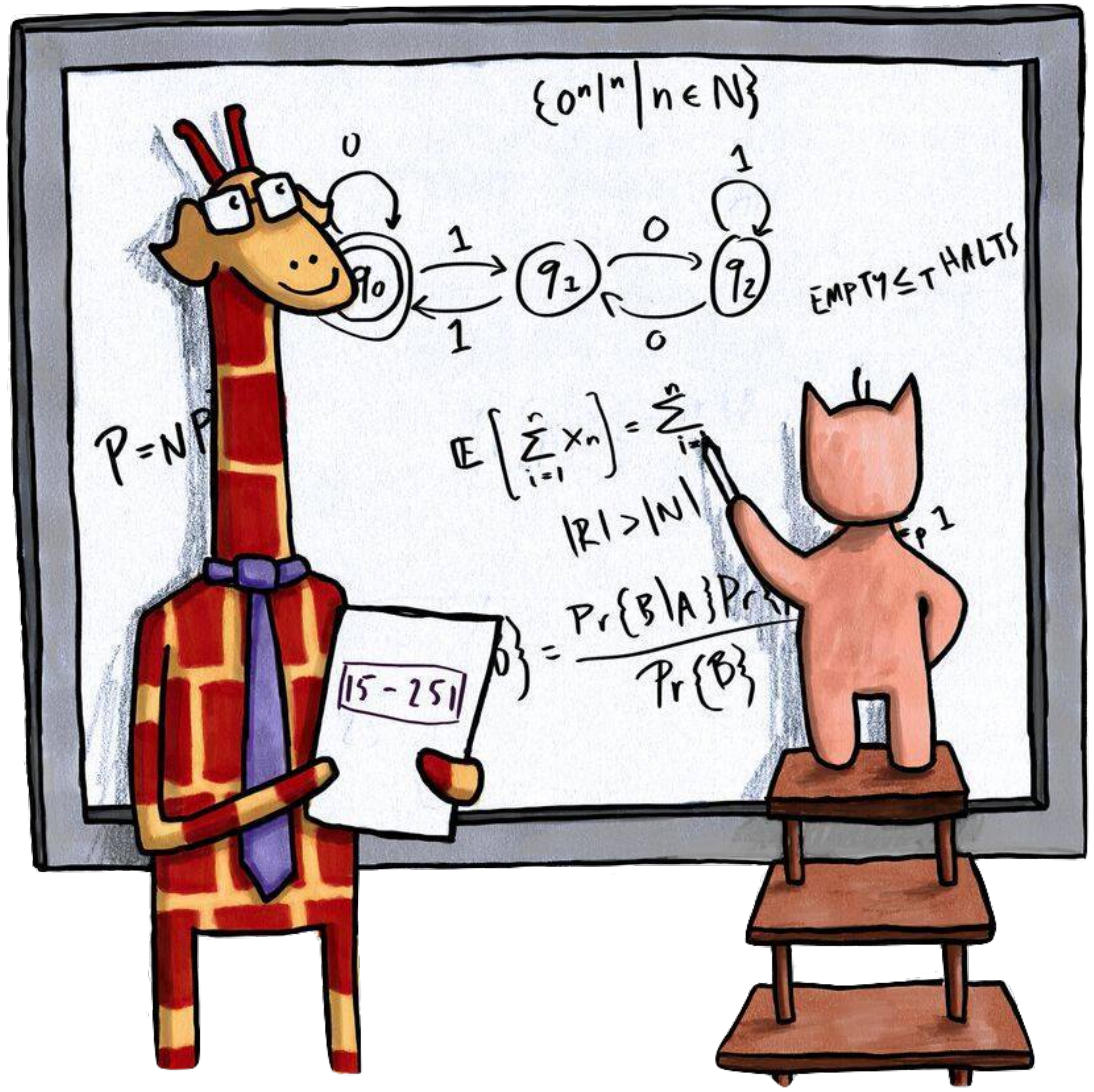
# CS251

## Great Ideas in *Theoretical* Computer Science




## Introduction to the Course Part 1

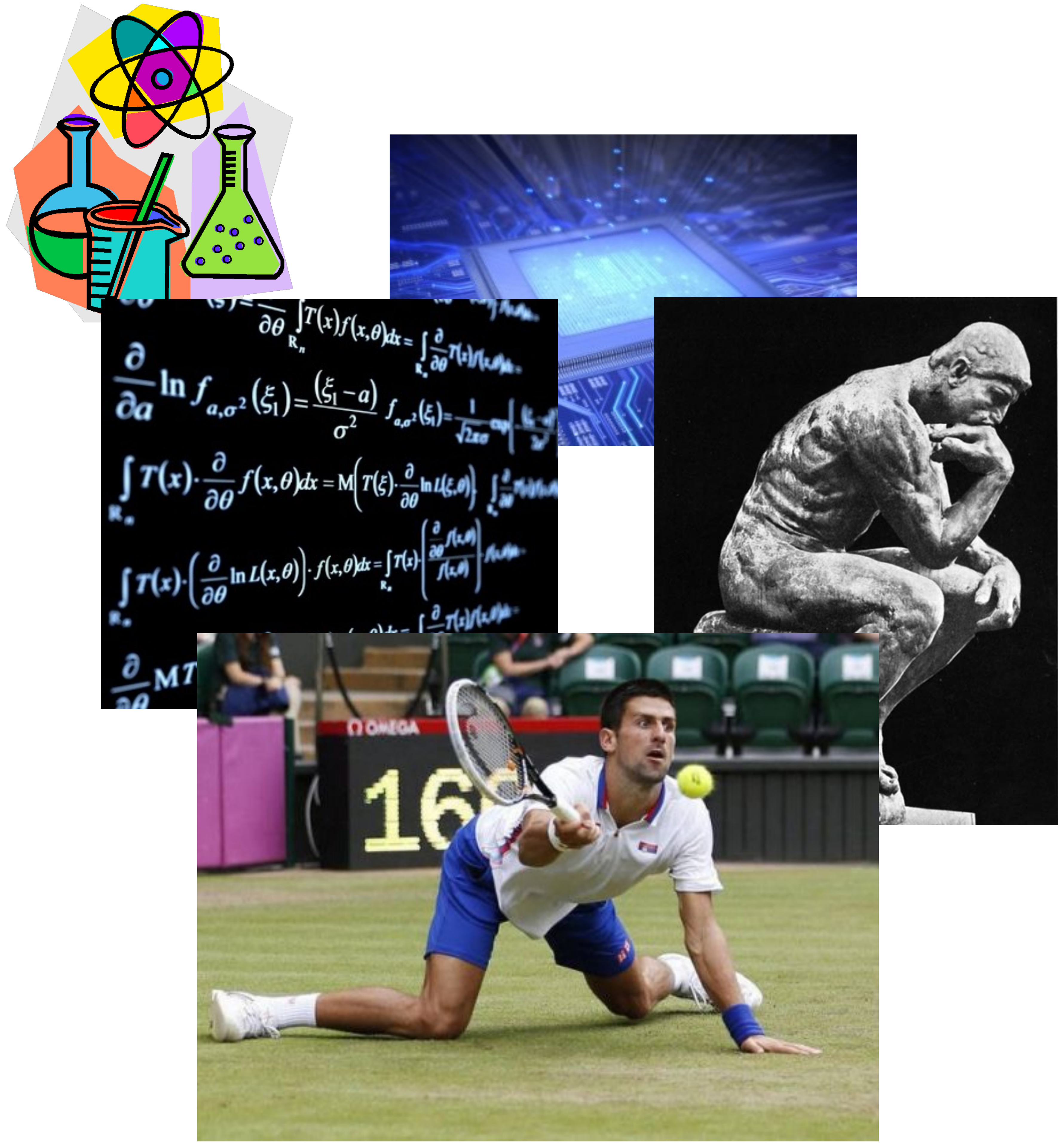
What is **computer science**?

What is ***theoretical* computer science**?

# What is computer science?

Is it a branch of:

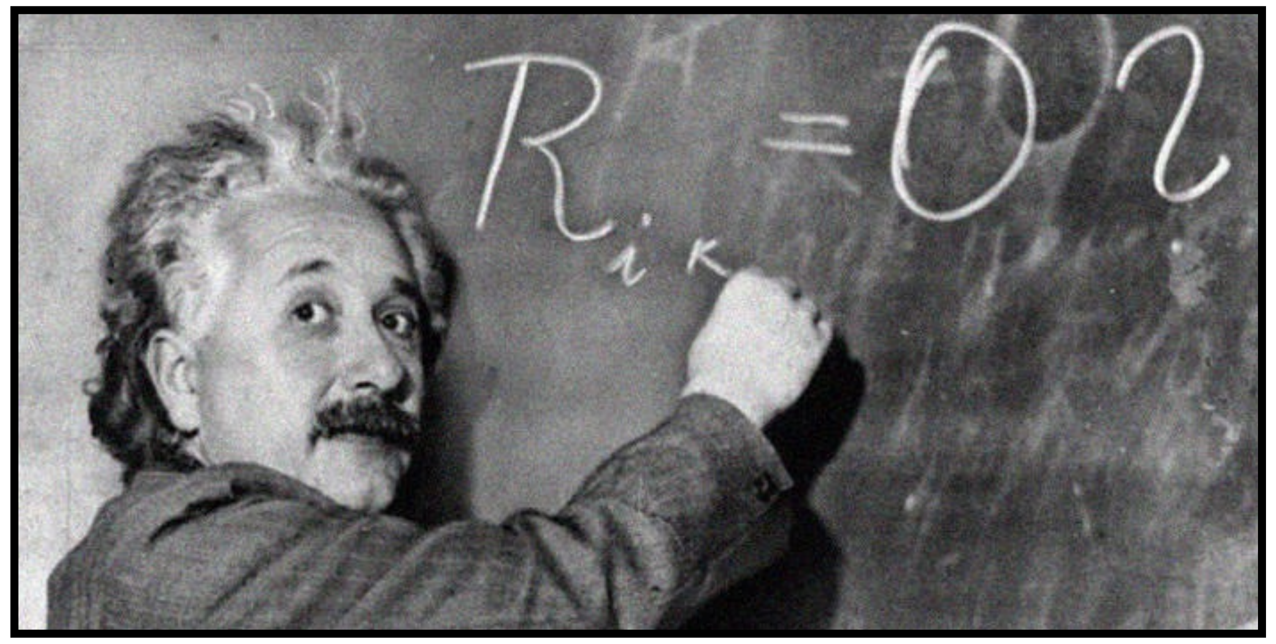
- science?
- engineering?
- math?
- philosophy?
- sports?

