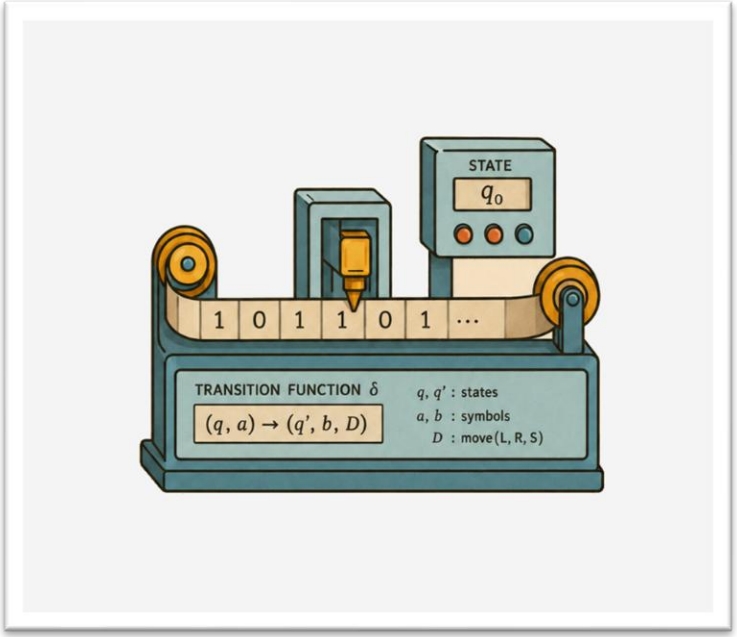
BU CS 332 – Elements of the Theory of Computation

- **Lecture 1:**
  - Course information
  - Overview
  - Strings and languages



Alexander Poremba

September 3, 2026

Course Staff

Instructor:

Alexander Poremba

Research interests: quantum computation and its intersection with theoretical computer science, cryptography, and physics.

Hobbies: Sailing, music, hiking, gardening

Teaching Assistant:

