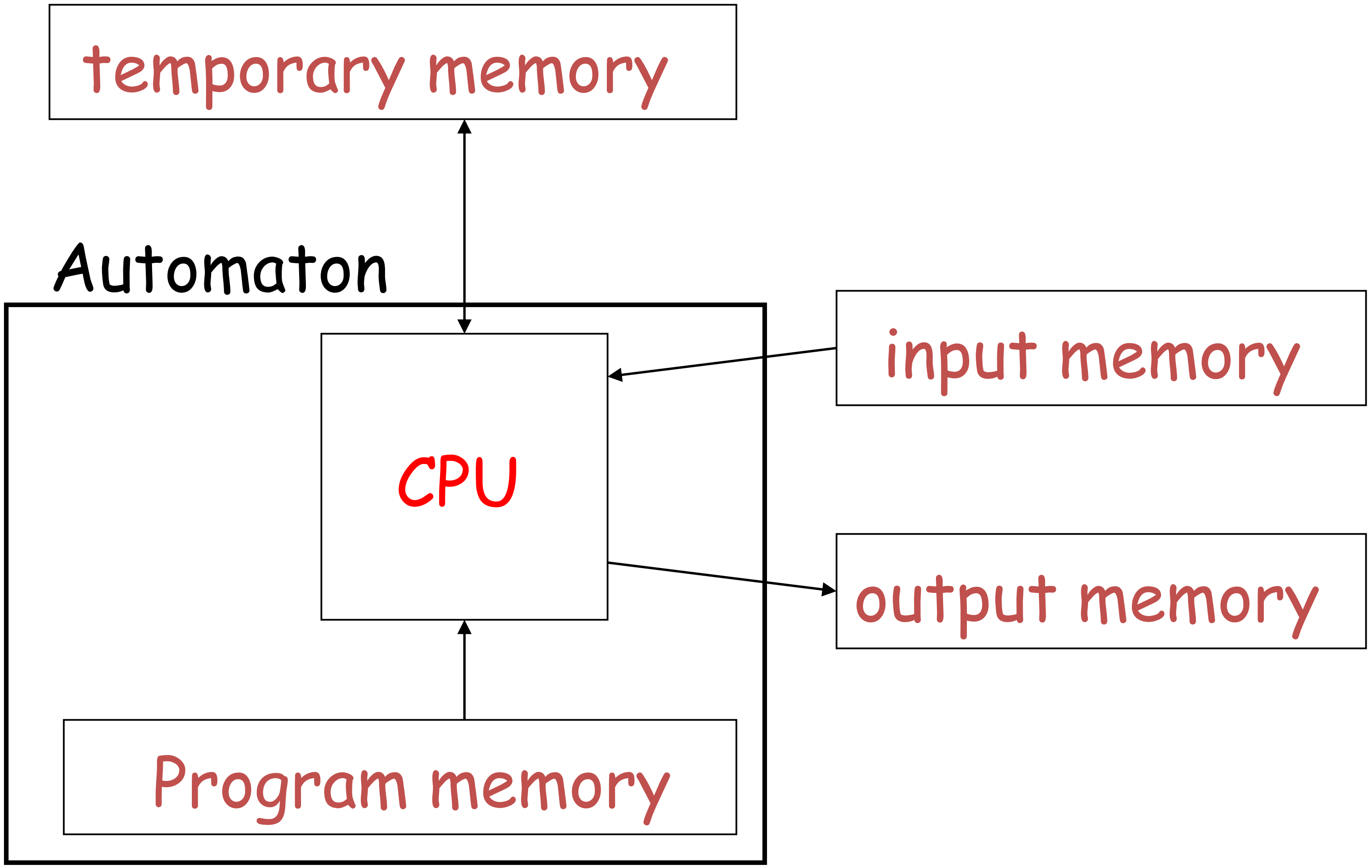
CS 3005- Theory of Automata

Lecture 01: Introduction

What is Computation?

- • Computation = Process of solving problems using a sequence of steps.
- • Processing input → producing output.
- • Can be performed by:
-    - Human (manual computation)
-    - Machine (computer computation)
-    - Mathematical models (automata, Turing machine, etc.)

Automaton


temporary memory
Automaton
CPU
input memory
output memory
Program memory

FUNDAMENTALS OF COMPUTATION

Symbols

- Smallest unit of information.
- Examples: a, b, 0, 1, x, y
- A symbol is like a single letter or digit.

