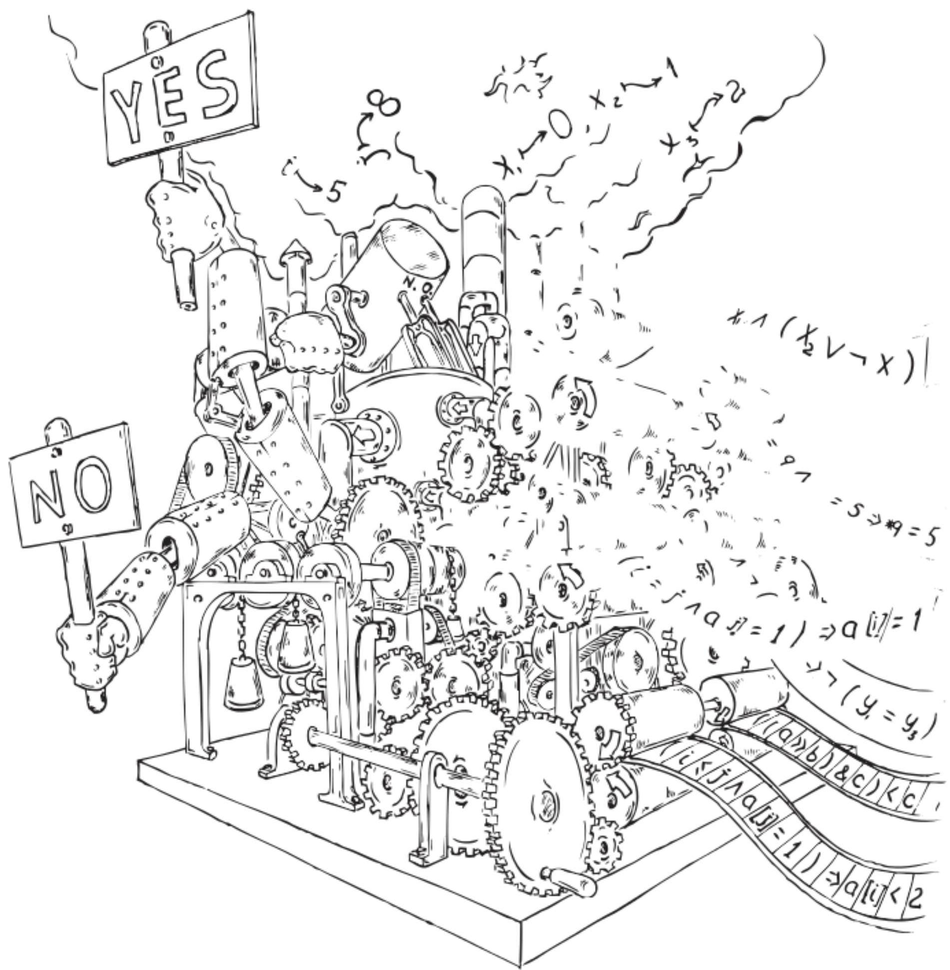
Theoretical Foundations of Computer Science



August 9, 2017

Motivational Quote of the Day

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

~~Edsger Dijkstra~~

- *Michael Fellows*

PART 1:
What is (theoretical) computer science?
How was it born?

PART 2:
Uncomputable problems

PART 1:
What is computer science?
How was it born?

What is **computer science**?

Is it:

“Writing (Python) programs
that do certain tasks.”

What is **theoretical computer science**?

What is computer science?

Is it branch of:

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

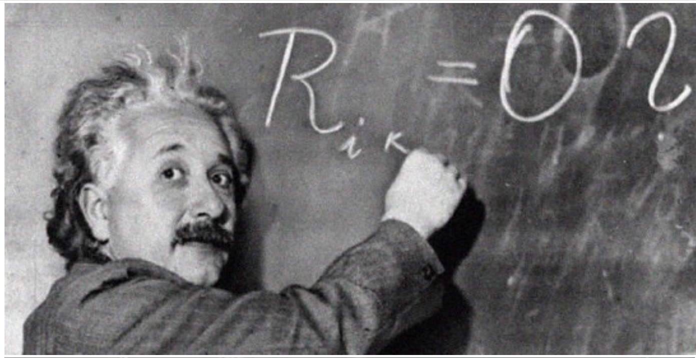
Physics

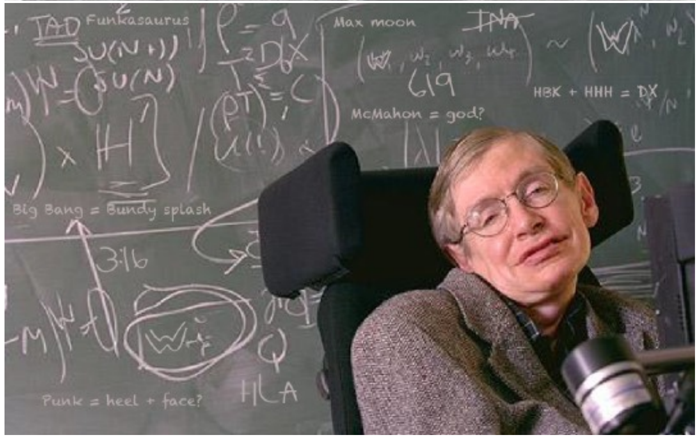
Theoretical physics

- come up with mathematical models

  Nature's language is mathematics

- derive the logical consequences



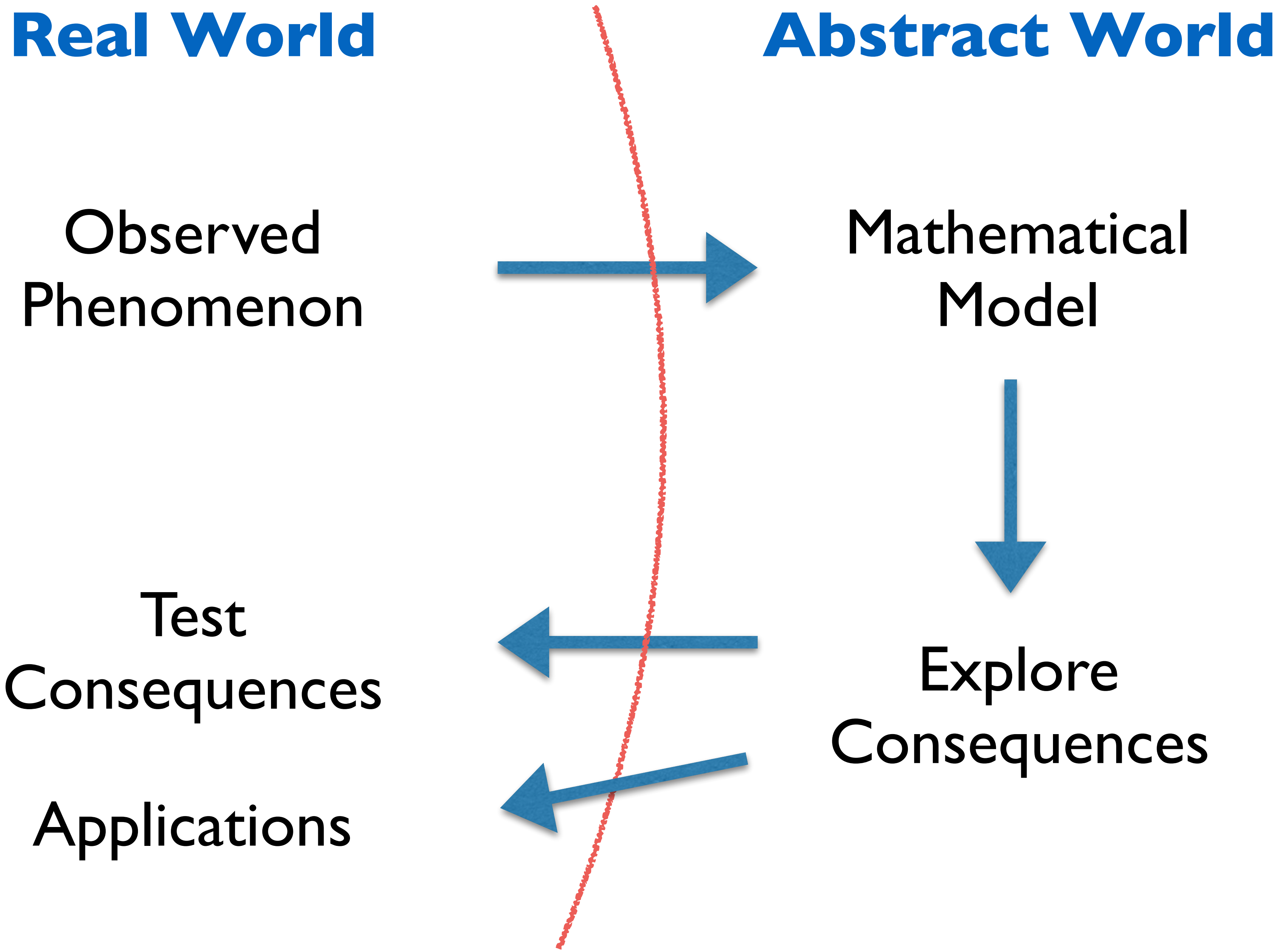


Experimental physics

- make observations about the universe
- test mathematical models with experiments

Applications/Engineering

The role of theoretical physics



Theoretical Physics

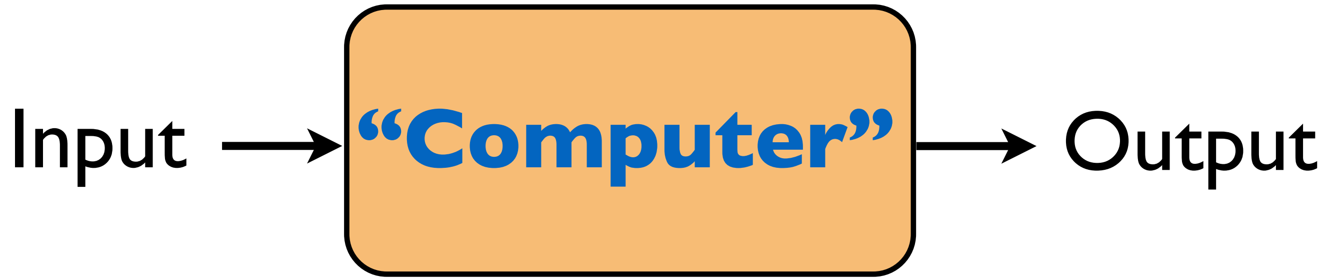
- science?
- engineering?
- math?
- philosophy?
- sports?

Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: rigorous description of how the data is manipulated.



Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: rigorous description of how the data is manipulated.

Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: rigorous description of how the data is manipulated.

Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: rigorous description of how the data is manipulated.

Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: rigorous description of how the data is manipulated.

The computational lens

Computational physics

Computational biology

Computational chemistry

Computational neuroscience

Computational economics

Computational finance

Computational linguistics

Computational statistics

...

The role of theoretical computer science

Build a mathematical model for computation.

