BU CS 332 – Theory of Computation

Link to polls:
https://forms.gle/exHaBu5mQgR57NDq5

Lecture 2:

- Parts of a Theory of Computation
- Sets, Strings, and Languages
- Start Finite Automata?

Reading:

Sipser Ch 0

Reminders:
- HW0 due tonight, HW1 out
- Lecture 1 check-in due on Gradescope

Mark Bun

September 8, 2022

What makes a good theory?

- General ideas that apply to many different systems
- Expressed simply, abstractly, and precisely

Parts of a Theory of Computation

- Models for machines (computational devices)
- Models for the problems machines can be used to solve
- Theorems about what kinds of machines can solve what kinds of problems, and at what cost

What is a (Computational) Problem?

For us: A problem will be the task of recognizing whether a *string* is in a *language*

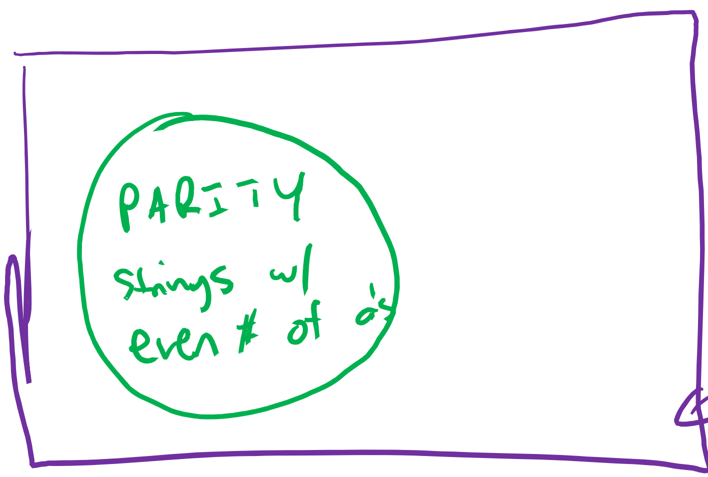
E.g. Parity: Given a string of a's and b's, does it contain an even number of a's?

Alphabet: $\Sigma_1 = \{a, b\}$

Given: A string $x \in \Sigma_1^*$

Goal: Determine if x has even # of a's

$\iff$ is x a member of the language PARITY?



Σ_1^* = all strings over $\{a, b\}$

What is a (Computational) Problem?

For us: A problem will be the task of recognizing whether a *string* is in a *language*

- **Alphabet**: A finite set Σ Ex. $\Sigma = \{\mathrm{a}, \mathrm{b}\}$
- **String:** A finite concatenation of alphabet symbols
  Ex. $\mathrm{bba}, \mathrm{ababb}$

  ε denotes empty string, length 0

  Σ^* = set of all strings using symbols from Σ

  Ex. $\{\mathrm{a}, \mathrm{b}\}^* = \{\varepsilon, \mathrm{a}, \mathrm{b}, \mathrm{aa}, \mathrm{ab}, \mathrm{ba}, \mathrm{bb}, \dots\}$
- **Language:** A set $L \subseteq \Sigma^*$ of strings

Examples of Languages

"|" = "such that"

Parity: Given a string consisting of a's and b's, does it contain an even number of a's?

$\Sigma = \{a, b\}$ $L = \{ x \in \{a,b\}^* \mid x \text{ has an even \# of a's} \}$

$\epsilon \in L$ (empty string has even # of a's)

Primality: Given a natural number x (represented in binary), is x prime?

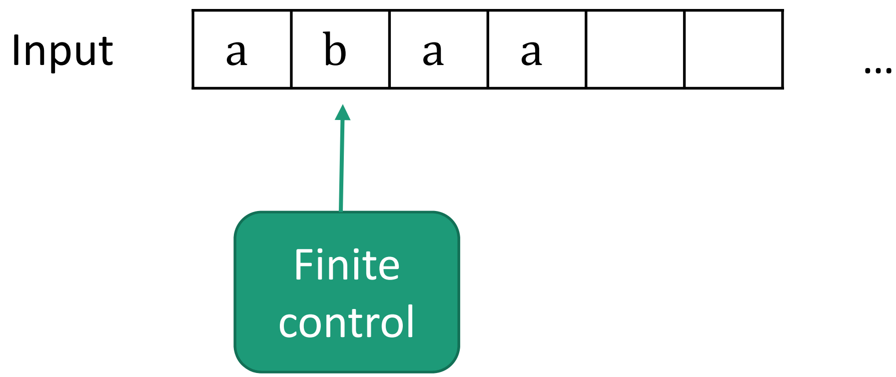
$\Sigma = \{0, 1\}$ $L = \{ x \in \{0,1\}^* \mid x \text{ is the binary representation of a prime} \}$

