



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 Languages**

» 4 **Finite Automata**

» 5 Deterministic and Nondeterministic Finite Automata

» 6 Epsilon Transition, Equivalence of Automata

» 7 Pumping Theorem

»

» 9 Context Free Grammars

» 10 Parse Tree, Ambiguity,

» 11 Pumping Theorem

» 13 Turing Machines, Recognition and Computation, Church-Turing Hypothesis

» 14 Turing Machines, Recognition and Computation, Church-Turing Hypothesis

» 15 Review



Regular Expressions

BLM2502 Theory of Computation

Regular Languages

» 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 $\Sigma_2 = \{a, b, c, \dots, z\}$, 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 $|w|$, 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 similar to 0 in a number system.

> If w has length n , we can write

$$w = w_1 w_2 \dots w_n \text{ where each } w_i \in \Sigma.$$

The reverse of w , written as w^R , is the string obtained by writing w in the opposite order (i.e., $w_n w_{n-1} \dots w_1$).

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 $x_1 \dots x_m y_1 \dots y_n$. To concatenate a string with itself many times we use the superscript notation:

$$x^3 = xxx,$$

$$x^n = xx \dots x \text{ (n times)}$$

To indicate all possible recurrences, 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**: $A^* = \{x_1 x_2 \dots x_k \mid k > 0 \text{ and each } x_i \in A\}$.

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:

Strings:

BLM2502 Theory of Computation

Decimal numbers

Alphabet:

Strings:

Binary numbers

Alphabet:

Strings:

BLM2502 Theory of Computation

Unary numbers

Alphabet:

Strings:

Decimal equivalent:

BLM2502 Theory of Computation

» Examples on String Operations

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

eg, $w = xyyxy$, $v = xxxyyy$

Concenation:

$$wv = a_1a_2...a_nb_1b_2...b_m$$

eg, $xyyxyxxxxyyy$

Reverse:

$$w^R = a_na_{n-1}...a_1$$

eg, $yxyyx$

BLM2502 Theory of Computation

» Examples on String Operations

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

Length:

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

$$\text{eg, } |yxyyx| = 5; |xxxyyy| = 6$$

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

$$\text{eg, } |yxyyxxxxyyy| = 11 \text{ (recall example above)}$$

Empty String:

$$|\varepsilon| = 0$$

BLM2502 Theory of Computation

» Examples on String Operations

$$w = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m \quad a, b \in \{x, y\}$$

Substring:

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

$a_1, a_1 a_2, a_1 a_2 a_3 \dots$ (these are prefix substrings)

$a_2, a_2 a_3, a_2 a_3 a_4, \dots$

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

special cases:

ε is prefix and suffix of each string

any string is prefix and suffix of itself

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:

$$\varepsilon w = w\varepsilon = w$$

BLM2502 Theory of Computation

» Examples on String Operations

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

Self Concenation:

$$ww = w^2, www = w^3, \text{ etc}$$

$$w^0 = \varepsilon$$

eg, $w = abba$;

$$(abba)^0 = \varepsilon$$

$$(abba)^1 = abba$$

$$(abba)^2 = abbaabba$$

$$(abba)^3 = abbaabbaabba$$

Kleene Star:

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

BLM2502 Theory of Computation

» The * operation:

The set of all possible strings from the alphabet

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

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

» The + operation:

The set of all possible strings from the alphabet excluding ϵ

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

$$\Sigma^+ = \Sigma^* - \{\epsilon\} = \{a, b, aa, ab, ba, bb, aaa, aab, \dots\}$$

BLM2502 Theory of Computation

» Language:

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

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

Examples:

$$L_1 = \{\epsilon\}$$

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

$$L_3 = \{\epsilon, a, aa, aaa, aaaa, \dots\}$$

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

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

$$\begin{aligned} \text{PRIME_NUMBERS} &= \{x \mid x \in \Sigma^* \text{ and } x \text{ is prime}\} \\ &= \{2, 3, 5, 7, 11, \dots\} \end{aligned}$$

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

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

BLM2502 Theory of Computation

» Unary Addition

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

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

$111 + 11 = 11111 \in \text{ADDITION}$

$111 + 111 = 1111 \notin \text{ADDITION}$

» Squaring

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

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

$111 \# 111111111 \in \text{SQUARES}$

$1111 \# 11111111 \notin \text{SQUARES}$

BLM2502 Theory of Computation

» Operations On Languages

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

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

$$B = \{ab, bb\}$$

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

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

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

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

Note that :

$$\emptyset = \{\} \neq \{\epsilon\}$$

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

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

$$|\epsilon| = 0 \text{ * This is string length}$$



Finite Automata

BLM2502 Theory of Computation

»

Input Tape

String

```
graph TD; String[String] --> FA[Finite Automaton]; FA --> Output["Accept or Reject"];
```

Finite
Automaton

Output

“Accept”
or
“Reject”

BLM2502 Theory of Computation

Finite Automata

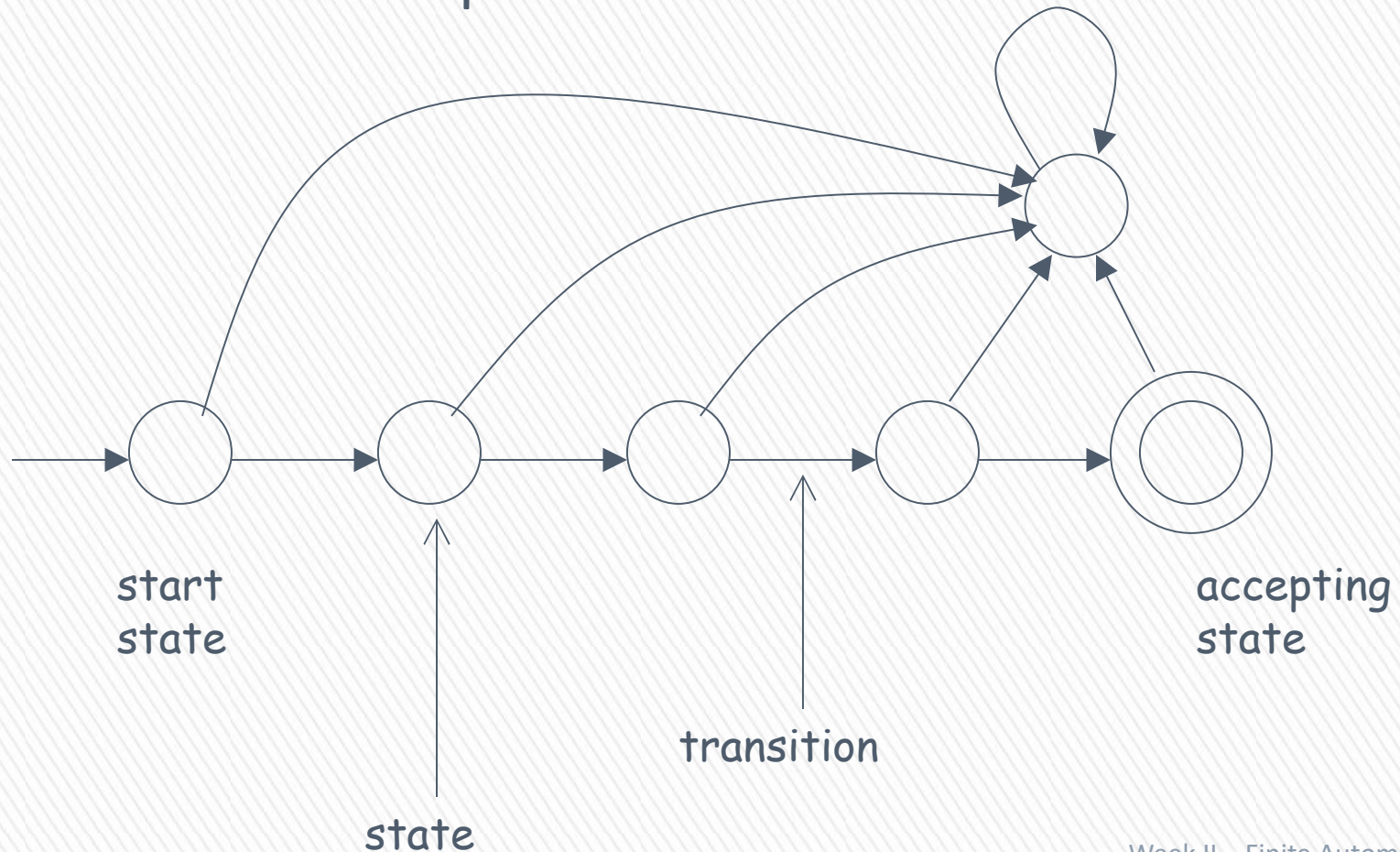
» A finite automaton is a 5-tuple:

$(Q, \Sigma, \delta, q_0, F)$ where;

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

BLM2502 Theory of Computation

Transition Graph

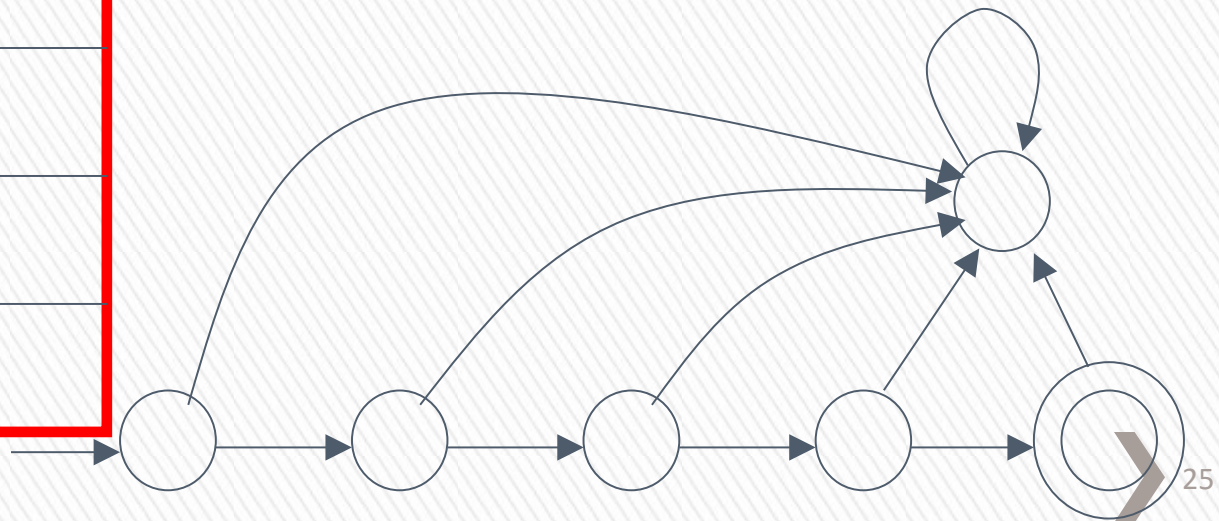


BLM2502 Theory of Computation

symbols

states

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:
 - » $A = \{w : w \text{ is a binary string containing an odd number of } 1\text{s}\}.$

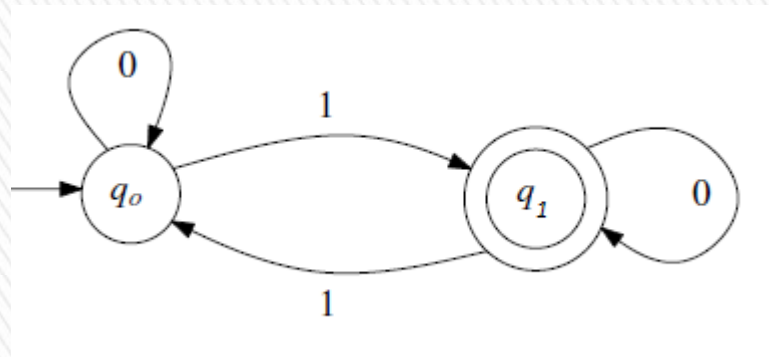
BLM2502 Theory of Computation

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 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

