A language is a set of strings, or of numbers or other types of objects encoded as strings.

Example: A graph is an object $G = (V, E)$ where
- $V$ is a set of elements called vertices or nodes.
- $E$, the edge relation, is a subset of $V \times V$.

The graph is undirected if $E$ is a symmetric relation:
1. $(\forall u, v \in V) \ (u, v) \in E \Leftrightarrow (v, u) \in E$.

By default, undirected graphs have no self-loops:
2. $(\forall u \in V) \ (u, u) \notin E \quad [E \text{ is "completely irreflexive"}].$

The graph is transitive — i.e., has a transitive edge relation — if
3. $(\forall u, v, w \in V) \ [(u, v) \in E \land (v, w) \in E \rightarrow (u, w) \in E]$
   i.e., $E(\{u, v\}) \land E(\{v, w\}) \rightarrow E(\{u, w\}).$

Example: $V = \{1, 2, 3, 4, 5\}$

$E = \{ \{(1, 3), (3, 1), (2, 3), (3, 2)\},$
\[
\{(1, 4), (4, 1), (2, 5), (5, 2)\},
\{(3, 4), (4, 3), (3, 5), (5, 3)\} \}
\]
A string encoding of the graph.

Example to 3: $u = 1, v = 3, w = 2$

Property 4: $\exists u, v, w \in V \ (u, v) \in E \land (v, w) \in E \land (w, u) \in E.$

This property says: "The graph $G = (V, E)$ has a triangle."
We can define both a computational problem and a formal language around this property. Define

$$L_{\Delta} = \{ \text{string encodings } G \mid G: G \text{ has a triangle } \}$$

Example

$$G = \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array}$$

Second solution: Consider $L_{\Delta}$ already limited to the domain of undirected graph without self loops.

Undirected graph:

- Basic type of graph
- $E = \{(1,1), (1,2), (2,3), (2,4), (2,1), (3,2), (4,2)\}$

Directed graph:

- Basic type of directed graph which do allow self-loops.
- $G^1 = \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array}$
- $E^1 = \{(1,1), (1,2), (2,3), (2,4)\}$ (no more self-loops)
- $G^2 = \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array}$
- $E'' = E' \cup \{(3,4)\}$, $V'' = V' = V$
- Not a directed triangle

$L_{\Delta} = \{ \text{digraphs } G \mid G \text{ has a triangle } \}$

Ultimately, $L_{\Delta}$ and $L_{\Delta}$ are languages composed of strings, namely, the strings by which we would encode graphs for algorithms, programs.
Operations on Strings and Languages As Sets of Strings.

I. Basic Set operations: Union, Intersection, Complement, other... 

Capitol letters: \( A, B, C, \ldots \), \( L, X, Y, Z \) languages.

Given any languages \( A \cap B \in \Sigma^* \) general type of strings over the alphabet \( \Sigma \).

Complement: \( \overline{L} = \{ x \in \Sigma^* : x \notin L \} \).

(footnote a)

\(~L_A = \{ \) all “whatsits” that are not undirected graphs with a triangle \( “101” \) belongs \( “whatsits” \) are any binary strings.

General: \( \overline{L} = \Sigma^* \setminus L \) difference of sets - intact

\(~L_A = \{ \) undirected graphs \( G: G \) does not have a triangle \( \).

Type specific = \( Ugraphs \setminus L_A \).

Universal \( U \) can = \( \Sigma^* \) by default, or can be specified as a basic type for a problem.

Difference: \( L \setminus B = \{ x : x \in L \land x \notin B \} \).

Symmetric difference \( \Lambda A B = (L \setminus B) \cup (B \setminus L) \),

also written \( \Lambda A B = \{ x : x \in L \leftrightarrow x \in B \} \).

\( \times \): Cartesian product

\( A \times B = \{ (a, b): a \in A \land b \in B \} \).

(*) Text uses overbar \( \overline{L} \) for complement, but \( \sim \) can go nicely in print too, so I use it.
II. Operations Specific to Strings.

**Concatenation** \( x \cdot y = x \) followed by \( y \).

- This lifts to a concatenation of languages.
- Usually \( y \cdot x \neq x \cdot y \); \( y \cdot x = 11010 \).

**A \cdot B = \{ x \cdot y : x \in A \text{ and } y \in B \}**. 

**A \cdot B \subseteq \Sigma^* = \{ w \in \Sigma^* : w \text{ can be broken up } w = x \cdot y \text{ such that } x \in A \text{ and } y \in B \}**. 

**A \times B = \{ (x, y) : x \in A \text{ and } y \in B \}**. 

**Example**:

\( A = \{ 0, 01 \} \)
\( B = \{ 00, 100 \} \).

\( A \cdot B = \{ 0.00, 0.100, 01.00, 01.100 \} \)
\( = \{ 000, 0100, 0100, 01100 \} \) \[ \text{collapse} \]

\( A \times B = \{ (0,00), (0,100), (01,00), (01,100) \} \) \[ \text{no collapse} \]

\( A \cdot B = \{ 0\#100, 0\#00, 01\#00, 01\#100 \} \)

four different strings over alphabet \( \Sigma^* = \{ 0, 1, \# \} \).

*Addendum:* The \# symbol is a "loud comma". Using it helps avoid ambiguities when one string is a prefix of another, like "0" is a prefix of "01" as members of \( A \).

If \( A \) is prefix-free (text, p14), then for any \( B \), \( A \cdot B \) has no "collapse."