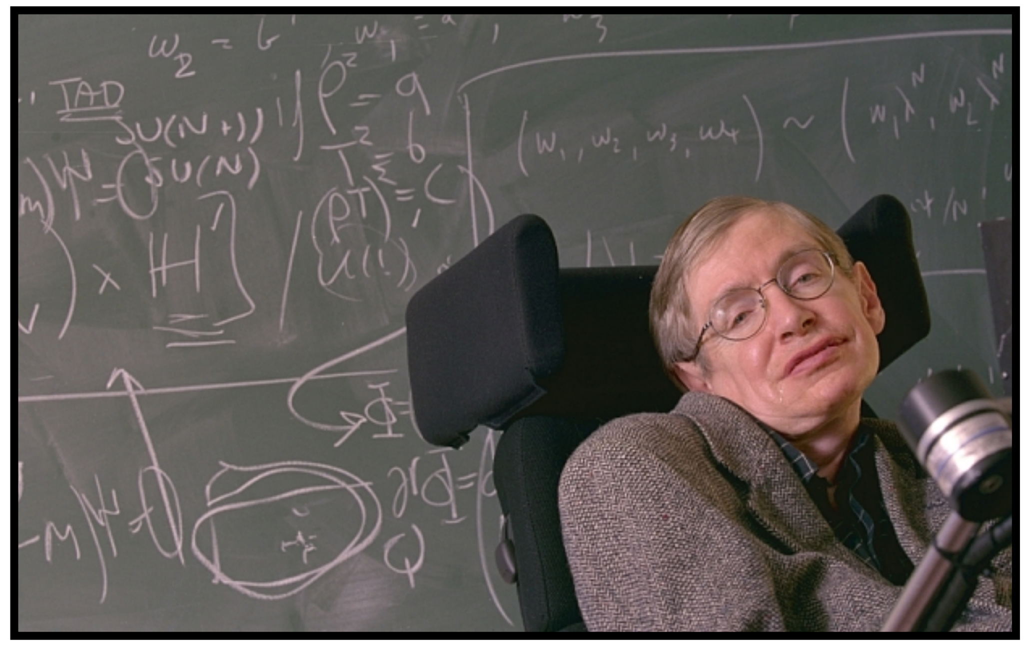


## Motivational Quote of the Course:

*Computer Science is no more about computers than astronomy is about telescopes.*

# Physics





## Theoretical physics

- come up with mathematical models

  *nature's language is mathematics*

- derive the logical consequences

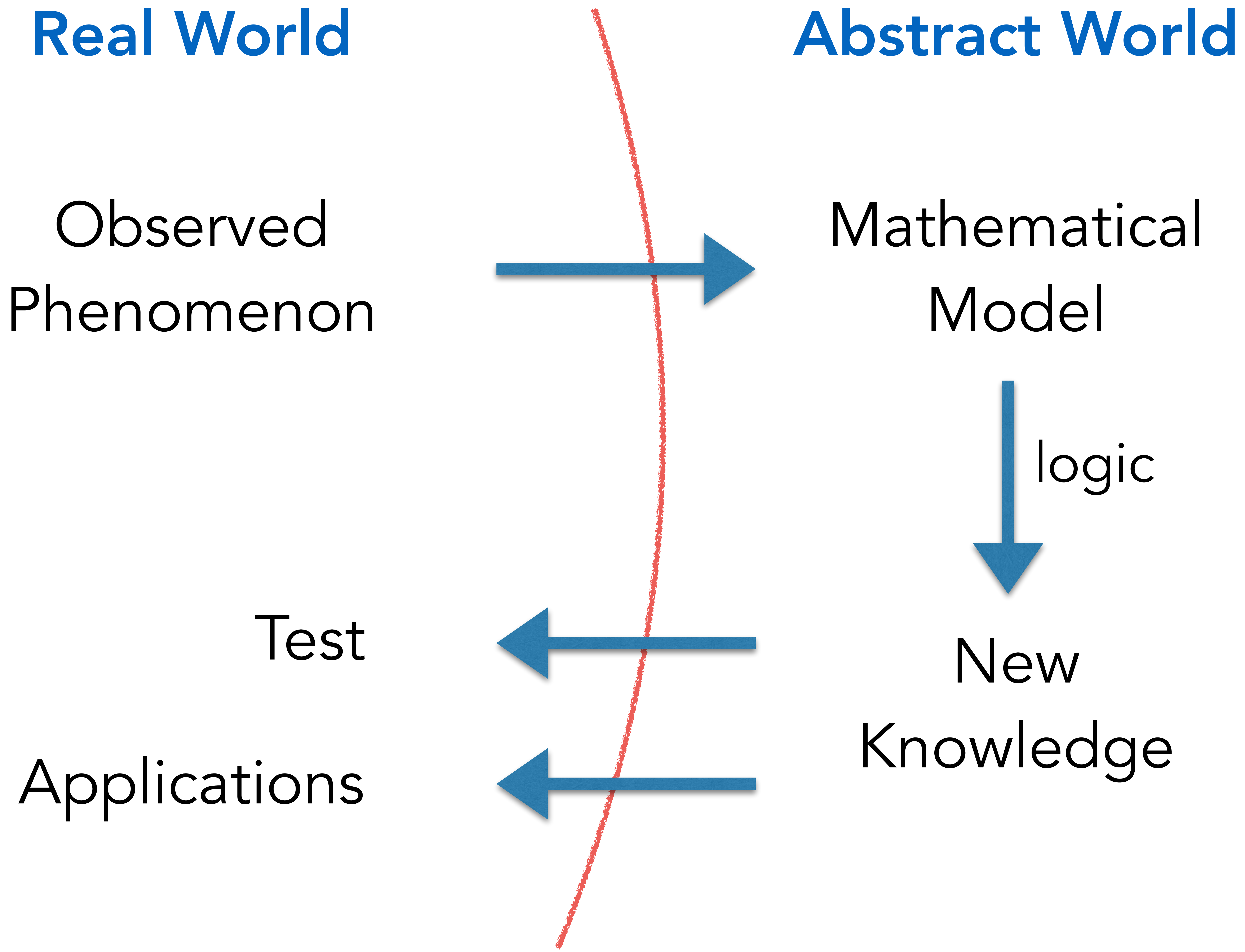
## Experimental physics

- make observations about the universe

- test mathematical models with experiments

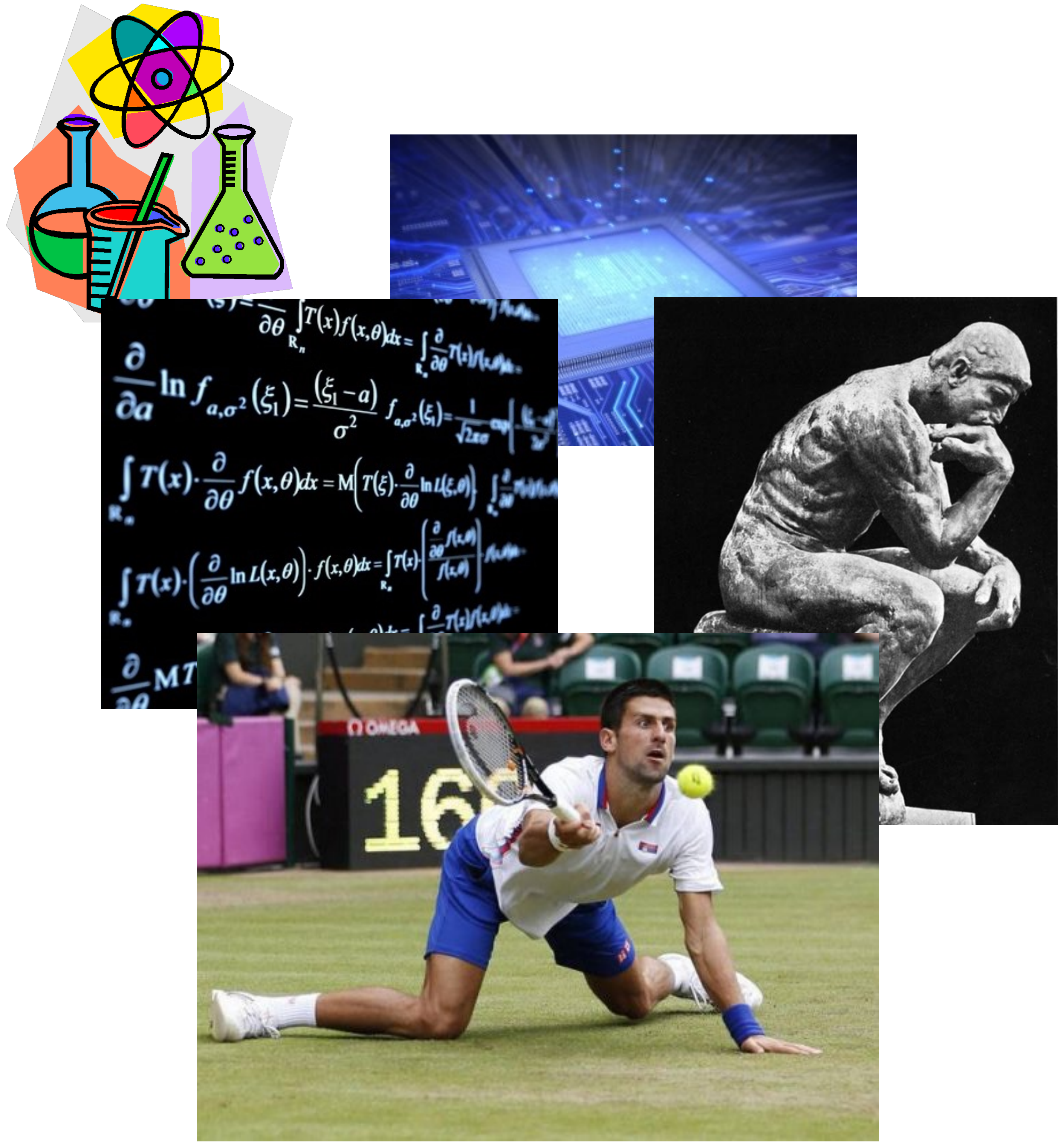
## Applications/Engineering

# The role of theoretical physics



# Physics

- science?
- engineering?
- math?
- philosophy?
- sports?



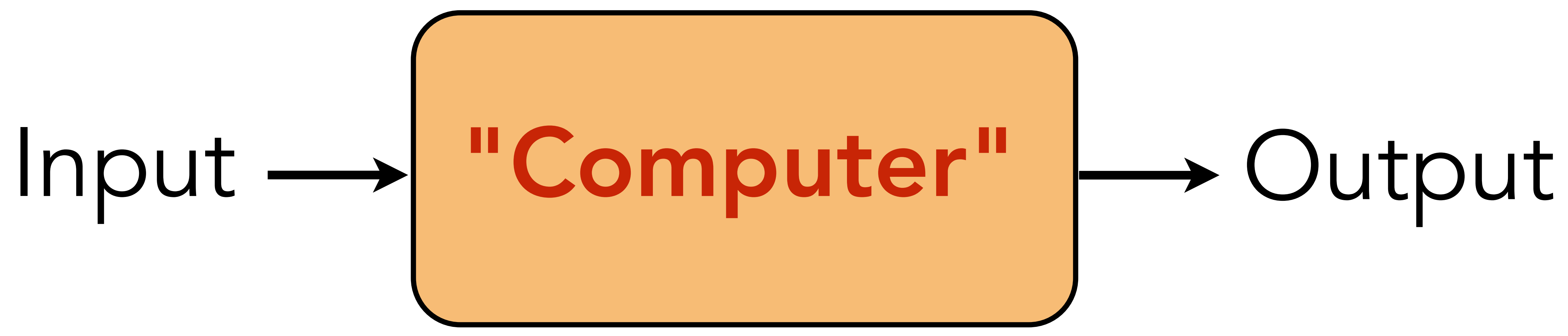
# Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.



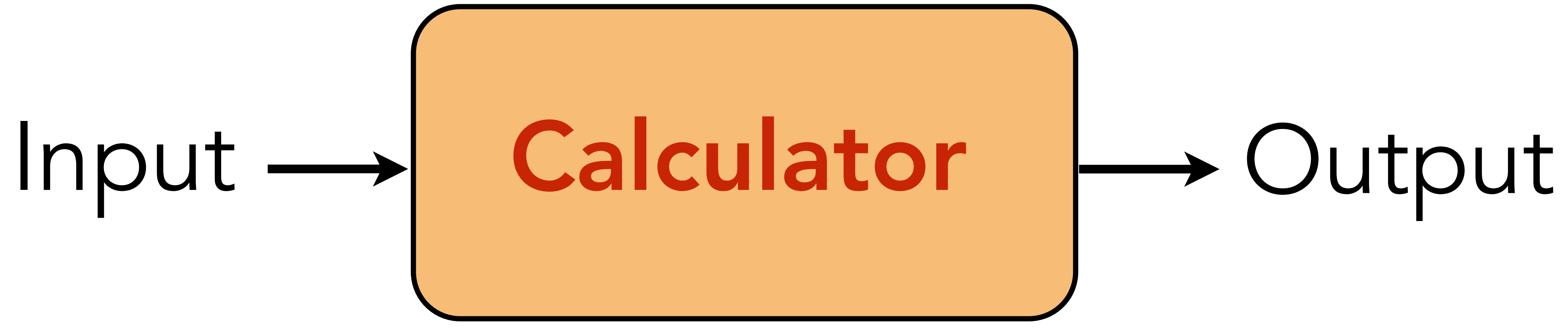
# Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.



# Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

# Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

# "Computers" in early 20th century

# Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

# The computational lens

physics

biology

chemistry

neuroscience

economics

finance

linguistics

statistics

social choice

…

# The computational lens

Computational physics

Computational biology

Computational chemistry

Computational neuroscience

Computational economics

Computational finance

Computational linguistics

Computational statistics

Computational social choice

...

# The role of *theoretical* computer science

Build a mathematical model for computation.

Explore the logical consequences.
Gain insight about computation.

