CS-E4530 Computational Complexity Theory

Lecture 9: Beyond NP

Aalto University
School of Science
Department of Computer Science

Spring 2019
Agenda

- Class coNP
- Structure of P, NP and coNP
- The Polynomial Time Hierarchy
- Classes EXP and NEXP
Beyond NP

- **We have so far focused on NP-complete problems**
  - Most common and natural type of *intractable* problems
  - NP-hardness is a strong argument for establishing that there is no polynomial-time algorithm

- **There are also problems outside NP**
  - Useful to be able to recognise such problems
  - Many algorithmic techniques for NP problems do not apply
Class coNP: Definition 1

- coNP contains the *complements* of languages in NP
- Essentially problems where *no-instances* are easy to verify

*Recall:* complement of language $L$ is $\overline{L} = \{x \in \{0, 1\}^*: x \notin L\}$

**Definition**

$$\text{coNP} = \left\{ L \subseteq \{0, 1\}^*: \overline{L} \in \text{NP}\right\}$$
Definition

The class $\text{coNP}$ is the class of all languages $L \subseteq \{0, 1\}^*$ for which there exists a polynomial-time Turing machine $M$ and a polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \{0, 1\}^*$ we have $x \in L$ if and only if for all $u \in \{0, 1\}^*$ with $|u| \leq p(|x|)$ it holds $M(x, u) = 1$.

- For no-instances there is a certificate $u$ such that $M(x, u) = 0$ (may assume $M$ outputs 0/1)
coNP-completeness

Definition
We say that a language \( L \) is *coNP-complete* if \( L \in \text{coNP} \) and for any language \( L' \in \text{coNP} \), we have \( L' \leq_p L \).

Theorem

\( L \) is NP-complete if and only if \( \overline{L} \) is coNP-complete.

**Proof:** The same reductions apply in both cases.
**coNP-completeness**: Example

**TAUTOLOGY**

- **Instance**: A Boolean formula $\varphi$ (not necessarily CNF).
- **Question**: Is $\varphi$ satisfied by *all* possible assignments to its variables?

**Tautology is coNP-complete:**

- Let $L \in \text{coNP}$
- Apply the Cook–Levin reduction from $\overline{L} \in \text{NP}$ to CNF-SAT to map instance $x$ to a CNF $\varphi_x$
- Transform $\varphi_x$ to $\overline{\varphi_x}$ to get a TAUTOLOGY instance
The following are open questions:

- $P \neq NP$?
- $P \neq \text{coNP}$?
- $NP \neq \text{coNP}$?
- $P = NP \cap \text{coNP}$?

Note the following relationships:

- If $P = NP$, then $P = \text{coNP}$ (exercise)
- $NP = \text{coNP}$ *does not* imply $P = NP$
NP-intermediate problems

Theorem (R. Ladner 1975)

*If* $P \neq NP$, *then there is a language* $L \in NP \setminus P$ *that is not NP-complete.*

- No natural problem known to be NP-intermediate
- One candidate: *graph isomorphism*
Graph Isomorphism

**Instance:** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2|$.

**Question:** Is there a bijection $f : V_1 \rightarrow V_2$ such that

$$\{u, v\} \in E_1 \text{ if and only if } \{f(u), f(v)\} \in E_2 ?$$
Possible Worlds

- $P = NP = coNP$
- $NP = coNP$
- $NP \cap coNP$
- $P = NP = coNP$
- $NP = coNP$
- $NP \cap coNP$
- $P = NP = coNP$
- $NP = coNP$
Varieties of the Independent Set Problem

Maximum independent set (MaxIS)
- **Instance:** Graph $G = (V, E)$, an integer $k \geq 1$.
- **Question:** Is there an independent set of size at least $k$ in $G$?

Exact independent set (ExactIS)
- **Instance:** Graph $G = (V, E)$, an integer $k \geq 1$.
- **Question:** Is the size of the largest independent set in $G$ exactly $k$?
Varieties of the Independent Set Problem

- **Maximum independent set**
  - Does there exist an independent set $I$ with $|I| \geq k$?

- **Complement of maximum independent set**
  - Does it hold for all independent sets $I$ that $|I| < k$?

- **Exact independent set**
  - Does there exist an independent set $I$ such that for all independent sets $J$ we have $|I| \geq |J|$?