Explore the logical consequences.
Gain insight about computation.

http://youtu.be/pTeZP-XfuKI

https://goo.gl/gGkpMv

http://youtu.be/J4TkHuTmHsg

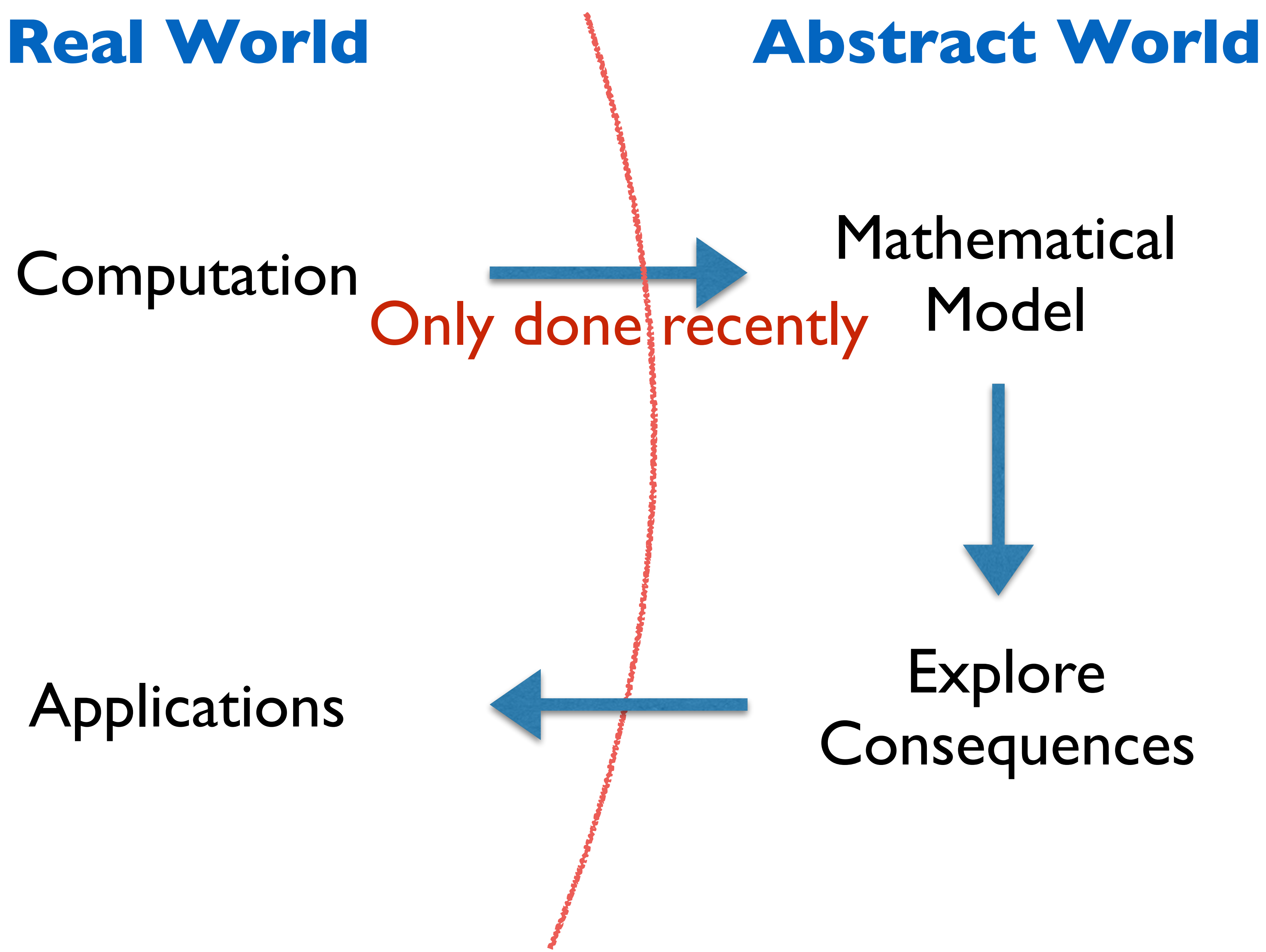
Look for interesting applications.

CMU undergrad

CMU Prof.

OK, we don't have everybody

The role of theoretical computer science



Simple examples of computation

$$
\begin{array}{r}
5127 \\
\times\ 4265 \\
\hline
25635 \\
307620 \\
1025400 \\
20508000 \\
\hline
21866655
\end{array}
$$

Doing computation by following a simple algorithm.

Simple examples of computation

Euclid's algorithm (~ 300BC):

```
def gcd(a, b):
    while (b != 0):
        t = b
        b = a % b
        a = t
    return a
```

We have been using algorithms for thousands of years.

Formalizing computation

We have been using algorithms for thousands of years.

Algorithm/Computation was only formalized in the 20th century!

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 logical expression?

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

Gödel (1934):

Discusses some ideas for mathematical definitions of computation. But not confident what is a good definition.

Church (1936):

Invents lambda calculus.
Claims it should be the definition of an "algorithm".

Gödel, Post (1936):

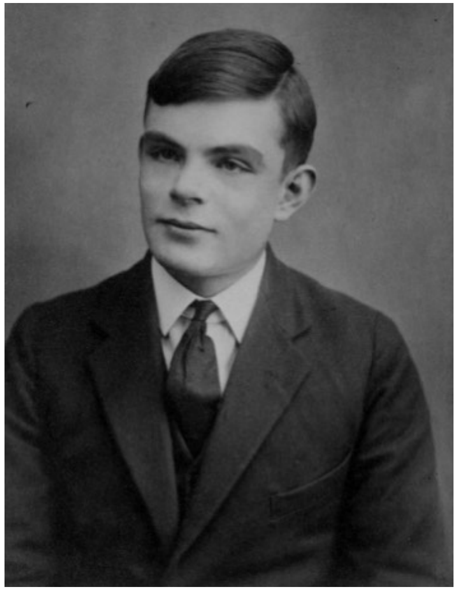
Arguments that Church's claim is not justified.