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

$$Q = \{q_0, q_1\}$$

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

$$q_0 \in Q$$

$$F = \{q_1\}$$

δ :

$$(q_0, 0) \rightarrow q_0$$

$$(q_0, 1) \rightarrow q_1$$

$$(q_1, 0) \rightarrow q_1$$

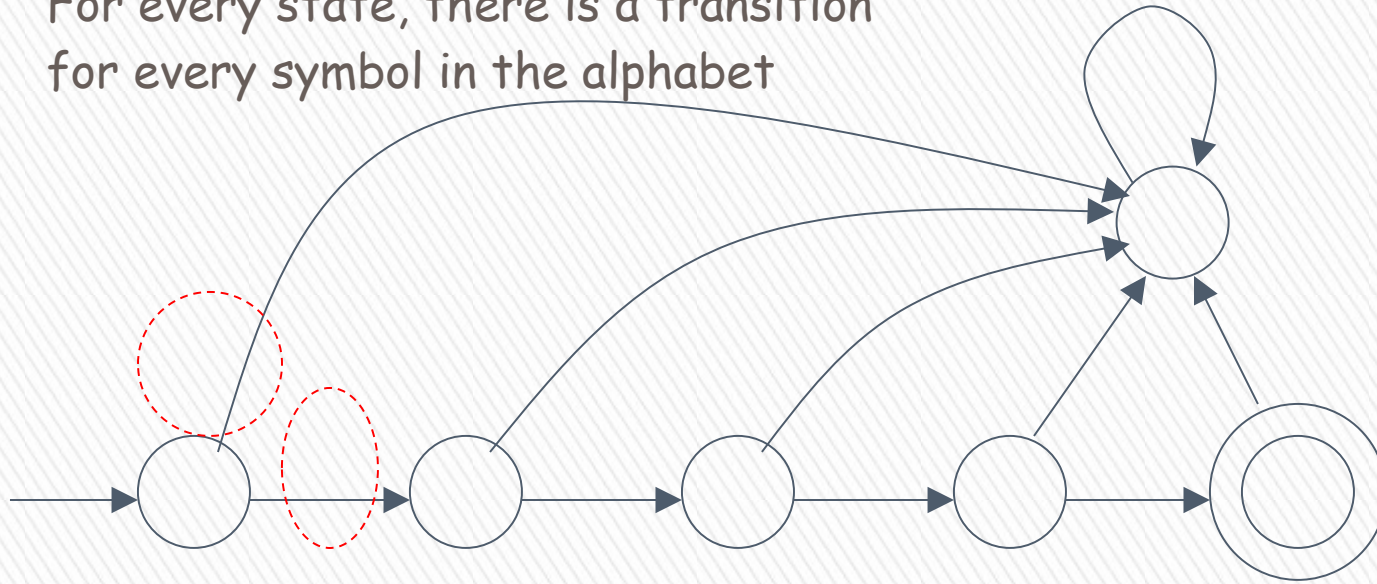
$$(q_1, 1) \rightarrow q_0$$

δ :	0	1
q_0	q_0	q_1
q_1	q_1	q_0

BLM2502 Theory of Computation

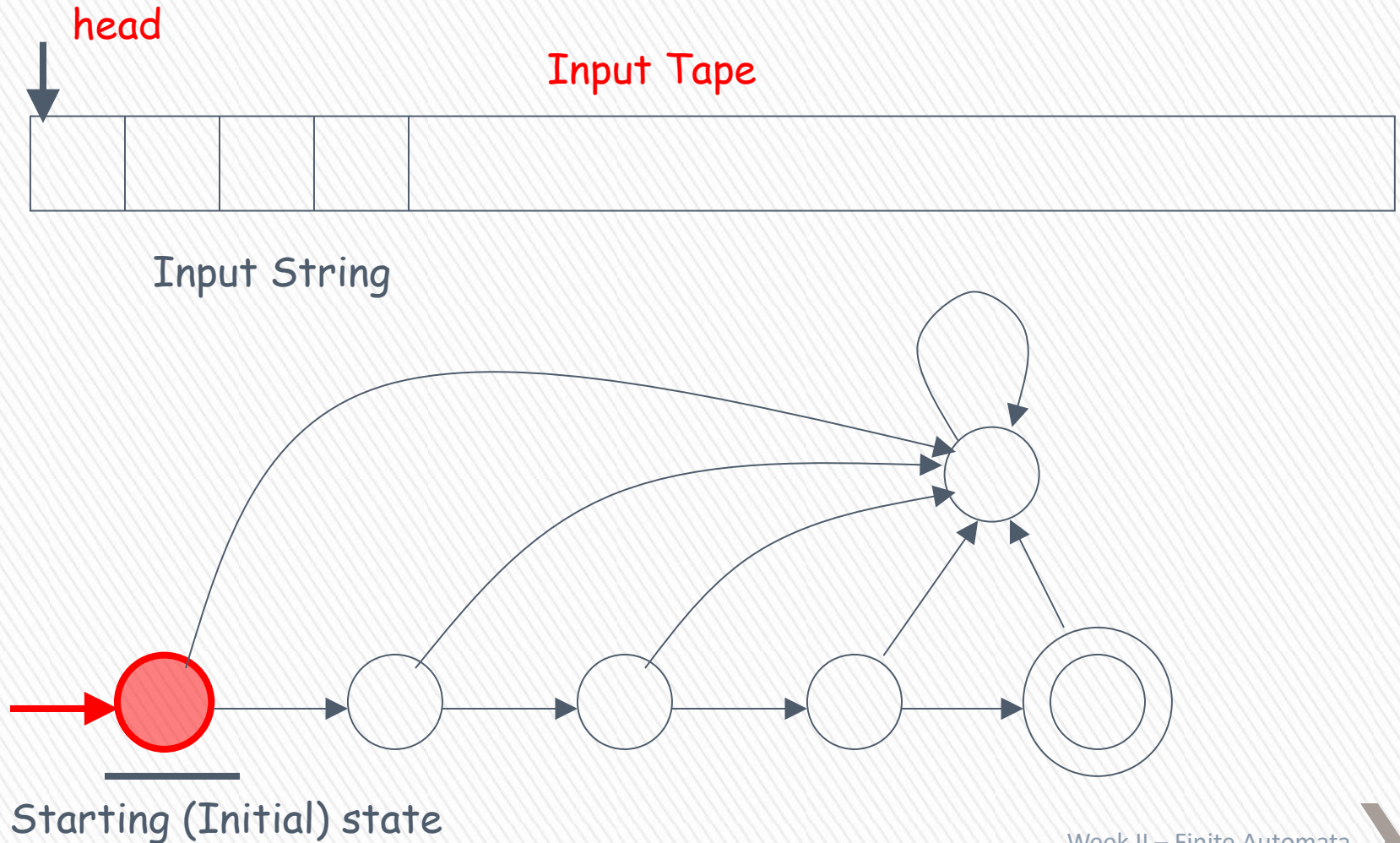
» DFA - Deterministic Finite Automata

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

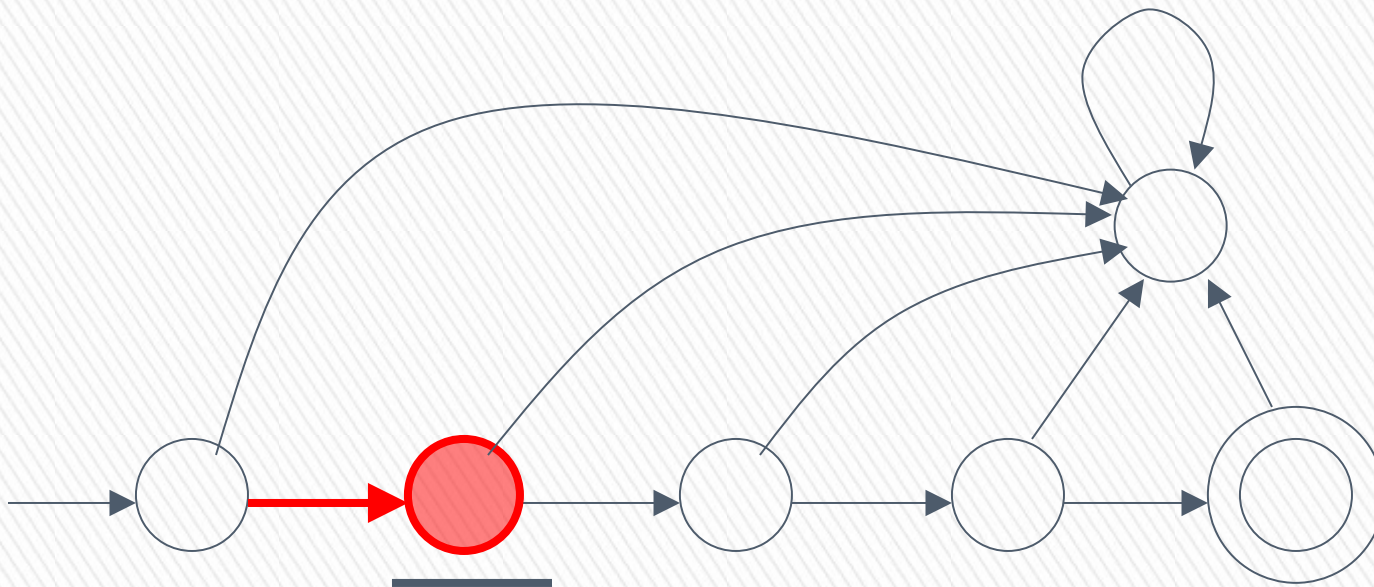