Halting Problem: Given a C program, can it ever get stuck in an infinite loop?

$\Sigma =$ ASCII $L = \{ P \in \Sigma^* \mid P \text{ describes a C program that loops forever on some input} \}$

Machine Models

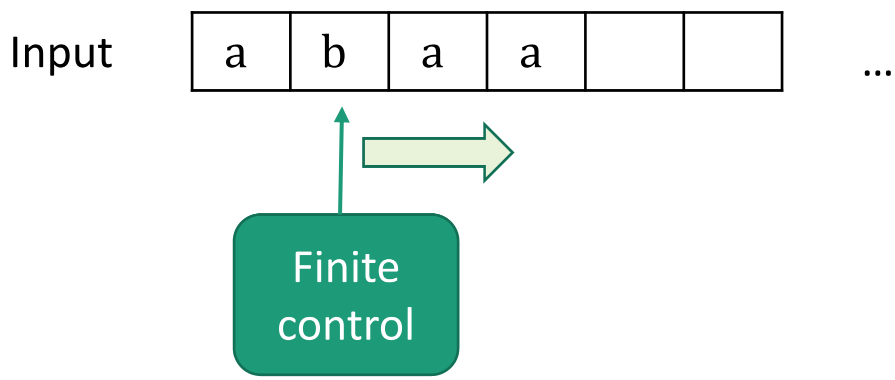
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



Abstraction: We don't care how the control is implemented. We just require it to have a finite number of states, and to transition between states using fixed rules.

Machine Models

- <u>Finite Automata (FAs):</u> Machine with a finite amount of unstructured memory

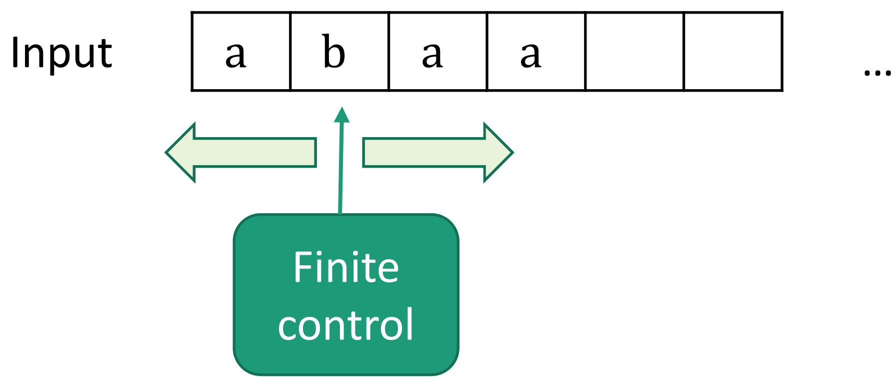


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

Useful for modeling chips, simple control systems, choose-your-own adventure games…

Machine Models

- <u>Turing Machines (TMs):</u> Machine with unbounded, unstructured memory



Control can scan in both directions
Control can both read and <u>write</u>

Model for general sequential computation

Church-Turing Thesis: Everything we intuitively think of as "computable" is computable by a Turing Machine

What theorems would we like to prove?

We will define classes of languages based on which machines can recognize them

Inclusion: Every language recognizable by a FA is also recognizable by a TM

Non-inclusion: There exist languages recognizable by TMs which are not recognizable by FAs

Completeness: Identify a “hardest” language in a class

Robustness: Alternative definitions of the same class

Ex. Languages recognizable by FAs = regular expressions

Why study theory of computation?

