Announcements:

- HW 5 due Tues
- Final next week: comprehensive post on Ed.

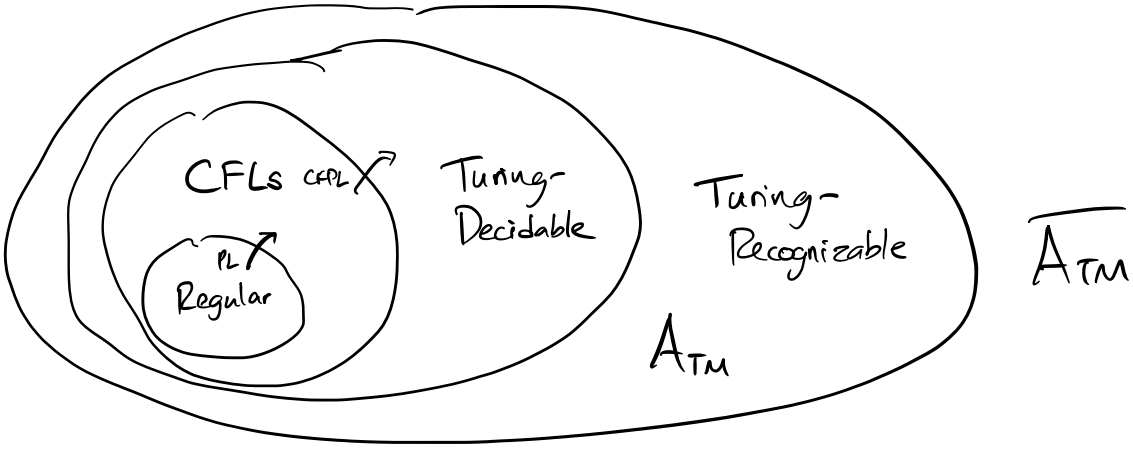
  Review times / O. Hours.

  Mon: Tim 10-12
  Yunya PM
  Melody evening

  Tues: Tim 9-11
  (class + 1 hour)
  Ronin in PM (but HW5-focused)

Today:

0. Undecidable / unrecognizable langs.
1. Proving undecidability via reduction.
2. Measuring time complexity.



$A_{TM} = \{\langle M, w\rangle \mid M \text{ a TM that accepts string } w\}$

(A_{TM} recognizable)

Theorem: A_{TM} is undecidable.

Proof sketch:

- Assumed H was a decider for A_{TM}
- Built a decider P that accepts $\langle M \rangle \longleftrightarrow H(\langle M, \langle M \rangle \rangle)$ rejects.
- $P(\langle P \rangle)$ accepts $\longleftrightarrow H(\langle P, \langle P \rangle \rangle)$ rejects $\longleftrightarrow P(\langle P \rangle)$ doesn't accept.

Theorem (**): $\overline{A_{TM}} = \{\langle M, w \rangle \mid M$ is a TM that doesn't accept w (or bad input)$\}$ is unrecognizable.

(*) First observation: A language B is decidable if and only if B and $\overline{B}$ are both recognizable.

Proof.

If B is decidable, we can use the decider as a subroutine for B, $\overline{B}$ recognizers.

If B and $\overline{B}$ are both recognizable, we can build a decider that simulates both recognizers in parallel. Then accept or reject based on the recognizer that accepts.

alternate steps between two simulations.

Proof of Theorem **.

- We know A_{TM} is recognizable.
- If $\overline{A_{TM}}$ were recognizable, then A_{TM} would be decidable by (*).
- A_{TM} is not decidable, so $\overline{A_{TM}}$ cannot be recognizable. ∎

1. Reductions.

"I could solve problem B if I could just solve problem A."

Example.

$$A_{NFA} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts } w\}$$

Know: can convert any NFA ⇒ DFA

"I can decide A_{NFA} if I can just decide A_{DFA}."

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on } w\}$$

What if we proved:

"I could decide A_{TM} if I could just decide $HALT_{TM}$."

Can't decide A_{TM} ⟹ Impossible to decide $HALT_{TM}$.

"Reducing" A_{TM} to $HALT_{TM}$.

Theorem. $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on } w\}$ is undecidable.

Proof. By contradiction.

Assume that R is a decider for $HALT_{TM}$. We'll show the existence of R would let us decide A_{TM}, which is a contradiction.