Alphabet $\Sigma = \{a, b\}$

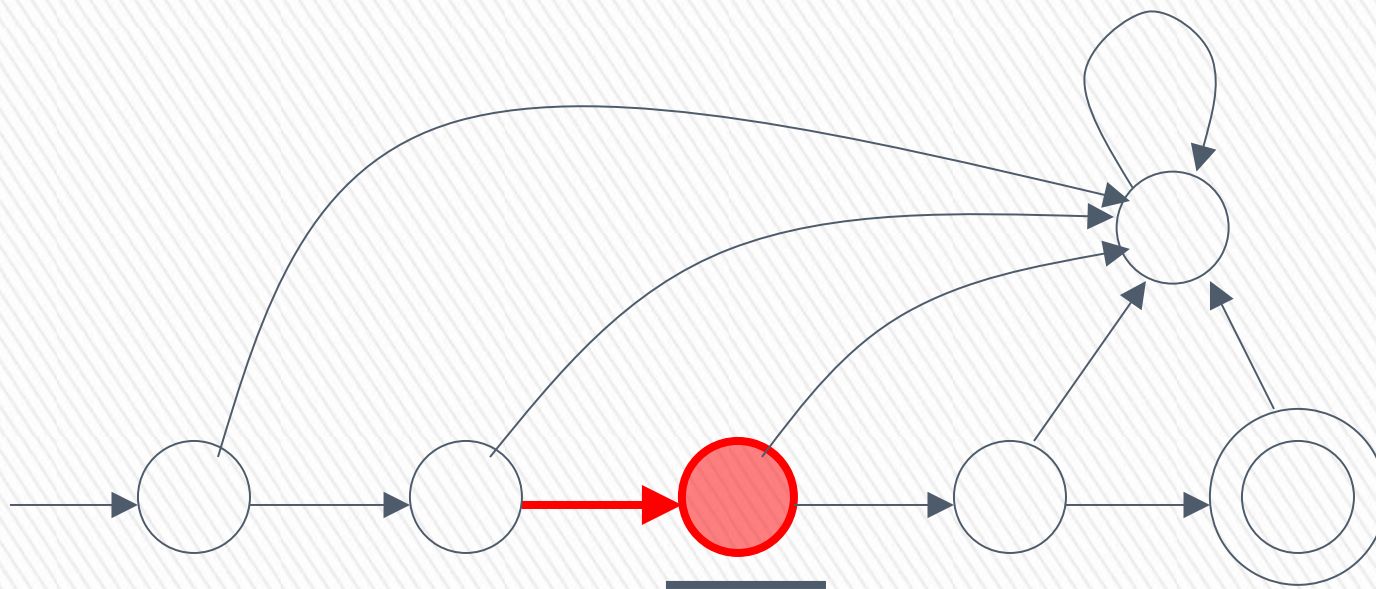
BLM2502 Theory of Computation



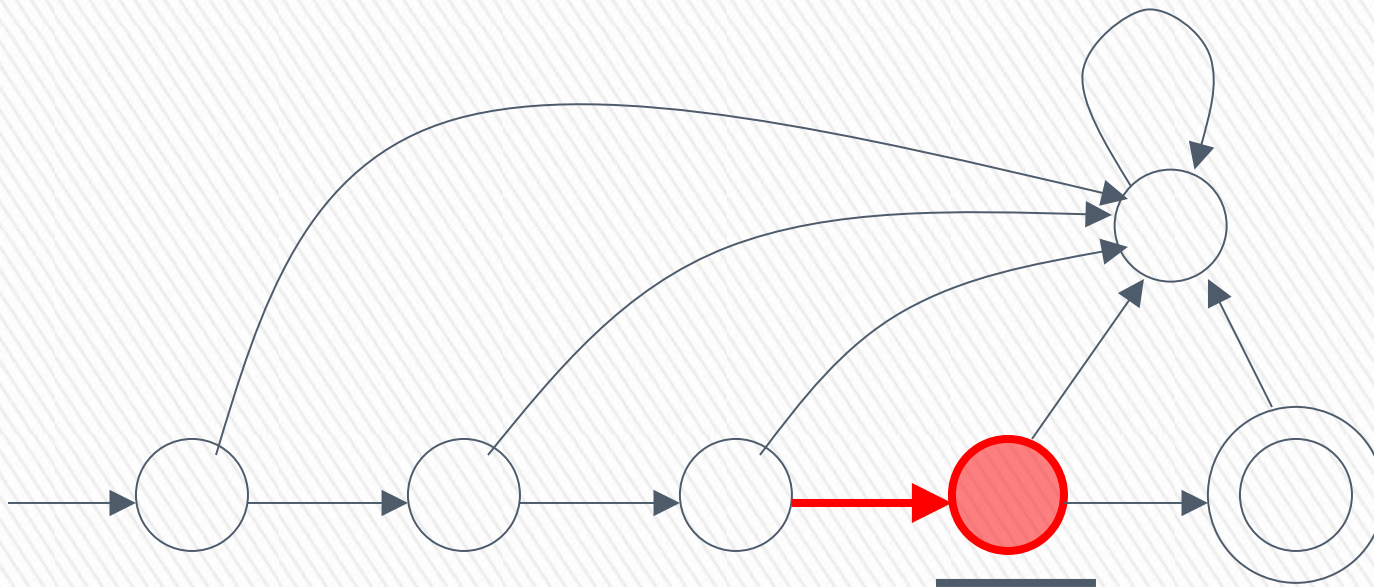
BLM2502 Theory of Computation



BLM2502 Theory of Computation

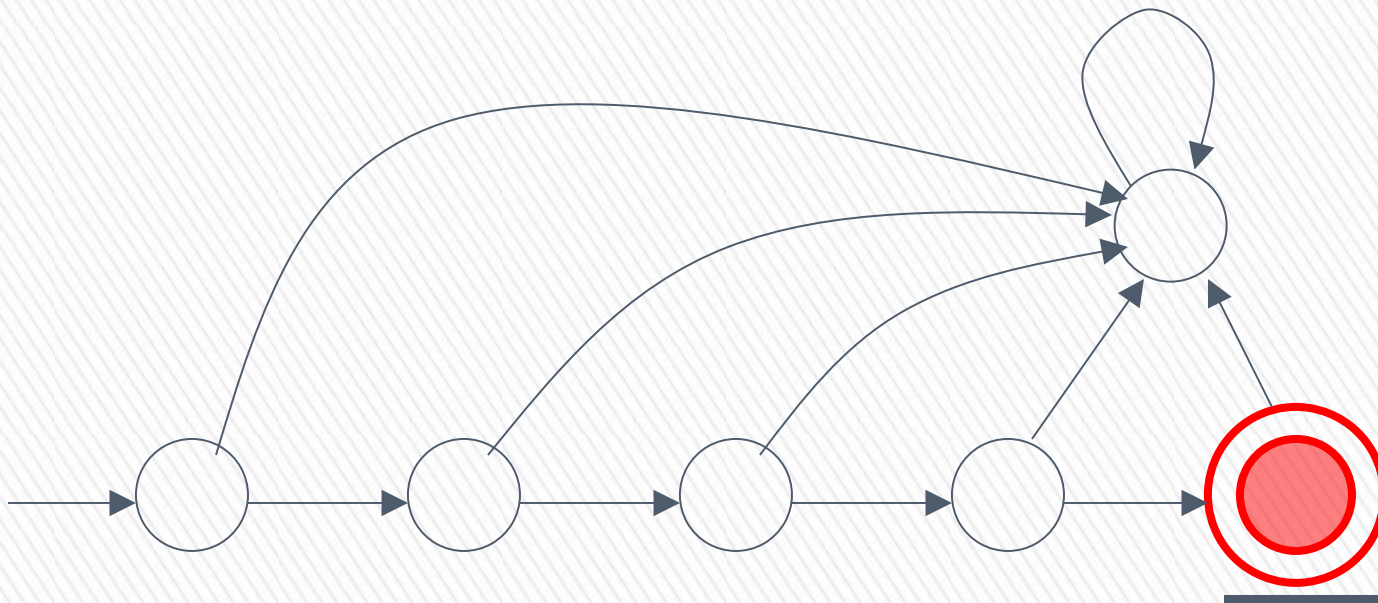


BLM2502 Theory of Computation

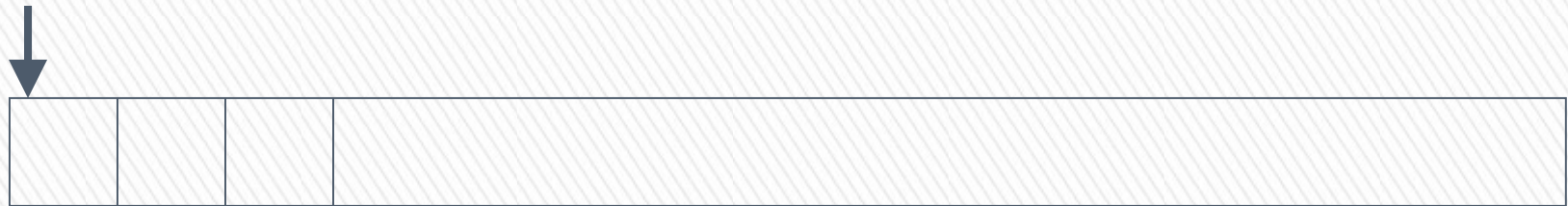


BLM2502 Theory of Computation

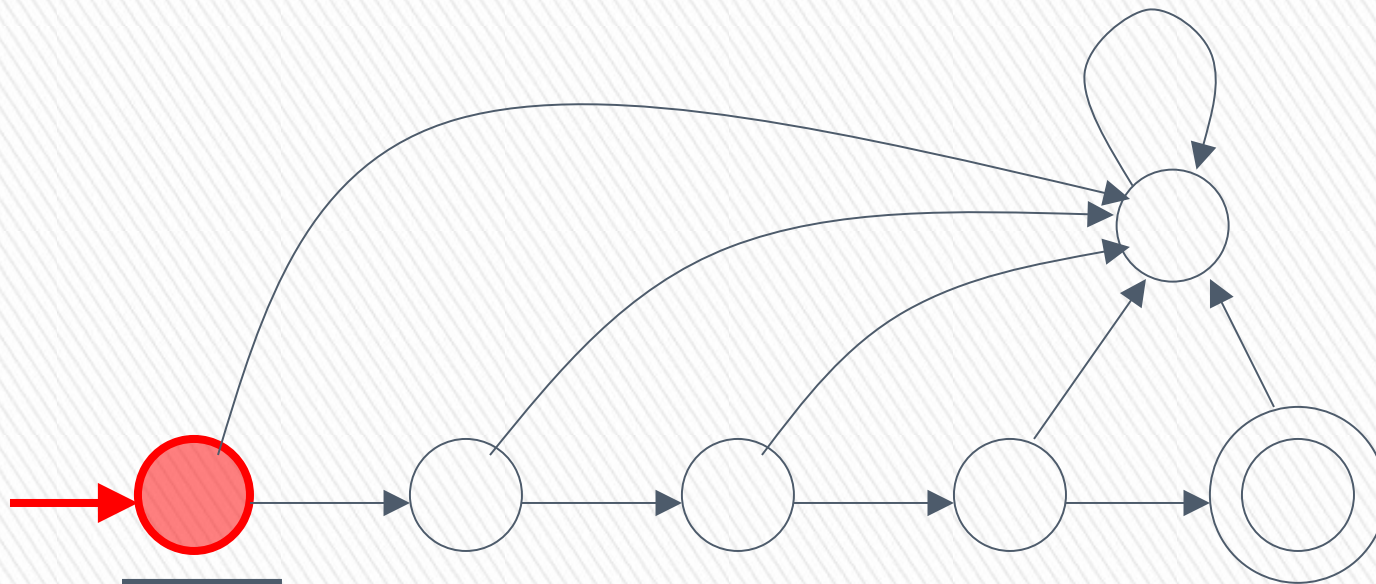
↓ Input finished



BLM2502 Theory of Computation

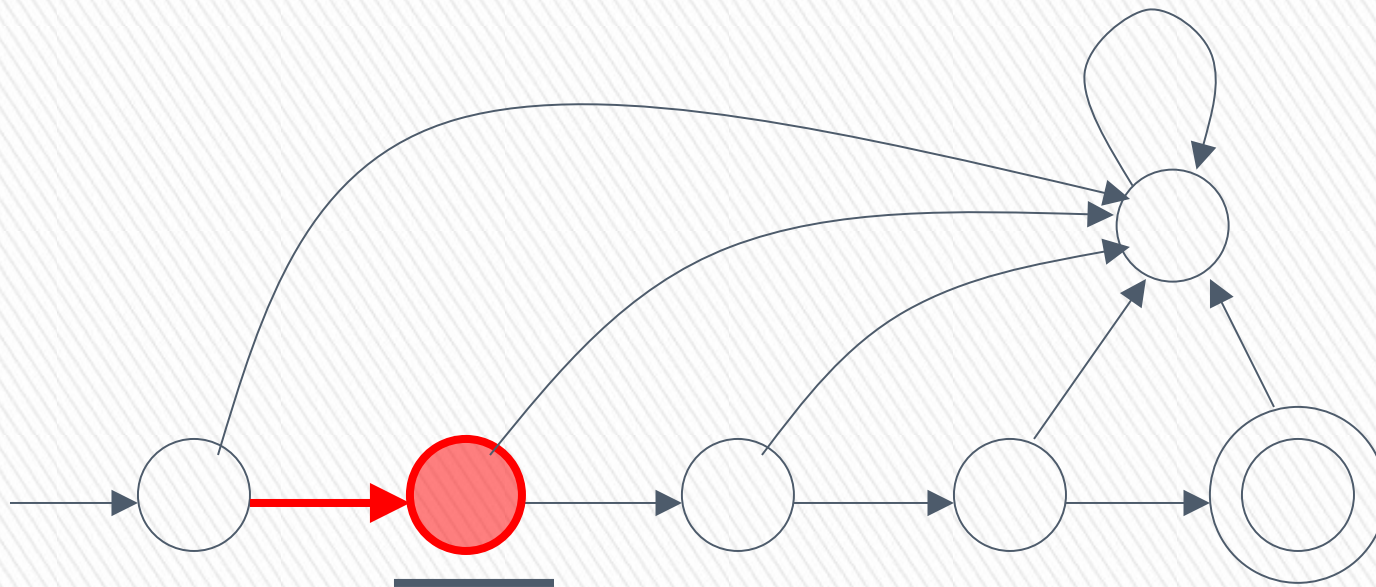
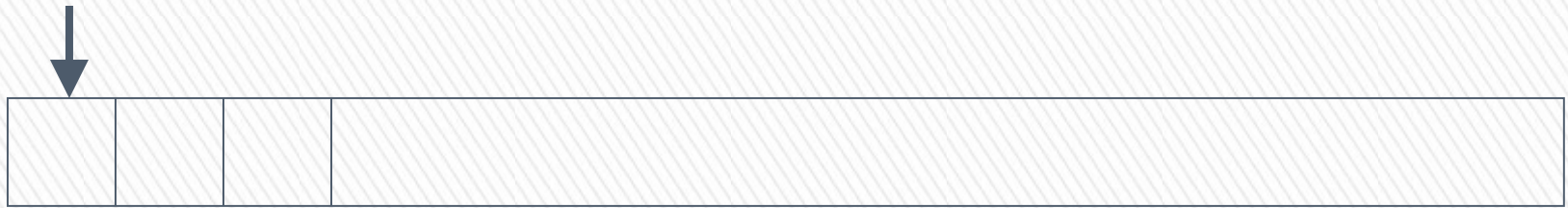


