

BU CS 332 – Theory of Computation

Link to polls:

<https://forms.gle/CZseiprgfDLkFqYJA>



Lecture 2:

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

Reading:

Sipser Ch 0

Reminders:

- HW#0 due tonight
(Thursday, 11:59PM)

Mark Bun & Alexander Poremba

January 22, 2026

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***

What is a (Computational) Problem?

For us: A problem will be the task of **determining 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?

$$\Sigma = \{a, b\} \quad \text{"alphabet"}$$

$$\mathcal{L}_{\text{PARITY}} = \{x \in \Sigma^* : x \text{ contains an even number of a's}\}$$

$$\Sigma^* = \{\epsilon, a, b, ab, ba, \dots\}$$

$$\text{Important: } \mathcal{L}_{\text{PARITY}} \subseteq \Sigma^*$$

What is a (Computational) Problem?

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

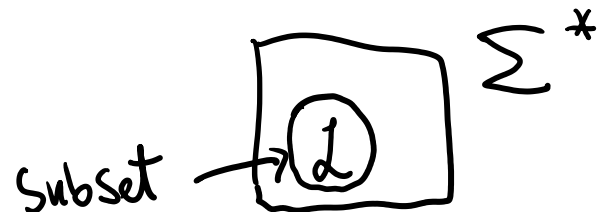
- **Alphabet:** A finite set Σ Ex. $\Sigma = \{a, b\}$
- **String:** A finite concatenation of alphabet symbols
Ex. bba, ababb

ε denotes empty string, length 0

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

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

- **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?

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

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

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

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

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

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.

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

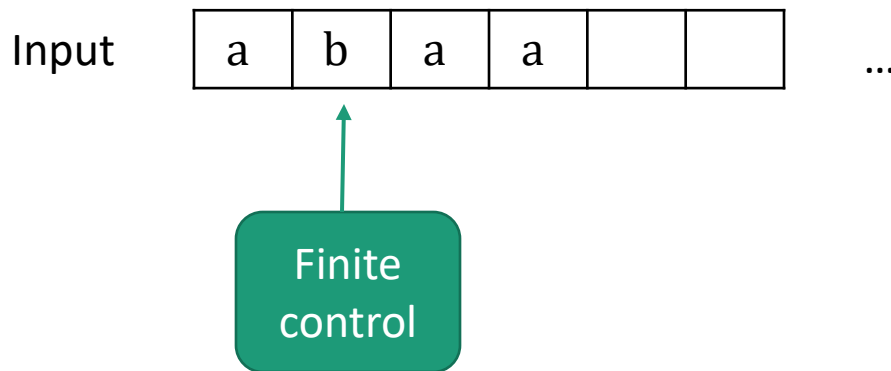
(b) $\{10, 11, 101, 111, \dots\}$

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

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

Machine Models

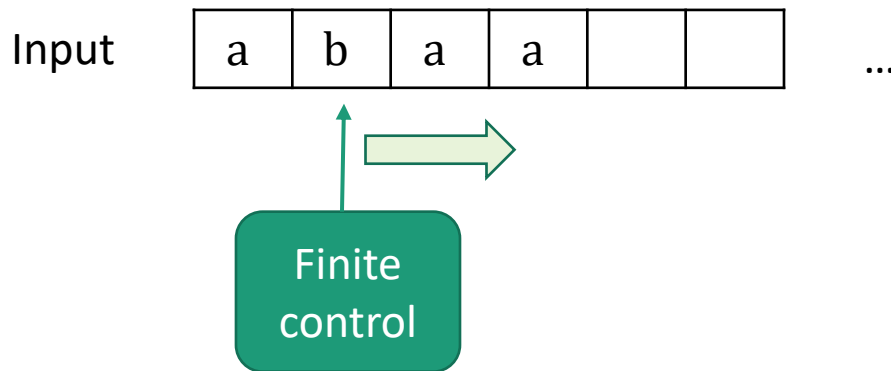
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