- You'll learn how to formally reason about computation
- You'll learn the technology-independent foundations of CS

Philosophically interesting questions:

- Are there well-defined problems which cannot be solved by computers?
- Can we always find the solution to a puzzle faster than trying all possibilities?
- Can we say what it means for one problem to be "harder" than another?

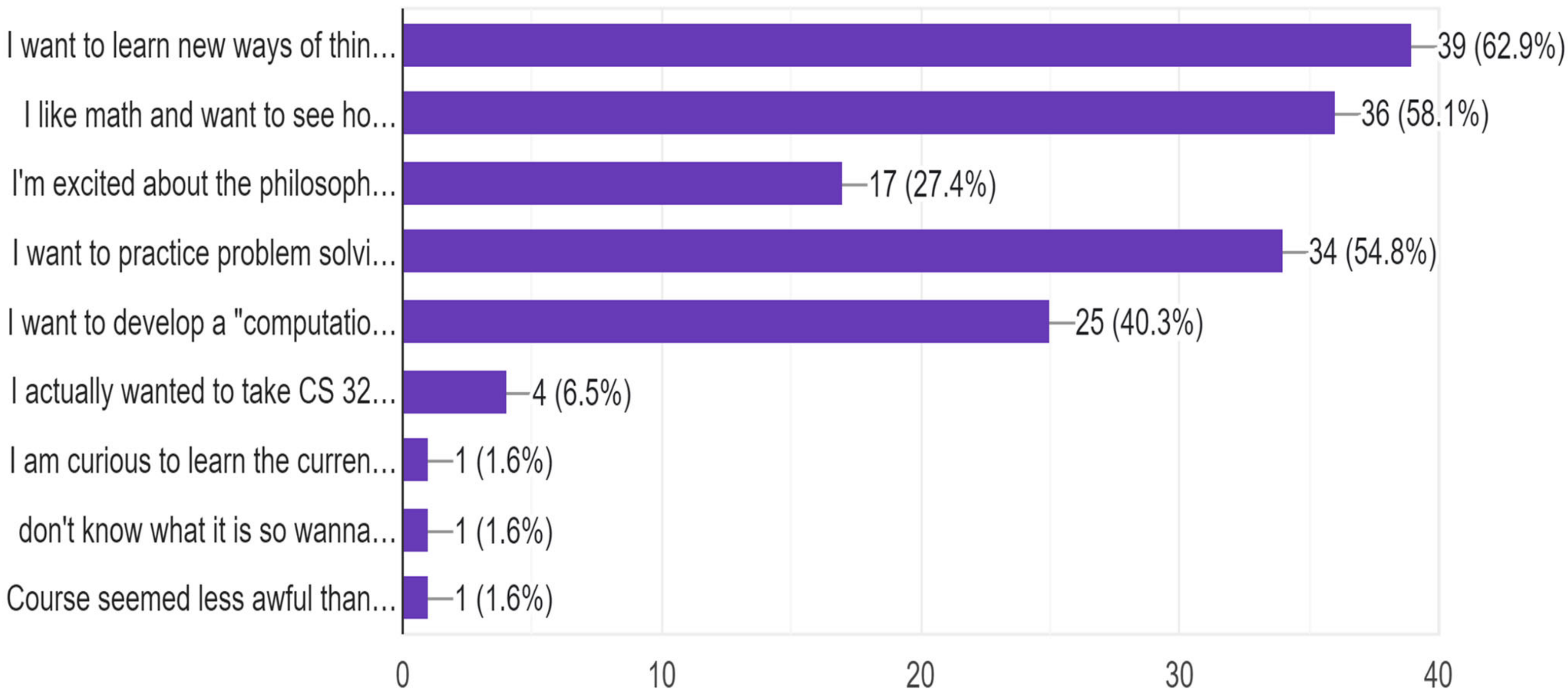
Why study theory of computation?

- You'll learn how to formally reason about computation
- You'll learn the technology-independent foundations of CS

Connections to other parts of science:

- Finite automata arise in compilers, AI, coding, chemistry
https://cstheory.stackexchange.com/a/14818
- Hard problems are essential to cryptography
- Computation occurs in cells/DNA, the brain, economic systems, physical systems, social networks, etc.

