BU CS 332 – Theory of Computation

Link to polls: https://forms.gle/2hWWjEjNge1P1V9J9

Lecture 2:

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

Reading:

Sipser Ch 0

HW 0 due tonight
(11:59 pm)

Mark Bun

September 7, 2021

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*

- **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

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

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

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

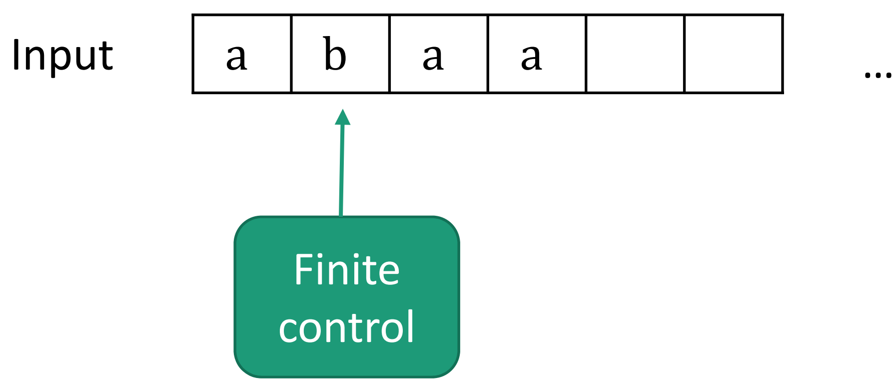
$\Sigma = \{0, 1\}$ $\quad L = \{x \mid x \text{ represents a prime}\}$

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

$\Sigma =$ ASCII $\quad L = \{P \mid P \text{ gets stuck in a loop}\}$

Machine Models

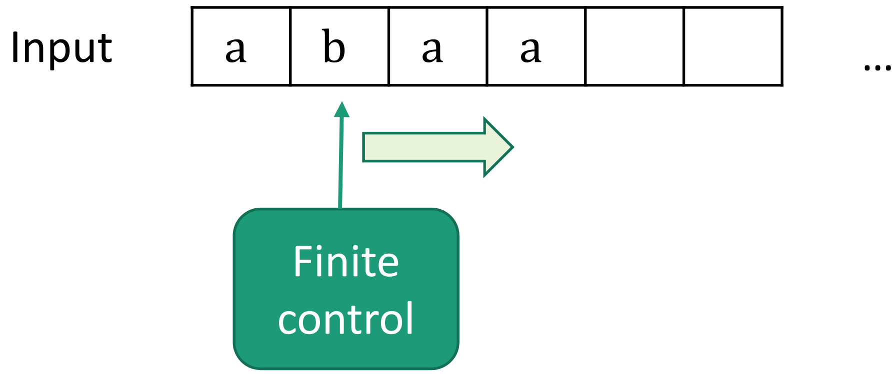
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



<u>Abstraction:</u> 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

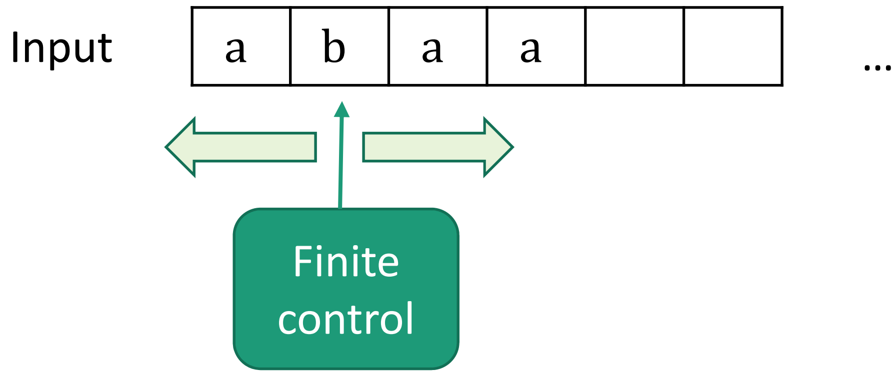


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

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

Machine Models

- <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 $\{L \mid L \text{ recognizable by an FA}\} \subseteq \{L \mid L \text{ recognizable by a TM}\}$

Non-inclusion: There exist languages recognizable by TMs which are not recognizable by FAs $\{L \text{ recog. by TMs}\} \not\subseteq \{L \text{ recogn. by FAs}\}$

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.

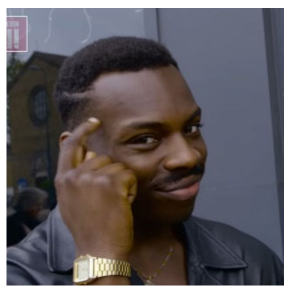
What appeals to you about the theory of computation?