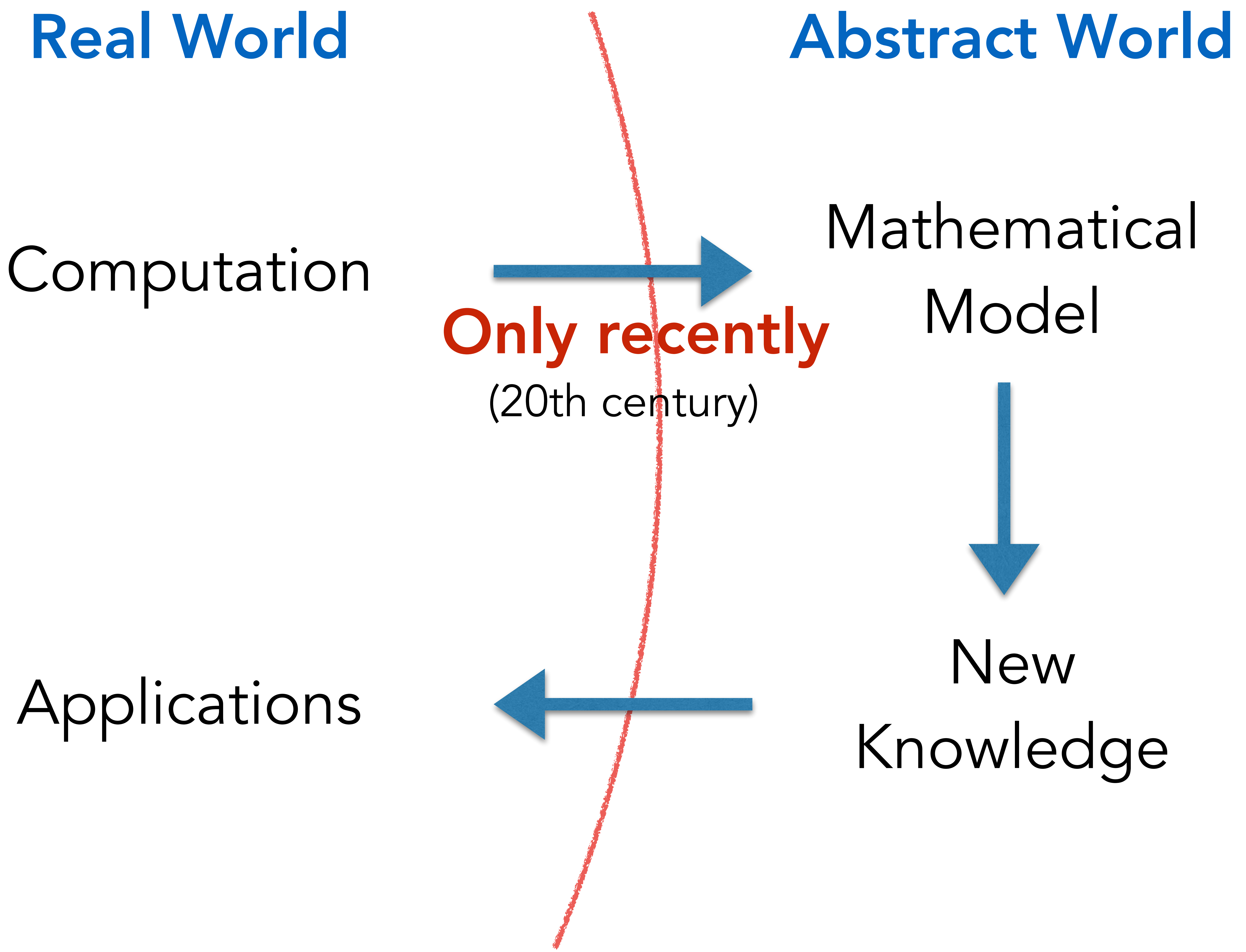
Look for interesting applications.

CMU undergrad

CMU Prof.

OK, we don't have everybody

# The role of *theoretical* computer science



We have been using algorithms for thousands of years.

**Multiplication algorithm**

```
        5127
  x     4265
  ----------
       25635
      30762
     10254
+   20508
  ----------
    21866655
```

**Euclid's algorithm (~ 300BC)**

```
def gcd(a, b):
  while (a != b):
    if (a > b): a = a - b
    else: b = b - a
  return a
```

# Formalizing computation

Someone had to ask the right **question**.

# David Hilbert, 1900

## The Problems of Mathematics

*"Who among us would not be happy to lift the veil behind which is hidden the future; to gaze at the coming developments of our science and at the secrets of its development in the centuries to come?*
*What will be the ends toward which the spirit of future generations of mathematicians will tend?*
*What methods, what new facts will the new century reveal in the vast and rich field of mathematical thought?"*

# 2 of Hilbert's Problems

## Hilbert's 10th problem (1900)

Is there a finitary procedure to determine if
a given multivariate polynomial with integral coefficients
has an integral solution?

e.g. $$5x^2yz^3 + 2xy + y - 99xyz^4 = 0$$

## Entscheidungsproblem (1928)

Is there a finitary procedure to determine the validity
of a given mathematical statement?

e.g. $$\neg\exists x, y, z, n \in \mathbb{N} : (n \geq 3) \wedge (x^n + y^n = z^n)$$

(Mechanization of mathematics)

# 2 of Hilbert's Problems

**Fortunately**, the answer turned out to be NO.

# 2 of Hilbert's Problems

**Post (1920s):** Had a definition, but not confident.

**Gödel (1934):** Ideas for a definition, but not confident.

**Church (1936):** Invents *lambda calculus*.
Claims it should be the definition of an "algorithm".

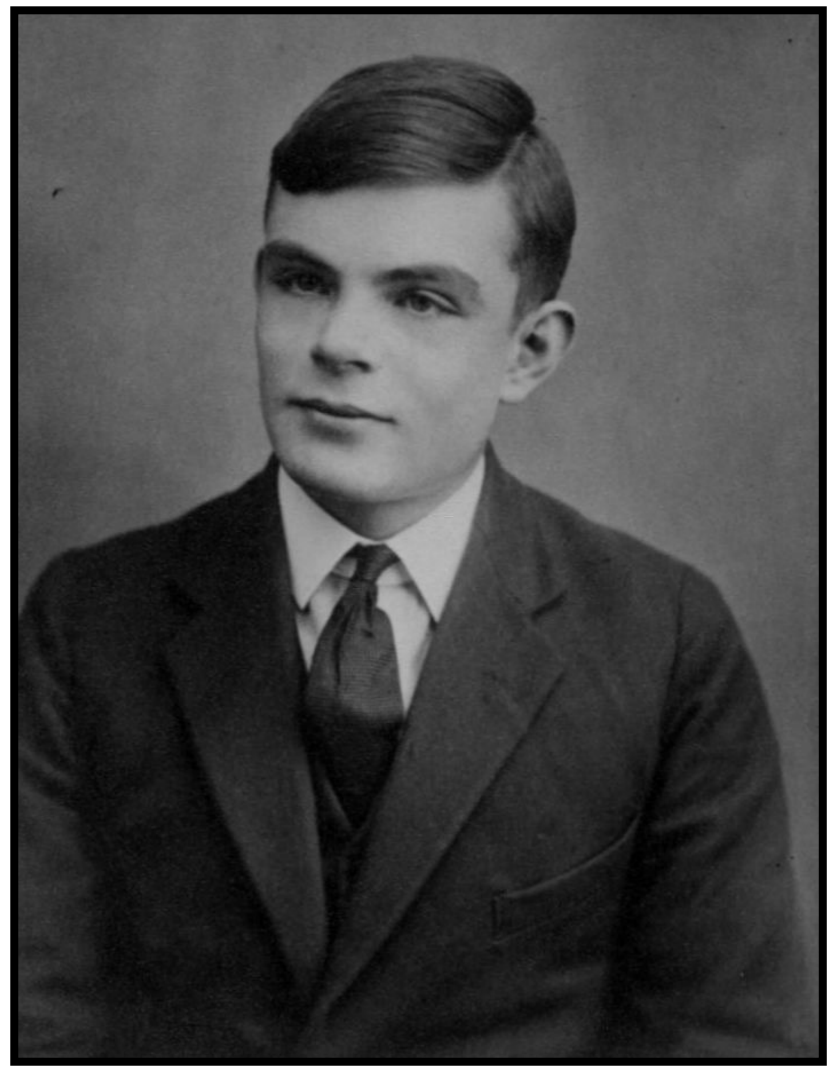
**Gödel, Post (1936):**
Arguments that Church's claim is not justified.

Meanwhile… a British grad student,
unaware of the debates…

# 2 of Hilbert's Problems

**Alan Turing (1936, age 22):**

Describes a new model for computation, now known as the Turing machine.™



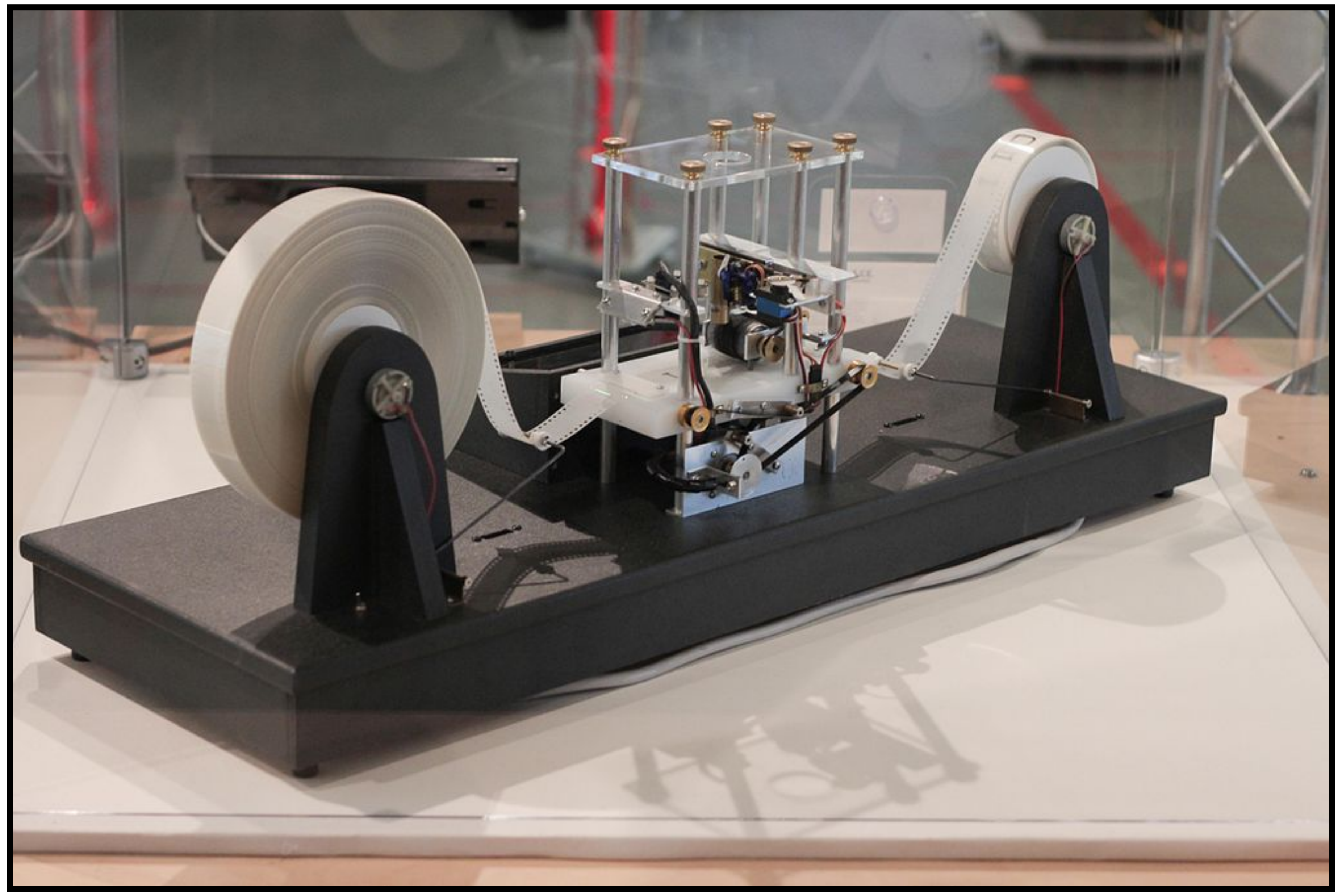
**Mathematicians at the time (even Church):**

