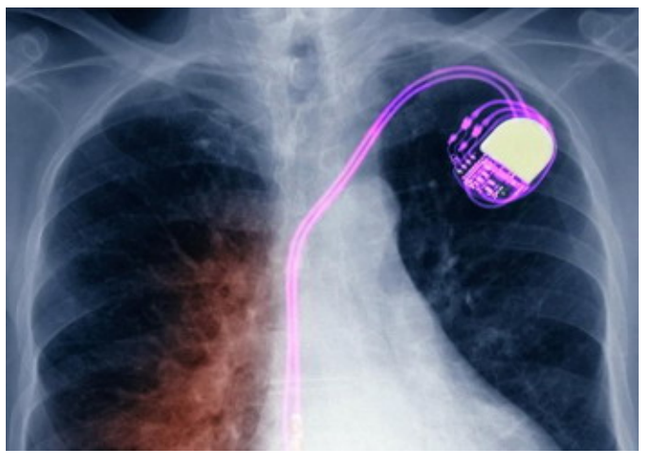
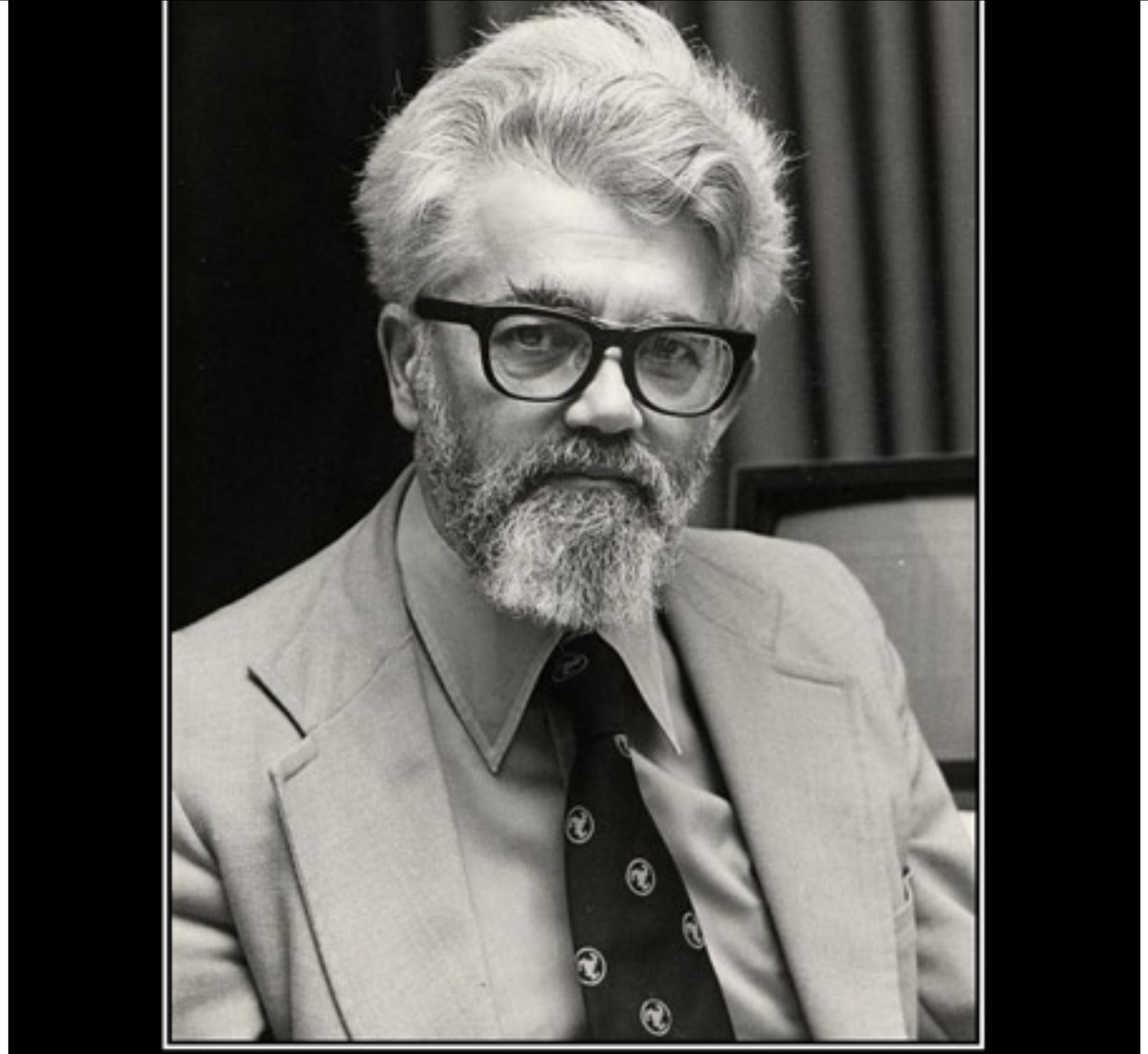
CSE 506
Proof Assistants

