CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

Which problems

can be solved

through computation?

The Church-Turing Thesis

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

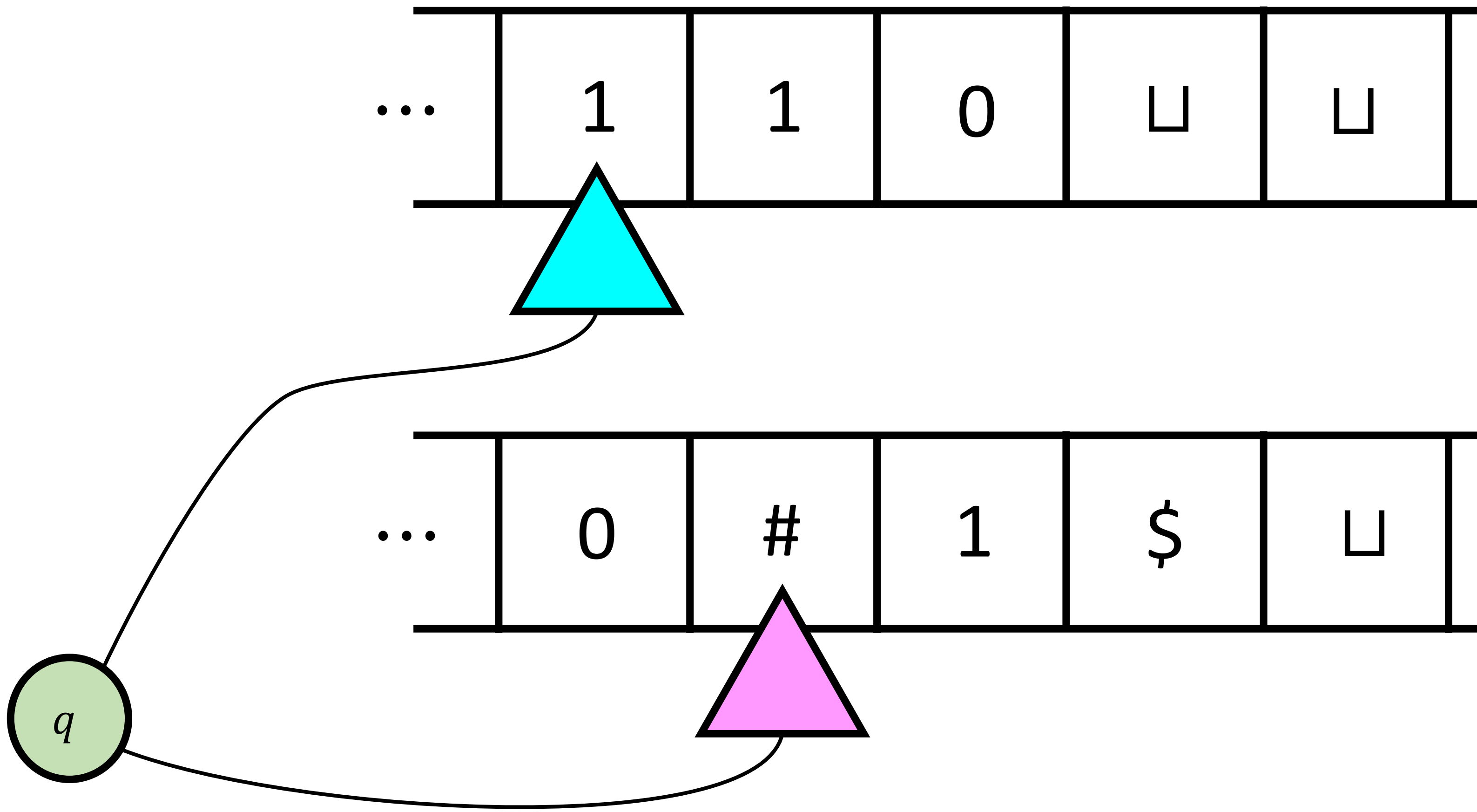
Church-Turing Thesis:

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

Multi-tape Turing machines



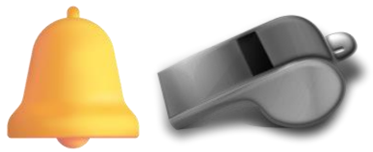
Multi-tape Turing machines

- Let k be any positive integer and let Y be a language

Theorem: There exists a k-tape TM that decides Y if and only if there exists a 1-tape TM that decides Y

TMs can simulate all "reasonable" machines

- We could add various other bells and whistles to the basic TM model
  - The ability to observe the two neighboring cells
  - The ability to "teleport" back to the initial cell in a single step
  - 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.”

Email machine??

Zoom machine??

General-purpose computer

Tetris machine??

Photoshop machine??

