Lecture 1

Pete Manolios
Northeastern

Objectives

- Introductions
- What's this class about?
  - What are the laws of computation?
  - This is a fundamental question in CS
  - The answer allows to reason about and gain *predictive power* over software and computational systems

Introductions

- Come to the front of the class
  - Background
    - Where are you from?
    - Your major
  - Why are you interested in CS
  - One interesting thing about you

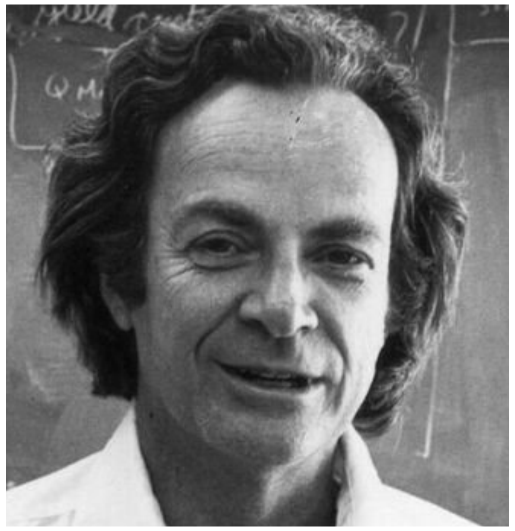
Science

- We are called computer science. We are a young field.
- What about the king of sciences, physics.
- What do physicists do?
  - They study the laws of the universe.

How Physics Works

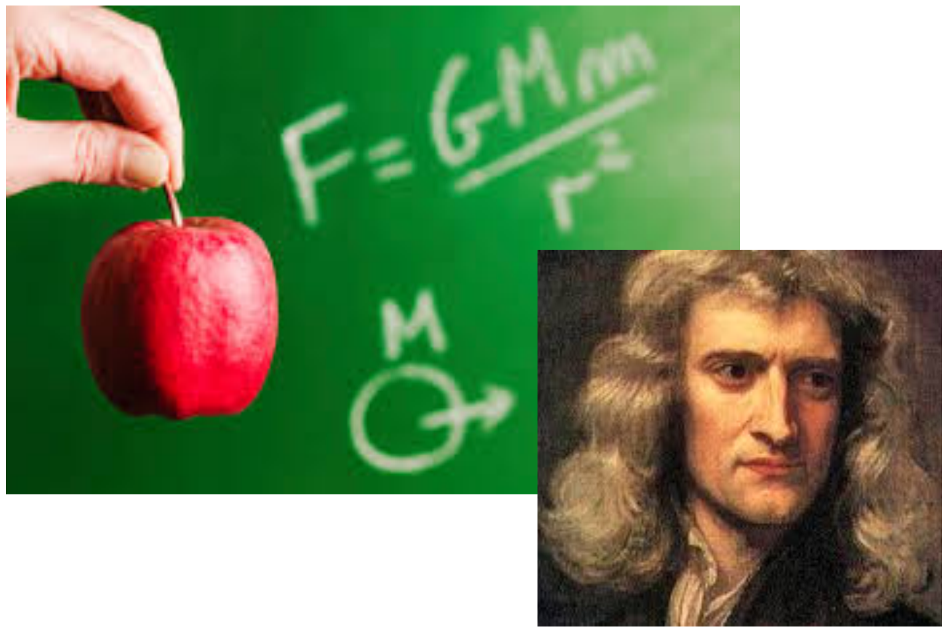
Feynman: *First you guess. Don't laugh, this is the most important step. Then you compute the consequences. Compare the consequences to experience. If it disagrees with experience, the guess is wrong. In that simple statement is the key to science. It doesn't matter how beautiful your guess is or how smart you are or what your name is. If it disagrees with experience, it's wrong. That's all there is to it.*

