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

Mark Bun

September 7, 2021

HW 0 due tonight (11:59 pm)
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 $\Sigma$  
  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?
\[ \Sigma = \{a, b\} \quad L = \{w \mid w \text{ has an even number of } a's\} \]

**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?
\[ \Sigma = \text{ASCII} \quad L = \{P \mid \text{P gets stuck in a loop}\} \]
Machine Models

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

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: \[a \ b \ a \ a \ \ldots\]

  - 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: a b a a ...

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.

\[ \exists L \in \text{recognizable by an FA} \subseteq \exists L \in \text{recognizable by a TM} \]

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

\[ \exists L \in \text{recognizable by TMs} \not\subseteq \exists L \in \text{recognizable by FAs} \]

**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](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. 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. $|abba| = 4$  \hspace{1cm}  $|\varepsilon| = 0$

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

  Ex. $x = ab, y = ba$  \hspace{1cm}  $\Rightarrow$  \hspace{1cm}  $xy = abba$

  $x = ab, y = \varepsilon$  \hspace{1cm}  $\Rightarrow$  \hspace{1cm}  $xy = ab$

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

  Ex. $x = aab$  \hspace{1cm}  $\Rightarrow$  \hspace{1cm}  $x^R = baa$
Fun with String Operations

What is $(xy)^R$?

Ex. $x = aba$, $y = bba$ \implies $xy = ababba$
\implies $(xy)^R = \overline{y^R x^R}$

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 \(y = y_1 y_2 \ldots y_m\).

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

= \(y_m y_{m-1} \ldots y_2 y_1, x_n x_{n-1} \ldots x_2 x_1\)

= \(y^R x^R\)

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

A language $L$ is a set of strings over an alphabet $\Sigma$ i.e., $L \subseteq \Sigma^*$

Languages = computational (decision) problems

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

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

Accept or Reject?
Some Simple Languages

- **∅ (Empty set)**
  \[\emptyset = \emptyset \]

- **\(\Sigma^*\) (All strings)**
  \[\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}\]

- **\(\Sigma^n\) (All strings of length \(n\))**
  \[\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}\]
  - \(n = 2\):
    \[\Sigma_2^2 = \{\varepsilon, 00, 01, 10, 11\}\]
  - \(n = 3\):
    \[\Sigma_3^3 = \{\varepsilon, 000, 001, 010, 011, 100, 101, 110, 111\}\]

- **\(\Sigma = \{0, 1\}\)**
  \[\Sigma = \{0, 1\}\]

- **\(\Sigma = \{a, b, c\}\)**
  \[\Sigma = \{a, b, c\}\]
Some More Interesting Languages

• $L_1 = \text{The set of strings } x \in \{a, b\}^* \text{ that have an equal number of } a\text{'s and } b\text{'s}$
  $$aa\overline{bb} \in L_1, \quad ab\overline{ab} \in L_1, \quad \overline{abb} \notin L_1.$$  

• $L_2 = \text{The set of strings } x \in \{a, b\}^* \text{ that start with } (0 \text{ or more}) a\text{'s and are followed by an equal number of } b\text{'s}$
  $aa\overline{bb} \in L_2, \quad ab\overline{ab} \in L_2$
  $$= \{a^n b^n \mid n \geq 0\}.$$  

• $L_3 = \text{The set of strings } x \in \{0, 1\}^* \text{ that contain the substring } \text{‘0100’}$
  $1\overline{0100}01 \in L_3, \quad 01\overline{1000} \notin L_3$
  $$= \{a b c \mid a, c \in \{0, 1\}^*, b = 0100 \}.$$  

9/7/2021
Some More Interesting Languages

- $L_4$ = The set of strings $x \in \{a, b\}^*$ of length at most 4
  
  \[
  L_4 = \{ x \mid |x| \leq 4 \}, \quad x \in \{a, b\}^3 = \{ \forall wxy \mid \forall \in \{a, b\}^2 \}
  \]

- $L_5$ = The set of strings $x \in \{a, b\}^*$ that contain at least two $a$'s
  
  \[
  L_5 = \{ x \mid |x| \geq 2 \}, \quad x \in \{a, b\}^3
  \]

\[
L_{50} = \{ x_0a x_1a x_2a \ldots a x_{50} \mid x_0, \ldots, x_{50} \in \{a, b\}^3 \}
\]
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:

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

\[ \overline{L_4} = \{ \text{strings of length } \geq 5 \} \]

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

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

- **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 $L_1 \circ L_2 = \{\varepsilon ab, abab, abbb, aabb, aabbb, aabbb, aabbb\}$
  
  Initialize $S = \emptyset$
  For each $x \in L_1$:
  - For each $y \in L_2$:
    - Add $xy$ to $S$
  Return $S \quad \# = L_1 \circ L_2$
  
  $|L_1 \circ L_2| \leq |L_1| |L_2|$
A Few “Traps”

String, language, or something else?

\( \varepsilon \)  
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 $\Sigma = \{a, b\}$

On input $x = x_1 x_2 \ldots x_n$

$\text{count} = 0$

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

If $x_i = a$:

$\text{count} = \text{count} + 1$

If $\text{count} \leq 4$: accept

Else: reject

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

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