Announcements:

- HW5: due today; clock paused on late days until Sat at 12:07am
- No class Thurs
- Today structure: start — $\boxed{\text{15 mins for course evals.}}$

---

0. Review of undecidability by reduction.
1. Flavors of Complexity
2. Information Theory/Description Cxty.
3. Wrap-up — end of class + ~an extra hour for review.

0. Undecidability via Reduction

$\boxed{\text{your perpetual motion machine.}} \Longrightarrow$ break laws of thermodynamics

$\boxed{\text{Decider for } L} \Longrightarrow$ decide $HALT_{TM}$

decide A_{TM}

decide E_{TM}

decide any language we've shown to be undecidable.

Prop.

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2$ are TMs and $L(M_1) = L(M_2)\}$

is undecidable.

- Assume EQ_{TM} is decidable. (Some decider S for EQ_{TM}).
- Know (past fact): $E_{TM} = \{\langle M\rangle \mid M$ is a TM that rejects every input string $L(M) = \emptyset\}$ is undecidable.

why E_{TM}

We'll show: Using S, we can decide E_{TM}. X

We'll build a decider T for E_{TM}.

T: "On input $\langle M\rangle$:

(Goal: tie a decision of "are these two TMs equivalent?" to "is this TM empty?")

- Write down $\langle M_{NO}\rangle$, where M_{NO} rejects all strings.
- Run $S(M, M_{NO})$, and accept/reject according to S."

T decides $E_{TM} \Longrightarrow$ X. Our assumption is false, no decider S exists.

If $L(M) = \emptyset \iff L(M) = L(M_{NO}) \iff S(M, M_{NO})$ accepts.

Complexity Flavors

time
- P: all languages a TM can decide in time $O(n^k)$, for some k.
- NP: all languages a NTM can decide in time $O(n^k)$, for some k.

Other resources?

space PSPACE: all languages that a TM can decide using unlimited time, and $O(n^k)$ tape squares.

$P \subseteq PSPACE$. space $\geq$ time.

$NP \subseteq PSPACE$.

randomness BPP = all languages that a probabilistic TM can decide with 99% probability in polynomial time.

quantumness BQP = all languages that a quantum TM can decide with high constant probability in polynomial time.

Complexity "Outlooks"

- worst-case complexity.

  measure runtime as $f(n)$ = max number of steps over any input of length n.

- average-case complexity

  measure runtime as $f(n)$ = "average" number of steps on a "random" input of length n.

  some distribution over the input.

  usually uniform

- "smoothed"

  runtime $f(n)$ = max number of steps over any length-n input

  (but you can fudge the input)

  (change the input a little bit (often w/ random noise)

- "testing"

REALLY BIG input.

Learning about the input takes time, using what you know to compute is cheap.

streaming, sketching, query complexity, PAC-learning.
fine-grained complexity $O(n^{\omega})$ $\omega = \underline{2.37}$

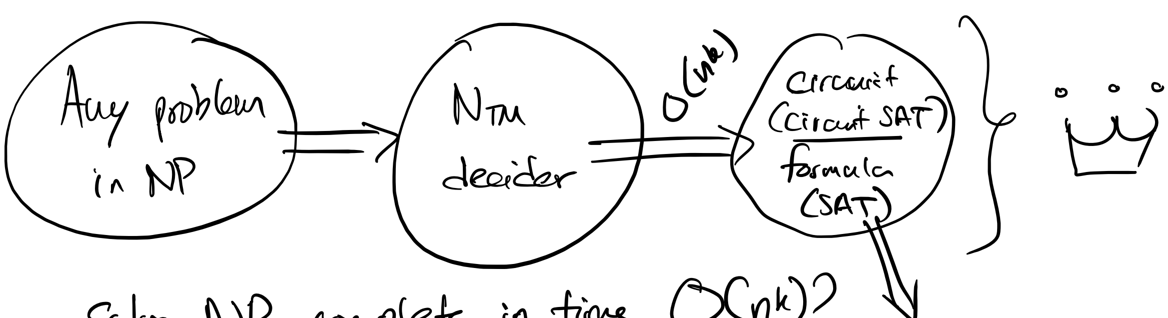
break: 2:18

NP $\approx$ efficiently solved/decided by some NTM (*)

$\approx$ efficiently verifiable (some verifier accepts input strings in language w/ compatible proofs attached.
deterministic TM

How might we show $NP \subseteq P$?

<u>Idea</u>: show how to turn <u>any</u> problem in NP, which is decided by some NTM, into one specific problem type that tells us if the NTM accepts on some input.