or: *reducing the"what is this I don't even" effect of Coq*

Why Proof Assistants?

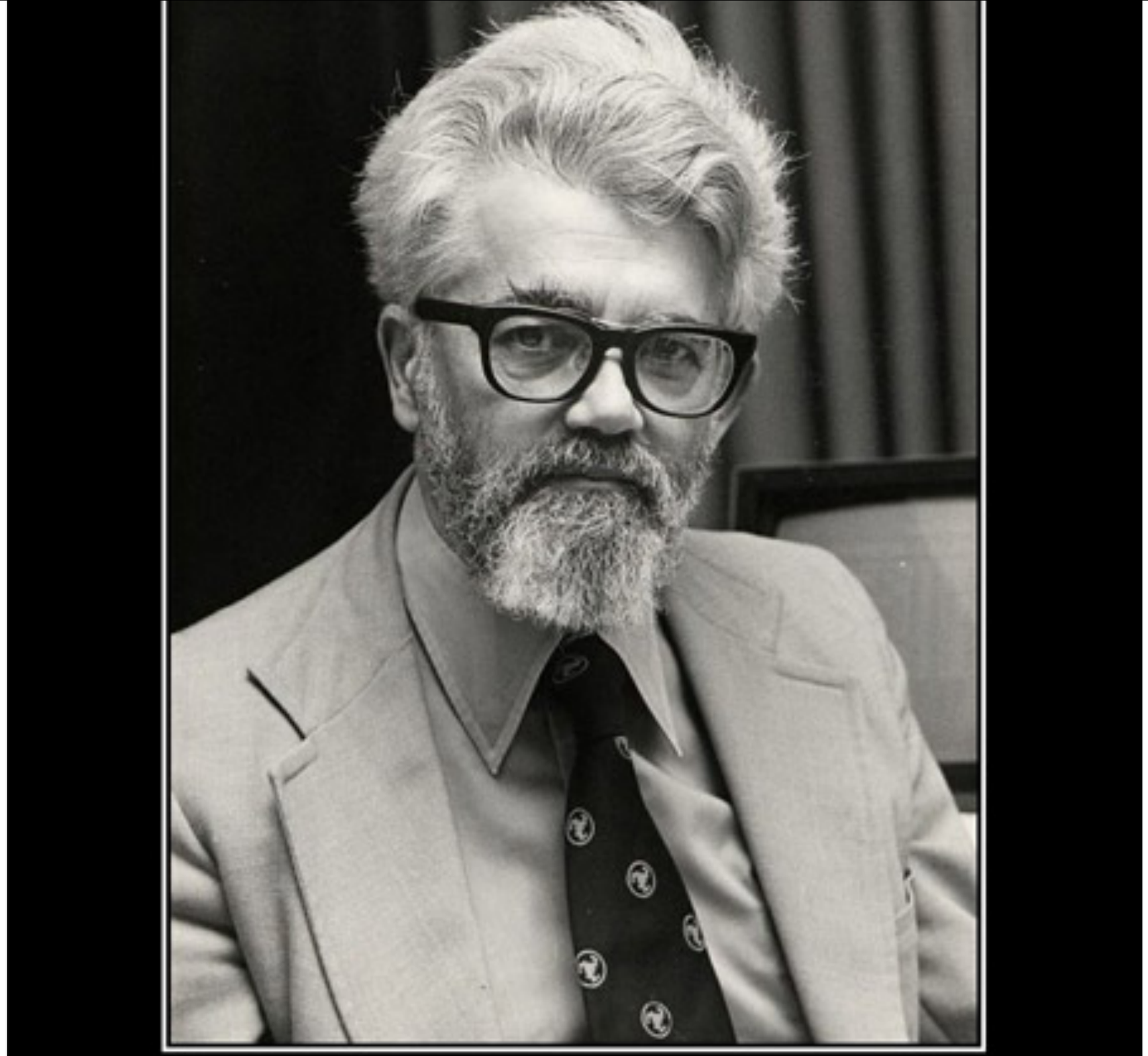
Why Proof Assistants?





