CMSC 28100

Introduction to Complexity Theory

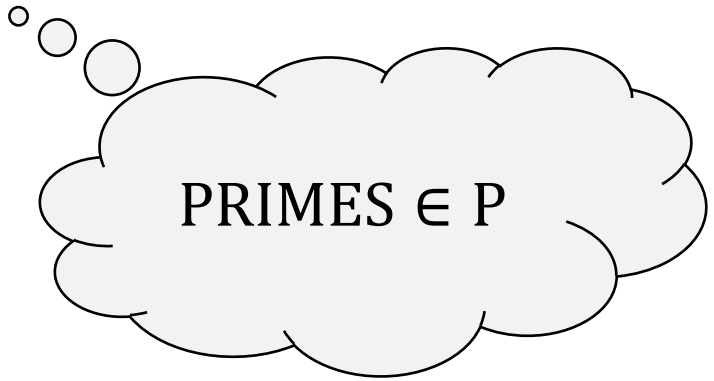
Spring 2025
Instructor: William Hoza

The complexity class coNP

- Let $Y \subseteq \{0,1\}^*$
- **Definition:** $Y \in \mathrm{coNP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \{0,1\}^*$:
  - If $w \in Y$, then $\Pr[M \text{ rejects } w] = 0$
  - If $w \notin Y$, then $\Pr[M \text{ rejects } w] \neq 0$

$\mathrm{FACTOR} \in \mathrm{coNP}$

- $\mathrm{FACTOR} = \{\langle K, R\rangle : K \text{ has a prime factor } p \text{ such that } p \leq R\}$
- **Claim:** $\mathrm{FACTOR} \in \mathrm{coNP}$
- **Proof:** Given $\langle K, R\rangle$:
  - Nondeterministically guess numbers $d \leq \log K$ and $p_1, p_2, \ldots, p_d \leq K$
  - If $p_1, \ldots, p_d$ are prime, $p_1 \cdot p_2 \cdot p_3 \cdots p_d = K$, and $\min(p_1, \ldots, p_d) > R$, reject
  - Otherwise, accept



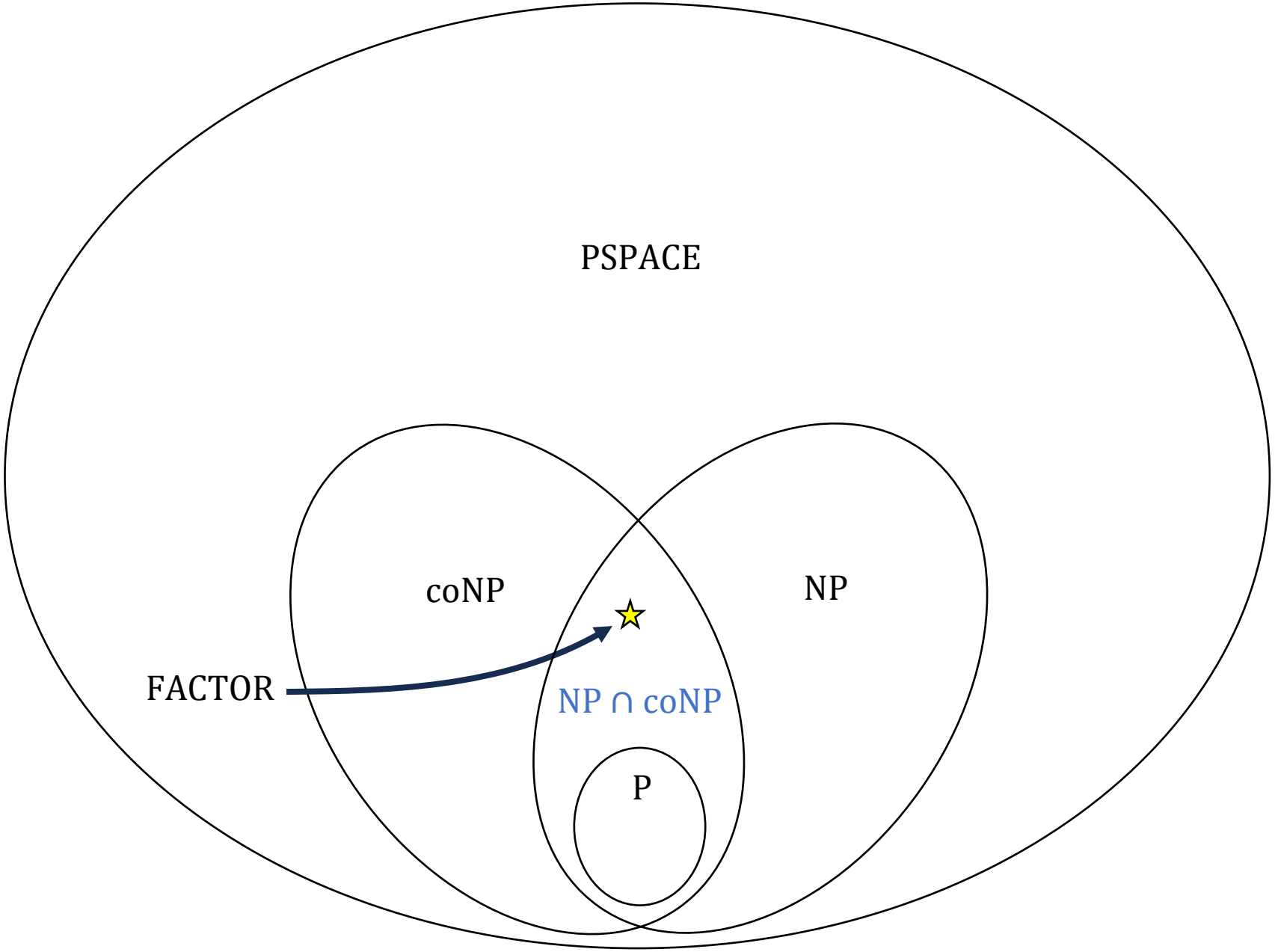
The complexity class $\mathbf{NP \cap coNP}$