Solve NP-complete in time $O(n^k)$?
↳ $NP \subseteq P$.
↳ $P = NP$.

Hamiltonian Path
Traveling Salesman
Independent Set
Subset Sum

Graph 3-coloring
⋮

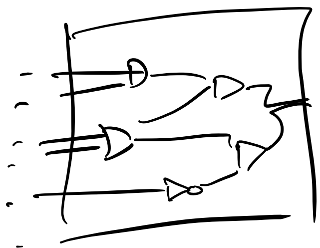
SAT = satisfiability.

$\langle \varphi, x_1, x_2, x_3 \cdots x_n \rangle$

variables can be T/F

$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_4 \vee x_1) \wedge \overline{x_2} \cdots$

Question: can we assign T/F values to the variables to make this formula true?



T/F

Circuit-SAT.

2. Information Theory

two strings:

$$x = \overbrace{01\ 01\ 01\ 01\ 01\ 01\ 01\ 01}^{\text{eight "01"'s}}$$

$$y = \underbrace{1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,0\,1}_{???}$$

Encoding English letters $\Rightarrow$ bits

E = 0
T = 1
I = 00
A = 01
N = 10
M = 11
⋮
Q = 1011
J
X

} Morse code

"description complexity" – how much information we need to describe a string, in bits ≈ "size" of the smallest TM that halts w/ our string on the tape.

- Communication
- compression
- philosophy of science / epistemology (Bayesian reasoning / Occam's razor)

break: back at 3:18

In-class review

Reducing Variant TMs

Given: a computational model like a TM, but a little different.

Goal: Show any languages new model recognizes ⟹ recognizable by TM

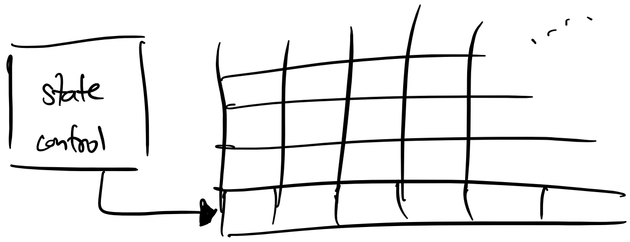
Any languages rec. by TM ⟹ recognizable by new model.

Def. A 2DTM is defined just like a TM, except

(1) instead of $\{L, R\}$ it can move $\{L, R, U, D\}$

(2) we operate on an infinite tape that extends up and right.

(3) input on bottom row.



1) Any language a TM can recognize some 2DTM can recognize. ($\Longrightarrow$).

2) Any language a 2DTM can recognize, some TM can recognize.

<u>Proof.</u> 1) Let T_{2D} be a 2DTM. We can simulate as follows with a TM T:

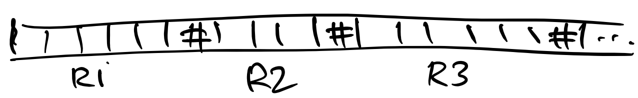
$T =$ "On input w:

1) Compute until we move "up" to a new row.

Simulate $T_{2D}(w)$

2) When we move up to a new row, add a delimiter # to the end of the tape and use that space to store the new row, continue from the appropriate space.

(3) Switch back and forth between rows as necessary.)



4) If we hit a delimiter and need more space in any row, move everything over one space to the right and continue."

(Very slick: $\mathbb{Z} \times \mathbb{Z}$ is countably infinite.
Let f be a mapping from $\mathbb{Z} \times \mathbb{Z} \to \mathbb{N}$.

Keep track of T_{2D}'s coordinates and use $f(x,y)$ as the tape square for (x,y).)

$E_{CFG} = \{\langle G\rangle \mid G$ is a grammar w/ $L(G)=\emptyset$
(equiv: G generates no strings.$\}$

$S \to AS$
$A \to 0$

$S \to 0A0$
$A \to 0A0 \mid B$
$B \to 1B1 \mid \#$

Idea: ① mark terminals.

② mark any variables that produce a string of only terminals. (Thus, marked variables can generate terminal strings.)

③ repeat (2), marking any var that produces a string of marked symbols.

At any point, the dot indicates "can generate a terminal string."

After no new vars get marked on some iteration, accept if S is not marked, reject otherwise

$INF_{CFG} = \{\langle G\rangle \mid L(G)$ is infinite$\}$

$INF_{TM} = \{\langle M\rangle \mid L(M)$ is infinite$\}$

Goal:

Show if we have a decider for INF_{TM}, we could decide A_{TM}.

Fact: $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ is undecidable.

Assume that S is a decider for INF_{TM}.

