Qed.
“We believe that, because no social process [comparable to proof in mathematics] can take place among program verifiers, program verification is bound to fail.”

[De Millo, Lipton, Perlis 1979]
Approaches to validation [Lec 1]

- **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.
False dichotomy?

“[De Millo et al.] framed the debate as one between a reasonable engineering approach that completely ignores verification and a completely unrealistic view of verification advocated only by its most naïve proponents.”

[Leslie Lamport]

http://lamport.azurewebsites.net/pubs/pubs.html, point 38
False dichotomy?

“The social nature of proof and program development is uncontroversial and ineluctable, but formal verification is not antithetical to it.”

[Asperti, Geuvers, Natarajan 2009]
40 years after DLP

- **CompCert**: verified C compiler
- **seL4**: verified microkernel OS
- **Ynot**: verified DBMS, web services
- **Four color theorem**
- **Project Everest**: verified HTTPS stack [in progress]
- **Etc.**

In another 40 years?
Some issues raised by the debate

- What is proof?
- What establishes meaningfulness?
- What is the role of insight? Simplicity? Replication?
- What benefits and harms could result from pursuit of verification?
How has this class changed your concept of proof?
WRAP UP
Thanks

• Thanks to SF community for materials
• Thanks to TAs: Samantha Deng, Tjaden Hess, Devin Lehmacher, Devin Smedira, Weiyu Wang
• Thank YOU for taking a chance on a new course!
What next?

• PL theory:
  – CS 4110 Programming Languages and Logics
  – CS 6110 Advanced Programming Languages

• Logic:
  – CS 4860 Applied Logic
  – CS 6764 Reasoning about Knowledge
  – CS 6860 Logics of Programs
  – CS 6861 Intro to Kleene Algebra

• Type theory:
  – CS 6180 Intro to Constructive Type Theory

• Coq:
  – CS 6115 Certified Software Systems
What next?

• Complete 4160 course eval
• Apply to 4160 course staff for next spring
• Email me your resume if interested in 4999 project expanding SF in fall