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

Link to polls:
https://forms.gle/exHaBu5mQgR57NDq5
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?
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?

\[ \Sigma = \quad L = \]

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

\[ \Sigma = \quad L = \]

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

\[ \Sigma = \quad L = \]
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

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

  - 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
  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 can be used: 36 (58.1%)
- I'm excited about the philosophy: 17 (27.4%)
- I want to practice problem solving: 34 (54.8%)
- I want to develop a "computational thinking" mindset: 25 (40.3%)
- I actually wanted to take CS 32: 4 (6.5%)
- I am curious to learn the current state of the art: 1 (1.6%)
- I don't know what it is so wanna check it out: 1 (1.6%)
- Course seemed less awful than I thought: 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

   $\Sigma^*$ = 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
  \[ \text{Ex. } |abba| = \quad |\varepsilon| = \]

• **Concatenation** of strings \(x\) and \(y\), written \(xy\), is the symbols from \(x\) followed by the symbols from \(y\)
  \[ \text{Ex. } x = ab, \ y = ba \quad \Rightarrow \quad xy = \]
  \[ x = ab, \ y = \varepsilon \quad \Rightarrow \quad xy = \]

• **Reversal** of string \(x\), written \(x^R\), consists of the symbols of \(x\) written backwards
  \[ \text{Ex. } x = aab \quad \Rightarrow \quad x^R = \]
Fun with String Operations

What is \((xy)^R\)?

Ex. \(x = aba, y = bba\) \(\Rightarrow xy =\)

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

a) \(x^R y^R\)
b) \(y^R x^R\)
c) \((yx)^R\)
d) \(xy^R\)
Fun proofs with String Operations

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

Proof: Let \(x = x_1 x_2 \ldots x_n\) and \(y = y_1 y_2 \ldots y_m\)

Then \((xy)^R =\)

Not even the most formal way to do this:

1. Define string length recursively
2. Prove by induction on \(|y|\)
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?)
Some Simple Languages

\[\Sigma = \{0, 1\} \quad \Sigma = \{a, b, c\}\]

\(\emptyset\) (Empty set)

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

\(\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}\)  
(All strings of length \(n\))
Some More Interesting Languages

• \( L_1 = \) 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’
Some More Interesting Languages

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

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

$A \cup B \quad \text{“union”}$

$A \cap B \quad \text{“intersection”}$

$\overline{A} \quad \text{“complement”}$
New Languages from Old

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

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

$\Rightarrow L_6 =$
Operations Specific to Languages

• **Reverse**: \( L^R = \{ x^R | x \in L \} \)

  Ex. \( L = \{ \varepsilon, a, ab, aab \} \) \( \Rightarrow L^R = \)

• **Concatenation**: \( L_1 \circ L_2 = \{ xy | x \in L_1, y \in L_2 \} \)

  Ex. \( L_1 = \{ ab, aab \} \quad L_2 = \{ \varepsilon, b, bb \} \)

  \( \Rightarrow L_1 \circ L_2 = \)
A Few “Traps”

String, language, or something else?

\(\varepsilon\)

\(\emptyset\)

\(\{\varepsilon\}\)

\(\{\emptyset\}\)
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 $x = x_1 x_2 \ldots x_n$:

count = 0

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

If $x_i = a$:

\[ \text{count} = \text{count} + 1 \]

If count $\leq 4$: 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) $\{x \in \Sigma^* \mid x \text{ has exactly 4 a's} \}$
Deterministic Finite Automata
A (Real-Life?) Example

• **Example:** Kitchen scale

• $P =$ Power button (ON / OFF)

• $U =$ Units button (cycles through g / oz / lb)
  
  Only works when scale is ON, but units remembered when scale is OFF

• Starts OFF in g mode

• **A computational problem:** Does a sequence of button presses in $\{P, U\}^*$ leave the scale ON in oz mode?
Machine Models

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

Input

\[P \quad U \quad P \quad U \quad \ldots\]

Control scans left-to-right

1) How does the control start?
2) What are the different “states” that the control can be in?
3) When the control reads a new input character, how does it transition to a new state?
4) How do I know if I’m in the desired state at the end?
A DFA for the Kitchen Scale Problem
A DFA Recognizing Parity

The language recognized by a DFA is the set of inputs on which it ends in an “accept” state

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

$\Sigma = \{a, b\} \quad L = \{w \mid w \text{ contains an even number of } a\’s\}$
Anatomy of a DFA