Last time:

P is the class of decision problems (languages) that are decidable in polynomial time (ptime).

Def: A language L is verifiable in ptime if there is a ptime TM V (the verifier) such that for every $n \in \mathbb{N}$ and input $w \in \Sigma^n$,

$w \in L \iff$ there exists a string y such that $|y| \in \text{Poly}(n)$ and V accepts $\langle w, y \rangle$.

Think of y as a proof or witness that $w \in L$.

Ex:
Input: a graph G
Question: does G have a Hamiltonian path?
} HP

Ex: Let φ be a Boolean formula over Boolean variables $x_1, \ldots, x_n$

Question: Is φ satisfiable?
Ie, is there a truth assignment $(a_1, \ldots, a_n) \in \{0,1\}^n$ to the variables $[x_1 := a_1, x_2 := a_2, \ldots]$ that makes φ true?

This is the SATISFIABILITY (SAT) problem.

HP & SAT are verifiable in ptime. Whether HP & SAT are decidable in ptime is an open question.

Ex: CLIQUE
Instance: A graph G and an integer K with $0 \le K \le |G.V|$.
Question: Does G have a clique of size K?

Def: NP is the class of all ptime verifiable languages.

Easy: $P \subseteq NP$:
Any ptime decider for a lang $L \in P$ can act as a verifier that ignores the proof string.

Open: $P = NP$?
Wide belief: $P \neq NP$.

Def: $\text{coNP} = \{ L \subseteq \Sigma^* : \overline{L} \in NP \}$

Polynomial reducibility

Def: A function $f : \Sigma^* \to \Sigma^*$ is ptime ($f \in FP$) if f is computable in ptime, i.e., there is a transducer that on any input w, outputs $f(w)$ in time $\text{Poly}(|w|)$.
Note: If $f \in FP$, then $\forall w$, $|f(w)| \in \text{Poly}(|w|)$.

Def: Let $A, B \subseteq \Sigma^*$ be languages. We say that A **p-reduces to** B $(A \leq_p B)$ if there exists an $f \in FP$ such that $\forall w \in \Sigma^*$, $w \in A \iff f(w) \in B$.

[i.e., $A \leq_m B$ via a ptime computable m-reduction]

Call such an f a p-reduction from A to B.

Prop: $\leq_p$ is reflexive and transitive.

Proof: Similar to the ~~sketch~~ proof for $\leq_m$. The only things to observe are
1) the identity function is in FP
2) the composition of two FP functions is in FP:
$f, g \in FP$, then $\forall w$
$(f \circ g)(w) = f(g(w))$:

"On input w:
1. Let $x := g(w)$. (ptime in $|w|$)
2. Output $f(x)$" (ptime in $|x|$)

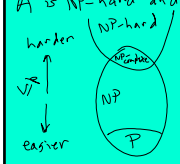
$|x| \in Poly(|w|)$, so this algo runs in time
$Poly(|w|)$ (step 1) $+ \underbrace{Poly(|x|)}_{\text{step 2}}$
$= Poly(|w|) + Poly(Poly(|w|))$
$= Poly(|w|)$. Use this to show transitivity. //

Prop: Suppose A, B are langs and $A \leq_p B$. Then
1) If $B \in P$, then $A \in P$
2) If $B \in NP$, then $A \in NP$.

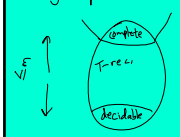
Note similarity with the Prop regarding m-reducibility

Def: A language is **NP-hard** if $\forall B \in NP$, $B \leq_p A$. [A is at least as hard as any NP lang]

A is **NP-complete** if A is NP-hard and $A \in NP$.



Analogous picture:



Def: A language L is **complete** if L is T-rec and $\forall$ T-rec B, $B \leq_m A$.

Prop: A_{TM} is complete.

Proof: Know that A_{TM} is T-rec. Given any T-rec B, let M be a fixed recognizer for B.

Define

$f :=$ "On input w:
1. Output $\langle M, w \rangle$"

For every w,

$w \in B \iff M$ accepts w

$\iff \langle M, w \rangle \in A_{TM}$

$\iff f(w) \in A_{TM}$,

So $B \leq_m A$ via f. //