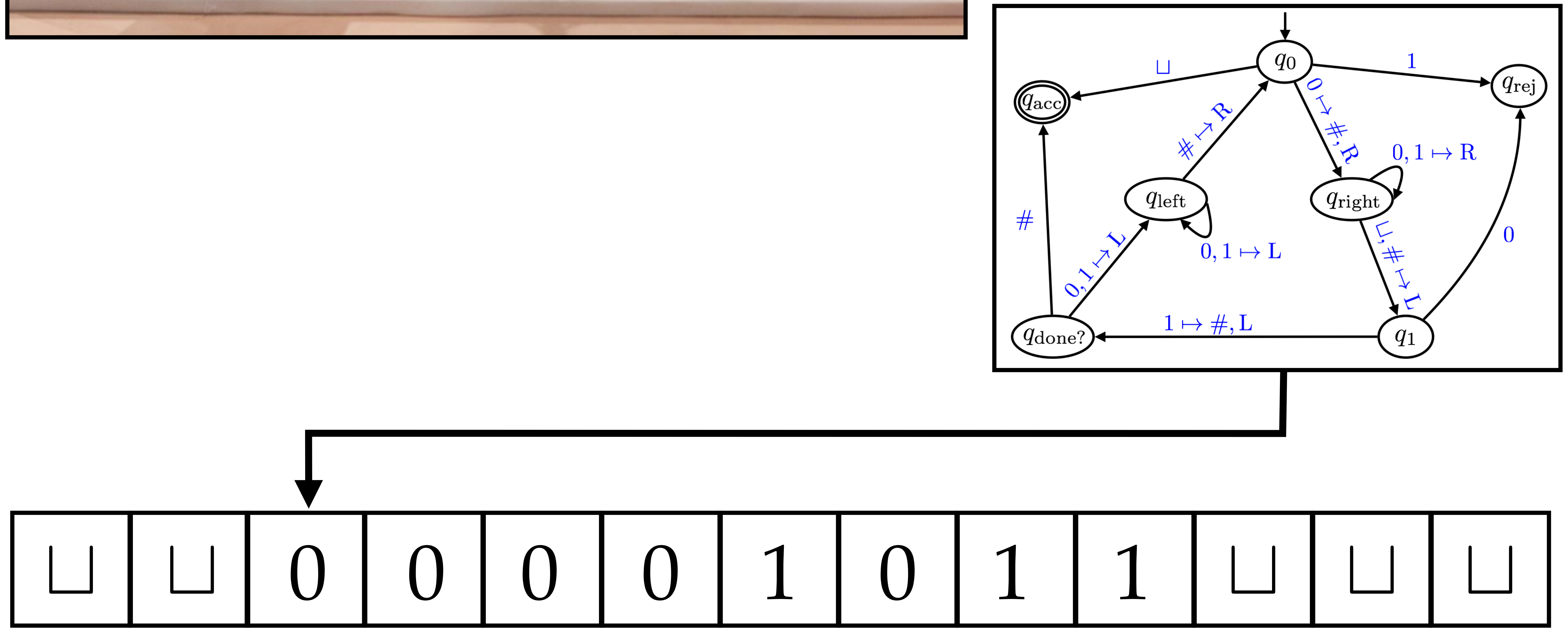
"Umm. Yeah. He nailed it. Game over.
'Algorithm' defined."

**Turing (1937):**

TMs ≡ lambda calculus

# Formalization of computation: Turing machine



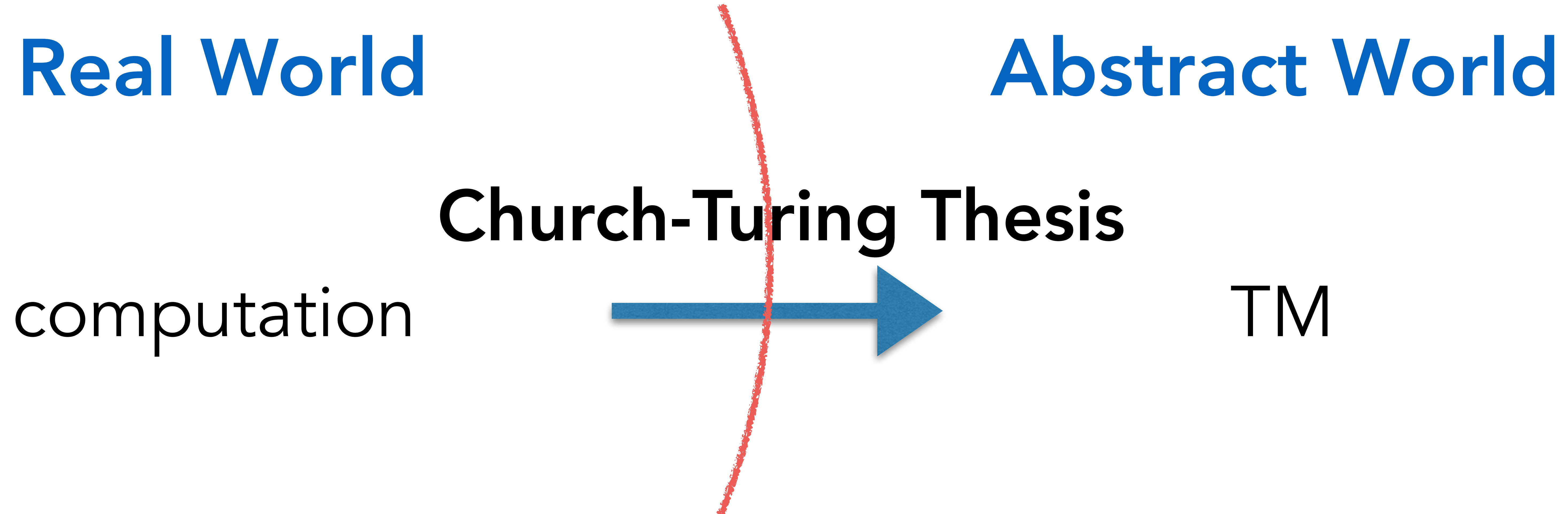


# Church-Turing Thesis

*The intuitive notion of "computable" is captured by functions computable by a Turing Machine.*

**(Physical) Church-Turing Thesis:**

Any computational problem that can be solved by any physical process, can be solved by a TM.



# Back to Hilbert's Problems

## Hilbert's 10th problem (1900)

Is there an algorithm (TM) to determine if a given multivariate polynomial with integral coefficients has an integral solution?

e.g. $$5x^2yz^3 + 2xy + y - 99xyz^4 = 0$$

## Entscheidungsproblem (1928)

Is there a algorithm (TM) to determine the validity of a given mathematical statement?

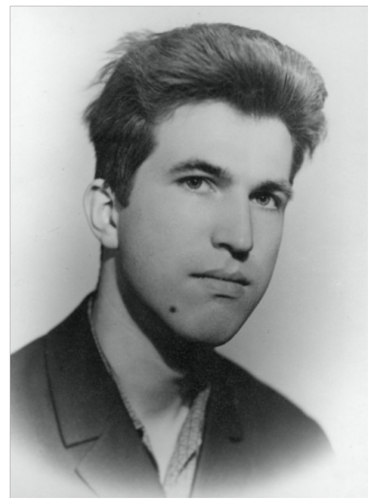
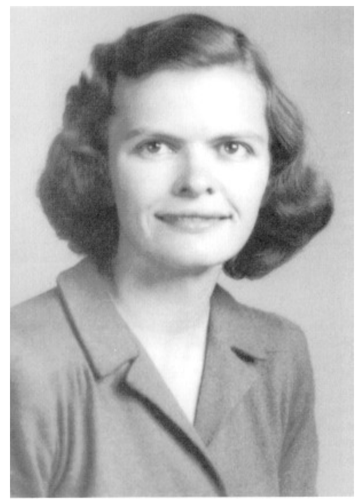
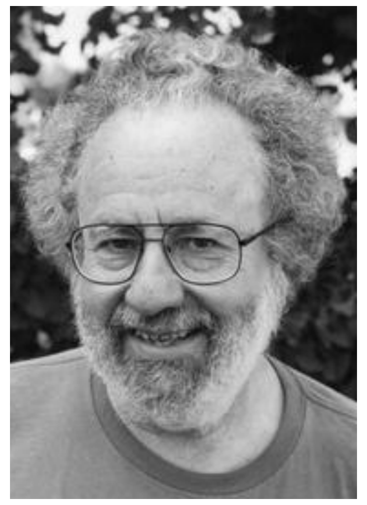
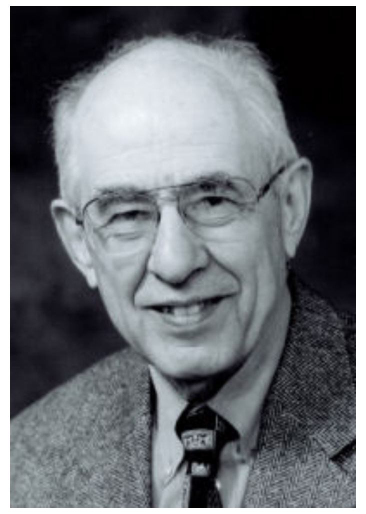
e.g. $$\neg\exists x, y, z, n \in \mathbb{N} : (n \geq 3) \wedge (x^n + y^n = z^n)$$

(Mechanization of mathematics)

