CS 332 : Elements of the Theory of Computation

Lecture 1:

- Overview
- Course information
- Finite automata

Reading:
Sipser Ch. 0, 1.1

Mark Bun

January 22, 2020

Course Information

Course Staff

- Me: **Mark** Bun
  - At BU since Sept. 2019
  - Office hours: Wed 4:00-6:00, MCS 114
  - Research interests: Theory of computation (!)

    More specifically: Computational complexity, data privacy, cryptography, foundations of machine learning

- TF: Nadya Voronova
  - Leading 9:30 and 12:30 discussion sections
  - Office hours: Fri 2:30-4:30, Location TBD

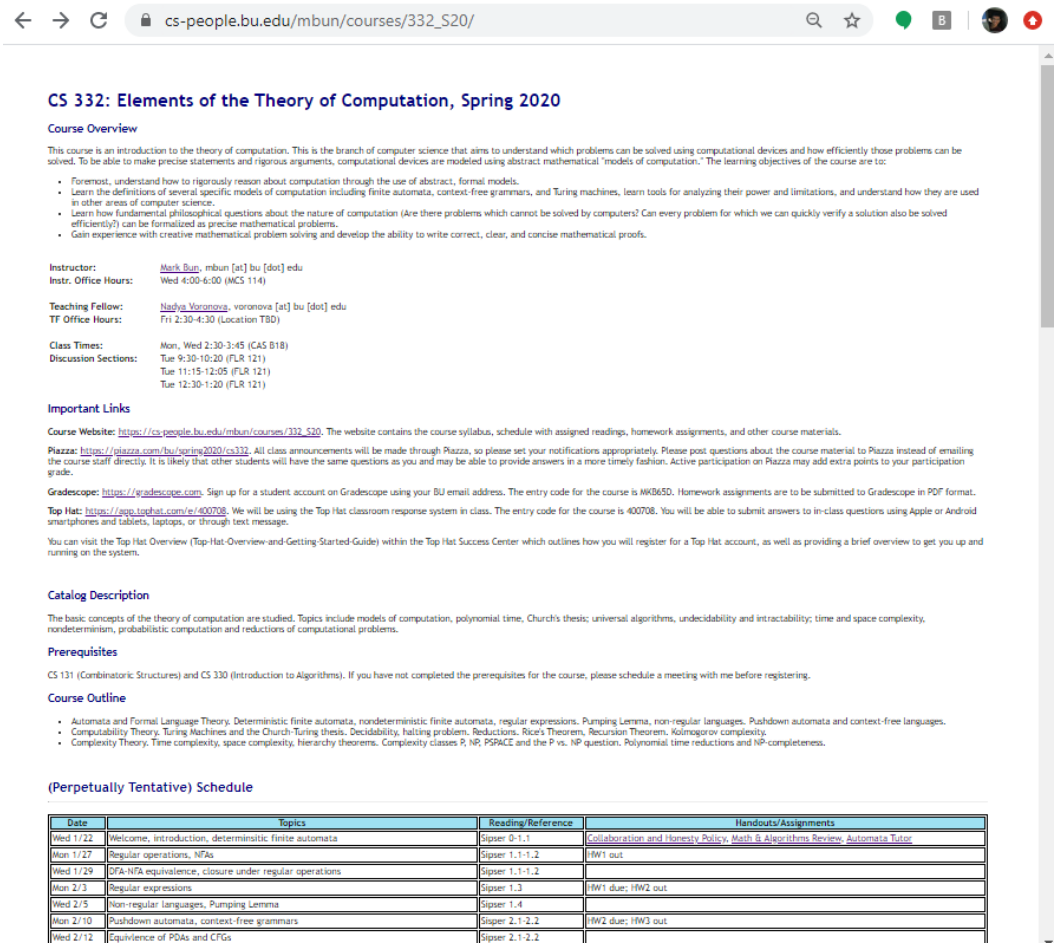
- Tim Jackman
  - Leading 11:15 discussion section

