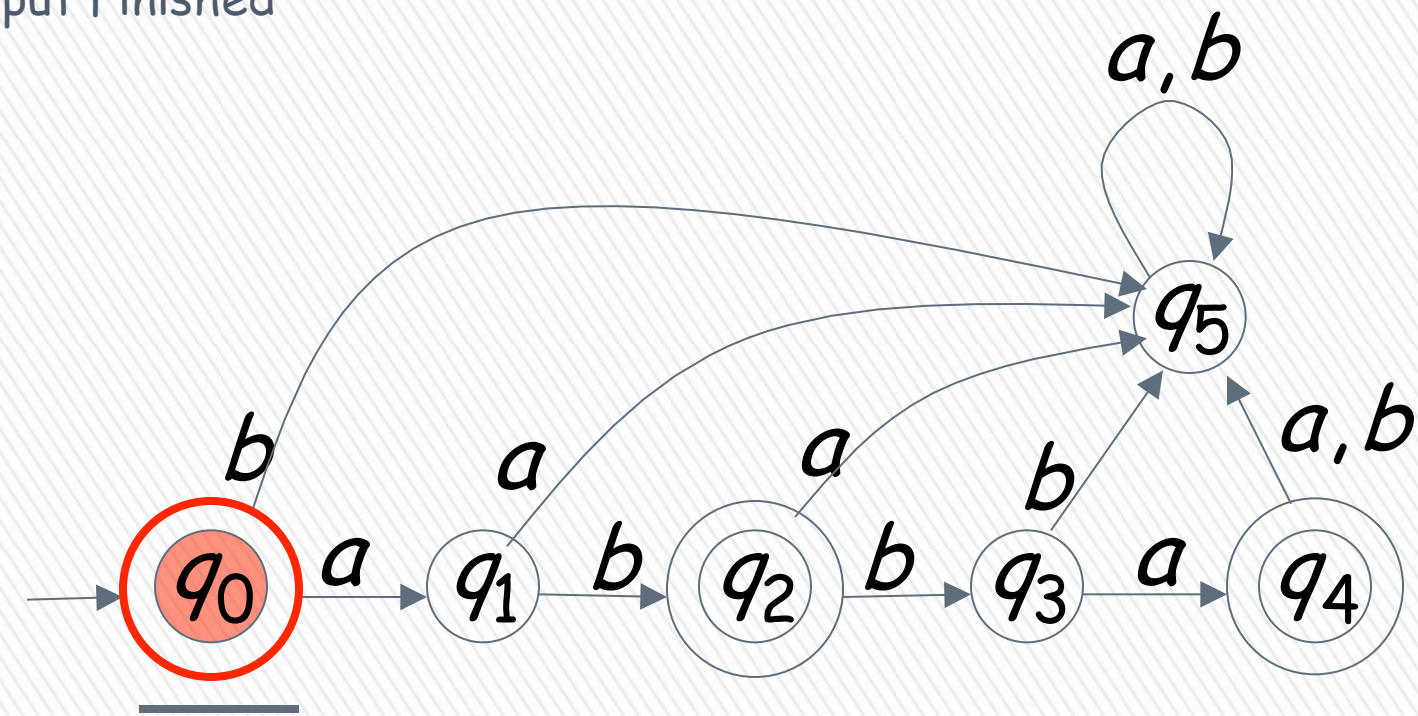
L = {ε, ab, abba}



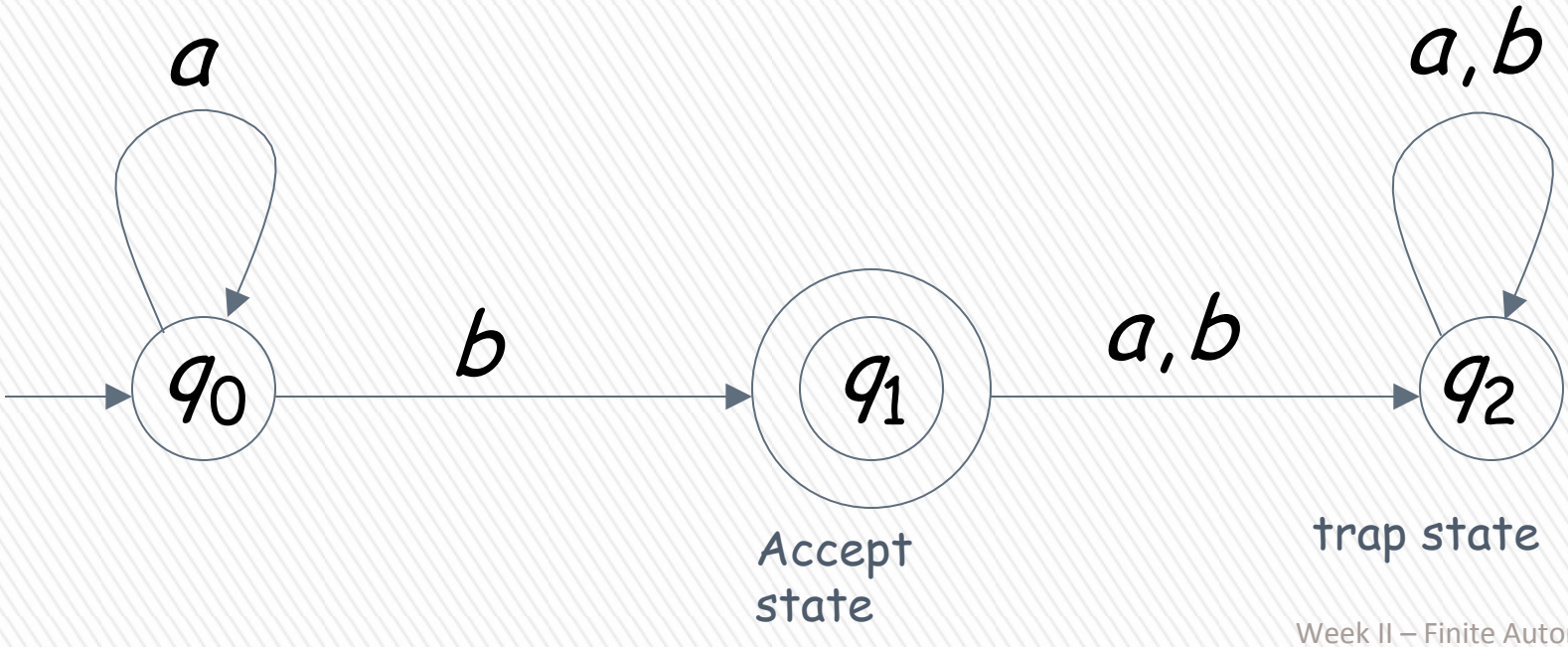
BLM2502 Theory of Computation

Empty Tape

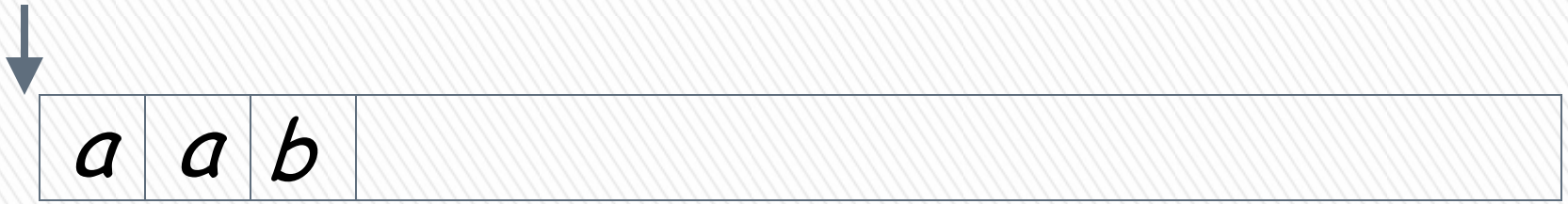
Input Finished



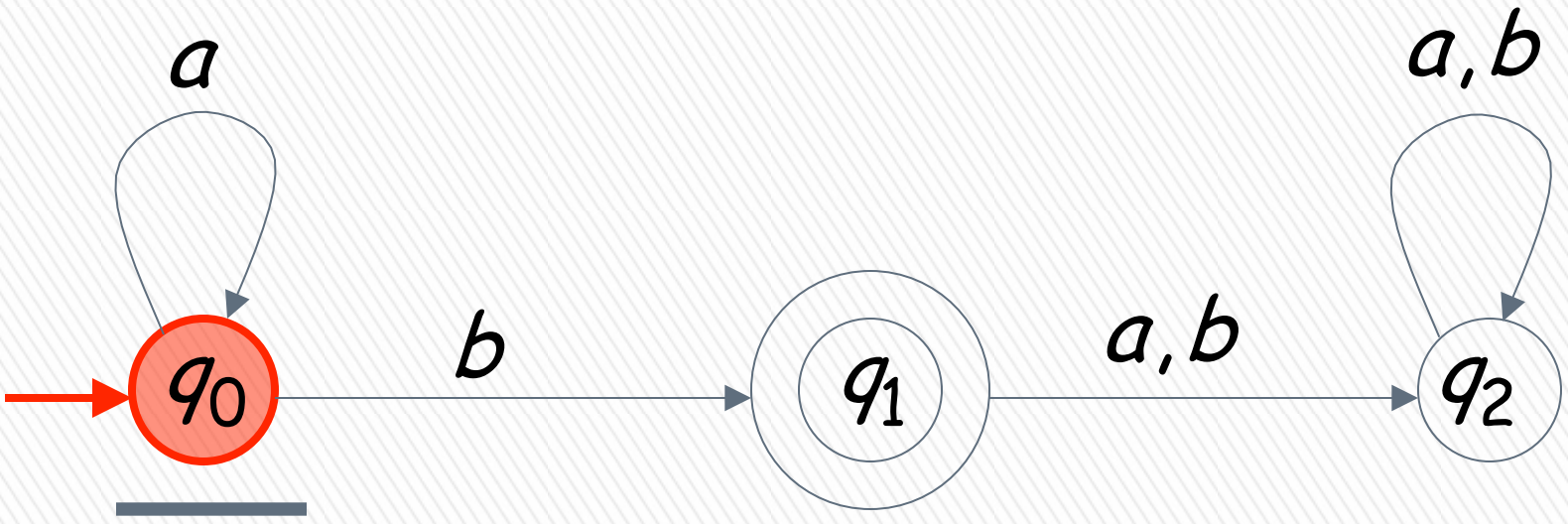
BLM2502 Theory of Computation