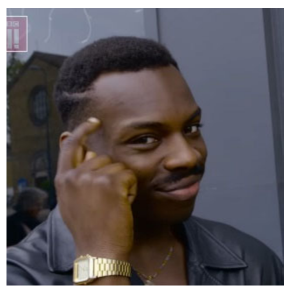
What appeals to you about the theory of computation?



Why study theory of computation?

Practical knowledge for developers

"Boss, I can't find an efficient algorithm.
I guess I'm just too dumb."



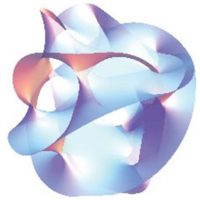
"Boss, I can't find an efficient algorithm
because no such algorithm exists."

Will you be asked about this material on job interviews?

No promises, but a true story…

More about strings and languages

String Theory



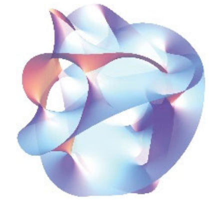
- **Symbol:** Ex. $\mathrm{a, b, 0, 1}$
- **Alphabet**: A finite set Σ Ex. $\Sigma = \{\mathrm{a, b}\}$
- **String:** A finite concatenation of alphabet symbols
  Ex. $\mathrm{bba, ababb}$

  ε denotes empty string, length 0

  Σ^* = set of all strings using symbols from Σ

  Ex. $\{\mathrm{a, b}\}^* = \{\varepsilon, \mathrm{a, b, aa, ab, ba, bb}, \ldots\}$
- **Language:** A set $L \subseteq \Sigma^*$ of strings

String Theory



- **Length** of a string, written $|x|$, is the number of symbols

Ex. $|\text{abba}| =$ 4 $\qquad |\varepsilon| =$ 0

- **Concatenation** of strings x and y, written xy ($x \circ y$), is the symbols from x followed by the symbols from y

Ex. $x = \text{ab}, y = \text{ba} \quad \Rightarrow \quad xy =$ abba

$x = \text{ab}, y = \varepsilon \quad \Rightarrow \quad xy =$ ab

- **Reversal** of string x, written x^R, consists of the symbols of x written backwards

Ex. $x = \text{aab} \quad \Rightarrow \quad x^R =$ baa

Fun with String Operations

What is $(xy)^R$?

Ex. $x = \text{aba}, y = \text{bba}$ $\Rightarrow$ $xy =$ ababba

$\Rightarrow$ $(xy)^R =$ $\underbrace{\text{abb}}_{y^R}\underbrace{\text{aba}}_{x^R}$

a) $x^R y^R$

b) $y^R x^R$ ←

c) $(yx)^R$

d) xy^R

Fun Proofs with String Operations

Claim: $(xy)^R =$ $y^R x^R$

Proof: Let $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_m$ be arbitrary strings

Then $(xy)^R =$ $(x_1 x_2 \ldots x_n y_1 y_2 \ldots y_m)^R$

$= y_m \ldots y_1 \; x_n \ldots x_1$

$= y^R x^R$ ∎

Not even the most formal way to do this:

1. Define string length recursively
2. Prove by induction on $|y|$

More formal than needed for us

Languages

A language L is a set of strings over an alphabet Σ

i.e., $L \subseteq \Sigma^*$ = set of all strings over Σ

Languages = computational (decision) problems

Input: String $x \in \Sigma^*$

Output: Is $x \in L$? (YES or NO?)

Some Simple Languages

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

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

Ø (Empty set)

$\phi = \{ \}$

$\phi = \{ \}$

Σ^* (All strings)

$\{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

$\{\epsilon, a, b, c, aa, ab, ac, \ldots aaa, aab, \ldots\}$

$\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$
(All strings of length n)

$n = 2$

$\{00, 01, 10, 11\}$

$n = 3$

$\{000, 001, \ldots, 111\}$

$n = 2$

$\{aa, ab, ac, \ldots, cc\}$

$n = 3$

$\{aaa, aab, \ldots, ccc\}$

Some More Interesting Languages

- $L_1 =$ The set of strings $x \in \{a, b\}^*$ that have an equal number of a's and b's $abba \in L_1$ $a \notin L_1$