- Where are these located in our complexity universe?
Classes $\Sigma_2^p$ and $\Pi_2^p$

**Definition**

The class $\Sigma_2^p$ is the class of all languages $L \subseteq \{0, 1\}^*$ for which there exists a polynomial-time Turing machine $M$ and a polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \{0, 1\}^*$,

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

**Definition**

$$\Pi_2^p = \text{co}\Sigma_2^p = \{ L \subseteq \{0, 1\}^* : \overline{L} \in \Sigma_2^p \}$$
The Polynomial Time Hierarchy

Definition

The class $\Sigma^p_k$ is the class of all languages $L \subseteq \{0, 1\}^*$ for which there exists a polynomial-time Turing machine $M$ and a polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \{0, 1\}^*$,

$$x \in L \iff \exists u_1 \forall u_2 \cdots Q u_k M(x, u_1, u_2, \ldots, u_k) = 1,$$

where each $u_i$ ranges over binary strings of length at most $p(|x|)$ and $Q$ is either $\exists$ or $\forall$, depending on whether $k$ is odd or even.

Definition

$$\Pi^p_k = \text{co} \Sigma^p_k = \{ L \subseteq \{0, 1\}^* : \overline{L} \in \Sigma^p_k \}$$
The Polynomial Time Hierarchy

Definition (The Polynomial Time Hierarchy)

$$\text{PH} = \bigcup_{k \geq 0} \Sigma^p_k$$

Some basic properties of the polynomial time hierarchy:

- $\Sigma^p_0 = \Pi^p_0 = \text{P}$
- $\Sigma^p_1 = \text{NP}, \; \Pi^p_1 = \text{coNP}$
- $\Sigma^p_k \subseteq \Pi^p_{k+1} \subseteq \Sigma^p_{k+2}$, for all $k \geq 0$
- $\text{PH} = \bigcup_{k \geq 0} \Pi^p_k$
The Polynomial Time Hierarchy

- Generally believed that:
  - $\Sigma^p_k \neq \Sigma^p_{k+1}$ for all $k \geq 1$ ("polynomial time hierarchy does not collapse")
  - $\Sigma^p_k \neq \Pi^p_k$

- Generalised versions of $P \neq NP$ and $NP \neq \text{coNP}$

Theorem

- For $k \geq 1$, if $\Sigma^p_k = \Pi^p_k$, then $\text{PH} = \Sigma^p_k$ ("hierarchy collapses to level $k$”).
- If $P = NP$, then $P = \text{PH}$ ("hierarchy collapses to $P$”).
Complete Problems in PH

- Completeness for $\Sigma^p_k$, $\Pi^p_k$ and PH is defined in terms of polynomial-time many-one reductions

- Complete problem for $\Sigma^p_k$: $\Sigma_k\text{SAT}$
  - Satisfiability for Boolean formulas of form
    \[ \exists u_1 \forall u_2 \cdots Q u_k \varphi(u_1, u_2, \ldots, u_k), \]
  where $\varphi$ is a Boolean formula (not necessarily CNF), each $u_i$ is a tuple of variables and $Q$ is either $\exists$ or $\forall$, depending on whether $k$ is odd or even.
Complete Problems in PH

- For PH, complete problems are believed *not* to exist

**Theorem**

*If there is a PH-complete problem, then there exists* $k$ *such that*

$$PH = \Sigma^p_k.$$  

**Proof sketch:**

- Suppose $L$ is PH-complete
- Since $L \in PH$, we have $L \in \Sigma^p_k$ for some $k$
- Let $L' \in PH$. Since $L' \leq_p L$, we have $L' \in \Sigma^p_k$.
- Hence $PH \subseteq \Sigma^p_k$.  

**PH: Characterisation via Oracle TM's**

- For any given language \( L \), we define the *relativised* complexity classes:

\[
P^L = \{ L' : L' = M^L \text{ for some (deterministic) polynomial-time oracle Turing machine } M \}\]

\[
NP^L = \{ L' : L' = M^L \text{ for some nondeterministic polynomial-time oracle Turing machine } M \}\.
\]

- Furthermore, for any family of languages \( C \), we define the relativised classes:

\[
P^C = \bigcup_{L \in C} P^L \quad \text{and} \quad NP^C = \bigcup_{L \in C} NP^L.
\]

**Theorem**

*For every* \( k \geq 0 \), \( \Sigma^p_{k+1} = NP^{\Sigma^p_k} \) *and* \( \Pi^p_{k+1} = coNP^{\Sigma^p_k} \).
PH: Characterisation via Oracle TM’s (Cont’d)

