Welcome to CS420!
Intro to Theory of Computation
UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2022
Welcome to CS420!
Intro to Theory of Computation
UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2022
CS 420 Lecture Logistics

• I expect lecture to be interactive
  • Participation is a part of your grade
  • It’s the best way to learn!

• I may call on students randomly
  • It’s ok to be wrong – will not affect your grade
  • It’s the best way to learn!

• Please tell me your name before speaking
  • Sorry in advance if I get it wrong
  • It’s the best way for me to learn!
Welcome to CS420!

Intro to Theory of Computation

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2022
Computation Is ...  

- $1 + 1 = ??$  
  - $= 2$

- $11 + 11 = ??$  
  - $= 22$

- $9999999999 + 9999999999 = ??$  
  - $= 19999999998$

- $1 + 1 = ??$  
  - $= 10$  
    (binary)

... some base definitions and assumptions ("axioms"),  
e.g., "A Number is either 0, or the successor of another Number" ...

... and rules that use the definitions and axioms ("algorithm"),  
e.g., grade school arithmetic

Computation rules can be executed by hand, or by a machine / automaton

There are many possible definitions (models) of computation  

(hint)
Many Different Kinds of Computation

This class is about:
- formally defining models of computation,
- and studying their relation to each other!

(definitions + rules)
Models of Computation Hierarchy

... and also look at what's out here??

... and get to here ...

More powerful
More complex
Less restricted

We'll start here ...
Computation = Programs!

- A model of computation is represented by a class of machines (each box)
- Think of: a class of machines = a Programming Language!
- Think of: a single machine instance = a Program!
Welcome to CS420! Intro to Theory of Computation & Programs

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2022
Welcome to CS420! Intro to Theory of Computation

This class is about **formally** defining models of computation!

“formally” = mathematically (This is a math course!)
A (Mathematical) Theory is ...

Mathematical theory

From Wikipedia, the free encyclopedia

A mathematical theory is a mathematical model of a branch of mathematics that is based on a set of axioms. It can also simultaneously be a body of knowledge (e.g., based on known axioms and definitions), and so in this sense can refer to an area of mathematical research within the established framework.[1][2]

Explanatory depth is one of the most significant theoretical virtues in mathematics. For example, set theory has the ability to systematize and explain number theory and geometry/analysis. Despite the widely logical necessity (and self-evidence) of arithmetic truths such as 1<3, 2+2=4, 6<1=5, and so on, a theory that just postulates an infinite blizzard of such truths would be inadequate. Rather an adequate theory is one in which such truths are derived from explanatorily prior axioms, such as the Peano Axioms or set theoretic axioms, which lie at the foundation of ZFC axiomatic set theory.

The singular accomplishment of axiomatic set theory is its ability to give a foundation for the derivation of the entirety of classical mathematics from a handful of axioms. The reason set theory is so prized is because of its explanatory depth. So a mathematical theory which just postulates an infinity of arithmetic truths without explanatory depth would not be a serious competitor to Peano arithmetic or Zermelo-Fraenkel set theory.[3][4]

... a mathematical model, i.e., axioms and definitions, of some domain, e.g. computation ...

... that explains (predicts) some real-world phenomena ...

i.e., a theory must be useful in practice!
Example: Theory of Classical Mechanics

\[ F = \frac{d}{dt}(mv) \]

Second law of motion (axiom)

Do 1000 ton explosion here

Predicts landing exactly here

Drop off $1b of delicate stuff here
Why make predictions about computation?

RANSOMWARE ATTACK

Predict result without running a program?
Can we make predictions about computation?

It’s tricky: Trying to predict computation requires computation!
Can we make predictions about computation?

- The **Halting Lemma** says: NOPE

- And **Rice’s Theorem** says:
  - “all non-trivial, semantic properties of programs are undecidable”

- **Actually:**
  - it depends on the computation model!
Predicting What Some Programs Will Do ...

SLAM is a project for checking that software satisfies critical behavioral properties of the interfaces it uses and to aid software engineers in designing interfaces and software that ensure reliable and correct functioning. Static Driver Verifier is a tool in the Windows Driver Development Kit that uses the SLAM verification engine.

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we’re building tools that can do actual proof about the software and how it works in order to guarantee the reliability. Bill Gates, April 18, 2002. Keynote address at WinHec 2002.

Holy grail of CS!

Overview of Static Driver Verifier Research Platform

Static Driver Verifier (SDV) is a compile-time static verification tool, included in the Windows Driver Kit (WDK). The SDV Research Platform (SDVRP) is an extension to SDV that allows you to adapt SDV to:

- Support additional frameworks (or APIs) and write custom SLIC rules for this framework.
- Experiment with the model checking step.
Knowing What Computers Can’t Do is Still Useful!

In Cryptography:

• **Perfect secrecy** is impossible in practice

• But with **slightly imperfect secrecy** (i.e., a computationally bounded adversary) we get:

  ![Cryptography Diagram](image)

• But there are still problems, even with strong mathematical foundations:
  • with users, implementors **who don’t understand theory of computation**
Which Computation Model to Choose?