Course Webpage

https://cs-people.bu.edu/mbun/courses/332_S20/

Serves as the syllabus and schedule

Check back frequently for updates!



cs-people.bu.edu/mbun/courses/332_S20/

CS 332: Elements of the Theory of Computation, Spring 2020

Course Overview

This course is an introduction to the theory of computation. This is the branch of computer science that aims to understand which problems can be solved using computational devices and how efficiently those problems can be solved. To be able to make precise statements and rigorous arguments, computational devices are modeled using abstract mathematical "models of computation." The learning objectives of the course are to:

- Foremost, understand how to rigorously reason about computation through the use of abstract, formal models.
- Learn the definitions of several specific models of computation including finite automata, context-free grammars, and Turing machines, learn tools for analyzing their power and limitations, and understand how they are used in other areas of computer science.
- Learn how fundamental philosophical questions about the nature of computation (Are there problems which cannot be solved by computers? Can every problem for which we can quickly verify a solution also be solved efficiently?) can be formalized as precise mathematical problems.
- Gain experience with creative mathematical problem solving and develop the ability to write correct, clear, and concise mathematical proofs.

Instructor: Mark Bun, mbun [at] bu [dot] edu
Instr. Office Hours: Wed 4:00-6:00 (MCS 114)

Teaching Fellow: Nadya Voronova, voronova [at] bu [dot] edu
TF Office Hours: Fri 2:30-4:30 (Location TBD)

Class Times: Mon, Wed 2:30-3:45 (CAS B18)
Discussion Sections: Tue 9:30-10:20 (FLR 121)
Tue 11:15-12:05 (FLR 121)
Tue 12:30-1:20 (FLR 121)

Important Links

Course Website: https://cs-people.bu.edu/mbun/courses/332_S20. The website contains the course syllabus, schedule with assigned readings, homework assignments, and other course materials.

Piazza: https://piazza.com/bu/spring2020/cs332. All class announcements will be made through Piazza, so please set your notifications appropriately. Please post questions about the course material to Piazza instead of emailing the course staff directly. It is likely that other students will have the same questions as you and may be able to provide answers in a more timely fashion. Active participation on Piazza may add extra points to your participation grade.

Gradescope: https://gradescope.com. Sign up for a student account on Gradescope using your BU email address. The entry code for the course is MKB65D. Homework assignments are to be submitted to Gradescope in PDF format.

Top Hat: https://app.tophat.com/e/400708. We will be using the Top Hat classroom response system in class. The entry code for the course is 400708. You will be able to submit answers to in-class questions using Apple or Android smartphones and tablets, laptops, or through text message.

You can visit the Top Hat Overview (Top-Hat-Overview-and-Getting-Started-Guide) within the Top Hat Success Center which outlines how you will register for a Top Hat account, as well as providing a brief overview to get you up and running on the system.

Catalog Description

The basic concepts of the theory of computation are studied. Topics include models of computation, polynomial time, Church's thesis; universal algorithms, undecidability and intractability; time and space complexity, nondeterminism, probabilistic computation and reductions of computational problems.

Prerequisites

CS 131 (Combinatoric Structures) and CS 330 (Introduction to Algorithms). If you have not completed the prerequisites for the course, please schedule a meeting with me before registering.