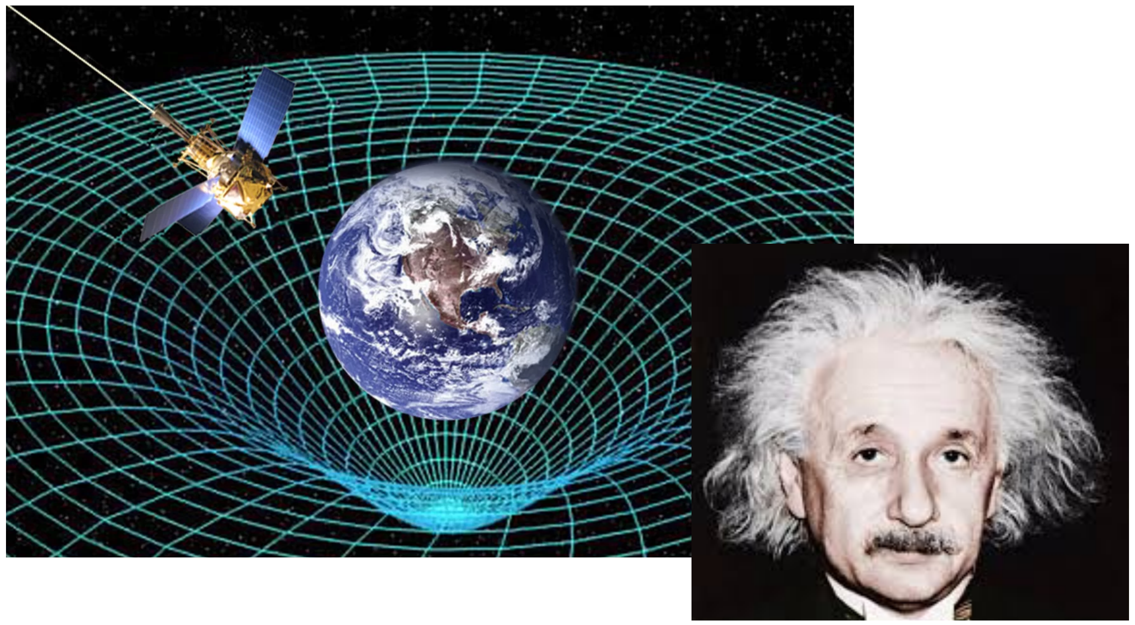
- Guess how the universe works (using math!)
- Conduct experiments to validate or disprove theories



When are we Done?

- When are we done? When do we have the final truth?
  - Never: Newtonian Mechanics displaced by Einstein displaced by …





Computer Science

- What do computer scientists do?
  - They study the laws of computation.
- In what ways are physics and CS similar?
  - The computers we build exist in the physical universe
  - They are subject to the laws of physics
  - So CS is just a branch of physics?

CS/Physics Differences

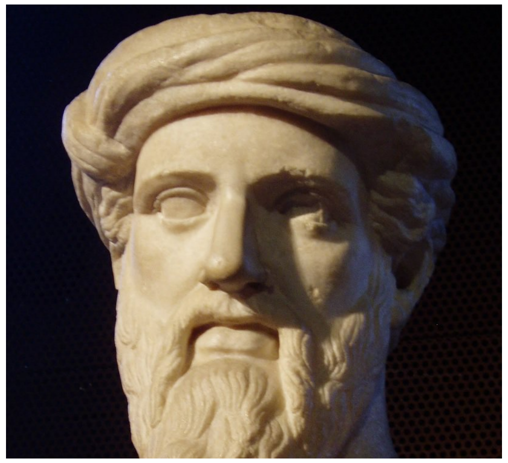
- When we describe computation, we use programming languages
- *But* we get to design languages and decide how they behave
- This is very different from physics
- We design the objects we study, as opposed to physicists who have to study the universe as it exists
- Einstein didn't like quantum mechanics, famously saying that God doesn't play dice with the universe,
  - but too bad, quantum mechanics is now firmly established
- On the other hand, if we don't like buffer overruns, we can just use a programming language which makes them impossible
- We get to define how our universe operates

Computer Science

- Computer Science
  - Study the laws of computation
  - Logic, computation, programming languages (hardware, software)
- Physics & Engineering
  - The physical stuff: transistors, electronics, memories, disks, fiber-optic communication, etc.