Alphabet (Σ)

- A finite set of symbols.
- Represented as Σ (sigma).
- Example:

  - Σ = {0,1} → Binary alphabet

  - Σ = {a,b,c} → English letters subset

String

- A finite sequence of symbols taken from an alphabet.
- Examples:
  - If $\Sigma = \{0,1\} \rightarrow$ '101', '1100' are strings
  - If $\Sigma = \{a,b\} \rightarrow$ 'abba', 'ba' are strings

Substring

- A part (portion) of a string.
- Example:

  - String = automata

  - Substrings = auto, tom, ta, mata

Length of String |w|

- The number of symbols in a string.
- Example:
    - w = 10101 → |w| = 5
    - w = aba → |w| = 3
- Special case: Empty string (ε) has length 0.

Language

- A set of strings formed from an alphabet.
- Example:
    - $\Sigma = \{0,1\}$
    - $L = \{0, 1, 01, 10, 0011\}$
- Infinite languages are also possible.

Summary

- Computation = Problem solving through steps
- Key concepts:
  - Symbol
  - Alphabet (Σ)
  - String & Substring
  - Length of String
  - Language
- These concepts form the foundation of Automata Theory

LECTURE:2

Introduction to languages

- There are two types of languages
  - Formal Languages (Syntactic languages)
  - Informal Languages (Semantic languages)

RECAP

Alphabets

- Definition:

  A finite non-empty set of symbols (letters), is called an alphabet. It is denoted by Σ (Greek letter sigma).
- Example:

  Σ={a,b}

  Σ={0,1} //important as this is the language //which the computer understands.

  Σ={i,j,k}

NOTE:

- A certain version of language ALGOL has 113 letters

  Σ (alphabet) includes letters, digits and a variety of operators including sequential operators such as GOTO and IF

Strings

- Definition:

  Concatenation of finite symbols from the alphabet is called a string.
- Example:

  If Σ= {a,b} then

  a, abab, aaabb, abababababababab

NOTE:

<u>EMPTY STRING or NULL STRING</u>

- Sometimes a string with no symbol at all is used, denoted by (Small Greek letter Lambda) λ or (Capital Greek letter Lambda) Λ, is called an empty string or null string.

  The capital lambda will mostly be used to denote the empty string, in further discussion.

Words

- Definition:

  Words are strings belonging to some language.

Example:

If Σ= {x} then a language L can be defined as

$L=\{x^n : n=1,2,3,\ldots..\}$ or $L=\{x,xx,xxx,\ldots.\}$

Here x,xx,… are the words of L

NOTE:

- All words are strings, but not all strings are words.

Length of Strings

- Definition:

  The length of string s, denoted by |s|, is the number of letters in the string.
- Example:

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

  s=ababa

  $|s|=5$

- Example:

  Σ= {B, aB, bab, d}

  s=BaBbabBd

  Tokenizing=(B), (aB), (bab), (d)

  |s|=5

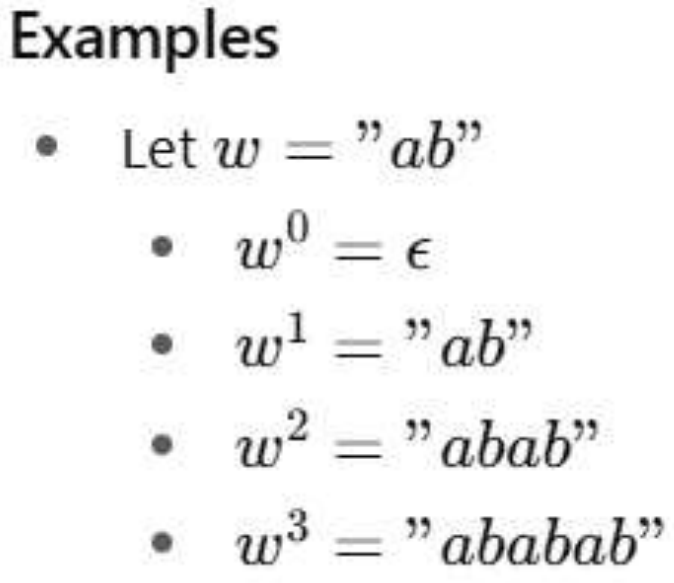
Operations of Strings

1-Power of strings:

Formal language theory: Powers help define how languages can be constructed via concatenation.

$$w^n = \begin{cases} \epsilon & \text{if } n = 0 \\ w \cdot w \cdot w \cdot \cdots \cdot w \quad (n \text{ times}) & \text{if } n > 0 \end{cases}$$

1-Power of Strings

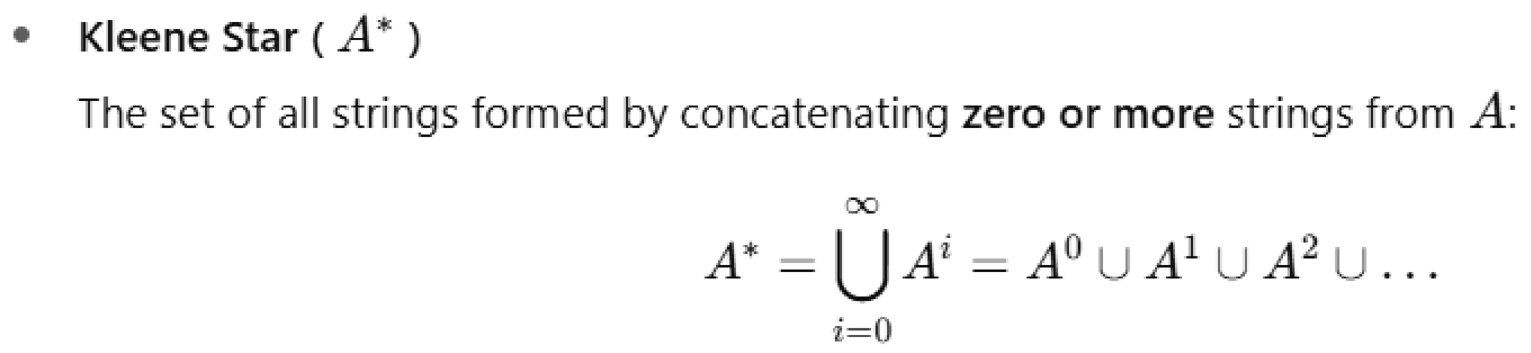


Examples

- Let $w = "ab"$
  - $w^0 = \epsilon$
  - $w^1 = "ab"$
  - $w^2 = "abab"$
  - $w^3 = "ababab"$

Power can not be negative:

1- Power of strings



- **Kleene Star (A^*)**

  The set of all strings formed by concatenating **zero or more** strings from A:

$$A^* = \bigcup_{i=0}^{\infty} A^i = A^0 \cup A^1 \cup A^2 \cup \dots$$

Universal Language:

1- Power of Strings

- **Σ** = Alphabet (set of symbols)
- **Σ*** = Set of **all possible strings** (including ε) that can be formed from Σ.
  - Example: If Σ = {a, b},

    Σ* = {ε, a, b, aa, ab, ba, bb, aaa, ...}
- **Any language L over Σ** is a **subset of Σ***:

$$L \subseteq \Sigma^*$$

Example:

- Σ = {a, b}
- L = {ab, aab, bba}
- Clearly, L ⊆ Σ*

1- Power of strings

Let the alphabet be:

$$\Sigma = \{a\}$$

Then the language generated is:

$$L = \Sigma^* = \{\epsilon, a, aa, aaa, \ldots\}$$

- Subsets formed by powers of Σ:
  - $\Sigma^0 = \{\epsilon\}$
  - $\Sigma^1 = \{a\}$
  - $\Sigma^2 = \{aa\}$
  - $\Sigma^3 = \{aaa\}$
  - ... and so on.
- Each Σ^n is **distinct**.
- Therefore, the infinite language L has **infinitely many subsets**.