- We have shown that $\mathrm{FACTOR} \in \mathrm{NP}$ and $\mathrm{FACTOR} \in \mathrm{coNP}$
- $\mathrm{FACTOR} \in \mathrm{NP} \cap \mathrm{coNP}$
- $Y \in \mathrm{NP} \cap \mathrm{coNP}$ means that for every instance, there is a certificate
  - A certificate of membership for YES instances
  - A certificate of non-membership for NO instances

The $\mathbf{NP}$ vs. $\mathbf{coNP}$ problem

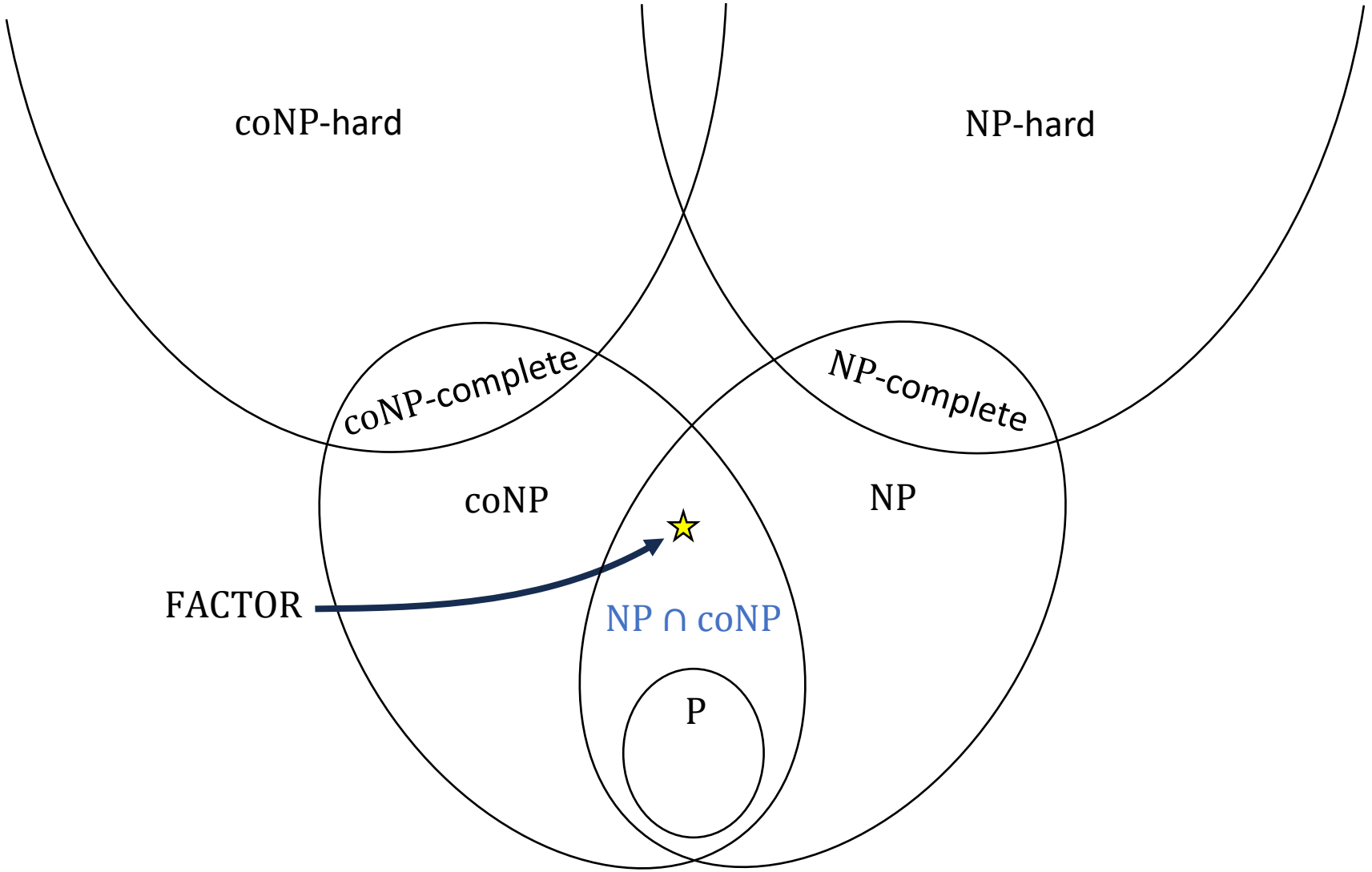
Conjecture: $\mathrm{NP} \neq \mathrm{coNP}$

- "$\mathrm{NP} = \mathrm{coNP}$" would mean that for every unsatisfiable circuit, there is some short certificate I could present to prove to you that a circuit is unsatisfiable
- That sounds counterintuitive! But we don't really know


PSPACE
coNP
NP
FACTOR
NP ∩ coNP
P

NP-completeness and $\mathrm{NP} \cap \mathrm{coNP}$

- Assume $\mathrm{NP} \neq \mathrm{coNP}$
- Under this assumption, we will prove that there are no NP-complete languages in $\mathrm{NP} \cap \mathrm{coNP}$
- This will provide evidence that FACTOR is not NP-complete


coNP-hard
NP-hard
coNP-complete
NP-complete
coNP
NP
FACTOR
NP ∩ coNP
P

coNP is closed under reductions

- Let $Y_1, Y_2 \subseteq \{0,1\}^*$

> **Lemma:** If $Y_1 \leq_{\mathrm{P}} Y_2$ and $Y_2 \in \mathrm{coNP}$, then $Y_1 \in \mathrm{coNP}$