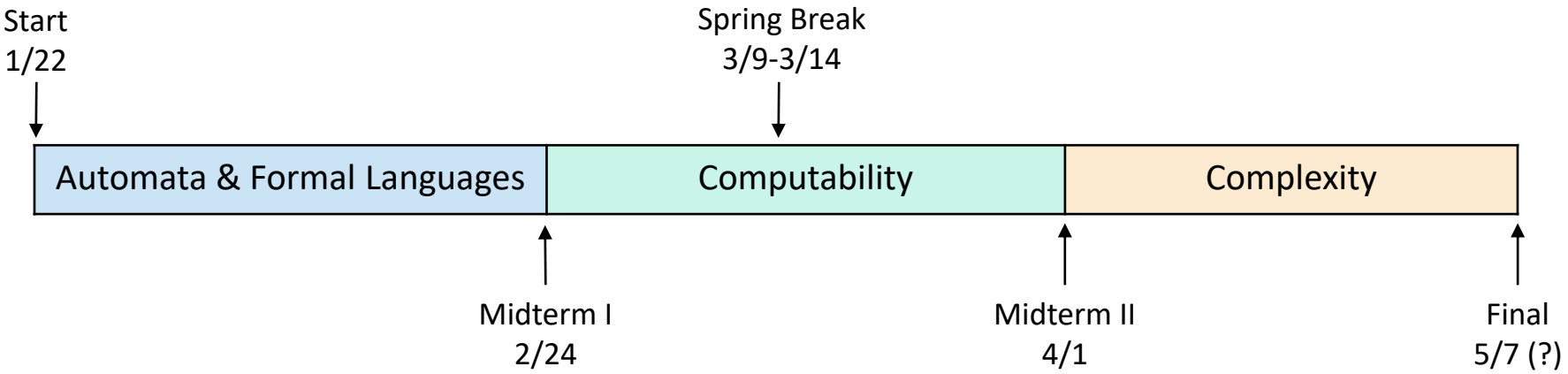
Course Outline

- Automata and Formal Language Theory. Deterministic finite automata, nondeterministic finite automata, regular expressions. Pumping Lemma, non-regular languages. Pushdown automata and context-free languages.
- Computability Theory. Turing Machines and the Church-Turing thesis. Decidability, halting problem. Reductions. Rice's Theorem, Recursion Theorem. Kolmogorov complexity.
- Complexity Theory. Time complexity, space complexity, hierarchy theorems. Complexity classes P, NP, PSPACE and the P vs. NP question. Polynomial time reductions and NP-completeness.

(Perpetually Tentative) Schedule

Date	Topics	Reading/Reference	Handouts/Assignments
Wed 1/22	Welcome, introduction, deterministic finite automata	Sipser 0-1.1	Collaboration and Honesty Policy, Math & Algorithms Review, Automata Tutor
Mon 1/27	Regular operations, NFAs	Sipser 1.1-1.2	HW1 out
Wed 1/29	DFA-NFA equivalence, closure under regular operations	Sipser 1.1-1.2	
Mon 2/3	Regular expressions	Sipser 1.3	HW1 due; HW2 out
Wed 2/5	Non-regular languages, Pumping Lemma	Sipser 1.4	
Mon 2/10	Pushdown automata, context-free grammars	Sipser 2.1-2.2	HW2 due; HW3 out
Wed 2/12	Equivlence of PDAs and CFGs	Sipser 2.1-2.2	

Course Structure



Grading

- Homework (30%): Roughly 10 of these
- Exams (60%):
  - Midterm I (15%)
  - Midterm II (15%)
  - Final (30%)
- Participation (10%): TopHat

Homework Policies

- Weekly assignments due Mondays @ 2PM
- No late days, no extensions
- Lowest homework score will be dropped

- Homework to be submitted via Gradescope
  - Entry code: MKB65D

- You are encouraged to typeset your solutions in LaTeX (resources available on course webpage)