Meanwhile… in New Jersey… a certain British grad student, unaware of all these debates…

2 of Hilbert's Problems

Alan Turing (1936, age 22):

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



Gödel, Kleene, and even Church:

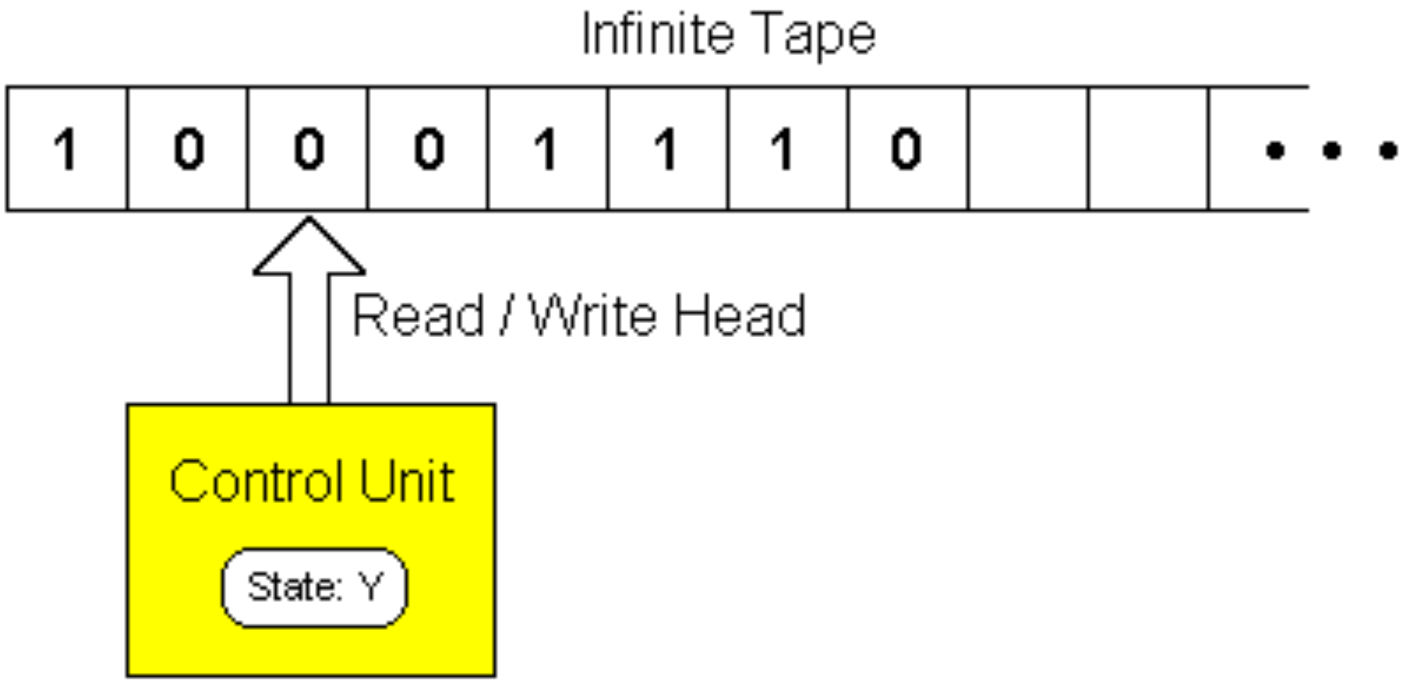
"Umm. Yeah. He nailed it. Game over. "Algorithm" defined."

Turing (1937):

TMs $\equiv$ lambda calculus

Formalization of computation: Turing Machine

Turing Machine:



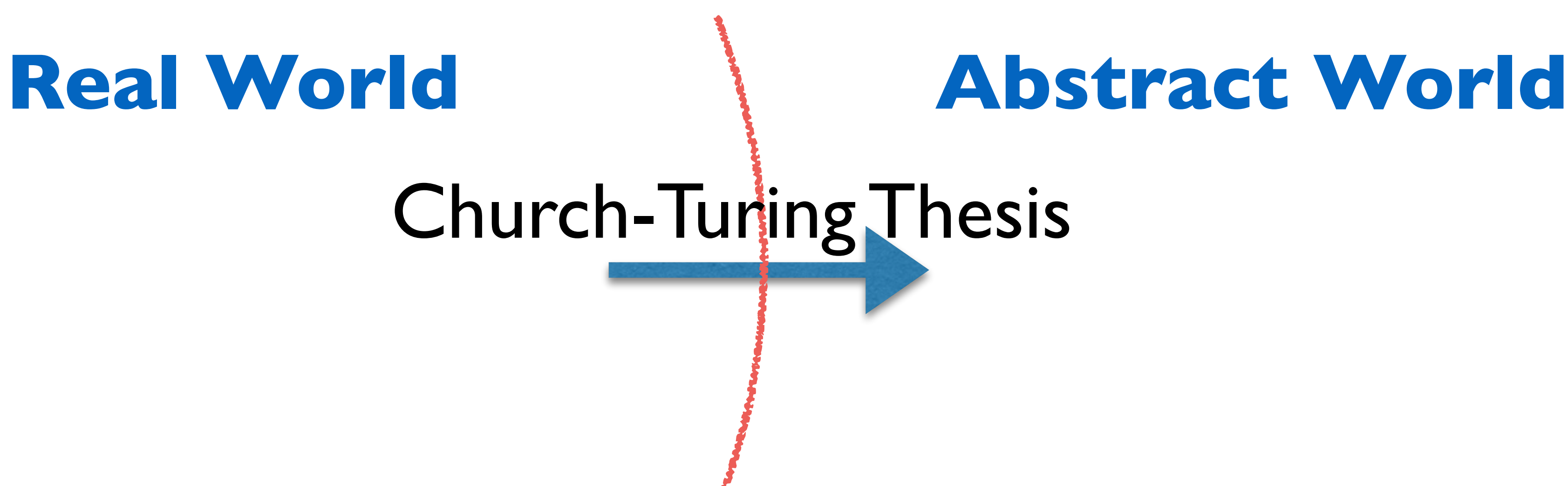
Church-Turing Thesis

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 a physical device, can be solved by a Turing Machine.



