Welcome to CS420!

Intro to Theory of Computation

UMass Boston Computer Science

Instructor: Stephen Chang

Wednesday, January 24, 2024
Welcome to CS420!
Intro to Theory of Computation
UMass Boston Computer Science
Instructor: Stephen Chang
Wednesday, January 24, 2024

Analogy:
Computation Model
(system of definitions and rules) ↔
Programming Language
Welcome to CS420!

Intro to Theory of Computation

UMass Boston Computer Science
Instructor: Stephen Chang
Wednesday, January 24, 2024

“Theory” = math
(This is a math course!)

(But programming is math too!)

A precisely defined
Computation Model
(system of definitions and rules)
⇔
Programming Language
Programming = Mathematical logic!

• “logic is the foundation of all computer programming”
  • https://www.technokids.com/blog/programming/its-easy-to-improve-logical-thinking-with-programming/

• “logic is the fundamental key to becoming a good developer”
  • https://www.geeksforgeeks.org/i-cant-use-logic-in-programming-what-should-i-do/

• “Analytical skill and logical reasoning are prerequisites of programming because coding is effectively logical problem solving at its core”
  • https://levelup.gitconnected.com/the-secret-weapon-of-great-software-engineers-22d57f427937
In CS 420 this semester, we will ...

1. *Formally define* and *study models of computation*
   - models will be as *simple* as possible (to make them easier to study)

2. *Compare & contrast* models of computation
   - which “programs” are *included / excluded* by a model
   - *Equality* or *overlap* between models?

3. *Prove* things about the models
You already do “Proof” when Programming

```python
def f(x):
    if (x > 0) | (x < 0) | (x == 0):
        return x + 1
    else:
        return 1 / 0
```

Can this function ever throw ZeroDivisionError?

How did you figure out the answer?  You did a proof!

(Let’s write it out formally)
How Mathematics (Proofs) Work

Adding next level is hard ...

**Preciseness is important**
(just like in programming)

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

Mathematician
(or student)

More Theorems
More Axioms
More Definitions

Theorem
Theorem
Axioms
Definitions
The “Modus Ponens” Inference Rule

(Precisely Fitting Blocks Together)

Premises (if we can show these statements are true)
• If \( P \) then \( Q \)
• \( P \) is TRUE

Conclusion (then we can say that this is also true)
• \( Q \) must also be TRUE
Deductive Proof Example

Prove: \( \text{fn f never throws ZeroDivisionError} \)

Proof: Prior steps are already-proved, can be used to prove later steps!

### Statements

1. If running “test expr” is True, then “first branch” runs
2. If running “test expr” is False, then “second branch” runs
3. running “test expr” is (always) True
4. “first branch” (always) runs

### Justifications

1. Rules of Python
2. Rules of Python
3. Definition of “numbers”
4. By steps 1, 3, and modus ponens

### Code Snippet

```python
def f(x):
    """test expr"
    if (x > 0) | (x < 0) | (x == 0):
        return x + 1 "first branch"
    else:
        return 1 / 0 "second branch"
```

### Statements / Justifications Table

- **Modus Ponens**
  - If we can prove these:
    - \( \text{if P then Q} \)
    - \( P \)
  - Then we’ve proved:
    - \( Q \)

7. \( \text{fn f never throws ZeroDivisionError} \)
Deductive Proof Example

Prove: fn f never throws ZeroDivisionError

Proof:

Statements
1. If running “test expr” is True, then “first branch” runs
2. If running “test expr” is False, then “second branch” runs
3. running “test expr” is (always) True
4. “first branch” (always) runs
5. “second branch” never runs
6. fn f never runs 1 / 0
7. fn f never throws ZeroDivisionError

Justifications
1. Rules of Python
2. Rules of Python
3. Definition of “numbers”
4. By steps 1, 3, and modus ponens
5. By steps 1, 2, and ???
6. By step 5
7. By step 6 and ???
What else can we prove about programs?

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: 

• And **Rice’s Theorem** says:
  
  • “all non-trivial, semantic properties of programs are undecidable”
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:
Can we make predictions about computation?

- The **Halting Lemma** says:

- 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

Predicting things about programs ... is the Holy grail of CS!

Overview of Static Driver Verifier Research Platform

Static Driver Verifier Research Platform README

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.
In this class, we will prove things about our simple computational models.
How CS 420 Works

Semester Start

Prerequisite (CS 220) (see hw0)

Graph Theory

Set Theory

Boolean Logic

Mathematical Logic

Semester End

Thm

CS 420

0

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 in proofs (just like in programming) is critical
(Proofs must connect facts from this course exactly)

HW problems are graded on precise steps in the proof, not on the final theorem itself!

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

Only axioms, definitions, and theorems from this course...
Textbooks

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

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

**Strongly Recommended** (but not required)
- Slides (posted) and lecture should be self-contained,
- BUT, Students who do well read the book

All course info available on web site: https://www.cs.umb.edu/~stchang/cs420/s24
How to Do Well in this Course

• **Learn** the “building blocks”
  • i.e., axioms, definitions, and theorems

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

• HW problems graded on **steps** to the **answer** (not final theorem)

• **Don’t** Fall Behind!
  • Start HW Early (HW 0 due Monday 1/22 12pm EST noon)

• **Participate** and Engage
  • Lecture
  • Office Hours
  • Message Boards (piazza)
Grading

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

• **Participation:** 20%
  • Lecture participation, in-class work, 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/s24
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 (but must cite)
- Using other resources to learn, e.g., youtube, other textbooks, ...
- Writing up answers on your own, from scratch, in your own words

**Not Allowed**
- Submitting someone else’s answer
- Submitting someone else’s answer with:
  - variables changed,
  - thesaurus words,
  - or sentences rearranged ...
- Using sites like Chegg, CourseHero, Bartleby, Study, ChatGPT, etc.
- Using theorems or definitions not from 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 self-report an honesty violation, you’ll only receive a zero on the problem and we move on.
All Up to Date Course Info

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

See course website:

https://www.cs.umb.edu/~stchang/cs420/s24/
hw0 (pre-req quiz)
(see gradescope)