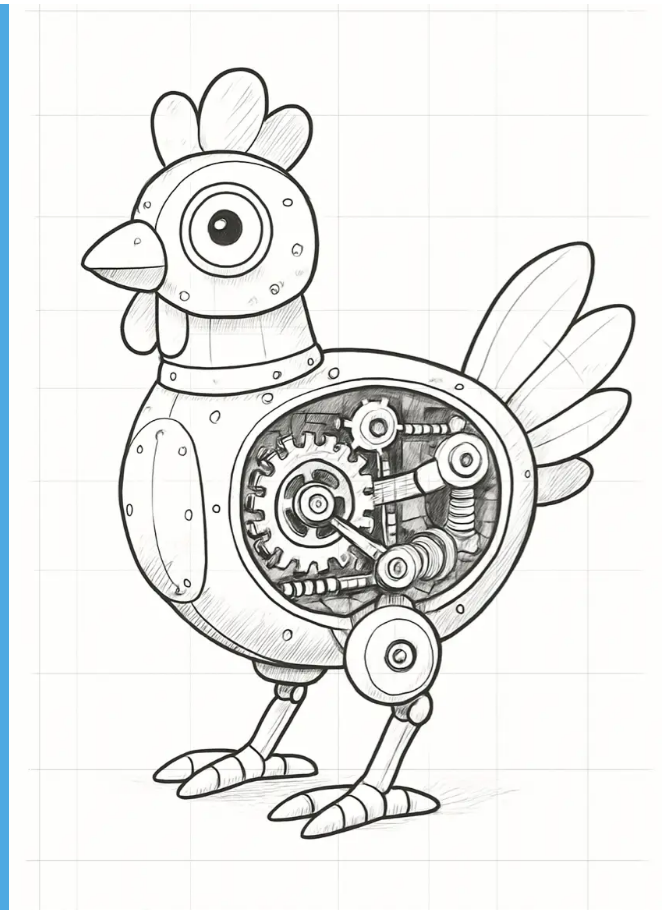
CS 839 Systems Verification

Lecture 2: Rocq



Create an image of a cute mechanical chicken. Make it like an engineering drawing, showing the inner workings. Clean, simple. Digital art.

Learning outcomes

1. Connect "informal" to "formal" proofs
2. Decode the Rocq interface
3. Use effective learning strategies

How do we know something is true?

Proofs

How do we know we have a proof?

A proof is a sequence of arguments

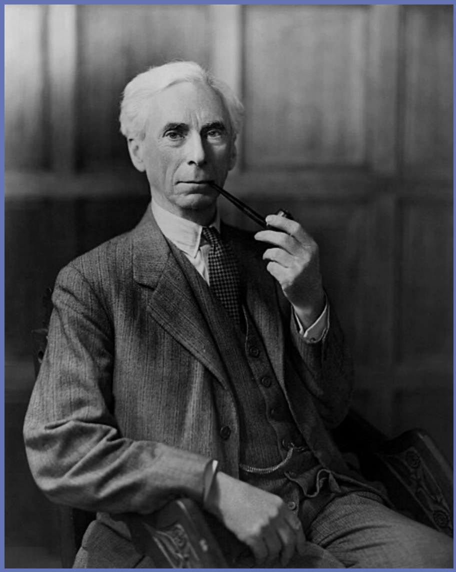
What's a "valid" argument?

Need some system of *logic*

History of (formal) logic

Gottlob Frege (1879)

Bertrand Russell (1910)



Skipping a bunch of earlier history, going back to Aristotle, Leibniz, Boole.
Frege attempted to create a language for logic. Much more formal and general (e.g., first-order and not just predicate logic) than Aristotle.
However, Bertrand Russell pointed out a fatal flaw in Frege's work.

Russell's paradox

1. $\{x \mid P(x)\}$
2. Let $X = \{Y \mid Y \notin Y\}$
3. Ask $X \in X$?
4. If $X \in X$, then by definition of X, $X \notin X$. If $X \notin X$, then by definition of X, $X \in X$.

Russell attempted to fix Frege's logic by developing a *theory of types*.

More logical foundations

- 1972 Girard introduces $F\omega$
- 1972 Per Martin-Löf introduces intuitionistic type theory
- 1970s: de Bruijn runs Automath project for mechanically checked mathematics

Rocq

- 1984: Thiery Coquand and Gérard Huet implement Coq v1. Combines $F\omega$ (higher order, polymorphism) with intuitionistic, constructive math.
- 1986: Christine Paulin-Mohring joins the team. v2 released.
- 1989: v4 adds inductive types.
- 2001: v7 adds a tactic language.
- 2004: Coq v8.
- 2025: Renamed to Rocq, now v9.

https://rocq-prover.org/doc/master/refman/history.html
Coq v1, implemented in CAML. Combined $F\omega$'s polymorphism and higher-order functions with intuitionistic foundations ("constructive logic").
Coq v2 added a proof of consistency, universes, type inference.
By v4 we had a "vernacular" and inductive types.
Only in v7 did we have a tactic language (Ltac).

