BU CS 332 – Theory of Computation

Link to polls:
https://forms.gle/eeRKEZf5phJ3GBZr5

Lecture 2:

- Parts of a Theory of Computation
- Sets, Strings, and Languages

Reading:

Sipser Ch 0

Reminders:

- HW1 due tomorrow night (Tue, 11:59PM)

Mark's (temporary) OH: today, 4:30-6 CDS 10th floor, yellow lounge outside 1021

Mark Bun

January 27, 2025

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 determining whether a *string* is in a *language*

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

$\Sigma = \{a, b\}$ $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$

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

What is a (Computational) Problem?

For us: A problem will be the task of determining 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. $\{\mathrm{a}, \mathrm{b}\}^* = \{\varepsilon, \mathrm{a}, \mathrm{b}, \mathrm{aa}, \mathrm{ab}, \mathrm{ba}, \mathrm{bb}, \ldots\}$
- **Language:** A set $L \subseteq \Sigma^*$ of strings

Examples of Languages

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

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

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

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

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

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

Primality language

Which best represents the language corresponding to the Primality problem? (I.e., strings over the alphabet $\{0, 1\}$ that are binary representations of prime numbers.)

Let's say the most significant bit is on the left, so "100" is the binary representation of 4. $\{x \in \{0,1\}^* \mid x$ is the binary rep. of a prime$\}$

(a) $\{2, 3, 5, 7, \dots\}$

(b) $\{10, 11, 101, 111, \dots\}$ · binary representations of 2, 3, 5, 7, ...

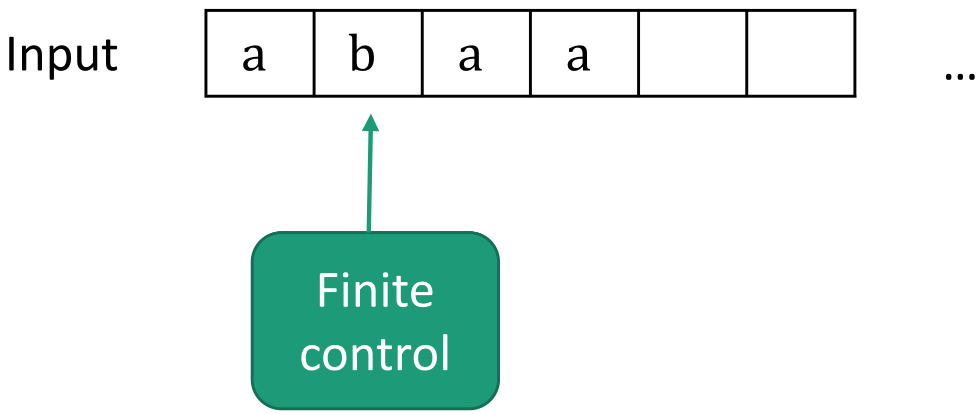
3 7 31

(c) $\{11, 111, 11111, 1111111, \dots\}$

(d) $\{11, 011, 101, 110, 111, 0111, \dots\}$

Machine Models

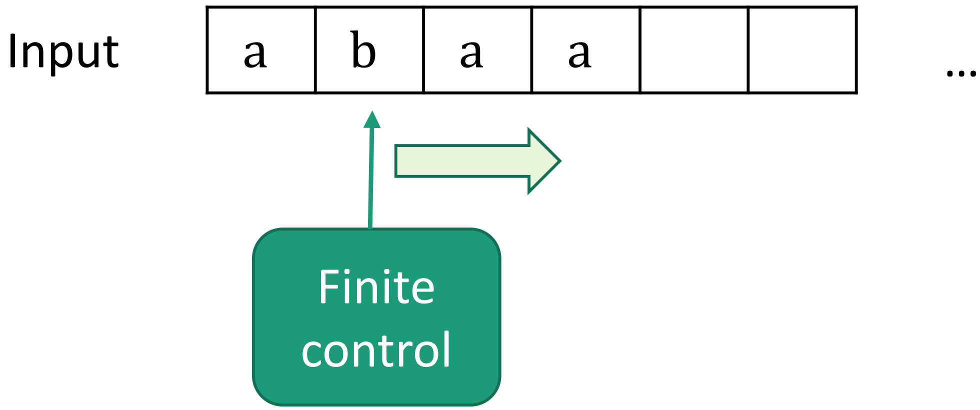
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