- **Proof:** Since $Y_2 \in \mathrm{coNP}$, there is a polynomial-time "co-nondeterministic" Turing machine M that decides Y_2
- Given $w \in \{0,1\}^*$, run the reduction to produce w', then run M on w'

NP-completeness and $\mathrm{NP} \cap \mathrm{coNP}$

- Let $Y \in \mathrm{NP} \cap \mathrm{coNP}$

Claim: If Y is NP-complete, then $\mathrm{NP} = \mathrm{coNP}$

- **Proof:** For any $Z \in \mathrm{NP}$, we have $Z \leq_{\mathrm{P}} Y$ and $Y \in \mathrm{coNP}$
- By the lemma, $Z \in \mathrm{coNP}$, so $\mathrm{NP} \subseteq \mathrm{coNP}$
- By symmetry, we also have $\mathrm{coNP} \subseteq \mathrm{NP}$

Intractability

- This course so far: How to identify intractability
- Up next: How to cope with intractability

Coping with intractability

- Suppose you really want to decide Y
- You find proof/evidence that $L \notin \mathrm{P}$ 😟
  - Undecidability, EXP-hardness, NP-hardness…
- That doesn't necessarily mean you're out of luck…
- There are several approaches for coping with the fact that $L \notin \mathrm{P}$

Coping with intractability

Nontrivial exponential-time algorithms

- Even if $Y \notin \mathrm{P}$, it still might have a nontrivial algorithm. Example:

> **Theorem:** There is an algorithm that determines whether a given n-variable 3-CNF formula is satisfiable in time $O(1.308^n)$.

