Formal Languages
**Alphabet and Word**

**Definition**

**Alphabet** is a nonempty finite set of **symbols**.

**Remark:** An alphabet is often denoted by the symbol $\Sigma$ (upper case sigma) of the Greek alphabet.

**Definition**

A **word** over a given alphabet is a finite sequence of symbols from this alphabet.

**Example 1:**


Words over alphabet $\Sigma$: HELLO ABRACADABRA ERROR
Example 2:


A word over alphabet \(\Sigma_2\): HELLO\underline{\square}WORLD

Example 3:

\[\Sigma_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\]

Words over alphabet \(\Sigma_3\): 0, 31415926536, 65536

Example 4:

Words over alphabet \(\Sigma_4 = \{0, 1\}\): 011010001, 111, 1010101010101010

Example 5:

Words over alphabet \(\Sigma_5 = \{a, b\}\): aababb, abbabbba, aaab
Alphabet and Word

Example 6:

Alphabet $\Sigma_6$ is the set of all ASCII characters.

Example of a word:

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```
Language — a set of (some) words of symbols from a given alphabet

Examples of problem types, where theory of formal languages is useful:

- Construction of compilers:
  - Lexical analysis
  - Syntactic analysis

- Searching in text:
  - Searching for a given text pattern
  - Searching for a part of text specified by a regular expression
To describe a language, there are several possibilities:

- We can enumerate all words of the language (however, this is possible only for small finite languages).

  **Example:** \( L = \{aab, babba, aaaaaa\} \)

- We can specify a property of the words of the language:

  **Example:** The language over alphabet \( \{0, 1\} \) containing all words with even number of occurrences of symbol 1.
In particular, the following two approaches are used in the theory of formal languages:

- To describe an (idealized) machine, device, algorithm, that recognizes words of the given language – approaches based on **automata**.

- To describe some mechanism that allows to generate all words of the given language – approaches based on **grammars** or **regular expressions**.
Some Basic Concepts

The **length of a word** is the number of symbols of the word.
For example, the length of word *abaab* is 5.

The length of a word $w$ is denoted $|w|$.
For example, if $w = abaab$ then $|w| = 5$.

We denote the number of occurrences of a symbol $a$ in a word $w$ by $|w|_a$.
For word $w = ababb$ we have $|w|_a = 2$ and $|w|_b = 3$.

An **empty word** is a word of length 0, i.e., the word containing no symbols.
The empty word is denoted by the letter $\varepsilon$ (epsilon) of the Greek alphabet.
(Remark: In literature, sometimes $\lambda$ (lambda) is used to denote the empty word instead of $\varepsilon$.)

$$|\varepsilon| = 0$$
Concatenation of Words

One of operations we can do on words is the operation of **concatenation**: For example, the concatenation of words **OST** and **RAVA** is the word **OSTRAVA**.

The operation of concatenation is denoted by symbol \( \cdot \) (similarly to multiplication). It is possible to omit this symbol.

\[
\text{OST} \cdot \text{RAVA} = \text{OSTRAVA}
\]

Concatenation is **associative**, i.e., for every three words \( u, v, \) and \( w \) we have

\[
(u \cdot v) \cdot w = u \cdot (v \cdot w)
\]

which means that we can omit parenthesis when we write multiple concatenations. For example, we can write \( w_1 \cdot w_2 \cdot w_3 \cdot w_4 \cdot w_5 \) instead of \( (w_1 \cdot (w_2 \cdot w_3)) \cdot (w_4 \cdot w_5) \).
Concatenation of Words

Concatenation is not **commutative**, i.e., the following equality does not hold in general

\[ u \cdot v = v \cdot u \]

**Example:**

\[ \text{OST} \cdot \text{RAVA} \neq \text{RAVA} \cdot \text{OST} \]

It is obvious that the following holds for any words \( v \) and \( w \):

\[ |v \cdot w| = |v| + |w| \]

For every word \( w \) we also have:

\[ \varepsilon \cdot w = w \cdot \varepsilon = w \]
Prefixes, Suffixes, and Subwords

Definition

A word $x$ is a **prefix** of a word $y$, if there exists a word $v$ such that $y = xv$.