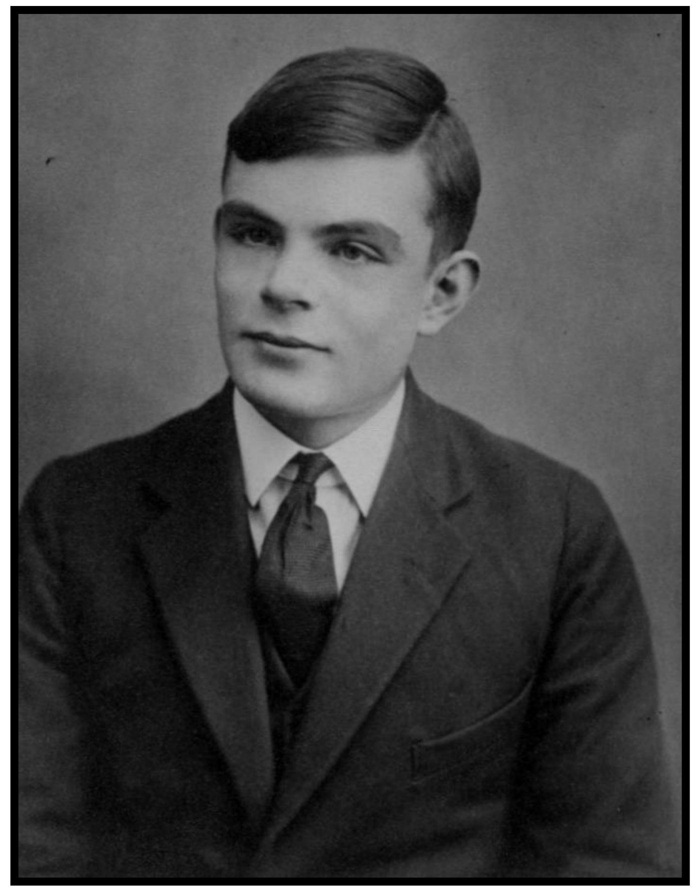
# Back to Hilbert's Problems

## Hilbert's 10th problem (1900)

Matiyasevich-Robinson-Davis-Putnam (1970): **NO!**

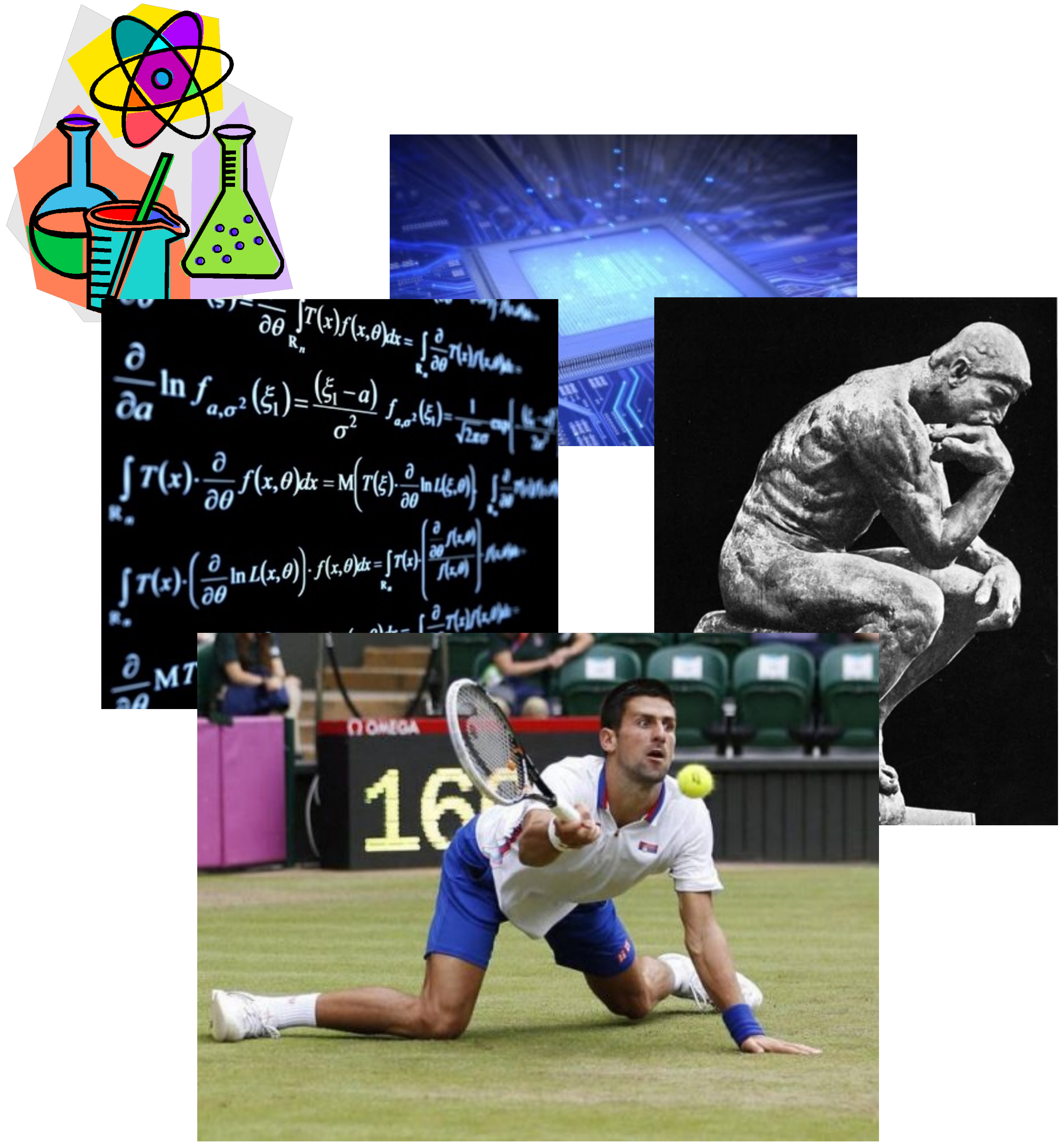
## Entscheidungsproblem (1928)



Turing (1936): **NO!**

# Computer science

- science?
- engineering?
- math?
- philosophy?
- sports?



# More on Theoretical Computer Science (TCS)

# 2 Main Questions in TCS

**Computability** of a problem:

Is there an algorithm to solve it?

**Complexity** of a problem:

Is there an *efficient* algorithm to solve it?

- time
- space (memory)
- randomness
- quantum resources

# Computational Complexity

**Complexity** of a problem:

Is there an *efficient* algorithm to solve it?

- time
- space (memory)
- randomness
- quantum resources

**2 camps:**

- ***algorithm designers***
  trying to come up with efficient algorithms
- ***complexity theorists***
  trying to show no efficient algorithm exists

# Computational Complexity

**2 camps:**

- ***algorithm designers***
  trying to come up with efficient algorithms

- ***complexity theorists***
  trying to show no efficient algorithm exists

multiplying two integers
factoring integers
detecting communities in social networks
protein structure prediction
simulation of quantum systems
computing Nash Equilibria of games

# P vs NP

ABOUT PROGRAMS MILLENNIUM PROBLEMS PEOPLE PUBLICATIONS EUCLID EVENTS

## Millennium Problems

### Yang–Mills and Mass Gap
Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

### Riemann Hypothesis
The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

### P vs NP Problem

**million dollar question**

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

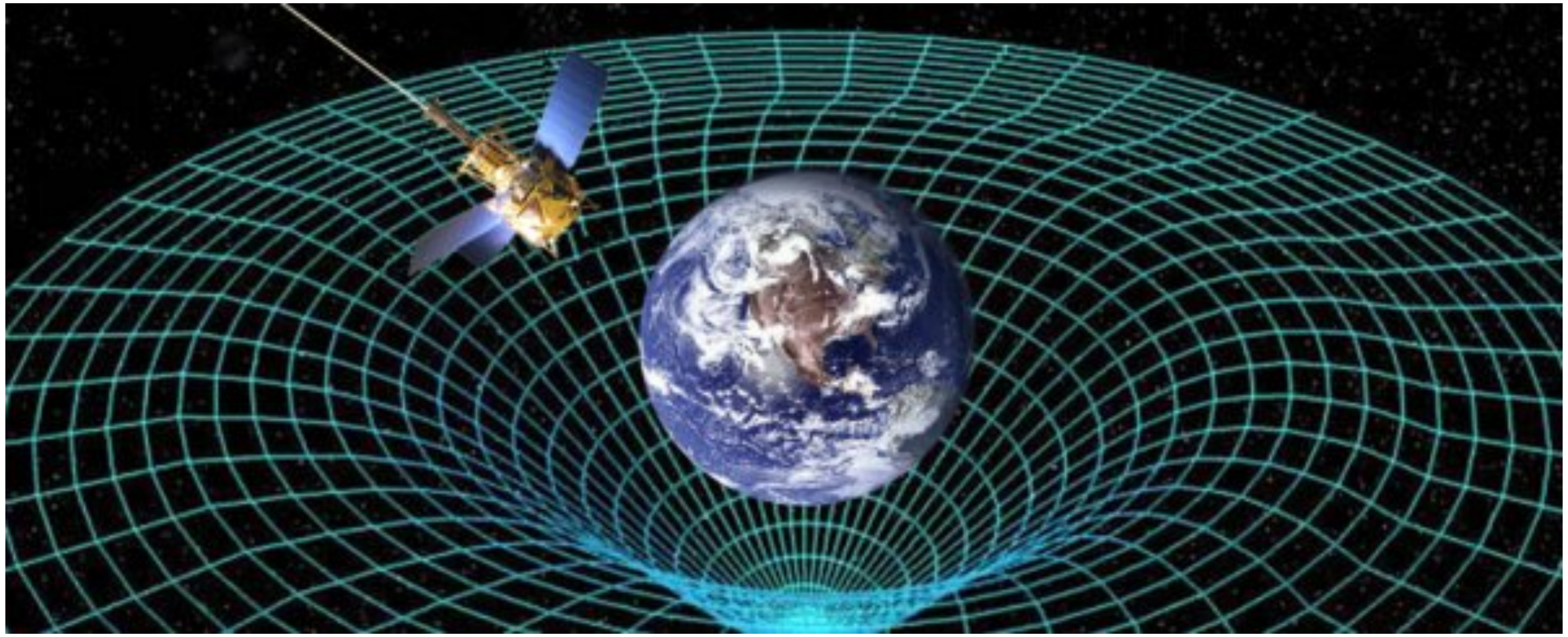
### Navier–Stokes Equation
This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certitude, but also understanding.

### Hodge Conjecture
The answer to this conjecture determines how much of the topology of the solution set of a system of algebraic equations can be defined in terms of further algebraic equations. The Hodge conjecture is known in certain special cases, e.g., when the solution set has dimension less than four. But in dimension four it is unknown.

### Poincaré Conjecture
In 1904 the French mathematician Henri Poincaré asked if the three dimensional sphere is characterized as the unique simply connected three manifold. This question, the Poincaré conjecture, was a special case of Thurston's geometrization conjecture. Perelman's proof tells us that every three manifold is built from a set of standard pieces, each with one of eight well-understood geometries.

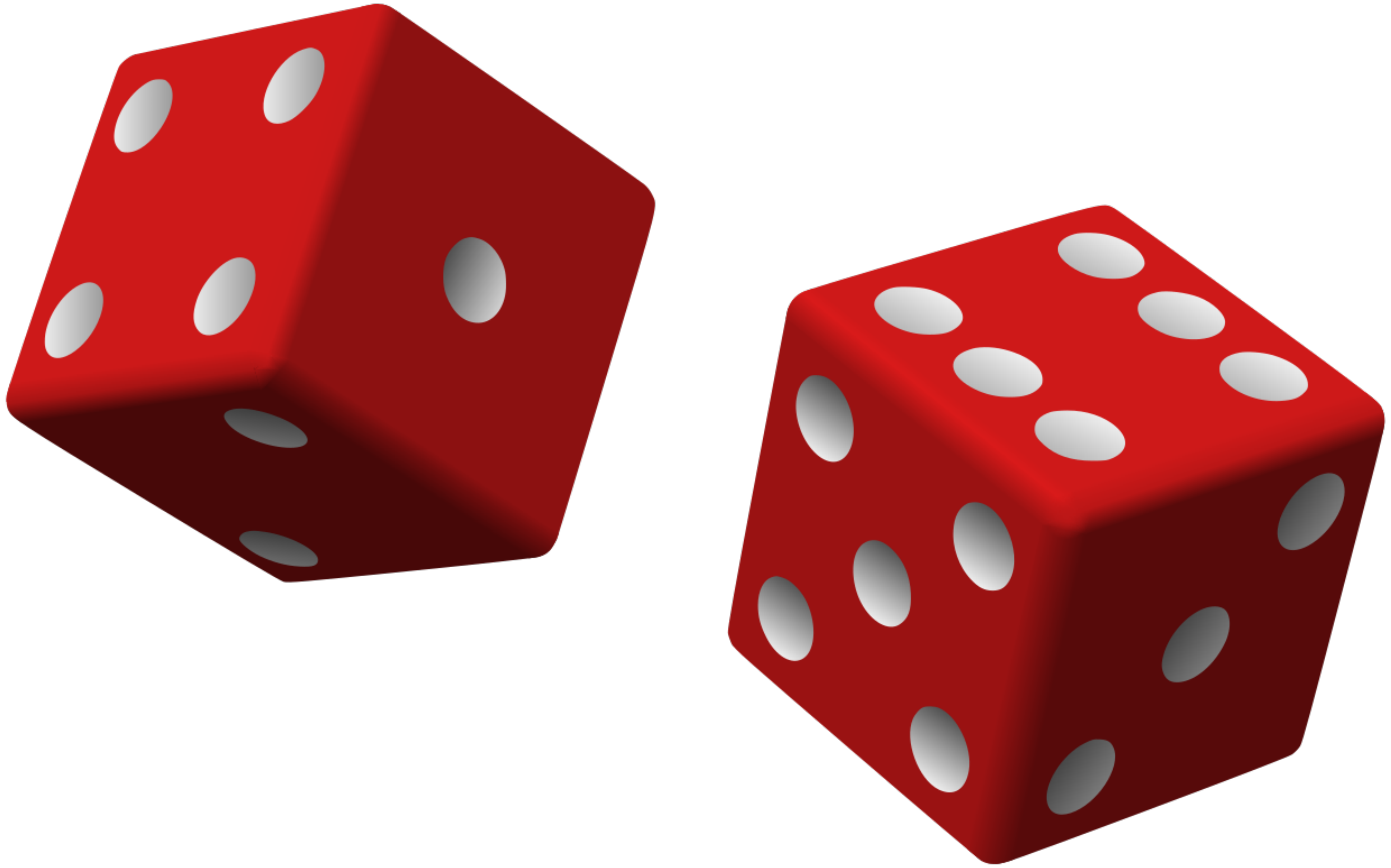
### Birch and Swinnerton-Dyer Conjecture
Supported by much experimental evidence, this conjecture relates the number of points on an elliptic curve mod p to the rank of the group of rational points. Elliptic curves, defined by cubic equations in two variables, are fundamental mathematical objects that arise in many areas: Wiles' proof of the Fermat Conjecture, factorization of numbers into primes, and cryptography, to name three.

