

CS70 — SPRING 2026

LECTURE 13: MAR. 3

# Last Lecture

- Sets  $A, B$  have same cardinality  $\Leftrightarrow \exists$  bijection  $f: A \rightarrow B$
- Set  $A$  is countable  $\Leftrightarrow \exists$  bijection between  $A$  and (some subset of)  $\mathbb{N}$
- Some countable sets (proved by enumeration):
  - all finite sets
  - $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{Z} \times \mathbb{Z}, \{0, 1\}^*, \mathbb{N}[x], \dots$
- Some uncountable sets (proved by diagonalization):
  - $\mathbb{R}, [0, 1], \mathcal{P}(\mathbb{N}),$  Cantor set,  $\dots$

# Today

**Q:** Is there any (mathematical/technical) problem that computers cannot solve?

**A:** Yes! [Turing: Halting Problem]

**Q':** Is there any true mathematical theorem that cannot be proved?

**A':** Yes! [Gödel: Incompleteness Theorem]

---

## Why?

- Great application of Diagonalization
- Core philosophy of Computer Science

# Timeline

~1900: Hilbert's Program: formalize all of Mathematics via axioms & inference rules



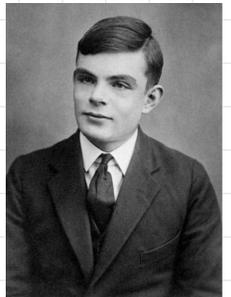
early 1900s: Russell & others: logical paradoxes



1931: Gödel: Not possible for arithmetic – must exist true statements that aren't provable



1936: Turing: There are mathematical problems that no computer can solve

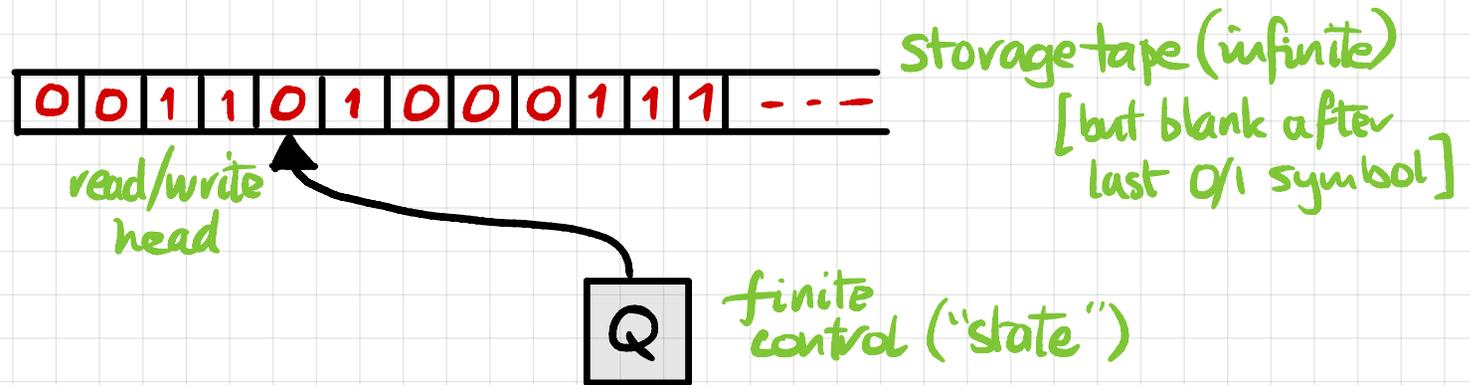


## Turing's Theorem Ingredients:

1. A model of computation  
(to define what's possible/impossible)
2. Programs = Data (= binary strings)
3. Self-reference ( $\approx$  diagonalization)



# 1. Model of Computation: Turing Machine



transition function

$$\delta: \underset{\text{state}}{Q} \times \underset{\text{tape symbol}}{\{0, 1\}} \longrightarrow \underset{\text{move}}{Q} \times \{0, 1\} \times \{L, R, S\}$$

Primitive, but equivalent to (i.e., can simulate) any other model, e.g.:

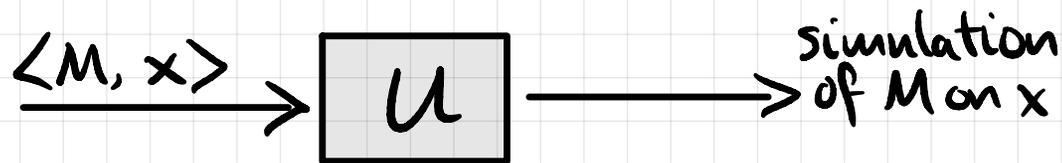
- Random Access Machine
- High Level Programming Language
- Quantum Computer

## 2. Programs = Data

Turing machine/Computer program is just a finite string of bits

⇒ it can be fed to another program as input !

Universal TM [Turing 1936]



Note: Turing invented programmable computers long before they were designed or built !

[Manchester Baby 1948, Mark 1 1949

Grace Hopper Compilers, 1952 ]

### 3. Self-Reference

Logical paradoxes [B. Russell etc.]