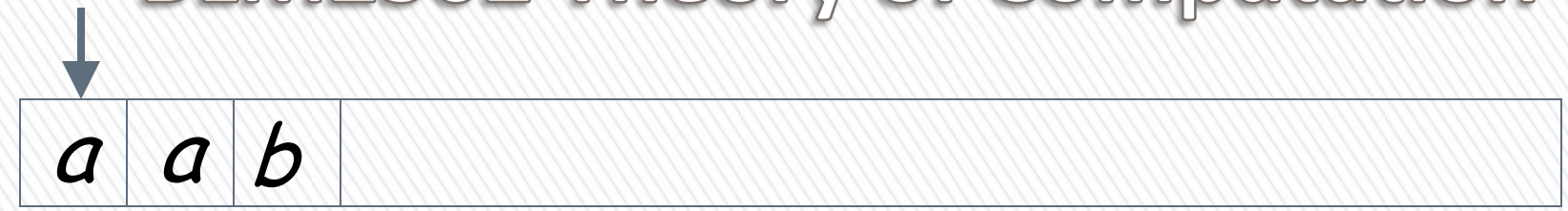


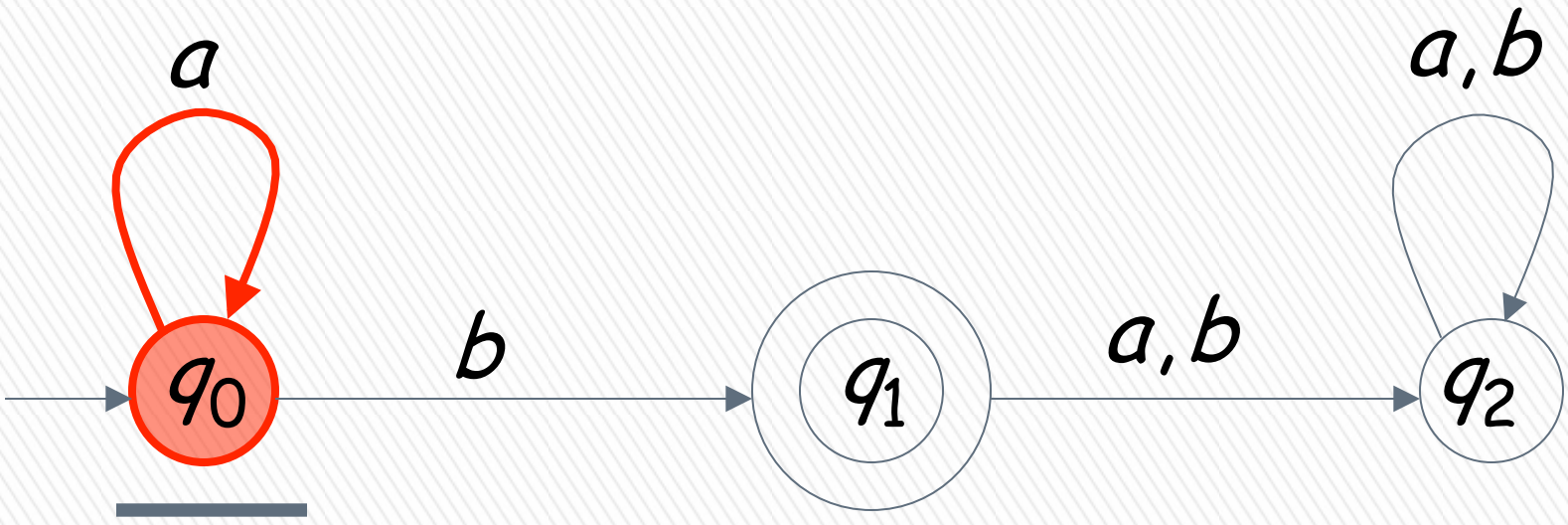
BLM2502 Theory of Computation

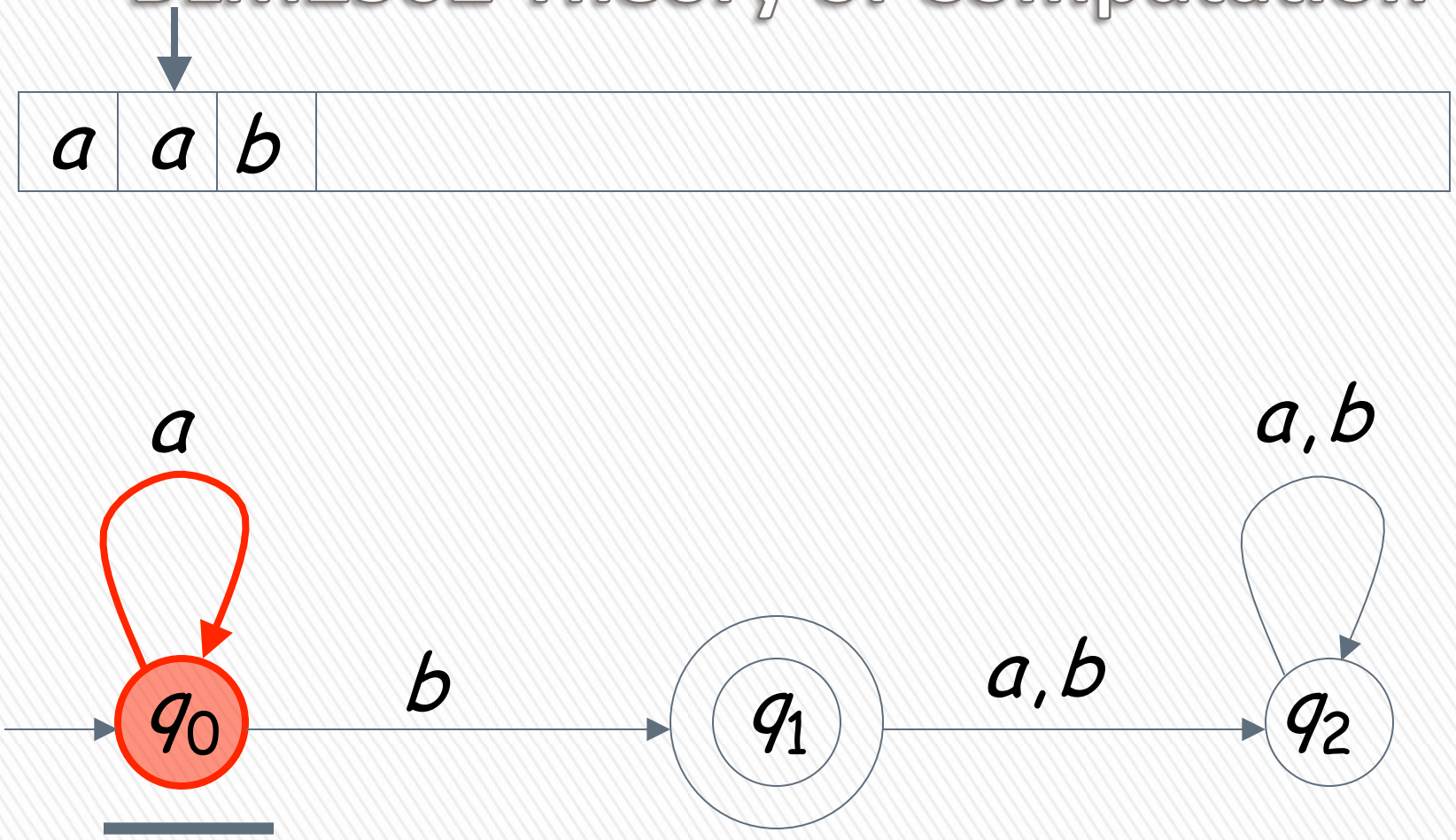


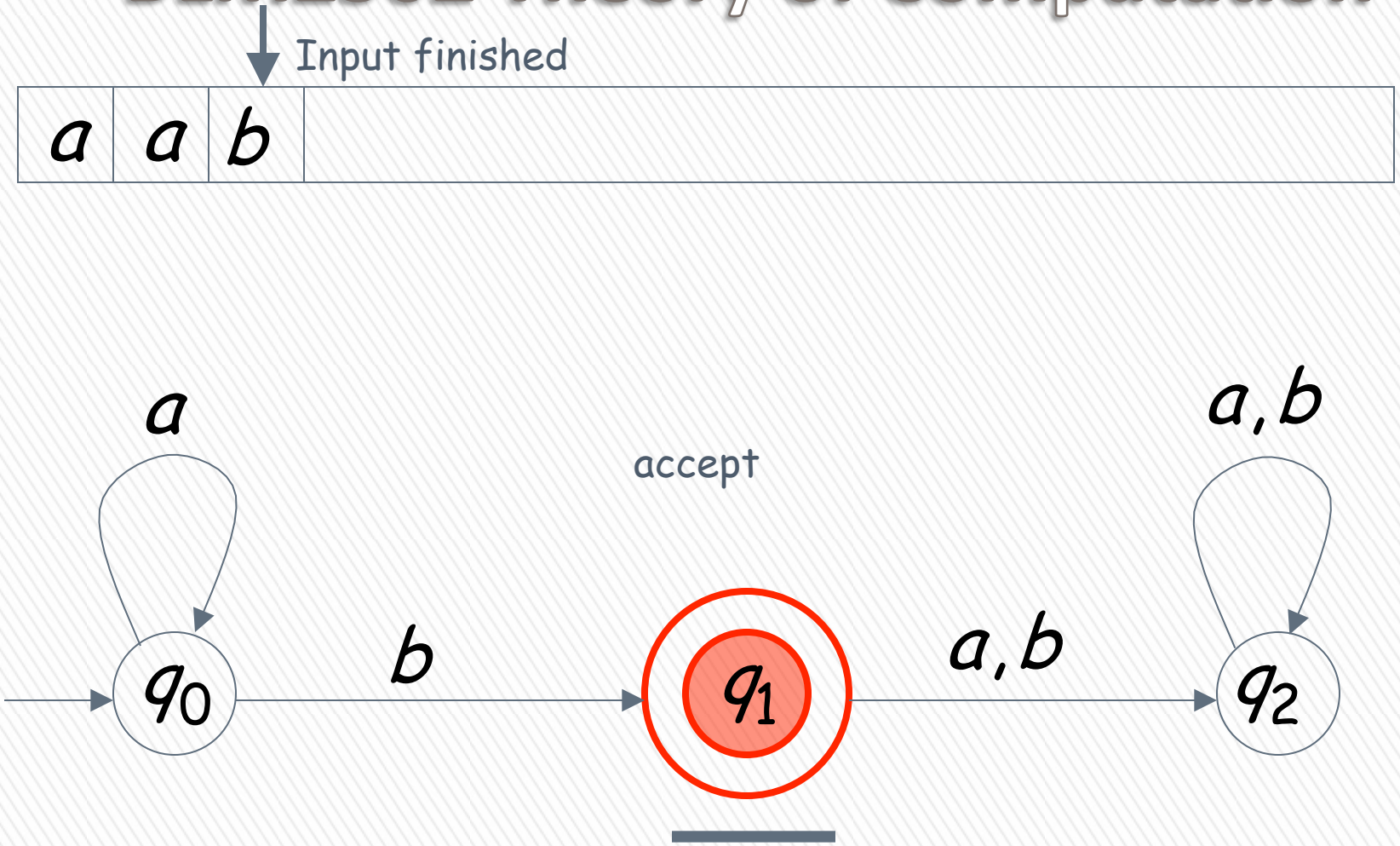
Input String

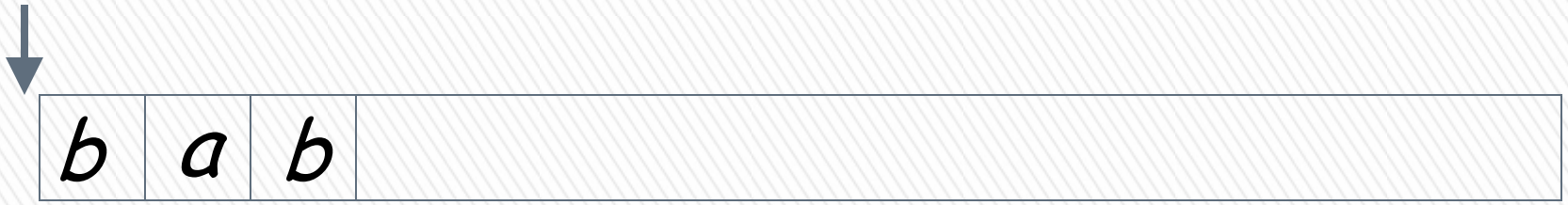



a
a
b

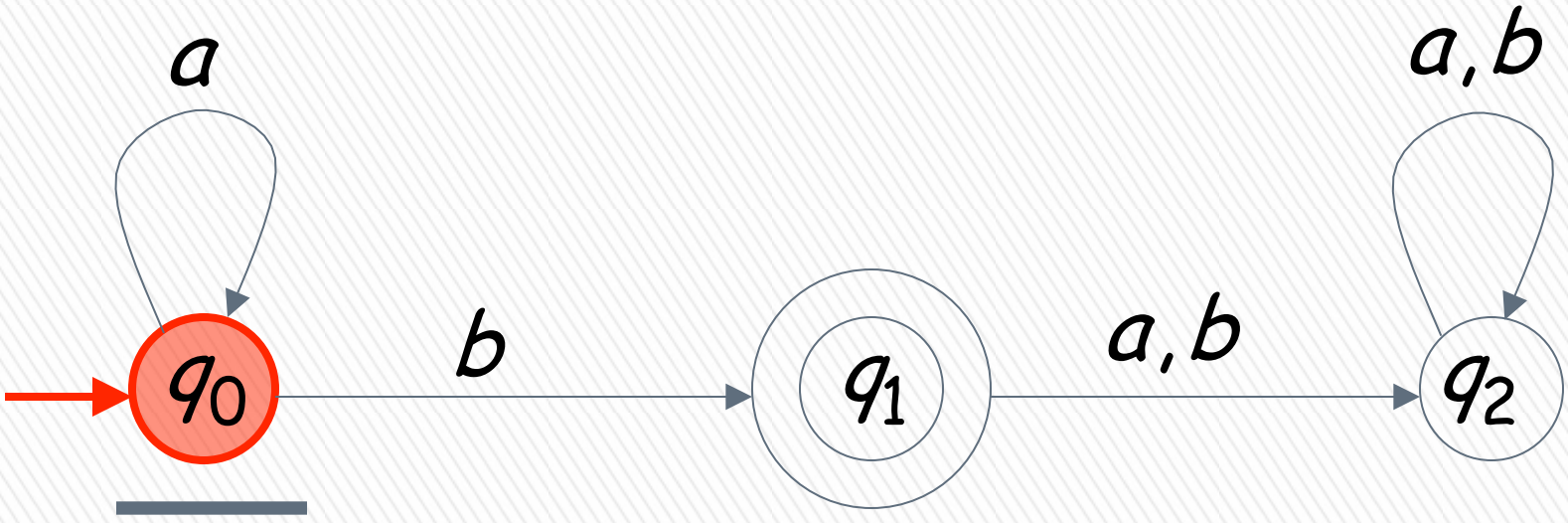