Math

- Mathematicians is considered the purest of the scientific fields
- Compare with Physics
  - Ideas about gravity have changed drastically from the time of Aristotle to Galileo to Newton to Einstein to the present. Theories are constantly replaced and refined.
  - Though one has to be careful here; I am not claiming these theories aren't useful: they got us to the moon, gave us airplanes, medical breakthroughs, …
- In contrast, the Pythagorean theorem is not going to change any time soon
- Theories are temporary, theorems are eternal



Logic & Computation

- Much of computer science is like mathematics
  - Programs are mathematical objects that can be reasoned about with mathematical precision, using logic
- Embedded systems and cyberphysical systems are interesting because they combine the cyber with the physical
  - Even here, much of the development of such systems is done using software modeling, which brings us back to programs and logic
- The point of this class is to explore the logical foundations of computer science at a level appropriate to freshmen

Motivation

- Why is reasoning about computation important?
- For one thing, we're scientists. We are the ones who are charged with understanding how things really work.
- You don't need science to build software
- Similarly, you don’t need physics and engineering to build a house
  - humans have been building mud huts for millenia
- However, if you want to build a skyscraper, you need to understand physics and engineering
- Similarly, if you want to build complex safety-critical systems you need to understand the underlying science

Safety & Reliability

- Why is reasoning about computation important?
  - Safety-critical systems, embedded systems, …: loss of life
  - Power plants, critical infrastructure, …: major disruptions
  - Medical records, …: loss of privacy
  - Financial institutions, e-commerce, …: loss of money
  - The AI apocalypse: extinction?

What Can Go Wrong?

- Ariane 5, $7B rocket devel. over 10 years, explodes due to software bug: cost $2B
- Space shuttle: testing costs $1000 per LOC
- Functional verification accounts for >40% of total chip costs (after Intel $475M FDIV bug)
- Toyota found guilty by Oklahoma jury for death due to software bugs
- NIST: cost of software bugs is $59.5B a year
- Gulf War: Patriot misses incoming Iraqi Scud: 28 soldiers killed; 100s injured; GAO: software bug
- USS Vincennes, using Aegis system, shot down an Iranian aircraft with pilgrims to Mecca: 290 dead
- People given lethal doses of radiation from Therac-25 machines (1980s) & Multidata software (2000s)
- Knight capital lost $500M in 1/2 an hour due to bug in trading software
- 1982 Soviet gas pipeline: CIA sabotaged trans-Siberian gas pipeline software: largest non-nuclear explosion
- Love virus: Virus from email attachment with subject “ILOVEYOU” infected millions of computers: $9B cost

Building Reliable Systems

- Problem for everyone: hardware, aerospace
  - Verification is ~2x the cost of software development
- Building large, interesting systems requires a lot of effort. One needs to consider life-cycle issues:
  - initial idea
  - requirements gathering
  - modeling
  - development
  - integration
  - evaluation
  - deployment
  - maintenance
- Finding flaws early in the life-cycle is exponentially better than finding flaws later

Overview of the Class

- The main focus is on the underlying basic, fundamental science
  - programming languages and their semantics
  - specifications: making clear, crisp statements about our programs
  - proving that our programs are correct
  - testing is widely used and we will see how that fits in
- The mathematics that underlies reasoning about computation is logic
- Computer science was created to answer questions in logic (more later)
- Logic will presented from a computational perspective

Overview of the Class

- The goal of the course is to introduce fundamental, foundational methods for
  - modeling, designing and reasoning about computation, including
    - contract-based design
    - invariants
    - testing
    - propositional logic
    - equational reasoning
    - recursion & induction
    - termination analysis
    - various proof techniques
    - software verification

Next Time

- Course Webpages
- Designing programs, review
- Introduction to the ACL2s language