Back to 2 of Hilbert's Problems

Hilbert's 10th problem (1900)

Is there an **algorithm** (a 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 an **algorithm** (a TM) to determine the validity of a given logical expression?

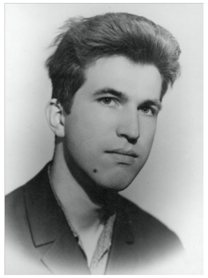
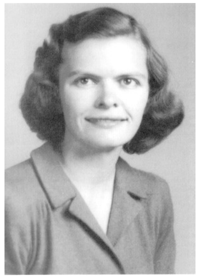
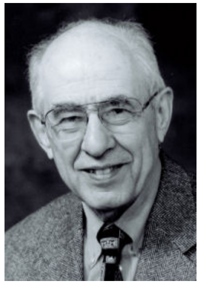
e.g. $$\neg\exists x, y, z, n \in \mathbb{N} : (n \geq 3) \wedge (x^n + y^n = z^n)$$

(Mechanization of mathematics)

Back to 2 of Hilbert's Problems

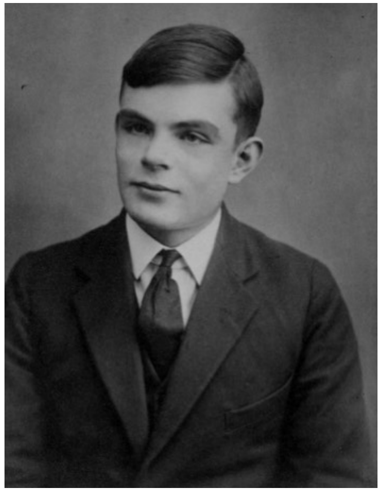
Hilbert's 10th problem (1900)

Matiyasevich-Robinson-Davis-Putnam (1970):

There is no algorithm to solve this problem.

Entscheidungsproblem (1928)

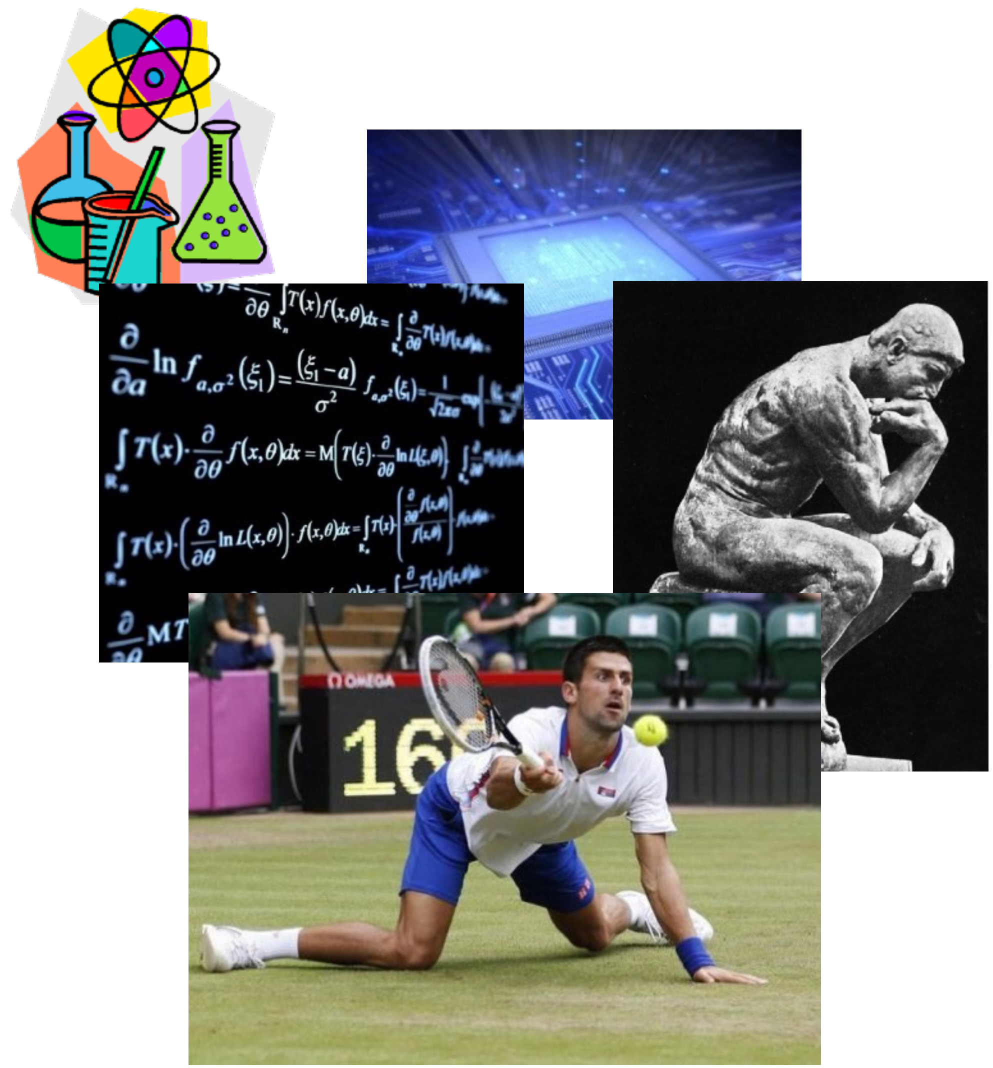


Turing (1936):

There is no algorithm to solve this problem.