- "This sentence is false"
- The barber shaves those - and only those - men who do not shave themselves (where everyone is clean shaven)
- $S =$  the set of all sets that do not contain themselves
- "Yields a falsehood when appended to its own quotation" yields a falsehood when appended to its own quotation"

## Halting & Non-Halting Programs

program count-by-threes ( $x$ )

count = 0

while count  $\neq$   $x$

count = count + 3

output "Done"

- if  $x$  is a non-negative multiple of 3, then count-by-threes( $x$ ) halts (& outputs "Done")
- otherwise, count-by-threes doesn't halt/loops forever

Halting Problem: Distinguish these two behaviors

# The Halting Problem [Turing 1936]

Input: program (TM)  $P$  & input  $x$  for  $P$

Question: does  $P$  halt on input  $x$ ?

Turing's Theorem: There is no computer program (TM) that solves the Halting Problem on all inputs

Note: We say that the Halting Problem is uncomputable or undecidable

# Proof of Turing's Theorem:

# The "Easy" Halting Problem

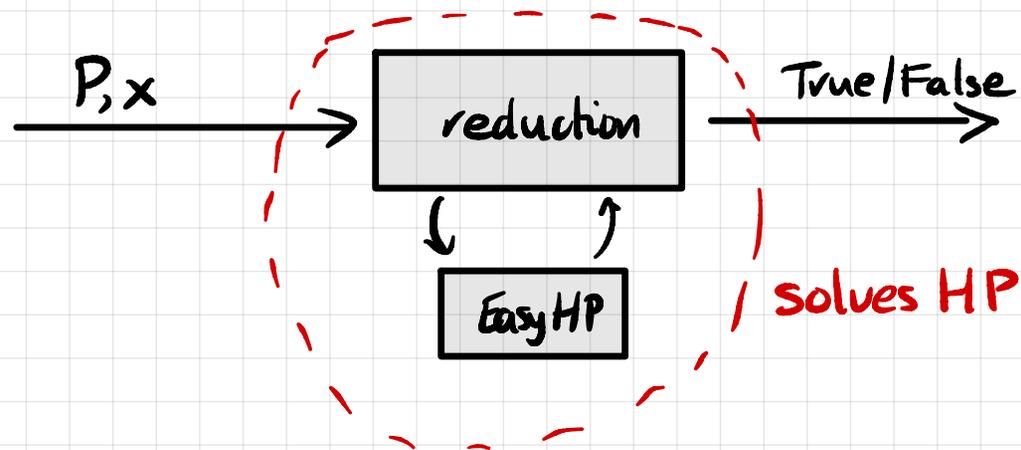
Input: Program  $P$

Question: Does  $P$  halt on input  $0$ ?

Theorem: "Easy" HP is undecidable

Proof:

We just used a reduction from HP to EasyHP:



The reduction shows that if EasyHP is solvable then  
so is HP ~~X~~

So EasyHP is not solvable

## Program Equivalence

Input: Two programs  $P_1, P_2$

Question: Do  $P_1, P_2$  behave the same (halt/not halt) on all inputs?

Theorem: Program Equiv. is undecidable

Proof:

## Other Undecidable Problems

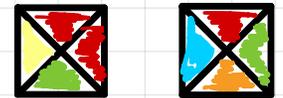
1. Most questions about software behavior, e.g., :

Input: Program  $P$ , Specification  $S$

Question: Does  $P$  satisfy  $S$ ?