Input String

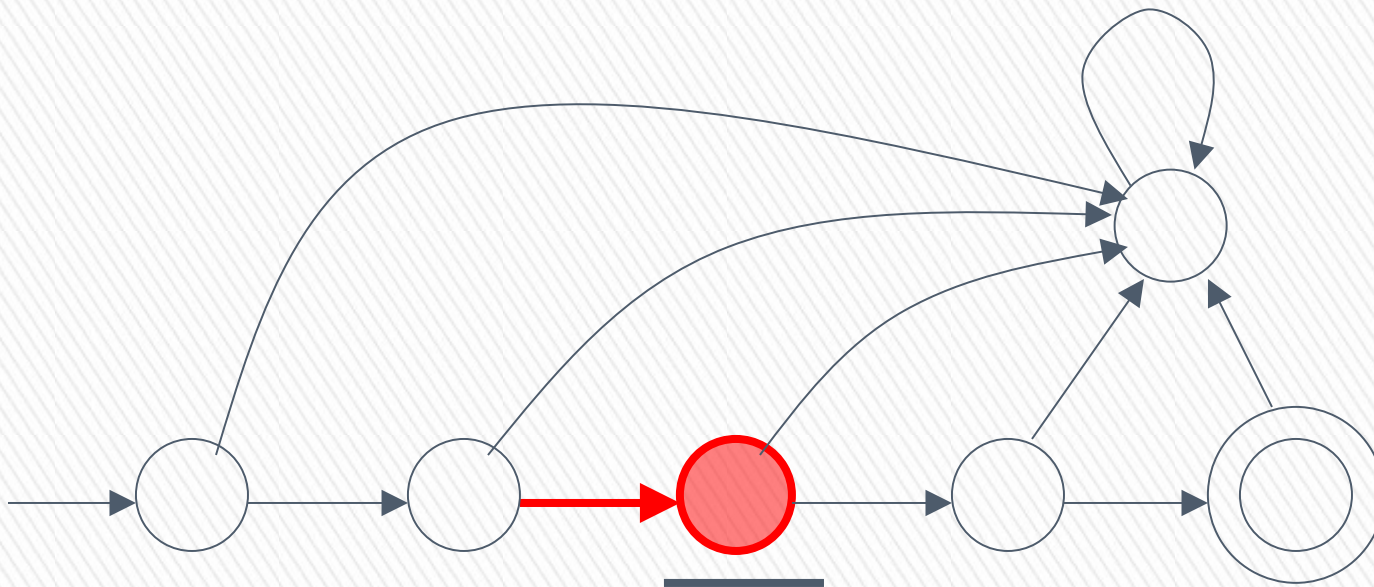


A Rejection Case

BLM2502 Theory of Computation

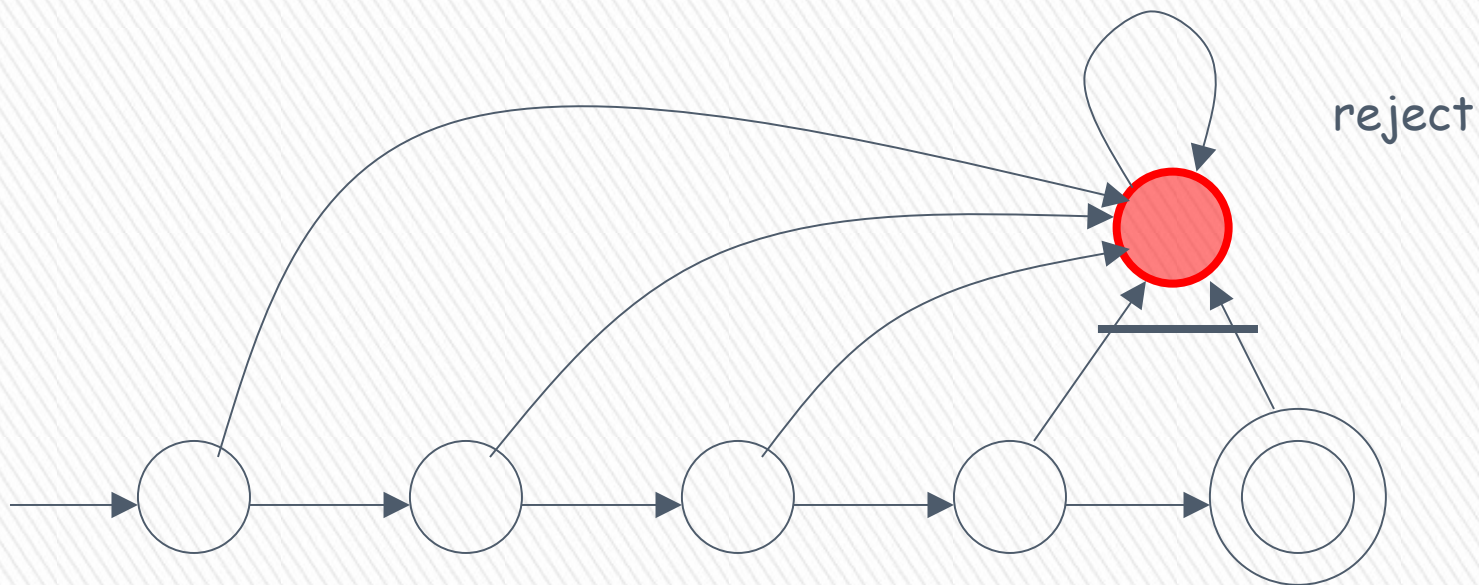


BLM2502 Theory of Computation



BLM2502 Theory of Computation

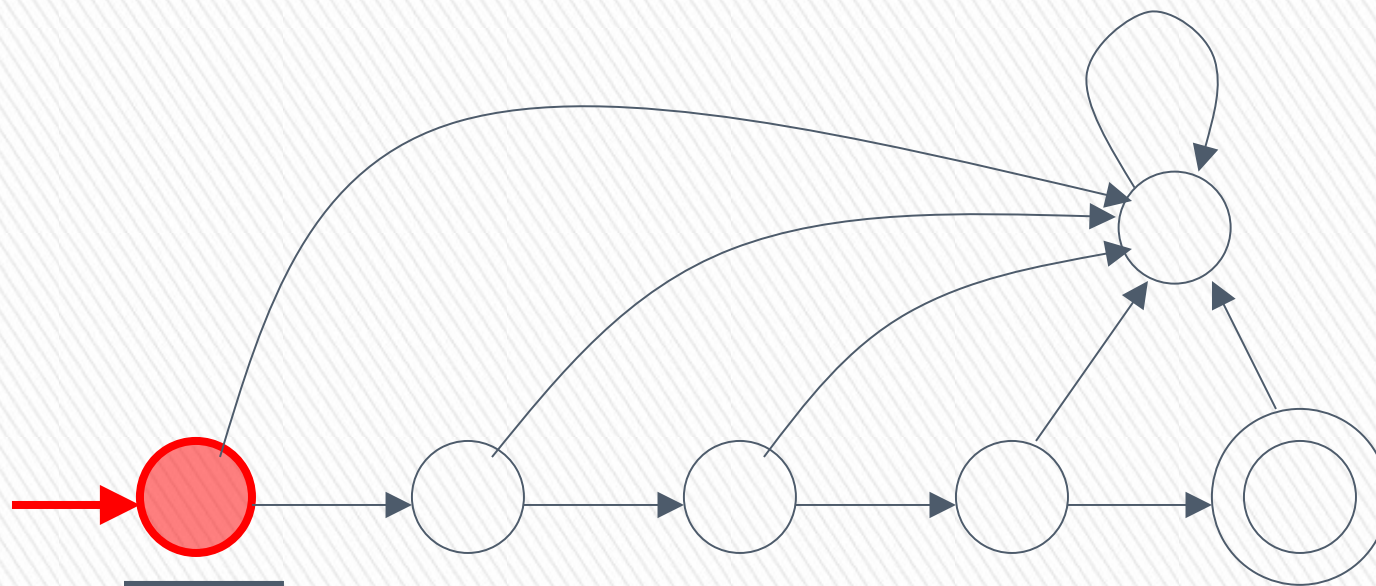
↓ Input finished



BLM2502 Theory of Computation



Input Finished

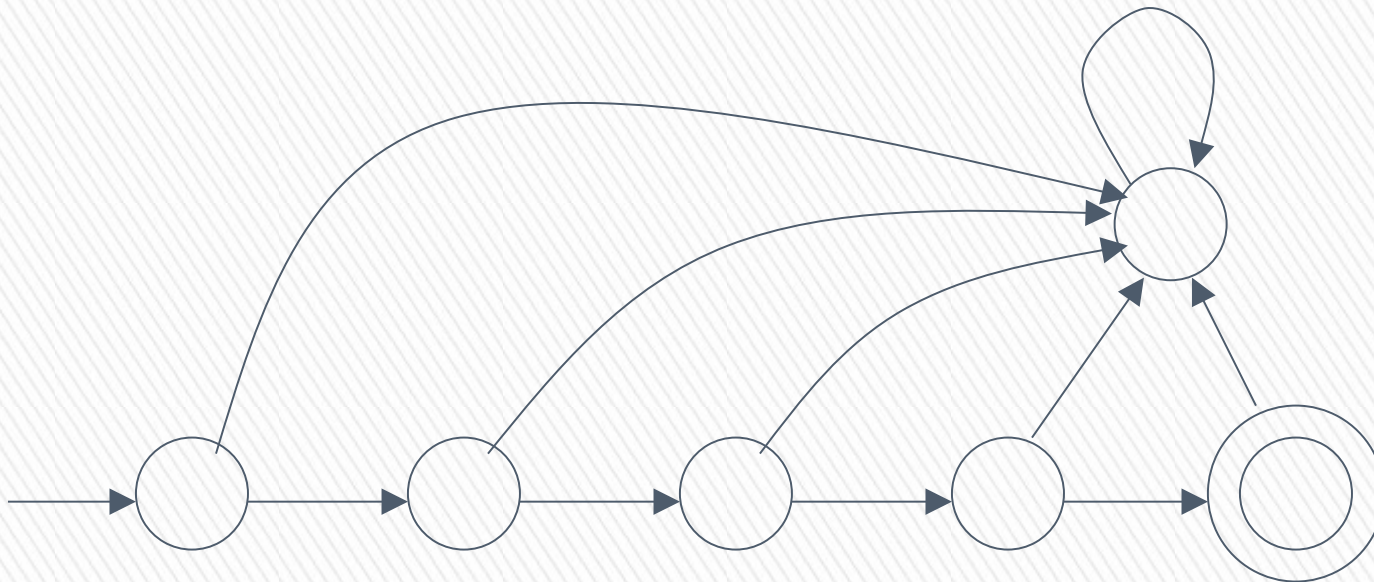


reject

Another Rejection Case

BLM2502 Theory of Computation

Language Accepted:



BLM2502 Theory of Computation

To accept a string:

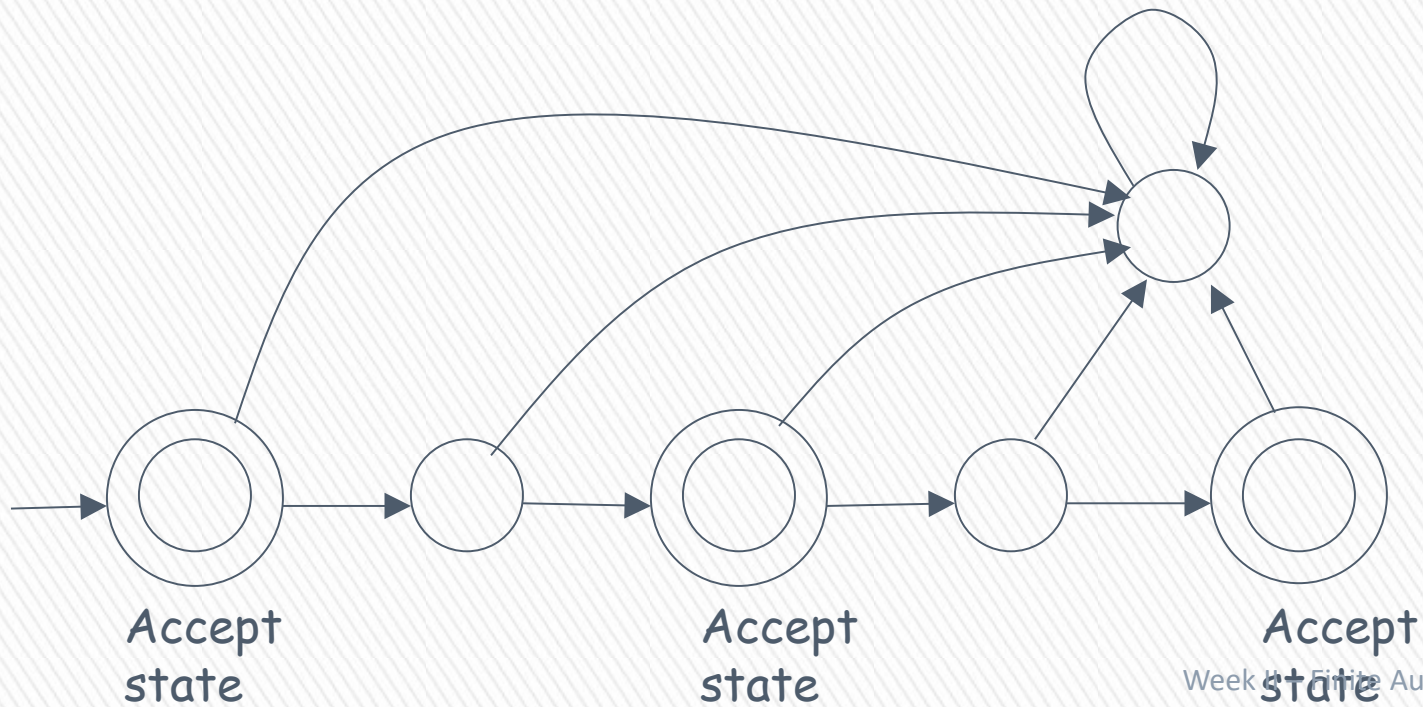
- all the input string is scanned and
- the last state is accepting

To reject a string:

- all the input string is scanned and
- the last state is non-accepting

BLM2502 Theory of Computation

$L = \{\epsilon, ab, abba\}$



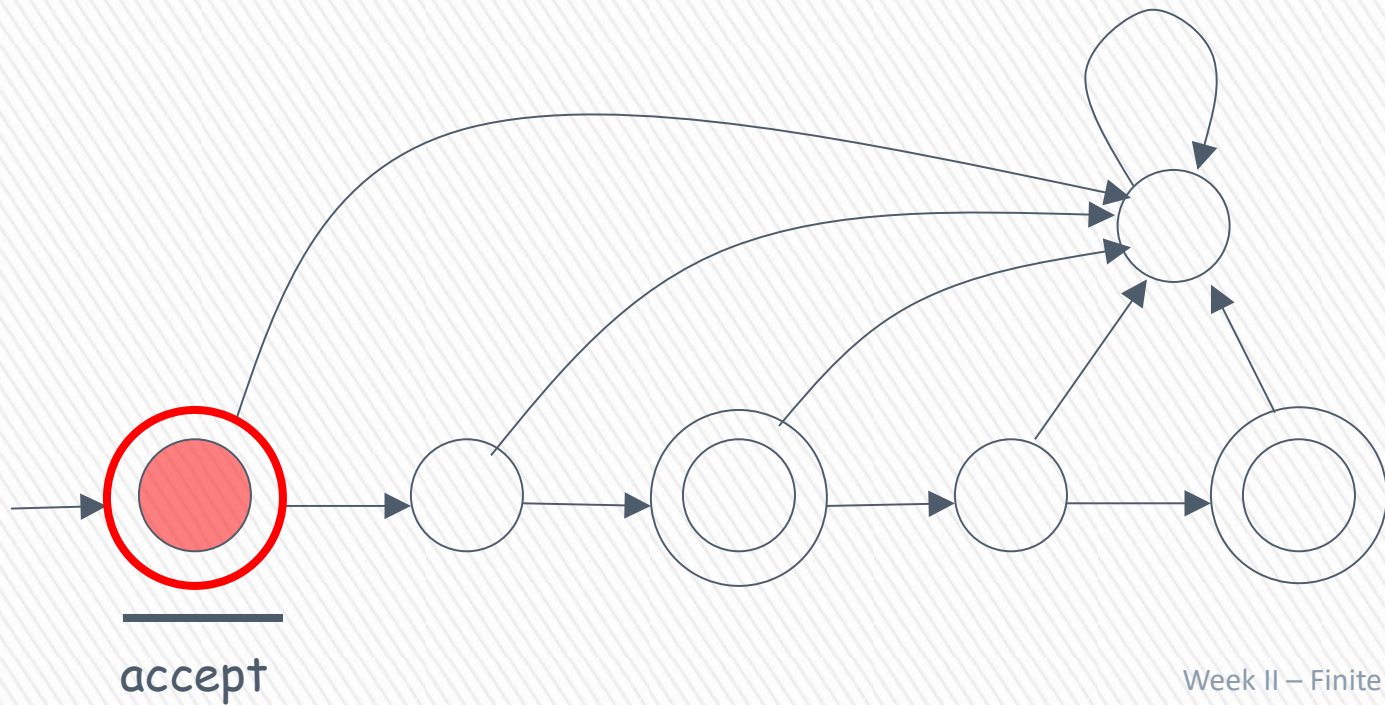
BLM2502 Theory of Computation



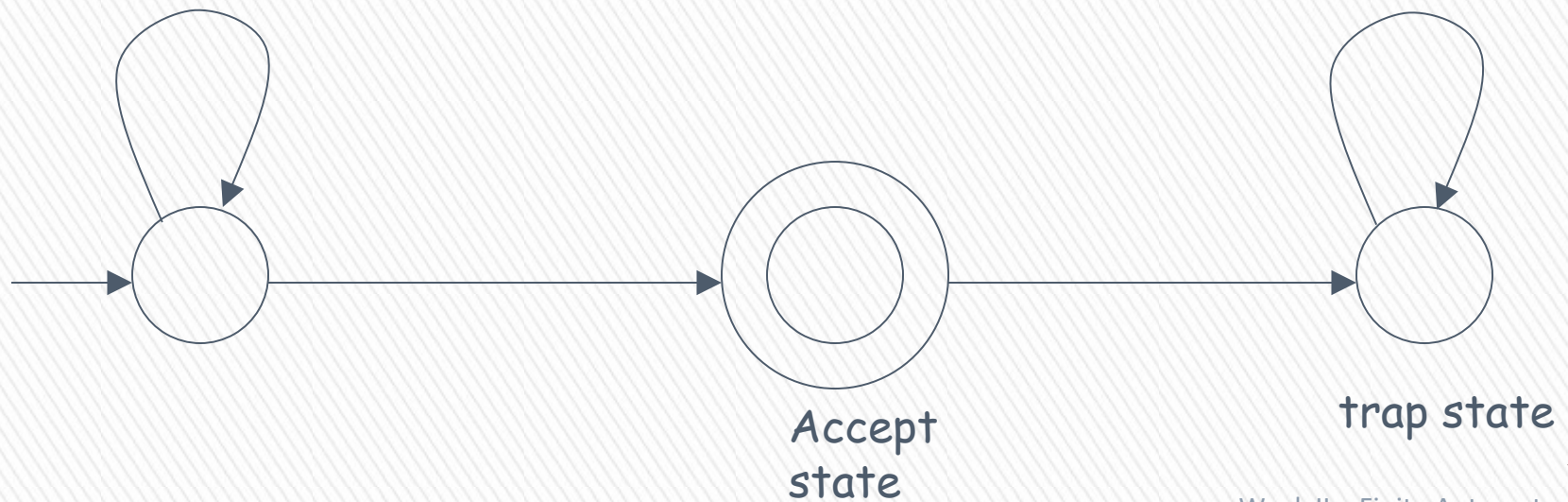
Empty Tape



Input Finished



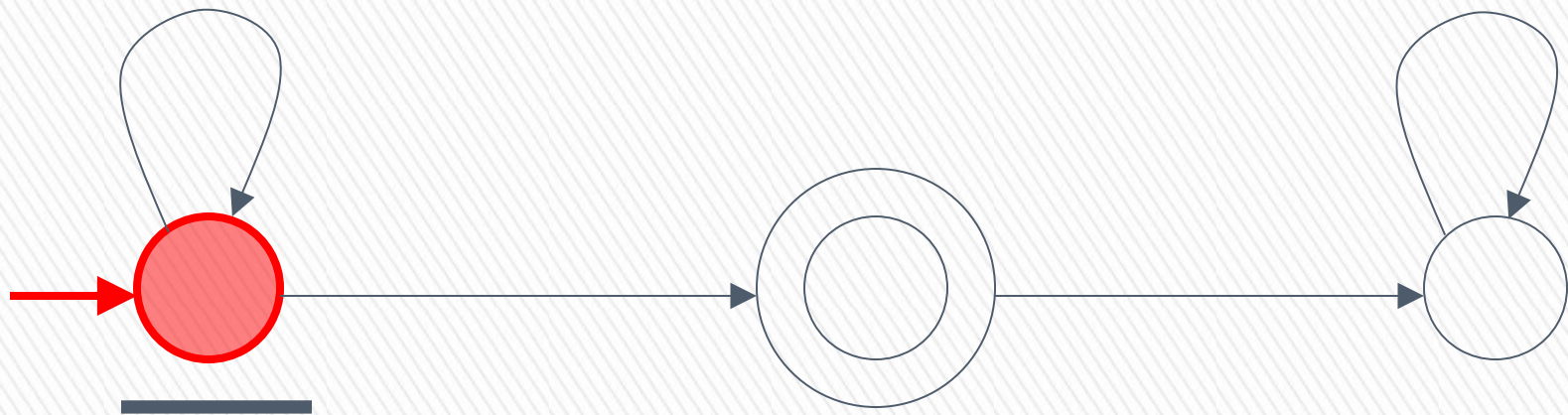
BLM2502 Theory of Computation



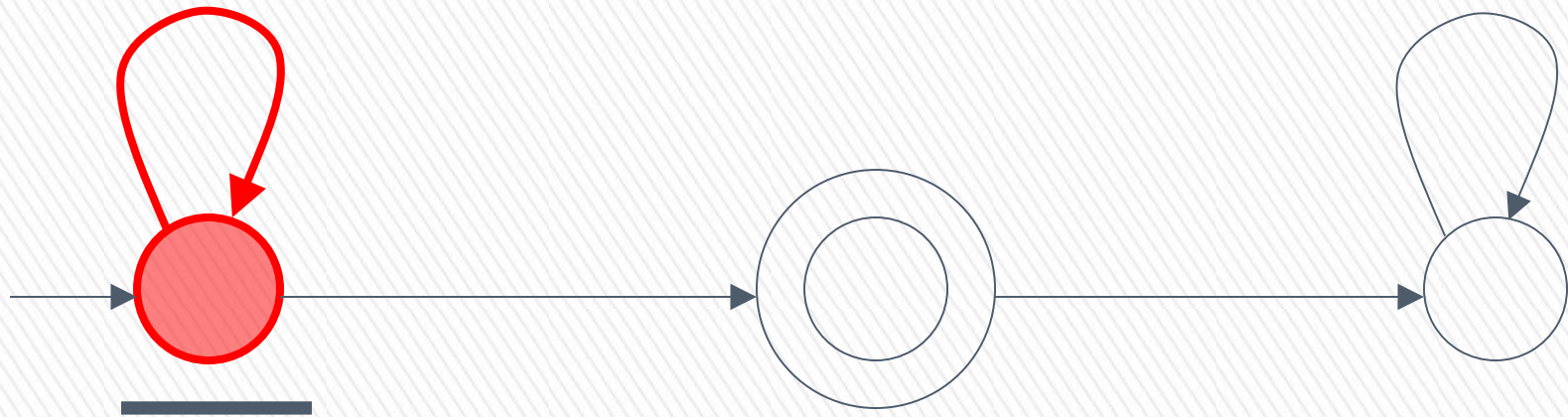
BLM2502 Theory of Computation



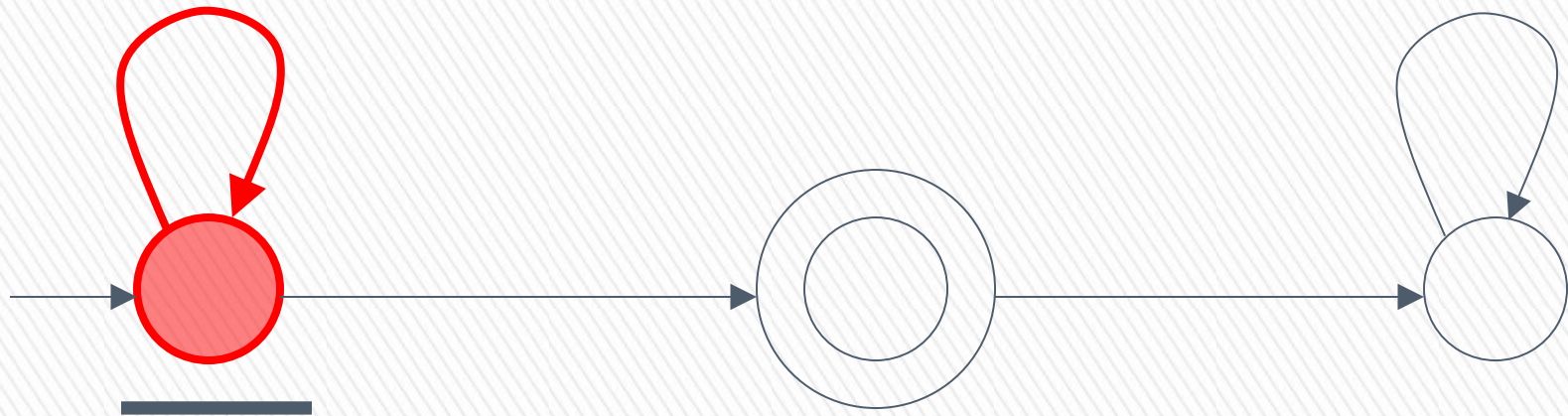
Input String



BLM2502 Theory of Computation

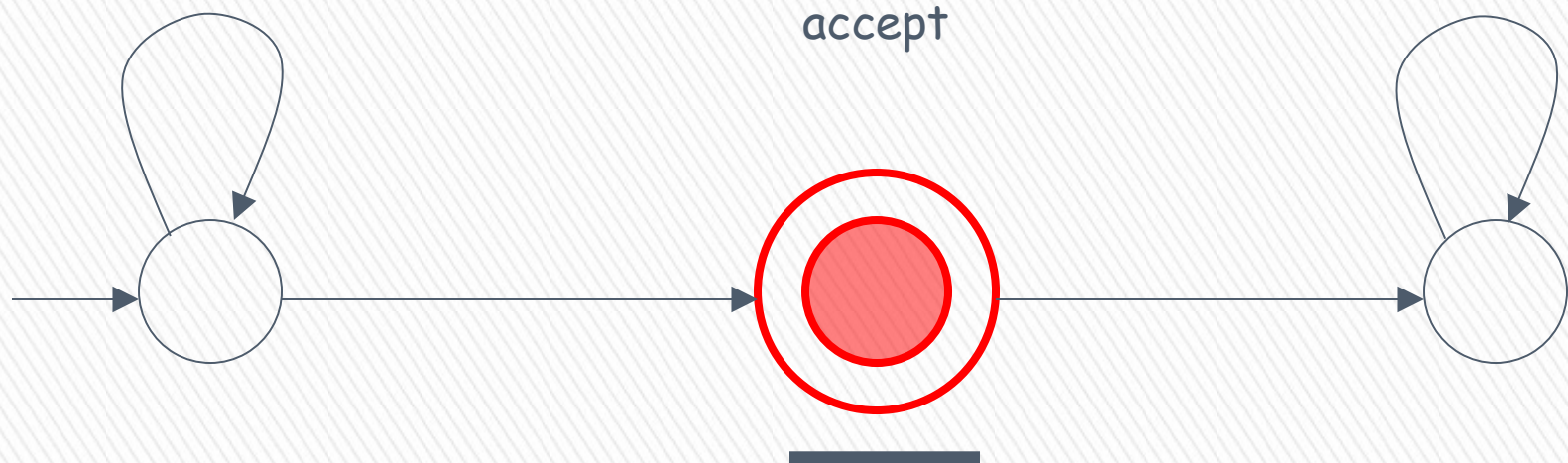


BLM2502 Theory of Computation