a
b
a, b
a, b
q0
q1
q2


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

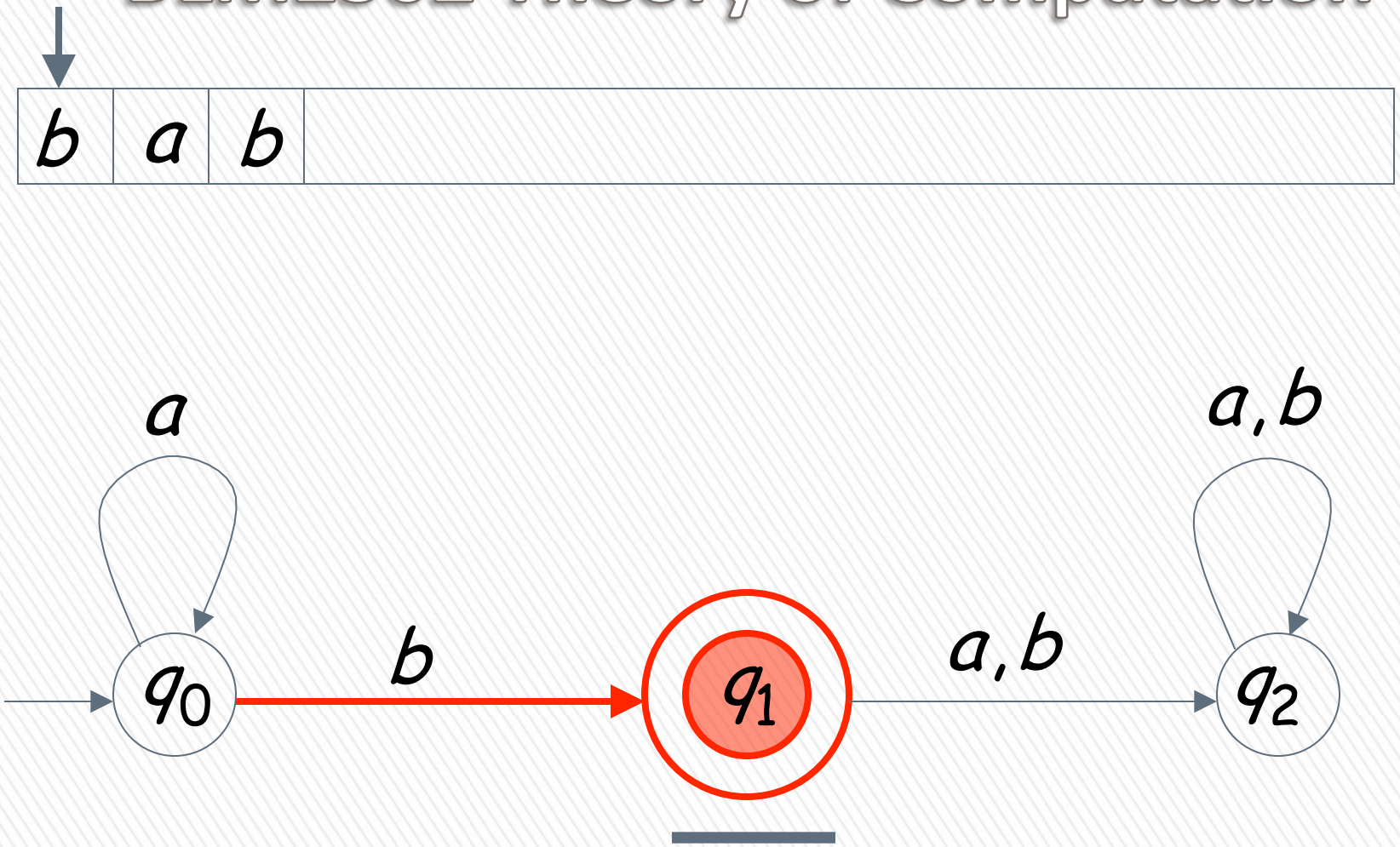

Input finished
a
a
b
a
accept
a, b
b
a, b
q_0
q_1
q_2

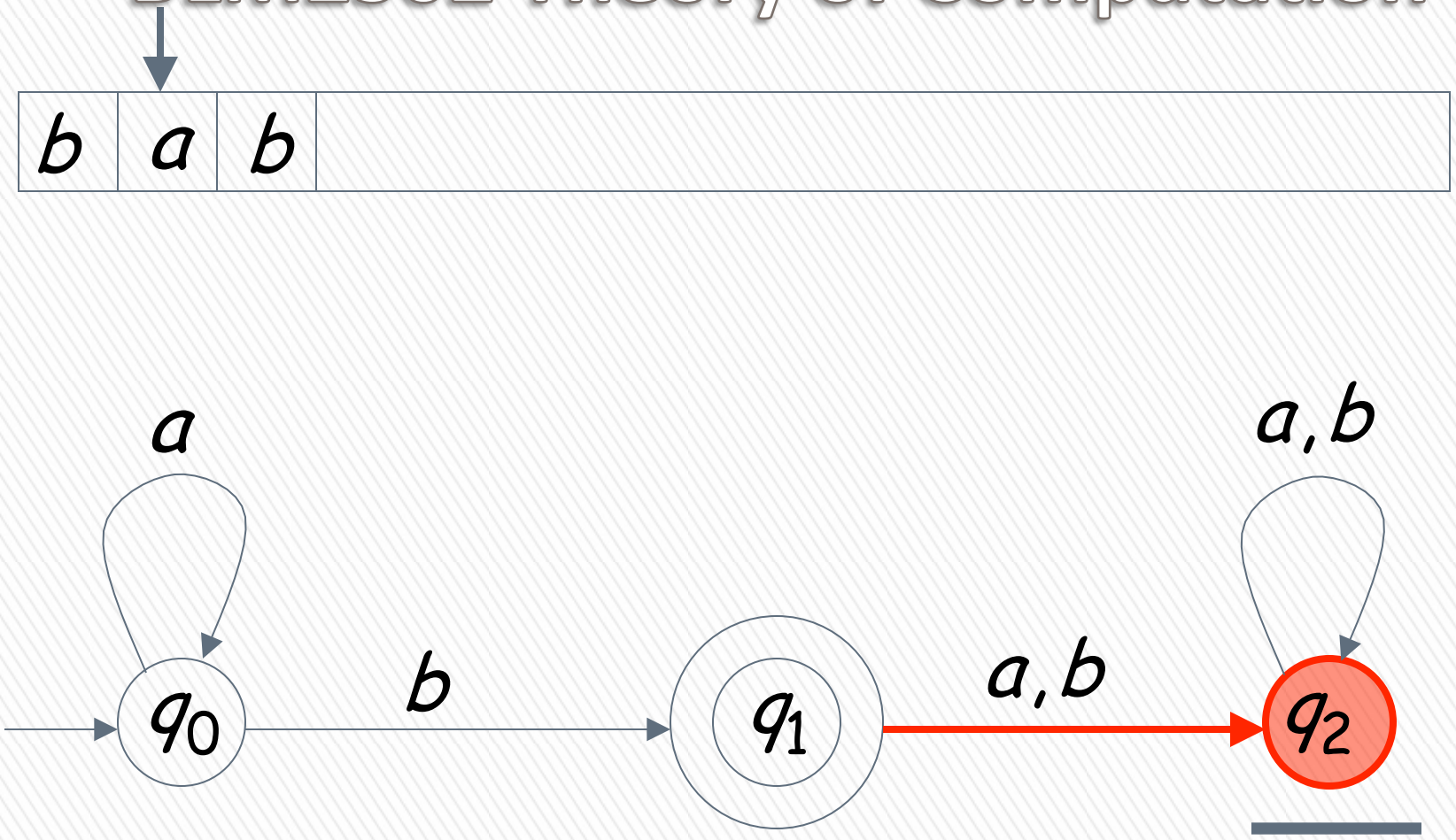


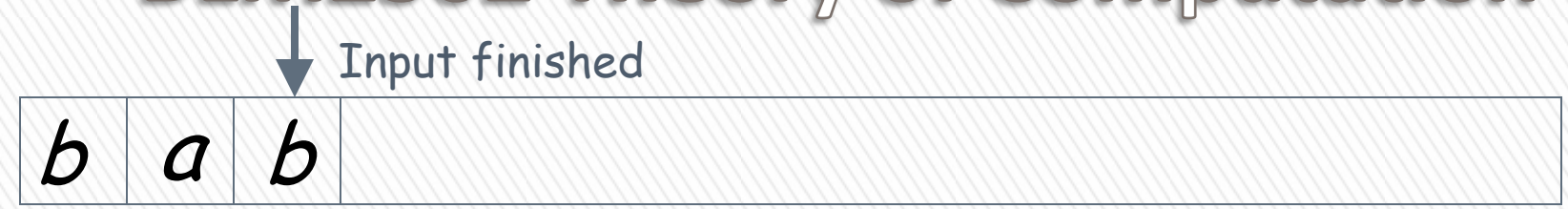
Input String