A word $x$ is a **suffix** of a word $y$, if there exists a word $u$ such that $y = ux$.

A word $x$ is a **subword** of a word $y$, if there exist words $u$ and $v$ such that $y = uxv$.

Example:

- Prefixes of the word **abaab** are $\varepsilon$, $a$, $ab$, $aba$, $abaa$, $abaab$.
- Suffixes of the word **abaab** are $\varepsilon$, $b$, $ab$, $aab$, $baab$, $abaab$.
- Subwords of the word **abaab** are $\varepsilon$, $a$, $b$, $ab$, $ba$, $aa$, $aba$, $baa$, $aab$, $abaa$, $baab$, $abaab$. 
The set of all words over alphabet $\Sigma$ is denoted $\Sigma^*$. 

**Definition**

A *(formal) language* $L$ over an alphabet $\Sigma$ is a subset of $\Sigma^*$, i.e., $L \subseteq \Sigma^*$. 

**Example 1:** The set $\{00, 01001, 1101\}$ is a language over alphabet $\{0, 1\}$. 

**Example 2:** The set of all syntactically correct programs in the C programming language is a language over the alphabet consisting of all ASCII characters. 

**Example 3:** The set of all texts containing the sequence *hello* is a language over alphabet consisting of all ASCII characters.
Set Operations on Languages

Since languages are sets, we can apply any set operations to them:

**Union** – \( L_1 \cup L_2 \) is the language consisting of the words belonging to language \( L_1 \) or to language \( L_2 \) (or to both of them).

**Intersection** – \( L_1 \cap L_2 \) is the language consisting of the words belonging to language \( L_1 \) and also to language \( L_2 \).

**Complement** – \( \overline{L_1} \) is the language containing those words from \( \Sigma^* \) that do not belong to \( L_1 \).

**Difference** – \( L_1 - L_2 \) is the language containing those words of \( L_1 \) that do not belong to \( L_2 \).

**Remark:** It is assumed the languages involved in these operations use the same alphabet \( \Sigma \).
Set Operations on Languages

Formally:

**Union:** \( L_1 \cup L_2 = \{ w \in \Sigma^* | w \in L_1 \lor w \in L_2 \} \)

**Intersection:** \( L_1 \cap L_2 = \{ w \in \Sigma^* | w \in L_1 \land w \in L_2 \} \)

**Complement:** \( \overline{L_1} = \{ w \in \Sigma^* | w \notin L_1 \} \)

**Difference:** \( L_1 - L_2 = \{ w \in \Sigma^* | w \in L_1 \land w \notin L_2 \} \)

**Remark:** We assume that \( L_1, L_2 \subseteq \Sigma^* \) for some given alphabet \( \Sigma \).
Set Operations on Languages

Example:

Consider languages over alphabet \( \{a, b\} \).

- \( L_1 \) — the set of all words containing subword \( baa \)
- \( L_2 \) — the set of all words with an even number of occurrences of symbol \( b \)

Then

- \( L_1 \cup L_2 \) — the set of all words containing subword \( baa \) or an even number of occurrences of \( b \)
- \( L_1 \cap L_2 \) — the set of all words containing subword \( baa \) and an even number of occurrences of \( b \)
- \( \overline{L_1} \) — the set of all words that do not contain subword \( baa \)
- \( L_1 - L_2 \) — the set of all words that contain subword \( baa \) but do not contain an even number of occurrences of \( b \)
Concatenation of Languages

**Definition**

**Concatenation of languages** $L_1$ and $L_2$, where $L_1, L_2 \subseteq \Sigma^*$, is the language $L \subseteq \Sigma^*$ such that for each $w \in \Sigma^*$ it holds that

$$w \in L \iff (\exists u \in L_1)(\exists v \in L_2)(w = u \cdot v)$$

The concatenation of languages $L_1$ and $L_2$ is denoted $L_1 \cdot L_2$.

