CS 6810: Theory of Computing **Spring 2026**

Lecture 5: February 5, 2026

Lecturer: Eshan Chattopadhyay *Scribe: Michael Ngo*

$$\begin{tikzcd}
\bullet & & \bullet \\
\Sigma_3^{\mathsf{P}} = \exists\forall\exists\mathsf{P} \arrow[u] \arrow[urr] & & \Pi_3^{\mathsf{P}} = \forall\exists\forall\mathsf{P} \arrow[u] \arrow[ull] \\
\Sigma_2^{\mathsf{P}} = \exists\forall\mathsf{P} \arrow[u] \arrow[urr] & & \Pi_2^{\mathsf{P}} = \forall\exists\mathsf{P} \arrow[u] \arrow[ull] \\
\Sigma_1^{\mathsf{P}} = \exists\mathsf{P} \arrow[u] \arrow[urr] & & \Pi_1^{\mathsf{P}} = \forall\mathsf{P} \arrow[u] \arrow[ull] \\
& \mathsf{P} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 1: The Polynomial Hierarchy. Arrows denote inclusion.

1 Exist and For All Complexity Classes

★ We start by defining ways to create complexities classes from other complexity classes. First, $\exists C$ is the complexity class that captures applying a non-deterministic ("there exists a certificate") over deciders for C.

Definition 1.1 (Exist Class)**.** *Let C be a complexity class. Define the complexity class $\exists C$ where $L \in \exists C$ if there exists some $L' \in C$ such that the following holds: $x \in L$ if and only if there exists $y \in \{0,1\}^{\text{poly}(|x|)}$ such that $(x, y) \in L'$.*

★ So, it should not be surprising that $\exists\mathsf{P}$ recovers NP! Simply unroll the definition of $\exists P$ and match the definition to that of NP.

Claim 1.2. $\exists\mathsf{P} = \mathsf{NP}$.

Proof. $L \in \exists\mathsf{P}$ represents the NP language; $L' \in \mathsf{P}$ represents the language of a poly-time verifier; y is the poly-sized certificate; and x is the input instance. □

★ In the same way, we define $\forall C$ as the class that captures applying a "for all certificates" quantifier over deciders for C.

Definition 1.3 (For All Class)**.** *Let C be a complexity class. Define the complexity class $\forall C$ where $L \in \forall C$ if there exists some $L' \in C$ such that the following holds: $x \in L$ if and only if for all $y \in \{0,1\}^{\text{poly}(|x|)}$, $(x, y) \in L'$.*

★ Since co-NP is exactly about satisfying a property for all poly-sized certificates, it is also not surprising that ∀P recovers co-NP!

Claim 1.4. $\forall\mathsf{P} = \text{co-}\mathsf{NP}$.

Proof. One can define co-NP in these two equivalent ways:

- $L \in \text{co-}\mathsf{NP}$ if and only if $\overline{L} \in \mathsf{NP}$;
- there exists a poly-time TM R such that $x \in L$ if and only if for all $y \in \{0,1\}^{\text{poly}(|x|)}$, $R(x,y) = 1$.

The latter definition is equivalent to the definition of $\forall\mathsf{P}$. □

★ We know SAT is an NP-complete language. What languages are co-NP-complete? We can modify SAT for this purpose as follows.

Definition 1.5. *Let* TAUTOLOGY *be the language containing all boolean formulas ϕ such that for all assignments x, $\phi(x)$ is true.*

★ Because of the " for all assignments" condition, it is not surprising that TAUTOLOGY is in co-NP. Furthermore, one can apply the same style of reduction as Cook-Levin to show that it is a complete problem for co-NP.

Claim 1.6. TAUTOLOGY $\in$ co-NP.

Claim 1.7. TAUTOLOGY *is* co-NP-complete.

★ Here's a language that is slightly weirder than just those in NP and co-NP.

Definition 1.8. *Let* MIN-DNF *be the language containing all DNFs ϕ such that all DNFs smaller ϕ compute a different function than ϕ.*

★ Notice MIN-DNF has two quantifiers, for all and exists. Therefore, we should suspect it belongs to the complexity class $\forall\exists\mathsf{P} = \forall\mathsf{NP}$.

Claim 1.9. MIN-DNF $\in \forall\mathsf{NP}$.

Proof. By definition, $\phi \in$ MIN-DNF is equivalent to the following: for all ψ such that $|\psi| < |\phi|$, there exists an x such that $\psi(x) \neq \phi(x)$. We can define a language $L' \in \mathsf{NP}$ such that $(\phi, \psi) \in L'$ if and only if there exists an x such that $\psi(x) \neq \phi(x)$.
Then $\phi \in$ MIN-DNF is equivalent to: for all ψ such that $|\psi| < |\phi|$, $(\phi,\psi) \in L'$. Since $|\psi| < |\phi|$, the size of $|\psi|$ is polynomial in the input length. Thus MIN-DNF $= \forall\{L'\} \subseteq \forall\mathsf{NP}$. □

2 The Polynomial Hierarchy

★ In MIN-DNF, we observed a chaining of ∀ and ∃. If we keep chaining these operators together, what complexity class will we achieve? This is the notion of the polynomial hierarchy.