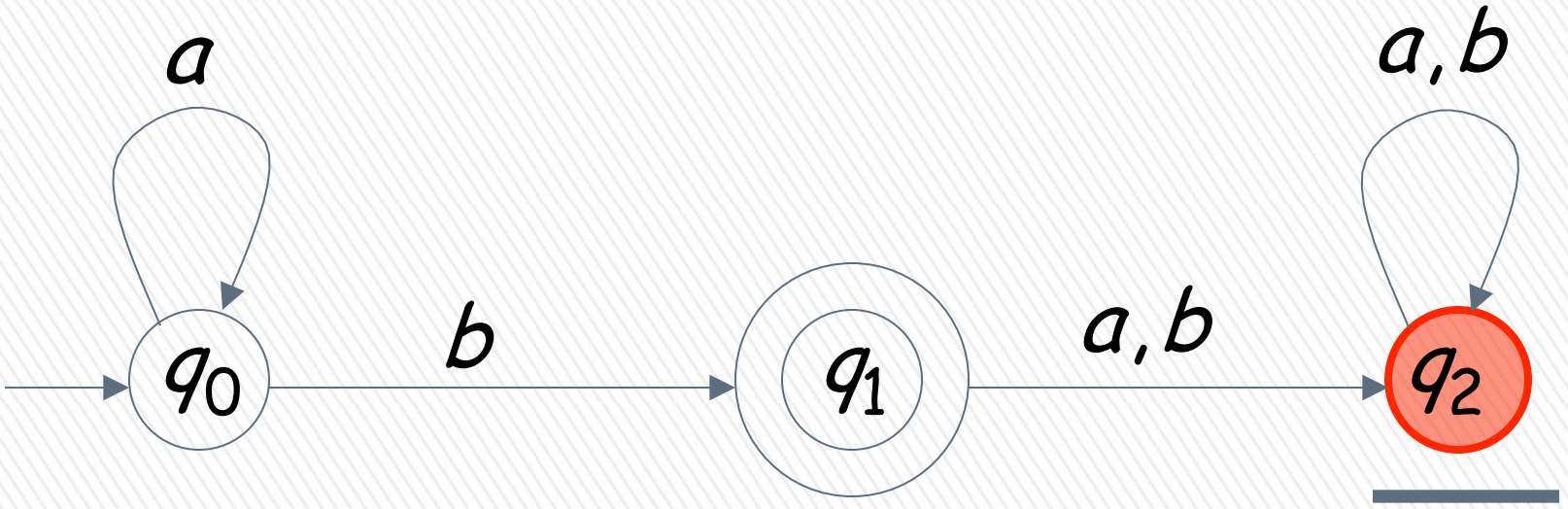


A rejection case


b
a
b
a
a, b
q0
b
q1
a, b
q2


b
a
b
a
a, b
q0
b
q1
a, b
q2

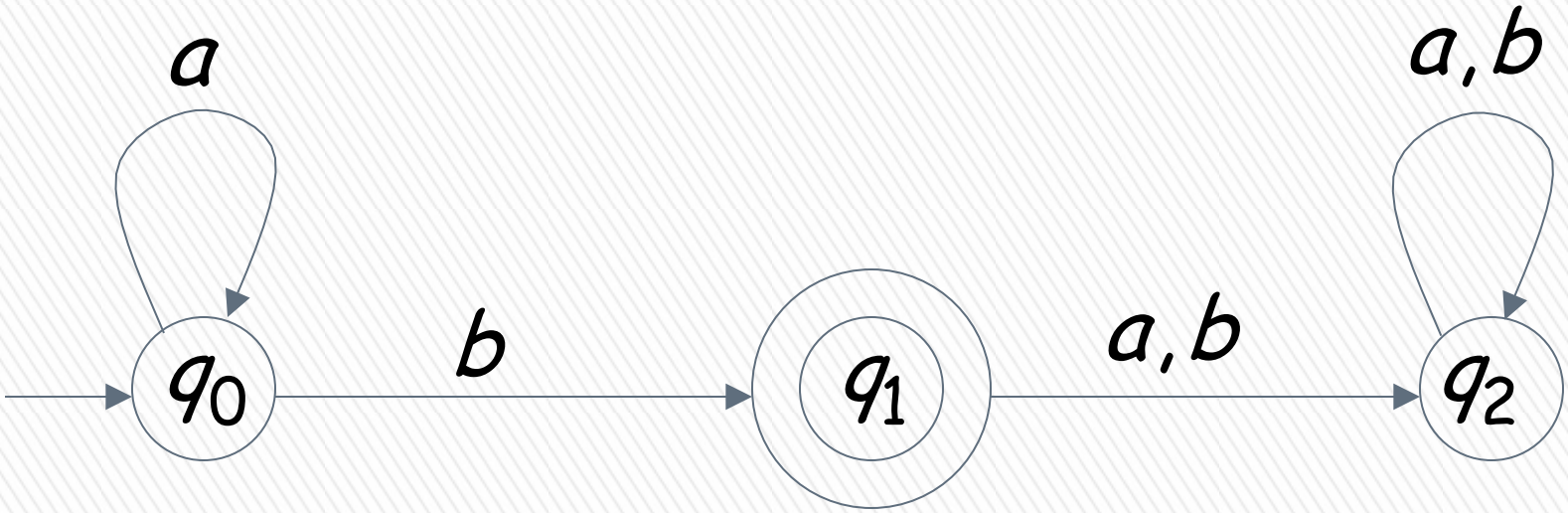

Input finished
b
a
b


a
a, b
b
a, b
q0
q1
q2
reject

BLM2502 Theory of Computation

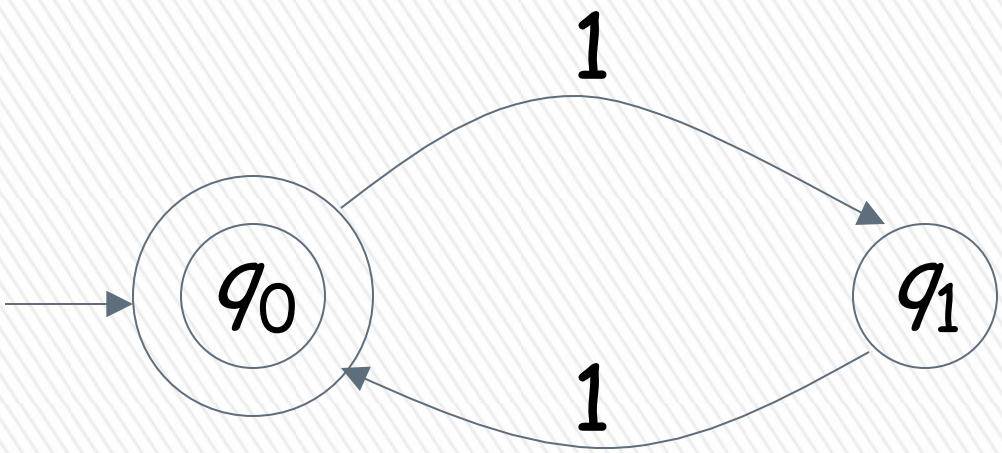
Language Accepted:

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



BLM2502 Theory of Computation

Alphabet: $\Sigma = \{1\}$



Language Accepted:

$$\text{EVEN} = \{x: x \in \Sigma^* \text{ and } |x| \text{ is even}\}$$
$$= \{\varepsilon, 11, 1111, 111111, \ldots\}$$

