CMPU 240

Theory of Computation

Fall 2023
What’s a computer?
“Hey, whatcha doing on your computer?”

“What’s a computer?”
Seriously, what is a computer?
🤔
314

+ 159

_____
1
314
+ 159
3
\[
\begin{array}{c}
1 \\
\hline
314 \\
+ 159 \\
\hline
73 \\
\end{array}
\]
\[
1
\]
\[
314
\]
\[
+ 159
\]
\[
473
\]
A small suan pan (Chinese abacus)

Photo from the Computer History Museum
Hand-cranked Curta calculator, c. 1950
Photo from the Computer History Museum
Some kinds of computers have more computational power than others.

We can abstract devices of the “same kind” to produce *models* of computers and ask what kinds of problems can be solved under a particular model.
“One of the most remarkable things about computers is that their essential nature transcends technology.”

Present-day computers are built out of transistors and wires.

But they could be built – according to the same principles – from valves and water pipes or from sticks and strings.
Mechanical implementation of the or function
Hydraulic implementation of the \texttt{or} function

W. Daniel Hillis,
\textit{The Pattern on the Stone},
1998
Computer technology changes quickly.

Studying theory enables you to understand the underlying models of *all* computation – not just technical details that become outdated in a few years.
A central idea in the theory of computation is that of a *universal computer*, a computer powerful enough to simulate any other computing device.

Most computers we encounter in everyday life are universal computers.

With the right software – and enough time and memory – they can simulate any other type of computer…
Universal computers
The idea of a universal computer was recognized and described in 1937 by Alan Turing.¹

He called it a “universal machine” since at the time, “computer” still meant “a person who performs computations”.

¹ Poor Alonzo Church is a footnote. Where’s his movie?
The idea of a universal computer was recognized and described in 1937 by Alan Turing.¹

He called it a “universal machine” since at the time, “computer” still meant “a person who performs computations”.

¹ Poor Alonzo Church is a footnote. Where’s his movie?
While a universal computer can compute anything that can be computed any other computing device, there are some things that are just impossible to compute.
Questions for which we lack data

“What is the winning number in tomorrow’s lottery?”
Vague questions

“What is the meaning of life?”

But there are also flawlessly defined computational problems that are impossible to solve.

We call these problems *noncomputable*. 
What exactly are the limits to what a computer can do?

We'll work to an answer of this over the semester, taking us through philosophically interesting topics including nondeterminism, Turing machines, computability, and Gödel's incompleteness theorem.
Course information
CMPU-240 Student information
Please fill out this short form to help me better prepare for the start of the semester.

jgordon@vassar.edu Switch account
* Indicates required question

Email *

☐ Record jgordon@vassar.edu as the email to be included with my response

What name would you like me to call you? *
Your answer

What are your preferred pronouns? (optional)
Your answer

forms.gle/BkUNT7R6MihSAVsf8
Prerequisites

CMPU 102: Data Structures and Algorithms

CMPU 145: Foundations of Computer Science
# CMPU 240

## Theory of Computation

**Fall 2023**

- **Tuesday**: 1:30–2:45 p.m.
- **Thursday**: 1:30–2:45 p.m.
- Sanders Physics 105

**Professor Gordon**

<table>
<thead>
<tr>
<th>Models of computation</th>
<th>Tuesday</th>
<th>Thursday</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Read Syllabus</td>
<td>Aug. 29</td>
<td>Aug. 31</td>
</tr>
<tr>
<td>- Read How to Succeed</td>
<td>Theory of computation</td>
<td>Finite automata</td>
</tr>
<tr>
<td>- Read §50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Assignment 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[cs.vassar.edu/~cs240](https://cs.vassar.edu/~cs240/)
Grading

Assignments: 25%
Grading

<table>
<thead>
<tr>
<th></th>
<th>Assignments</th>
<th>Exam 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage</td>
<td>25%</td>
<td>25%</td>
</tr>
</tbody>
</table>
Grading

- Assignments: 25%
- Exam 1: 25%
- Exam 2: 25%
Grading

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>25%</td>
</tr>
<tr>
<td>Exam 1</td>
<td>25%</td>
</tr>
<tr>
<td>Exam 2</td>
<td>25%</td>
</tr>
<tr>
<td>Exam 3</td>
<td>25%</td>
</tr>
</tbody>
</table>
Your Materials

We've made it easy to get your course materials for each class. Just review and select below.

**PRICE MATCH GUARANTEE IS HERE!** — Excludes Marketplace items — Learn more

Course 1 of 1: Fall 2023 • Vassar College • CMPU 240-01 GORDON/JONATHAN • 08/26/2023 - 12/15/2023

**REQUIRED**

Introduction to the Theory of Computation

3RD 13

Author:
Sipser, Michael

ISBN-13:
978-1-133-18779-0

ISBN-10:
1-133-18779-X

3RD 13

Publisher:
Course Technology, Inc.

