香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF DATA SCIENCE
數據科學學院

CSC 6011 - Theory of Computation

Lecture 14 - Co-NP

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

TA: Yefan GAO (高叶繁)

Fall 2026

Lecture 14 - Co-NP

Last time:

- Cook-Levin Theorem: SAT is NP-complete
- $3\mathrm{SAT}$ is NP-complete

Today:

- co-NP: Complements of NP languages
- Complexity for optimization problems

News about Exponential Time Hypothesis (ETH)

ETH: 3SAT requires exponential time ($2^{\Omega(n)}$) to solve

THE EXPONENTIAL TIME HYPOTHESIS IS FALSE

OPENAI **(Oct 25, 2026)**

ABSTRACT. We give a deterministic algorithm for exactly counting the satisfying assignments of an arbitrary k-CNF formula. For every fixed $k \geqslant 2$, on a formula with n variables and m clauses, our algorithm uses $\mathrm{poly}_k(n, m)$ space and runs in time $(n+m)^{O_k(1)} 2^{n/(\log n)^{\frac{1}{2}+o(1)}}$. Our algorithm refutes the Exponential Time Hypothesis (ETH), which postulates that deciding 3-SAT on n variables requires $2^{\Omega(n)}$ time in the worst case.

- "P versus NP" not solved yet.

- New possibilities for hard problems

A $2^{o(n)}$ TIME ALGORITHM FOR 3-SAT

CLAUDE
ANTHROPIC **(Oct 26, 2026)**

ABSTRACT. We refute the Exponential Time Hypothesis (ETH) by presenting a deterministic algorithm for 3-SAT on n variables that runs in $2^{o(n)}$ time. Our algorithm in fact solves the harder problem #3-SAT, providing a count of the number of satisfying assignments.

The Unsatisfiability Problem

Boolean formula $\varphi = x \wedge (\neg x \vee y) \wedge \neg y$

Is φ **un**satisfiable?

(φ = False for any assignment to x and y)

Method 1 (truth table):

Enumerate all assignments to x and y:

x	y	$(\neg x \vee y)$	$\neg y$	φ
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	1	0	0

Method 2 (specific to this instance):

- The first clause forces $x = 1$.
- The third clause forces $y = 0$.
- Then, the second clause becomes 0.

So, φ cannot be satisfied.

SAT and UNSAT

Boolean formula $\varphi = x \wedge (\neg x \vee y) \wedge \neg y$

Satisfiability problem (SAT):

Is φ satisfiable?

A "Yes" instance is a satisfiable φ

Has a poly-time verifiable certificate
(a satisfying assignment)

Unsatisfiability problem (UNSAT):

Is φ unsatisfiable?

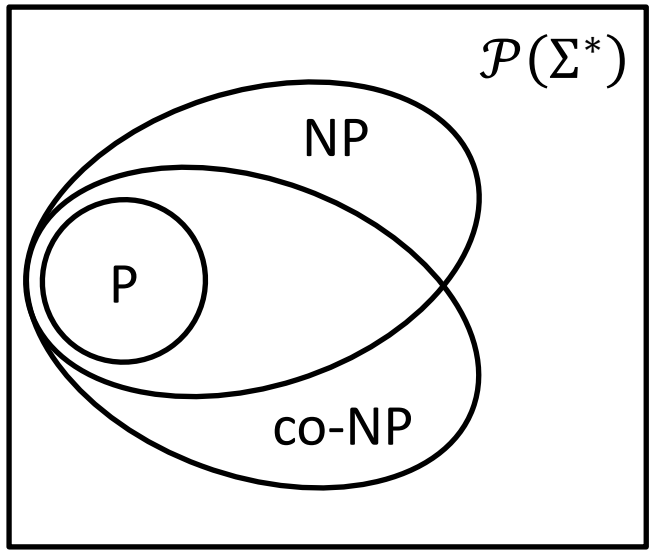
A "Yes" instance is an unsatisfiable φ

Can't easily verify in poly time
(because we might need to
enumerate all assignments)

However, a "No" instance (a satisfiable φ)
has a poly-time verifiable certificate
(a satisfying assignment).

The class co-NP

Let $\overline{L} = \Sigma^* \setminus L$ denote the complement of a language.



Defn: co–NP $= \{ L \mid \overline{L} \in \text{NP} \}$ or equivalently, $\{ \overline{L} \mid L \in \text{NP} \}$

"the collection of complements of languages in NP".

Note: co-NP $\neq \mathcal{P}(\Sigma^*) \setminus$ NP ("all languages outside NP").