Definition 2.1 (Polynomial Hierarchy)**.** *Define the 0-th level: $\Sigma_0^{\mathsf{P}} = \mathsf{P} = \Pi_0^{\mathsf{P}}$. We inductively define the i-th level where $\Sigma_i^{\mathsf{P}} = \exists\Pi_{i-1}^{\mathsf{P}}$ and $\Pi_i^{\mathsf{P}} = \forall\Sigma_{i-1}^{\mathsf{P}}$. Then the polynomial hierarchy is $\mathsf{PH} = \bigcup_{i\geq 0}\Sigma_i^{\mathsf{P}} = \bigcup_{i\geq 0}\Pi_i^{\mathsf{P}}$.*

★ We start by observing basic relationships about the polynomial hierarchy. First, we observe that $\exists$ and $\forall$ preserve complexity classes. Then, we observe an inclusion between Σ_i^{P} and Π_{i+1}^{P}. Then an inclusion between Σ_i^{P} and $\Sigma_{i+1}^{\mathsf{P}}$. This allows us to build a picture of hierarchy in Figure 1.

Claim 2.2. *For any complexity class C, $\exists C \supseteq C$ and $\forall C \supseteq C$.*

Proof. For the $\exists$ and $\forall$ complexity classes, if we ignore the certificate y (set its length to 0), then we can easily show $L \in C$ implies $L \in \exists C$ and $L \in \forall C$. $\square$

Claim 2.3. *For all $i \geq 0$, $\Sigma_i^{\mathsf{P}} \subseteq \Pi_{i+1}^{\mathsf{P}}$ and $\Pi_i^{\mathsf{P}} \subseteq \Sigma_{i+1}^{\mathsf{P}}$.*

Proof. Observe that $\Pi_{i+1}^{\mathsf{P}} = \forall \Sigma_i^{\mathsf{P}}$ and $\Sigma_{i+1}^{\mathsf{P}} = \exists \Pi_i^{\mathsf{P}}$. The rest follows by Claim 2.2. $\square$

Claim 2.4. *For all $i \geq 0$, $\Sigma_i^{\mathsf{P}} \subseteq \Sigma_{i+1}^{\mathsf{P}}$ and $\Pi_i^{\mathsf{P}} \subseteq \Pi_{i+1}^{\mathsf{P}}$.*

Proof. By Claim 2.2, we have $\Pi_0^{\mathsf{P}} \subseteq \forall \Pi_0^{\mathsf{P}}$ and $\Pi_0^{\mathsf{P}} \subseteq \exists \Pi_0^{\mathsf{P}}$. Applying this to the unrolled definition of $\Sigma_{i+1}^{\mathsf{P}}$ and Π_{i+1}^{P} obtains the claim. $\square$

★ Recall our priors claims to characterize Σ_1^{P} and Π_1^{P} as NP and co-NP.

Claim 2.5. $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ *and* $\Pi_1^{\mathsf{P}} = \text{co-}\mathsf{NP}$.

Proof. By Claim 1.2 and Claim 1.4, $\Sigma_1^{\mathsf{P}} = \exists \Pi_0^{\mathsf{P}} = \exists \mathsf{P} = \mathsf{NP}$ and $\Pi_1^{\mathsf{P}} = \forall \Sigma_0^{\mathsf{P}} = \forall \mathsf{P} = \text{co-}\mathsf{NP}$. $\square$

★ Next, we show that each level of the hierarchy are complements. This follows by unrolling the definition and showing that negating a language flips all the quantifiers.

Claim 2.6. *For all $i \geq 0$, $\text{co-}\Sigma_i^{\mathsf{P}} = \Pi_i^{\mathsf{P}}$.*

Proof. We proceed by induction on i. This holds for $i = 0$ since $\text{co-}\mathsf{P} = \mathsf{P}$, and for $i = 1$ by Claim 2.5. Now we suppose $i \geq 2$. We want to show that $L \in \Sigma_i^{\mathsf{P}}$ if and only if $\overline{L} \in \Pi_i^{\mathsf{P}}$.
Let $L \in \Sigma_i^{\mathsf{P}}$. Then there exists an $L' \in \Pi_{i-1}^{\mathsf{P}}$ such that $x \in L$ if and only if there exists a y such that $(x, y) \in L'$. Negating this, there exists an $\overline{L'} \in \text{co-}\Pi_{i-1}^{\mathsf{P}}$ such that $x \in L$ if and only if for all y, $(x, y) \in \overline{L'}$.
Applying the inductive hypothesis that $\text{co-}\Pi_{i-1}^{\mathsf{P}} = \Sigma_{i-1}^{\mathsf{P}}$, we have that there exists an $\overline{L'} \in \Sigma_{i-1}^{\mathsf{P}}$ such that $x \in L$ if and only if for all y, $(x, y) \in \overline{L'}$. Which exactly fits the definition that $\overline{L} \in \Pi_i^{\mathsf{P}}$. The same reasoning works in the converse. $\square$

3 Making the Hierarchy Collapse

★ The polynomial hierarchy contains this massive collection of (perhaps) increasingly hard complexity classes. So when someone has a cryptographic result such as "assuming X, the polynomial heirarchy collapses (that is $\mathsf{PH} = \mathsf{P}$)", it provides strong evidence that X is false since it is widely believed that $\mathsf{PH} \neq \mathsf{P}$. Although, we have no proof techniques to prove this separation, and are still not close to proving even $\mathsf{P} \neq \mathsf{PSPACE}$.

