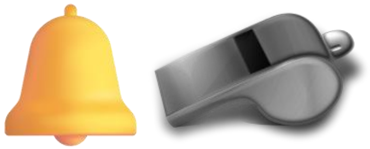
CMSC 28100

Introduction to Complexity Theory

Autumn 2025
Instructor: William Hoza

TMs can simulate all "reasonable" machines

- We could add various bells and whistles to the basic TM model
  - Left-right-stationary Turing machines
  - Multi-tape Turing machines
  - A Turing machine with a two-dimensional tape



- None of these changes has any effect on the power of the model

The Church-Turing Thesis

- Let $Y \subseteq \{0,1\}^*$

Church-Turing Thesis:

There exists an "algorithm" / "procedure" for figuring out whether a given string is in Y if and only if there exists a Turing machine that decides Y.

Intuitive notion

Mathematically precise notion

Turing machines vs. your laptop

- **OBJECTION:**
  - “Each individual Turing machine can only solve one problem.
  - My laptop is a single device that can run arbitrary computations.
  - Therefore, Turing machines don’t properly model my laptop.”

Code as data

- The response to this objection is based on the “code as data” idea
- A Turing machine M can be encoded as a binary string $\langle M \rangle$
- Plan: We will show how to simulate a Turing machine M, given its encoding $\langle M \rangle$

Universal Turing machines

Theorem: There exists a Turing machine U such that for every Turing machine M and every input $w \in \{0,1\}^*$:

- If M accepts w, then U accepts $\langle M, w \rangle$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on w, then U loops on $\langle M, w \rangle$.

- One super-algorithm that contains all other algorithms inside it!

Example: Exercise 3

M

	Symbols				
	0	1	_	#	$
a	(a, _, R)	(b, _, R)	(c, _, R)	(d, _, R)	
b	(y, 0, R)	(b, 0, R)	(c, 1, R)	(d, #, R)	
c	(y, 1, R)	(b, 1, R)	(c, _, R)	(d, #, R)	
d	(y, #, R)	(c, #, L)	(b, #, L)	(a, 0, L)	
e					
f					

$\Downarrow$

Download

$\Downarrow$

$\langle M \rangle$


```
… {"a": {"0": ["a", "_", "R"], "1": ["b",
"_", "R"], "_": ["c", "_", "R"], "#": ["d",
"_", "R"], "$": null, "&": null, "%": null,
"@": null}, "b": {"0": ["y", "0", "R"], "1":
["b", "0", "R"], "_": ["c", "1", "R"], "#":
["d", …
```


$\Uparrow$

Upload

$\Rightarrow$

Autograder Results

1) Inputs that are not edge cases (0/3)

```
Test Failed: 'Accept' != 'Reject'
- Accept
+ Reject
 : Your Turing machine behaves incorrectly
```

2) Edge case: Strings of zeroes (0/0.5)

```
Test Failed: 'Timeout' != 'Reject'
- Timeout
+ Reject
 : Your Turing machine behaves incorrectly
```

3) Edge case: Strings beginning with 1 (0/0.5)

$\approx U$

Universal Turing machines

Theorem: There exists a single Turing machine U such that for every Turing machine M and every input $w \in \{0,1\}^*$:

- If M accepts w, then U accepts $\langle M, w \rangle$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on w, then U loops on $\langle M, w \rangle$.

- To properly prove it, we need to clarify how $\langle M \rangle$ is defined

Encoding a Turing machine as a string

- To encode a Turing machine $M = \left(Q, q_0, q_{\text{accept}}, q_{\text{reject}}, \Sigma, \sqcup, \delta\right)$:
  - WLOG, $|Q| = |\Sigma| = 2^k$ for some $k \in \mathbb{N}$
  - WLOG, $Q = \{0,1\}^k$, $q_0 = 0^k$, $q_{\text{accept}} = 1^k$, and $q_{\text{reject}} = 01^{k-1}$
  - Encode $b \in \Sigma$ as $\langle b \rangle \in \{0,1\}^k$, with $\langle 0 \rangle = 0^k$, $\langle 1 \rangle = 10^{k-1}$, and $\langle \sqcup \rangle = 1^k$
  - Encode $(q, b, D) \in Q \times \Sigma \times \{\mathrm{L}, \mathrm{R}\}$ as $\langle q, b, d \rangle = q \langle b \rangle \langle D \rangle \in \{0,1\}^{2k+1}$
  - Then $\langle M \rangle = 1^k 0 \langle \delta \rangle$, where $\langle \delta \rangle$ is the list of $\langle \delta(q, b) \rangle$ for all $(q, b) \in Q \times \Sigma$