**Example:**

$$L_1 = \{abb, ba\}$$
$$L_2 = \{a, ab, bbb\}$$

The language $L_1 \cdot L_2$ contains the following words:

```
abba  abbab  abbbbb  baa  baab  babbb
```
Iteration of a Language

**Definition**

The **iteration (Kleene star) of language** $L$, denoted $L^*$, is the language consisting of words created by concatenation of some arbitrary number of words from language $L$.

I.e. $w \in L^*$ iff

$$\exists n \in \mathbb{N} : \exists w_1, w_2, \ldots, w_n \in L : w = w_1w_2 \cdots w_n$$

**Example:** $L = \{aa, b\}$

$$L^* = \{\varepsilon, aa, b, aaaa, aab, baa, bb, aaaaaa, aaaaab, aabaa, aabb, \ldots\}$$

**Remark:** The number of concatenated words can be 0, which means that $\varepsilon \in L^*$ always holds (it does not matter if $\varepsilon \in L$ or not).
Iteration of a Language – Alternative Definition

At first, for a language \( L \) and a number \( k \in \mathbb{N} \) we define the language \( L^k \):

\[
L^0 = \{ \varepsilon \}, \quad L^k = L^{k-1} \cdot L \quad \text{for } k \geq 1
\]

This means

\[
\begin{align*}
L^0 &= \{ \varepsilon \} \\
L^1 &= L \\
L^2 &= L \cdot L \\
L^3 &= L \cdot L \cdot L \\
L^4 &= L \cdot L \cdot L \cdot L \\
L^5 &= L \cdot L \cdot L \cdot L \cdot L \\
&\quad \ldots
\end{align*}
\]

Example: For \( L = \{ aa, b \} \), the language \( L^3 \) contains the following words:

\[
\text{aaaaaa aaaaab aabaa aabb baaaa baab bbbaa bbb}
\]
Alternative definition

The iteration (Kleene star) of language $L$ is the language

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

Remark:

$$\bigcup_{k \geq 0} L^k = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$$
Remark: Sometimes, notation $L^+$ is used as an abbreviation for $L \cdot L^*$, i.e.,

$$L^+ = \bigcup_{k \geq 1} L^k$$
The **reverse** of a word $w$ is the word $w$ written from backwards (in the opposite order).

The reverse of a word $w$ is denoted $w^R$.

**Example:** $w = \text{HELLO}$ $w^R = \text{OLLEH}$

Formally, for $w = a_1a_2\cdots a_n$ (where $a_i \in \Sigma$) is $w^R = a_na_{n-1}\cdots a_1$. 
The **reverse** of a language $L$ is the language consisting of reverses of all words of $L$.

Reverse of a language $L$ is denoted $L^R$.

$$L^R = \{w^R \mid w \in L\}$$

**Example:** $L = \{ab, baaba, aaab\}$

$$L^R = \{ba, abaab, baaa\}$$
Order on Words

Let us assume some (linear) order $<$ on the symbols of alphabet $\Sigma$, i.e., if $\Sigma = \{a_1, a_2, \ldots, a_n\}$ then

$$a_1 < a_2 < \ldots < a_n.$$ 

**Example:** $\Sigma = \{a, b, c\}$ with $a < b < c$.

The following (linear) order $<_L$ can be defined on $\Sigma^*$:

$x <_L y$ iff:

- $|x| < |y|$, or
- $|x| = |y|$ there exist words $u, v, w \in \Sigma^*$ and symbols $a, b \in \Sigma$ such that

$$x = uav \quad y = ubw \quad a < b$$

Informally, we can say that in order $<_L$ we order words according to their length, and in case of the same length we order them lexicographically.
Order on Words

All words over alphabet $\Sigma$ can be ordered by $<_L$ into a sequence

$$w_0, w_1, w_2, \ldots$$

where every word $w \in \Sigma^*$ occurs exactly once, and where for each $i, j \in \mathbb{N}$ it holds that $w_i <_L w_j$ iff $i < j$.

