

Welcome to !

A thick, horizontal yellow brushstroke with a textured, painterly appearance, spanning most of the width of the slide.

Theory Of Automata

Text and Reference Material



1. *Introduction to Computer Theory*, by Daniel I. Cohen, John Wiley and Sons, Inc., 1991, Second Edition
2. *Introduction to Languages and Theory of Computation*, by J. C. Martin, McGraw Hill Book Co., 1997, Second Edition

Grading



Mid-Term Exams.	20%
Final Exams.	70%
Sessional	10%

What does automata mean?



⌘ It is the plural of automaton, and it means “something that works automatically”

Introduction to languages



⌘ There are two types of languages

- ☑ Formal Languages (Syntactic languages)
- ☑ Informal Languages (Semantic languages)

Formal Languages



- ⌘ A formal language is a precise and mathematically defined set of strings (sequences of symbols) over a finite alphabet.
- ⌘ **Examples:** Regular languages, context-free languages, and recursively enumerable languages are examples of formal languages.

Informal Languages



- ⌘ An informal language is a natural language used by humans for communication, which may not have a rigorous mathematical definition.
- ⌘ **Examples:** English, Spanish, French, etc.

Alphabets

⌘ Definition:

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

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



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

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

Σ_1 is a valid alphabet while Σ_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), (B), (d)

$|s| = 5$

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, using 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 $\{a, aa, aaaaaa, \dots\}$. It is to be noted that the language FACTORIAL can be defined over any single letter alphabet.

- 
- ⌘ Example: The language **DOUBLEFACTORIAL**, of strings defined over $\Sigma = \{a, b\}$, as $\{a^{n!}b^{n!} : n=1,2,3,\dots\}$, can be written as $\{ab, aabb, aaaaaabbbbbbb,\dots\}$
 - ⌘ Example: The language **SQUARE**, of strings defined over $\Sigma = \{a\}$, as $\{a^{n^2} : n=1,2,3,\dots\}$, can be written as $\{a, aaaa, aaaaaaaaaa,\dots\}$



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



⌘ Example: The language **PRIME**, of strings defined over $\Sigma = \{a\}$, as $\{a^p : p \text{ is prime}\}$, can be written as $\{aa, aaa, aaaaa, aaaaaaa, aaaaaaaaaaaaaa\dots\}$

An Important language

⌘ PALINDROME:

The language consisting of Λ and the strings s defined over Σ such that $\text{Rev}(s)=s$.

It is to be denoted that the words of PALINDROME are called palindromes.

⌘ Example: For $\Sigma=\{a,b\}$,
PALINDROME= $\{\Lambda, a, b, aa, bb, aaa, aba, bab, bbb, \dots\}$

Remark



⌘ There are as many palindromes of length $2n$ as there are of length $2n-1$.

To prove the above remark, the following is to be noted:

Note

⌘ Number of strings of length 'm' defined over alphabet of 'n' letters is n^m .

⌘ Examples:

☒ The language of strings of length 2, defined over $\Sigma=\{a,b\}$ is $L=\{aa, ab, ba, bb\}$ *i.e.* number of strings = 2^2

☒ The language of strings of length 3, defined over $\Sigma=\{a,b\}$ is $L=\{aaa, aab, aba, baa, abb, bab, bba, bbb\}$ *i.e.* number of strings = 2^3