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 4

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 (0/5.5)

```
Running the machine on "01"... Timeout

Test Failed: 'Timeout' != 'Accept'
- Timeout
+ Accept
```

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

```
Running the machine on "0"... Timeout

Test Failed: 'Timeout' != 'Reject'
- Timeout
+ Reject
```

$\approx U$

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 $M = (Q, q_0, q_{\text{accept}}, q_{\text{reject}}, \Sigma, \sqcup, \delta)$:
  - WLOG, $|Q| = |\Sigma| = 2^k$ for some $k \in \mathbb{N}$
  - WLOG, $Q = \{0, 1\}^k$, $q_0 = 0^k$, $q_{\text{accept}} = 1^{k-1}0$, and $q_{\text{reject}} = 1^k$
  - 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 $(q, b) \in Q \times \Sigma$

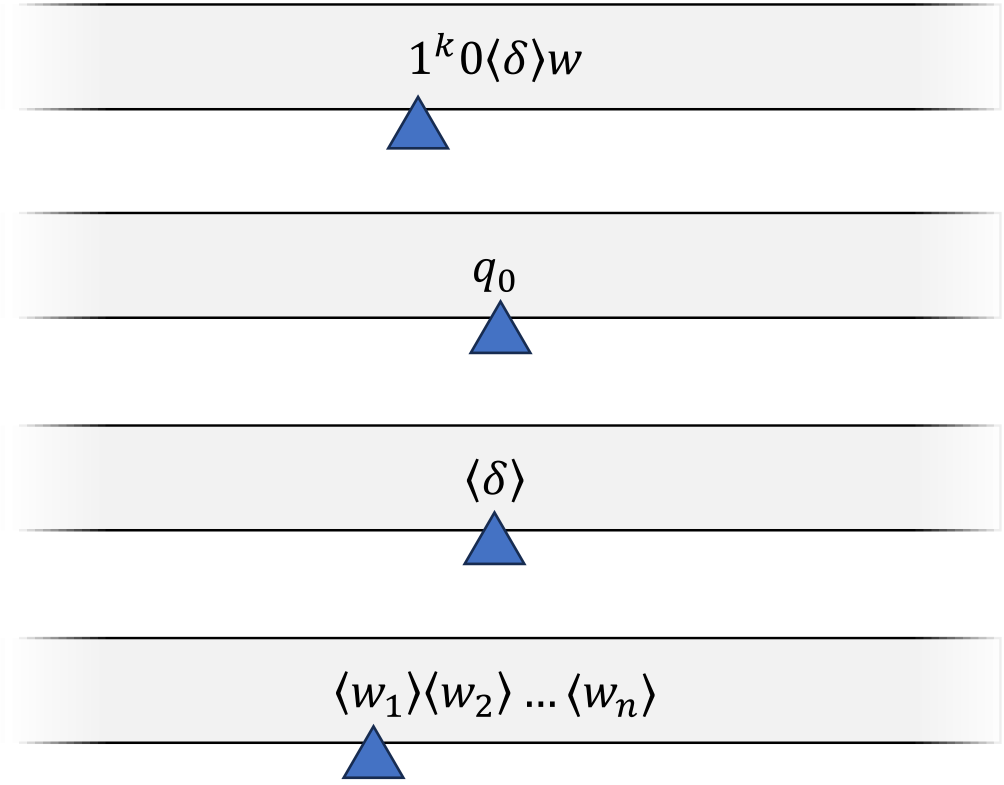
Universal Turing machines

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 $\langle M, w \rangle$.

