Welcome to COMS W3261 CS Theory!

Instructors: Tal Malkin (section 1 & 2)
William Pires (section 3)

Today: - Intro to class - Administrative + content
- some formal definitions

www.cs.columbia.edu/~tal/3261/fall26/

[Notes posted jointly for section 1+2, very lightly edited.

We went over the class webpage to discuss some logistics, grading, and motivation for the class.
Note that some parts of the webpage are still in progress]

Which of these questions are EASY / HARD / IMPOSSIBLE to solve?

1) on input an integer, is it a prime?
2) on input a sequence of 0s and 1s, is the number of 0s prime?
3) on input a sequence of 0s and 1s, is the number of 0's even?
4) on input a sequence of characters from {A, B, ..., Z, blank} is this a grammatical English sentence?
5) given a C program, does it compile?
6) given a C program, does it solve problem (1) above?
7) given a C program, is there an input that sends it into an infinite loop?
8) Given a neural network and an input image, does the network classify the image as a cat?
9) Given a neural network and an input image, is there a very close image that is classified differently?

[we had a nice discussion about some of these, students' intuitions, and about how EASY/HARD/IMPOSSIBLE may have different interpretations]

Definitions:

- an **alphabet** Σ is a finite, non-empty set of symbols. (aka "characters" or "letters")

eg $\Sigma = \{0,1\} = \{1,0\}$

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

- A **string** over an alphabet Σ is a finite sequence of characters from Σ.

eg 11011001 is a string over $\{0,1\}$

abba " " " $\{a,b,y\}$

$\neq$ baba

- The **length of** a string s, denoted $|s|$, is the number of characters in the string

eg. $|abba| = 4$

- The **empty string**, denoted ε, is the string of length 0

$$|\varepsilon| = 0$$

- The **concatenation** of strings s, t, denoted st (or $s \| t$) is the string st

eg $1 \circ 001 = 1001 = 10 \circ 01$

$= \varepsilon \circ 1001$

$\neq$

$001 \circ 1 = 0011$

- Σ^k for $k \geq 0$ is the set of all strings over Σ that are of length k

$= \{ s \mid s \text{ is a string over } \Sigma \text{ and } |s| = k \}$

eg for $\Sigma = \{0, 1\}$

$\Sigma^0 = \{ \varepsilon \}$

$\Sigma^1 = \{ 0, 1 \}$

$\Sigma^2 = \{ 01, 10, 00, 11 \}$

- $\Sigma^* = \{ \text{all strings over } \Sigma \} = \bigcup_{k=0}^{\infty} \Sigma^k$

eg $\{0,1\}^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \}$

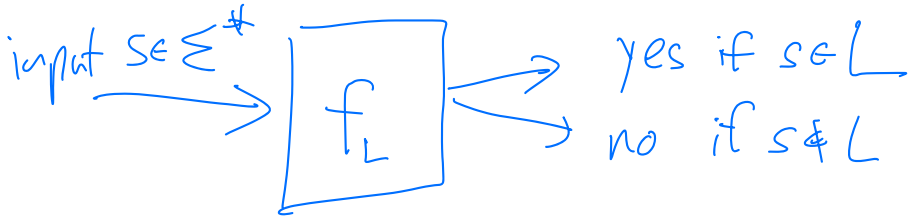
this is an infinite set - the first def today that can be infinite size.

- A language over Σ is a set of strings

$L \subseteq \Sigma^*$

We will use "language" to represent yes/no problems

every language corresponds to a function $f_L : \Sigma^* \to \{yes, no\}$ (or $\{0,1\}$)



A language is "easy to solve" (within some computational model) if we can write a algorithm/program implementing this f_L

(i.e. given $s \in \Sigma^*$, we can decide whether $s \in L$)

Example languages → are these easy or hard?

- $L = \Sigma^*$
- $L = \{\} = \emptyset$
- $L = \{00, 1, \varepsilon, 01\}$
- $L = \{hello\}$
- $L = \{\varepsilon\}$
- $L = \{$all words in English dictionary$\}$
- $L = \{s \in \{0,1\}^* \mid s$ has an even # of 0s$\}$

To show that a language L is solvable (within some model of computation) you need to show you can implement an algorithm that can determine whether or not an input is in L.

In section 1 we wrote down algorithms for some of the examples (see below). For section 2 we didn't, but I told students to think about it - we'll review next time.

Example languages

- $L = \Sigma^*$ → Easy! Algorithm: On input s, output yes
- $L = \{\} = \emptyset$ → Algorithm: On input s, output no
- $L = \{00, 1, \varepsilon, 01\}$ → Algorithm: on input s, if $s = 00$ output yes, else if $s = 1$ output yes, else if $s = \varepsilon$ output yes, else if $s = 01$ output yes, else output no.

  similar:
  - $L = \{\text{hello}\}$
  - $L = \{\varepsilon\}$
  - $L = \{\text{all words in English dictionary}\}$
- $L = \{s \in \{0,1\}^* \mid s \text{ has an even \# of 0's}\}$

  → Algorithm: On input s:
  - go over s, count how many 0's it has
  - if this number is 0 mod 2 output yes, else output no

[There are many other algorithms for each of these. Eg see notes from section 3 for different algorithms for the last L]