PROGRAMMING

You're Doing It Completely Wrong.



PROGRAMMING

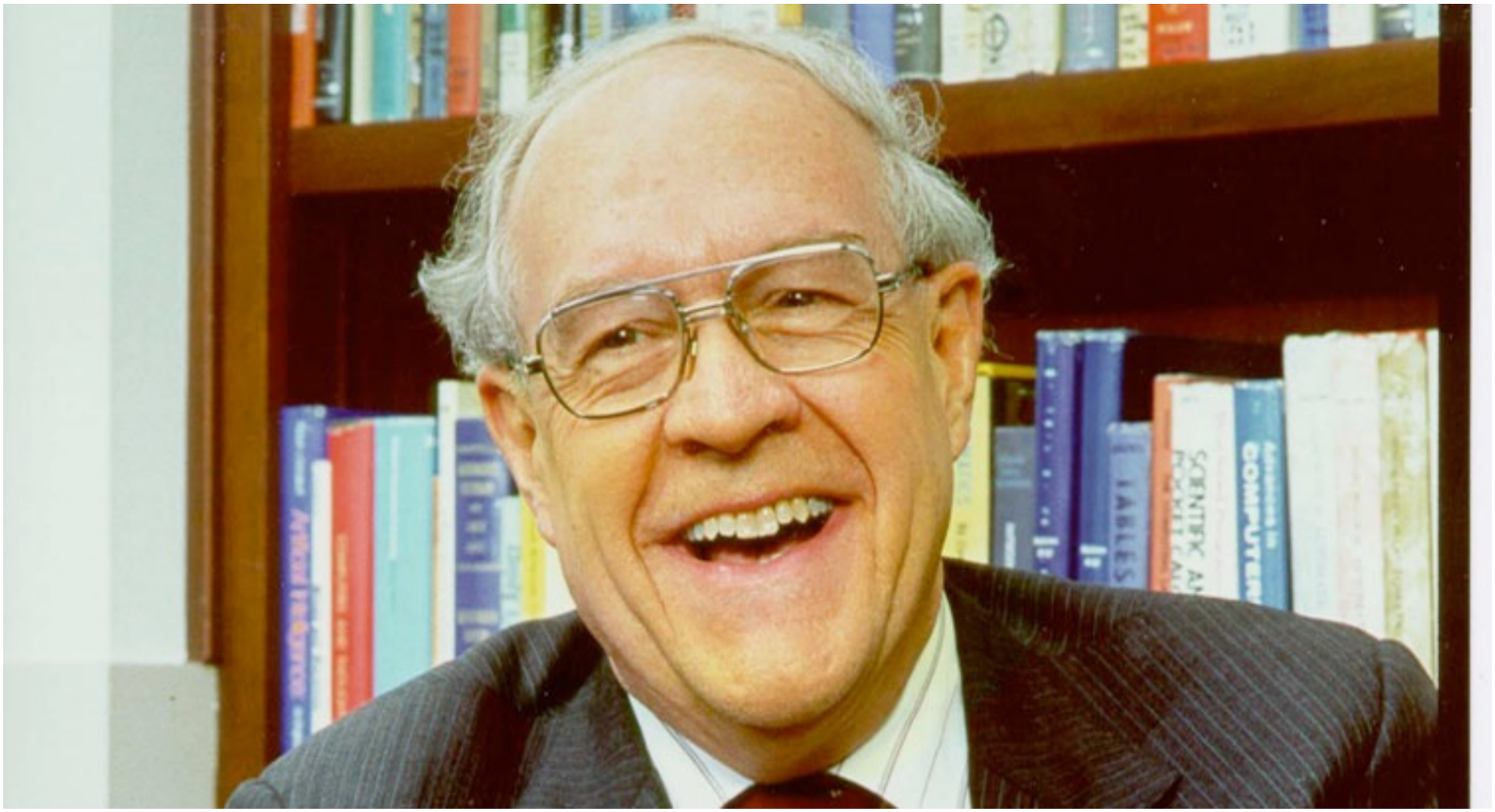
SOFTWARE

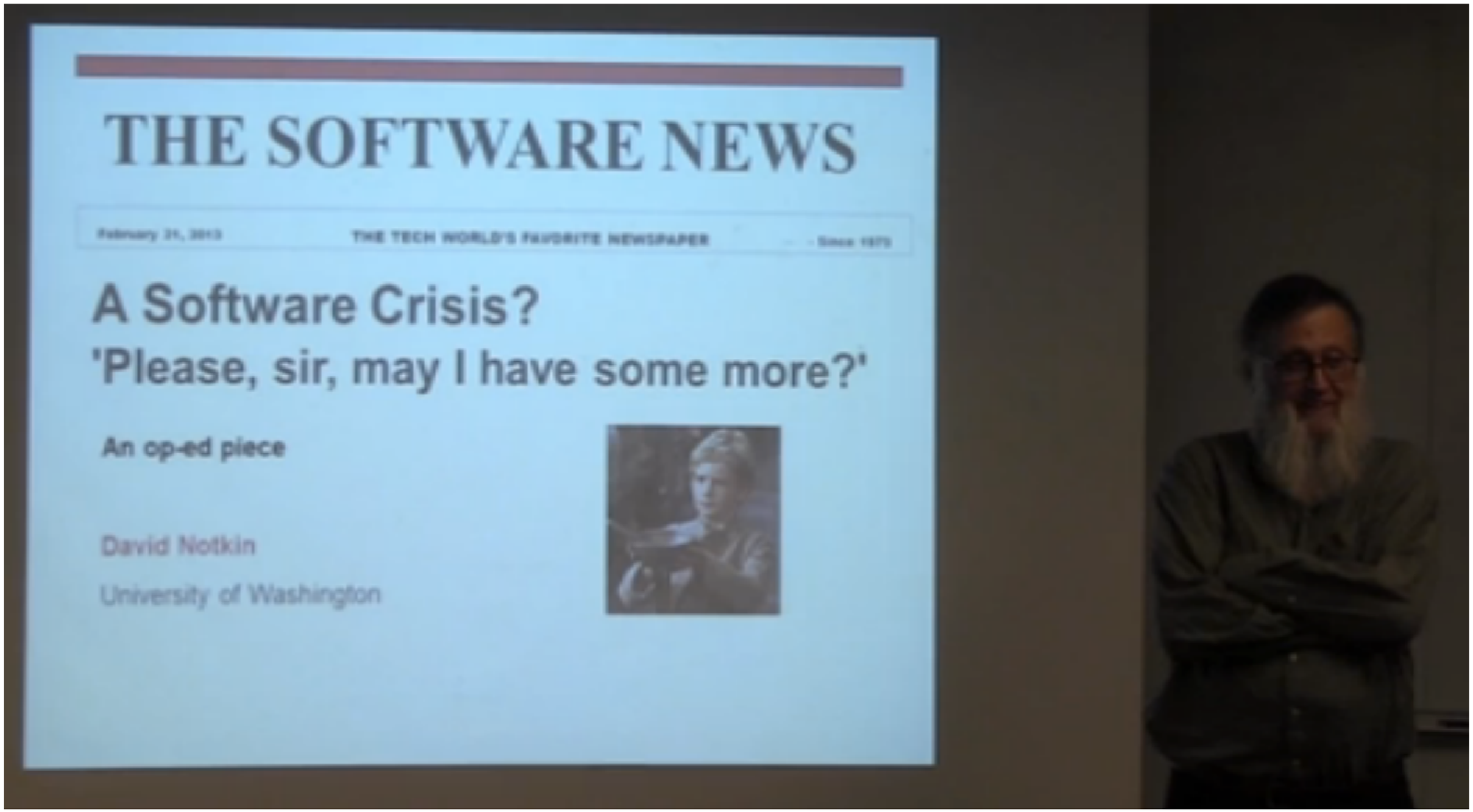
IS

GREAT!!!

The magic of myth and legend has been realized in our time. One simply types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.

Fred Brooks





http://cs.brown.edu/events/talks/notkin.html

SOFTWARE IS GREAT!!!

SOFTWARE IS GREAT!!!

... but still poorly understood!

https://www.destroyallsoftware.com/talks/wat

SOFTWARE IS GREAT!!!

... but still poorly understood!

https://www.destroyallsoftware.com/talks/wat

Why Proof Assistants?

- PL proofs numerous, big, subtle
- Who verifies the verifiers?
- Coq et al. help us avoid kidding ourselves
- Infrastructure bugs really are bad news
- Increasingly prevalent in PL research

Why Proof Assistants?

- PL proofs numerous, big, subtle
- Who verifies the verifiers?
- Coq et al. help us avoid kidding ourselves
- Infrastructure bugs really are bad news
- Increasingly prevalent in PL research

POPL '09	Submitted
Declare PA	42
Total	159

Why Proof Assistants?

- Numerous success stories:
    - CompCert, Vellvm, Bedrock, Ynot
    - RockSalt, Frenetic, seL4, Quark
    - Four Color Theorem, Homotopy Type Theory
- The closest we know how to get to bug free code

Why Proof Assistants?

- Can be extremely fun and enlightening

Building proof scripts is surprisingly addictive, in a videogame kind of way.

Xavier Leroy

... [rigorous proofs about programs] are an absolute scientific ideal, like purity of materials in chemistry or accuracy of measurement in mechanics. The value of purity and accuracy (just like correctness) are often not appreciated until after the scientist has built the tools that make them achievable.