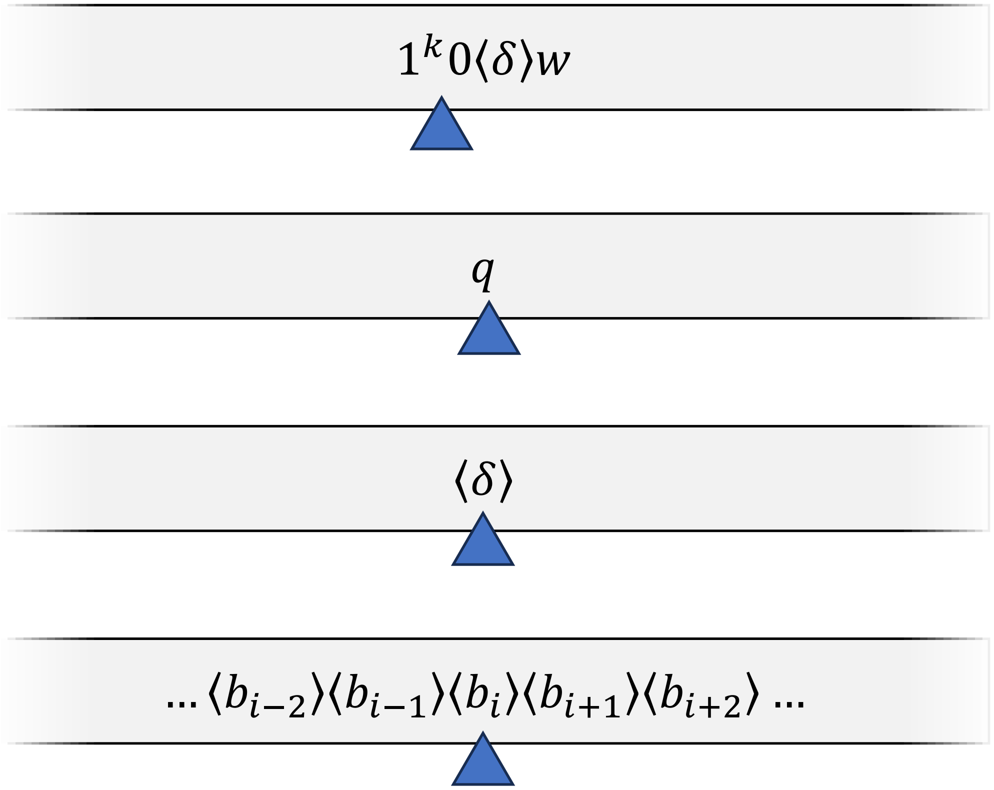
- **Proof sketch:** Next two slides

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: To figure out where $\langle\delta\rangle$ ends and w starts, count to 2^{2k}
- Initialize a tape containing $\langle w_1\rangle\langle w_2\rangle \dots \langle w_n\rangle$
  - 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
   - Count to N
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

- A universal Turing machine can be “programmed” to do anything that is computationally possible
- 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

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 rejection pro

Does the proposed algorithm successfully decide REJECT?

A: No. Step 1 isn't legal, so the algorithm isn't well-defined

B: No. Step 2 isn't legal, so the algorithm isn't well-defined

C: Yes

D: No. The algorithm behaves incorrectly in some cases

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

- **Informal problem statem**

  w, determine whether M

- **The same problem, formulated as a language:**

$$\mathrm{REJECT} = \{\langle M, w\rangle : M \text{ is a Turing machine that rejects } w\}$$

- Attempted algorithm:

> Given $\langle M, w\rangle$:
>
> 1. Simulate M on w.
>
> 2. If it rejects, accept. Otherwise, reject.

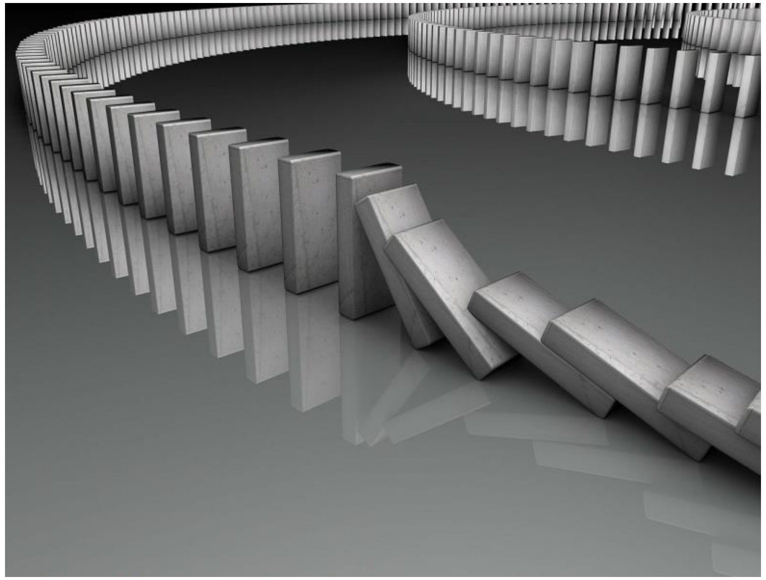
The rejection problem is undecidable

- $\text{REJECT} = \{\langle M, w\rangle : M \text{ is a Turing machine that rejects } w\}$

Theorem: REJECT is undecidable.

- How should we prove it?

Reductions



- We already proved that SELF-REJECTORS is undecidable
- Plan: Let's show that if REJECT were decidable, then SELF-REJECTORS would also be decidable – a contradiction
- "Proof by reduction"

Proof that REJECT is undecidable

- Assume for the sake of contradiction that there is some Turing machine R that decides REJECT
- Let's construct a new TM S that decides SELF-REJECTORS

S

> Given the input $\langle M \rangle$:
>
> 1. "Copy and paste" to construct the string $\langle M, \langle M \rangle \rangle$
> 2. Simulate R on $\langle M, \langle M \rangle \rangle$
> 3. If R accepts, accept. If R rejects, reject.

- If $\langle M \rangle \in \text{SELF-REJECTORS}$, then R accepts $\langle M, \langle M \rangle \rangle$, and therefore S accepts $\langle M \rangle$
- If $\langle M \rangle \notin \text{SELF-REJECTORS}$, then R rejects $\langle M, \langle M \rangle \rangle$, and therefore S rejects $\langle M \rangle$