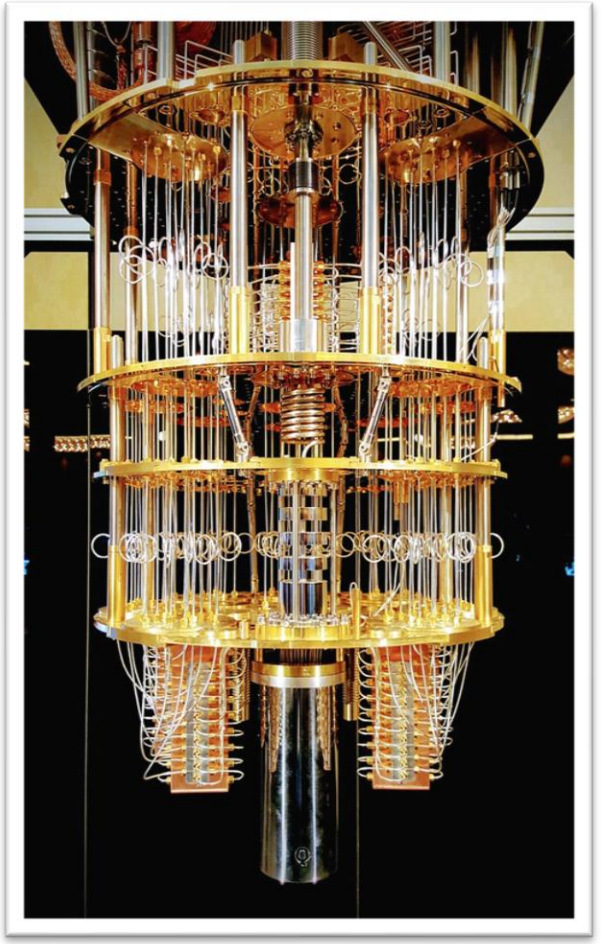
Rafay Cheema

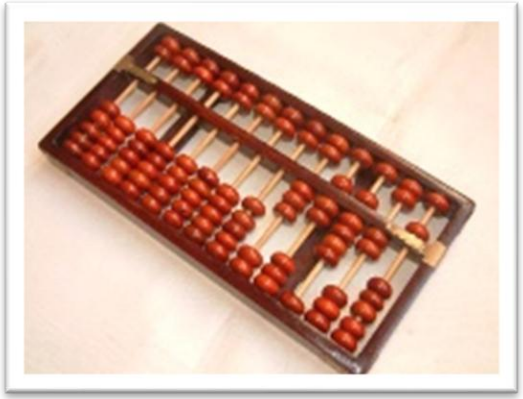
Computer science
senior undergrad

Interests: developing web apps with React, developing video games (Unity/C#), and competing in hackathons

Why study computation?

Computation Is All Around Us





Why study the theory of computation?

The Science of What Computers Can and Cannot Do

The theory of computation allows us to explore deep, philosophical questions about computation:

- Are certain models of computation strictly more powerful than others?
- Are there problems that no computer (no matter how powerful) can solve?

AI Solved A Mathematical Problem That Had Stumped The World's Best Minds For Decades

By Anisha Sircar, Contributor. Anisha Sircar is a journalist covering tech, fin... Follow Author

Published Apr 17, 2026, 06:15am EDT

Add Us On Google

0:00 / 6:23

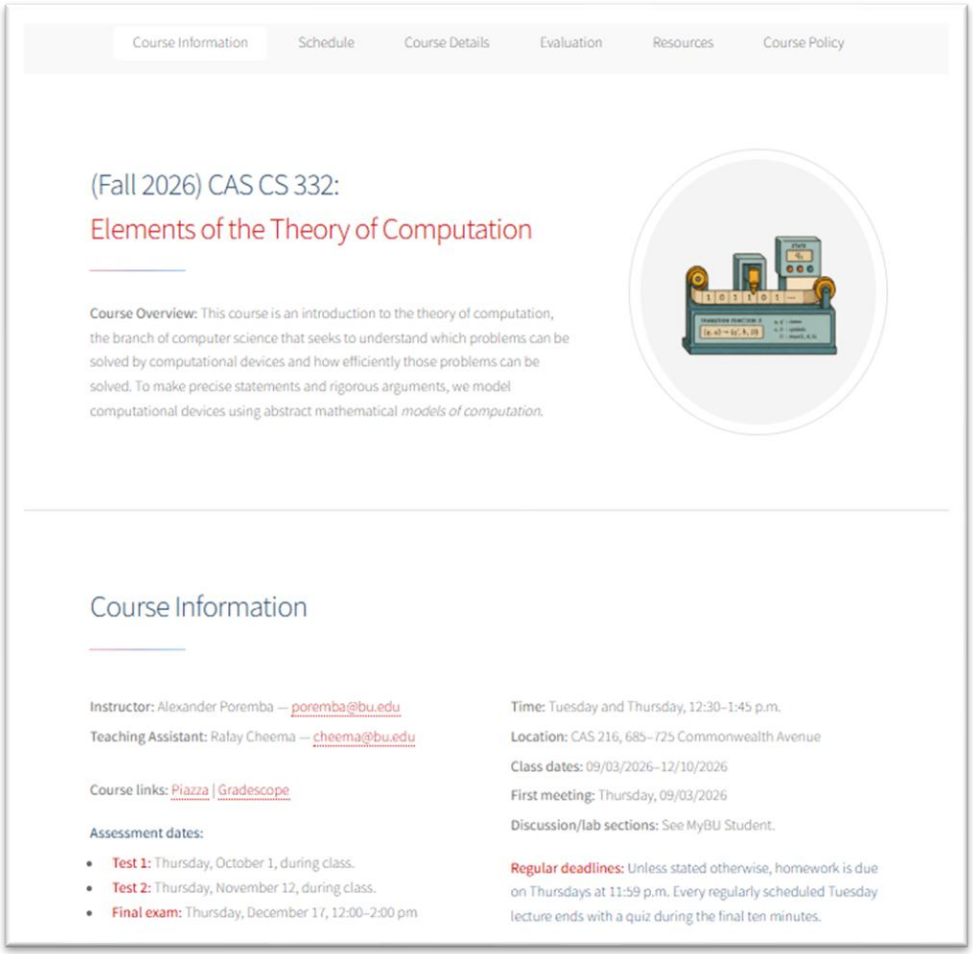
This voice experience is generated by AI. Learn more.

