Computability and Complexity
COSC 4200

NP vs. coNP

The Tautology Problem

A propositional formula ϕ is a *tautology* if every assignment to its variables is a satisfying assignment. Let

$$\mathrm{TAUT} = \{\phi \,|\, \phi \text{ is a tautology}\}\,.$$

In our framework of complexity classes, where can we place this problem?

The Tautology Problem

A propositional formula ϕ is a *tautology* if every assignment to its variables is a satisfying assignment. Let

$$\mathrm{TAUT} = \{\phi \,|\, \phi \text{ is a tautology}\}.$$

In our framework of complexity classes, where can we place this problem?

We can easily solve this problem in polynomial space–by looping over all assignments and checking whether all of them satisfy ϕ. Thus $\mathrm{TAUT} \in \mathrm{PSPACE}$.

The Tautology Problem

Is $\mathrm{TAUT} \in \mathrm{NP}$?

It isn't clear what polynomial-size witnesses could be for TAUT.
A witness would have to prove that all assignments are satisfying.

The Tautology Problem

Is $\mathrm{TAUT} \in \mathrm{NP}$?

It isn't clear what polynomial-size witnesses could be for TAUT. A witness would have to prove that all assignments are satisfying.

On the other hand, there are witnesses that formulas are *not* tautologies – an assignment that makes a formula ϕ evaluate to false proves that $\phi \notin \mathrm{TAUT}$.

$$\begin{aligned} \phi \notin \mathrm{TAUT} \quad &\iff \quad \phi \text{ is not a tautology} \\ &\iff \quad (\exists \tau)\ \tau \text{ does not satisfy } \phi. \end{aligned}$$

co-classes

For a language $A \subseteq \{0,1\}^*$, the *complement* of A is

$$A^c = \{x \in \{0,1\}^* \,|x \notin A\} = \{0,1\}^* - A.$$

co-classes

For a language $A \subseteq \{0,1\}^*$, the *complement* of A is

$$A^c = \{x \in \{0,1\}^* \,|x \notin A\} = \{0,1\}^* - A.$$

Definition

For a class $\mathcal{C}$ of languages, the *co-class* of $\mathcal{C}$ is

$$\mathrm{co}\mathcal{C} = \{A^c \,|A \in \mathcal{C}\}.$$

In other words, $\mathrm{co}\mathcal{C}$ is the class of all languages whose complements are in $\mathcal{C}$. Note that $\mathrm{co}\mathcal{C}$ is *not* the same as the complement $\mathcal{C}^c$.

From our discussion, we have $\mathrm{TAUT} \in \mathrm{coNP}$.

Closure Under Complementation

Definition

A class $\mathcal{C}$ is *closed under complement* if $\mathcal{C} = \mathrm{co}\mathcal{C}$.

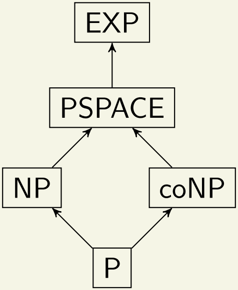
We have studied several classes that have this property, such as P, PSPACE, E, and EXP.

In fact, all deterministic time and space classes are closed under complement. This is because any deterministic algorithm for a language A can easily be modified to get an algorithm for A^c with the same time and space bounds – just flip the accepting and rejecting decisions.

The NP vs. coNP Question

Whether NP is closed under complement, that is whether $\mathrm{NP} = \mathrm{coNP}$, is a major open problem in computational complexity.

If we prove $\mathrm{NP} \neq \mathrm{coNP}$, then we also have that $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{NP} \neq \mathrm{EXP}$.



