Lecture 5 - Polynomial Hierarchy, Alternating TMs, Time-Space

Rafael Oliveira
rafael.oliveira.teaching@gmail.com
University of Waterloo

CS 860 - Graduate Complexity Theory
Fall 2022

Overview

- Polynomial Hierarchy (PH)
- Time vs Alternations: time-space tradeoffs

Some Problems of Interest

- Often times interested in problems of the form: "find the smallest/largest object with certain property"

Some Problems of Interest

- Often times interested in problems of the form: "find the smallest/largest object with certain property"
- MIN-EQ-DNF : $\{\langle \varphi, k \rangle \mid \exists$ DNF ψ of size $\leq k$ s.t. $\psi \equiv \varphi\}$

Some Problems of Interest

- Often times interested in problems of the form: "find the smallest/largest object with certain property"
- MIN-EQ-DNF : $\{\langle\varphi, k\rangle \mid \exists$ DNF ψ of size $\leq k$ s.t. $\psi \equiv \varphi\}$
- EXACT-INDSET $:= \{\langle G, k\rangle \mid$
  G has largest independent set of size $k\}$

Some Problems of Interest

- Often times interested in problems of the form: "find the smallest/largest object with certain property"
- MIN-EQ-DNF : $\{\langle \varphi, k \rangle \mid \exists$ DNF ψ of size $\leq k$ s.t. $\psi \equiv \varphi\}$
- EXACT-INDSET $:= \{\langle G, k \rangle \mid$ G has largest independent set of size $k\}$
- There isn't obvious notion (or none at all) of a small/efficient certificate for the problems above
- Seem to need both $\exists, \forall$ quantifiers

Some Problems of Interest

- Often times interested in problems of the form: "find the smallest/largest object with certain property"
- MIN-EQ-DNF : $\{\langle\varphi, k\rangle \mid \exists$ DNF ψ of size $\leq k$ s.t. $\psi \equiv \varphi\}$
- EXACT-INDSET $:= \{\langle G, k\rangle \mid$ G has largest independent set of size $k\}$
- There isn't obvious notion (or none at all) of a small/efficient certificate for the problems above
- Seem to need both $\exists, \forall$ quantifiers

Definition

The class Σ_2^p is the set of languages $L \subseteq \{0,1\}^*$ such that there is poly-time TM M and polynomial q such that

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} \forall v \in \{0,1\}^{q(|x|)}\ M(x,u,v) = 1.$$

Polynomial Hierarchy (PH)

Definition (Polynomial Hierarchy)

For $i \geq 1$, a language $L \subseteq \{0,1\}^*$ is in Σ_i^p if there is a poly-time TM M and a polynomial $q : \mathbb{N} \to \mathbb{N}$ s.t.

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \\ M(x, u_1, \ldots, u_i) = 1.$$

where $Q_i = \exists$ iff i odd.

The polynomial hierarchy is the set $\mathsf{PH} := \bigcup_i \Sigma_i^p$.

Polynomial Hierarchy (PH)

Definition (Polynomial Hierarchy)

For $i \geq 1$, a language $L \subseteq \{0,1\}^*$ is in Σ_i^p if there is a poly-time TM M and a polynomial $q : \mathbb{N} \to \mathbb{N}$ s.t.

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \\ M(x, u_1, \ldots, u_i) = 1.$$

where $Q_i = \exists$ iff i odd.

The polynomial hierarchy is the set PH $:= \bigcup_i \Sigma_i^p$.

Can also define $\Pi_i^p := \{\overline{L} \mid L \in \Sigma_i^p\}$. Equivalently, $L \in \Pi_i^p$ iff:

$$x \in L \Leftrightarrow \forall u_1 \in \{0,1\}^{q(|x|)} \exists u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \\ M(x, u_1, \ldots, u_i) = 1.$$

where we alternate the quatifiers.

PH in terms of oracles

Theorem

For every $i \geq 2$, $\Sigma_i^p = NP^{\Sigma_{i-1}^p}$.

- Will prove statement above for $i = 2$.

PH in terms of oracles

Theorem

For every $i \geq 2$, $\Sigma_i^p = \mathsf{NP}^{\Sigma_{i-1}^p}$.