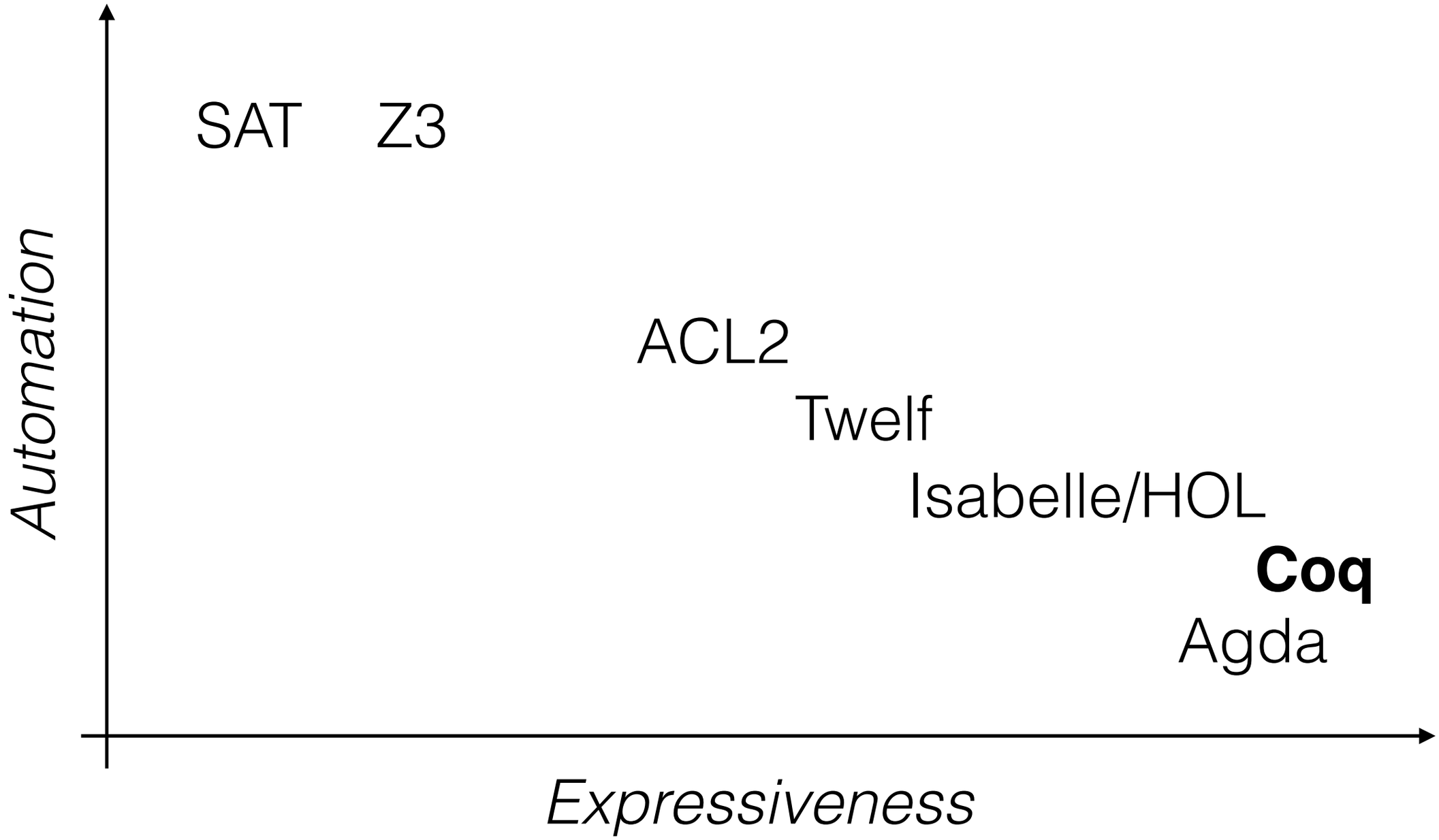
Sir C.A.R. Hoare

What Are Proof Assistants?

- Programs to help us construct formal proofs
- Large design space:
  - *underlying logic, automation, witnesses, executable, …*
- Typically expressive and interactive
- “I know it when I see it”

Theorem Prover Space

(woefully incomplete caricature)



Which Proof Assistant?

- Coq: The Proof Assistant for the Discerning Hacker
- Matured over decades at INRIA
- Architecture minimizes TCB
    - *small, simple proof checker*
    - *produce independently verifiable proofs*
    - *sugar and tactics aid proof construction*
- Popular, what I know best, good textbooks…

CPDT

- Traditional proving in Coq can be unpleasant
  - *ad hoc, imperative, brittle, opaque*
- Chlipala's style pushes automation to the extreme
- Assumes little background
- Gets to advanced material quickly
- Learning curve can be *steep*

Goals

- Develop working knowledge of Coq
- Verify something interesting (tied to your research)
- Practice teaching your peers
- Learn dependent types and some metatheory
- Have a good time

Attack Plan

- Projects:
  - *form teams of 3*
  - *work up a couple proposals by Monday and email me*
  - *automate, optimization, DSL compiler, improve tools, …*
  - *remember 10 weeks is short!*
- Presentations:
  - *lead class through a chapter of CPDT*
  - *provide guided tour of significant verification*
  - *who's next? fill up schedule by Friday*

Chit Chat

- Course Webpage
  - http://courses.cs.washington.edu/courses/cse506/14wi/
- Mailing List (IMPORTANT)
  - http://mailman1.u.washington.edu/mailman/listinfo/cse506a_wi14
- IRC (LESS IMPORTANT)
  - #uwplse on freenode (also #coq)
- Office Hours (546) whenever door is open
  - or by appointment

Let’s Dive In!

Setup = Coq 8.4, Emacs, ProofGeneral, CPDT