Course information

Course Webpage

https://scc1.bu.edu/poremba/courses/Fall%202026/index.html

Check back frequently
for updates and schedule!



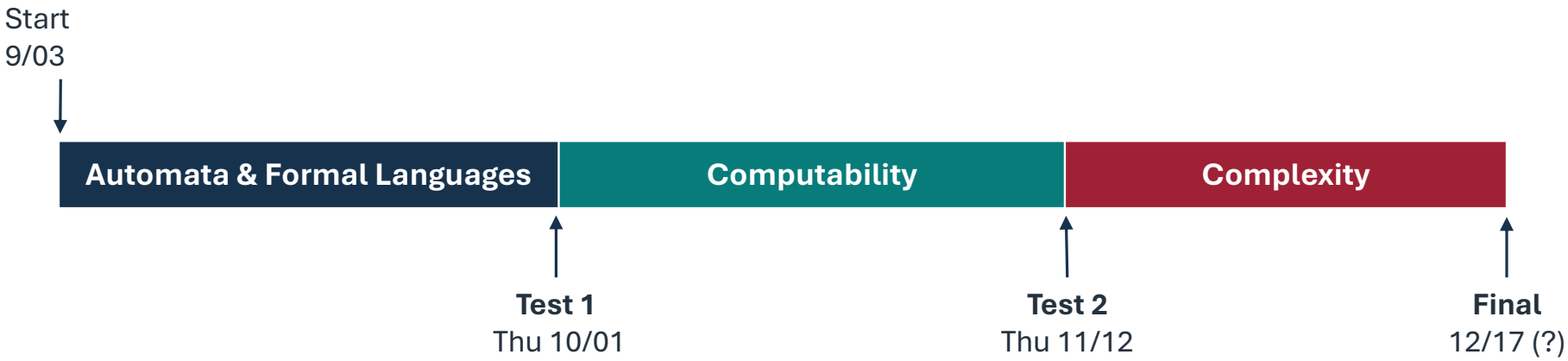
Piazza

We will use **Piazza** for announcements and discussions

- Ask questions here and help your classmates
- Please use private messages / email sparingly
- Check frequently for new updates

https://piazza.com/bu/fall2026/cs332

Course Structure



Grading

- **In-class tests** (60%):
  - Test 1 (17%)
  - Test 2 (17%)
  - Final (26%)
- **Homework** (15%): Roughly 11 of these
- **Quizzes** (15%): In-class, at the end of every Tuesday lecture
- **Participation** (10%): asking questions, being active in discussions

Homework Policies

- Weekly assignments due **Thursday @ 11:59PM**

- No late days
- But: lowest homework score will be dropped

- Homework to be submitted via **Gradescope**
  - Entry code: **NR86EN**

- You may typeset your solutions in LaTeX (resources available on course webpage) or submit handwritten solutions

- HW0 out, due Thu 09/10 (just housekeeping)

The Importance of Doing Your Homework

- When working on a homework problem, it is tempting to skip ahead to the solution…
- **BUT**: Merely understanding a solution is not the same as coming up with one yourself
- The process of struggling with a problem, testing ideas, and designing a solution is an important part of the learning experience

Homework Policies: Collaboration

- You are encouraged to work with classmates to discuss homework problems
- HOWEVER:
  - You may collaborate with at most 3 other students
  - You must acknowledge your collaborators and write “Collaborators: none” if you worked alone
  - You must write your solutions by yourself
  - You may not share written solutions
  - You may not search for solutions on the web or use **AI chatbots** to produce solutions for you
  - You may not receive help from anyone outside the course (including students from previous years)
- Some problems will be marked “INDIVIDUAL”.