- Turing Machines
  - Which programs need more powerful computation ... ?
  - More powerful
  - More complex
  - Less restricted

- Linear bounded Automata
- Push-down Automata
- Finite State Automata

... and which need weaker computation?

e.g., logging?
Log4Shell

Log4Shell (CVE-2021-44228) was a zero-day vulnerability in Log4j, a popular Java logging framework, involving arbitrary code execution.[2][3] The vulnerability has existed unnoticed since 2013 and was privately disclosed to the Apache Software Foundation, of which Log4j is a project, by Chen Zhaojun of Alibaba Cloud’s security team on 24 November 2021, and was publicly disclosed on 9 December 2021.[1][4][5][6] Apache gave Log4Shell a CVSS severity rating of 10, the highest available score.[7] The exploit is simple to execute and is estimated to affect hundreds of millions of devices.[6][8]

The vulnerability takes advantage of Log4j’s allowing requests to arbitrary LDAP and JNDI servers.[2][9][10] allowing attackers to execute arbitrary Java code on a server or other computer, or leak sensitive information.[5] A list of its affected software projects has been published by the Apache Security Team.[11] Affected commercial services include Amazon Web Services,[12] Cloudflare, iCloud,[13] Minecraft: Java Edition,[14] Steam, Tencent QQ and many others.[9][15][16] According to Wiz and EY, the vulnerability affected 93% of enterprise cloud environments.[17]
The Computing Power of Fonts?

IN THE WILD —

Windows code-execution zero day is under active exploit, Microsoft warns

There's no patch available now. Here's what to do until Microsoft issues one.

DAN GOODIN - 3/23/2020, 3:40 PM

The font-parsing remote code-execution vulnerability is being used in “limited targeted attacks,” against Windows 7 systems, the software maker said in an advisory published on Monday morning. The security flaw exists in the Adobe Type Manager Library, a Windows DLL file that a wide variety of apps use to manage and render fonts available from Adobe Systems. The vulnerability consists of two code-execution flaws that can be triggered by the improper handling of maliciously crafted master fonts in the Adobe Type 1 Postscript format. Attackers can exploit them by convincing a target to open a booby-trapped document or viewing it in the Windows preview pane.
Which Computation Model to Choose?

- Turing Machines
- Linear bounded Automata
- Push-down Automata
- Finite State Automata

- Which programs need more powerful computation ... ?
  - More powerful
  - More complex
  - Less restricted

- e.g., do loggers and fonts need maximum computational power?
- ... and which need weaker computation?
A (Mathematical) Theory Is ...

... a mathematical model, i.e., axioms and definitions, of some domain, e.g. computers ...

... that explains (predicts) some real-world phenomena ...

... and can derive additional results (lemmas & theorems) ...
How Mathematics Works

Mathematician (or student)

Actually, it's not always so easy to create the next level ... Preciseness is important

Proofs = Figuring out how to (precisely) stack the pieces together

More Theorems
More Axioms
More Definitions
Theorem
Theorem
Axioms
Definitions
How CS 420 Works

Semester Start

CS 420

Graph Theory

Set Theory

Boolean Logic

Mathematical Logic

Prerequisite (CS 220) (see hw0)

Semester End

0

Thm

2

4

More CS420 Definitions, Axioms, & Theorems

CS420 Theorems

CS420 Definitions & Axioms

(What you will learn this semester)
A Word of Advice

Important:
Do not fall behind in this course

To prove a (new) theorem...
... need to know all axioms, definitions, and (previous) theorems below it
Another Word of Advice

“Blocks” from outside the course won’t work in the proof

Remember: Preciseness is important (Proofs must connect facts from this course exactly)

... can be used to prove (new) theorems in this course

Only axioms, definitions, and theorems from this course...
How to Do Well in this Course

- Learn the “blocks”
  - i.e., axioms, definitions, and theorems

- To solve a problem (prove a new theorem) ...
  - ... think about how to (precisely) combine and use existing “blocks”

- Don’t Fall Behind!
  - Start HW Early (HW 0 due Sunday 11:59pm EST)

- Participate and Engage
  - Lecture
  - Office Hours
  - Message Boards (piazza)
Textbooks

• Sipser. *Intro to Theory of Computation*, 3rd ed.

• Hopcroft, Motwani, Ullman. *Intro to Automata Theory, Languages, and Computation*, 3rd ed.

- Recommended but not required,
  - slides and lecture should be self-contained,
- Readings to accompany lectures will be posted

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs420/f22
Grading

• **HW**: 80%
  • Weekly: Out Monday, In Sunday
  • Approx. 12 assignments
  • Lowest grade dropped

• **Quizzes**: 5%
  • End of every lecture
  • To help everyone keep up

• **Participation**: 15%
  • Lecture, office hours, piazza

• **No exams**

• **A range**: 90-100
• **B range**: 80-90
• **C range**: 70-80
• **D range**: 60-70
• **F**: < 60

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs420/f22
Late HW

• Is bad ... try not to do it please
  • Grades get delayed
  • Can’t discuss solutions
  • You fall behind!

• Late Policy: 3 late days to use during the semester
HW Collaboration Policy

Allowed
• Discussing HW with classmates
• Using other resources, e.g., youtube, other books, etc.
• Writing up answers on your own, from scratch, in your own words

Not Allowed
• Submitting someone else’s answer
• It’s still someone else’s answer if:
  • variables are changed,
  • words are omitted,
  • or sentences rearranged ...
• Using sites like Chegg, CourseHero, Bartleby, Study, etc.
• Don’t use “blocks” from outside this course
Honesty Policy

• 1\textsuperscript{st} offense: zero on problem
• 2\textsuperscript{nd} offense: zero on hw, reported to school
• 3\textsuperscript{rd} offense+: F for course

Regret policy
• If you \textit{self-report} an honesty violation, you’ll only receive a zero on the problem and the issue will be immediately resolved \textit{(don’t abuse this please)}. 
All Up to Date Course Info

Survey, Schedule, Office Hours, HWs, ...

See course website:

https://www.cs.umb.edu/~stchang/cs420/f22/
Check-In Quiz 9/6
(see gradescope)