Example:

- $\text{SAT} = \{\langle\varphi\rangle \mid \varphi \text{ is a satisfiable formula}\} \in$ NP
  - So $\overline{\text{SAT}} = \{\langle\varphi\rangle \mid \varphi \text{ is an unsatisfiable formula}\} \cup \{w \mid w \text{ is not a valid encoding}\} \in$ co-NP
- $\text{UNSAT} = \{\langle\varphi\rangle \mid \varphi \text{ is an unsatisfiable formula}\} \in$ co-NP
  - Because $\overline{\text{UNSAT}} = \{\langle\varphi\rangle \mid \varphi \text{ is a satisfiable formula}\} \cup \{w \mid w \text{ is not a valid encoding}\} \in$ NP

For simplicity, ignore invalid encodings and assume $\text{UNSAT} = \overline{\text{SAT}}$

Key feature: For $L \in$ co-NP, any $w \notin L$ ("No" instance) has a poly-time verifiable certificate.

More examples of co-NP problems

No Hamiltonian path:

Decide if a directed graph G *doesn't* have a Hamiltonian path from s to t.

$$\text{co}-\text{HAMPATH} = \{\langle G, s, t\rangle \mid G \text{ has no Hamiltonian path from } s \text{ to } t\}$$

"NO" certificate: An s-to-t path visiting every node exactly once.

Tautology (重言式) problem:

Decide if a Boolean formula φ *is always true*.

$$\text{TAUT} = \{\langle \varphi \rangle \mid \forall \text{assignment } a, \varphi(a) = 1\}$$

"NO" certificate: An assignment that makes φ false.

Primality test:

Decide if an integer doesn't have factors other than 1 and itself.

$$\text{PRIMES} = \{\langle n \rangle \mid n \text{ is prime}\}$$

"NO" certificate: d with $1 < d < n$ and d divides n.

Mini-Quiz 14.1

Do NP and co-NP have the same cardinality (size)?

(a) No, card(NP) > card(co-NP)

(b) No, card(NP) < card(NP)

(c) Yes, card(NP) = card(co-NP)

There is a one-to-one mapping between $L \in$ NP and $\overline{L} \in$ co-NP

Note: PRIMES is in P, and P is closed under complementation: P = co-P.

Other equivalent definitions of co-NP

Recall (Lecture 11): $L \in$ NP iff there are a deterministic poly-time verifier V and a polynomial p such that

$$L = \{x \mid \exists y,\ |y| \leq p(|x|),\ V(x,y) = 1\}.$$

"certificate for $x \in L$"

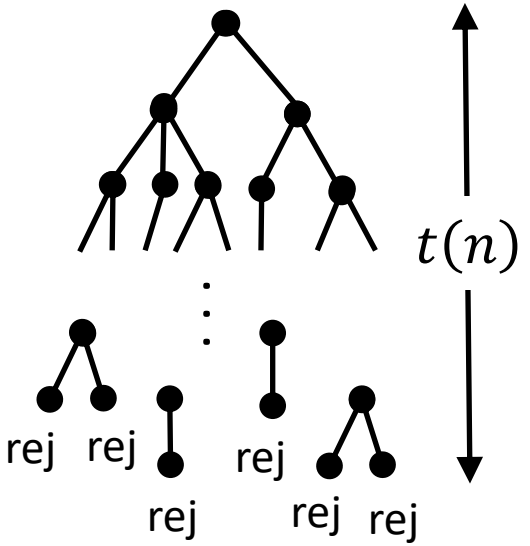
Analogy: $L \in$ co-NP iff there are a deterministic poly-time verifier V and a polynomial p such that

$$L = \{x \mid \forall y,\ |y| \leq p(|x|),\ V(x,y) = 0\}.$$

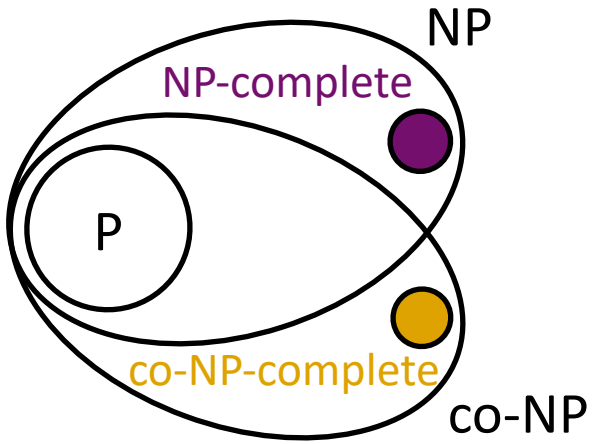
NTM definition: $L \in$ co-NP iff there is a nondeterministic TM M with a polynomial running time $t(n)$ such that