  No collaboration is allowed on these problems.

Question: Collaboration Dilemma

If you worked *alone* on a homework assignment…

(a) You don't need to include a collaboration statement

(b) You should invent a fake collaborator and acknowledge them appropriately

(c) You should include the collaboration statement "Collaborators: none"

(d) You should hurriedly call up three of your friends in the class at 11:55PM, briefly discuss the problems, and acknowledge them

Academic Integrity

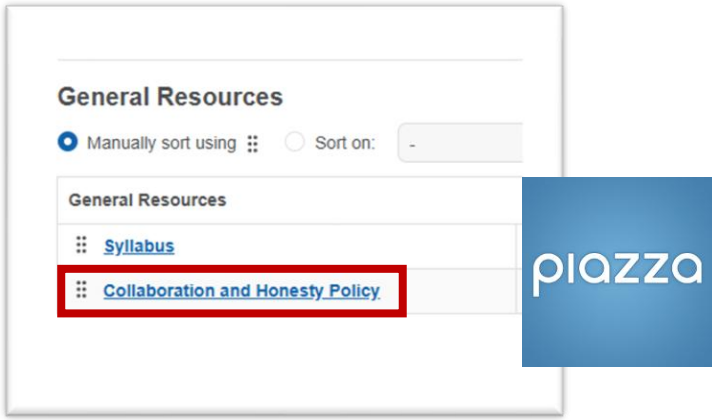
Extremely important: Read and understand the Collaboration and Honesty policy before you sign it

Violations of the collaboration policy…will result in an automatic failing grade and will be reported to the Academic Conduct Committee (ACC). The ACC often suspends or expels students deemed guilty of plagiarism or other forms of cheating.

If you find yourself in a desperate situation:

- Hand in as much of the assignment as you're able to complete
- Remember the lowest HW grade is dropped
- Talk to us! We want to help

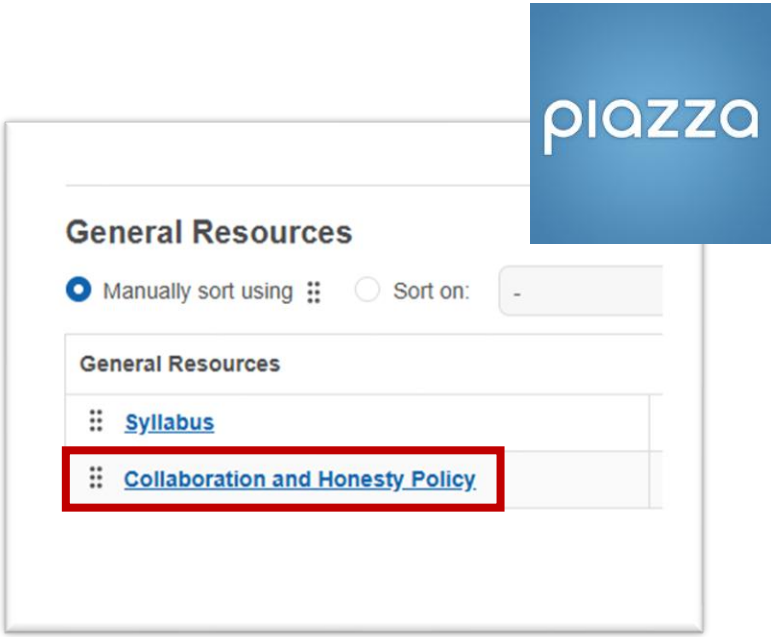
…cheating is seriously not worth it



Homework Policies: Collaboration

You can find the collaboration policy on **Piazza**

Important: Sign this document to affirm you understand it, and turn it in via **Gradescope** by 11:59PM, Thu 09/10



Quizzes

- An in-class quiz will take place during the **final ten minutes** of every Tuesday lecture. See schedule on the website.