- HW1 to be released on Mon 1/27, due Mon 2/3

Homework Policies: Collaboration

- You are encouraged to work with your classmates to discuss homework problems

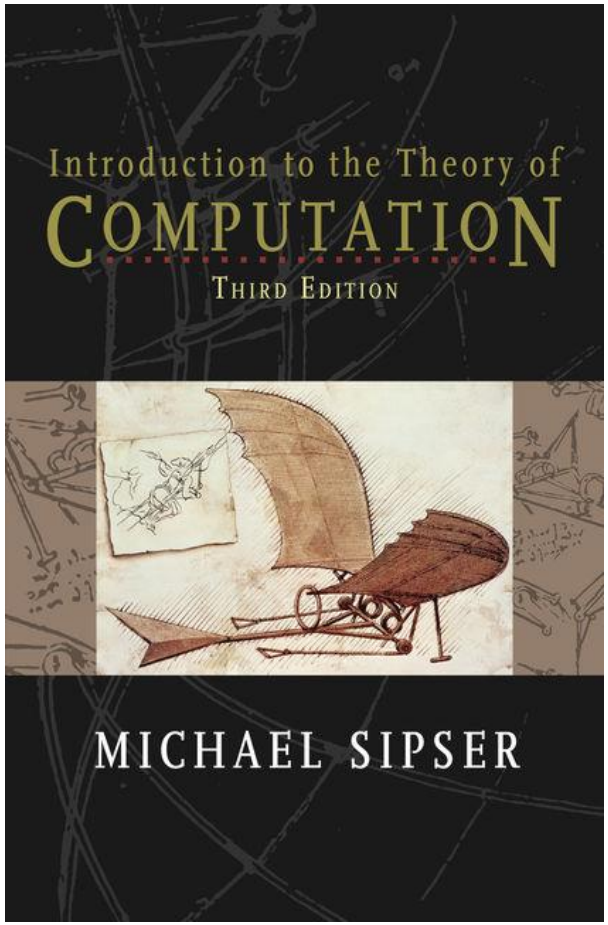
- HOWEVER:
  - You may collaborate with at most 3 other students
  - You must acknowledge your collaborators
  - You must write your solutions by yourself
  - You may not share written solutions
  - You may not search for solutions using the web or other outside resources
  - You may not receive help from anyone outside the course (including students from previous years)

Homework Policies: Collaboration

Details of the collaboration policy may be found here: https://cs-people.bu.edu/mbun/courses/332_S20/handouts/collaboration.pdf

Important: Sign this document to affirm you understand it, and turn it in via Gradescope by 2PM, Mon 1/27

Textbook



Introduction to the Theory of Computation
(Third Edition)
by Michael Sipser

- It's fine if you want to use an older edition, but section numbers may not be the same
- Other resources available on course webpage

Piazza

- We will use Piazza for announcements and discussions
  - Ask questions here and help your classmates
  - Please use private messages / email sparingly

  https://piazza.com/bu/spring2020/cs332

  You can earn bonus points toward your participation grade by participating thoughtfully on Piazza

TopHat

- Your class participation score (10% of the course grade) will be determined by your engagement with TopHat

https://tophat.com/ Entry code: 400708

- You will be graded only on participation, not correctness
- Answering 80% of the questions in TopHat will earn you a full participation grade

Expectations and Advice for Succeeding in CS 332

Our (the Course Staff's) Responsibilities

- Guide you through difficult parts of the material in lecture
- Encourage active participation in lectures / section
- Assign practice problems and homework that will give you a deep understanding of the material
- Give detailed (formative) feedback on assignments
- Be available outside of class (office hours, Piazza)
- Regularly solicit feedback to improve the course

  Anonymous feedback to Mark:
  https://forms.gle/AV38CBgLKTSBmtqRA

Your Responsibilities

- Concepts in this course take some time to sink in. Keep at it, and be careful not to fall behind.
- Do the assigned reading on each topic before the corresponding lecture.
- Take advantage of office hours.
- Participate actively in lectures/sections and on Piazza.
- Allocate lots of time for the course: comparable to a project-based course, but spread more evenly.

Prerequisites

This class is fast-paced and assumes experience with mathematical reasoning and algorithmic thinking

You must have passed CS 330 – Intro to Algorithms

This means you should be comfortable with:

- Set theory
- Functions and relations
- Graphs
- Pigeonhole principle
- Propositional logic
- Asymptotic notation
- Graph algorithms (BFS, DFS)
- Dynamic programming
- NP-completeness

Come talk to me if you have questions about your preparation for the course

Advice on Homework

- Start working on homework early! You can get started as soon as it's assigned.
- Spread your homework time over multiple days.
- You may work in groups (of up to 4 people), but think about each problem for at least 30 minutes before your group meeting.