★ We provide two results of collapsing. The former is that if the bottom part collapses, that is $\mathsf{P} = \mathsf{NP}$, then the whole hierarchy collapses to P. The latter is that if a higher level i collapses into itself, then the hierarchy collapses at that level. To show these, they make use of the complementation claim from above.

Claim 3.1. *If* $\mathsf{P} = \mathsf{NP}$, *then* $\mathsf{PH} = \mathsf{P}$.

Proof. We show that for all $i \geq 0$, $\Sigma_i^{\mathsf{P}} = \mathsf{P}$, assuming $\mathsf{P} = \mathsf{NP}$. We proceed by induction on i. For $i = 0$, we know $\Sigma_0^{\mathsf{P}} = \mathsf{P}$ by definition. For $i = 1$, we know $\mathsf{P} = \mathsf{NP} = \Sigma_1^{\mathsf{P}}$ by assumption. Now suppose that $i \geq 2$ and $\Sigma_{i-1}^{\mathsf{P}} = \mathsf{P}$. Applying Claim 2.6, $\Pi_{i-1}^{\mathsf{P}} = \text{co-}\Sigma_{i-1}^{\mathsf{P}} = \text{co-}\mathsf{P} = \mathsf{P}$. Therefore, $\Sigma_i^{\mathsf{P}} = \exists\Pi_{i-1}^{\mathsf{P}} = \exists\mathsf{P} = \mathsf{NP} = \mathsf{P}$. By induction, for all $i \geq 0$, $\Sigma_i^{\mathsf{P}} = \mathsf{P}$. It follows that $\mathsf{PH} = \bigcup_{i \geq 0} \Sigma_i^{\mathsf{P}} = \mathsf{P}$. □

Claim 3.2. *For any fixed* $i \geq 1$, *if* $\Sigma_i^{\mathsf{P}} = \Pi_i^{\mathsf{P}}$, *then* $\mathsf{PH} = \Sigma_i^{\mathsf{P}}$.

Proof. We sketch a similar approach to the prior claim. We show by induction on $k \geq i$ that $\Sigma_k^{\mathsf{P}} = \Sigma_i^{\mathsf{P}} = \Pi_k^{\mathsf{P}}$. Note that $\Sigma_k^{\mathsf{P}} = \exists\Pi_{k-1}^{\mathsf{P}} = \exists\Sigma_{k-1}^{\mathsf{P}} = \Sigma_{k-1}^{\mathsf{P}}$ by the inductive hypothesis. Similarly, we can show $\Pi_k^{\mathsf{P}} = \forall\Sigma_{k-1}^{\mathsf{P}} = \forall\Pi_{k-1}^{\mathsf{P}} = \Pi_{k-1}^{\mathsf{P}}$. Thus, $\mathsf{PH} = \bigcup_{\ell \geq 0} \Sigma_\ell^{\mathsf{P}} = \Sigma_i^{\mathsf{P}}$. □

4 Complete Problems

★ Do there exist complete problems for these alternating complexity classes? Yes! All we need to do is modify SAT with the right sequence of quantifiers, and proof follows by the same style as Cook-Levin.

Definition 4.1. *Let* $\Sigma_i^{\mathsf{P}}\text{-SAT}$ *be the language containing all formulas* ϕ *such that* $\exists x^{(1)}\ \forall x^{(2)}\ \cdots\ Q_i x^{(i)}$ *such that* $\phi(x^{(1)}, x^{(2)}, \ldots, x^{(i)})$ *is true. Here,* Q_i *is a quantifier:* $\exists$ *when* i *is odd, and* $\forall$ *when* i *is even.*
Similarly, define $\Pi_i^{\mathsf{P}}\text{-SAT}$ *as the same language, but with quantifiers* $\exists$ *and* $\forall$ *flipped.*

Claim 4.2. $\Sigma_i^{\mathsf{P}}\text{-SAT}$ *is complete for* Σ_i^{P}, *and* $\Pi_i^{\mathsf{P}}\text{-SAT}$ *is complete for* Π_i^{P}.

5 Formulation via Oracle TMs

★ We can present an alternative formulation of these alternating classes via oracle TMs. For example, using non-deterministic TMs, we can simulate $\exists C$ with an oracle for C.

Claim 5.1. $\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$ *and* $\Pi_{i+1}^{\mathsf{P}} = \text{co-}\mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$

Proof. Roughly, NP allows us to simulate an $\exists$ class operator whereas co-NP allows us to simulate a $\forall$ class operator. Recall by Claim 2.6 that Σ_i^{P} and Π_i^{P} are complements of each other. Therefore, an oracle for Σ_i^{P} is equivalent to an oracle for Π_i^{P}, which is why we can write Σ_i^{P} as the oracle for both cases. □