Schedule

Date	Topic
Thu 09/03	Welcome and introduction
Tue 09/08 **(Quiz)**	Sets, strings, and languages
Thu 09/10	Finite automata
Tue 09/15 **(Quiz)**	More on NFAs; DFA–NFA equivalence
Thu 09/17	Closure properties and regular expressions
Tue 09/22 **(Quiz)**	Regular expressions versus finite automata

Quizzes

- The quizzes count **15%** toward your final grade.
- Quizzes are written tests (closed-book and closed-device).
- They will contain short questions about lecture material from the previous week & the discussions/labs.
- The questions are supposed to be simple checks that you are following the course.

Quizzes

- If you actively follow the lectures, attend discussions/labs, and complete the homeworks, you should not need to devote additional study time to prepare for the quizzes.
- If you miss a Tuesday lecture, you will receive a score of zero for that day's quiz. It will not be possible to re-take a quiz.
- At the end of the semester, your three lowest quiz grades will be dropped to account for unexpected circumstances.
- Please reach out to the instructor if you need special accommodations.

Participation

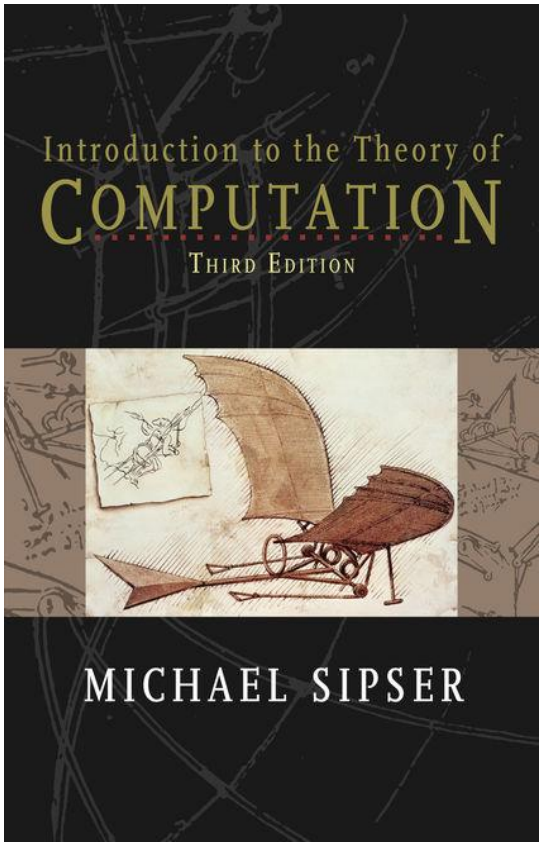
- Your class participation score (10% of the grade) will be determined by
  - Actively participating in class
  - Attending discussions & completing the worksheet during each discussion (at minimum, 80% attendance)
  - Completing other activities, e.g., course feedback forms

- You can also increase your participation score by participating thoughtfully and regularly in discussions, office hours, and on Piazza

Discussions / Labs

- Discussions take place on Wednesdays (check MyBU Student)
  - **Lab #A:** 10.10am – 11am, CDS B62
  - **Lab #B:** 11.15am – 12.05pm, KCB 104
- The discussions are run by the TA (Rafay Cheema).
- There will be a weekly in-class discussion worksheet with individual (relevant for quizzes!) and collaborative problems
- At the end of each lab, the TA will keep track of attendance and check whether you have completed your worksheet

Suggested 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

Expectations and Advice for Succeeding in CS 332

Learning Objectives

- Understand how to rigorously reason about computation using mathematical models
- Learn about several specific models of computation, how to analyze them, and how they are used throughout computer science
- Learn how to pose deep philosophical questions about the nature of computation as precise mathematical problems
- Gain experience with problem solving and develop proof-writing skills

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

Office hours will be announced soon (stay tuned!)

