Lecture

Polynomial Hierarchy.

We will characterize the same thing three ways, and hopefully prove just one theorem.

Recall P is the class of languages we characterize as having a machine M which decides in polynomial time.

$$M(w) \text{ accepts} \iff w \in L.$$

Recall NP is the class of languages verifiable in polynomial time, or decidable in nondeterministic polytime.

$$\exists x \, M(w, x) \text{ accepts} \iff w \in L.$$

You may either think of x as the witness of a deterministic ~~determiner~~ verifier or as a sequence of decisions or guesses that a nondeterministic machine makes. Recall that an NTM accepts if there exists an computation branch. You may also Recall that SAT is NP-complete, and we say that $\phi \in SAT$ if there exists ($\exists$) an ~~accepting~~ satisfying assignment of the boolean variables of ϕ.

Recall coNP is the class of languages such that $\bar{L} \in NP \iff L \in coNP$. We can take the logical complement of the definition of NP for a definition of coNP as

$$\forall x \, M(w, x) \text{ accepts} \iff w \in \bar{L}$$

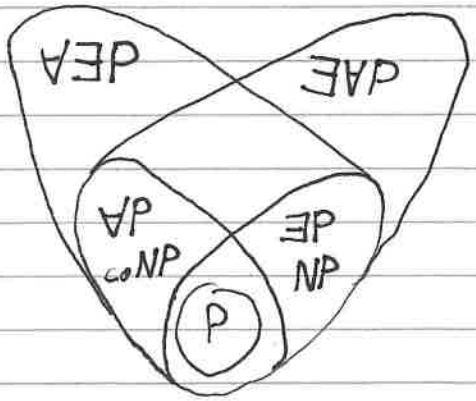
Like SAT is NP-complete, coNP has its own complete problem called tautologies. $TAUT = \{ \phi \mid$ every assignment of ϕ is satisfying $\}$. We observe that NP and coNP have this interesting duality. A yin-yang structure emerges. NP is characterized by the existential quantifier $\exists$. Like how $\exists$ a witness, $\exists$ an assignment of SAT or $\exists$ an accepting computation. coNP is characterized by the universal quantifier $\forall$. Like how $\forall$ witnesses, $\forall$ assignments of TAUT or $\forall$ ~~accepting~~ computation branches must be accepting. This duality motivates our discussion.

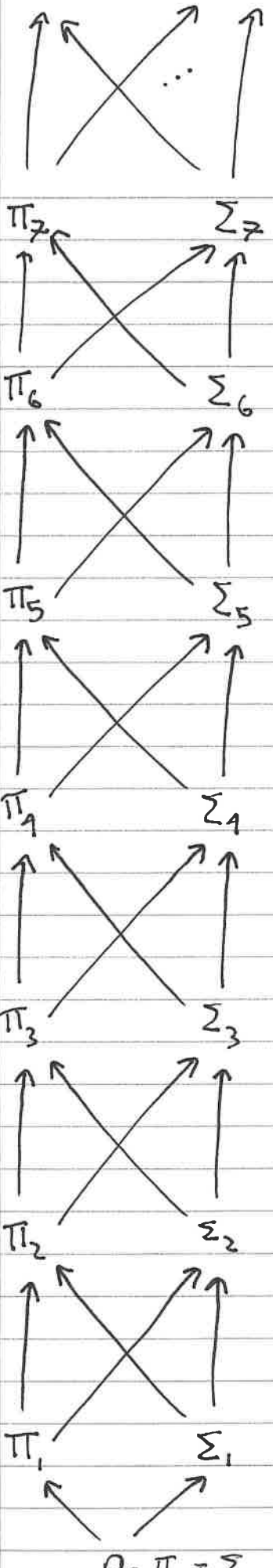
For any class C, define the class $\exists C$ such that if M was a C machine with definition (i.e. $M(w)$ accepts $\Leftrightarrow w \in L \in C$ then M' is a $\exists C$ machine such that $\exists x\, M(w,x)$ accepts $\Leftrightarrow w \in L \in \exists C$. We naturally augment these deciders to take on witnesses. Similarly define $\forall C$.

