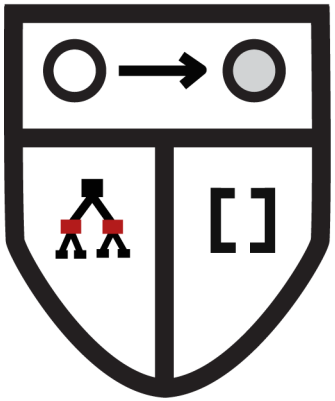


CS 3110

Functional Programming in Coq

Prof. Clarkson

Fall 2018

Today's music: Theme from *Downton Abbey* by John Lunn

Attendance question

A. Go Vote
B. Go Vote
C. Go Vote
D. Go Vote
E. Go Vote

Review

Previously in 3110:

- Functional programming
- Modular programming
- Data structures
- Interpreters

Next unit of course: formal methods

Today:

- Proof assistants
- Functional programming in Coq
- Proofs about simple programs

Approaches to validation [lec 11]

- Social
  - Code reviews
  - Extreme/Pair programming
- Methodological
  - Design patterns
  - Test-driven development
  - Version control
  - Bug tracking
- Technological
  - Static analysis ("lint" tools, FindBugs, …)
  - Fuzzers
- Mathematical
  - Sound type systems
  - "Formal" verification

Less formal: Techniques may miss problems in programs

All of these methods should be used!

Even the most formal can still have holes:

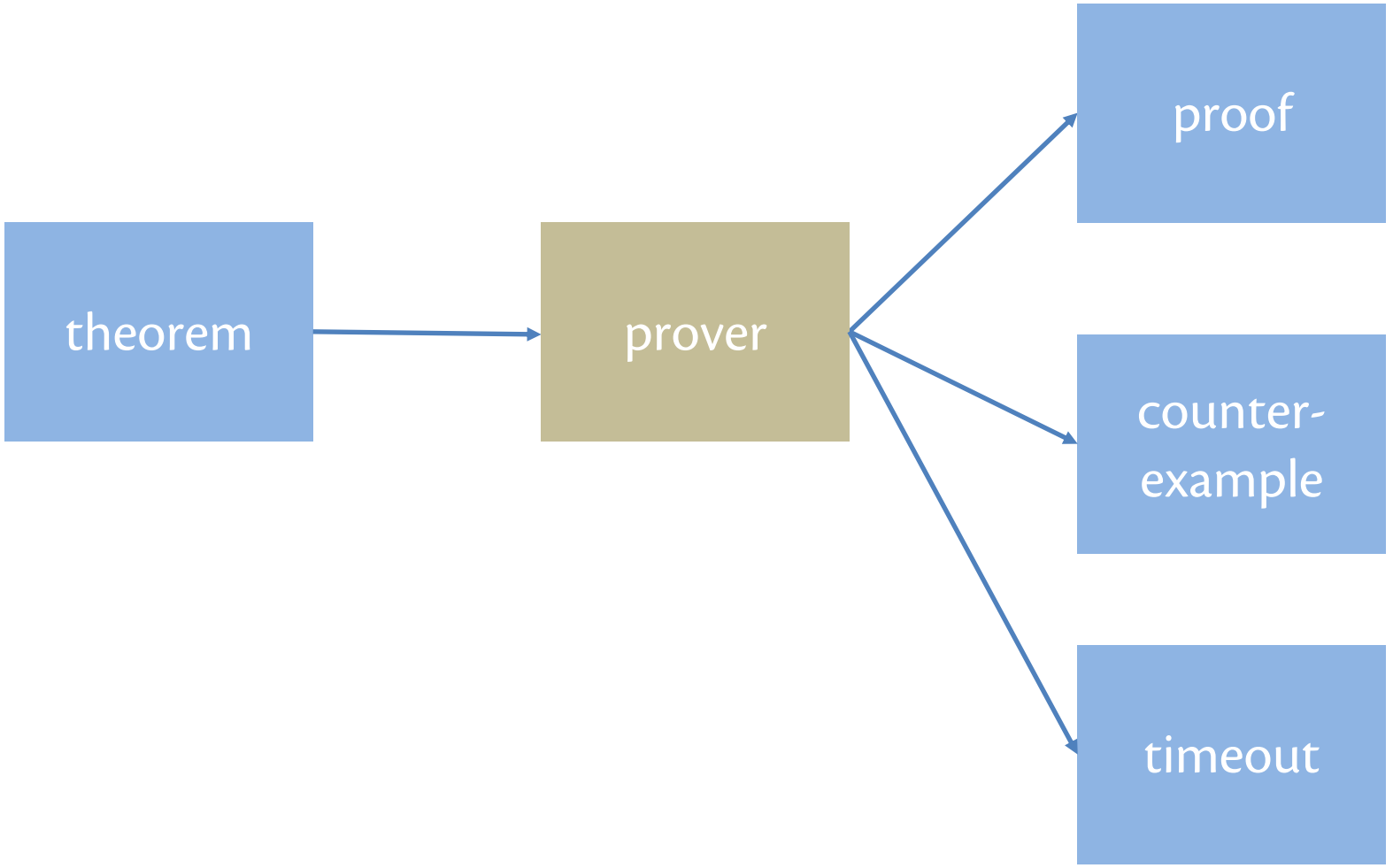
- did you prove the right thing?
- do your assumptions match reality?