$L =$ strings rejected by all branches of M

Proof: $L = \{\, x \mid \forall$ branch y with depth $\leq t(n)$, branch y rejects $x \,\}$



co-NP-hardness and co-NP-completeness



Defn: B is co-NP-hard if for every $A \in$ co-NP, $A \leq_m^p B$

Defn: B is co-NP-complete if

1) $B \in$ co−NP.
2) B is co-NP-hard.

Theorem: if B is NP-complete, then $\overline{B}$ is co-NP-complete.

Corollary:
UNSAT is co-NP-complete.

Proof:

1) **In co-NP:** $B \in$ NP $\Rightarrow$ $\overline{B} \in$ co−NP.
2) **co-NP-hard:** Pick any $A \in$ co-NP.

   By definition, $\overline{A} \in$ NP. So, $\overline{A} \leq_m^P B$ (why?)

   Taking complements on both sides, $A \leq_m^P \overline{B}$ (why can we do this?)

   Thus every $A \in$ co-NP is poly-time reducible to $\overline{B}$

TAUT is co-NP-complete

The Tautology problem: decide if a Boolean formula is always true.

$$\text{TAUT} = \{ \langle \varphi \rangle \mid \forall \text{ assignment } a, \quad \varphi(a) = 1 \}$$

Theorem: TAUT is co-NP-complete.

Recall: Template for proving C is hard:
reduce a known hard problem to C

Proof:

1) **In co-NP:** $\text{TAUT} \in \text{co-NP}$ (a falsifying assignment certifies that φ is not a tautology).

2) **co-NP-hard:** Give a poly-time reduction f from UNSAT to TAUT:

$$f(\varphi) = \neg\varphi$$

$$\varphi \in \text{UNSAT} \Leftrightarrow \neg\varphi \in \text{TAUT}$$

So, $\text{UNSAT} \leq_m^P \text{TAUT}$, which implies TAUT is co-NP-hard.

What is known, and what is open?

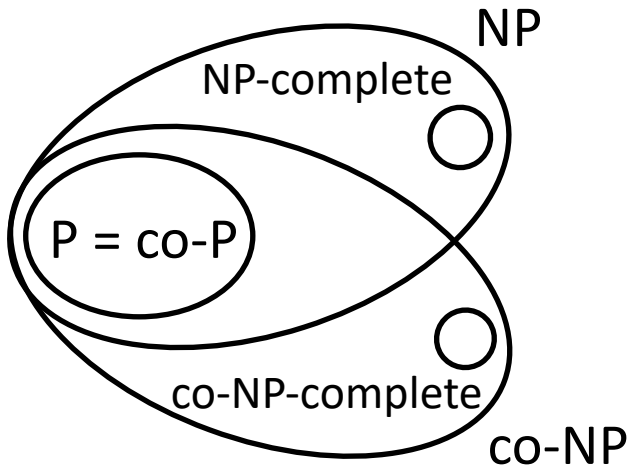
Known implication:

$$P = NP \quad \Rightarrow \quad P = NP = co\text{–}NP$$

Equivalently:

$$NP \neq co\text{–}NP \quad \Rightarrow \quad P \neq NP$$

The reverse implications are not known.



Conjectured picture

Open questions:

$$P = NP\,? \qquad NP = co\text{–}NP\,? \qquad P = NP \cap co\text{–}NP\,?$$

Optimization Problem

Learning outcomes

By the end of the course, students will be able to:

1. Formalize problems using standard models of computation, analyze their capabilities and limitations, and prove results about decidability, reducibility, and complexity.
2. Classify problems into major complexity classes and construct rigorous reductions and impossibility arguments.
3. Apply theoretical computer science concepts to the analysis of algorithms and data science problems.

Many problems in practice are optimization problems (instead of decision problems):

$$\max f(x)$$

$$\text{s.t. } x \in \mathcal{F}$$

How to formally define their complexity?

Example: MAX-SAT

Input: a CNF formula $\varphi = C_1 \wedge \cdots \wedge C_m$ on n variables (each C_i is the OR of some variables).

