15-251
Great Ideas in
Theoretical Computer Science

Lecture 6:
Church-Turing Thesis + Decidability

February 1st, 2018

Last Time

A Turing machine (TM) M is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{rej})$$

where

- Q is a finite set (which we call the set of states);
- Σ is a finite set with $\sqcup \notin \Sigma$
  (which we call the input alphabet);
- Γ is a finite set with $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$
  (which we call the tape alphabet);
- δ is a function of the form $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$
  (which we call the transition function);
- $q_0 \in Q$ (which we call the start state);
- $q_{\text{acc}} \in Q$ (which we call the accept state);
- $q_{\text{rej}} \in Q$, $q_{\text{rej}} \neq q_{\text{acc}}$ (which we call the reject state);

Last Time

Definition: A TM is called a *decider* if it halts on all inputs.

Definition: A language L is called *decidable* (*computable*) if $L = L(M)$ for some decider TM M.

Theorem: Any language that can be computed in Python, C, Java, etc. can be decided by a TM.

QUESTIONS

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

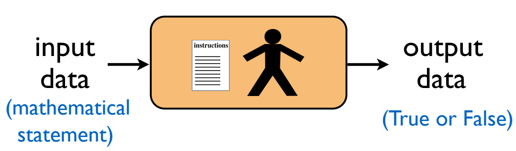
Entscheidungsproblem (1928)

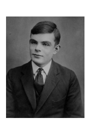
Is there a finitary procedure to determine the validity of a given logical expression?

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

(Mechanization of mathematics)

The quest for the right definition



"Alright, let's define this thing mathematically."

Turing's thinking

Turing's legacy

The beauty of his definition:

Simplicity

1. simplicity

a reasonable definiton of computation

↓

strong enough to capture computation the way TMs do.

Generality

2. “clearly” captured what a human does given a set of instructions.

Church-Turing Thesis

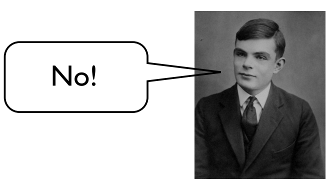
What else did Turing do in his paper?

Entscheidungsproblem (1928)

Is there a finitary procedure to determine the validity of a given logical expression?

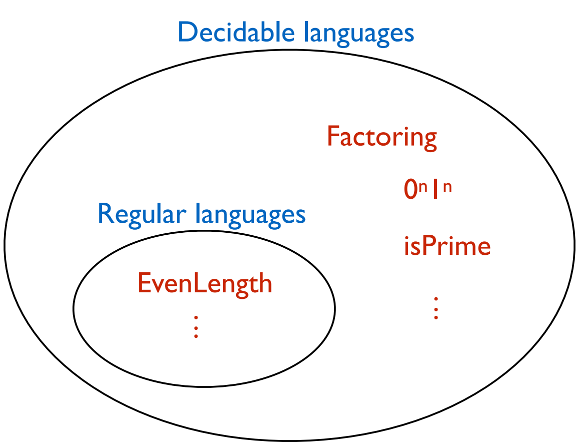
e.g. $\neg\exists x, y, z, n \in \mathbb{N} : (n \geq 3) \wedge (x^n + y^n = z^n)$

(Mechanization of mathematics)



Entscheidungsproblem

Are there others?



What else did Turing do in his paper?

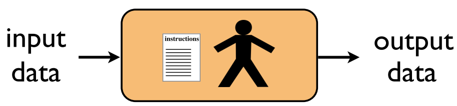
Universal Machine
(one machine to rule them all)

+ | **isPrime** | **Sorting** | **DFA “|x| even?”**

Do we really need a separate machine for each task?

What else did Turing do in his paper?

Universal Machine
(one machine to rule them all)



A human is a universal machine:

What else did Turing do in his paper?

Universal Machine
(one machine to rule them all)

+ | **isPrime** | **Sorting** | **DFA “|x| even?”**

All can be encoded!
(e.g. think source code)

What else did Turing do in his paper?

