Aalto University
School of Science

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

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

Class coNP: Definition 2

Definition

The class 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 $p\colon \mathbb{N} \to \mathbb{N}$ 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 $L \in$ coNP and for any language $L' \in$ 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 φ (not necessarily CNF).
- **Question:** Is φ satisfied by *all* possible assignments to its variables?

- **Tautology is** coNP**-complete:**
  - Let $L \in$ coNP
  - Apply the Cook–Levin reduction from $\overline{L} \in$ NP to CNF-SAT to map instance x to a CNF φ_x
  - Transform φ_x to $\neg\varphi_x$ to get a TAUTOLOGY instance

coNP, NP and P

- **The following are open questions:**
  - $\mathrm{P} \neq \mathrm{NP}$?
  - $\mathrm{P} \neq \mathrm{coNP}$?
  - $\mathrm{NP} \neq \mathrm{coNP}$?
  - $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$?
- **Note the following relatioships:**
  - If $\mathrm{P} = \mathrm{NP}$, then $\mathrm{P} = \mathrm{coNP}$ (exercise)
  - $\mathrm{NP} = \mathrm{coNP}$ *does not* imply $\mathrm{P} = \mathrm{NP}$

NP-intermediate problems

Theorem (R. Ladner 1975)

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

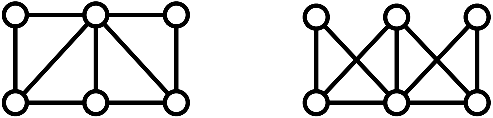
- **No natural problem known to be** NP**-intermediate**
- **One candidate:** *graph isomorphism*

Graph Isomorphism

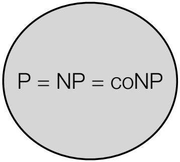
Graph Isomorphism

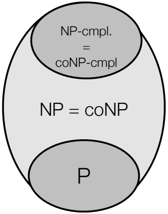
- **Instance:** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2|$.
- **Question:** Is there a bijection $f\colon V_1 \to V_2$ such that

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



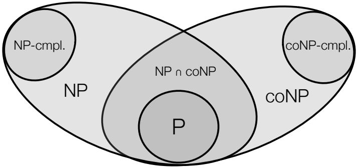
Possible Worlds


P = NP = coNP


NP-cmpl.
=
coNP-cmpl

NP = coNP

P


NP-cmpl.

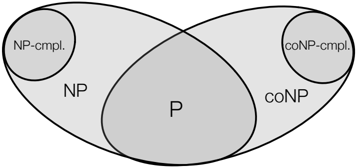
NP

NP ∩ coNP

P

coNP-cmpl.

coNP


NP-cmpl.

NP

P

coNP-cmpl.

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 Σ_2^p and Π_2^p

Definition

The class Σ_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 $p\colon \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$,

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

Definition

$$\Pi_2^p = \text{co}\Sigma_2^p = \left\{L \subseteq \{0,1\}^* : \overline{L} \in \Sigma_2^p\right\}$$

The Polynomial Time Hierarchy

Definition

The class Σ_k^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 $p\colon \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$,

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

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

$$\Pi_k^p = \text{co}\Sigma_k^p = \left\{L \subseteq \{0,1\}^* : \overline{L} \in \Sigma_k^p\right\}$$

The Polynomial Time Hierarchy

Definition (The Polynomial Time Hierarchy)

$$\mathsf{PH} = \bigcup_{k \geq 0} \Sigma_k^p$$

- **Some basic properties of the polynomial time hierarchy:**
  - $\Sigma_0^p = \Pi_0^p = \mathsf{P}$
  - $\Sigma_1^p = \mathsf{NP},\ \Pi_1^p = \mathsf{coNP}$
  - $\Sigma_k^p \subseteq \Pi_{k+1}^p \subseteq \Sigma_{k+2}^p$, for all $k \geq 0$
  - $\mathsf{PH} = \bigcup_{k \geq 0} \Pi_k^p$

The Polynomial Time Hierarchy

- **Generally believed that:**
  - $\Sigma_k^p \neq \Sigma_{k+1}^p$ for all $k \geq 1$ ("*polynomial time hierarchy does not collapse*")
  - $\Sigma_k^p \neq \Pi_k^p$
- Generalised versions of P $\neq$ NP and NP $\neq$ coNP

Theorem

- *For $k \geq 1$, if $\Sigma_k^p = \Pi_k^p$, then* PH $= \Sigma_k^p$ *("hierarchy collapses to level k").*
- *If* P $=$ NP, *then* P $=$ PH *("hierarchy collapses to* P*").*

Complete Problems in PH

- **Completeness for Σ_k^p, Π_k^p and PH is defined in terms of polynomial-time many-one reductions**
- **Complete problem for Σ_k^p:** Σ_kSAT
  - 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 φ 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* $\mathsf{PH} = \Sigma_k^p$.