Your Responsibilities

- Concepts in this course take 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 your instructors if you have concerns about your preparation for the course

More Advice on Homework

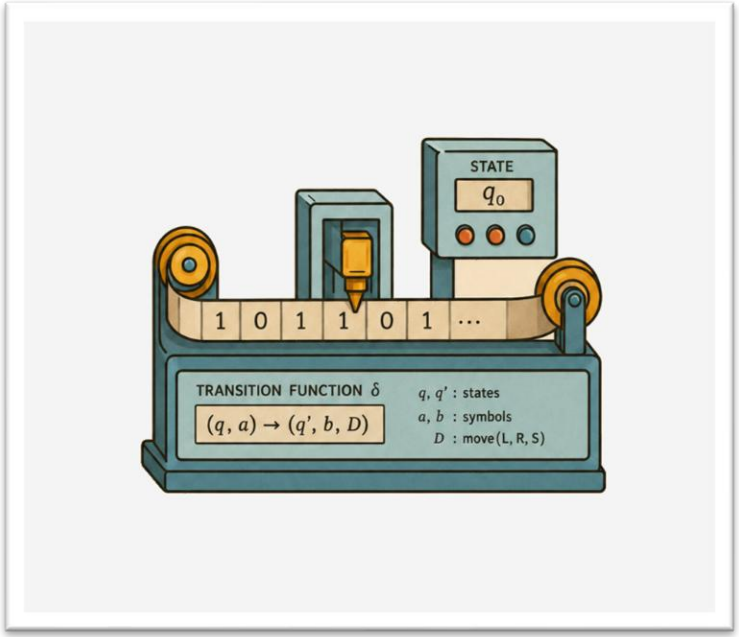
- Start working on homework early! You can get started as soon as it's assigned.
- Spread your homework time over multiple days.
- You may work in groups (of up to 4 people) **but** think about each problem for at least 30 minutes before your group meeting.

- 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

Course Overview

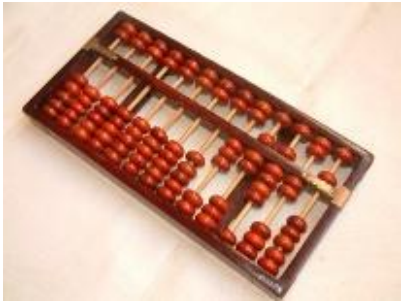
Objective

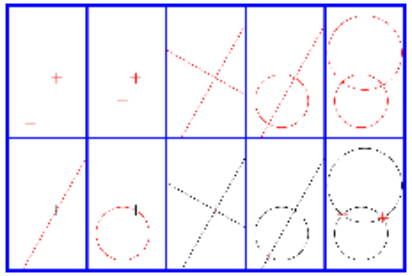
Build a *theory* out of the idea of *computation*



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:

abba

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

In this class, we focus on **decision problems** (yes/no answers) on *discrete* inputs

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

- **Alphabet**: A finite set Σ

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

Ex: PARITY $\Sigma = \{a, b\}$

PRIMALITY $\Sigma = \{0, 1\}$

- **String:** A finite concatenation of alphabet symbols

  Ex. $bqr, ababb$

  ε denotes empty string, length 0

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

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

- **Language:** A set L 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?

$\Sigma = \{a, b\}$ $\quad L = \{ x \in \{a,b\}^* \mid x$ has an even number of a's $\}$

$abba \in L$ $\quad abb \notin L$

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

$\Sigma = \{0, 1\}$ $\quad L = \{ x \in \{0,1\}^* \mid x$ is a binary repr. of a prime number $\}$

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

$\Sigma =$ ASCII $\quad L = \{ x \in \Sigma^* \mid x$ is a C program that gets stuck in an infinite loop $\}$

Question: Primality language

Which best represents the language corresponding to the Primality problem? (i.e., strings over the alphabet $\{0, 1\}$ that are binary representations of prime numbers.)

Let's say the most significant bit is on the left, so "100" is the binary representation of 4.