Universal Machine

(one machine to rule them all)

We could use:

$$\langle M \rangle =$$

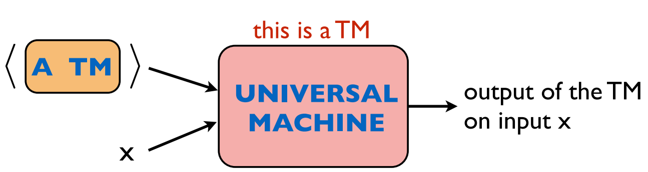
```
def foo(input):
  i = 0
  STATE 0:
    letter = input[i];
    switch(letter):
      case 'a':  input[i] = ' '; i++; go to STATE a;
      case 'b':  input[i] = ' '; i++; go to STATE b;
      case ' ':  input[i] = ' '; i++; go to STATE rej;
  STATE a:
    letter = input[i];
    switch(letter):
      case 'a':  input[i] = ' '; i--;  go to STATE acc;
      case 'b':  input[i] = ' '; i--;  go to STATE rej;
      case ' ':  input[i] = ' '; i--;  go to STATE rej;
```

Code is data!

What else did Turing do in his paper?

Universal Machine

(one machine to rule them all)

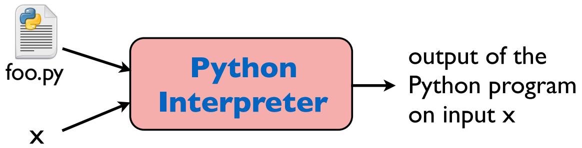


Could you write a Python function that does this?

What else did Turing do in his paper?

Universal Machine
(one machine to rule them all)

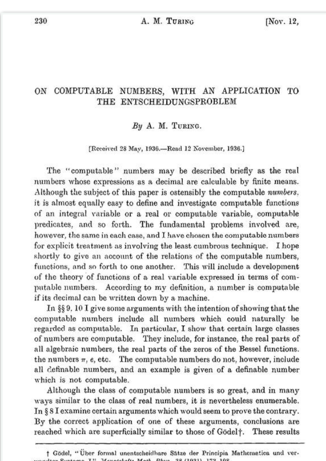
This is exactly what an **interpreter** does.



Code is data!

The **positive** side

The **negative** side



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

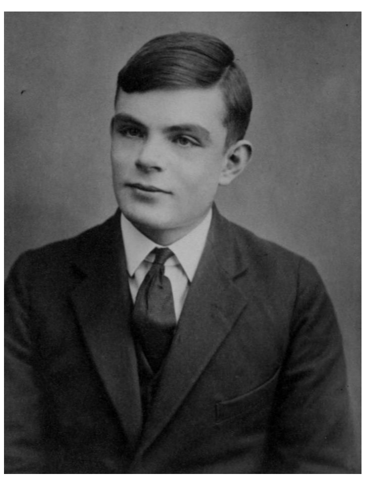
The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

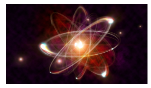
1936



1912 - 1954

Perhaps Turing and others weren't ambitious enough!

Solvable by any physical process



||| ???

Solvable by a TM

Physical Church-Turing Thesis

Physical Church-Turing Thesis

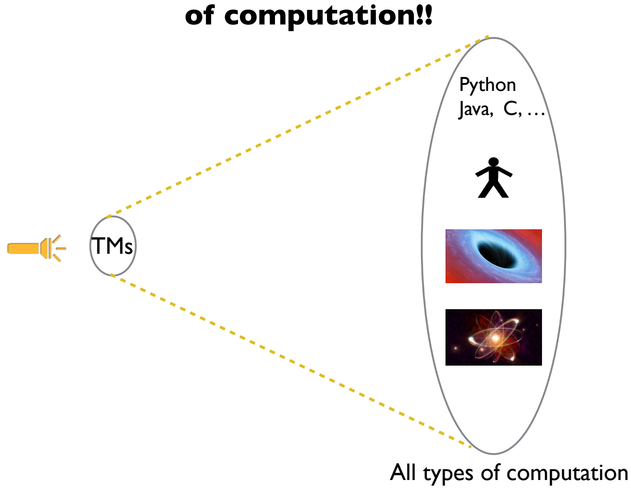
What can be computed in this universe, by any physical process or device, can by computed by a (rand.) TM.

Why should we expect this to be true?

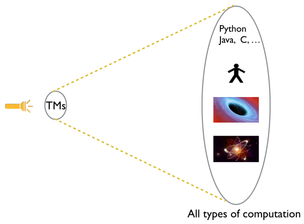
Strong Physical Church-Turing Thesis

What can be computed **efficiently** in this universe, by any physical process or device, can by computed **efficiently** by a **Q**TM.

This is the grand unification/simplification of computation!!



This is the grand unification/simplification of computation!!



Complex things can be explained by simple rules.