What is $\exists P$? $\exists x\, M(w,x)$ accepts $\Leftrightarrow w \in L$ and M runs in poly time. M is just a polytime verifier! so $\exists P$ = class of languages verifiable in polytime. so $\exists P = NP$. Similiarly $\forall P = coNP$.

What is $\exists\exists P$? $\exists x_1\, \exists x_2\, M(w, x_1, x_2)$ accepts and M is polytime? Thats just two witnesses. It just complicates things. Each witness can be at most a polynomial number of bits anyway so two witnesses is just a constant more. so $\exists\exists P = \exists P$. Similiarly $\forall\forall P = \forall P$. Anytime we have a sequence of the same quantifier, we may compress them to one quantifier. $\exists^k P = \exists P = NP$.

What about $\exists\forall P$ and $\forall\exists P$? Now things are getting interesting. $\exists\forall P := \exists x_1 \forall x_2\, M(w, x_1, x_2)$ accepts. Note that we may add an $\exists$ quantifier to a $\forall P$ machine which it ignores to show $\forall P \subseteq \exists\forall P$. Similiarly since we are ignoring it, $\exists P \subseteq \exists\forall P$. Similiarly for $\forall\exists P$. Does $\exists\forall P = \forall\exists P$? We don't know! $\exists\forall P$ and $\forall\exists P$ appear to have the same structure that NP and $coNP$ have.





We may repeat this argument on $\forall\exists P$ and $\exists\forall P$ as we did on $\exists P$ and $\forall P$. We see by an inductive or recursive argument this creates an infinite alternating hierarchy. ~~defined~~ A countably many more generalizations of NP and coNP and their duals/ties.

Let $\Pi_0 = \Sigma_0 = P$ and define

$$\Sigma_i = \exists \Pi_{i-1} = \underbrace{\exists\forall\exists\forall\ldots}_{i} P$$

$$\Pi_i = \forall \Sigma_{i-1} = \underbrace{\forall\exists\forall\exists\ldots}_{i} P$$

It is important that the first quantifier of Σ_i is existential, and the first quantifier of Π_i is universal. We define the polynomial hierarchy to be the class $PH = \bigcup_{i=0}^{\infty} \Sigma_i = \bigcup_{i=0}^{\infty} \Pi_i$. We define a "level" of the polynomial hierarchy to be $\Pi_i \cap \Sigma_i$ for some i.

Note that by ~~a~~ similar argument, we see

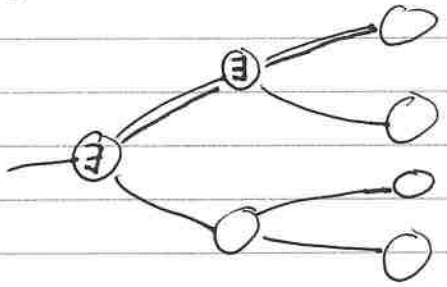
$\forall i$	$\Pi_i \subseteq \Pi_{i+1}$	we simply
$\forall i$	$\Sigma_i \subseteq \Sigma_{i+1}$	generalize the
$\forall i$	$\Sigma_i \subseteq \Pi_{i+1}$	argument we made
$\forall i$	$\Pi_i \subseteq \Sigma_{i+1}$.	for NP and coNP

Whether or not these levels are strict is an open problem. Each level is defined only using finitely many quantifiers, so it would appear that $PH \subseteq PSPACE$, since TQBF is a PSPACE-complete problem. If that containment is strict, is also an open problem. We would hope to show it is since $P \subseteq PH \subsetneq PSPACE \Rightarrow P \neq PSPACE$. Truly a beautiful class with beautiful structure of mostly theoretical interest.

