BU CS 332 – Theory of Computation

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:* $\Sigma = \{a, b\}$

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

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

$\Leftrightarrow$ is $x$ a member of the language **PARITY**?

$\Sigma^* = \text{all strings over } \{a, b\}$
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 $\Sigma$  
  Ex. $\Sigma = \{a, b\}$
- **String**: A finite concatenation of alphabet symbols  
  Ex. bba, ababb

  $\varepsilon$ denotes empty string, length 0  
  $\Sigma^* = \text{set of all strings using symbols from } \Sigma$

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

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

*E \in L* (empty string has even # of a’s)

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

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

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

\[ \Sigma = \text{ASCII} \quad L = \{p \in \Sigma^* \mid p \text{ describes a C program that loops forever on some input} \} \]
Machine Models

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

Input: \[ \begin{array}{cccc} a & b & a & a \\ \end{array} \ldots \]

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

Input: 

| a | b | a | a | ... |

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

Input: \[ \text{a} \quad \text{b} \quad \text{a} \quad \text{a} \quad \ldots \] 

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 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
  [link](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?

- I want to learn new ways of thinking: 39 (62.9%)
- I like math and want to see how it's used: 36 (58.1%)
- I'm excited about the philosophy behind it: 17 (27.4%)
- I want to practice problem solving: 34 (54.8%)
- I want to develop a "computational" mindset: 25 (40.3%)
- I actually wanted to take CS 32: 4 (6.5%)
- I am curious to learn the current state of the field: 1 (1.6%)
- I don't know what it is so wanna check it out: 1 (1.6%)
- Course seemed less awful than expected: 1 (1.6%)
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. a, b, 0, 1
- **Alphabet:** A finite set $\Sigma$ Ex. $\Sigma = \{a, b\}$
- **String:** A finite concatenation of alphabet symbols
  Ex. bba, ababb
  $\varepsilon$ denotes empty string, length 0
  $\Sigma^* = \text{set of all strings using symbols from } \Sigma$
  Ex. $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, \ldots\}$
- **Language:** A set $L \subseteq \Sigma^*$ of strings
String Theory

• **Length** of a string, written $|x|$, is the number of symbols
  Ex. $|abba| = 4$  \quad $|\varepsilon| = 0$

• **Concatenation** of strings $x$ and $y$, written $xy$, is the symbols from $x$ followed by the symbols from $y$
  Ex. $x = ab, y = ba$  \quad \Rightarrow  \quad xy = abba$
  \quad $x = ab, y = \varepsilon$  \quad \Rightarrow  \quad xy = ab$

• **Reversal** of string $x$, written $x^R$, consists of the symbols of $x$ written backwards
  Ex. $x = aab$  \quad \Rightarrow  \quad x^R = baab$
Fun with String Operations

What is $(xy)^R$?

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

a)  $x^R y^R$

b)  $y^R x^R$  $\leftarrow$

c)  $(yx)^R$

d)  $xy^R$
Fun with String Operations

Claim: \((xy)^R = y^Rx^R\)

Proof: Let \(x = x_1x_2 \ldots x_n\) and \(y = y_1y_2 \ldots y_m\). Then \((xy)^R = (x_1x_2 \ldots x_n y_1y_2 \ldots y_m)^R = y_m \ldots y_1 x_n \ldots x_1 = y^Rx^R\).

Not even the most formal way to do this:
1. Define string length recursively
2. Prove by induction on \(|y|\)
Languages

A language $L$ is a set of strings over an alphabet $\Sigma$

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

Languages = computational (decision) problems

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

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

\[ \Sigma = \{0, 1\} \]

\[ \phi = \varepsilon \]

\[ \Sigma = \{a, b, c\} \]

\[ \phi = \varepsilon \]

\( \emptyset \) (Empty set)

\[ \Sigma^* \) (All strings)

\[ \Sigma^n = \{ x \in \Sigma^* \mid |x| = n \} \)

(All strings of length \( n \))
Some More Interesting Languages

• $L_1 = \{x \in \{a, b\}^* \mid \text{have an equal number of } a\text{’s and } b\text{’s} \}$
  
  \[ aabb \in L_1, \quad a \notin L_1 \]

  
  
  \[ L_1 = \{ x \in \{a, b\}^* \mid x \text{ has the same number of } a\text{’s as } b\text{’s} \} \]

• $L_2 = \{x \in \{a, b\}^* \mid \text{start with } (0 \text{ or more}) \text{ a’s and are followed by an equal number of } b\text{’s} \}$

  
  \[ e \in L_2, \quad aabb \in L_2, \quad baa \notin L_2, \quad abab \notin L_2 \]

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

• $L_3 = \{x \in \{0, 1\}^* \mid \text{contain the substring } ‘0100’ \}$

  
  \[ 10100 \in L_3, \quad 001100 \notin L_3 \]

  
  
  \[ L_3 = \{ x \in \{0, 1\}^* \mid x, y \in \{0, 1\}^* \quad \text{DANGER: } (\notin \{ x \in \{0, 1\}^* \mid x \in \{0, 1\}^* \} \} \]
Some More Interesting Languages

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

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

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

\[
L_5 = \{ x \in \{a, b\}^* \mid \text{contains at least two } a\text{'s} \}
\]

\[
\begin{align*}
abaa & \in L_5 \\
ba & \notin L_5
\end{align*}
\]

\[
L_5 = \{ x ayaz \mid x, y, z \in \{a, b\}^* \}
\]

\[
L_6 = \{ x_1 a x_2 a \cdots a x_{20} \mid x_1, \ldots, x_{20} \in \{a, b\}^* \}
\]
New Languages from Old

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

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

- $A \cup B$ \text{ “union”}
- $A \cap B$ \text{ “intersection”}
- $\overline{A}$ \text{ “complement”}

\[ A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \]

\[ A \cap B = \{ x \mid x \in A \text{ and } x \in B \} \]

\[ \overline{A} = \{ x \mid x \notin A \} \]
New Languages from Old

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

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

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

$\Rightarrow L_6 = L_1 \cap \overline{L_4}$
Operations Specific to Languages

- **Reverse:** $L^R = \{x^R \mid x \in L\}$
  
  Ex. $L = \{\varepsilon, a, ab, aab\}$  \implies $L^R = \{\varepsilon, a, ba, baa\}$
  
  $= \{a, aa, aab, aab\}$

- **Concatenation:** $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$
  
  Ex. $L_1 = \{ab, aab\}$ \quad $L_2 = \{\varepsilon, b, bb\}$
  
  $\implies L_1 \circ L_2 = \{\varepsilon ab, ab\varepsilon, abbb, aab, aab\varepsilon, aab\varepsilon\}$
A Few “Traps”

String, language, or something else?

\( \varepsilon \)  
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 set of languages)  
“class”