A feasible solution: an assignment to all variables $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

Objective function: the number of satisfied clauses

$$s_\varphi(x) = |\{i \in \{1, \ldots, m\} : C_i(x) = 1\}|$$

The $\mathrm{MAX{-}SAT}$ **problem:** maximize the number of satisfied clauses:

$$\mathrm{OPT}(\varphi) = \max_{x \in \{0,1\}^n} s_\varphi(x)$$

Output: any assignment $x^* \in \{0,1\}^n$ with $s_\varphi(x^*) = \mathrm{OPT}(\varphi)$

How can we define the complexity of "finding an optimal assignment"?

Via the complexity of decision problems

Decision Problems for MAX-SAT and their complexity

*Under polynomial-time general (Turing) reduction

0. $\mathrm{MAX{-}SAT}$ **(optimization)** NP-hard and co-NP-hard*

Input φ. Output any $x^* \in \arg\max_{x\in\{0,1\}^n} s_\varphi(x)$.

1. $\mathrm{MAX{-}SAT{-}Optimality}$ **(decision)**

Input $\langle \varphi, x \rangle$. Does x maximize the number of satisfied clauses? $(s_\varphi(x) = \mathrm{OPT}(\varphi))$

2. $\mathrm{MAX{-}SAT}_{\geq}$ **(decision)** NP-complete (Special case with $k = m$ is SAT)

Input $\langle \varphi, k \rangle$. Can at least k clauses be satisfied? $(\mathrm{OPT}(\varphi) \geq k)$

3. $\mathrm{MAX{-}SAT}_{\leq}$ **(decision)** co-NP-complete (Special case with $k = m - 1$ is UNSAT)

Input $\langle \varphi, k \rangle$. Does every assignment satisfy at most k clauses? $(\mathrm{OPT}(\varphi) \leq k)$

Poly-Time Turing Reduction & Hardness of Optimization Problem

Let Q be an optimization problem and A be a decision problem.

Defn: polynomial-time general (Turing) reduction $A \leq_T^P Q$

means that there exists a polynomial-time algorithm that decides A using an optimizer for Q as an oracle ("black box").

Theorem: if $A \leq_T^P Q$ and Q can be solved in poly time, then $A \in$ P

Hardness of optimization problem:

- Q is NP-hard if $A \leq_T^P Q$ for every decision problem A in NP.
- Q is co-NP-hard if $A \leq_T^P Q$ for every decision problem A in co-NP.

Prove that $\mathrm{MAX{-}SAT}$ **is** NP-hard **and** co-NP-hard**:**

Given an instance $\langle \varphi, k \rangle$ for $\mathrm{MAX{-}SAT}_{\geq}$ (or $\mathrm{MAX{-}SAT}_{\leq}$):

1. Use the optimizer for $\mathrm{MAX{-}SAT}$ to obtain an optimal assignment x^* for φ.
2. Compute $r = s_\varphi(x^*)$; output "YES" iff $r \geq k$ (or $r \leq k$).

Decision Problems for MAX-SAT and their complexity

0. $\mathrm{MAX{-}SAT}$ **(optimization)** NP-hard and co-NP-hard*

Input φ. Output any $x^* \in \arg\max_{x \in \{0,1\}^n} s_\varphi(x)$.

1. $\mathrm{MAX{-}SAT{-}Optimality}$ **(decision)** ? co-NP-complete (left as exercise)

Input $\langle \varphi, x \rangle$. Does x maximize the number of satisfied clauses?

2. $\mathrm{MAX{-}SAT}_{\geq}$ **(decision)** NP-complete

Input $\langle \varphi, k \rangle$. Can at least k clauses be satisfied?

3. $\mathrm{MAX{-}SAT}_{\leq}$ **(decision)** co-NP-complete

Input $\langle \varphi, k \rangle$. Does every assignment satisfy at most k clauses?

Mini-Quiz 14.2: Solve $\mathrm{MAX{-}SAT{-}Optimality}$ as follows: Given $\langle \varphi, x \rangle$, solve $\mathrm{MAX{-}SAT}_{\leq}$ on input $\langle \varphi, s_\varphi(x) \rangle$. Does this reduction imply $\mathrm{MAX{-}SAT{-}Optimality}$ is co-NP-complete?

(a) Yes (b) No (c) I don't know

This shows $\mathrm{MAX{-}SAT{-}Optimality} \leq_m^P \mathrm{MAX{-}SAT}_{\leq}$

Quick review of today

1. co-NP = collection of complements of NP languages:
   poly-time verifiable certificates for "NO" instances.

2. UNSAT and TAUT are co-NP-complete.

3. Define the complexity of optimization problems via decision problems.