- To learn problem solving, you have to do it:
  - Try to think about how you would solve any presented problem before you read/hear the answer
  - Do exercises in the textbook in addition to assigned homework problems

Advice on Reading

- Not like reading a novel
- The goal is not to find out the answers, but to learn and understand the techniques
- Always try to predict what's coming next
- Always think about how you would approach a problem before reading the solution
- This applies to things that are not explicitly labeled as exercises or problems!

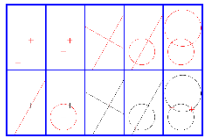
Course Overview

Objective

Build a *theory* out of the idea of *computation*

What is “computation”

- Examples:
  - Paper + pencil arithmetic
  - Abacus
  - Mechanical calculator
  - Ruler and compass geometry constructions
  - Java/C programs on a digital computer



- For us: Computation is the processing of information by the unlimited application of a finite set of operations or rules

What do we want in a "theory"?

- General ideas that apply to many different systems
- Expressed simply, abstractly, and precisely

- Generality
  - Independence from Technology: Applies to the future as well as the present
  - Abstraction: Suppresses inessential details

- Precision: Can prove formal mathematical theorems
  - Positive results (what *can* be computed): correctness of algorithms and system designs
  - Negative results (what *cannot* be computed): proof that there is no algorithm to solve some problem in some setting (with certain cost)

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

This course: Sequential, single-processor computing

Not covered:

- Parallel machines
- Distributed systems
- Quantum computation
- Real-time systems
- Mobile computing
- Embedded systems

What is a (Computational) Problem?

A single question with infinitely many instances

Examples:

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

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

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

For us: Focus on *decision* problems (yes/no answers) on *discrete* inputs

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. $\Sigma = \{a, b, \ldots, z\}$

- **String:** A finite concatenation of alphabet symbols (order matters)

  Ex. $bqr, ababb$ $\neq baabb$

  ε denotes empty string, length 0

  Σ^* = set of all finite strings over Σ

- **Language:** A (possibly infinite) set L of strings $L \subseteq \Sigma^*$

Examples of Languages

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

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

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

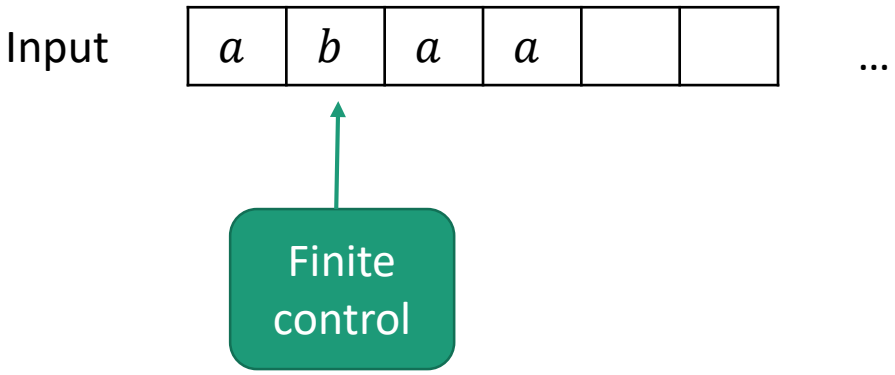
Σ = $\{0, 1\}$ L = $\{ x \mid x$ is a prime (written in binary)$\}$

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

Σ = $\{$ASCII characters$\}$ L = $\{ P \mid P$ gets stuck in a loop$\}$

Machine Models

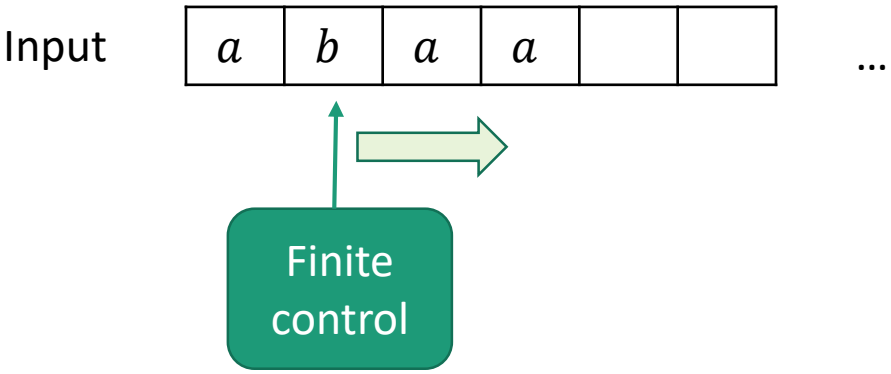
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