We'll build T that decides A_{TM}.

decider for A_{TM}

$T =$ "On input $\langle M, w\rangle$:

??? Write down $\langle M_2\rangle$, which works as follows:

$M_2 =$ "On input x:

Ignore x, simulate $M(w)$, if $M(w)$ accepts, accept."

Run $S(\langle M_2\rangle)$ and accept if and only if it accepts.

If $M(w)$ accepts: M_2 accepts all strings.

If $M(w)$ runs forever or rejects: M_2 accepts nothing: $L(M_2) = \emptyset$.

M_2 accepts an infinite language (Σ^*) $\Longleftrightarrow$ $M(w)$ accepts.

How did we come up with M_2 here?

- Can do: use S to test if a machine recognizes an infinite language.
- Want to know: does M accept w?
- Idea: Let's build a machine that recognizes an ∞ language $\Longleftrightarrow$ M accepts w.

Conclusion: If we had some decider S for INF_{TM}, we could use it to build a decider T for A_{TM}. This is impossible, So we can't possibly have S.

Proof. If we have a decider S for
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$
we can decide $HALT_{TM}$.

Decider for HALT. T = "On input $\langle M, w\rangle$:

// Goal: get M_1, M_2 s.t. $L(M_1) = L(M_2)$
// if and only if $M(w)$ halts.

$M_{\cancel{yes}\ no}$: "On input x, ~~accept~~ reject."

M_2: "On input x: If $x \neq w$, reject. If $x = w$: Run $M(w)$, and if $M(w)$ halts, accept."

Run $S(\langle M_{\cancel{yes}\ no}, M_2\rangle)$, and accept if S ~~accepts~~ rejects."

$S(\langle M_{yes}, M_2\rangle)$ accepts? This means $L(M_2) = L(M_{yes}) =$ all strings.
So we know $M(w)$ must halt.

$S(\langle M_{yes}, M_2\rangle)$ reject? This means $L(M_2) \neq L(M_{yes}) =$ all strings
$\Rightarrow L(M_2)$ not all strings $\Rightarrow M(w)$ didn't halt.

Context-free pumping lemma.

$$K = \{a^i b^j c^k \mid i,j,k \geq 1 \text{ and } ij = k\}.$$

Assume for contradiction that K is context-free.
By the CFPL, $\exists p$ such that for all $s \in K$ with $|s| \geq p$, s can be divided into 5 substrings $uvxyz$ such that

(1) $uv^i x y^i z \in K$ for all $i \geq 0$
(2) $|vxy| \leq p$
(3) $|vy| > 0$.

Contradiction string: $s = a^p b^p c^{p^2}$. $|s| \geq p$, $s \in K$.

✓ (*) By (2) and (3), vxy contains at most two of the three letters and vy contains at least one symbol.

(left: vxy has $\leq$ 2 diff symbols)

$a \cdots ab \cdots \overset{v}{b}\, c \cdots \overset{y}{c}\, c \cdots c$

Case 1) vxy contains no c's.

(vxy has $\leq$ 2 diff symbols and some c's)

Then uv^2xy^2z must increase # of a's or b's, so $i' \cdot j' > k$.

Case 2) vxy contains only c's.

Then uv^2xy^2z makes $ij' < k'$.

case 3) vxy contains both b's and c's.

- vy contains only one letter → done.
- vy contains both b's and c's.

$\underset{v}{\overset{1}{b}}\ \underset{x=\varepsilon}{}\ \underset{y}{\overset{p-1}{ccc\cdots ccc}}$

uv^2xy^2z has: p a's.

at least $p+1$ b's

at most $p^2 + (p-1)$ c's.

so: $\underset{\#a\text{'s}}{i'} \cdot \underset{\#b\text{'s}}{j'} \geq p(p+1) = p^2 + p > k'$, new # c's

---

(*1) A decidable $\Longleftrightarrow$ $\bar{A}$ decidable.

(*2) A decidable $\Longleftrightarrow$ A and $\bar{A}$ are both recognizable.

let M_A be a rec. for A

$M_{\bar{A}}$ be a rec for $\bar{A}$

$M_{\text{decide-}A}$: "On input w

Simulate $M_A(w)$ and $M_{\bar{A}}(w)$ in parallel.

(One will halt.) Decide accordingly.

($w \in A$ if $M_A(w)$ halts, $w \notin A$ if $M_{\bar{A}}$ halts.)

A_{TM} recognizable, undecidable

$\overline{A_{TM}}$ must be unrecognizable, else A_{TM} would be decidable by $\ast$2.