Implications

1. Studying the power and limits of TMs

→ Studying the power and limits of our universe

(Can you come up with laws of physics that would allow it to compute any problem?)

2. Computation in its full generality is everywhere. Even in extremely simple systems!

(What is the simplest universe you can create that has the same computational capacity of our universe?)

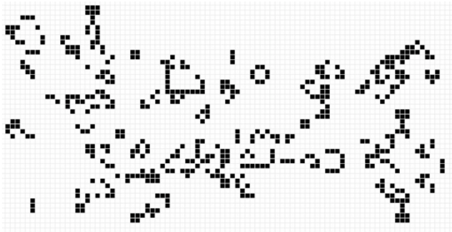
3. The universe may be a simulation. (a philosophical musing)

What is the simplest universe you can create that has the same computational capacity of our universe?

Conway's Game of Life

Imagine an infinite 2D grid.

Each cell can be dead or alive.



Laws of physics

Loneliness: live cell with fewer than 2 neigbors dies.

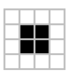
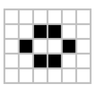
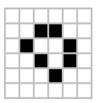
Overcrowding: live cell with more than 3 neighbors dies.

Procreation: dead cell with exactly 3 neighbors gets born.

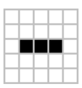
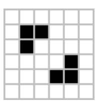
Conway's Game of Life

Some Patterns

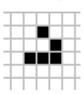
Stable

Periodic

Moving



Conway's Game of Life

Can a TM simulate any instance of Game of Life?

Can Game of Life simulate any TM?

Can Game of Life simulate Game of Life?

That's all for the Church-Turing Thesis.

Let's talk **decidability**.

Languages involving encodings of machines

Code is data!

There are many interesting problems
where the input data is code.

Working as a TA for 15-112

Autograder program

student submission

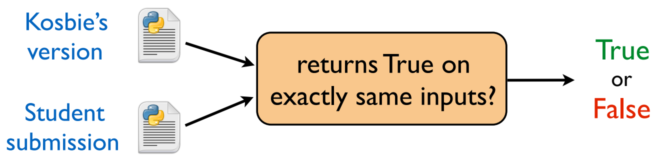
isPrime

the correct program

isPrime

Do they return True on exactly the same inputs?

Working as a TA for 15-112



Does such a program exist?

i.e., can we solve the following:

$$\text{EQ} = \{\langle M_1, M_2\rangle : M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$$

Working as a TA for 15-112

Similar but simpler looking languages:

$$\text{ACCEPTS} = \{\langle M, x\rangle : M \text{ is a TM and } x \in \Sigma^* \text{ s.t. } x \in L(M)\}$$

$$\text{EMPTY} = \{\langle M\rangle : M \text{ is a TM s.t. } L(M) = \emptyset\}$$

Poll

Which ones do you think are decidable?

$$\text{ACCEPTS}_{\text{DFA}} = \{\langle D, x\rangle : D \text{ is a DFA and } x \in \Sigma^* \text{ s.t. } x \in L(D)\}$$

$$\text{SELF-ACCEPTS}_{\text{DFA}} = \{\langle D\rangle : D \text{ is a DFA s.t. } \langle D\rangle \in L(D)\}$$

$$\text{EMPTY}_{\text{DFA}} = \{\langle D\rangle : D \text{ is a DFA s.t. } L(D) = \emptyset\}$$

$$\text{EQ}_{\text{DFA}} = \{\langle D_1, D_2\rangle : D_1 \text{ and } D_2 \text{ are DFAs s.t. } L(D_1) = L(D_2)\}$$

$\mathrm{ACCEPTS_{DFA}}$

$\mathrm{ACCEPTS_{DFA}} = \{\langle D, x\rangle : D \text{ is a DFA and } x \in \Sigma^* \text{ s.t. } x \in L(D)\}$

$\mathrm{SELF\text{-}ACCEPTS_{DFA}}$

$\mathrm{SELF\text{-}ACCEPTS_{DFA}} = \{\langle D\rangle : D \text{ is a DFA s.t. } \langle D\rangle \in L(D)\}$

$\mathrm{EMPTY_{DFA}}$

$\mathrm{EMPTY_{DFA}} = \{\langle D\rangle : D \text{ is a DFA s.t. } L(D) = \emptyset\}$

EQ_{DFA}

$\text{EQ}_{\text{DFA}} = \{\langle D_1, D_2 \rangle : D_1 \text{ and } D_2 \text{ are DFAs s.t. } L(D_1) = L(D_2)\}$

Reduction