  - (Proof omitted. Not on exercises / exams)

- If your inputs happen to be relatively small, then maybe an exponential time complexity is tolerable

Pseudo-polynomial time algorithms

- Suppose $Y = \{\langle x, k\rangle : k \in \mathbb{N} \text{ and } (\text{something})\}$
- "Polynomial time" means $\text{poly}(n)$ time where $n \approx |x| + \log k$
- However, if it's reasonable to assume that k is small, then we might be okay with $\text{poly}(n')$ time where $n' = |x| + k$
  - "Pseudo-polynomial time"
  - $Y' = \{\langle x, 1^k\rangle : k \in \mathbb{N} \text{ and } (\text{something})\}$
- Interesting example: The knapsack problem

The knapsack problem

- Given: Positive integers $w_1, \ldots, w_k, v_1, \ldots, v_k, B$
  - Interpretation: There are k items
  - Item i has weight w_i (in pounds) and value v_i (in dollars)
  - We can carry up to B pounds of stuff in our knapsack
- Goal: Find a set $S \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{i \in S} v_i$ is as large as possible, subject to the constraint $\sum_{i \in S} w_i \leq B$

KNAPSACK is NP-complete

- $\text{KNAPSACK} = \{\langle w_1, \ldots, w_k, v_1, \ldots, v_k, B, V\rangle$: there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S}\, w_i \leq B$ and $\Sigma_{i \in S}\, v_i \geq V\}$

Theorem: KNAPSACK is NP-complete

- **Proof:** It's in NP ✔ We'll show SUBSET-SUM $\leq_{\text{P}}$ KNAPSACK
- Given $\langle a_1, \ldots, a_k, T\rangle$, produce $\langle a_1, \ldots, a_k, a_1, \ldots, a_k, T, T\rangle$

Knapsack in pseudo-polynomial time

- $\text{UNARY-VAL-KNAPSACK} = \{\langle w_1, \ldots, w_k, 1^{v_1}, \ldots, 1^{v_k}, B, 1^V \rangle$: there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S} w_i \leq B$ and $\Sigma_{i \in S} v_i \geq V\}$

Theorem: $\text{UNARY-VAL-KNAPSACK} \in \text{P}$

- **Proof technique:** Dynamic programming

Theorem: UNARY-VAL-KNAPSACK $\in$ P

- **Proof:** We are given $\langle w_1, \ldots, w_k, 1^{v_1}, \ldots, 1^{v_k}, B, 1^V \rangle$
- Let $S_{j,v} \subseteq \{0, 1, \ldots, j\}$ minimize $\sum_{i \in S_{j,v}} w_i$ subject to $\sum_{i \in S_{j,v}} v_i \geq v$
  - Dummy item: $w_0 = v_0 = \infty$
- For $j = 1$ to k, for $v = 1$ to V:
  - Compute $S_{j,v} =$ whichever is less heavy: $S_{j-1,v}$ or $\{j\} \cup S_{j-1,v-v_j}$
- If $\sum_{i \in S_{k,V}} w_i \leq B$, then accept, otherwise reject

Approximation algorithms

- Next approach for coping with intractability: approximation algorithms
- Example: Knapsack

Approximation algorithm for Knapsack

- For every $w_1, \ldots, w_k, v_1, \ldots, v_k, B$, define

$$\mathrm{OPT} = \max\left\{\sum_{i \in S} v_i : S \subseteq \{1, \ldots, k\} \text{ and } \sum_{i \in S} w_i \leq B\right\}$$

Theorem: For every $\epsilon > 0$, there exists a poly-time algorithm such that given $w_1, \ldots, w_k, v_1, \ldots, v_k, B$, the algorithm outputs $S \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in S} w_i \leq B$ and $\sum_{i \in S} v_i \geq (1 - \epsilon) \cdot \mathrm{OPT}$