BLM2502 Theory of Computation

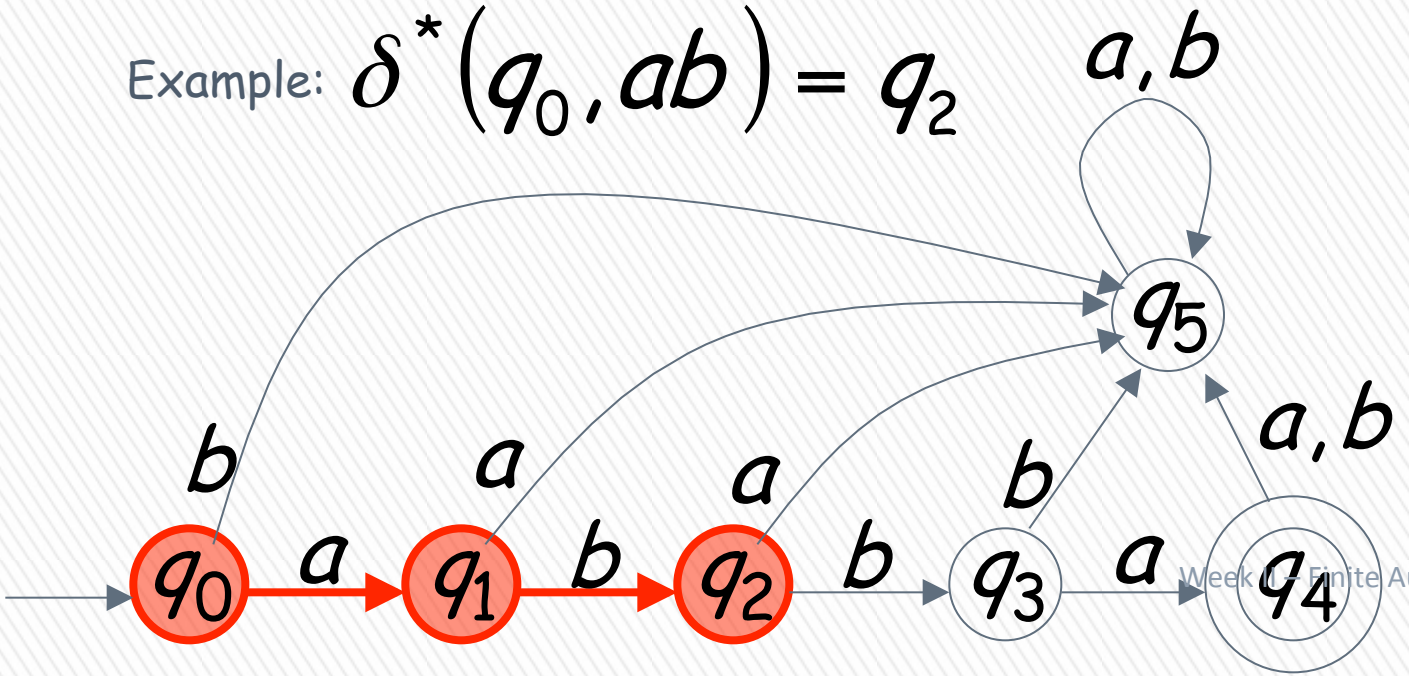
» Extended Transition Function

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

$\delta(q, w) \rightarrow q'$

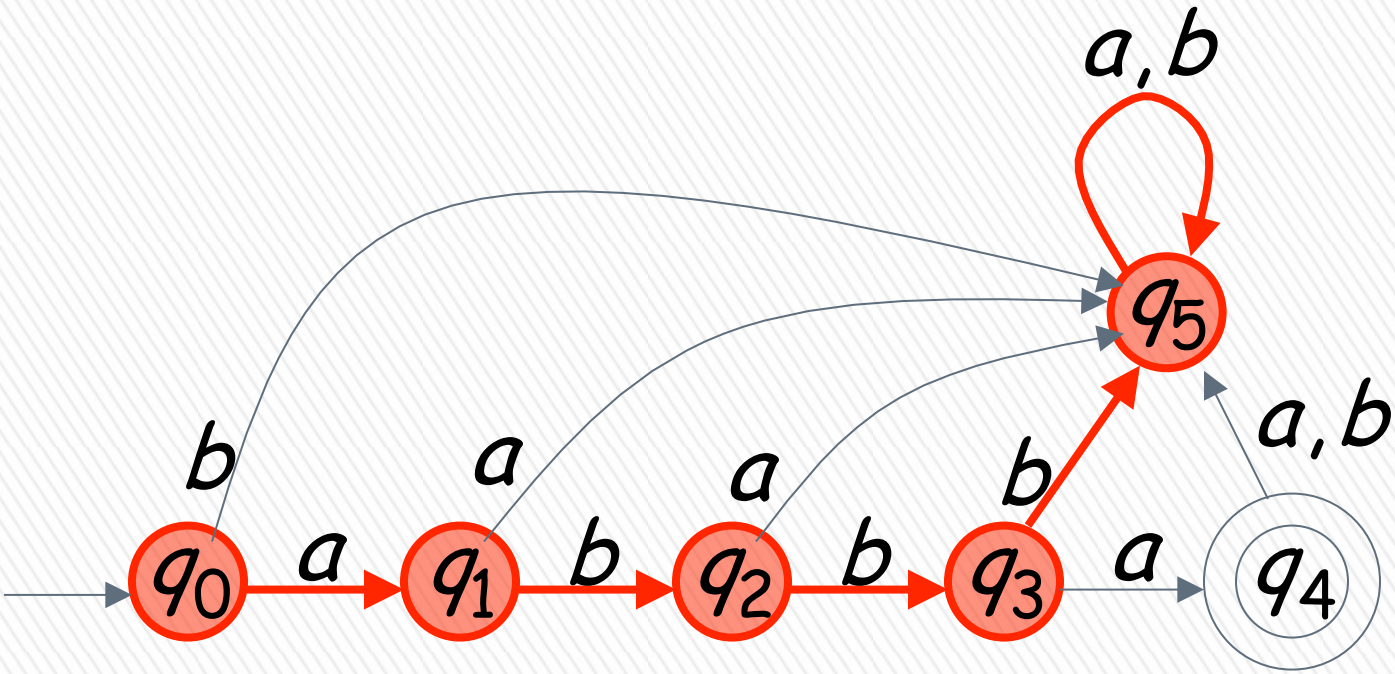
> Describes the resulting state after scanning string **w** from state **q**

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



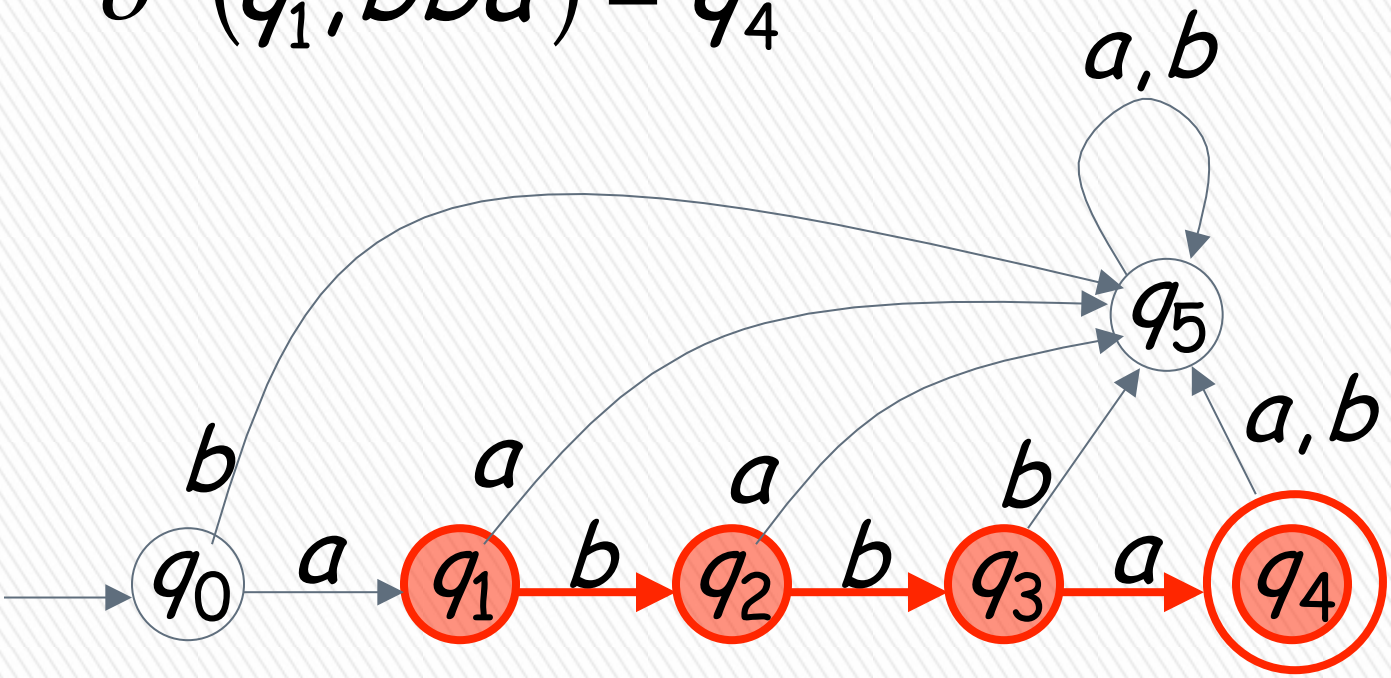
BLM2502 Theory of Computation

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



BLM2502 Theory of Computation

$$\delta^*(q_1, bba) = q_4$$



BLM2502 Theory of Computation

In general:

δ (q, w) →q' implies that there is a walk of transitions

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

states may be repeated

BLM2502 Theory of Computation

Language Accepted By DFA

The language accepted by an automaton M, is denoted as L(M) and contains all the strings accepted by M

We say that a language L' is accepted (or recognized) by the DFA M if

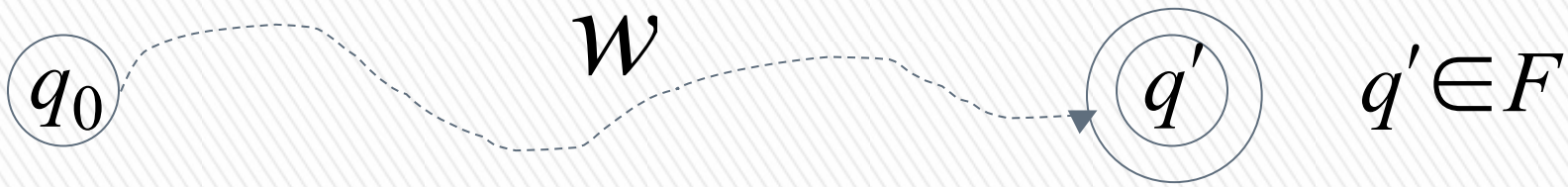
$$L(M) = L'$$

An automaton accepts one and only one language.
A language can ve accepted by a number of automata

BLM2502 Theory of Computation

- For a DFA **M** = (Q, Σ, δ, q_0, F)
- Language accepted by **M**:

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



BLM2502 Theory of Computation

» Language rejected by **M** :

