Welcome to CS622!

Theory of Formal Languages

UMass Boston Computer Science

Instructor: Stephen Chang

Spring 2024

Today’s Theme: What’s CS 622 about?
Welcome to CS622!

Theory of Formal Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Spring 2024

What’s this?
Interlude: CS 622 Lecture Logistics

• I expect: lecture to be **interactive**
  • Participation is a part of your grade
  • Also, it’s the best way to **learn**!

• I may: **call** on students randomly
  • It’s ok to be wrong in class! – will not affect your grade
  • Also, it’s the best way to **learn**!

• Please: **tell me your name** before speaking
  • Sorry in advance if I get it wrong
  • Also, it’s the best way for me to **learn**!
What’s a “language”?
What's a “language”?
A Language Represents ... Computation

A programming language allows **expressing** and reasoning about **computations**

It’s a **model of computation**

```
def f(x):
    if x > 0:
        return x +
    else:
        return x - 1
```

**different??**

If they are different: how can we know?

**Or same??**

If they are the same: what is a (simple) model for all of them
In CS 622 this semester, we will ...

1. **Define** and **study** models of computation
   - models will be *as simple as possible* (to make them easier to study)
Models of Computation
In CS 622 this semester, we will ...

1. **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** by a model
   - which “programs” are **excluded** by a model
   - **overlap** between models?
Models of Computation

Q: Are there computational models ... other than Turing Machines?
Q: Are there computational models ... “weaker” than Turing Machines?
Q: What does “weaker” or “more powerful” even mean?!

A: Yes!
Models of Computation Hierarchy

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

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

We’ll start here...

... and get to here...

More powerful
More complex
Less restricted
But remember ... Computation = Programs!

Helpful analogy for this course:
- a class of machines (each rectangle above) ~ a Programming Language!
- a single machine (one thing in a rectangle) ~ a Program!
Welcome to CS622!
Theory of Formal Languages
UMass Boston Computer Science
Instructor: Stephen Chang
Spring 2024
Welcome to CS622!

Theory of Formal Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Spring 2024

“What’s this?”

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

(But programming is math too!)
Programming Is (What) Math?

Math(ematical) logic!

def f(x):
    if x > 0:
        return x + 1
    else:
        return x - 1

print( f(10) )

How did you figure out the answer?

(But programming is math too!)
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
Programming = Mathematical logic!

**Programming Concepts**
- Functions
- Variables
- If-then
- Recursion
- Strings
- Sets (and other data structures)

**Mathematical Logic Concepts**
- Functions
- Variables
- If-then (implication)
- Recursion
- Strings
- Sets (and other groupings of data)
In CS 622 this semester, we will ...

1. **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* by a model
   - which “programs” are *excluded* by a model
   - *overlap* between models?

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

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

print( f(10) )

11

Can this function ever throw ZeroDivisionError?

How did you figure out the answer?

You did a proof!

(Let’s write it out formally)
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. computers ...

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

... and can derive (prove) additional results (theorems) ...

39
How Mathematics Works

Mathematician (or student)

Actually, it’s not always so easy to create the next level...
Preciseness is important (just like in programming)

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

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
Kinds of Mathematical Proof

Deductive Proof

• *Start with:* known facts and statements

• *Use:* logical *inference rules* (like modus ponens) to prove new facts and statements