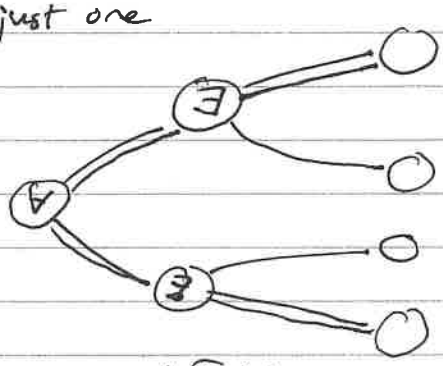
We give two more quick equivalent characterizations of the polynomial hierarchy. First is using oracles.

$$\Sigma_0 = P,\ \Sigma_1 = NP,\ \Sigma_2 = NP^{NP},\ \Sigma_3 = NP^{NP^{NP}} \qquad \Sigma_k = NP^{NP^{\cdots^{NP}}} \ (k \text{ times})$$

and $\Pi_k = co\text{-}\Sigma_k$. We only mention this and leave it unjustified but it is perhaps believable. The second characterization uses a generalized non-deterministic Turing machine called an alternating Turing machine. While a nondeterministic Turing accepts if just one branch accepts, an alternating machine may pick from two transition functions at any step if it wants to require all branches to accept or just one



NTM



ATM

For AP = alternating polynomial time, since TQBF is PSPACE-complete, and an unbounded alternating polytime machine can simulate a TQBF problems alternating quantifiers. Like how SAT is NP-complete, TQBF is AP-complete, and so AP = PSPACE.

We say a Σ_i-machine is one in which the first branching is an existential one, and there are at most i existential or universal branching steps. We similarly define a Π_i-machine as an ATM which the first branching is a universal one, and there are at most i universal or existential branching steps (to depth i). Convince yourself that NP = Σ_1-TIME(poly), as you reformulate many guesses into just one.

for each i then $\Sigma_i = \bigcup_{k=0}^{\infty} \Sigma_i\text{-TIME}(n^k)$

$\Pi_i = \bigcup_{k=0}^{\infty} \Pi_i\text{-TIME}(n^k)$

Although the polynomial hierarchy may seem of a flamboyant, unapplicable interest, like other parts of complexity, there are deep connections and ties. It may also be used to seperate the complexity classes worth studying. Also it looks cool.

If for any i $\Pi_i = \Sigma_i$ then $PH \subseteq \Pi_i = \Sigma_i$. The polynomial hierarchy collapses to the ith level. Each Π_i, Σ_i behave like a struct or pillar, supporting the levels above them. Here in the case the two distinct pillars were the same, our tower of Babel collapses. We prove a weaker thm, that if $P=NP$ then $PH \subseteq P$. That is, if $\Sigma_0 = \Sigma_1$, then the entire hierarchy collapses to Σ_0.

Assume $P=NP$, we proceed by induction. For $i=1$, $\Pi_i = coNP$ and $\Sigma_i = NP$ are both $\subseteq P$ by assumption. Now suppose that $\Pi_{i-1}, \Sigma_{i-1} \subseteq P$. We prove $\Sigma_i \subseteq P$. Since P is closed under complement and $co\Sigma_i = \Pi_i$, this will also prove $\Pi_i \subseteq P$ as desired. Let $L \in \Sigma_i$ then

$$w \in L \Leftrightarrow \underbrace{\exists x_1 \forall x_2 \ldots}_{i \text{ times.}} M(w, x_1, x_2, \ldots) \text{ accepts}$$

define $L' = \{\langle w, x_1 \rangle \mid \forall x_2 \exists x_3 \ldots M(w, x_1, x_2 \ldots) \text{ accepts}\}$

Notice $L' \in \Pi_{i-1}$ since $\uparrow$ occurs $i-1$ times beginning with $\forall$.
By our assumption, $\Pi_{i-1} \subseteq P$ so $L' \in P$. So $\exists M'$ such that

$w \in L' \Leftrightarrow M'(w)$ accepts. We may plug this into the original definition of L to get

$$w \in L \Leftrightarrow \exists x_1 M'(w, x) \text{ accepts.}$$

This is a polytime verifier so $L \in NP$, since $P=NP$ by assumption, we get $L \in P$. Since L was any Σ_i sentence, we see $\Sigma_i \subseteq P$.
So if $P=NP$, we may conclude that the polynomial hierarchy collapses to its zeroth level. To P, to ashes.