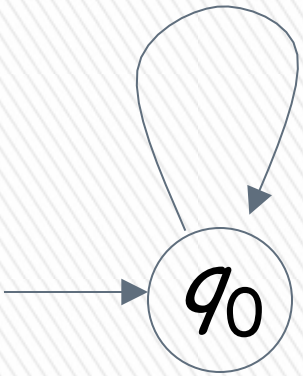
$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}$$

BLM2502 Theory of Computation

More DFA Examples

a, b

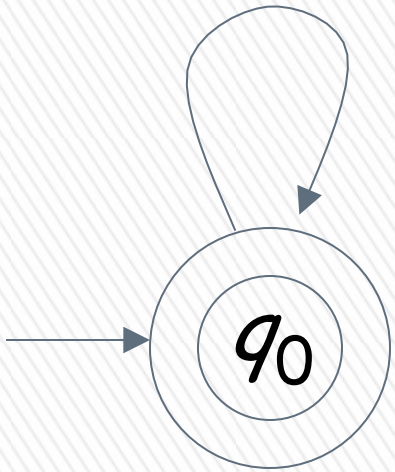
$\Sigma = \{a, b\}$



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

Empty language

a, b

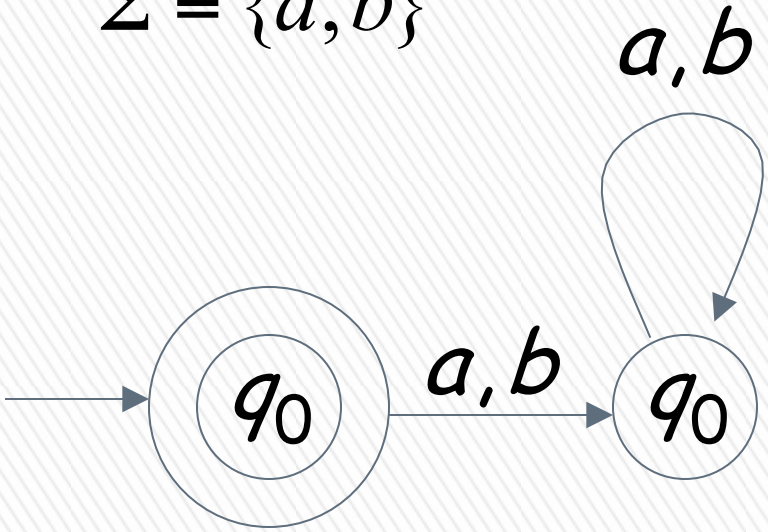


$L(M) = \Sigma^*$

All strings

BLM2502 Theory of Computation

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



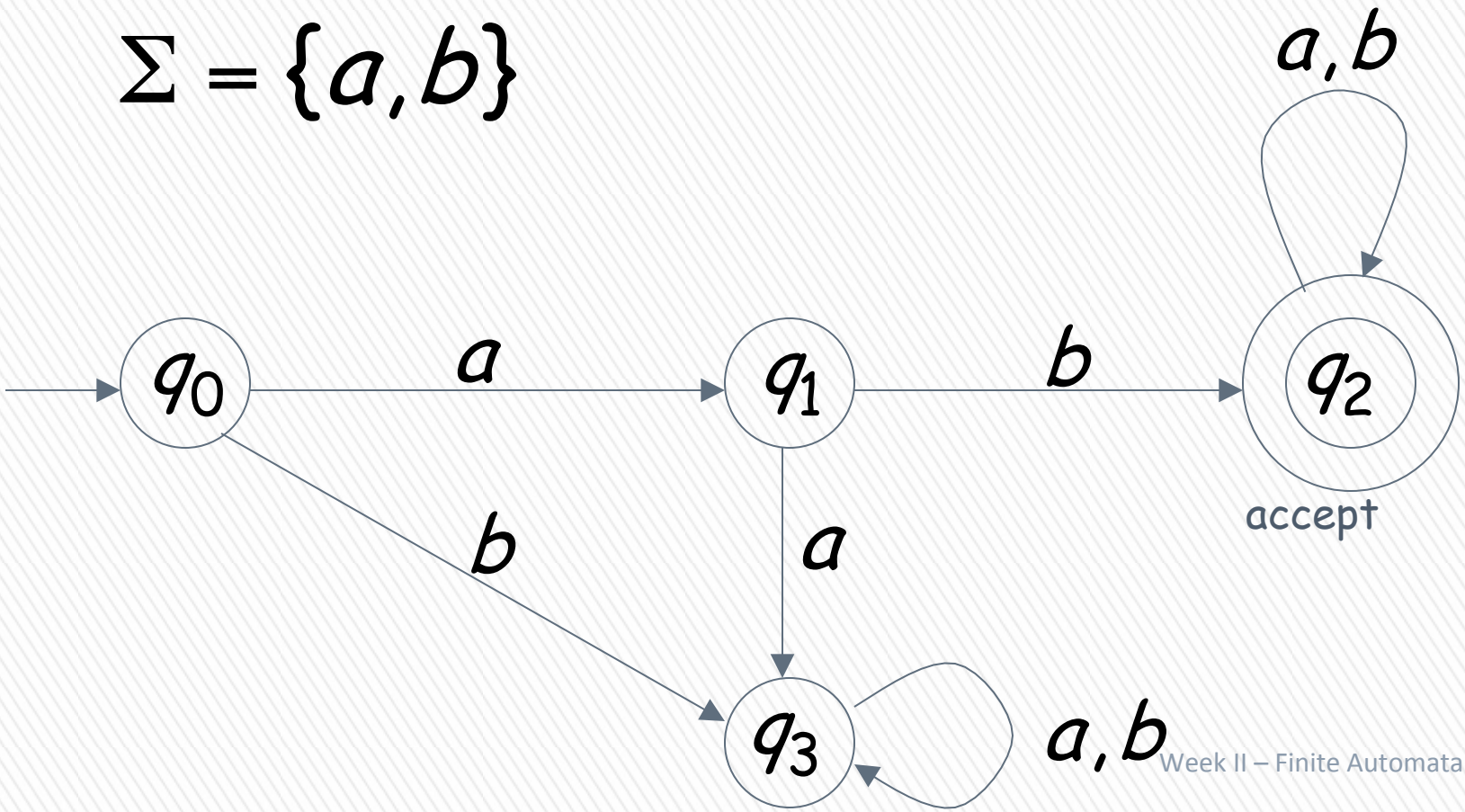
$$L(M) = \{\varepsilon\}$$

Language of the empty string

BLM2502 Theory of Computation

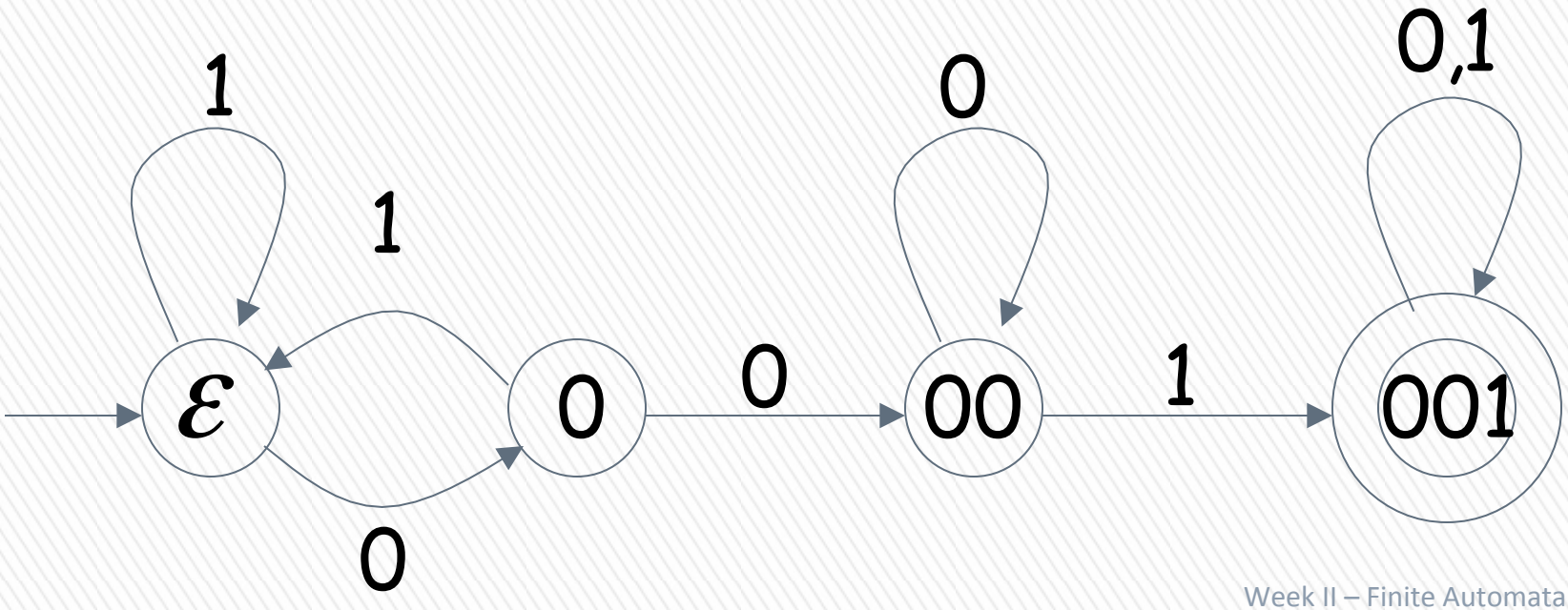
L(M) = { all strings with prefix ab}

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



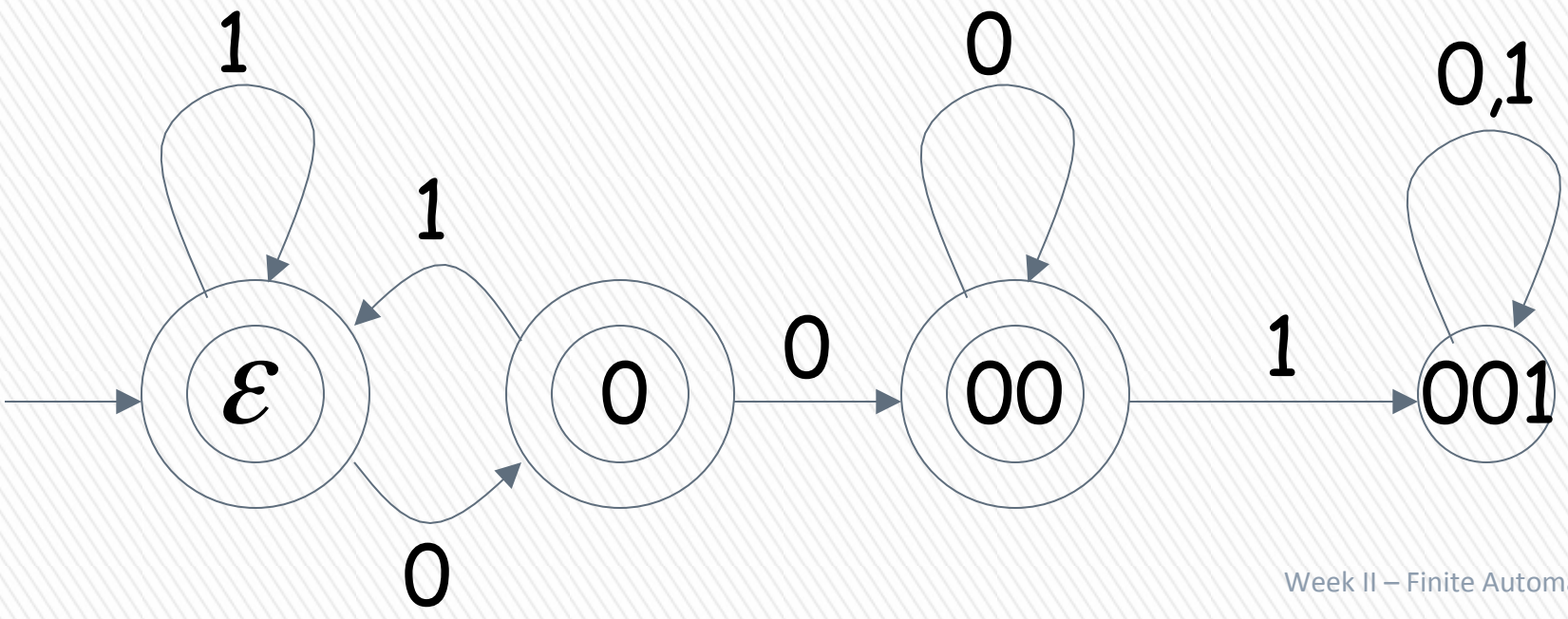
BLM2502 Theory of Computation

L(M) = { all binary strings containing substring 001 }



BLM2502 Theory of Computation

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



BLM2502 Theory of Computation

Regular Sets Examples

For each of the following parts, give an example satisfying the given conditions. Give a brief explanation for each of your examples.

(a) Give an example of a set S of strings such that $S^* = S^+$.

(b) Give an example of a set S of strings such that $S^* \neq S^+$.

(c) Give an example of a set S of strings such that $S = S^*$.

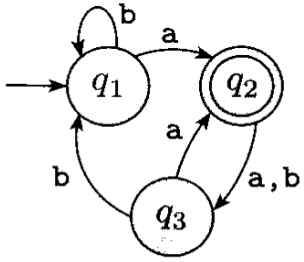
(d) Give an example of a set S of strings such that $S \neq S^*$.

(e) Give an example of a set S of strings such that S^* is finite.

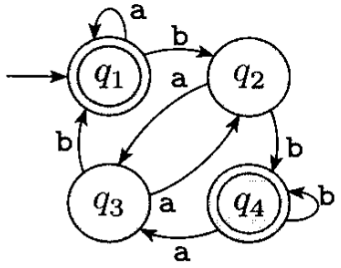
BLM2502 Theory of Computation

DFA

The following are the state diagrams of two DFAs, M_1 and M_2. Answer the following questions about these machines.



M_1



M_2

a. What is the start state of M_1?

b. What is the set of accept states of M_1?