More formal: eliminate *with certainty* as many problems as possible.

Verification

- In the 1970s, scaled to about tens of LOC
- Now, research projects scale to real software:
  - CompCert: verified C compiler
  - seL4: verified microkernel OS
  - Ynot: verified DBMS, web services
- In another 40 years?

Automated theorem provers

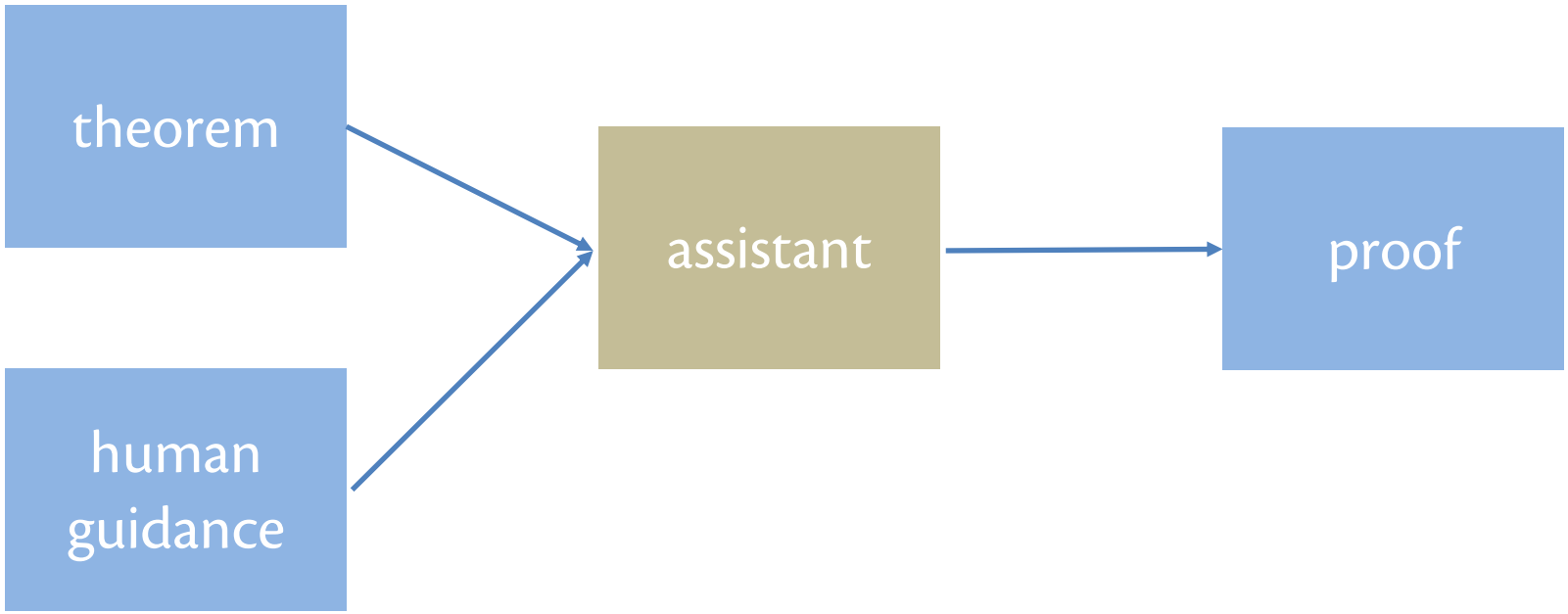


Automated theorem provers

- Z3: Microsoft started shipping with device driver developer's kit in Windows 7

- ACL2: used to verify AMD chip compliance with IEEE floating-point specification, as well as parts of the Java virtual machine