Computer Science

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



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:

- trying to come up with efficient algorithms (algorithm designers)

- trying to show no efficient algorithm exists (complexity theorists)

Computational Complexity

2 camps:

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

multiplying two integers
factoring integers
sorting a list
protein structure prediction
simulation of quantum systems
computing Nash Equilibria of games

PART 2:
Uncomputable problems

Working as a TA for 15-112

We need to write an autograder for

isPrime

student submission

isPrime

the correct program

isPrime

Do they return True on exactly the same inputs?

Working as a TA for 15-112

We need to write an autograder for

isPrime

Kosbie's version

Student submission

returns True on exactly same inputs?

True

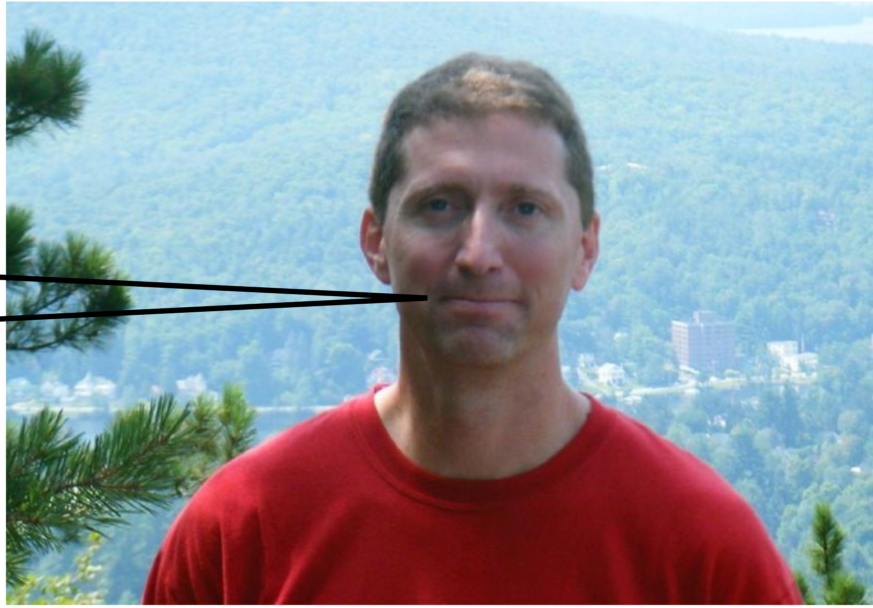
or

False

Working as a TA for 15-112

A “simpler” problem

Write an “autograder”
that checks if a given program
goes into an infinite loop.



Working as a TA for 15-112

Halting Problem

Inputs: A Python program source code. 
An *input* to the program. x

Output: True if the program halts for the given *input*.
False otherwise.

Theorem: The **halting problem** is uncomputable.

"Proof"

Assume, for the sake of contradiction, such a program exists:

def **halt**(*program*, *inputToProgram*):
  ***# program* and *inputToProgram* are both strings**

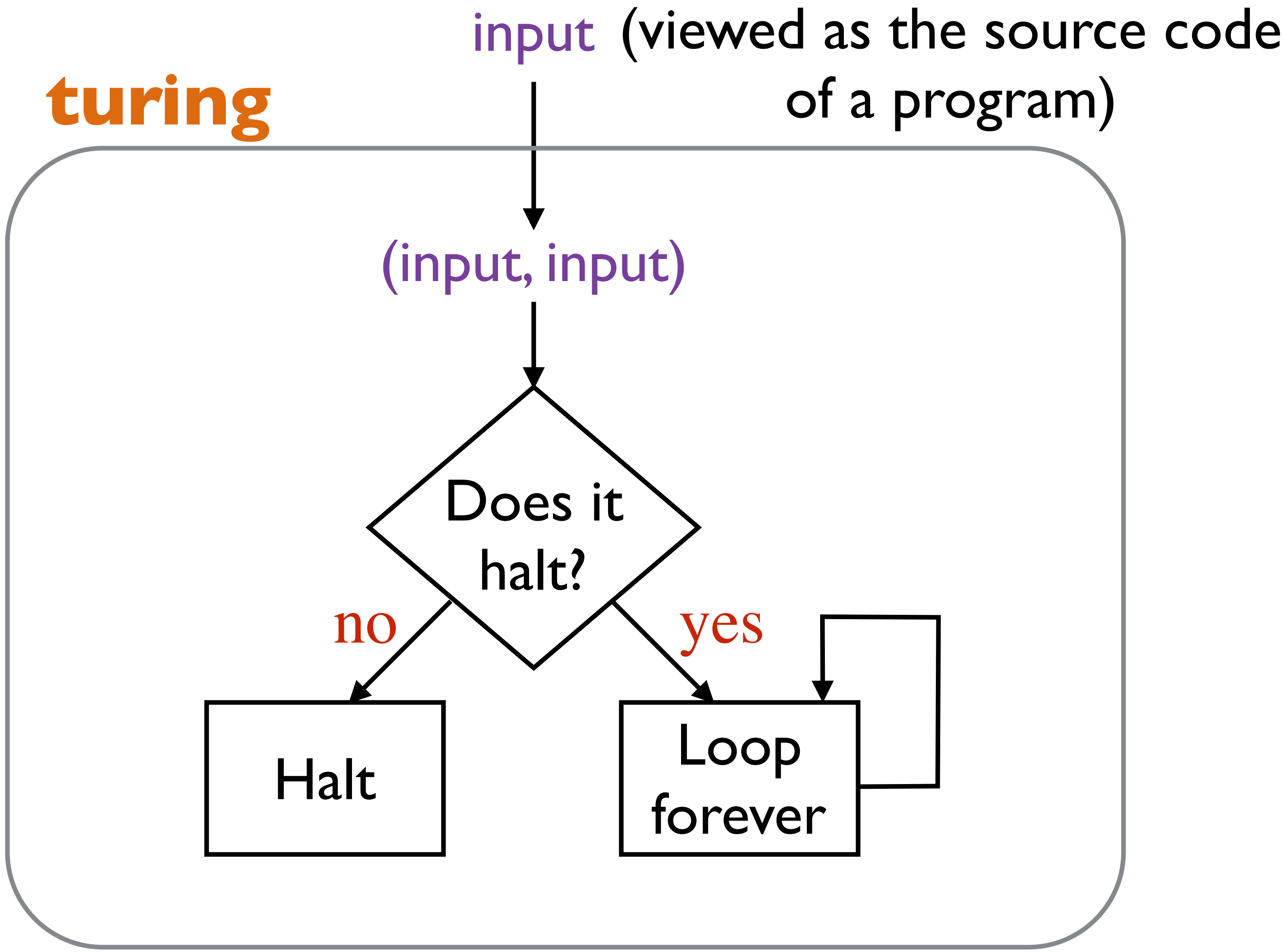
“Proof”