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

- Finite Automata (FAs): Machine with a finite amount of unstructured memory

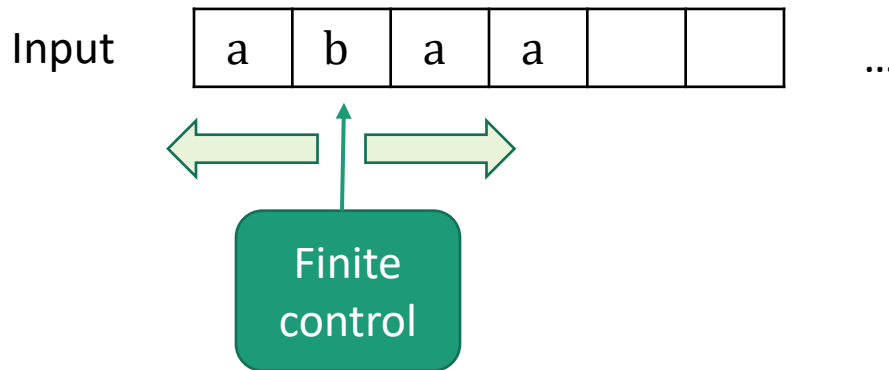


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

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

Machine Models

- Turing Machines (TMs): Machine with unbounded, unstructured memory



Control can scan in both directions
Control can both read and write

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

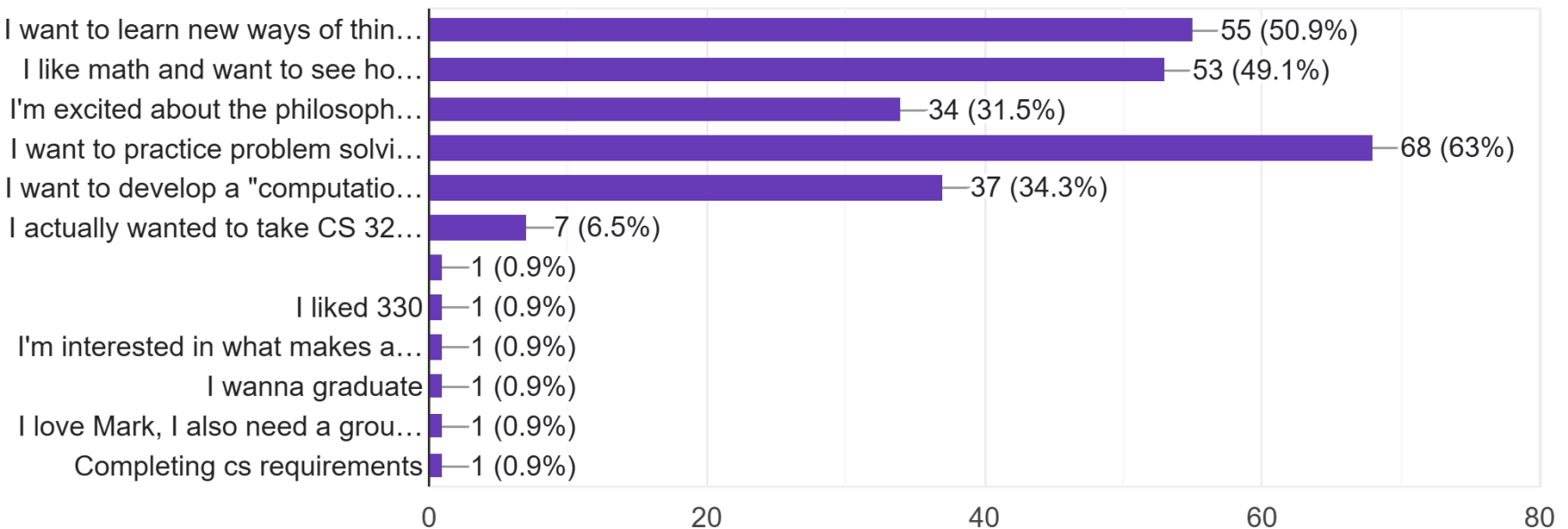
Connections to other parts of science:

- 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 do you want to study the theory of computation?

108 responses



Why study theory of computation?

Practical knowledge for developers



“Boss, I can’t find an efficient algorithm.
I guess I’m just too dumb.”



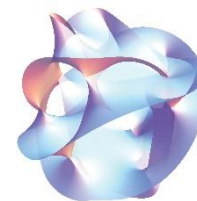
“Boss, I can’t find an efficient algorithm
because no such algorithm exists.”

Will you be asked about this material on job interviews?

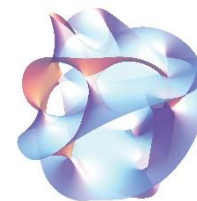
No promises, but a true story...

More about strings and languages

String Theory

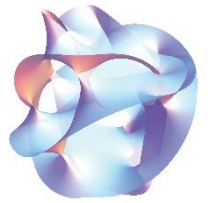


String Theory



- **Symbol:** Ex. a, b, 0, 1
- **Alphabet:** A finite set Σ of symbols Ex. $\Sigma = \{a, b\}$
- **String:** A finite concatenation of alphabet symbols
Ex. bba, ababb
 ε denotes empty string, length 0
 Σ^* = set of all strings using symbols from Σ
Ex. $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$
- **Language:** A set $L \subseteq \Sigma^*$ of strings

String Theory



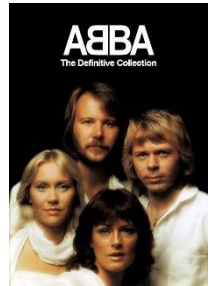
- **Length** of a string, written $|x|$, is the number of symbols

Ex. $|abba| = 4$ $|\varepsilon| = 0$

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

Ex. $x = ab, y = ba$ \Rightarrow $xy = abba$

$x = ab, y = \varepsilon$ \Rightarrow $xy = ab$



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

Ex. $x = aab$ \Rightarrow $x^R = baa$

Fun with String Operations



What is $(xy)^R$?