(a) $\{2, 3, 5, 7, \ldots\}$

(b) $\{10, 11, 101, 111, \ldots\}$

(c) $\{11, 111, 11111, 1111111, \ldots\}$

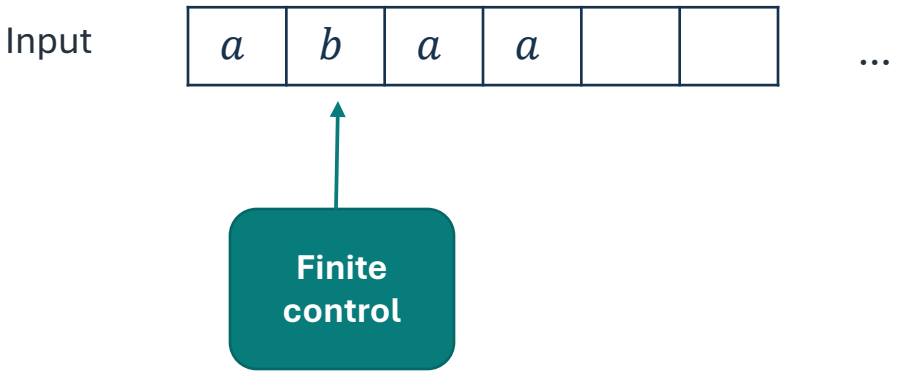
(d) $\{11, 011, 101, 110, 111, 0111, \ldots\}$

2, 3, 5, ...

10 11 101 ...

Machine Models

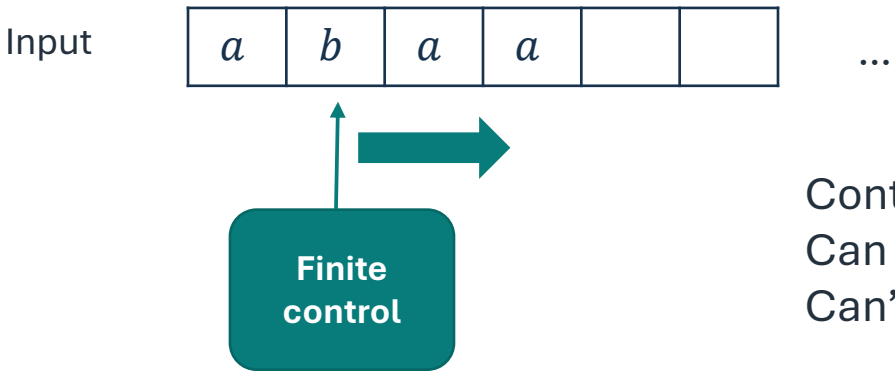
Computation is the processing of information by the unlimited application of a finite set of operations or rules



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

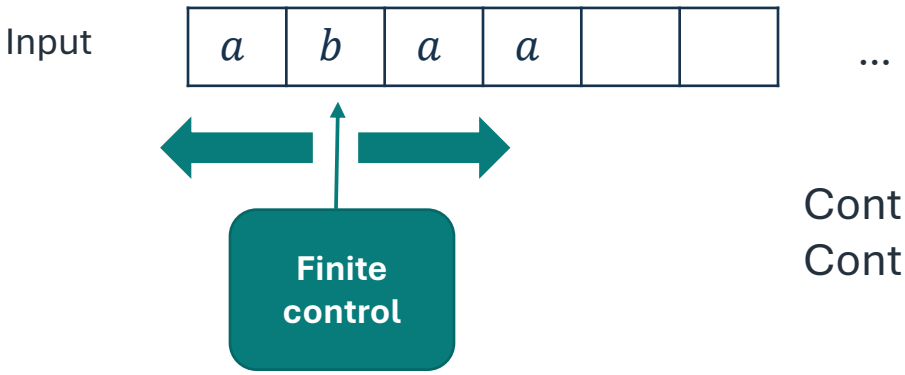


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

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

Machine Models

- 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'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.

Why do you want to study the theory of computation?

(a) I want to learn new ways of thinking about computation

(b) I like math and want to see how it's used in computer science

(c) I'm excited about the philosophical questions about the nature of computation

(d) I want to practice problem solving and algorithmic thinking

(e) I want to develop a "computational perspective" on other areas of math/science

(f) I actually wanted to take CS 320 or CS 350 but they were full