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

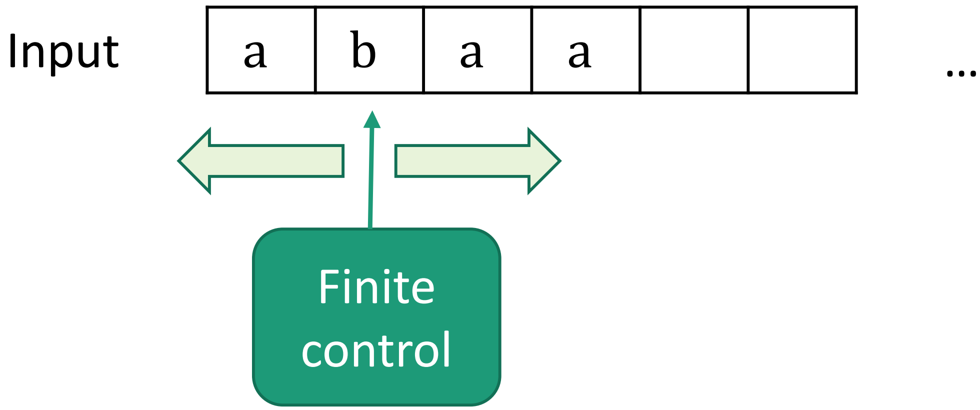


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

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 solve the associated computational problems

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" or "no harder" than another?

Why study theory of computation?

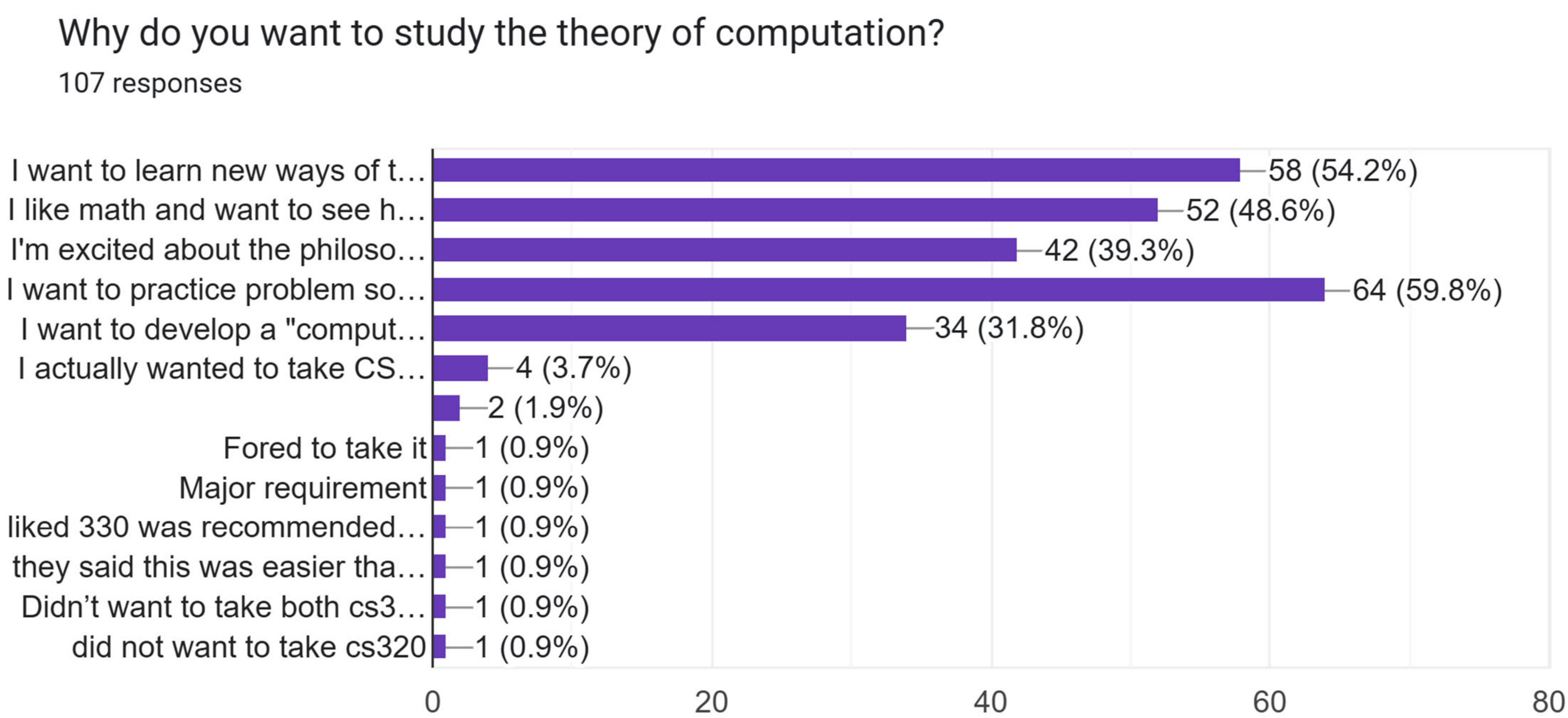
- You'll learn how to formally reason about computation
- You'll learn the technology-independent foundations of CS