Rocq as a system

Programming, proofs, and interactivity

Use `rocq_intro.v`

~10 MIN - functional programming, query, unit test theorem, different languages

Observe

What vernacular commands have we seen so far?

What tactics have we seen?

What syntax have we seen for writing terms? (slightly tricky)

Predict

What other vernacular commands do you think there are?

What tactics do you expect?

What unseen syntax do you expect for writing terms?

Type error exercises

Read the error message

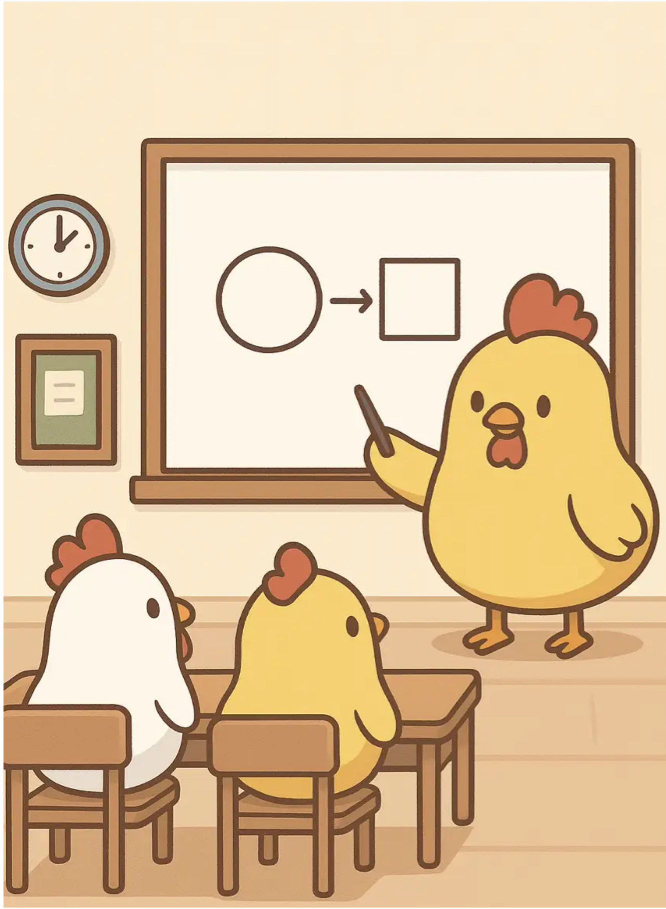
Explain the error message

Fix the error

Think on your own (2-3 min), then compare explanations and fixes with your pod.
8 MIN

5-minute break

Create an image of a cute chicken with red hair relaxing on an easy chair, with a closed MacBook on a side table. Digital art, clean and simple.



How learning works

Create a drawing of a classroom with a chicken explaining at a whiteboard to a class with 2 chickens. Make it cute, digital art, with a clean and bright style.
a bit of cognitive science (how people learn), educational psychology (understanding and enhancing student learning), and learning sciences (applying those principles to education)

How memory works

Activity: memory is the residue of thought

activity data

This activity was adapted from Daniel Willingham's article "What Will Improve a Student's Memory?" in American Educator 2008-2009, available at https://www.aft.org/sites/default/files/willingham_0.pdf

How memory works

1. Create a spreadsheet in Google Docs or locally
2. We will do 30 words
3. For each word, one of three tasks

Tasks

- "Spoken to the left"
- "A or U?"
- Rate for pleasantness, 1-7

You have to listen carefully because there are three tasks, and I'm going to mix them up. I'll tell you right before each word which task you should do for that word. Let's try a couple of each for practice; you don't need to write your answers for these.

15 MIN

Write down as many words as you can remember in a new column.

Memory is the residue of thought: conclusions

You remembered things you weren't trying to remember.

Thinking about meaning is more effective than other thinking.

How learning works

Remembering things

three principles from Willingham's work:
memory is the residue of thought
memories are lost due to missing/ambiguous cues (not due to disuse)
individual's self-assessment of knowledge is generally an over-estimate

How do we learn to program?

(1) orientation

(2) notional machine

(3) notation

(4) structures

(5) pragmatics

orientation - what programs are for, what we use them for
notional machine - how does a computer run a program?
notation - syntax/semantics of a particular PL
structures - schemata/plans that can be used to construct parts of programs
pragmatics - how to plan, develop, test, debug programs

How do we learn interactive theorem proving?

(1) orientation - what are we proving, and why?

(2) notional machine - how do we evaluate proofs?

(3) notation - understand theorem statements and tactics

(4) structures - proof strategies

(5) pragmatics - how to get information, identify next steps

How learning works

Principles

- *organizing* knowledge is crucial
- *component skills* are needed for mastery
- goal-directed practice + targeted *feedback* enhances learning
- to become self-directed, students must monitor and adjust their *approach* to learning

How students organize knowledge influences how they learn and apply what they know
Computer Science is unique in that we get a tremendous amount of computer-generated feedback. But you must learn to interpret and use it!