Assume, for the sake of contradiction, such a program exists:

```
def halt(program, inputToProgram):
    # program and inputToProgram are both strings


def turing(program):
    if (halt(program, program)):
        while True:
            pass   # i.e. do nothing
    return False
```

“Proof”


input (viewed as the source code of a program)
turing
(input, input)
Does it halt?
no
yes
Halt
Loop forever

"Proof"

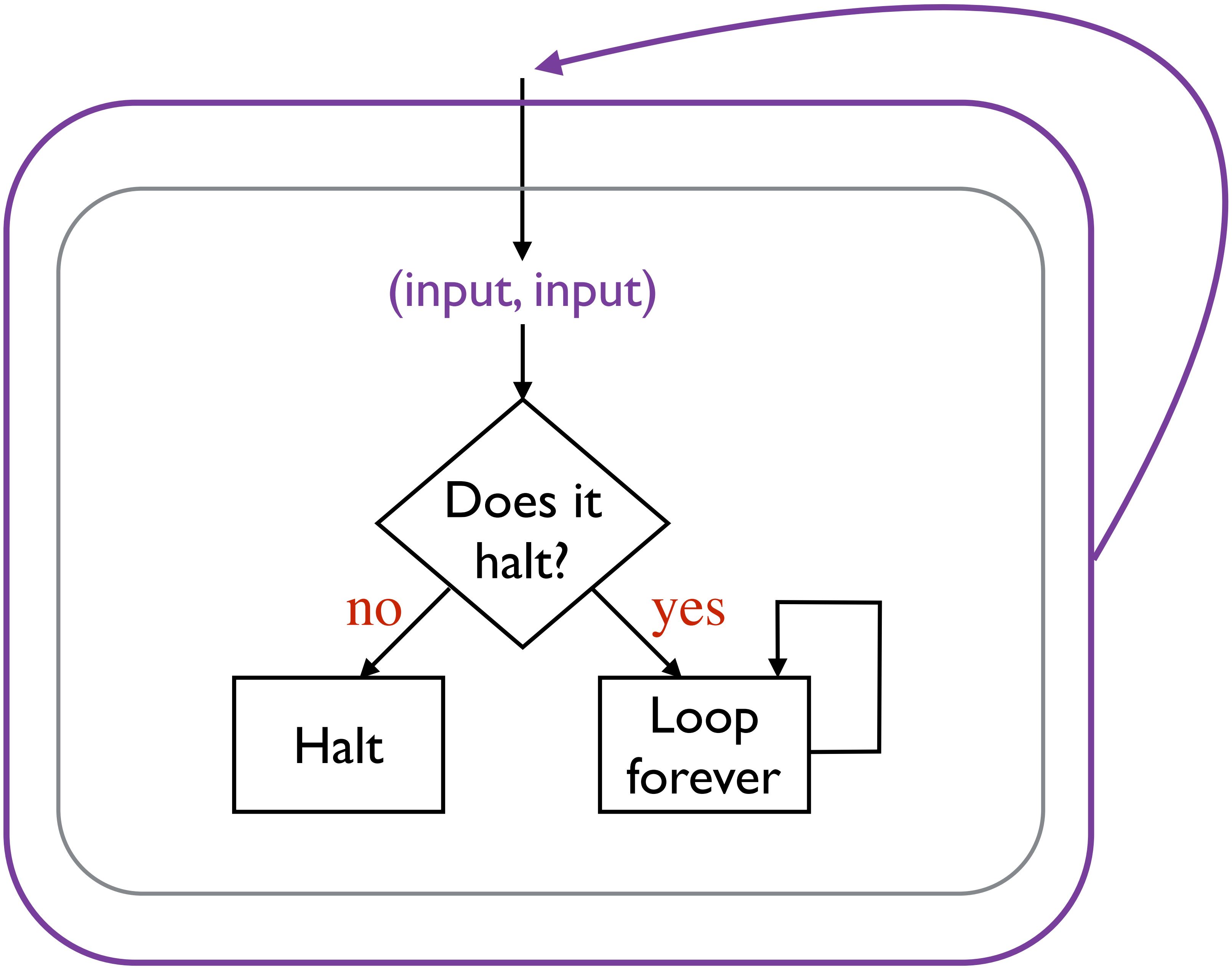
Assume, for the sake of contradiction, such a program exists:

```
def halt(program, inputToProgram):
    # program and inputToProgram are both strings


def turing(program):
    if (halt(program, program)):
        while True:
            pass   # i.e. do nothing
    return False
```

What happens when you call **turing**(*turing*) ?