- It is also customary to define the following “deterministic” classes in the polynomial-time hierarchy:

  \[ \Delta^p_0 = \text{P}, \quad \Delta^p_{k+1} = \text{P}^{\Sigma^p_k}, \text{ for } k \geq 0. \]

- One easily obtains the following relations among these classes:
  - \( \Delta^p_1 = \text{P}^{\Sigma^p_0} = \text{P}^\text{P} = \text{P} \)
  - \( \Sigma^p_1 = \text{NP}^{\Sigma^p_0} = \text{NP}^\text{P} = \text{NP} \)
  - \( \Pi^p_1 = \text{coNP}^{\Sigma^p_0} = \text{coNP} \)
  - \( \Delta^p_2 = \text{P}^{\Sigma^p_1} = \text{P}^{\text{NP}} \)
  - \( \Sigma^p_2 = \text{NP}^{\Sigma^p_1} = \text{NP}^{\text{NP}} \)
  - \( \Pi^p_2 = \text{coNP}^{\Sigma^p_1} = \text{coNP}^{\text{NP}} \)
  - \( \Delta^p_k \subseteq \Sigma^p_k \subseteq \Delta^p_{k+1} \subseteq \Sigma^p_{k+1} \subseteq \Pi^p_{k+1} \subseteq \Delta^p_{k+2}, \text{ for all } k \geq 0. \)
The Class EXP

Definition (EXP)

\[ \text{EXP} = \bigcup_{d=1}^{\infty} \text{DTIME}(2^{n^d}) \]

- Problems solvable in *exponential time*
- \( \text{P} \subseteq \text{NP} \subseteq \text{PH} \subseteq \text{EXP} \)
Problems in EXP

- Contains problems such as determining who wins in generalised versions of games
- Canonical problems: time-bounded halting

Time-bounded halting problem

- **Instance:** A Turing machine $M$, an integer $t$ (encoded in binary)
- **Question:** Does $M$ halt on empty input in at most $t$ steps?

- Can be solved by simulating $M$ for $t$ steps
- **Note:** $t \leq 2^{|x|}$
The class NEXP

Definition (NEXP)

The class NEXP is the class of all languages \( L \subseteq \{0, 1\}^* \) for which there exists a Turing machine \( M \) and polynomial functions \( p, q : \mathbb{N} \to \mathbb{N} \) such that

- \( M \) halts on any input \((x, u)\) in time \( O(2^{q(|x|)}) \),
- for all \( x \in \{0, 1\}^* \) we have \( x \in L \) if and only if there is \( u \in \{0, 1\}^* \) with \(|u| \leq 2^{p(|x|)}\) such that \( M(x, u) = 1 \).

- **Equivalent definition**: problems solvable in exponential time with *nondeterministic Turing machines*
- **Unknown if** \( \text{EXP} = \text{NEXP} \)
Exp-completeness and Nexp-completeness

- Completeness for EXP and NEXP is defined in terms of polynomial-time many-one reductions.

- Typical complete problems: succinct versions of P-complete and NP-complete problems.
  - Succinct means that the input is a representation of an exponential-sized instance, e.g. as a circuit.
  - EXP-complete problems include generalised versions of some games.
Polynomial vs. Exponential Time

Theorem

It holds that $P \subseteq EXP$ and $NP \subseteq NEXP$.

- Follows from the time hierarchy theorems (next lecture)
Padding and ‘Scaling Up’

**Theorem**

*If* $P = NP$, *then* $EXP = NEXP$.

**Proof sketch:**

- Assume $P = NP$ and let $L \in NEXP$ be a language that can be verified in time $O(2^{n^c})$.
- Define $L_{pad} = \{(x, 1^{2|x|^c}) : x \in L\}$.
- $L_{pad} \in NP$: any certificate for $x$ (as an instance of $L$) has length at most $2|x|^c$, which is polynomial in $\lvert (x, 1^{2|x|^c}) \rvert$.
- Since $P = NP$, we have $L_{pad} \in P$, implying there is a polynomial-time Turing machine $M$ deciding $L_{pad}$.
- $L \in EXP$: on input $x$, pad $x$ and solve with $M$. 
Lecture 9: Summary

- Complexity classes beyond P and NP
- coNP
- $\Sigma^p_k$, $\Pi^p_k$, $\Delta^p_k$ and PH
- EXP and NEXP