Our last theorem of the class: Karp-Lipton. Interpreting its statement is more difficult than the actual proof. $NP \subseteq P/poly \Rightarrow PH \subset \Pi_2 \cup \Sigma_2$. If SAT has a polynomial sized circuit family, then we do not have a polynomial hierarchy, it collapses to the second level! Originally proved by Karp and Lipton a collapse to the third level, Sipser improved it to the second level. The proof idea is we will show $\Pi_2 \subseteq \Sigma_2$. conversion of any $\forall\exists$-sentence to a $\exists\forall$-sentence means for any sentence higher in the hierarchy, we can repeatedly alternate and compress quantifiers until a sentence with many quantifiers only has two.

Let $NP \subseteq P/poly$, then SAT has a polynomial sized circuit family $\{C_0, C_1, \ldots\}$. Each C_i takes as input an i-variable formula and outputs a single bit for yes/no if the input formula was satisfiable. By a search to decision transformation, there exists a circuit family $\{C_0', C_1', \ldots\}$ where each C_i' outputs i bits for not just if it was satisfiable or not, but the assignment itself. Since each C_i is polynomially sized, so is each C_i'.

Let $L \in \Pi_2$. So $w \in L \Leftrightarrow \forall x_1 \exists x_2 M(w, x_1, x_2)$ accepts.

We may convert M to a CNF φ_M using the Cook-Levin style construction. Note since M runs in a polynomial number of steps, φ_M is polynomially sized, and its construction takes polynomial time. Now notice $\exists k$ such that $C_k'(\varphi_M, x_1) = x_2$. Since SAT has a polynomial sized circuit family, there exists a polysized circuit to search for this witness instead of quantifying over it. so the formula has an equivalent statement:

$$\forall x_1 \exists x_2 M(w, x_1, x_2) \text{ accepts} \Leftrightarrow \exists C_k' \forall x_1 M(w, x_1, C_k'(\varphi_M, x_1)) \text{ accepts.}$$

where we may simply use existential quantification to guess the C_k' circuit. We converted a Π_2 sentence into a Σ_2 one! $L \in \Pi_2 \Rightarrow L \in \Sigma_2 \Rightarrow \Pi_2 \subseteq \Sigma_2$. We observe $NP \subseteq P/poly$ collapses PH.

I want to conclude with some advice on how to self study complexity Theory.

First, finish the Sipser book. It ~~contains~~ doesn't contain everything but it does contain the best proofs of what it does cover. I wish it had a second volume. Its coverage of randomness, interaction, cryptography, and more may surprise you. Before you go further, you should finish Sipser.

After you finish Sipser, go through the first six chapters of the Arora-Barak book. It might be a typographic disaster but the material coverage is decent. All the other chapters in it cover an incredible breadth of material, and good pointers to other sources. These may include communication complexity, quantum complexity, the complexity of counting, and so on. Each of these chapters deserves (and has) their own books, but it's a good introduction to those theorems.

Then go through Goldreich's and Papadimitriou's books. Goldreich also has 400 pages of incredible notes.

Finally, I recommend the books "on the nature of computation" and Wigderson's "Mathematics and computation". Both of these are light on proofs, as a trade-off of caring with incredible wisdom. If you want a proof, use the other books. If you want to know what a proof really means, use Wigderson's book.

Finally, you may use me as a resource. If you have any questions, or come across anything in your own independent study, I would be happy to help and answer.

Thank you for taking my class. I had a lot of fun.