- Will prove statement above for $i = 2$.
- Want to show $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{SAT}}$

$\Sigma_2^p \subseteq \mathsf{NP}^{\mathsf{NP}}$

1. $L \in \Sigma_2^p$, then for $p : \mathbb{N} \to \mathbb{N}$ polynomial and poly-time TM V

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{p(|x|)} \forall u_2 \in \{0,1\}^{p(|x|)} V(x, u_1, u_2).$$

$\Sigma_2^p \subseteq \mathsf{NP}^{\mathsf{NP}}$

1. $L \in \Sigma_2^p$, then for $p : \mathbb{N} \to \mathbb{N}$ polynomial and poly-time TM V

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{p(|x|)} \forall u_2 \in \{0,1\}^{p(|x|)} V(x, u_1, u_2).$$

2. Construction of NTM $M \in \mathsf{NP}^{\mathsf{NP}}$
   - On input $x \in \{0,1\}^n$, M guesses $u_1 \in \{0,1\}^{p(n)}$ and asks NP oracle:

$$\exists u_2 \in \{0,1\}^{p(n)} V(x, u_1, u_2) = 0$$

$\Sigma_2^p \subseteq \mathsf{NP}^{\mathsf{NP}}$

1. $L \in \Sigma_2^p$, then for $p : \mathbb{N} \to \mathbb{N}$ polynomial and poly-time TM V

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{p(|x|)} \forall u_2 \in \{0,1\}^{p(|x|)} V(x, u_1, u_2).$$

2. Construction of NTM $M \in \mathsf{NP}^{\mathsf{NP}}$
   - On input $x \in \{0,1\}^n$, M guesses $u_1 \in \{0,1\}^{p(n)}$ and asks NP oracle:
     $$\exists u_2 \in \{0,1\}^{p(n)} V(x, u_1, u_2) = 0$$
   - By Cook-Levin and V being a TM, above query can be described by a SAT instance

$\Sigma_2^p \subseteq \mathsf{NP}^{\mathsf{NP}}$

1. $L \in \Sigma_2^p$, then for $p : \mathbb{N} \to \mathbb{N}$ polynomial and poly-time TM V

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{p(|x|)} \forall u_2 \in \{0,1\}^{p(|x|)} V(x, u_1, u_2).$$

2. Construction of NTM $M \in \mathsf{NP}^{\mathsf{NP}}$
   - On input $x \in \{0,1\}^n$, M guesses $u_1 \in \{0,1\}^{p(n)}$ and asks NP oracle:

$$\exists u_2 \in \{0,1\}^{p(n)} V(x, u_1, u_2) = 0$$

   - By Cook-Levin and V being a TM, above query can be described by a SAT instance
   - Thus M accepts $\Leftrightarrow x \in L$

$\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.

$\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.
2. There is $p : \mathbb{N} \to \mathbb{N}$ polynomial such that on input $x \in L \cap \{0,1\}^n$:

$\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.
2. There is $p : \mathbb{N} \to \mathbb{N}$ polynomial such that on input $x \in L \cap \{0,1\}^n$:
3. Let NTM M' be such that on input x,
   - M' guesses an accepting path and all oracle queries of $M(x)$ with their respective answers

$\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.
2. There is $p : \mathbb{N} \to \mathbb{N}$ polynomial such that on input $x \in L \cap \{0,1\}^n$:
3. Let NTM M' be such that on input x,
   - M' guesses an accepting path and all oracle queries of $M(x)$ with their respective answers
   - For each oracle query with YES answer, M' guesses a satisfying assignment

NP$^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.
2. There is $p : \mathbb{N} \to \mathbb{N}$ polynomial such that on input $x \in L \cap \{0,1\}^n$:
3. Let NTM M' be such that on input x,
   - M' guesses an accepting path and all oracle queries of $M(x)$ with their respective answers
   - For each oracle query with YES answer, M' guesses a satisfying assignment
   - M' left with the questions with NO answer, say $\phi_1, \ldots, \phi_k$ formulae

     M' asks its oracle whether $\phi_1 \vee \cdots \vee \phi_k$ is satisfiable

     Returns YES $\Leftrightarrow$ answer to the above is NO

$\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p$

1. $L \in \mathsf{NP}^{\mathsf{SAT}}$. Thus, there is oracle NTM M which decides L in poly-time.
2. There is $p : \mathbb{N} \to \mathbb{N}$ polynomial such that on input $x \in L \cap \{0,1\}^n$:
3. Let NTM M' be such that on input x,
   - M' guesses an accepting path and all oracle queries of $M(x)$ with their respective answers
   - For each oracle query with YES answer, M' guesses a satisfying assignment
   - M' left with the questions with NO answer, say $\phi_1, \ldots, \phi_k$ formulae

     M' asks its oracle whether $\phi_1 \vee \cdots \vee \phi_k$ is satisfiable

     Returns YES $\Leftrightarrow$ answer to the above is NO
4. Conversely if $M'(x)$ has accepting computation, then $M(x) = 1$ and thus $x \in L$.