**Time** vs **Space**



$$\text{space-efficient} \overset{?}{\Longrightarrow} \text{time-efficient}$$

# **Deterministic** vs **Randomized**

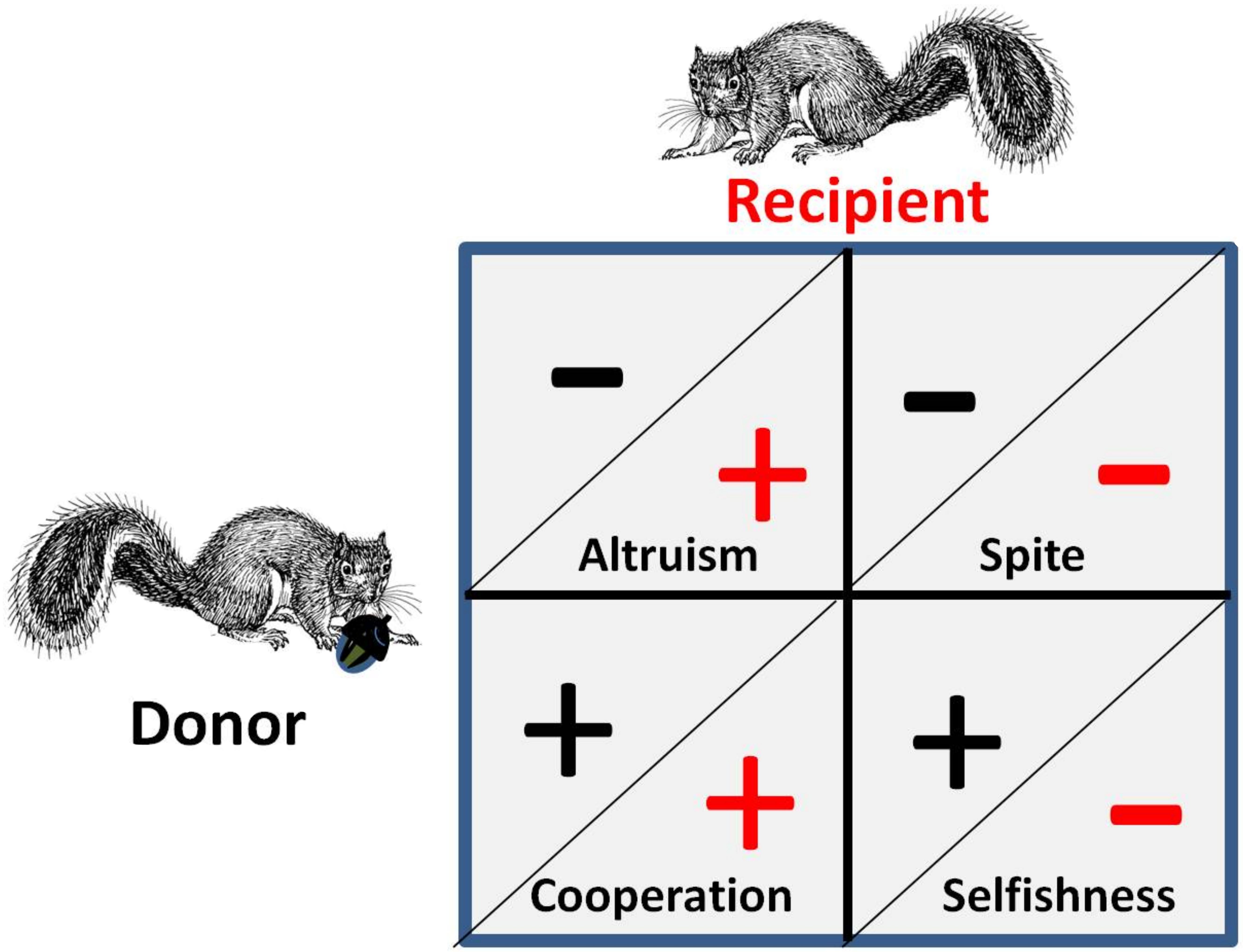


# Cryptography & Security

**Socioeconomics**

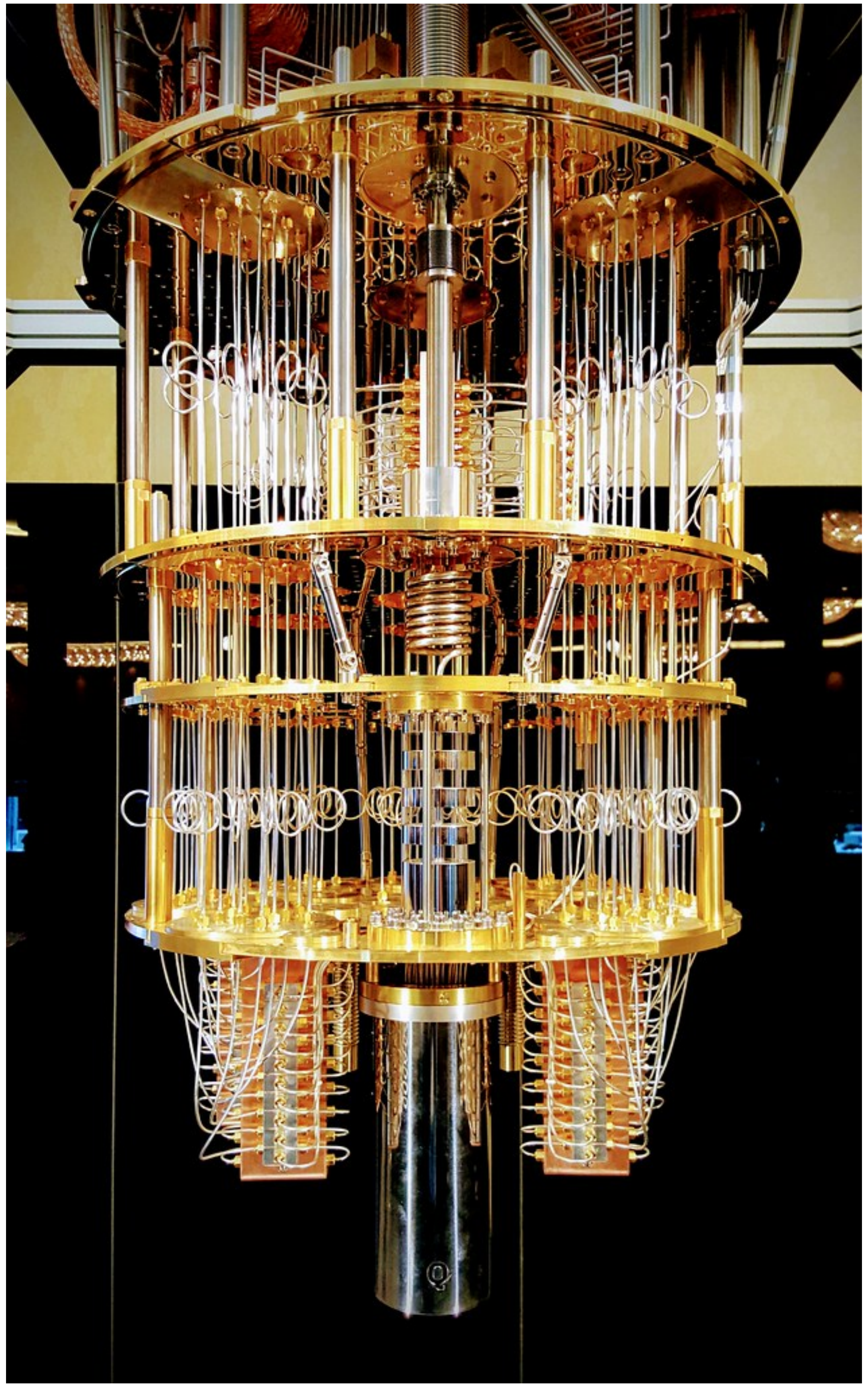
(e.g. privacy, fairness)

# Algorithmic Game Theory



# Learning Theory

# Quantum Computation



• • •

# Learning Objectives

# Perspective 0

…

# Perspective 1

## Overview of Topics

**Part 1**: Formalizing the notions of problems, algorithms, and computability.

**Part 2**: Computational complexity: theory and applications.

**Part 3**: Highlights in theoretical CS.

# BIG-PICTURE COURSE

Finite automata

Turing machines

Uncountability, Undecidability, Unprovability

Graph theory

Time complexity

P vs NP

Approximation algorithms

Probability

Randomized algorithms

Basic number theory

Cryptography

Quantum computation

Game Theory

Communication Complexity

…

# Perspective 1

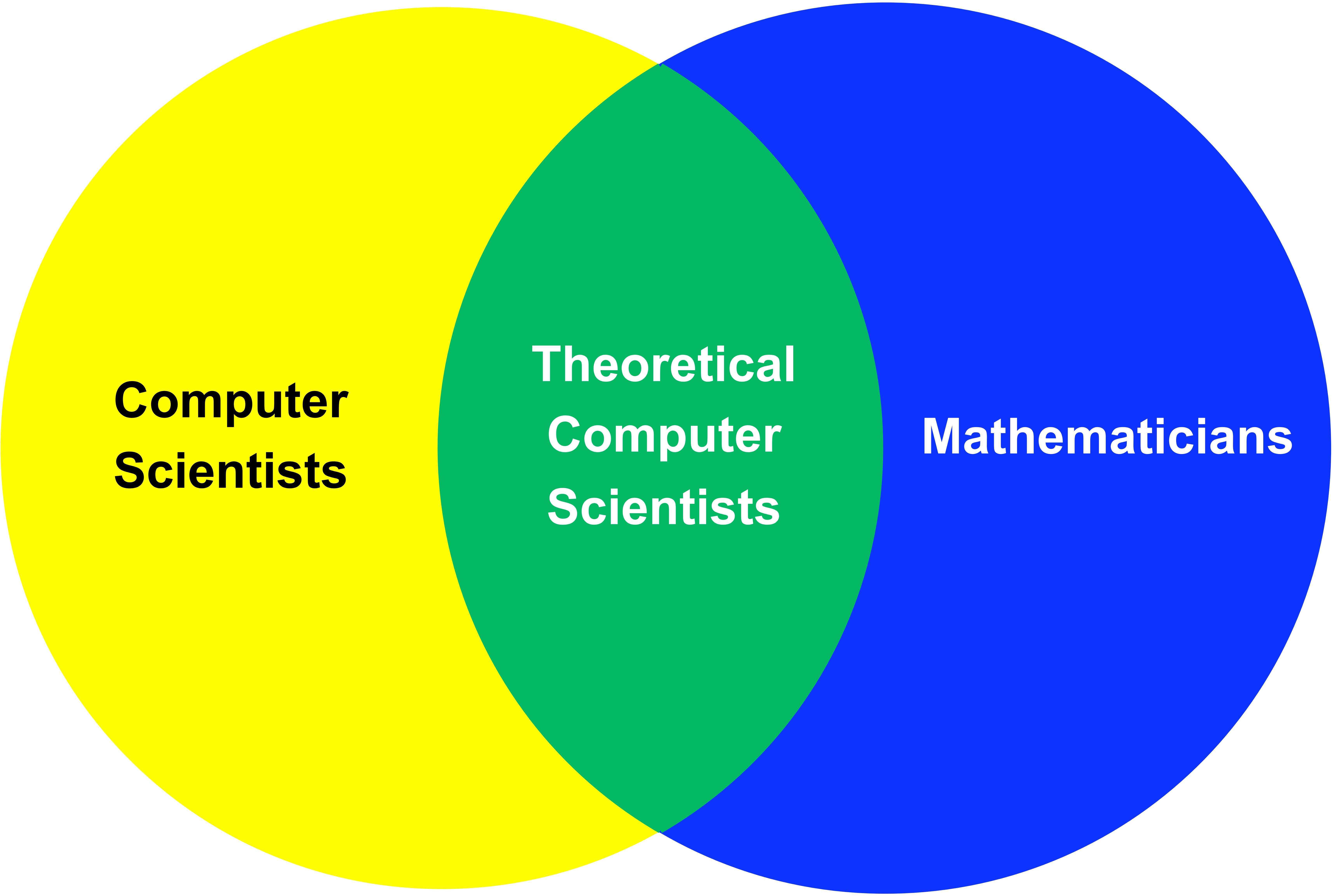
## The Skills

- Improve your rigorous, logical and abstract thinking skills.