**Example:** For alphabet $\Sigma = \{a, b, c\}$ (where $a < b < c$), the initial part of the sequence looks as follows:

$\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, aba, abb, abc, \ldots$

For example, when we talk about the first ten words of a language $L \subseteq \Sigma^*$, we mean ten words that belong to language $L$ and that are smallest of all words of $L$ according to order $<_L$. 
Regular Expressions
Regular expressions describing languages over an alphabet $\Sigma$:

- $\emptyset$, $\varepsilon$, $a$ (where $a \in \Sigma$) are regular expressions:
  - $\emptyset \ldots$ denotes the empty language
  - $\varepsilon \ldots$ denotes the language $\{\varepsilon\}$
  - $a \ldots$ denotes the language $\{a\}$

- If $\alpha$, $\beta$ are regular expressions then also $(\alpha + \beta)$, $(\alpha \cdot \beta)$, $(\alpha^*)$ are regular expressions:
  - $(\alpha + \beta) \ldots$ denotes the union of languages denoted $\alpha$ and $\beta$
  - $(\alpha \cdot \beta) \ldots$ denotes the concatenation of languages denoted $\alpha$ and $\beta$
  - $(\alpha^*) \ldots$ denotes the iteration of a language denoted $\alpha$

- There are no other regular expressions except those defined in the two points mentioned above.
**Example:** alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
- Since 0 is a regular expression, $(0^*)$ is also a regular expression.
Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, $0$ and $1$ are regular expressions.
- Since $0$ and $1$ are regular expression, $(0 + 1)$ is also a regular expression.
- Since $0$ is a regular expression, $(0^*)$ is also a regular expression.
- Since $(0 + 1)$ and $(0^*)$ are regular expressions, $((0 + 1) \cdot (0^*))$ is also a regular expression.
Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
- Since 0 is a regular expression, $(0^*)$ is also a regular expression.
- Since $(0 + 1)$ and $(0^*)$ are regular expressions, $((0 + 1) \cdot (0^*))$ is also a regular expression.

Remark: If $\alpha$ is a regular expression, by $[\alpha]$ we denote the language defined by the regular expression $\alpha$.

$$[[((0 + 1) \cdot (0^*))]] = \{0, 1, 00, 10, 000, 100, 0000, 1000, 00000, \ldots\}$$
The structure of a regular expression can be represented by an abstract syntax tree:
The formal definition of semantics of regular expressions:

- \([\emptyset] = \emptyset\)
- \([\varepsilon] = \{\varepsilon\}\)
- \([a] = \{a\}\)
- \([\alpha^*] = [\alpha]^*\)
- \([\alpha \cdot \beta] = [\alpha] \cdot [\beta]\)
- \([\alpha + \beta] = [\alpha] \cup [\beta]\)
To make regular expressions more lucid and succinct, we use the following conventions:

- The outward pair of parentheses can be omitted.
- We can omit parentheses that are superfluous due to associativity of operations of union (+) and concatenation (·).
- We can omit parentheses that are superfluous due to the defined priority of operators (iteration (·) has the highest priority, concatenation (·) has lower priority, and union (+) has the lowest priority).
- A dot denoting concatenation can be omitted.

**Example:** Instead of

\[
((((((0 \cdot 1)^*) \cdot 1) \cdot (1 \cdot 1)) + (((0 \cdot 0) + 1)^*)))
\]

we usually write

\[
(01)^*111 + (00 + 1)^*
\]
Examples: In all examples $\Sigma = \{0, 1\}$.

$0$ \ldots the language containing the only word $0$
Examples: In all examples $\Sigma = \{0, 1\}$.