Proof assistant



Proof assistants

- NuPRL [Prof. Constable]: Formalization of mathematics, distributed protocols, security

- Coq: CompCert, Ynot [Dean Morrisett]

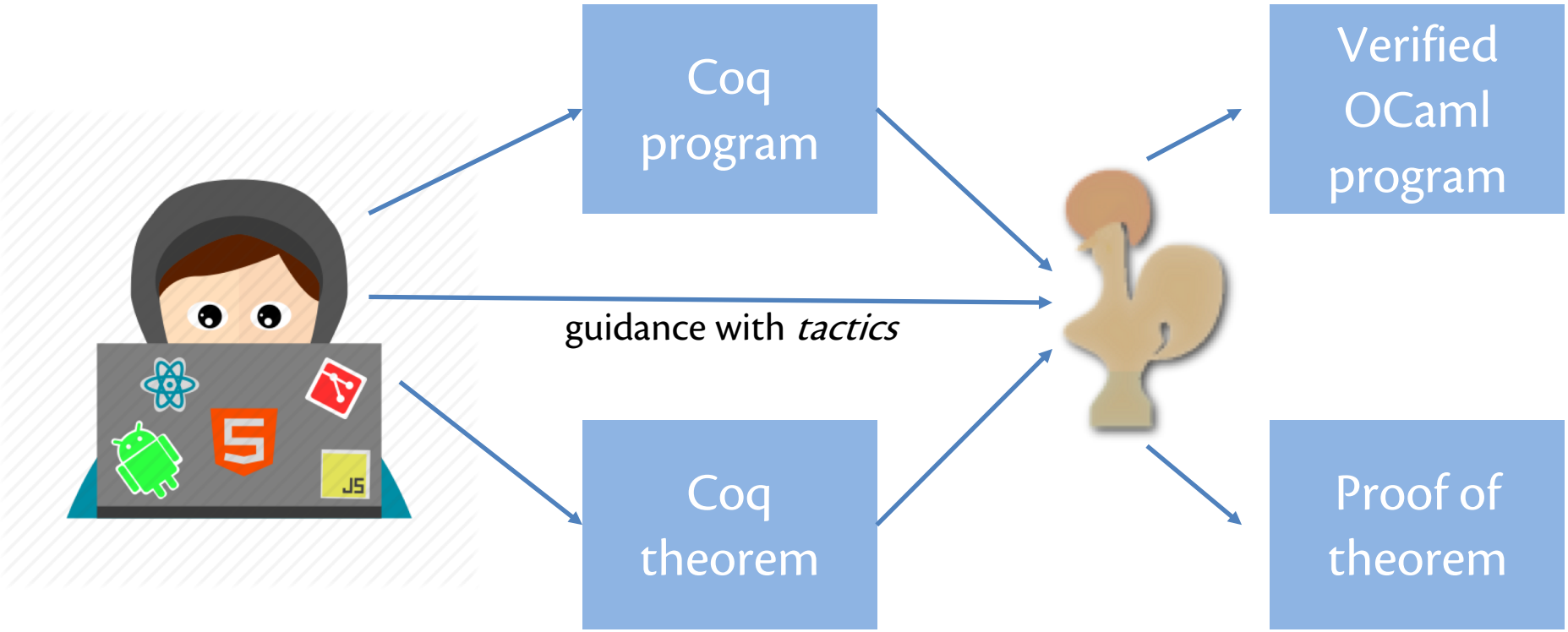
COQ

Coq

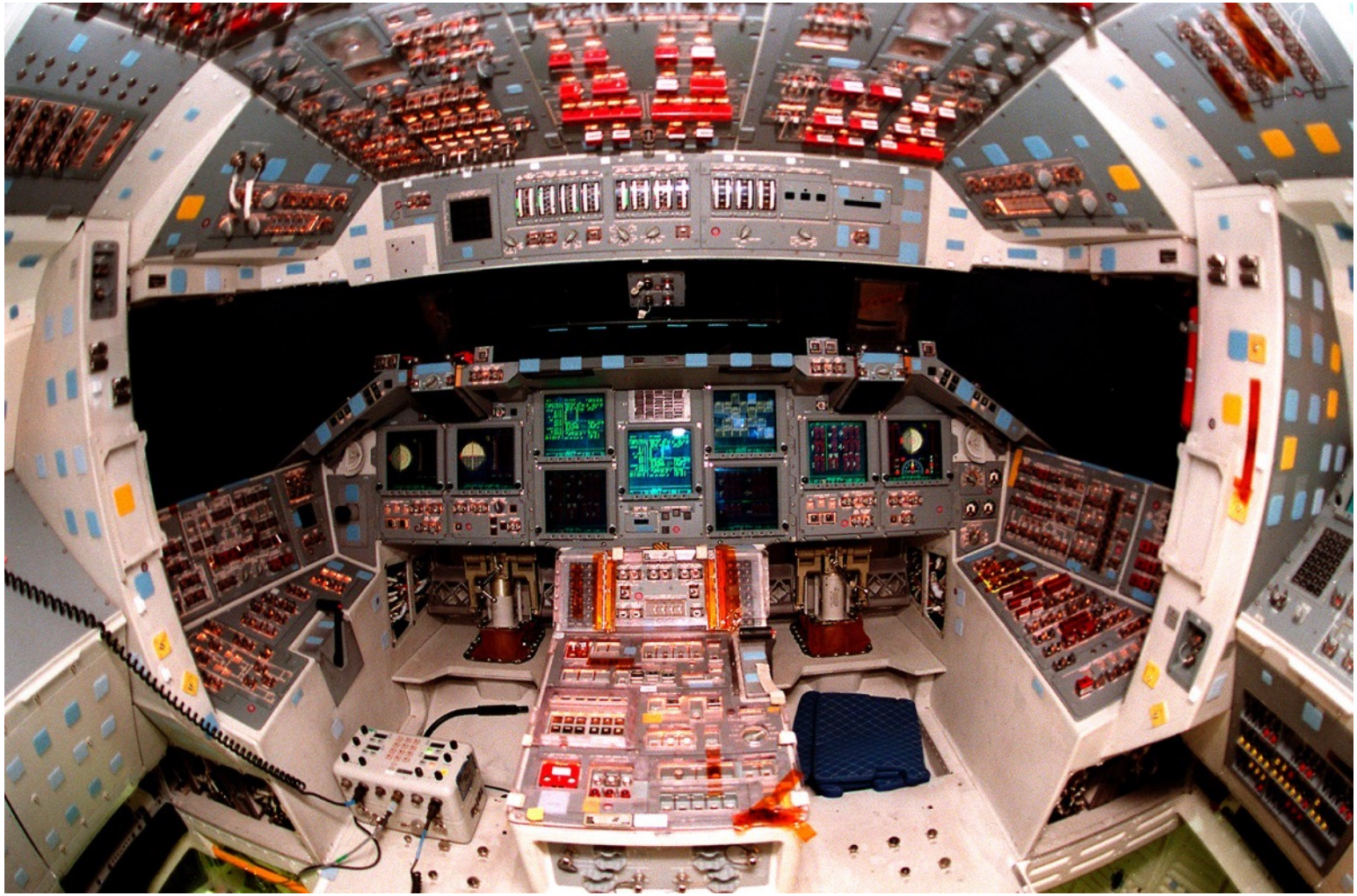
- **1984:** Coquand and Huet implement Coq based on *calculus of inductive constructions*
- **1992:** Coq ported to Caml
- Now implemented in OCaml

Thierry Coquand
1961 –

Coq for program verification



Coq's full system



Subset of Coq we'll use

Our goals

- Write basic functional programs in Coq
  - no side effects, mutability, I/O
- Prove simple theorems in Coq
  - CS 3110 programs: lists, options, trees
  - CS 2800 mathematics: induction, logic

- Non goal: full verification of large programs
- Rather:
  - help you understand what verification involves
  - expose you to the future of functional programming
  - solidify concepts about proof and induction by developing machine-checked proofs


CAUTION: HIGHLY ADDICTIVE

Definitions and Functions

Pairs, Lists, and Variants

FUNCTIONAL PROGRAMMING IN COQ

Demo

PROVING THEOREMS ABOUT PROGRAMS

Demo

Upcoming events

- N/A

This is formal.

THIS IS 3110