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."

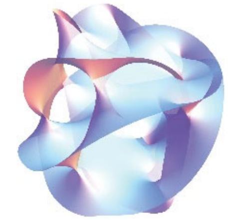

"Boss, I can't find an efficient algorithm
because no such algorithm exists."

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

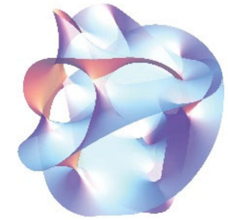
  Ex. $\mathrm{bba, ababb}$

  ε denotes empty string, length 0

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

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

String Theory

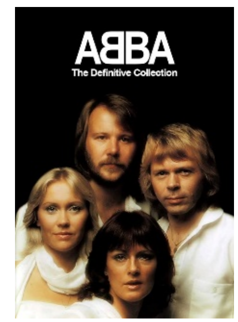


- **Length** of a string, written $|x|$, is the number of symbols

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

$x \circ y$ [might see other symbols to denote concat.]

- **Concatenation** of strings x and y, written xy, is the symbols from x followed by the symbols from y



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

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

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

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

Fun with String Operations

What is $(xy)^R$?

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

$\Rightarrow$ $(xy)^R =$ abbaba

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 $x = x_1 x_2 \dots x_n$ and $y = y_1 y_2 \dots y_m$

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

$= y_m \dots y_1 \; x_n \dots x_1 = y^R x^R$

Not even the most formal way to do this:

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

Generally not needed for us, but you may get a bonus problem

Languages

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

i.e., $L \subseteq \Sigma^*$

Languages = computational (decision) problems

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

Output: Is $x \in L$? (Yes or No?)

Some Simple Languages

	$\Sigma = \{0, 1\}$	$\Sigma = \{a, b, c\}$
$\emptyset$ (Empty set)	$\{\ \}$	$\{\ \}$
Σ^* (All strings)	$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$	$\{a,b,c\}^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, \ldots\}$
$\Sigma^n = \{x \in \Sigma^* \mid \ \lvert x \rvert = n\}$ (All strings of length n)	$n=2$: $\{0,1\}^2 = \{00, 01, 10, 11\}$ $n=3$: $\{0,1\}^3 = \{000, 001, \ldots\}$	$\{a,b,c\}^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$ $\{a,b,c\}^3 = \{aaa, aab, \ldots\}$

Some More Interesting Languages

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

$\{x \in \{a,b\}^* \mid \#a\text{'s in } x = \#b\text{'s in } x\}$

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

$\{x \in \{a,b\}^* \mid \text{green text}\} = \{yz \mid y \in \{a\}^*, z \in \{b\}^*, |y| = |z|\}$

$= \{a^n b^n \mid n \geq 0\}$ where $a^n = \underbrace{aa \ldots a}_{n}$

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

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

DANGER: NOT $\{x0100x \mid x \in \{0,1\}^*\}$ THE SAME!