This product is included in Cengage Unlimited. Save and simplify when you purchase a Cengage Unlimited plan.

- **Print**
  - $245.50 Used Print
  - $327.25 New Print

- **Digital**
  - Digital Rights
  - A digital delivery fee of $3.99 will be applied to each applicable digital material.
    - $44.49 180-Day eRental
    - $60.99 365-Day eRental
    - $77.49 Perpetual Offline Access

- **Rental**
  - $141.75 Used Print Rental
    - Return by 10/22/2023.
  - $204.75 New Print Rental

**Marketplace Sellers**
- From $38.29
- View Sellers

[Image and text content]
"THIS IS THE BEST BOOK ON COMPUTERS I HAVE EVER READ."
—PETER THOMAS, NEW SCIENTIST

THE PATTERN ON THE STONE

THE SIMPLE IDEAS THAT MAKE COMPUTERS WORK

W. DANIEL HILLIS
Hillis’s Connection Machine *CM-2a*

Photo by Steve Grohe for Thinking Machines Corporation
Hillis’s Connection Machine CM-5 in Jurassic Park
Generative AI

You *can* use ChatGPT, Bing Chat, Claude, etc. when studying or working on homework assignments (with acknowledgment).

You *cannot* use it on exams.

Be sure that you’re using it to *help* you learn the material, not as a way to *avoid* learning!
Welcome!  #1

Hi everyone,

We're using Ed Discussion for class Q&A.

This is the best place to ask questions about the course, whether curricular or administrative. You will get faster answers here from staff and peers than through email.

Here are some tips:

- Search before you post
- Heart questions and answers you find useful
- Answer questions you feel confident answering
- Share interesting course related content with staff and peers

For more information on Ed Discussion, you can refer to the Quick Start Guide.

All the best this semester!

Prof. Gordon
CMPU 240

Theory of Computation

Fall 2023

Tuesday 1:30–2:45 p.m.
Thursday 1:30–2:45 p.m.
Sanders Physics 105

Professor Gordon

cs.vassar.edu/~cs240

Overview

Are there fundamental restrictions to what computers can and cannot do? If so, what do these restrictions look like? And what would such restrictions mean for our ability to computationally solve meaningful problems?
What problems can we solve with a computer?
What problems can we solve with a computer?

What kind of computer?
Two challenges

Computers are dramatically better now than they’ve ever been, and we expect that trend to continue!

Writing proofs on formal definitions is hard, and today’s computers are already way more complicated than the mathematical structures you wrote proofs about in CMPU 145.
How can we prove what computers can and can’t do…

…so our results are still true in 20 years?

…without multi-hundred-page proofs?
Enter automata

An *automaton* is a mathematical model of a computing device.

It’s an abstraction of a real computer, like how graphs are abstractions of social networks, transportation grids, etc.
The automata we’ll explore are

Powerful enough to capture huge classes of computer devices, but
Simple enough that we can reason about them in a small space!
What do these automata look like?
A tale of two computers
A tale of two computers
A tale of two computers
<table>
<thead>
<tr>
<th></th>
<th>Basic Calculator</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>Small amount of memory</td>
<td>Large amount of memory</td>
</tr>
<tr>
<td>Functions</td>
<td>Fixed set of functions</td>
<td>Reprogrammable; run lots of different programs</td>
</tr>
</tbody>
</table>
**Basic Calculator**

- Small amount of memory
- Fixed set of functions

**Laptop**

- Large amount of memory
- Reprogrammable; run lots of different programs
Computing with finite memory
Data stored electronically

Algorithm is in silicon

Memory limited by display
Data stored electronically
Algorithm is in silicon
Memory limited by display
Data stored electronically
Algorithm is in silicon
Memory limited by display

Data stored in wood
Algorithm is in your brain
Memory limited by beads
How do we model “memory” and “an algorithm” when they can take on so many forms?
Focus on what’s in common:

The machines *receive input* from an external source.
That input is provided *sequentially*, one discrete unit at a time.
Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!
Once all input is provided, we can *read off an answer* based on the configuration of the device.
Finite automata
A *finite automaton* – also called a *finite-state machine* – is the simplest model of a computer that’s still interesting to study.
Each finite automaton consists of a set of states connected by transitions.
Each circle represents a state of the automaton
One state is designated as the start state, indicated by an arrow.
A deterministic finite automaton (DFA) with four states: $q_0$, $q_1$, $q_3$, and $q_2$. The transitions are as follows:

- From $q_0$: On 0, go to $q_1$; on 1, go to $q_3$.
- From $q_1$: On 0, go to $q_0$; on 1, go to $q_2$.
- From $q_2$: On 0, go to $q_1$; on 1, go to $q_3$.
- From $q_3$: On 0, go to $q_2$; on 1, go to $q_0$.