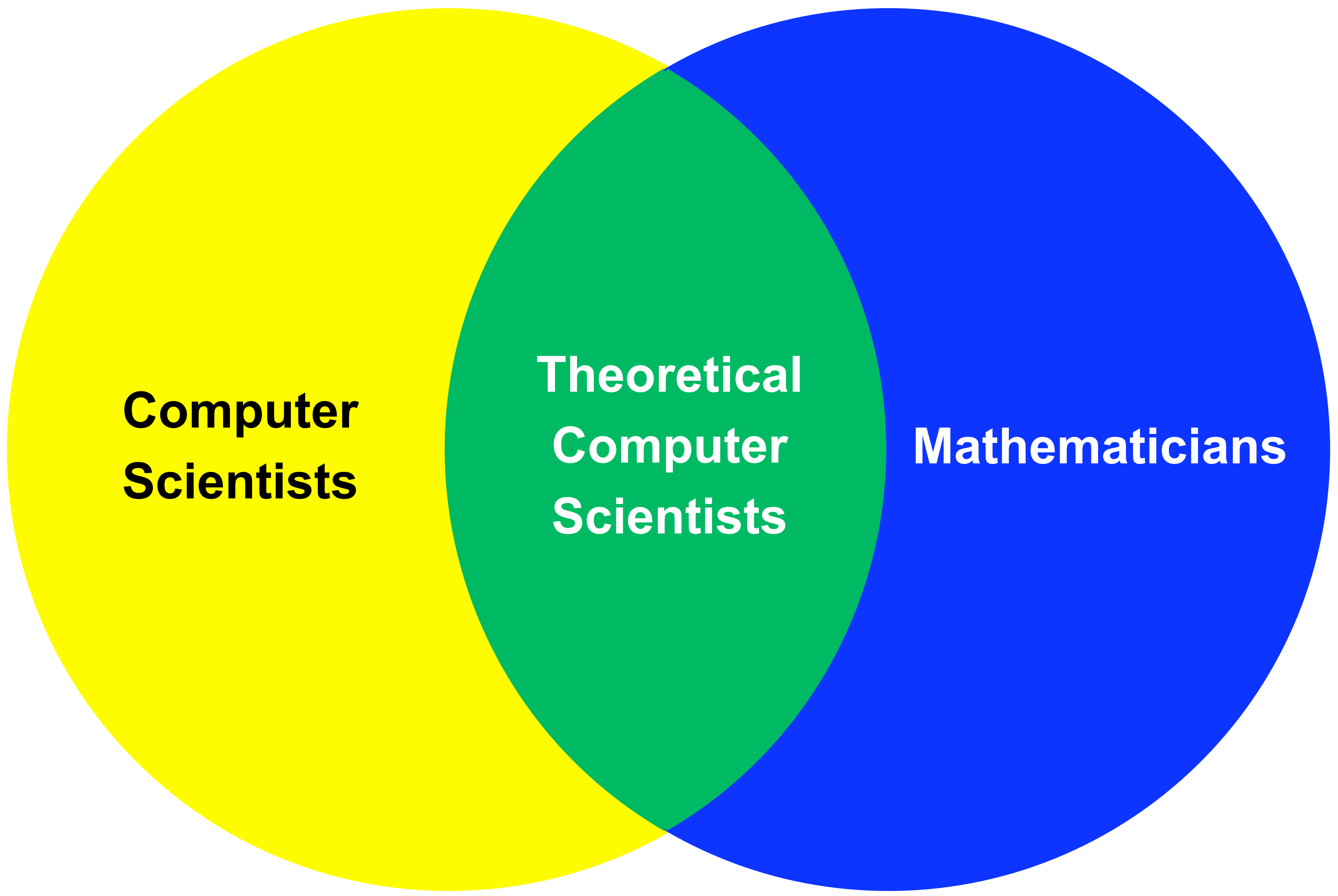
- Prepare you to be innovators.

- Push you to strive for clarity of thought and clarity in expression of thought.