Kleene Plus (A^+)

The set of all strings formed by concatenating **one or more** strings from A:

$$A^+ = \bigcup_{i=1}^{\infty} A^i = A^1 \cup A^2 \cup A^3 \cup \ldots$$

$A^+ = A^* - \{\epsilon\}$

2- Concatenation of Strings

Let Σ = {a, b}

- Example 1:

  $x = \text{"}ab\text{"},\ y = \text{"}ba\text{"}$

  $\rightarrow xy = \text{"}abba\text{"}$

- Example 2:

  $x = \text{"}a\text{"},\ y = \text{"}bbb\text{"}$

  $\rightarrow xy = \text{"}abbb\text{"}$

- Example 3 (with empty string ε):

  $x = \text{"}abc\text{"},\ y = \varepsilon$

  $\rightarrow xy = \text{"}abc\text{"}$

Properties of Concatenation

- **Associative:** $(xy)z = x(yz)$
- **Identity element:** $x\varepsilon = \varepsilon x = x$
- **Not commutative:** $xy \neq yx$ in general
  - Example: “ab”·“c” = “abc” ≠ “cab”
- **Length Property:**

$$|xy| = |x| + |y| = |yx|$$

  - Example: $x = ''ab''$ (length 2), $y = ''cd''$ (length 2)
    $\rightarrow |xy| = |''abcd''| = 4$
    $\rightarrow |yx| = |''cdab''| = 4$

3- **Reverse of a String**

- Definition:

  The reverse of a string s denoted by Rev(s) or s^r, is obtained by writing the letters of s in reverse order.

- Example:

  If s=abc is a string defined over Σ={a,b,c}

  then Rev(s) or s^r = cba

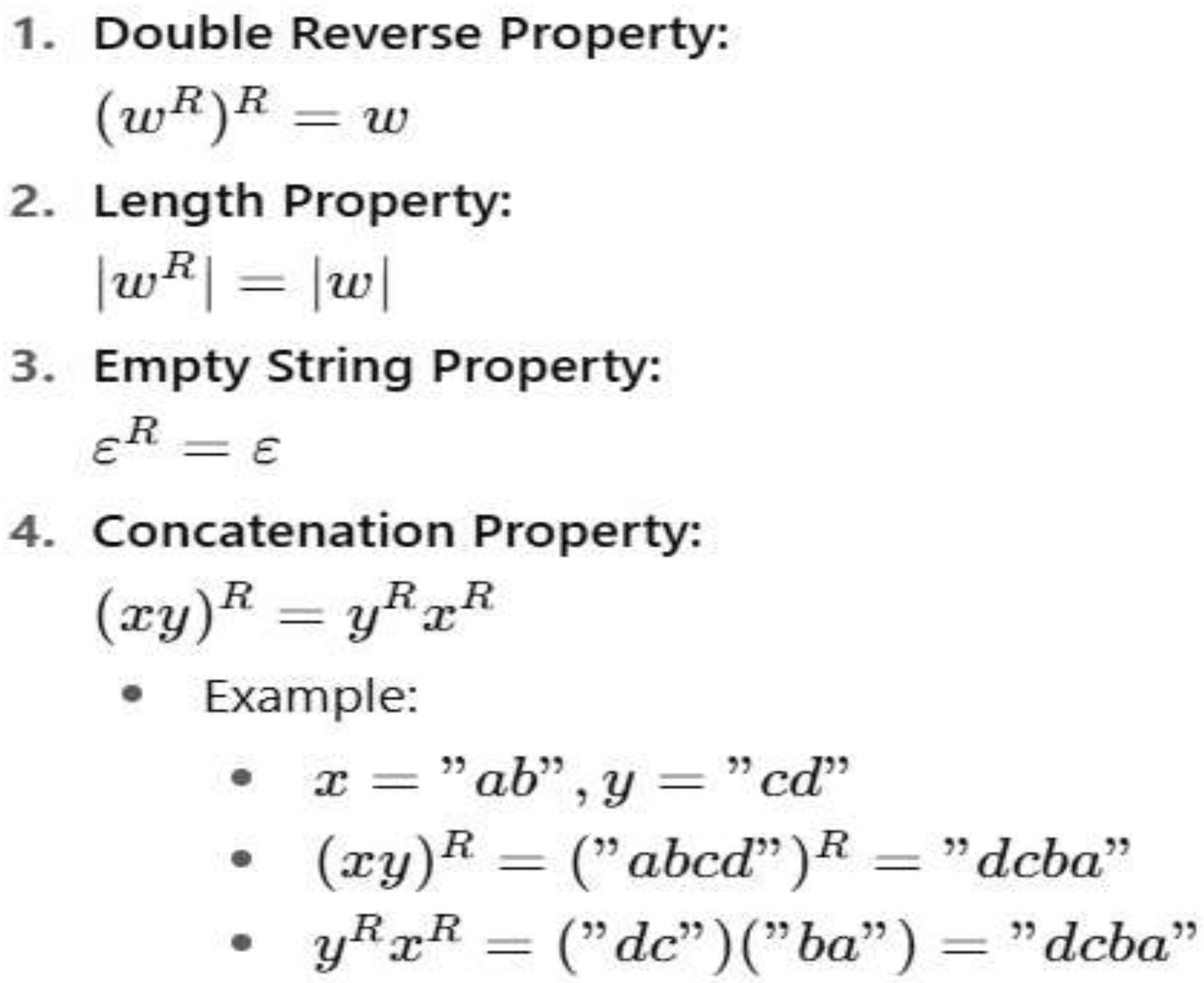
- Example:

  Σ= {B, aB, bab, d}

  s=BaBbabBd

  Rev(s)=dBbabBaB

Properties of Reverse



1. **Double Reverse Property:**
   $(w^R)^R = w$
2. **Length Property:**
   $|w^R| = |w|$
3. **Empty String Property:**
   $\varepsilon^R = \varepsilon$
4. **Concatenation Property:**
   $(xy)^R = y^R x^R$
   - Example:
     - $x = ”ab”, y = ”cd”$
     - $(xy)^R = (”abcd”)^R = ”dcba”$
     - $y^R x^R = (”dc”)(”ba”) = ”dcba”$

5. **Unary Alphabet Property:**

- If $\Sigma = \{a\}$, then for any string $w = a^n$,

$$w^R = w$$

- Example:
  - $w = "aaa"$, then $w^R = "aaa"$.

Palindrome

1. **Even Palindrome**

   A string w is called an **even palindrome** if

$$w = U \cdot U^R$$

for some $U \in \Sigma^*$.

- Here, the string is split into two equal halves, and the second half is the reverse of the first.

Example:

- $U = "ab" \rightarrow w = U \cdot U^R = "ab" \cdot "ba" = "abba".$

Palindrome

2. **Odd Palindrome**

   A string w is called an **odd palindrome** if

$$w = U \cdot x \cdot U^R$$

for some $U \in \Sigma^*$ and $x \in \Sigma$.

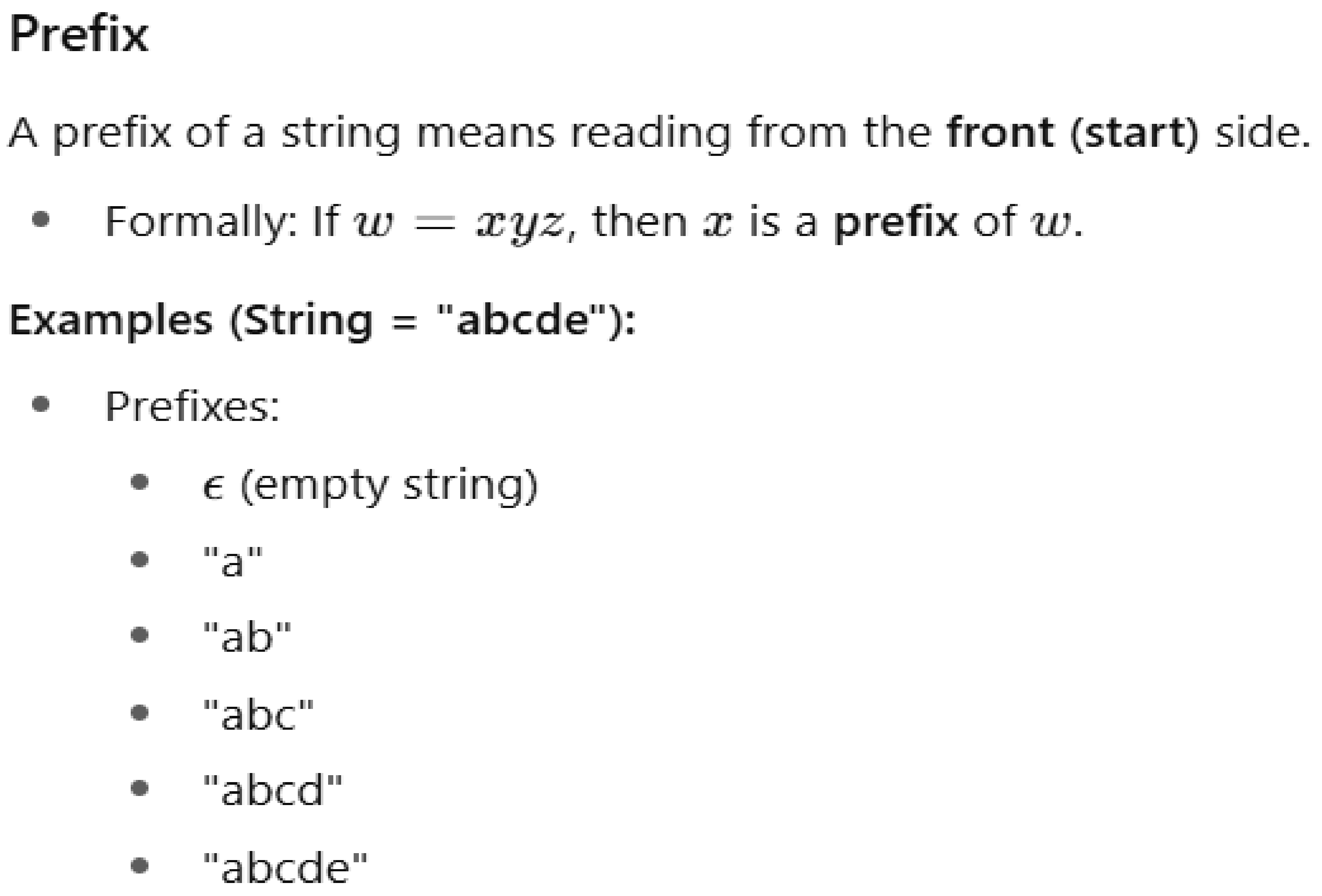
- Here, the string has a middle character x,