Some Facts about PH

1. Note that $\mathsf{NP} = \Sigma_1^p$ and $\mathsf{coNP} = \Pi_1^p$

Some Facts about PH

1. Note that $\mathsf{NP} = \Sigma_1^p$ and $\mathsf{coNP} = \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$

Some Facts about PH

1. Note that NP $= \Sigma_1^p$ and coNP $= \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
3. For any $i \geq 1$, Σ_i^p and Π_i^p have complete problems (under poly-time mapping reductions)

Some Facts about PH

1. Note that NP $= \Sigma_1^p$ and coNP $= \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
3. For any $i \geq 1$, Σ_i^p and Π_i^p have complete problems (under poly-time mapping reductions)
4. If $\Sigma_i^p = \Pi_i^p$ for some i, then PH $= \Sigma_i^p$.

Some Facts about PH

1. Note that $\mathsf{NP} = \Sigma_1^p$ and $\mathsf{coNP} = \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
3. For any $i \geq 1$, Σ_i^p and Π_i^p have complete problems (under poly-time mapping reductions)
4. If $\Sigma_i^p = \Pi_i^p$ for some i, then $\mathsf{PH} = \Sigma_i^p$.
5. If $\Sigma_i^p = \Sigma_{i+1}^p$ for some i then $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$ for all $j \geq i$

Some Facts about PH

1. Note that NP $= \Sigma_1^p$ and coNP $= \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
3. For any $i \geq 1$, Σ_i^p and Π_i^p have complete problems (under poly-time mapping reductions)
4. If $\Sigma_i^p = \Pi_i^p$ for some i, then PH $= \Sigma_i^p$.
5. If $\Sigma_i^p = \Sigma_{i+1}^p$ for some i then $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$ for all $j \geq i$
6. If $\Pi_i^p = \Pi_{i+1}^p$ for some i then $\Sigma_j^p = \Pi_j^p = \Pi_i^p$ for all $j \geq i$

Some Facts about PH

1. Note that NP $= \Sigma_1^p$ and coNP $= \Pi_1^p$
2. $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
3. For any $i \geq 1$, Σ_i^p and Π_i^p have complete problems (under poly-time mapping reductions)
4. If $\Sigma_i^p = \Pi_i^p$ for some i, then PH $= \Sigma_i^p$.
5. If $\Sigma_i^p = \Sigma_{i+1}^p$ for some i then $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$ for all $j \geq i$
6. If $\Pi_i^p = \Pi_{i+1}^p$ for some i then $\Sigma_j^p = \Pi_j^p = \Pi_i^p$ for all $j \geq i$
7. If PH has a complete problem, then PH collapses.

Collapse of PH

Proposition

If NP $=$ coNP then $\Sigma_i^p =$ NP for every $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP

Collapse of PH

Proposition

If NP = coNP then Σ_i^p *= NP for every* $i \geq 1$*.*

1. We first prove NP = coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)

Collapse of PH

Proposition

If NP = coNP then Σ_i^p *= NP for every* $i \geq 1$*.*

1. We first prove NP = coNP $\Rightarrow \Sigma_2^p$ = NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L

Collapse of PH

Proposition

If NP $=$ coNP then $\Sigma_i^p =$ NP for every $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP $=$ coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathsf{SAT}}$

Collapse of PH

Proposition

If NP $=$ coNP then $\Sigma_i^p =$ NP for every $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \text{NP}^{\text{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP $=$ coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\text{SAT}}$
   - Let M'' be the NTM such that on input x:

     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers

Collapse of PH

Proposition

If NP $=$ coNP then $\Sigma_i^p =$ NP for every $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \mathrm{NP}^{\mathrm{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP $=$ coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathrm{SAT}}$
   - Let M'' be the NTM such that on input x:

     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers

     1.2 for each YES answer, guess satisfying assignment

Collapse of PH

Proposition

If NP $=$ coNP then $\Sigma_i^p =$ NP for every $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP $=$ coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathsf{SAT}}$
   - Let M'' be the NTM such that on input x:
     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers
     1.2 for each YES answer, guess satisfying assignment
     1.3 for each NO answer to φ, M'' guesses accepting computation of $M'(\varphi)$

Collapse of PH

Proposition

If NP = coNP then Σ_i^p = NP for every $i \geq 1$.

1. We first prove NP = coNP $\Rightarrow \Sigma_2^p$ = NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP = coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathsf{SAT}}$
   - Let M'' be the NTM such that on input x:
     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers
     1.2 for each YES answer, guess satisfying assignment
     1.3 for each NO answer to φ, M'' guesses accepting computation of $M'(\varphi)$
   - $M(x) = 1 \Leftrightarrow M''(x) = 1$

Collapse of PH

Proposition

If NP = coNP then Σ_i^p *= NP for every* $i \geq 1$.

1. We first prove NP = coNP $\Rightarrow \Sigma_2^p$ = NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP = coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathsf{SAT}}$
   - Let M'' be the NTM such that on input x:
     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers
     1.2 for each YES answer, guess satisfying assignment
     1.3 for each NO answer to φ, M'' guesses accepting computation of $M'(\varphi)$
   - $M(x) = 1 \Leftrightarrow M''(x) = 1$
2. Prove by induction Σ_i^p = NP for $i \geq 2$.