Define a new machine M_1 that will decide A_{TM} using R.

M_1 = "On input $\langle M, w\rangle$, an encoded TM M and string w:

1. Run R on $\langle M, w\rangle$. If R rejects, reject.
2. If R accepts, simulate M(w) and accept if and only if M(w) accepts."

M_1 is a decider for A_{TM}, which is a contradiction.

Our assumption is false: R cannot exist. ∎

Q. Is $HALT_{TM}$ recognizable? Yes – simulate M(w) and accept if M(w) stops.

Q. Is $\overline{HALT_{TM}}$ recognizable? No – if it could, we could make a decider for $HALT_{TM}$.

Theorem. $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM that doesn't accept any strings}\}$, is undecidable.

Proof. We'll show that if E_{TM} were decidable, we could decide A_{TM}.

Assume for contradiction that some TM S decides E_{TM}.

We'll use S to build a decider T for A_{TM}.

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

1. Using M, write down the description of a new TM M' that simulates M(w) on input w and rejects on all input $x \neq w$.
2. T will run our decider S for E_{TM} on $\langle M' \rangle$.
   If $S(\langle M' \rangle)$ accepts, then M' accepts nothing so M doesn't accept w. Reject.
   If $S(\langle M' \rangle)$ rejects, L(M') isn't empty, which means M(w) accepts. Accept."

T decides A_{TM}, which is a contradiction. So E_{TM} undecidable.

① $ALL_{TM} = \{\langle M \rangle \mid M \text{ is a TM that accepts all strings}\}$

Show ALL_{TM} decidable $\Rightarrow$ A_{TM} decidable.

Goal: a decider for A_{TM}. On input $\langle M, w \rangle$, how can we make M' s.t. $\langle M' \rangle \in ALL \leftrightarrow \langle M, w \rangle \in A_{TM}$?

② $EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Show EQ_{TM} decidable $\Rightarrow$ E_{TM} decidable.

E_{TM}: TMs that don't accept anything.

(**) $REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$

Show $REGULAR_{TM}$ decidable $\Rightarrow$ A_{TM} decidable.

We'll show

$ALL_{TM} = \{\langle M\rangle \mid M \text{ accepts every string}\}$ is undecidable.

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

Proof. Assume for contradiction that ALL_{TM} is decidable.

Want to show A_{TM} is decidable.

A_{TM} is decidable ✗

We'll build a TM T that decides A_{TM} and uses a hypothetical decider S for ALL_{TM} as a subroutine.

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

1. T writes down the description of a machine M' that accepts on all $x \neq w$, and simulates M(w) on input w, accepting if M(w) accepts.
2. Run $S(\langle M'\rangle)$.

   If $S(\langle M'\rangle)$ accepts, this means $L(M')$ = all strings, which implies M(w) accepts. Accept.

   If $S(\langle M'\rangle)$ rejects, this means M' doesn't accept some string, which means M(w) doesn't accept. Reject.

T decides A_{TM}, which is a contradiction. ∎

use S to decide whether some machine $M' \in ALL_{TM}$, for an M' that gives us info about M(w).

Break: Back at 2:40

See Example 8: Showing undecidability via reduction.

Up soon.

(For fun)

Rice's Theorem: Consider a language of TM descriptions (like $\langle M \rangle$)

$$P_X = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ has property } X.\}$$

If (1) P_X includes some, but not all TMs and // nontrivial
(2) X depends only on $L(M)$ — if $L(M_1) = L(M_2)$ either $M_1, M_2 \in P_X$ or $M_1, M_2 \notin P_X$,

then P_X is undecidable.

---

3. Complexity.

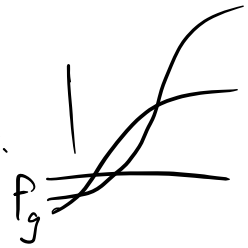
Computability := whether or not a TM can recognize a language (or compute a function.)

Aside: A function $f: \Sigma^* \to \Sigma^*$ is computable if there is some TM that computes it: halts with $f(w)$ on the tape for any input w.

Complexity := how many time steps (or other resources) does it take? (To recognize language / compute function.)

3.1 Big-O

Def. Let f and g be positive, increasing functions.