BLM2502 Theory of Computation

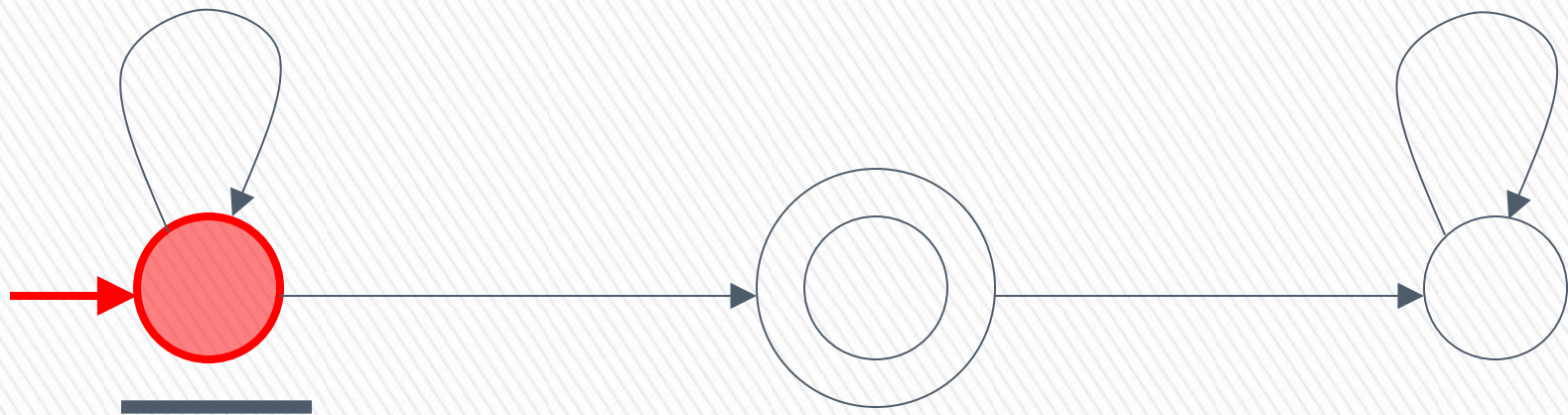
↓ Input finished



BLM2502 Theory of Computation

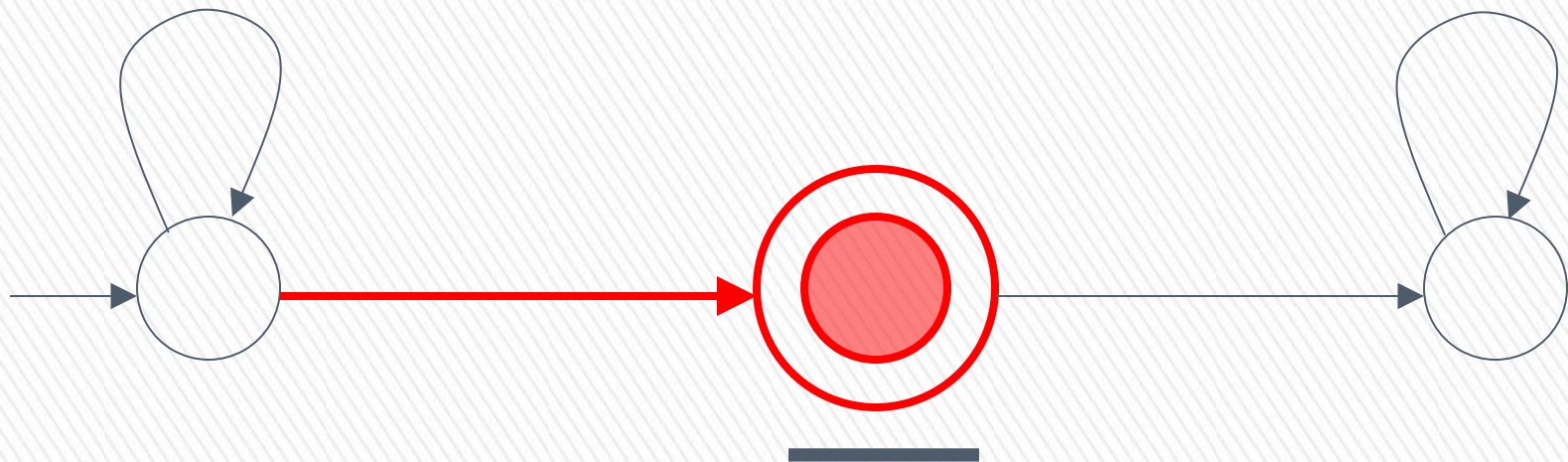
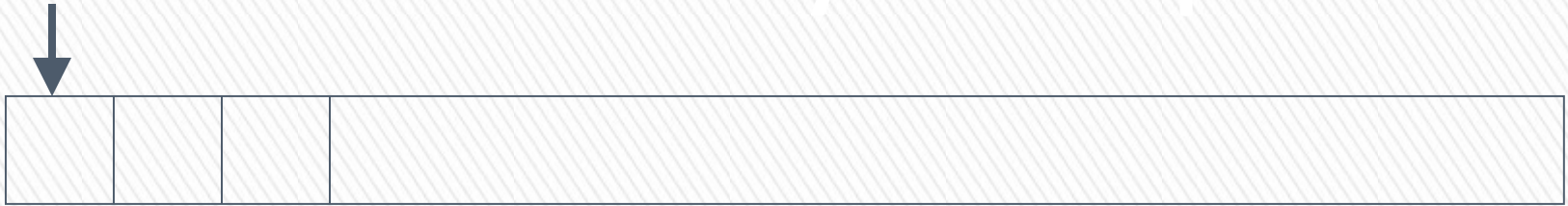


Input String

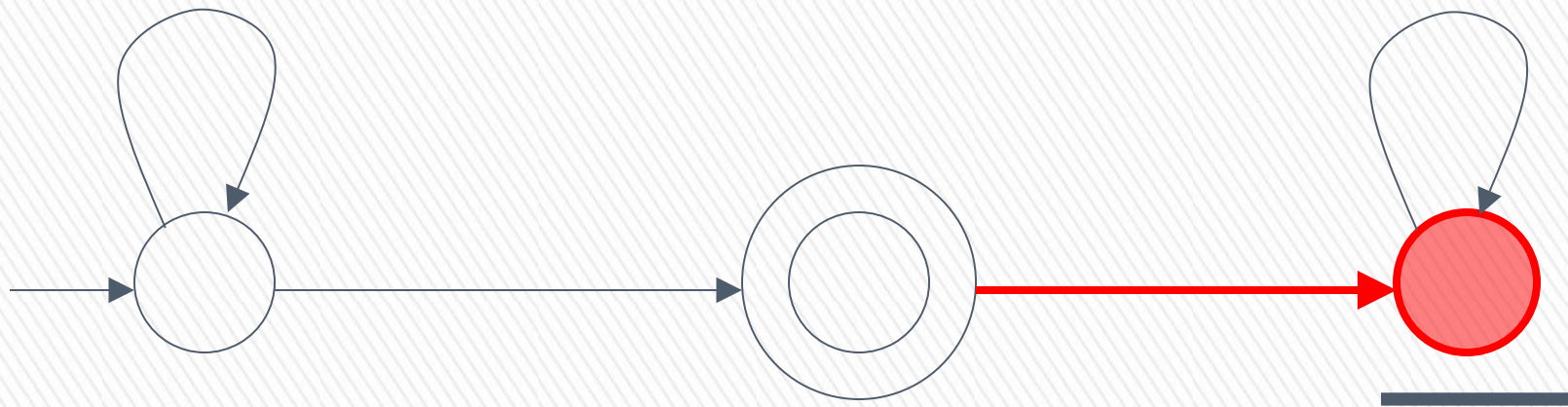


A rejection case

BLM2502 Theory of Computation

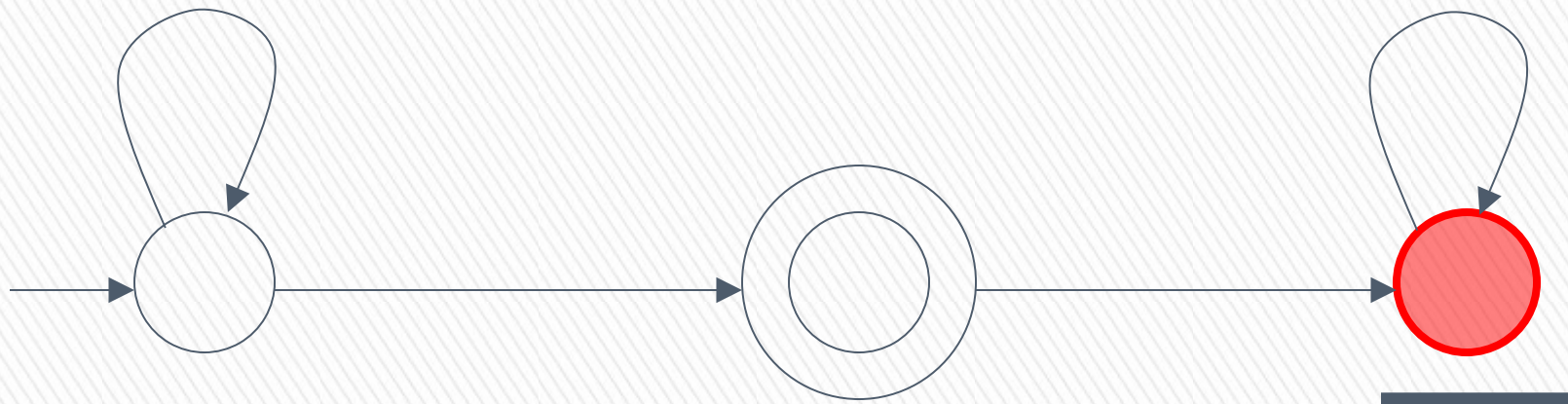


BLM2502 Theory of Computation



BLM2502 Theory of Computation

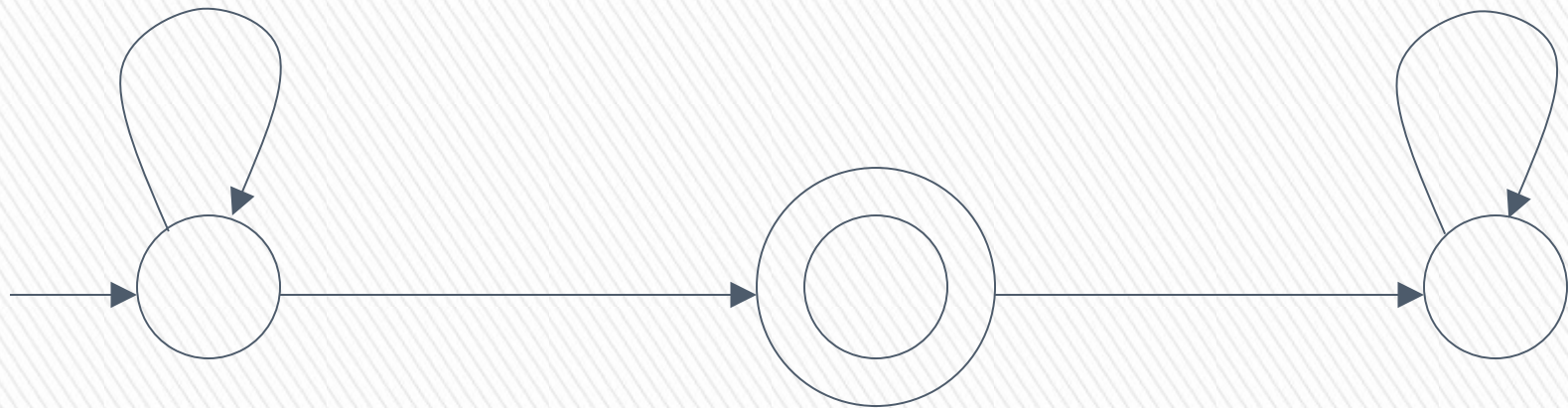
↓ Input finished



reject

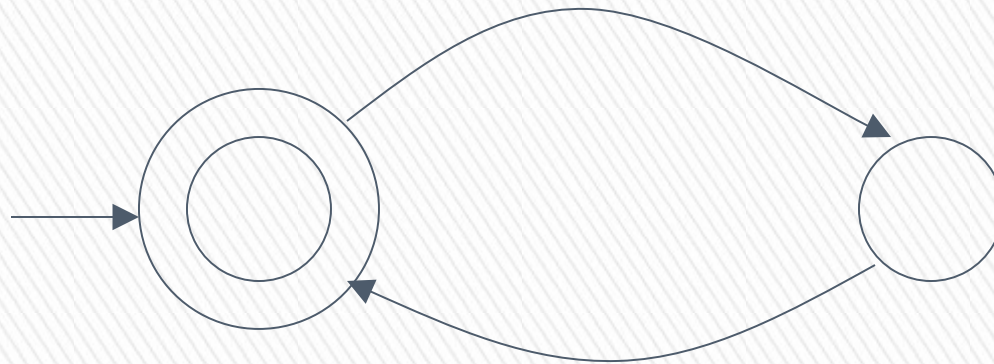
BLM2502 Theory of Computation

Language Accepted:



BLM2502 Theory of Computation

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



Language Accepted:

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

BLM2502 Theory of Computation

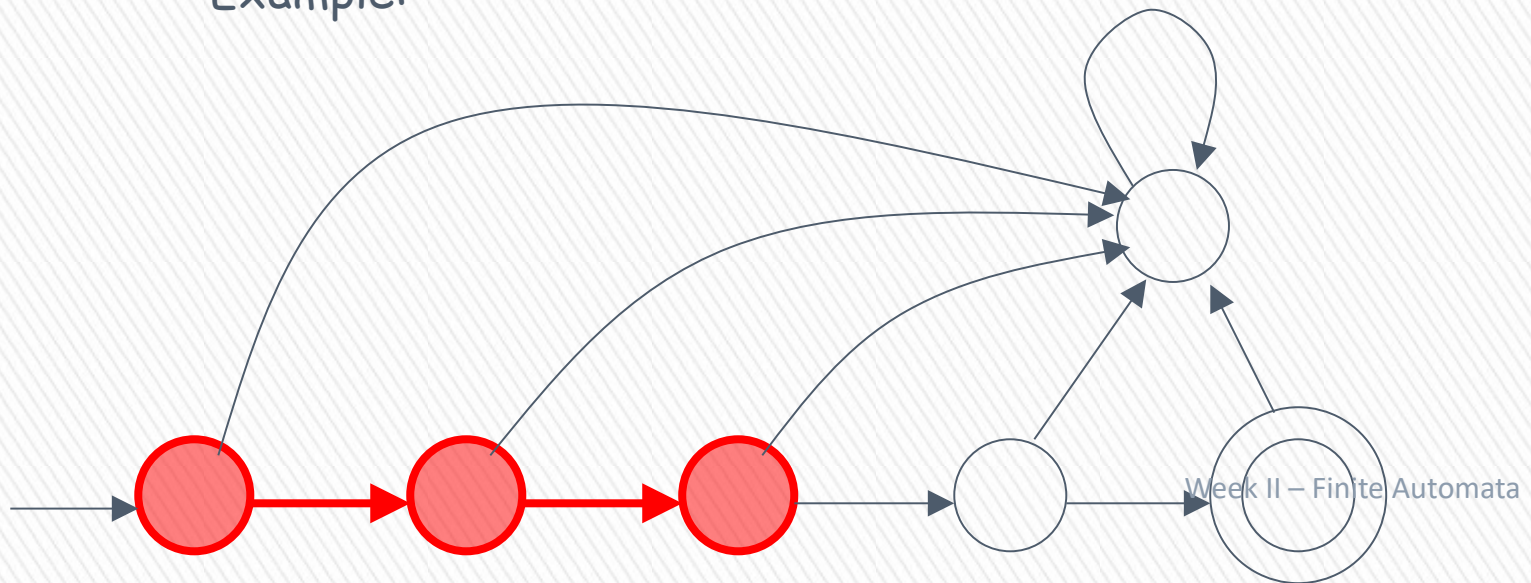
» Extended Transition Function

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

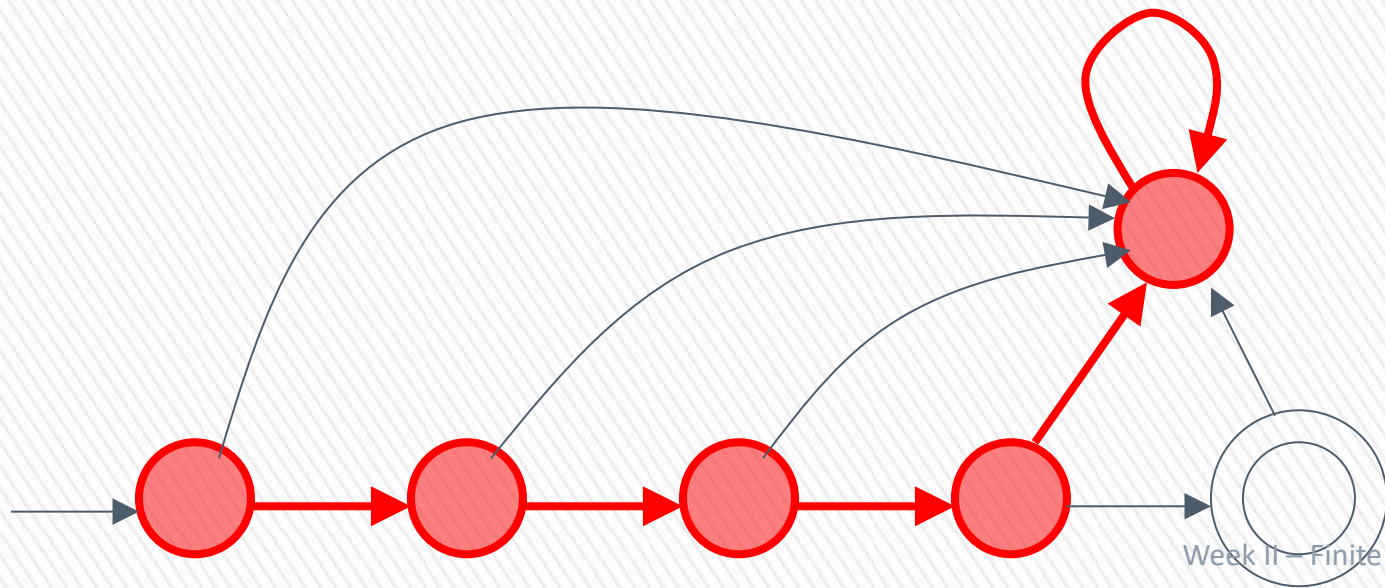
$$\delta(q, w) \rightarrow q'$$

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

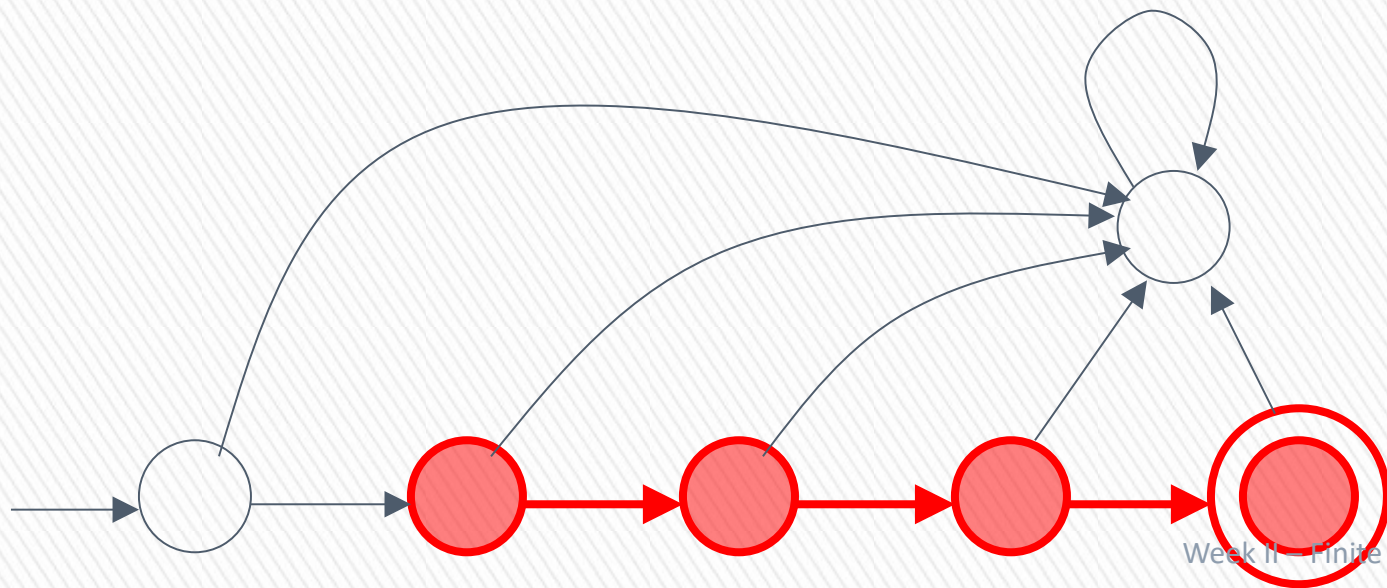
Example:



BLM2502 Theory of Computation



BLM2502 Theory of Computation



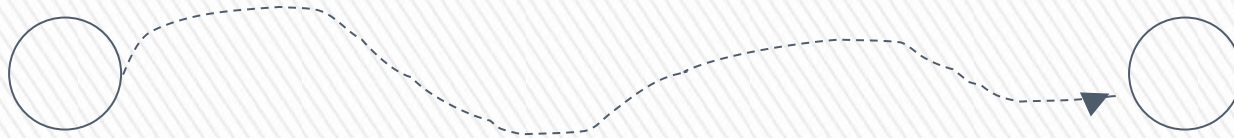
BLM2502 Theory of Computation

In general:

$\delta^*(q, w) \rightarrow q'$ implies that there is a walk of transitions



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

$$L(M) = L'$$

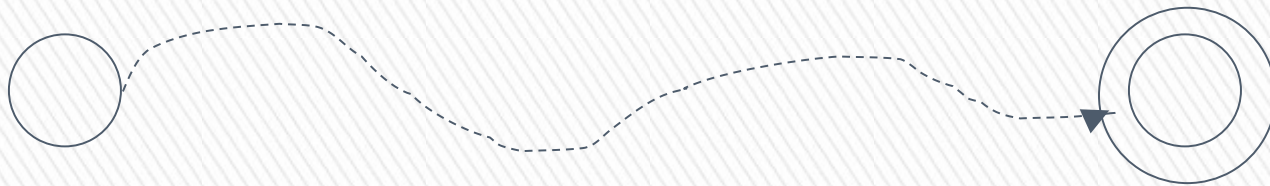
An automaton accepts one and only one language.
A language can be accepted by a number of automata

BLM2502 Theory of Computation

» For a DFA $\mathbf{M} = (Q, \Sigma, \delta, q_0, F)$

» Language accepted by \mathbf{M} :

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



BLM2502 Theory of Computation

» Language rejected by **M** :