Collapse of PH

Proposition

If NP = coNP then $\Sigma_i^p = $ *NP for every* $i \geq 1$.

1. We first prove NP $=$ coNP $\Rightarrow \Sigma_2^p =$ NP
   - $\Sigma_2^p = \mathsf{NP}^{\mathsf{SAT}}$ (earlier proposition)
   - $L \in \Sigma_2^p$ then let M be SAT-oracle poly-time NTM deciding L
   - NP $=$ coNP $\Rightarrow \exists$ NTM M' deciding $\overline{\mathsf{SAT}}$
   - Let M'' be the NTM such that on input x:
     1.1 M'' guesses an accepting path of $M(x)$ with oracle queries φ and answers
     1.2 for each YES answer, guess satisfying assignment
     1.3 for each NO answer to φ, M'' guesses accepting computation of $M'(\varphi)$
   - $M(x) = 1 \Leftrightarrow M''(x) = 1$
2. Prove by induction $\Sigma_i^p =$ NP for $i \geq 2$.
3. Assuming that $\Sigma_{i-1} =$ NP we have
   $\Sigma_i^p = \mathsf{NP}^{\Sigma_{i-1}^p} = \mathsf{NP}^{\mathsf{NP}} = \Sigma_2^p = \mathsf{NP}$

- Polynomial Hierarchy (PH)

- Time vs Alternations: time-space tradeoffs

Alternating Turing Machines (ATM)

Definition (ATM)

An alternating TM (ATM) M is a NTM where each state $q \in \mathcal{Q} \setminus \{q_{halt}, q_{accept}\}$ is labeled with a quantifier from $\{\exists, \forall\}$.
NTM M accepts input x iff:

- for each $\exists$ state, one of its transitions accepts,
- for each $\forall$ state, both of its transitions accept.

We say that M runs in time $t(n)$ if on inputs $x \in \{0,1\}^n$ M every sequence of transition function choices halts in $\leq t(n)$ steps.

Alternating Turing Machines (ATM)

Definition (ATM)

An alternating TM (ATM) M is a NTM where each state $q \in \mathcal{Q} \setminus \{q_{halt}, q_{accept}\}$ is labeled with a quantifier from $\{\exists, \forall\}$. NTM M accepts input x iff:

- for each $\exists$ state, one of its transitions accepts,
- for each $\forall$ state, both of its transitions accept.

We say that M runs in time $t(n)$ if on inputs $x \in \{0,1\}^n$ M every sequence of transition function choices halts in $\leq t(n)$ steps.

Definition (Alternating Time)

Language $L \subseteq \{0,1\}^*$ is in ATIME$(t(n))$ if there is a constant $c > 0$ and a $c \cdot t(n)$ ATM M such that $x \in L \Leftrightarrow M(x) = 1$.

Alternating Turing Machines (ATM)

Definition (Alternating Time)

Language $L \subseteq \{0,1\}^*$ is in ATIME$(t(n))$ if there is a constant $c > 0$ and a $c \cdot t(n)$ ATM M such that $x \in L \Leftrightarrow M(x) = 1$.

If for every input x and every directed path in $G_{M,x}$, M's states' quantifiers alternate at most $i-1$ times, then

- $L \in \Sigma_i$TIME$(t(n))$ if q_{start} has $\exists$
- $L \in \Pi_i$TIME$(t(n))$ if q_{start} has $\forall$

Alternating Turing Machines (ATM)

Definition (Alternating Time)

Language $L \subseteq \{0,1\}^*$ is in ATIME$(t(n))$ if there is a constant $c > 0$ and a $c \cdot t(n)$ ATM M such that $x \in L \Leftrightarrow M(x) = 1$.

If for every input x and every directed path in $G_{M,x}$, M's states' quantifiers alternate at most $i-1$ times, then

- $L \in \Sigma_i$TIME$(t(n))$ if q_{start} has $\exists$
- $L \in \Pi_i$TIME$(t(n))$ if q_{start} has $\forall$

Proposition

$$\Sigma_i^p = \bigcup_c \Sigma_i \mathit{TIME}(n^c)$$

$$\Pi_i^p = \bigcup_c \Pi_i \mathit{TIME}(n^c)$$

Time-Space tradeoff

Theorem

$$\mathit{SAT} \notin \mathit{TISP}(n^{1.1}, n^{0.1})$$

References I

Arora, Sanjeev and Barak, Boaz (2009)
Computational Complexity, A Modern Approach — Chapter 5
Cambridge University Press

Trevisan, Luca (2002)
Lecture notes — Chapter 3
See webpage

Goldreich, Oded (2006)
Computational complexity: a conceptual perspective.
https://www.wisdom.weizmann.ac.il/~oded/cc-drafts.html