We say $f(n) = O(g(n))$ if we can pick positive integers n_0, c such that

$$f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0.$$

— $0.5n$, $3n$, $10n+6$, $0.1n$, $\log_2(n)$, $\sqrt{n} = O(n)$

- $6n^2$, $3n^2+6n+2$, $10n$, 4 $= O(n^2)$
- $100 \cdot 3^n$, 2^n, n^{100}, n^k for any finite $k = O(3^n)$

If $n = 1000$, n^4 = a trillion

$2^n \geq$ # quarks in the universe.

Def. If M is a deterministic TM that halts on all inputs, the running time or time complexity of M is a function $f(n)$ that counts the maximum number of steps on any input of length n.

Big O makes this tractable — we can ignore constant scaling.

Time Complexity Examples.

Consider $A = \{0^k 1^k \mid k \geq 0\}$.

Decider M_1:

M_1 = "On input w, with length $|w| = n$:

	Maximum Time
1. scan the input tape L → R and see if we match 0^*1^*. Reject if not.	$O(n)$
2. shuttle back and forth, crossing off one 0 and one 1 each time. Reject if we run out of either number first.	$O(n^2)$
3. Accept if we cross off the last 0, 1 at the same time.	$O(1)$

Runtime: $O(n) + O(n^2) = \underline{O(n^2)}$.

Decider M_2 — two tapes!

M_2 = "On input w of length $|w| = n$:

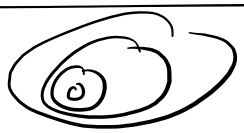
1. Scan input and reject if we don't match 0^*1^*.	$O(n)$
2. Scan L → R on the first tape. Whenever we pass	$O(n)$

a zero, copy it to the second tape in the same step.

3. Scan tapes one and two at the same time, crossing out a 1 on tape one and a 0 on tape two on every step. Accept if the number is equal. | $O(n)$

Runtime: $O(n) + O(n) + O(n) = O(n)$.

P: Polynomial Time



Definition: P is the class (set) of all languages decidable in polynomial time — that is, decidable by a TM with runtime $O(n^k)$ for some fixed k, on inputs of size n.

Why polynomial time?

(1) polynomials $\ll$ exponentials

$$n^{1000} \ll 1.01^n$$

(2) If we don't restrict the polynomial, any polynomial-time TM can use other polynomial-time TMs as subroutines.

A – decider, in time $O(n^3)$

B – decider, runs subroutine A $O(n^2)$ times

Total runtime: $O(n^3) \cdot O(n^2) = O(n^5)$.

P $\approx$ efficiently decidable languages.

Problems in P:

- all regular languages (DFAs will take n steps, when $|w| = n$).
- all CFLs
- sorting, searching, testing connectivity,

multiplying, element distinctness.

HAMPATH = $\{\langle G \rangle \mid G$ has a Hamiltonian path$\}$.

SUDOKU = $\{\langle S \rangle \mid S$ is a sudoku puzzle with a solution$\}$.

SUBSET SUM = $\{\langle x_1, x_2, \cdots x_n, t \rangle \mid$ some subset of the x's adds to $t\}$.

SAT = $\{\langle y_1, y_2 \cdots y_n, \varphi \rangle \mid \varphi$ encodes a logical formula with $\wedge, \vee, \neg$, and the y's can be given T/F values that make the formula true.$\}$

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

Hard problem similarities? Brute force solutions

Hard to answer, easy to check.

NP: Verifiable in Polynomial Time

Def: A Verifier V for some language A is a deterministic TM such that for all $w \in A$, there is some certificate (proof) string c such that $V(\langle w, c \rangle)$ accepts.

(No certificate makes V accept on $w \notin A$.)

Def: NP is the class of all languages that have a verifier which runs in polynomial time.

HAMPATH, SUDOKU, SUBSET SUM, SAT $\in$ NP

NP $\approx$ "efficiently verifiable languages." $P \subseteq NP$.

Theorem: NP is the class of all languages decidable by a Nondeterministic TM in polynomial time.

Proof sketch:

If $L \in NP$ — then there is some polynomial-time verifier for L. An NTM can decide L by simulating the verifier and nondeterministically guessing the proof.

Likewise, if L is the language recognized by some NTM, our verifier can use the sequence of "correct guesses" corresponding to an accepting branch for a string $w \in L$ as the certificate c. □

$P \stackrel{?}{=} NP.$