“Proof”



"Proof"

Assume, for the sake of contradiction, such a program exists:

```
def halt(program, inputToProgram):
    # program and inputToProgram are both strings


def turing(program):
    if (halt(program, program)):
        while True:
            pass   # i.e. do nothing
    return False
```

What happens when you call **turing**(*turing*) ?

if **halt**(*turing*, *turing*) is True ----> **turing**(*turing*) doesn't halt

if **halt**(*turing*, *turing*) is False ----> **turing**(*turing*) halts

"Proof"

Assume, for the sake of contradiction, such a program exists:

```
def halt(program, inputToProgram):
    # program and inputToProgram are both strings


def turing(program):
    if (halt(program, program)):
        while True:
            pass   # i.e. do nothing
    return False
```

What happens when you call **turing**(*turing*) ?

if **halt**(*turing*, *turing*) is True ----> **turing**(*turing*) doesn't halt

if **halt**(*turing*, *turing*) is False ----> **turing**(*turing*) halts □

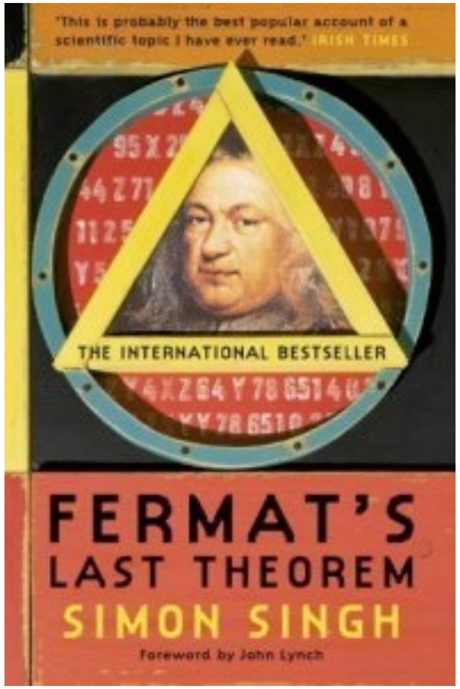
CONTRADICTION

So what?

- No guaranteed autograder program. ☹

- Consider the following program:

```
def fermat():
    t = 3
    while (True):
        for n in range(3, t+1):
            for x in range(1, t+1):
                for y in range(1, t+1):
                    for z in range(1, t+1):
                        if (x**n + y**n == z**n):  return (x, y, z, n)
        t += 1
```



<u>Question</u>: Does this program halt?

So what?

- Consider the following program (written in MAPLE):

```
numberToTest := 2;
flag := 1;
while flag = 1 do
    flag := 0;
    numberToTest := numberToTest + 2;
    for p from 2 to numberToTest do
        if isPrime(p) and isPrime(numberToTest–p) then
            flag := 1;
            break;      #exits the for loop
        end if
    end for
end do
```

Goldbach Conjecture

Question: Does this program halt?

So what?

- Reductions to other problems
  imply that those problems are uncomputable as well.

Entscheidungsproblem

Is there an algorithm to determine the validity of a given logical expression?

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

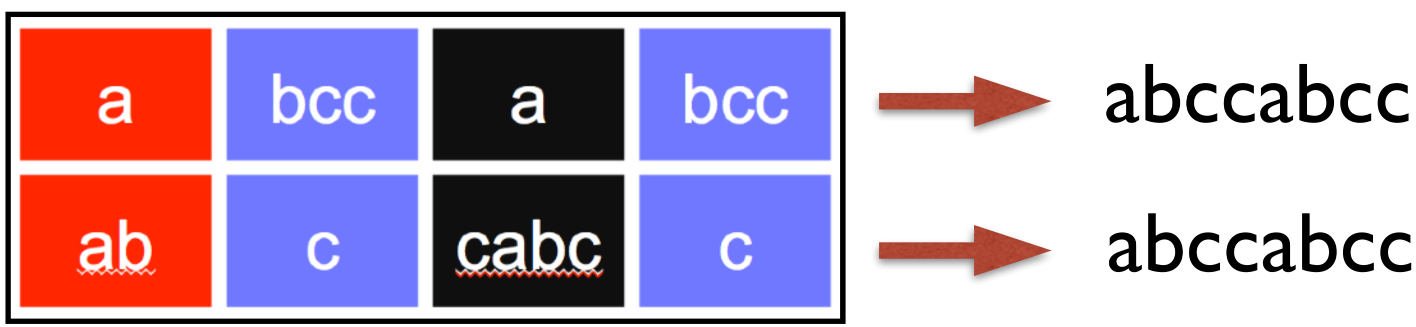
Hilbert's 10th Problem

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

So what?

Input: A finite collection of "dominoes" having strings written on each half.

Output: **True** if it is possible to match the strings.



Uncomputable!

Proved in 1946 by Post.

So what?

Input: A finite collection of "Wang Tiles" (squares) with colors on the edges.

Output: **True** if it is possible to make an infinite grid from copies of the given squares, where touching sides must color-match.

Uncomputable!

Proved in 1966 by Berger.

So what?

Input: Two 21x21 matrices of integers A and B.

Output: **True** iff it is possible to multiply A and B together (multiple times in any order) to get to the 0 matrix.

Uncomputable!

Proved in 2007 by Halava, Harju, Hirvensalo.

So what?

Different laws of physics ----->

Different computational devices ----->

Every problem computable (???)

Can you come up with sensible laws of physics such that the Halting Problem becomes computable?

That was about the basic question on whether every problem is computable.

Some other interesting questions in TCS

Time vs Space

If a problem has a space-efficient solution does it also have a time-efficient solution?

Power of randomness

Can every randomized algorithm be derandomized efficiently?

Power of quantum information

Can we use quantum properties of matter to build faster computers?

P vs **NP**