  $L_1 = \{ x \in \{a,b\}^* \mid x$ has the same number of a's as b's $\}$

- $L_2 =$ The set of strings $x \in \{a, b\}^*$ that start with (0 or more) a's and are followed by an equal number of b's

  $\varepsilon \in L_2$ $aabb \in L_2$ $baa \notin L_2$ $abab \notin L_2$

  $L_2 = \{ x \in \{a,b\}^* \mid$ green stuff $\} = \{ a^n b^n \mid n \geq 0 \}$

- $L_3 =$ The set of strings $x \in \{0, 1\}^*$ that contain the substring '0100' $10100 \in L_3$ $001100 \notin L_3$

  $1010001 \in L_3$

  $L_3 = \{ x0100y \mid x, y \in \{0,1\}^* \}$

  DANGER: $\left(\neq \{ x0100x \mid x \in \{0,1\}^* \} \right)$

Some More Interesting Languages

- $L_4 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ of length at most 4

  $L_4 = \{ x \in \{a,b\}^* \mid |x| \leq 4 \}$

- $L_5 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ that contain at least two a's

  $aba \in L_5 \qquad ba \notin L_5$

  $L_5 = \{ x a y a z \mid x, y, z \in \{a,b\}^* \}$

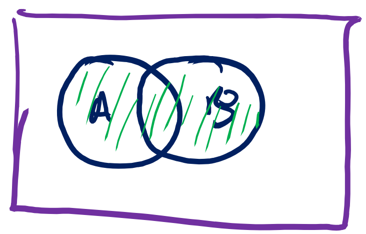
  $L_6 = \{ x_1 a x_2 a \ldots a x_{201} \mid x_1, \ldots, x_{201} \in \{a,b\}^* \}$

New Languages from Old

$L_6 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ that have an equal number of a's and b's and length greater than 4

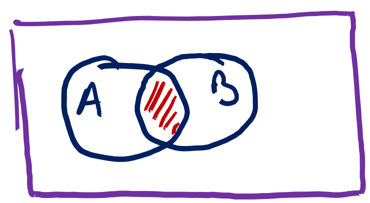
Since languages are just sets of strings, can build them using set operations:

$A \cup B$ "union"



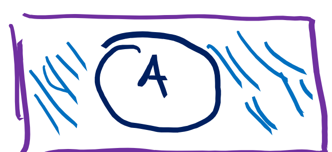
$A \cup B = \{x \mid x \in A \ \underline{\text{OR}} \ x \in B\}$

$A \cap B$ "intersection"



$A \cap B = \{x \mid x \in A \ \underline{\text{AND}} \ x \in B\}$.

$\bar{A}$ "complement"



$\bar{A} = \{x \mid x \notin A\}$

New Languages from Old

$L_6 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ that have an equal number of a’s and b’s and have length greater than 4

- $L_1 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ that have an equal number of a’s and b’s
- $L_4 =$ The set of strings $x \in \{\mathrm{a}, \mathrm{b}\}^*$ of length at most 4

$$L_6 = \{ x \mid x \in L_1 \text{ AND } x \notin L_4 \}$$
$$= L_1 \cap \{ x \mid \notin L_4 \}$$

$$\Rightarrow L_6 = L_1 \cap \overline{L_4}$$

Operations Specific to Languages

- **Reverse**: $L^R = \{x^R | x \in L\}$

  Ex. $L = \{\varepsilon, \mathrm{a}, \mathrm{ab}, \mathrm{aab}\}$ $\Rightarrow L^R = \{\epsilon, a, ba, baa\}$

  $= \{aab, a, ab, \epsilon\}$

- **Concatenation:** $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

  Ex. $L_1 = \{\mathrm{ab}, \mathrm{aab}\}$ $L_2 = \{\varepsilon, \mathrm{b}, \mathrm{bb}\}$

  $\Rightarrow L_1 \circ L_2 = \{ab, abb, abbb,$

  $aab, aabb, aabbb\}$

A Few "Traps"

String, language, or something else?

ε string (empty string, length 0)

$\emptyset$ set, language (set of strings)

$\{\varepsilon\}$ set, language (set containing empty string)

$\{\emptyset\}$ set of sets (could be a $\underbrace{\text{set of languages}}_{\text{"class"}}$)