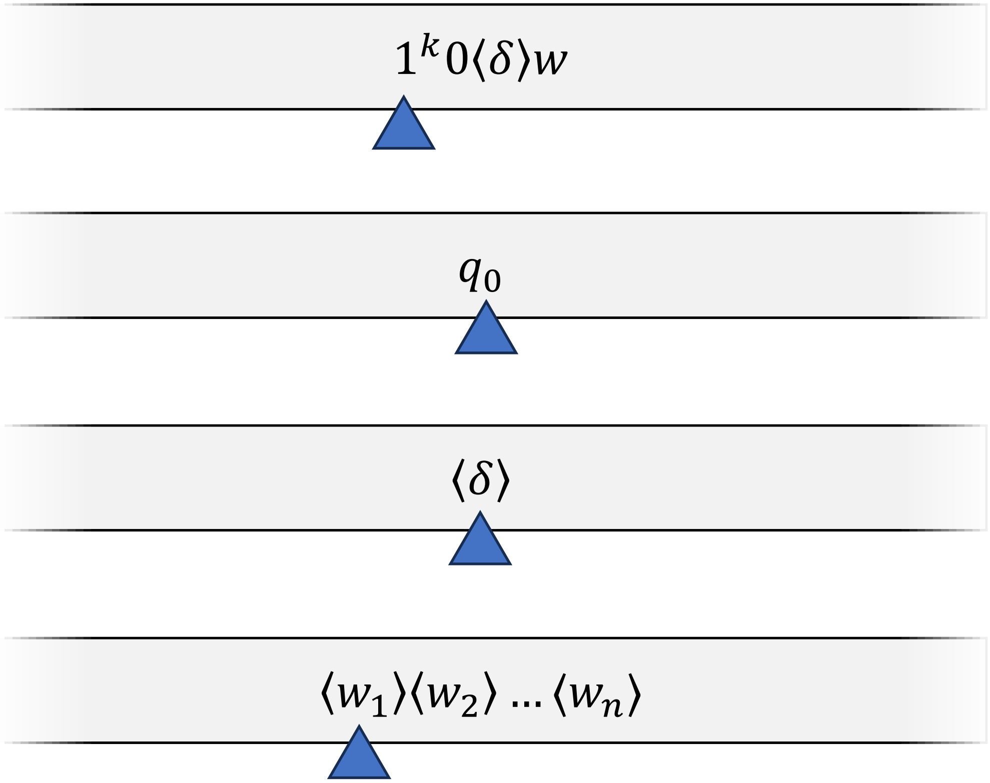
Universal Turing machines

Theorem: There exists a single Turing machine U such that for every Turing machine M and every input $w \in \{0,1\}^*$:

- If M accepts w, then U accepts $\langle M, w \rangle \coloneqq \langle M \rangle w$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on w, then U loops on $\langle M, w \rangle$.