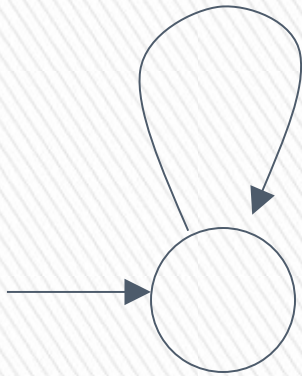


BLM2502 Theory of Computation

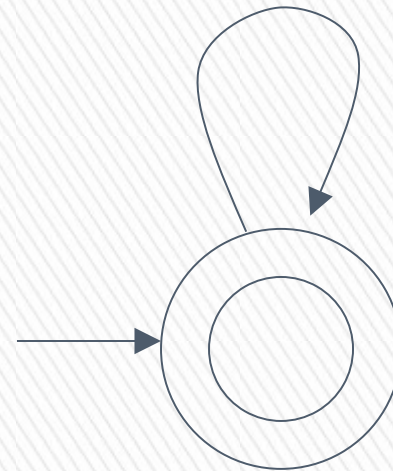


BLM2502 Theory of Computation

More DFA Examples

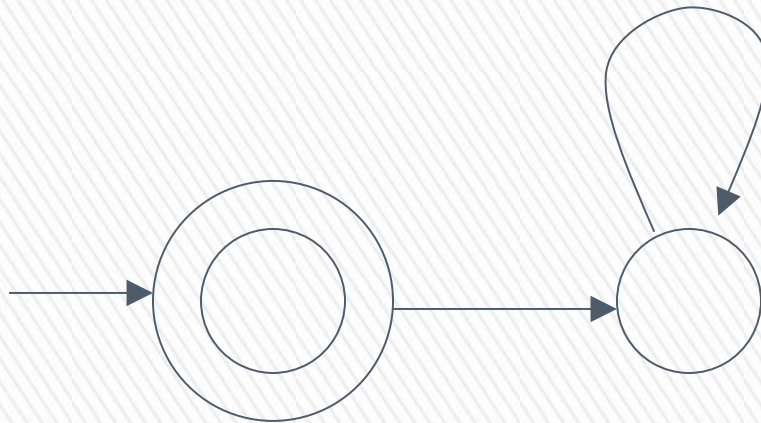


Empty language



All strings

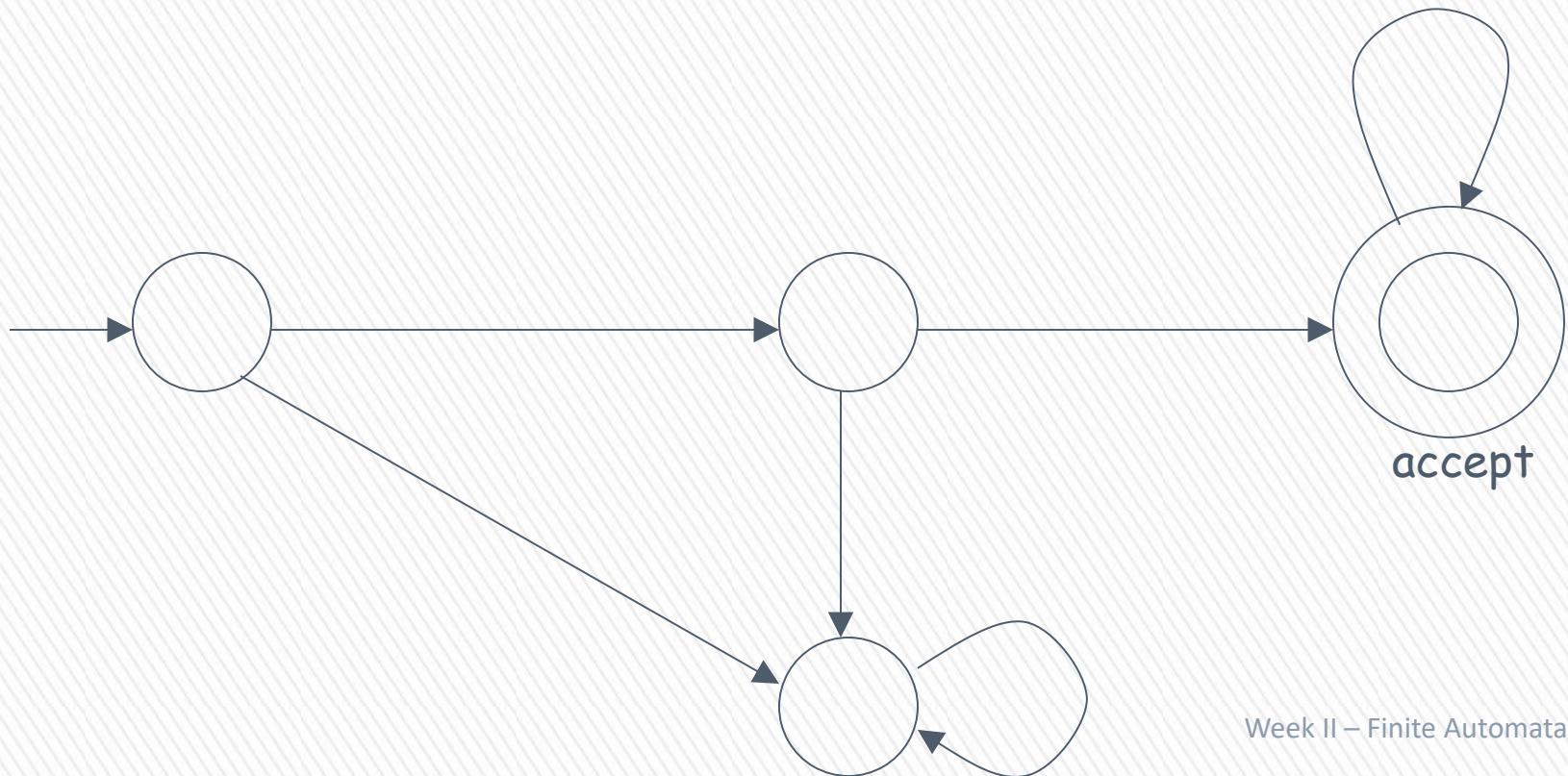
BLM2502 Theory of Computation



Language of the empty string

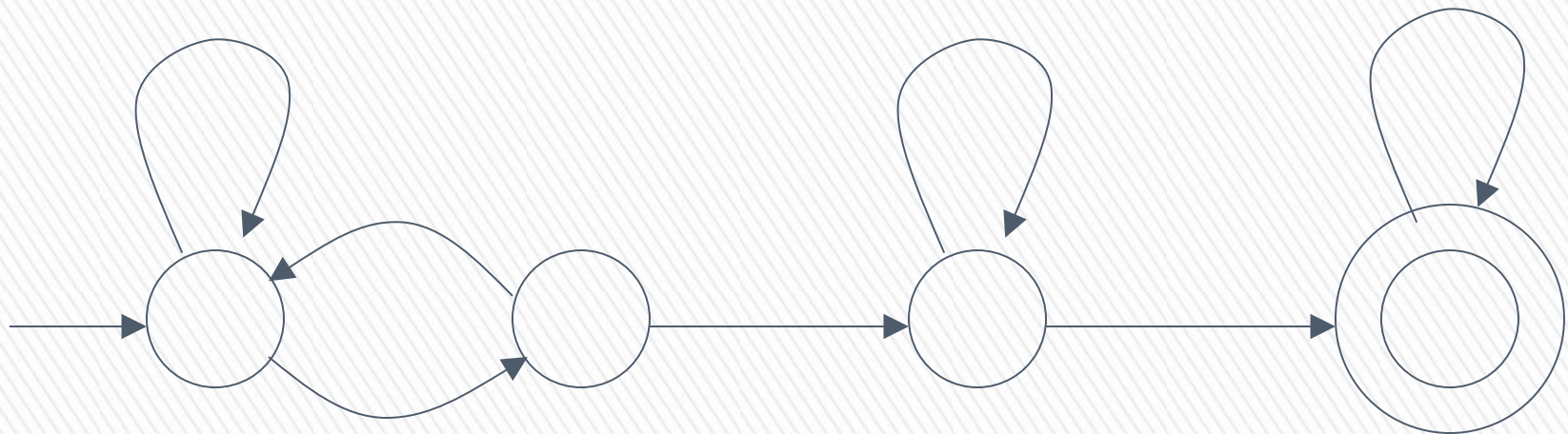
BLM2502 Theory of Computation

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



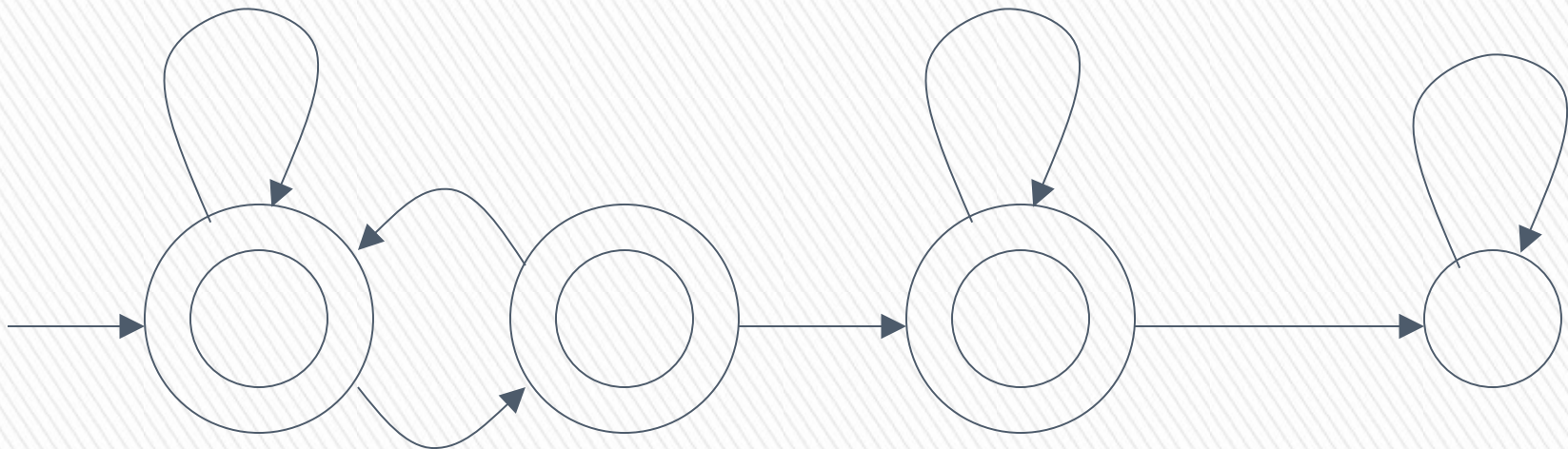
BLM2502 Theory of Computation

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

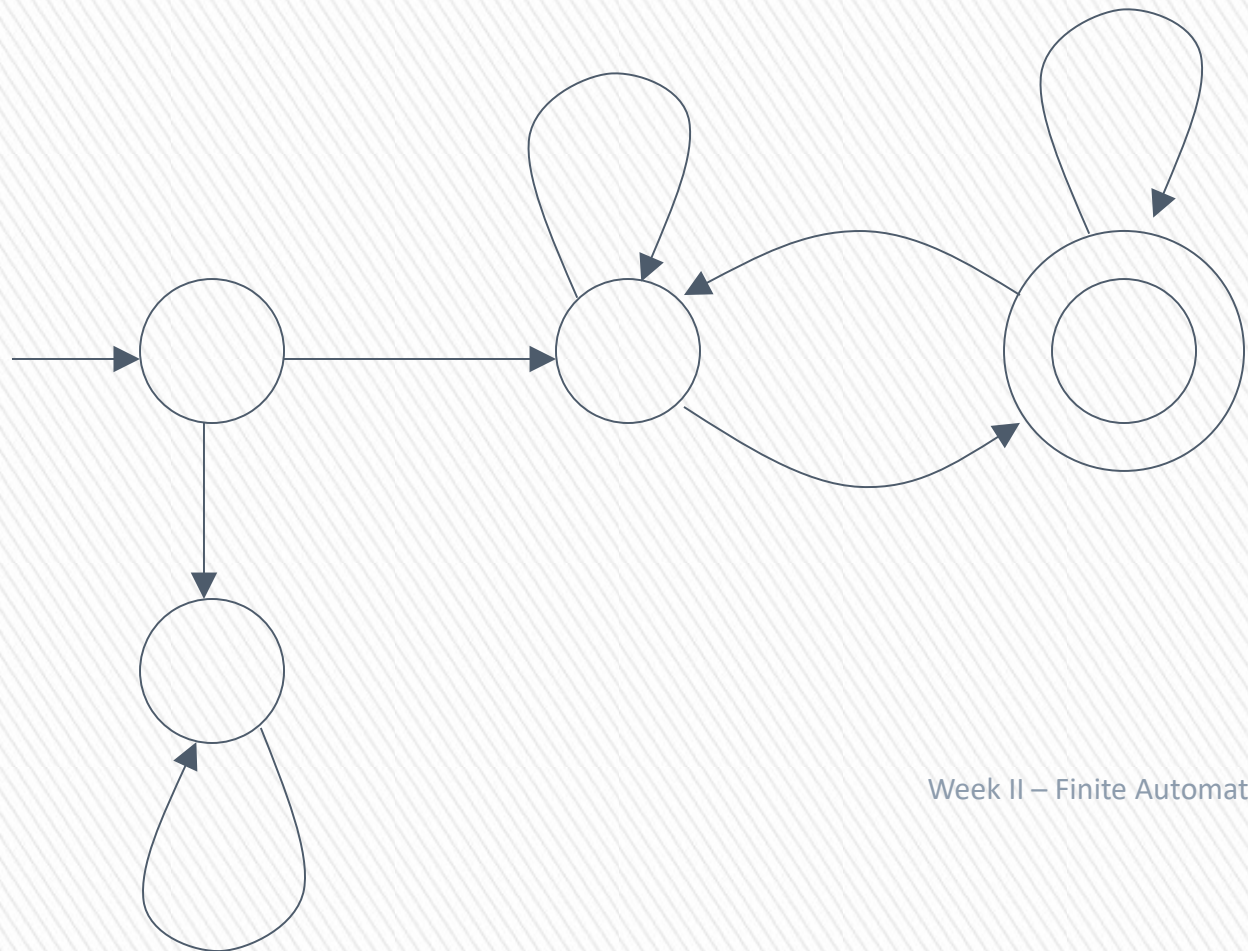


BLM2502 Theory of Computation

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



BLM2502 Theory of Computation



BLM2502 Theory of Computation

{ all binary strings without substring 001 }

{ all strings in $\{a,b\}^*$ with prefix ab }

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

BLM2502 Theory of Computation

There exist languages which are not Regular:

There is no DFA that accepts these languages