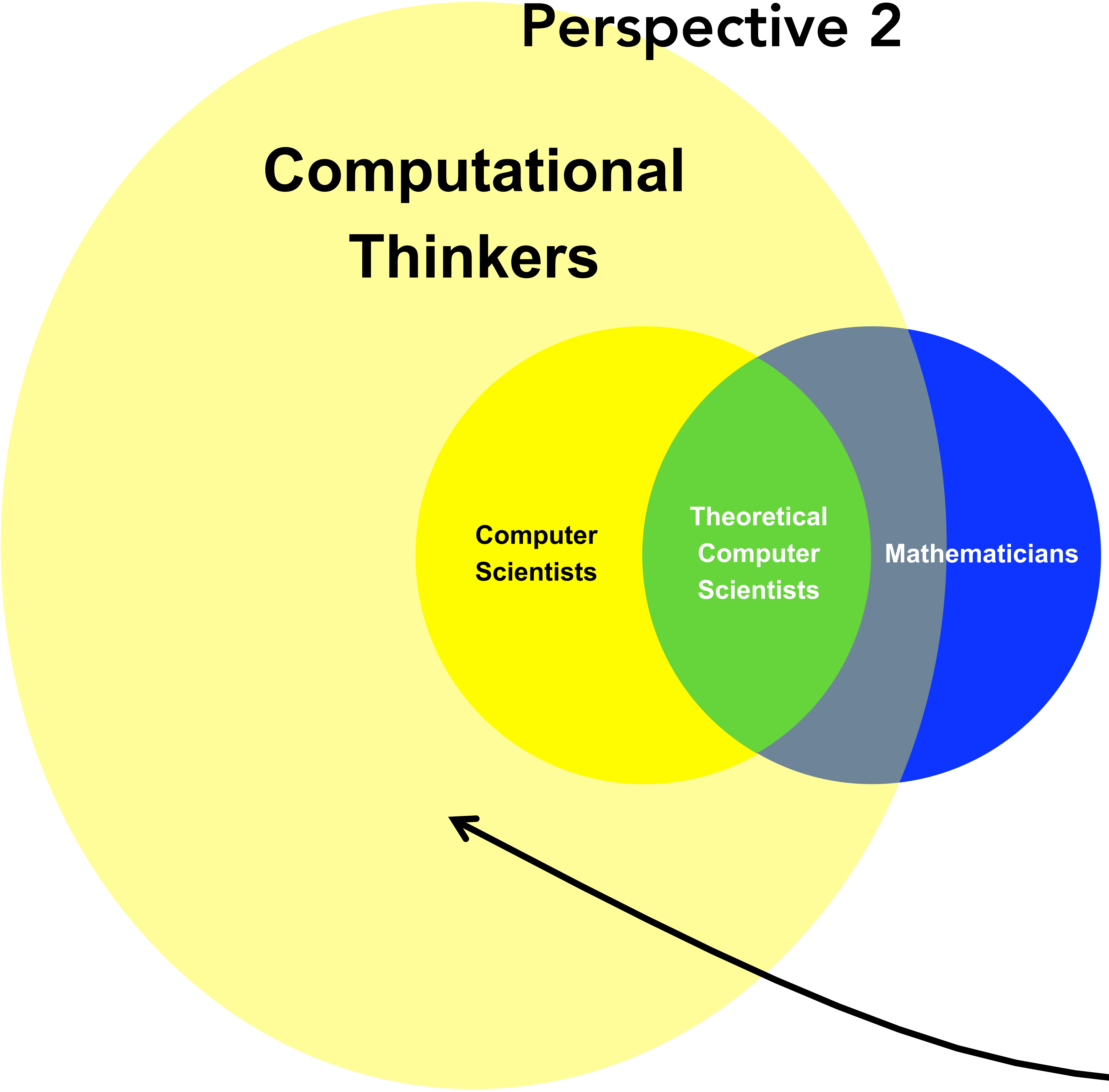
# Perspective 2



# Perspective 2



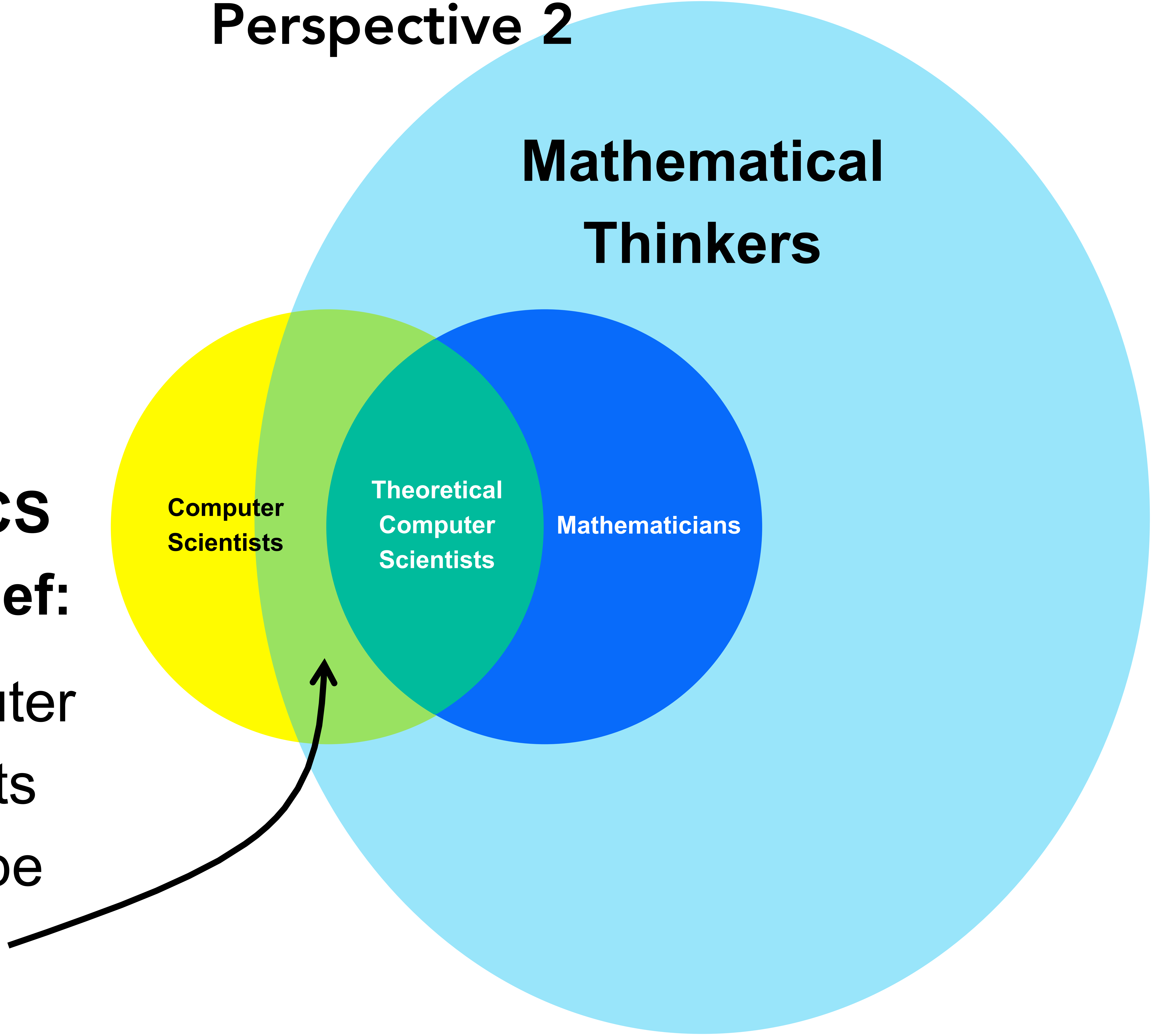
# Perspective 2




**CMU SCS core belief:**

all scientists should be here

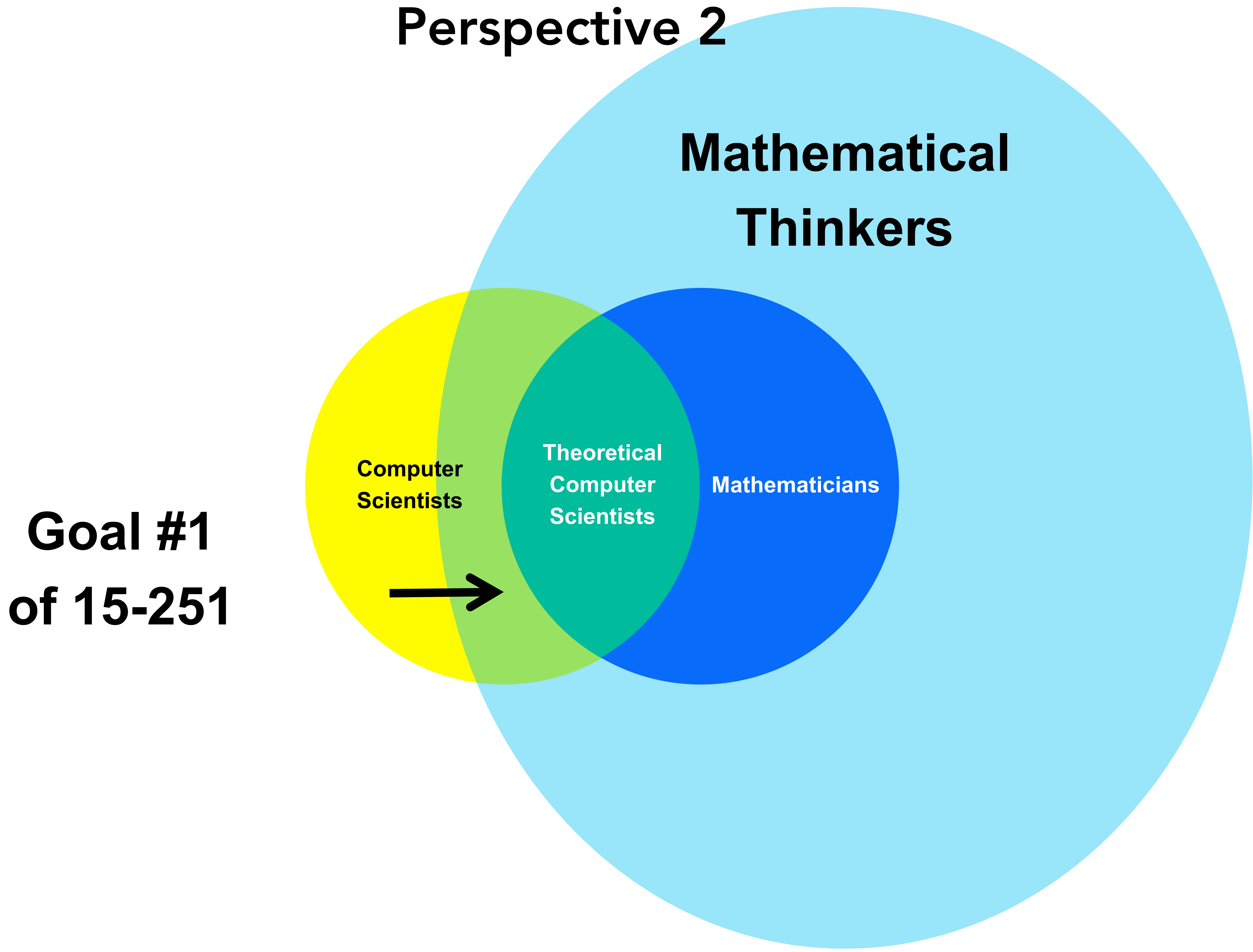
# Perspective 2




**CMU SCS core belief:**

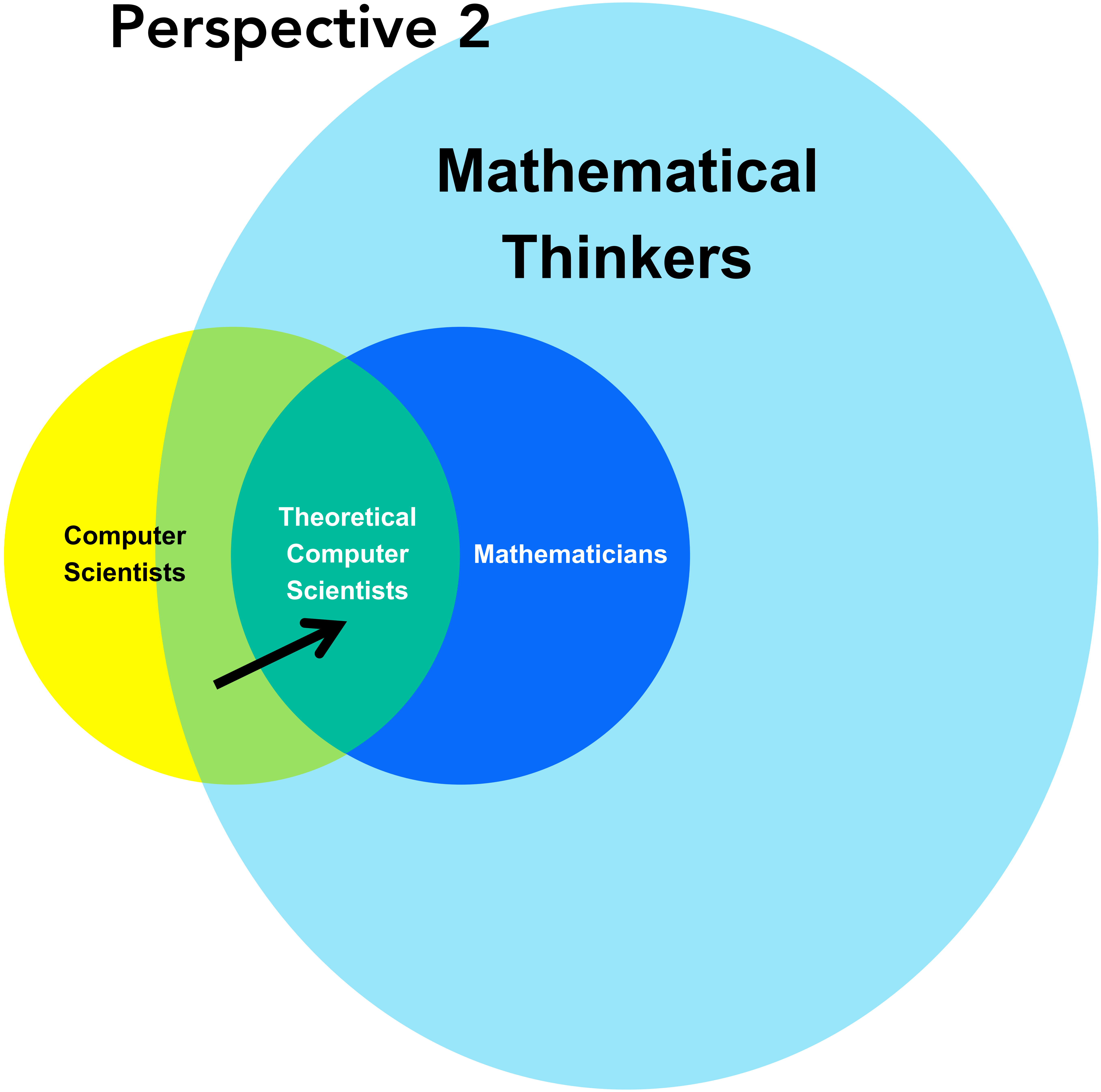
all computer scientists should be here

# Perspective 2



# Perspective 2

**Goal #2 of 15-251 (maybe)**



# Perspective 3

Mathematics is like… cilantro.

There are **5** kinds of people when it comes to cilantro.

# Perspective 3

1. *People who do not know what cilantro is.*

# Perspective 3

1. *People who do not know what cilantro is.*

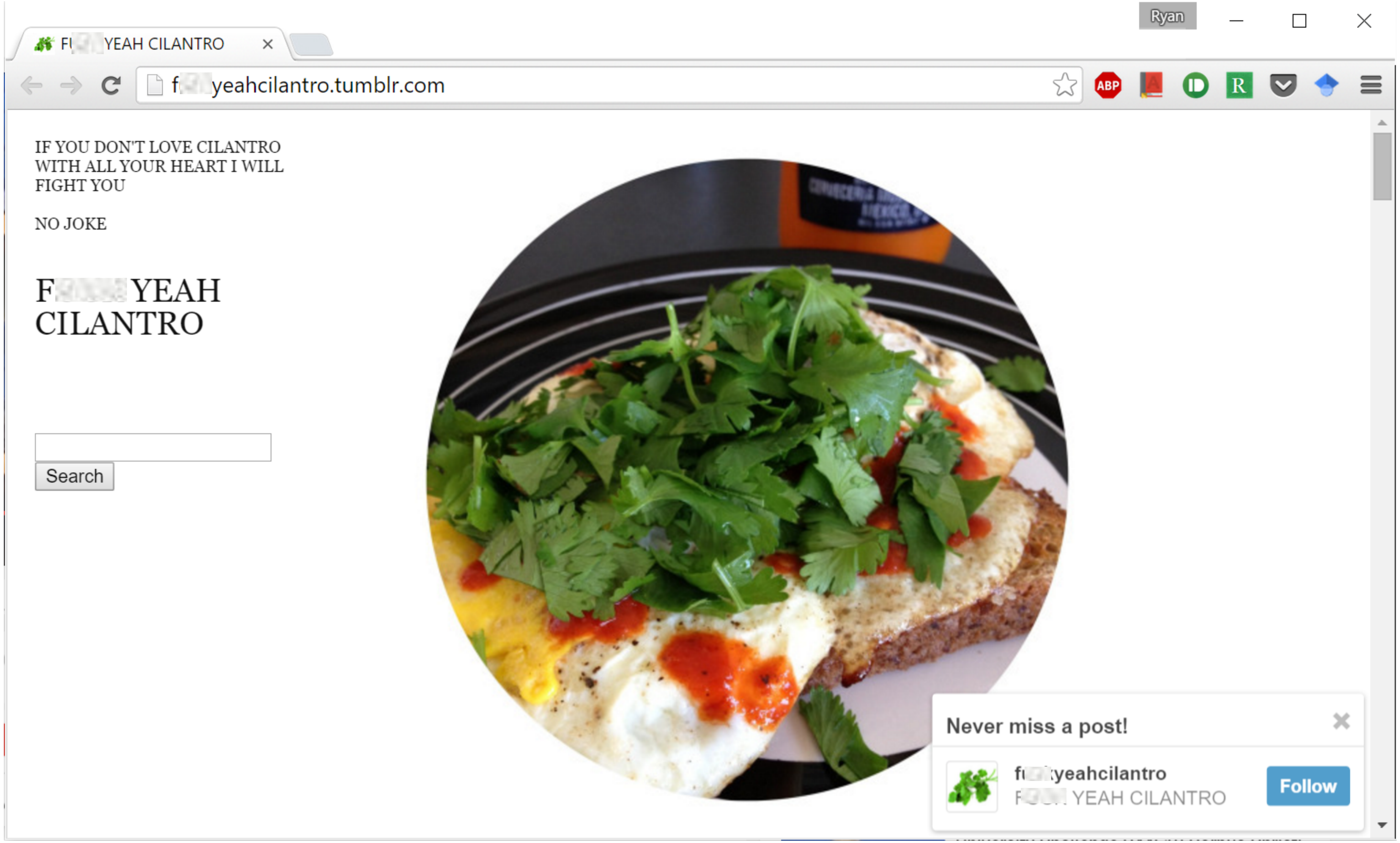
# Perspective 3

1. *People who do not know what cilantro is.*

- Coriandrum sativum
- Coriander (leaves)
- 香菜
- धनिया
- 고수
- گشنیز
- الكزبرة
- ngò
- φύλλα κόλιανδρου
- கொத்தமல்லி
- ধনে
- kişniş
- кинза
- כוסברה

# Perspective 3

2. *People who love cilantro.*



# Perspective 3

3. *People who think cilantro is fine.*

**Goal**: have everyone at least in this category by the end of the course.

# Perspective 3

4. *People who don’t like cilantro.*

Still gotta try it.

Hope you move to previous category.

If not, hope you can eat cilantro if necessary.

# Perspective 3

5. *People with a genetic condition that makes cilantro taste like soap.*

Is this true?

https://www.youtube.com/watch?v=Xs9aGVUZ3YA