1. I want to learn new ways of thinking about computation
2. I like math and want to see how it's used in computer science
3. I'm excited about the philosophical questions about computation
4. I want to practice problem solving and algorithmic thinking
5. I want to develop a "computational perspective" on other areas of math/science
6. I actually wanted to take CS 320 or 350 but they were full

Why study theory of computation?

Practical knowledge for developers

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



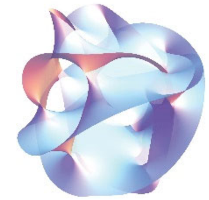
"Boss, I can't find an efficient algorithm
because no such algorithm exists."

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

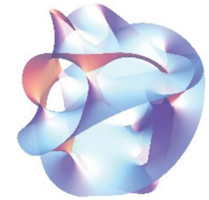
  Ex. $\mathrm{bba, ababb}$

  ε denotes empty string, length 0

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

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

String Theory



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

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

$x \circ y$

- **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

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

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

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

$= y_m y_{m-1} \dots y_2 y_1 x_n x_{n-1} \dots x_2 x_1$

$= y^R x^R$

Not even the most formal way to do this:

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

More detailed than necessary for us

Languages

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

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

Languages = computational (decision) problems

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

<u>Output:</u> Is $x \in L$? (YES or NO?)

Accept or Reject?

Some Simple Languages

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

Some More Interesting Languages

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

$aabb \in L_1$, $abab \in L_1$, $abb \notin L_1$

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

$aabb \in L_2$, $abab \in L_2$

$= \{ a^n b^n \mid n \geq 0 \}$

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

$1\underline{0100}01 \in L_3$ $\quad$ $01100 \notin L_3$

$= \{ abc \mid a, c \in \{0,1\}^*, b = 0100 \} = \{ a0100c \mid a, c \in \{0,1\}^* \}$

Some More Interesting Languages

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

$abba \in L_4$, $ab \in L_4$, $\varepsilon \in L_4$, $abbab \notin L_4$

$L_4 = \{ x \mid |x| \leq 4,\ x \in \{a,b\}^* \} = \{ vwxyz \mid v,w,x,y \in \{\varepsilon, a,b\},\ z \in \{a,b\}^* \}$

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

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

$abba \in L_5$, $abbaa \in L_5$, $abbb \notin L_5$

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

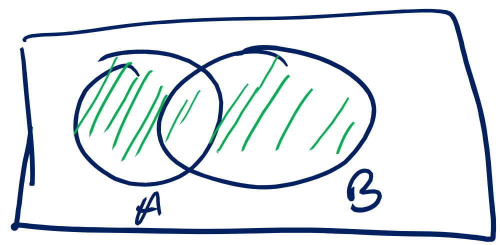
$L_{50} = \{ x_0\, a\, x_1\, a\, x_2\, a \ldots a\, x_{50} \mid x_0, \ldots, x_{50} \in \{a,b\}^* \}$

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

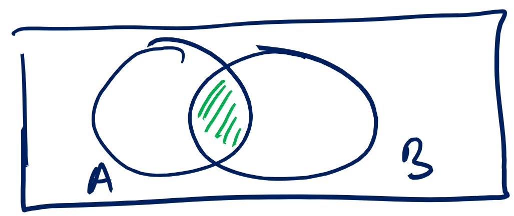
Since languages are just sets of strings, can build them using set operations:

$A \cup B$ "union"



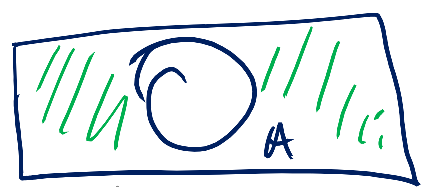
$\{w \mid w \in A \text{ or } w \in B\}$

$A \cap B$ "intersection"



$\{w \mid w \in A \text{ and } w \in B\}$

$\bar{A}$ "complement"



$\{w \mid w \notin A\}$

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

$\overline{L_4} = \{$ strings of length $> 4 \}$

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

Operations Specific to Languages

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

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

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

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

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

initialize $S = \phi$
for each $x \in L_1$:
  for each $y \in L_2$:
    add xy to S
return S // $= L_1 \circ L_2$

$|L_1 \circ L_2| \leq |L_1||L_2|$

A Few "Traps"

String, language, or something else?

ε — The empty string

$\emptyset$ — The empty language (a set)

$\{\varepsilon\}$ — The language consisting (only) of the empty string

$\{\emptyset\}$ — Something else: The set containing the empty set / language

"class" of languages

Languages

Languages = computational (decision) problems

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

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

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

$L = \{ \text{strings } x \in \{a,b\}^* \mid \text{program accepts } x \}$