Operations on Language : Kleene Closure

We define $L^0 = \{\epsilon\}$

$$L^i = LL^{i-1} \text{ for } i > 0$$

(that is $L^i = \underbrace{L \cdots L}_{i \text{ times}}$)

$$L^* = \bigcup_{i \geq 0} L^i$$

Note : For $z \neq \epsilon$, $z \in L^*$ if and only if there exists i such that $z = x_1 \cdots x_i$ where for all $j \in \{1, \ldots, i\},\ x_j \in L$

L^* is called the Kleene's closure of L

Also we define $L^+ = \bigcup_{i \geq 1} L^i$

Note : If we choose $L = \Sigma$ then L^* is same as Σ^* which we defined earlier.

Operations on Language : Kleene Closure

We define $L^0 = \{\epsilon\}$

$$L^i = LL^{i-1} \text{ for } i > 0$$

(that is $L^i = \underbrace{L \cdots L}_{i \text{ times}}$)

$$\boxed{L^* = \bigcup_{i \geq 0} L^i}$$

Note : For $z \neq \epsilon$, $z \in L^*$ if and only if there exists i such that $z = x_1 \cdots x_i$ where for all $j \in \{1, \ldots, i\},\ x_j \in L$

L^* is called the Kleene's closure of L

Also we define $\boxed{L^+ = \bigcup_{i \geq 1} L^i}$

Note : If we choose $L = \Sigma$ then L^* is same as Σ^* which we defined earlier.

Operations on Language : Concatenation

Note : This is different from (but related to) string concatenation, which we defined earlier.

For languages $L_1, L_2 \subset \Sigma^*$ their concatenation $L_1 L_2$ is defined as,

$$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

Example 1: $L_1 = \{a, b\},\ L_2 = \{c, d\} \Rightarrow L_1 L_2 = \{ac, ad, bc, bd\}$

Example 2: $L_1 = \{a\}^*,\ L_2 = \{b\}^* \Rightarrow L_1 L_2 = \{a^i b^j \mid i, j \geq 0\}$

Operations on Language : Complement

For a language L over alphabet Σ, complement of L is defined as,

$$\overline{L} = \Sigma^* \setminus L$$

(which is also a language over the same alphabet)

Example : Let $\Sigma = \{a, b\}$ and $L = \{ax \mid x \in \Sigma^*\}$

then $\overline{L} = \{bx \mid x \in \Sigma^*\} \cup \{\epsilon\}$

Operations on Language : Union and Intersection

Languages are sets. All the set operations are defined on them.

If L_1 and L_2 are languages over Σ then,

$L_1 \cup L_2$ and $L_1 \cap L_2$ are also languages over Σ

$$L_1, L_2 \subset \Sigma^* \Rightarrow L_1 \cup L_2, L_1 \cap L_2 \subset \Sigma^*$$

Example : Let $\Sigma = \{a, b\}$

$$L_1 = \{a^i b^j \mid i \geq j\}, \qquad L_2 = \{a^i b^j \mid i \leq j\}$$

then $L_1 \cap L_2 = \{a^i b^j \mid i = j\}, \qquad L_1 \cup L_2 = \{a^i b^j \mid i, j \in \mathbb{N}\}$

$L_1, L_2, L_1 \cap L_2, L_1 \cup L_2$ all are languages over Σ

(Formal) Language

A **language** over alphabet Σ is a set of strings over Σ .

Equivalently, a language over alphabet Σ is a subset of Σ^*.

Always read the expression $L \subset \Sigma^*$ as L is a language over alphabet Σ

It is sometimes called a **formal language** to distinguish it from **natural language**s like english, bengali etc. We shall always use the term language in the sense formal language.

Example : Consider the alphabet $\Sigma = \{a, b\}$

Here are some examples of languages over Σ

$\phi = \{\}$ is the empty language

$\{x \in \Sigma^* \quad | \quad |x| < 5\}$ is a finite language

$\{x \in \Sigma^* \quad | \quad |x| > 5\}$ is an infinite language

Σ^* is also a language

(Formal) Language

A **language** over alphabet Σ is a set of strings over Σ .

Equivalently, a language over alphabet Σ is a subset of Σ^*.

Always read the expression $L \subset \Sigma^*$ as L is a language over alphabet Σ

It is sometimes called a **formal language** to distinguish it from **natural language**s like english, bengali etc. We shall always use the term language in the sense formal language.

Example : Consider the alphabet $\Sigma = \{a, b\}$

Here are some examples of languages over Σ

$\phi = \{\}$ is the empty language

$\{x \in \Sigma^* \mid |x| < 5\}$ is a finite language

$\{x \in \Sigma^* \mid |x| > 5\}$ is an infinite language

Σ^* is also a language

Prefix, Suffix and Substring

If $z = xy$ where $x, y, z \in \Sigma^*$ then,

x is a prefix of z and y is a suffix of z

Note : For all $x \in \Sigma^*$, $x = x\epsilon = \epsilon x$ (ϵ is the empty string) . So,

x is a prefix of x and x is a suffix of x

ϵ is a prefix of x and ϵ is a suffix of x

Example : Prefixes of aab are ϵ, a, aa, aab

Suffixes of aab are aab, ab, b, ϵ

If $z = xwy$ where $w, x, y, z \in \Sigma^*$ then, w is a substring of z

Example : aab is a substring of $bb\underline{aab}b$

(Notice that all the prefixes and suffixes are also substrings.)

