The terms **alphabet**, **word** and **language** are used in a strict technical sense in this course.

- An **alphabet** is a **finite set** of symbols.
- A **word** is a **finite sequence** of symbols.
- A **language** is a **set** of words.
- Languages can be finite or infinite.
- The term **string** is often used interchangeably with the term **word**.
Symbols and Alphabets

- What is a symbol, then?
- Anything, but it has to come from an alphabet.
- Usually, \( \Sigma \) is used to denote an alphabet.
- Example alphabets:

\[
\begin{align*}
\Sigma_1 & = \{0, 1\} \\
\Sigma_2 & = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\} \\
\Sigma_3 & = \{\circ, \Box, \triangle\} \\
\Sigma_4 & = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /\}
\end{align*}
\]

- Important exception: \( \varepsilon \) is never used as an alphabet symbol.
The Empty Word

- $\varepsilon$ is used to denote the empty word: the sequence of zero symbols.
- But $\varepsilon$ itself is not a symbol!
- $\varepsilon$ is a word, not a set.
- So don’t confuse it with the empty set (denoted $\emptyset$ or $\{\}$.)
- Thus, $\{\varepsilon\} \neq \{\}$. 
The set of all words over an alphabet $\Sigma$ is denoted by $\Sigma^*$. $\Sigma^*$ can be defined inductively as follows:

- $\varepsilon \in \Sigma^*$
- If $x \in \Sigma$ and $w \in \Sigma^*$ then $xw \in \Sigma^*$

Note that $\varepsilon \in \Sigma^*$ for any alphabet $\Sigma$ (including $\Sigma = \emptyset$).

Iff $\Sigma \neq \emptyset$ then $\Sigma^*$ is an infinite set (of finite words).
Example

- Given $\Sigma = \{0, 1\}$, some elements of $\Sigma^*$ are:

  - $\varepsilon$,
  - 0, 1,
  - 00, 10, 01, 11,
  - 000, 100, 010, 110, 001, 101, 011, 111,
  - 0000, . . .

- This is just applying the inductive definition.

- Important note: only write $\varepsilon$ if it appears on its own, as it denotes an absence of symbols.
The set of all words over $\Sigma$ of length $n$ is denoted by $\Sigma^n$ (where $n \in \mathbb{N}$).

For example, if $\Sigma = \{a, b\}$, then $\Sigma^2 = \{aa, ab, ba, bb\}$.

This can be used to give an alternative (but equivalent) definition of $\Sigma^*$:

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Remember that in computer science, $0 \in \mathbb{N}$. 
A language $L$ over an alphabet $\Sigma$ is a subset of $\Sigma^*$:

$$L \subseteq \Sigma^*$$

or

$$L \in \mathcal{P}(\Sigma^*)$$

A language may be a finite or infinite set.

Note that while $\varepsilon$ is always an element of $\Sigma^*$, it may or may not be an element of an arbitrary language.
Exercise

Given $\Sigma = \{a, b, c\}$, define some languages over $\Sigma$.

- $\{a, abba, baa, cab\}$
- $\{c\}$
- $\{\varepsilon, a, bbb\}$
- $\{\varepsilon\}$
- $\{a^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n \mid n \in \mathbb{N}, n \geq 10\}$
- $\{w \mid w \in \Sigma^*, \text{odd (length (w))}\}$
- $\emptyset$
- $\Sigma^*$
An important operation on words ($\Sigma^*$) is **concatenation**.

Concatenation is denoted by **juxtaposition** (i.e. writing the words side by side without using an operator symbol).

If $v \in \Sigma^*$ and $w \in \Sigma^*$ then $vw \in \Sigma^*$

Concatenation can be defined by primitive recursion:

\[
\begin{align*}
\varepsilon w &= w \\
(xv)w &= x(vw)
\end{align*}
\]

where

\[
\begin{align*}
x &\in \Sigma \\
v, w &\in \Sigma^*
\end{align*}
\]
Properties of Word Concatenation

- Concatenation is **associative** and has unit $\varepsilon$:

  $$u \ (vw) = (uv) \ w$$
  $$\varepsilon u = u = u \varepsilon$$

  where

  $$u, v, w \in \Sigma^*$$

- Concatenation of words is **not commutative** (i.e. order matters), as words are sequences.

  $$vw \neq wv$$
Remember, languages are *sets*, not sequences.

Given two languages $M$ and $N$ over an alphabet $\Sigma$, their concatenation $(MN)$ is defined:

$$MN = \{uv \mid u \in M \land v \in N\}$$

**Example:**

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

$$M = \{\varepsilon, a, aa\}$$

$$N = \{b, c\}$$

$$MN = \{uv \mid u \in \{\varepsilon, a, aa\} \land v \in \{b, c\}\}$$

$$= \{\varepsilon b, \varepsilon c, ab, ac, aab, aac\}$$

$$= \{b, c, ab, ac, aab, aac\}$$
Properties of Language Concatenation (1)

- Concatenation of languages is **associative**:

  \[ L(MN) = (LM)N \]

- Concatenation of languages has **zero** \(\emptyset\) (the empty language):

  \[ L\emptyset = \emptyset = \emptyset L \]

- Concatenation of languages has **unit** \(\{\varepsilon\}\) (the language containing only the empty word):

  \[ L\{\varepsilon\} = L = \{\varepsilon\}L \]
Properties of Language Concatenation (2)

- Concatenation of languages distributes through set union:

\[ L (M \cup N) = LM \cup LN \]
\[ (L \cup M) N = LN \cup MN \]

- But it **does not** distribute through set intersection:

\[ L (M \cap N) \neq LM \cap LN \]

- Counterexample:

\[ L = \{ \varepsilon, a \}, M = \{ \varepsilon \}, N = \{ a \} \]
\[ L (M \cap N) = L\emptyset = \emptyset \]
\[ LM \cap LN = \{ \varepsilon, a \} \cap \{ a, aa \} = \{ a \} \]
A language can be concatenated with itself.

Exponent notation is often used for this:
- $L^1 = L$
- $L^2 = LL$
- $L^3 = LLL$
- $L^4 = LLLL$
- etc.

$L^0$ is defined to be $\{\varepsilon\}$.
(As $\{\varepsilon\}$ is the unit of concatenation.)
Kleene Star

- Given $L \subseteq \Sigma^*$, $L^*$ is zero or more concatenations of $L$.
- Note that these are different stars (but both mean ‘zero or more’).

$$L^* = \{w_0 w_1 \ldots w_{n-1} \mid n, i \in \mathbb{N}, \forall i < n, w_i \in L\}$$

or

$$L^* = \bigcup_{n=0}^{\infty} L^n = L^0 \cup L^1 \cup L^2 \cup \ldots$$

or

$$\varepsilon \in L^*$$

$$w \in L \quad \Rightarrow \quad w \in L^*$$

$$v \in L^* \land w \in L^* \quad \Rightarrow \quad vw \in L^*$$
Language Membership

- Fundamental question of this module:
  
  Given a language \( L \subseteq \Sigma^* \) and a word \( w \in \Sigma^* \), can we determine if \( w \in L \)?

- If \( L \) is finite, this is easy.

- But not so easy if \( L \) is infinite, which most interesting languages are.

- We need:
  
  - A finite (and preferably concise) description of the (infinite) language.
  
  - A method to decide if \( w \in L \) or not, given such a description.

- Over the course of this module we are going to encounter a number of possibilities, with varying descriptive power.
Recommended Reading

- Introduction to Automata Theory, Languages, and Computation (3rd edition), pages 28–33.
- G52MAL Lecture Notes, page 6.