Some More Interesting Languages

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

$\{x \in \{a,b\}^* \mid |x| \le 4\} = \Sigma^4 \cup \Sigma^3 \cup \Sigma^2 \cup \Sigma \cup \{\varepsilon\}$

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

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

aabbbbbb
realized via
$x = \varepsilon$, $y = \varepsilon$, $z =$ bbbbbb

$= \{z \in \{a,b\}^* \mid z \text{ contains at least two a's}\}$

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

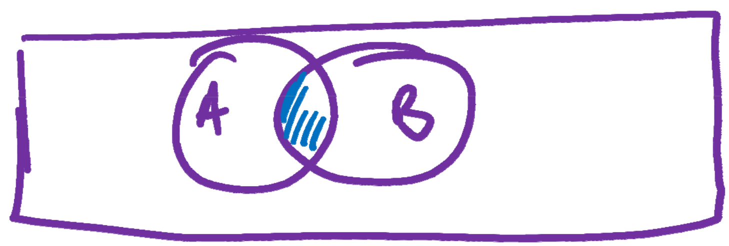
Since languages are just sets of strings, can build them using set operations:

$A \cup B$ "union"

$= \{x \mid x \in A$ or $x \in B\}$

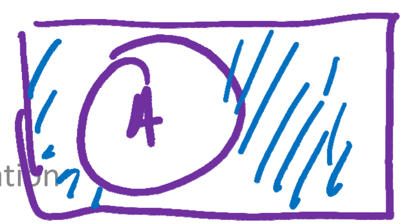
$A \cap B$ "intersection"

$= \{x \mid x \in A$ and $x \in B\}$



$\bar{A}$ "complement"

$= \{x \in \Sigma^* \mid x \notin A\}$



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

$L_6 = \{x \in \{a,b\}^* \mid x \text{ has equal \# of a's \& b's}\} \cap \{x \in \{a,b\}^* \mid |x| > 4\}$

- $L_1 =$ 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

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

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

$\overline{L_4}$ should be interpreted as $\{x \in \{a,b\}^* \mid x \notin L_4\}$

Operations Specific to Languages

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

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

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

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

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

  $\Rightarrow L_1 \circ L_2 = \{ab, abb, abbb, aab, aabb, aabbb\}$

```
init S = ∅
for each x∈L1:
    for each y∈L2:
        add x∘y to S
return S
```

A Few "Traps"

String, language, or something else?

ε string (empty string)

$\emptyset$ language (empty language) containing no strings

also a set

$\{\varepsilon\}$ language, also a set

$\{\emptyset\}$ set containing empty (not a language)

Languages

Languages = computational (decision) problems

<u>Input:</u> String $x \in \Sigma^*$

<u>Output:</u> Is $x \in L$? (Yes or No? I.e., Accept or Reject?)

The language **recognized** by a program is the set of strings $x \in \Sigma^*$ that it *accepts*

What Language Does This Program Recognize?

Alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}\}$

On input $x = x_1 x_2 \dots x_n$:

count = 0

  For $i = 1, \dots, n$:

    If $x_i = \mathrm{a}$:

      count $=$ count $+ 1$

  If count ≤ 4: accept

  Else: reject

a) $\{x \in \Sigma^* \mid |x| > 4\}$

b) $\{x \in \Sigma^* \mid |x| \leq 4\}$

c) $\{x \in \Sigma^* \mid |x| = 4\}$

d) $\{x \in \Sigma^* \mid x \text{ has more than 4 a's}\}$

e) $\{x \in \Sigma^* \mid x \text{ has at most 4 a's}\}$

f) $\{x \in \Sigma^* \mid x \text{ has exactly 4 a's}\}$