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

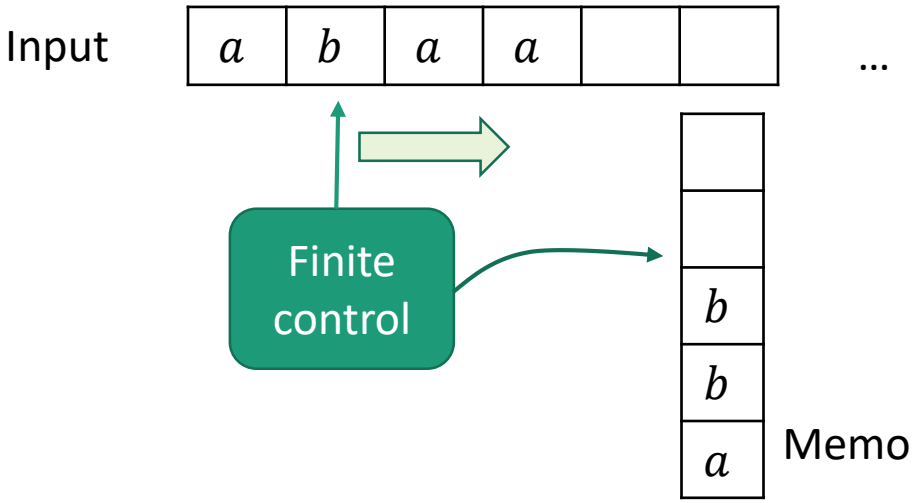


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

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

Machine Models

- Pushdown Automata (PDAs): Machine with unbounded structured memory in the form of a stack

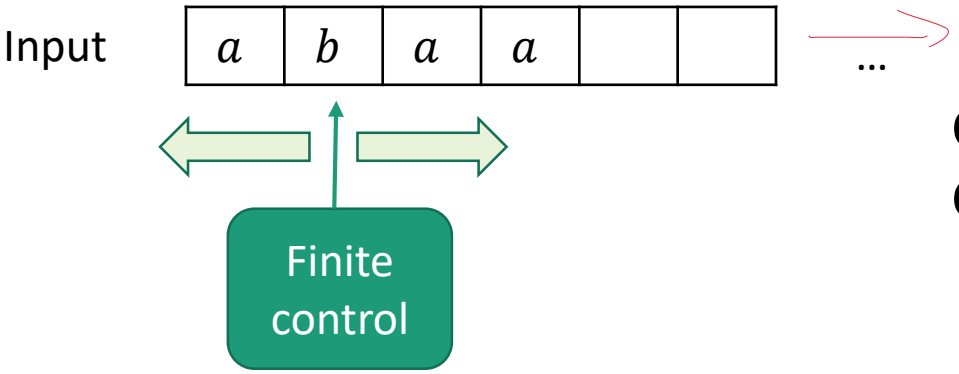


Control scans left-to-right
Can use stack to count, balance parentheses

Useful for modeling parsers, compilers, some math calculations

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 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 will learn how to formally reason about computation
- You will 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 will learn how to formally reason about computation
- You will learn the technology-independent foundations of CS

Connections to other parts of science:

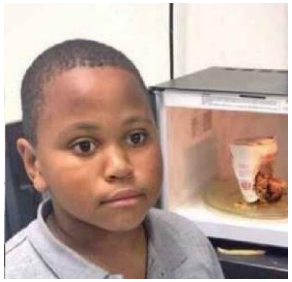
- Finite automata arise in compilers, AI, coding, chemistry

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.

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…