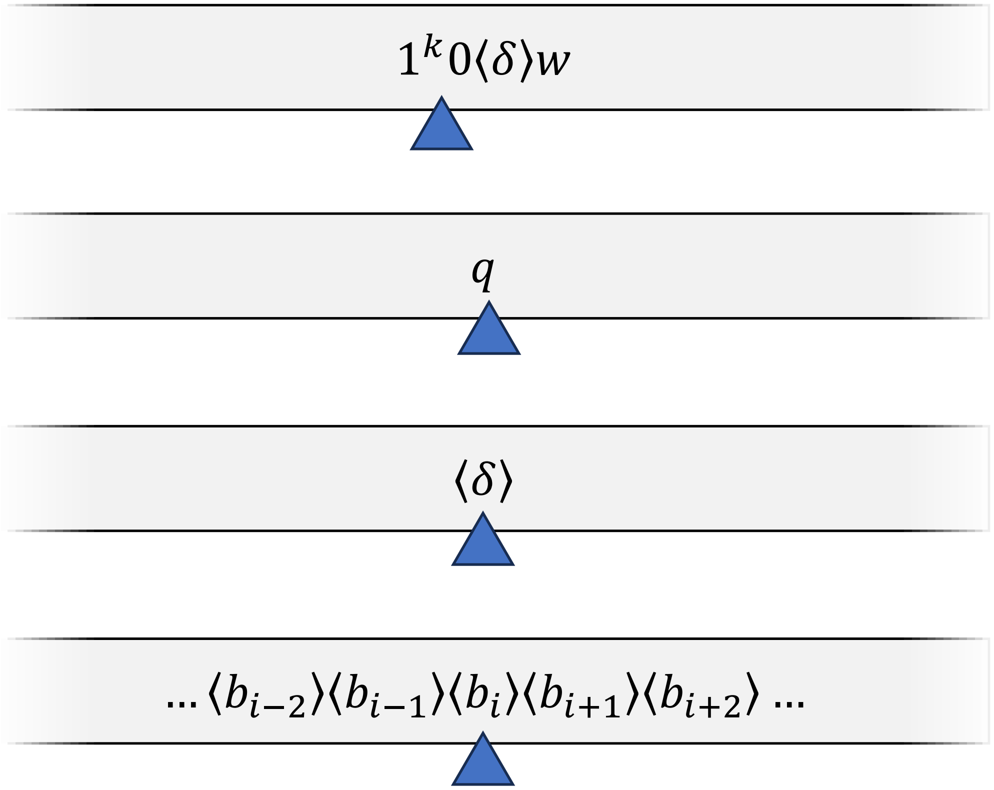
- **Proof sketch:** Next two slides

Initializing the simulation



- U is given $\langle M, w\rangle = 1^k0\langle\delta\rangle w$
- Initialize a tape containing $q_0 = 0^k$
- Initialize a tape containing $\langle\delta\rangle$
  - Note: $|\langle\delta\rangle| = 2^{2k} \cdot (2k+1)$. Can compute using binary counter
- Initialize a tape containing $\langle w_1\rangle\langle w_2\rangle \ldots \langle w_n\rangle$
  - Note: $\langle w_i\rangle = w_i0^{k-1}$

Advancing the simulation



- Until the simulation reaches a halt state:

1. Find $\langle\delta(q, b_i)\rangle = \langle q', b', D\rangle$ within $\langle\delta\rangle$
   - Idea: Treat $q\langle b_i\rangle$ as a number N in binary
   - Use a binary counter to go to position $N \cdot (2k+1)$
2. Replace q with q' and replace $\langle b_i\rangle$ with $\langle b'\rangle$
3. Move this head k cells in direction D

Interpretation of universal Turing machines

- One piece of "hardware" that can run arbitrary "software"
- It's a general-purpose, programmable computer
- This is why you don't need a separate laptop for each task
- If you want to build a computer from scratch in some post-apocalyptic future, then your job is to build a universal Turing machine

The Church-Turing Thesis

- Let $Y \subseteq \{0,1\}^*$

Church-Turing Thesis:

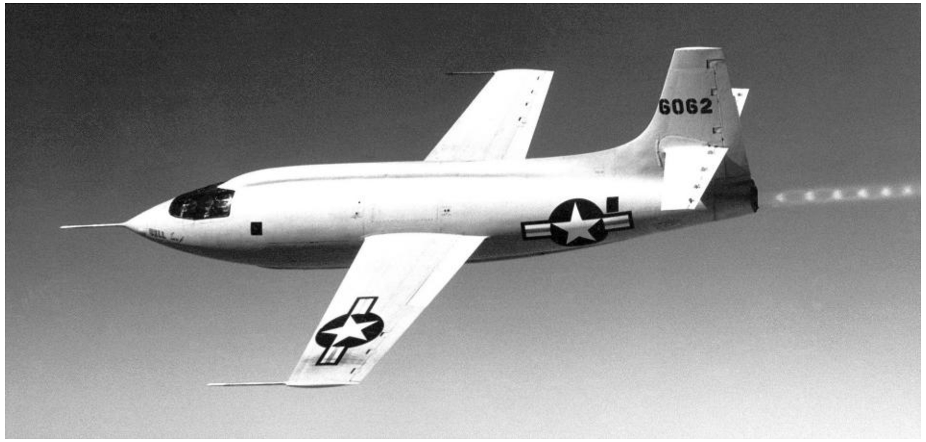
There exists an “algorithm” / “procedure” for figuring out whether a given string is in Y if and only if there exists a Turing machine that decides Y.

Intuitive notion

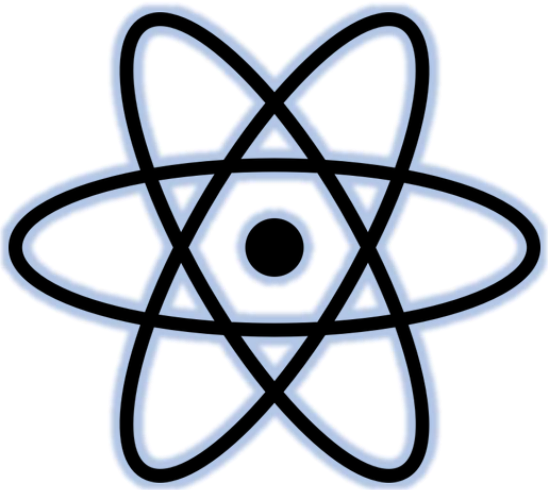
Mathematically precise notion

Humans vs. technology

- **OBJECTION:** “The Turing machine model is based on paper-and-pencil computation. Maybe we can solve undecidable problems using advanced science and technology!”



Hypercomputers



- A hypercomputer is a hypothetical device that can solve some computational problem that cannot be solved by Turing machines, such as $\mathrm{SELF\text{-}REJECTORS}$
- Could it be possible to build a hypercomputer?
- We could try using quantum physics, antimatter, black holes, dark energy, superconductors, wormholes, closed timelike curves, …

The Physical Church-Turing Thesis

- Let $Y \subseteq \{0,1\}^*$

Physical Church-Turing Thesis:

It is physically possible to build a device that decides Y if and only if there exists a Turing machine that decides Y.

The Physical Church-Turing Thesis

- The standard Church-Turing thesis is a philosophical statement
- The Physical Church-Turing thesis is a scientific law
- Conceivably, it could be disproven by future discoveries… but that would be very surprising
- Analogy: Second Law of Thermodynamics
- Analogy: Cannot travel faster than the speed of light

Which problems

can be solved

through computation?

What are Turing machines capable of?

Which languages are decidable?

Contrived vs. natural

- $\text{SELF-REJECTORS} = \{\langle M \rangle : M \text{ is a self-rejecting Turing machine}\}$
- We proved that SELF-REJECTORS is undecidable
- **OBJECTION:** "SELF-REJECTORS seems like a very contrived example."
- **RESPONSE:** There are other undecidable languages that are natural/well-motivated/interesting!

The halting problem

- **Informal problem statement:** Given a Turing machine M and an input w, determine whether M halts on w.
- **The same problem, formulated as a language:**

$$\mathrm{HALT} = \{\langle M, w\rangle : M \text{ is a Turing machine that halts on input } w\}$$

- It's the problem of identifying bugs in someone else's code! 🐞

Attempting to decide HALT

- Given $\langle M, w \rangle$:
    1. Simulate M on w
    2. If it halts, accept
    3. Otherwise, reject

Does the proposed algorithm work?

A: No. It's not necessarily possible to simulate M on w

B: No. There are inputs for which it should accept, but it doesn't

C: No. There are inputs for which it should reject, but it doesn't

D: Yes. HALT is decidable

Respond at PollEv.com/whoza or text "whoza" to 22333