

# Introduction to languages



⌘ Language: Set of strings with rules

⌘ There are two types of languages

- ☐ Formal Languages (Syntactic languages)

- ☐ Informal Languages (Semantic languages)

# Alphabets

## ⌘ Definition:

A finite non-empty set of symbols (letters), is called an alphabet. It is denoted by  $\Sigma$  ( Greek letter sigma).

## ⌘ Example:

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

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

$$\Sigma = \{i, j, k\}$$

## NOTE:



⌘ A certain version of language ALGOL has 113 letters

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

a, abab, aaabb, abababababababababab

# NOTE:



## EMPTY STRING or NULL STRING

⌘ Sometimes a string with no symbol at all is used, denoted by (Small Greek letter Lambda)  $\lambda$  or (Capital Greek letter Lambda)  $\Lambda$ , 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  $\Sigma = \{x\}$  then a language  $L$  can be defined as

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

Here  $x, xx, \dots$  are the words of  $L$

## **NOTE:**



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

# Valid/In-valid alphabets

- ⌘ While defining an alphabet, an alphabet may contain letters consisting of group of symbols for example  $\Sigma_1 = \{B, aB, bab, d\}$ .
- ⌘ Now consider an alphabet  $\Sigma_2 = \{B, Ba, bab, d\}$  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  $\Sigma = \{a, b\}$ .



⌘ As when this string is scanned by the compiler (Lexical Analyzer), first symbol B is identified as a letter belonging to  $\Sigma$ , 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.

## 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.

# Conclusion



$$\text{⌘} \Sigma_1 = \{B, aB, bab, d\}$$

$$\text{⌘} \Sigma_2 = \{B, Ba, bab, d\}$$

$\Sigma_1$  is a valid alphabet while  $\Sigma_2$  is an in-valid alphabet.

# 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:

$\Sigma = \{B, aB, bab, d\}$

$s = BaBbabBd$

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

$|s| = 4$

# Reverse of a String

## ⌘ Definition:

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

## ⌘ Example:

If  $s=abc$  is a string defined over  $\Sigma=\{a,b,c\}$   
then  $\text{Rev}(s)$  or  $s^r = cba$



⌘ Example:

$\Sigma = \{B, aB, bab, d\}$

$s = BaBbabBd$

$\text{Rev}(s) = dBbabaBB$



# Defining Languages

- ⌘ The languages can be defined in different ways , such as
  - ☐ Descriptive definition,
  - ☐ Recursive definition,
  - ☐ Regular Expressions(RE)
  - ☐ and using Finite Automaton(FA) etc.

## **Descriptive definition of language:**

The language is defined, describing the conditions imposed on its words.



## ⌘ Example:

The language  $L$  of strings of odd length, defined over  $\Sigma=\{a\}$ , can be written as

$$L=\{a, aaa, aaaaa, \dots\}$$

## ⌘ Example:

The language  $L$  of strings that does not start with  $a$ , defined over  $\Sigma=\{a,b,c\}$ , can be written as

$$L=\{b, c, ba, bb, bc, ca, cb, cc, \dots\}$$




## ⌘ Example:

The language  $L$  of strings of length 2, defined over  $\Sigma = \{0, 1, 2\}$ , can be written as  $L = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}$

## ⌘ Example:

The language  $L$  of strings ending in 0, defined over  $\Sigma = \{0, 1\}$ , can be written as  $L = \{0, 00, 10, 000, 010, 100, 110, \dots\}$




⌘ Example: The language **EQUAL**, of strings with number of a's equal to number of b's, defined over  $\Sigma=\{a,b\}$ , can be written as

$\{\Lambda, ab, aabb, abab, baba, abba, \dots\}$

⌘ Example: The language **EVEN-EVEN**, of strings with even number of a's and even number of b's, defined over  $\Sigma=\{a,b\}$ , can be written as

$\{\Lambda, aa, bb, aaaa, aabb, abab, abba, baab, baba, bbaa, bbbb, \dots\}$




⌘ Example: The language **INTEGER**, of strings defined over  $\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ , can be written as

$$\text{INTEGER} = \{\dots, -2, -1, 0, 1, 2, \dots\}$$


⌘ Example: The language **EVEN**, of strings defined over  $\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ , can be written as

$$\text{EVEN} = \{\dots, -4, -2, 0, 2, 4, \dots\}$$



⌘ Example: The language  $\{a^n b^n\}$ , of strings defined over  $\Sigma = \{a, b\}$ , as  $\{a^n b^n : n=1, 2, 3, \dots\}$ , can be written as  $\{ab, aabb, aaabbb, aaaabbbb, \dots\}$

⌘ Example: The language  $\{a^n b^n a^n\}$ , of strings defined over  $\Sigma = \{a, b\}$ , as  $\{a^n b^n a^n : n=1, 2, 3, \dots\}$ , can be written as  $\{aba, aabbaa, aaabbbbaaa, aaaabbbbbaaaaa, \dots\}$



⌘ Example: The language **factorial**, of strings defined over  $\Sigma = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$  *i.e.*  $\{1, 2, 6, 24, 120, \dots\}$

⌘ Example: The language **FACTORIAL**, of strings defined over  $\Sigma = \{a\}$ , as  $\{a^{n!} : n = 1, 2, 3, \dots\}$ , can be written as  $\{a, aa, aaaaaa, \dots\}$ .