c. What is the start state of M_2?

d. What is the set of accept states of M_2?

e. What sequence of states does M_1 go through on input aabb?

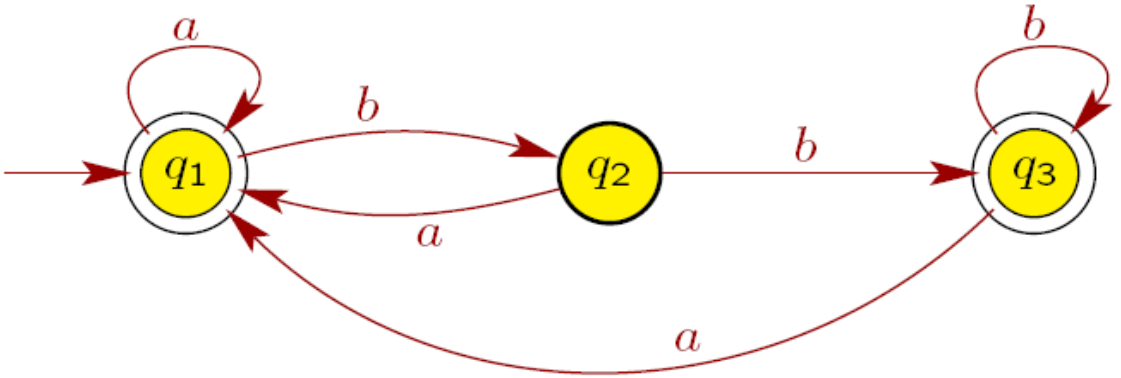
f. Does M_1 accept the string aabb?

g. Does M_2 accept the string ε?

BLM2502 Theory of Computation

DFA Examples

For the DFA M below, give its formal definition as a 5-tuple.



BLM2502 Theory of Computation

DFA Examples

For each of the following languages over the alphabet $\Sigma = \{a, b\}$, give a DFA that recognizes the language:

(a) $A = \{\varepsilon, b, ab\}$.

(b) For any string $w \in \Sigma^*$, let $n_a(w)$ denote the number of a's in w. For example, $n_a(abaaba) = 4$. Define the language

$$B = \{ w \in \Sigma^* \mid n_a(w) \bmod 3 = 1 \},$$

i.e., $w \in B$ if and only if the number of a's in w is $3k + 1$ for some $k \geq 0$.

(c) $C = \{ w \in \Sigma^* \mid w = saba \text{ for some string } s \in \Sigma^* \}$, i.e., C consists of strings that end in aba.

(d) $D = \overline{C}$, where C is the language in the previous part; i.e., D consists of strings that do not end in aba.

(e) $E = \{ w \in \Sigma^* \mid w \text{ begins with } b \text{ and ends with } a \}$.

(f) For any string $w \in \Sigma^*$, let $n_b(w)$ denote the number of b's in w. Define the language $F = \{ w \in \Sigma^* \mid n_a(w) \geq 2, n_b(w) \leq 1 \}$.

(g) $G = \{ w \in \Sigma^* \mid |w| \geq 2, \text{ second-to-last symbol of } w \text{ is } b \}$. If string $w = w_1 w_2 \cdots w_n$ where each $w_i \in \Sigma$, then the second-to-last symbol of w is w_{n-1}.

BLM2502 Theory of Computation

DFA Examples

- Construct DFA that accepts all strings with at least one a and exactly two b's.
- Construct DFA for the language L = $\{ab^n a^m; m > 1, n > 2\}$
- Find DFA for tha language L = $\{w : n_a(w) + 2n_b(w) \bmod 3 < 2\}$