  and the parts before and after are mirror images.

Example:

- $U = "ab", x = "c"$
  $\rightarrow w = U \cdot x \cdot U^R = "ab" \cdot "c" \cdot "ba" = "abcba".$

4- Prefix and Suffix



Prefix

A prefix of a string means reading from the **front (start)** side.

- Formally: If $w = xyz$, then x is a **prefix** of w.

Examples (String = "abcde"):

- Prefixes:
  - ϵ (empty string)
  - "a"
  - "ab"
  - "abc"
  - "abcd"
  - "abcde"

4- Suffix

Suffix

A suffix of a string means reading from the **end (last)** side.

- Formally: If $w = xyz$, then z is a **suffix** of w.

Examples (String = "abcde"):

- Suffixes:
  - ϵ (empty string)
  - "e"
  - "de"
  - "cde"
  - "bcde"
  - "abcde"

4- Prefix and Suffix

- Alphabet: $\Sigma = \{a, b, c, d\}$
- String: $u = abcd$
- Prefix notation:

$$Prefix(u) = \{\, x \mid u = xy,\ y \in \Sigma^* \,\}$$

- Suffix notation:

$$Suffix(u) = \{\, y \mid u = xy,\ x \in \Sigma^* \,\}$$

Remarks:

- While defining an alphabet of letters consisting of more than one symbols, no letter should be started with the letter of the same alphabet *i.e.* one letter should not be the prefix of another. However, a letter may be ended in the letter of same alphabet *i.e.* one letter may be the suffix of another.

Valid/In-valid alphabets

- While defining an alphabet, an alphabet may contain letters consisting of group of symbols for example Σ_1= {B, aB, bab, d}.

- Now consider an alphabet
  Σ_2= {B, Ba, bab, d} and a string BababB.

This string can be tokenized in two different ways

- (Ba), (bab), (B)
- (B), (abab), (B)

Which shows that the second group cannot be identified as a string, defined over

Σ = {a, b}.

- As when this string is scanned by the compiler (Lexical Analyzer), first symbol B is identified as a letter belonging to Σ, while for the second letter the lexical analyzer would not be able to identify, so while defining an alphabet it should be kept in mind that ambiguity should not be created.

Conclusion

- Σ_1= {B, aB, bab, d}
- Σ_2= {B, Ba, bab, d}

Σ_1 is a valid alphabet while Σ_2 is an in-valid alphabet.

Summary

Operations on Strings