- **Proof sketch:**
  - Suppose L is PH-complete
  - Since $L \in \mathsf{PH}$, we have $L \in \Sigma_k^p$ for some k
  - Let $L' \in \mathsf{PH}$. Since $L' \leq_p L$, we have $L' \in \Sigma_k^p$.
  - Hence $\mathsf{PH} \subseteq \Sigma_k^p$.

PH: Characterisation via Oracle TM's

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

$$\begin{aligned} \mathsf{P}^L &= \{L' : & L' = M^L \text{ for some (deterministic) polynomial-time} \\ & & \text{oracle Turing machine } M\} \\ \mathsf{NP}^L &= \{L' : & L' = M^L \text{ for some nondeterministic polynomial-time} \\ & & \text{oracle Turing machine } M\}. \end{aligned}$$

- Furthermore, for any family of languages $\mathcal{C}$, we define the relativised classes:

$$\mathsf{P}^{\mathcal{C}} = \bigcup_{L \in \mathcal{C}} \mathsf{P}^L \qquad \mathsf{NP}^{\mathcal{C}} = \bigcup_{L \in \mathcal{C}} \mathsf{NP}^L.$$

Theorem

For every $k \geq 0$, $\Sigma^p_{k+1} = \mathsf{NP}^{\Sigma^p_k}$ *and* $\Pi^p_{k+1} = \mathsf{coNP}^{\Sigma^p_k}$.

PH: Characterisation via Oracle TM's (Cont'd)

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

$$\Delta_0^p = \mathsf{P}, \qquad \Delta_{k+1}^p = \mathsf{P}^{\Sigma_k^p}, \text{ for } k \geq 0.$$

- One easily obtains the following relations among these classes:
- $\Delta_1^p = \mathsf{P}^{\Sigma_0^p} = \mathsf{P}^{\mathsf{P}} = \mathsf{P}$
  $\Sigma_1^p = \mathsf{NP}^{\Sigma_0^p} = \mathsf{NP}^{\mathsf{P}} = \mathsf{NP}$
  $\Pi_1^p = \mathsf{coNP}^{\Sigma_0^p} = \mathsf{coNP}$
- $\Delta_2^p = \mathsf{P}^{\Sigma_1^p} = \mathsf{P}^{\mathsf{NP}}$
  $\Sigma_2^p = \mathsf{NP}^{\Sigma_1^p} = \mathsf{NP}^{\mathsf{NP}}$
  $\Pi_2^p = \mathsf{coNP}^{\Sigma_1^p} = \mathsf{coNP}^{\mathsf{NP}}$
- $\Delta_k^p \subseteq \begin{matrix} \Sigma_k^p \\ \Pi_k^p \end{matrix} \subseteq \Delta_{k+1}^p \subseteq \begin{matrix} \Sigma_{k+1}^p \\ \Pi_{k+1}^p \end{matrix} \subseteq \Delta_{k+2}^p,$ for all $k \geq 0$.

The Class EXP

Definition (EXP)

$\mathsf{EXP} = \bigcup_{d=1}^{\infty} \mathsf{DTIME}(2^{n^d})$

- **Problems solvable in *exponential time***
- $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PH} \subseteq \mathsf{EXP}$

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 $p,q\colon \mathbb{N} \to \mathbb{N}$ 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** EXP $=$ 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 $\mathrm{P} \subsetneq \mathrm{EXP}$ *and* $\mathrm{NP} \subsetneq \mathrm{NEXP}$*.*

- **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 $L_{\text{pad}} = \{(x, 1^{2^{|x|^c}}) : x \in L\}$
  - $L_{\text{pad}} \in$ NP: any certificate for x (as an instance of L) has length at most $2^{|x|^c}$, which is polynomial in $\left|\llcorner(x, 1^{2^{|x|^c}})\lrcorner\right|$.
  - Since P $=$ NP, we have $L_{\text{pad}} \in$ 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
- Σ_k^p, Π_k^p, Δ_k^p and PH
- EXP and NEXP