0 \ldots the language containing the only word 0

01 \ldots the language containing the only word 01
Examples: In all examples $\Sigma = \{0, 1\}$.

- $0 \ldots$ the language containing the only word 0
- $01 \ldots$ the language containing the only word 01
- $0 + 1 \ldots$ the language containing two words 0 and 1
Examples: In all examples $\Sigma = \{0, 1\}$.

- $0 \ldots$ the language containing the only word $0$
- $01 \ldots$ the language containing the only word $01$
- $0 + 1 \ldots$ the language containing two words $0$ and $1$
- $0^* \ldots$ the language containing words $\epsilon, 0, 00, 000, \ldots$
Regular Expressions

**Examples:** In all examples $\Sigma = \{0, 1\}$.

- $0 \ldots$ the language containing the only word $0$
- $01 \ldots$ the language containing the only word $01$
- $0 + 1 \ldots$ the language containing two words $0$ and $1$
- $0^* \ldots$ the language containing words $\varepsilon, 0, 00, 000, \ldots$
- $(01)^* \ldots$ the language containing words $\varepsilon, 01, 0101, 010101, \ldots$
**Regular Expressions**

**Examples:** In all examples $\Sigma = \{0, 1\}$.

- $0$  
  the language containing the only word $0$
- $01$  
  the language containing the only word $01$
- $0 + 1$  
  the language containing two words $0$ and $1$
- $0*$  
  the language containing words $\epsilon, 0, 00, 000, \ldots$
- $(01)^*$  
  the language containing words $\epsilon, 01, 0101, 010101, \ldots$
- $(0 + 1)^*$  
  the language containing all words over the alphabet $\{0, 1\}$
Examples: In all examples $\Sigma = \{0, 1\}$.

- $0$ ... the language containing the only word 0
- $01$ ... the language containing the only word 01
- $0 + 1$ ... the language containing two words 0 and 1
- $0^*$ ... the language containing words $\varepsilon$, 0, 00, 000, ...  
- $(01)^*$ ... the language containing words $\varepsilon$, 01, 0101, 010101, ...
- $(0 + 1)^*$ ... the language containing all words over the alphabet $\{0, 1\}$
- $(0 + 1)^*00$ ... the language containing all words ending with 00
Regular Expressions

Examples: In all examples $\Sigma = \{0, 1\}$.

- $0$ \ldots the language containing the only word $0$
- $01$ \ldots the language containing the only word $01$
- $0 + 1$ \ldots the language containing two words $0$ and $1$
- $0^*$ \ldots the language containing words $\varepsilon$, $0$, $00$, $000$, $\ldots$
- $(01)^*$ \ldots the language containing words $\varepsilon$, $01$, $0101$, $010101$, $\ldots$
- $(0 + 1)^*$ \ldots the language containing all words over the alphabet $\{0, 1\}$
- $(0 + 1)^*00$ \ldots the language containing all words ending with $00$
- $(01)^*111(01)^*$ \ldots the language containing all words that contain a subword $111$ preceded and followed by an arbitrary number of copies of the word $01$
\((0 + 1)^*00 + (01)^*111(01)^*\) … the language containing all words that either end with 00 or contain a subwords 111 preceded and followed with some arbitrary number of words 01
\((0 + 1)^*00 + (01)^*111(01)^*\) \ldots the language containing all words that either end with 00 or contain a subwords 111 preceded and followed with some arbitrary number of words 01

\((0 + 1)^*1(0 + 1)^*\) \ldots the language of all words that contain at least one occurrence of symbol 1
\((0 + 1)^*00 + (01)^*111(01)^*\) \ldots the language containing all words that either end with 00 or contain a subwords 111 preceded and followed with some arbitrary number of words 01

\((0 + 1)^*1(0 + 1)^*\) \ldots the language of all words that contain at least one occurrence of symbol 1

\(0^*(10^*10^*)^*\) \ldots the language containing all words with an even number of occurrences of symbol 1