2. Wang Tiling Problem

Input: Finite set of Wang tiles, e.g.,



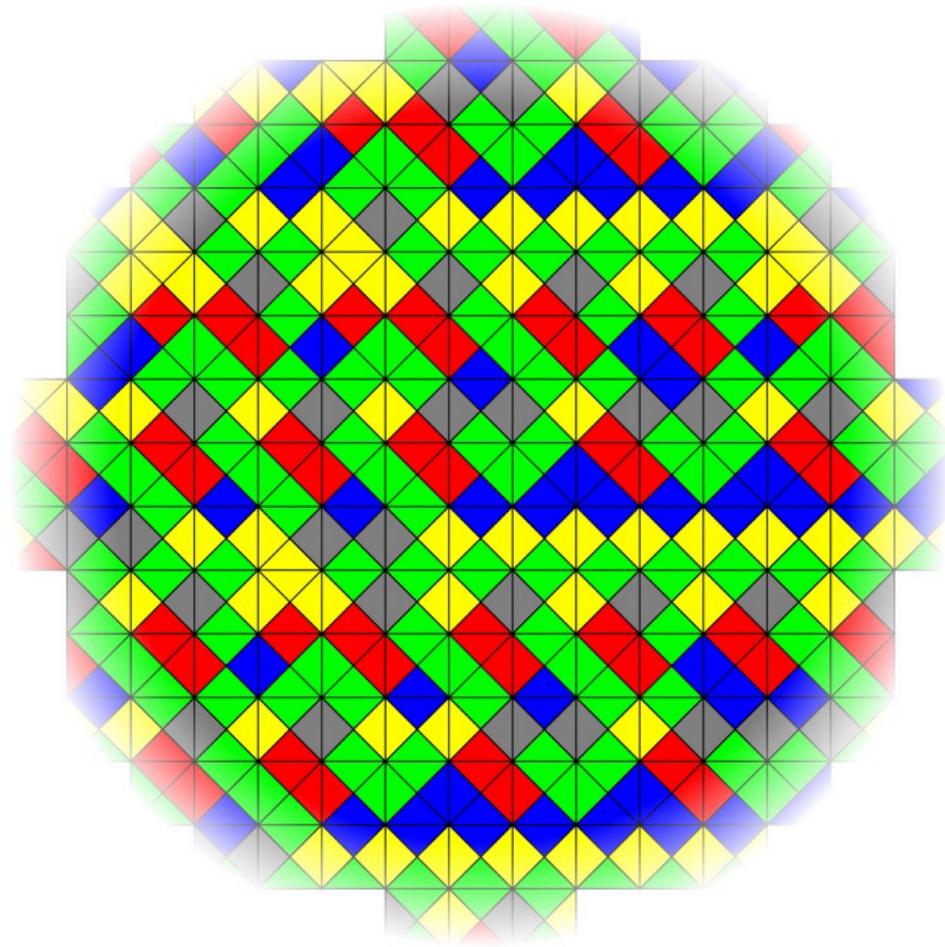
Question: Can we tile the infinite plane with these tiles?

3. Matrix Mortality Problem

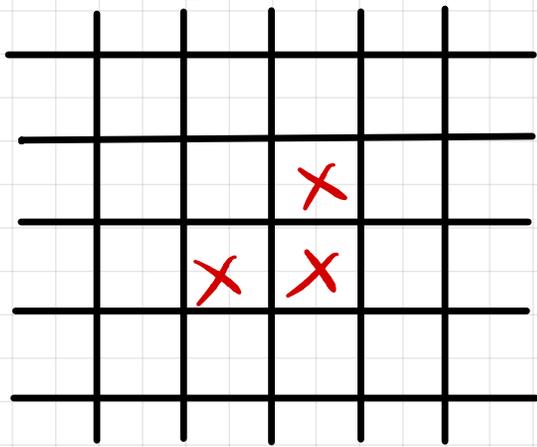
Input: Finite set of  $n \times n$  integer matrices

Question: Can we multiply them, in any order (repetitions allowed) to get the zero matrix?

[Note: Undecidable for 6  $3 \times 3$  matrices, or for 2  $15 \times 15$  matrices]



## 4. Conway's Game of Life



< 2 occupied nbrs.  $\rightarrow$  die  
> 3 " "  $\rightarrow$  die  
= 3 " "  $\rightarrow$  born

Input: Initial config. I; final config. F

Question: Is F reachable from I in a finite no. of generations?

## 5. Diophantine Equations [Hilbert's 10th Problem]

Input: Polynomial equation with integer coefficients

e.g.  $x^2 + y^2 + 1 = 0$

$4x^2 + xy^2 + 5 = 0$

Question: Does the equation have an integer solution?

# Gödel's Incompleteness Theorem

There does not exist a proof system for arithmetic that is both consistent and complete

Can't prove both  
 $P$  and  $\neg P$

Can prove all  
true statements, i.e.,  
 $\forall P$ , can prove either  
 $P$  or  $\neg P$

## Gödel's Theorem: Sketch of Proof

Write a statement in arithmetic of the form

$S$ : "This statement is not provable"

Case (i):  $S$  is provable.

Then by consistency  $S$  is true, so  $S$  is not provable ~~✗~~

Case (ii):  $S$  is not provable.

Then by completeness  $S$  is false, so  $S$  is provable ~~✗~~

Writing  $S$  is tricky – relies on clever arithmetic coding of statements ("Gödel numbering")

General form of  $S$ :  $\neg \exists z$  ( $z$  is a proof of this statement)  
self-reference

# Proof of Gödel's Theorem via Halting Problem

1. Given  $\langle M, x \rangle$ , write an arithmetic statement of the form

$\varphi_{M,x}$  :  $M$  halts on input  $x$

Form of  $\varphi_{M,x}$  :

$\exists z$  ( $z$  encodes a halting execution sequence)  
of  $M$  on input  $x$ )

## Proof of Gödel's Theorem via Halting Problem (cont.)

2. Solve Halting Problem by searching for a proof of either  $\varphi_{M,x}$  or  $\neg \varphi_{M,x}$

program Test Halt ( $M, x$ )

construct  $\varphi_{M,x}$

for each possible proof  $q$

if  $q$  is a proof of  $\varphi_{M,x}$  output "True"

if  $q$  is a proof of  $\neg \varphi_{M,x}$  output "False"

If proof system is complete & consistent, this program always decides the H.P.

Hence can't be complete & consistent  $\square$

Note: Uses fact that proofs can be enumerated & checked!