The initial state is $q_0$. The accepting states are $q_0$, $q_1$, and $q_2$. The input string is 010110.
The automaton is run on an input string.
The automaton can be in one state at a time. It begins in the start state.
The automaton now begins processing characters in the order in which they appear.
Each arrow in this diagram represents a transition. The automaton always follows the transition for the symbol being read.
A finite automaton with transitions:
- Start at state $q_0$.
- From $q_0$, on input 0, transition to $q_1$.
- From $q_1$, on input 0, transition to $q_0$.
- From $q_0$, on input 1, transition to $q_3$.
- From $q_3$, on input 1, transition to $q_2$.
- From $q_2$, on input 1, transition to $q_3$.
- From $q_3$, on input 0, transition to $q_2$.

Input sequence: 0 1 0 1 1 0.
After transitioning, the automaton considers the next symbol in the input.
The image shows a finite state machine (FSM) with four states: $q_0$, $q_1$, $q_3$, and $q_2$. The states are connected by transitions labeled with symbols 0 and 1. The FSM starts in state $q_0$. The transitions are as follows:

- From $q_0$ to $q_1$: 0
- From $q_0$ to $q_3$: 1
- From $q_1$ to $q_2$: 0
- From $q_3$ to $q_2$: 0
- From $q_2$ to $q_3$: 0
- From $q_2$ to $q_0$: 1

The input sequence is 0 1 0 1 1 0, indicating transitions through the states as follows:

- Start in $q_0$.
- Move to $q_1$ on input 0.
- Move to $q_3$ on input 1.
- Move to $q_2$ on input 0.
- Move to $q_3$ on input 0.
- Move to $q_2$ on input 0.
- Move to $q_0$ on input 1.

The sequence completes the cycle.
The diagram represents a finite state machine (FSM) with states $q_0$, $q_1$, $q_3$, and $q_2$. The transitions are labeled with inputs 0 and 1. The start state is $q_0$. The input sequence is 010110.
A finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with 0s and 1s, and the start state is $q_0$.
Now that the automaton has read all of the input, it can decide whether to accept or reject.
Now that the automaton has read all of the input, it can decide whether to accept or reject. The double circle indicates this is an accept state, so it accepts!
Illustration by
Gemma Correll
Let’s try another input

1 0 1
Let’s try another input
A finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The start state is $q_0$. Transitions are as follows:
- From $q_0$ to $q_1$ on input 0
- From $q_0$ to $q_3$ on input 1
- From $q_1$ to $q_2$ on input 1
- From $q_2$ to $q_0$ on input 0
- From $q_3$ to $q_0$ on input 0

The input sequence is 101.
The image shows a finite state machine (FSM) with four states: $q_0$, $q_1$, $q_2$, and $q_3$. The FSM is labeled with transitions labeled 0 and 1. The initial state is $q_0$, and there is a transition from $q_0$ to $q_1$, from $q_1$ to $q_2$, from $q_2$ to $q_0$, and from $q_3$ to $q_2$. The bottom of the image includes a sequence of 1, 0, 1, which is likely the input sequence that the FSM processes.
This state is not an accept state, so the automaton rejects.
This state is not an accept state, so the automaton rejects.
A finite automaton consists of a set of states connected by transitions.

One state is designated the start state.

Some states are accept states.

Transition arcs are labeled with one or more symbols from some alphabet.
An automaton processes a string by beginning in the start state and following the indicated transitions.

The new state is completely determined by the current state and the symbol it just read.

When the input is exhausted,

If the automaton is in an accept state, it accepts the input.
Otherwise, it rejects the input.
A finite automaton does *not* accept as soon as it enters an accept state.

It only accepts if it *ends* in an accept state.
Finite-state machines are all around us.
Finite automaton for a newspaper vending machine
Other real-world finite automata?
Finite automata in action

Used in

  - Text editors and search engines for pattern matching
  - Compilers for lexical analysis
  - Web browsers for HTML parsing

Serve as the control unit in many physical systems, including

  - Elevators, traffic signals, vending machines
  - Computer microprocessors
  - Network protocol stacks and old VCR clocks

Play a key role in natural language processing and machine learning

  *Markov chains* are probabilistic FAs used in part of speech tagging, speech processing, and optical character recognition
A final note
Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame
W. Daniel Hillis, *The Pattern on the Stone*
John Hopcroft, Cornell University
Nancy Ide, Vassar College
Keith Schwarz, Stanford University
Michael Sipser, *Introduction to the Theory of Computation*