Lecture Thu 1/28 CSE 396 Spr 2016

A formal language is a finite or infinite set of {words / 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 $(\forall u, v \in V) \quad (u,v) \in E \Leftrightarrow (v,u) \in E.$

By default, undirected graphs have no self-loops:

② $(\forall u \in V) \quad (u,u) \notin E$ [E is "completely irreflexive"]

The graph is transitive — i.e. has a transitive edge relation — if

③ $(\forall u, v, w \in V)\ [(u,v) \in E \wedge (v,w) \in E \rightarrow (u,w) \in E]$.
ie. $E(u,v) \wedge E(v,w) \rightarrow E(u,w)$.

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.

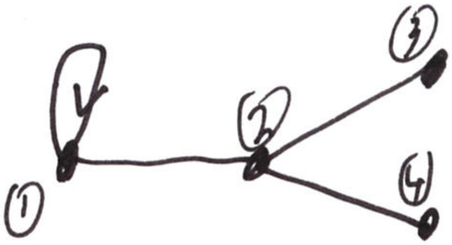
unterexample to ③: $u=1, v=3, w=2$

property: ④: $(\exists u, v, w \in V)\ (u,v) \in E \wedge (v,w) \in E \wedge (w,u) \in E$."
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

$L_\Delta = \{$ string encodings of graphs $(V,E) = G$: G <u>has a triangle</u> $\}$

Example

$G =$ 

add ↓

i.e. $\boxed{4}$ $(\exists u,v,w \in V)\ E(u,v) \wedge E(v,w) \wedge u \neq v \wedge u \neq w \wedge v \neq w \wedge (E(w,u))$.

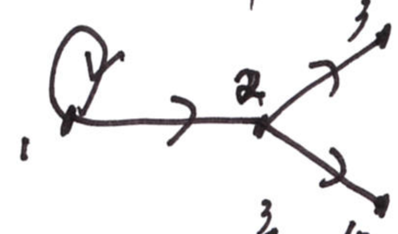
$E = \{(1,1), (1,2), (2,3), (2,4), (2,1), (3,2), (4,2)\}$.

Second solution: Consider L_Δ already limited to the <u>domain</u> of $\boxed{\text{undirected graphs without self loops}}$ as a <u>basic type</u>, eg. "<u>ugraph</u>".

$\boxed{4}$ <u>is</u> <u>satisfied</u> by $u=1$, $v=1$, $w=2$

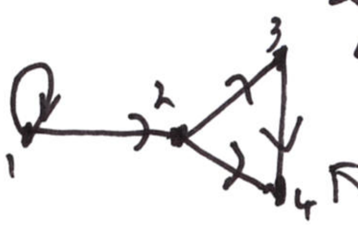
G is not a <u>ugraph</u>, so "fuggedaboudit."

<u>digraph</u>: Basic type of <u>directed graphs</u> which <u>do</u> allow self-loops.

$G' =$ 

$E' = \{(1,1), (1,2), (2,3), (2,4)\}$. (no more)

Still, $1 \to 1 \to 1 \to 1$ is an issue ☹.

$G'' =$ 

$E'' = E' \cup \{(3,4)\}$. $V'' = V' = V$

not a directed triangle.

$L_{D\Delta} = \{$ <u>digraphs</u> G : G has a triangle $\}$. ← understood as a directed triangle

Ultimately L_Δ and $L_{D\Delta}$ are languages composed of strings namely, the strings by which we would encode graphs for algorithms, programs. But it helps to think of L_Δ as a language of undirected graphs and $L_{D\Delta}$ as having the different <u>type</u> <u>domain</u> of digraphs.

Operations on Strings and Languages As Sets of Strings.

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

Capital letters $A, B, C, \ldots L,$ X, Y, Z: languages.

$$L = A \cup B \qquad L' = A \cap B$$

given any languages $A, B \subseteq \underbrace{\Sigma^*}$ general type of strings over the alphabet Σ.

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

$\sim L_\Delta = \{$all "whatsits" that are not undirected graphs with a triangle$\}$

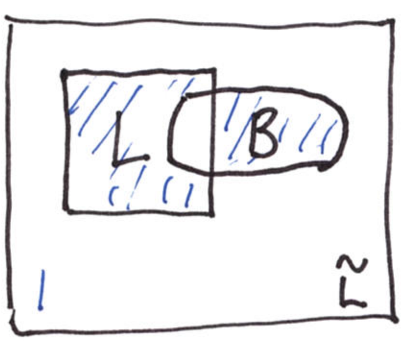
"101" belongs if "whatsits" are any binary strings

OR

General: $\tilde{L} = \boxed{\Sigma^* \setminus L}$ difference of sets — in text

"$\sim L_\Delta$" $= \{$ugraphs G: G does not have a triangle$\}$

V



Type-specific $= \boxed{\text{Ugraphs} \setminus L_\Delta}$.

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

$L \Delta B$

Difference: $L \setminus B = \{x : x \in L \wedge x \notin B\}$.

* Symmetric Difference $L \Delta B = (L \setminus B) \cup (B \setminus L) =$

also written $L \oplus B = \{x : x \in L \text{ XOR } x \in B\}$.

$= (L \cup B) \setminus (L \cap B)$

I': Cartesian Product

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

(*) Text uses overbar $\overline{L}$ for complement, but $\sim$ can go nicely in front too, so I use it.

II. Operations Specific to Strings.

<u>Concatenation</u> $x \cdot y$ = x followed by y.

"010" · "11" = "01011".

⚠ Usually $y \cdot x \neq x \cdot y$; $y \cdot x =$ "11010"

★ This <u>lifts</u> to a concatenation op on languages.

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

Corresponds to the idea of "And Then".

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

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

Same as $A \cdot B$? $|A \times B| = |A \cdot B|$? <u>no</u>.

<u>Ex</u>: $A = \{ 0, 01 \}$

$B = \{ 00, 100 \}$.

$A \cdot B = \{ 0 \cdot 00, 0 \cdot 100, 01 \cdot 00, 01 \cdot 100 \}$

$= \{ 000, \underbrace{0100, 0100}_{\text{"collapse"}}, 01100 \} = \{ 000, 0100, 01100 \}$

$A \times B = \{ (0,100), (0,00), (01,00), (01,100) \}$ no collapse.

$A \# B = \{ 0\#100, 0\#00, 01\#00, 01\#100 \}$

four different strings over alphabet $\Sigma' = \{ 0, 1, \# \}$.

<u>Added</u>: The # symbol is a "loud comma". Using it helps avoid ambiguities when one string is a <u>prefix</u> of another, like "0" is of "01" as members of A. If A is <u>prefix-free</u> (text, p14), then for any B, $A \cdot B$ has <u>no</u> "collapse".