String Concatenation

The concatenation of two strings $a_1 \cdots a_n$ and $b_1 \cdots b_m$ is $a_1 \ldots a_n b_1 \ldots b_m$

(a_i s and b_j s are symbols)

Example : Fix alphabet *{a,b,c,...,r,s,t}*

dog and *house* are strings over our chosen alphabet

concatenation of *dog* and *house* is *doghouse*

Notation : Concatenation of strings x and y is simply written as xy

Notice : For all string x , $\epsilon x = x\epsilon = x$ where ϵ is the empty string

Notation : For a string $x \in \Sigma^*$,

x^n denotes $\underbrace{x \ldots x}_{n \text{ times}}$

x^0 is defined as the empty string ϵ

Example : If $x = ab$ then $x^3 = ababab$

Example 1 : Let $\Sigma = \{a, b\}$

Then,

$$\Sigma^2 = \{aa, ab, ba, bb\}$$

$$\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

etc.

Example 2 : If $\Sigma = \{0, 1\}$ then $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

Example 3 : If $\Sigma = \{a\}$ then $\Sigma^* = \{\epsilon, a, aa, aaa, \ldots\}$

Note : Σ^* is an infinite set but, each string in it is of finite length.

An infinite sequence like $aaa\ldots$ is NOT a string.

Formal Definitions (Length of String)

- Length of a string is the number of symbols in it.
  Length of a string x is denoted by $|x|$.

Example : If $x = ccabba$ then $|x| = 6$

Note: Length of Empty string is zero i.e. $|\epsilon| = 0$

- A symbol is identified with the string of length 1 , consisting of that symbol.
  Thus the alphabet Σ is also the set all strings of length 1.

- For integer n>1, the n-tuple $(a_1, \ldots, a_n) \in \Sigma^n$ is identified with the string $a_1 \cdots a_n$
  Σ^0 is defined as $\{\epsilon\}$

Thus for **all non-negative integer n**, Σ^n is the set of all strings of length n over Σ

Σ^* is the **set of all strings over** Σ i.e. , $\Sigma^* = \bigcup_{i \geq 0} \Sigma^i$

Σ^+ is the **set of all non-empty strings over** Σ i.e. , $\Sigma^+ = \bigcup_{i \geq 1} \Sigma^i$

`Symbol' is primitive notion

Symbols are not described in terms of anything else.

Consider the strings over the alphabet $\Sigma = \{0, 1\}$

$0, 1, 00, 11$ are **NOT NUMBERS**, they are just symbols / strings

$00 \neq 0$ They are **DIFFERENT STRINGS**

$1 + 1$ is **UNDEFINED**. + is not defined on Σ^*

If we want to interpret the strings as numbers

we must define functions to map the strings to numbers.

For example, Let $f : \Sigma^* \to \mathbb{N}$ such that,

for all $x \in \Sigma^*$, x is a binary representation of the integer $f(x)$

Thus 00 is a string but, $f(00)$ is the integer 'zero'

Formal Definitions (Symbol, Alphabet and String)

- \`**symbol**' is a primitive notion. In cannot be explained in terms of anything else.
  (like \`point' is a primitive notion in geometry)

- An **alphabet** is a nonempty finite set of symbols.
  Example : We may choose an alphabet $\Sigma = \{a, b, c\}$

- A **string** or **word** , over an alphabet Σ , is a
  sequence of finitely many (**zero** of more) symbols in Σ .
  Example: $a, bc, cba, caa, abbcbaca$ etc. are strings over alphabet Σ

- **Empty string** is the string of zero symbols. It is denoted as ϵ .

Notational Conventions

Σ almost always denotes an alphabet

Σ^* set of all strings over alphabet Σ

$a, b, c, 0, 1, 2$ etc. usually denote symbols

u, v, w, x, y, z etc. usually denote strings

Until mentioned otherwise, the expressions

$a \in \Sigma$ means a is a **symbol** in alphabet Σ

$x \in \Sigma^*$ means x is a **string** over alphabet Σ

Encoding the Information

Information is encoded as patterns or **strings** constructed out of a **fixed FINITE set of symbols**.

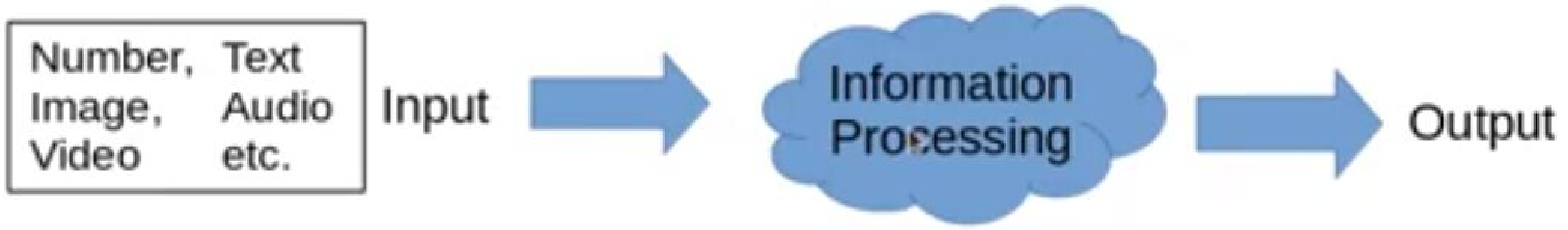
This finite set of symbols is called **alphabet**.

Example 2 : Texts / programs / codes are usually written as strings over the alphabet consisting of ASCII symbols.

```
{ <space>, <tab>, <newline>,...
  ,!,@,#,$,%,&,*,(,),?,~,...
  ,1,2,3,...,A,...,Z,a,...,z,... }
```

What is Theory of Computation ?

A theory to mathematically formalize the intuitive notion of `computation'.



1. Formalize how to encode the Input / Output / Intermediate information

2. Mathematically describe a model of computation that is,
   `some kind of machine'
   or `some sort of rules'
   to capture the processing.

3. The ultimate goal is to classify computational problems according to their hardness.
   (A problem is harder if it requires a machine with more power/resources.)