Ex. $x = aba, y = bba \Rightarrow xy = aba\ bba$
 $\Rightarrow (xy)^R = (aba\ bba)^R$
 $= abbaba$

- a) $x^R y^R$
- b) $y^R x^R$
- c) $(yx)^R$
- d) xy^R

Fun ^{Proofs} with String Operations

Claim: $(xy)^R = y^R x^R$, $\forall x, y \in \Sigma^*$

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

$$\text{Then } (xy)^R = \left(\underline{x_1 x_2 \dots x_n y_1 y_2 \dots y_m} \right)^R$$

$$= y_m y_{m-1} \dots y_1 x_n x_{n-1} \dots x_1$$

$$= \left(\underline{y_m y_{m-1} \dots y_1} \right) \left(\underline{x_n x_{n-1} \dots x_1} \right) = y^R x^R$$

concatenation

Reverse rule
←

Not even the most formal way to do this:

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

generally
not required
in this class!

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 $\mathcal{L} \subseteq \Sigma^*$

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

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

\emptyset (Empty set)

$\{ \}$

$\{ \}$

Σ^* (All strings)

$\{0, 1\}^*$

$\{a, b, c\}^*$

$\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$ $n=2$:

$n=2$:

(All strings of length n) $\{0, 1\}^2 = \{00, 01, 10, 11\}$

$\{a, b, c\}^2$

$= \{ab, bc, ac, aa, bb, cc, ba, cb, ca\}$

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

$$L_2 = \{ yz \mid y \in \{a\}^*, z \in \{b\}^* \text{ s.t. } |z| = |y| \}$$

$$L_3 = \{ y 0100 z \mid y \in \{0, 1\}^*, z \in \{0, 1\}^* \}$$

Some More Interesting Languages

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

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

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

$$L_5 = \{\alpha \underline{a} \beta \underline{a} \gamma \mid \alpha, \beta, \gamma \in \{a, b\}^*\}$$

-

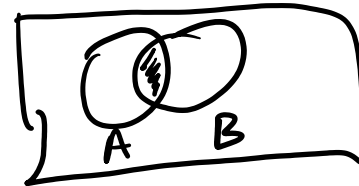
New Languages from Old

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

$$L_1 \cap \bar{L}_4$$

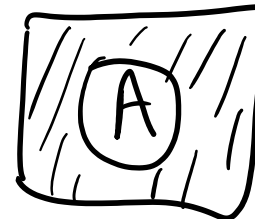
Since languages are just sets of strings, can build them using set operations:

$A \cup B$ “union”



$A \cap B$ “intersection”

\bar{A} “complement”



New Languages from Old

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

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

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

Operations Specific to Languages

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

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

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

Ex. $L_1 = \{ab, aab\} \quad L_2 = \{\epsilon, b, bb\}$

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

A Few “Traps”



String, language, or something else?

ϵ (empty) string

\emptyset (empty) language

$\{\epsilon\}$ language

$\{\emptyset\} \stackrel{?}{\subseteq} \sum^*$
 \uparrow Set of Sets \nwarrow set of strings

something else

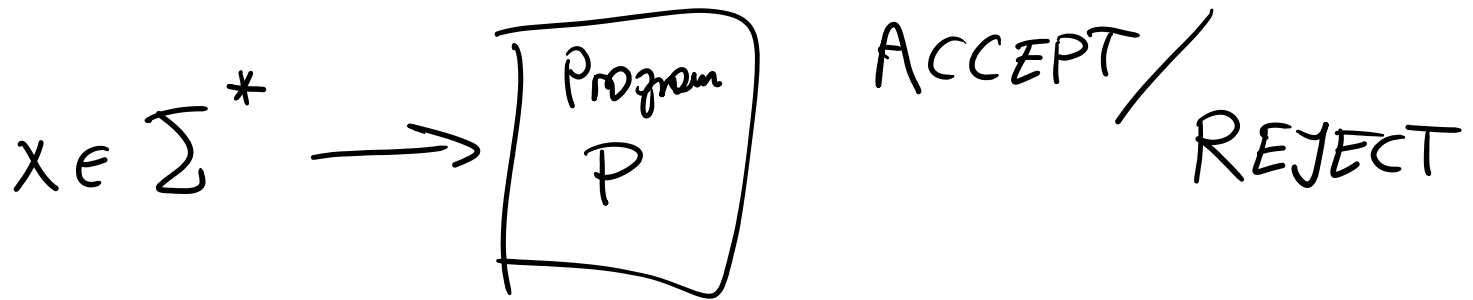
Languages

Languages = computational (decision) problems

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

Output: 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*



$$L = \{ x \in \Sigma^* \mid P \text{ accepts } x \} \subseteq \Sigma^*$$

What Language Does This Program Recognize?

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

On input $x = x_1x_2 \dots x_n$:

count = 0

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

If $x_i = a$:

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