Proposition

1. $\mathrm{P} \neq \mathrm{NP} \iff \mathrm{P} \neq \mathrm{coNP}$
2. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$
3. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{NP} \neq \mathrm{EXP}$

Proposition

1. $\mathrm{P} \neq \mathrm{NP} \iff \mathrm{P} \neq \mathrm{coNP}$
2. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$
3. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{NP} \neq \mathrm{EXP}$

Proof. For 1, we have

$$\begin{aligned} \mathrm{P} = \mathrm{NP} \quad &\iff \quad \mathrm{coP} = \mathrm{coNP} \\ &\iff \quad \mathrm{P} = \mathrm{coNP}. \end{aligned}$$

Proposition

1. $\mathrm{P} \neq \mathrm{NP} \iff \mathrm{P} \neq \mathrm{coNP}$
2. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$
3. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{NP} \neq \mathrm{EXP}$

Proof. For 1, we have

$$\begin{aligned} \mathrm{P} = \mathrm{NP} \quad &\iff \quad \mathrm{coP} = \mathrm{coNP} \\ &\iff \quad \mathrm{P} = \mathrm{coNP}. \end{aligned}$$

For 2, we have

$$\begin{aligned} \mathrm{P} = \mathrm{NP} \quad &\implies \quad \mathrm{P} = \mathrm{NP} \textit{ and } \mathrm{coP} = \mathrm{coNP} \\ &\implies \quad \mathrm{NP} = \mathrm{P} = \mathrm{coNP}. \end{aligned}$$

Proposition

1. $\mathrm{P} \neq \mathrm{NP} \iff \mathrm{P} \neq \mathrm{coNP}$
2. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$
3. $\mathrm{NP} \neq \mathrm{coNP} \implies \mathrm{NP} \neq \mathrm{EXP}$

Proof. For 1, we have

$$\begin{aligned} \mathrm{P} = \mathrm{NP} &\iff \mathrm{coP} = \mathrm{coNP} \\ &\iff \mathrm{P} = \mathrm{coNP}. \end{aligned}$$

For 2, we have

$$\begin{aligned} \mathrm{P} = \mathrm{NP} &\implies \mathrm{P} = \mathrm{NP} \textit{ and } \mathrm{coP} = \mathrm{coNP} \\ &\implies \mathrm{NP} = \mathrm{P} = \mathrm{coNP}. \end{aligned}$$

For 3, we have

$$\begin{aligned} \mathrm{NP} = \mathrm{EXP} &\implies \mathrm{NP} = \mathrm{EXP} \textit{ and } \mathrm{coNP} = \mathrm{coEXP} \\ &\implies \mathrm{NP} = \mathrm{EXP} = \mathrm{coNP}. \quad \square \end{aligned}$$

Quantifiers and Predicates

Recall we may define NP using a polynomial-time predicate with polynomial-size witnesses and an existential quantifer:

Theorem

$A \in \mathrm{NP}$ if and only if there is a $D \in \mathrm{P}$ and a polynomial p such that for all $x \in \Sigma^*$,

$$x \in A \iff (\exists w \in \{0,1\}^{\leq p(n)})\ \langle x, w \rangle \in D.$$

For coNP, we use a universal quantifier:

Theorem

$A \in \mathrm{coNP}$ if and only if there is a $D \in \mathrm{P}$ and a polynomial p such that for all $x \in \Sigma^*$,

$$x \in A \iff (\forall w \in \{0,1\}^{\leq p(n)})\ \langle x, w \rangle \in D.$$

Just as NP is analogous to Σ_1, coNP is analogous to Π_1.

coNP-Completeness and the Unsatisfiability Problem

Definition

A language B is coNP-complete if

1. B is in coNP, and
2. every A in coNP is polynomial-time reducible to B.

We will soon prove that TAUT is coNP-complete. To do that, we will first prove a very similar problem is coNP-complete.

A formula ϕ is a *unsatisfiable* if there is no assignment τ that satisfies ϕ. Then we define

$$\mathrm{UNSAT} = \{\phi \,|\, \phi \text{ is a unsatisfiable}\}$$

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT.

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT. Since $A^c \in \mathrm{NP}$, we know $A^c \leq_{\mathrm{P}} \mathrm{SAT}$. There is a reduction f such that

$$x \in A^c \iff f(x) \in \mathrm{SAT}.$$

We can assume that $f(x)$ always encodes a formula.

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT. Since $A^c \in \mathrm{NP}$, we know $A^c \leq_{\mathrm{P}} \mathrm{SAT}$. There is a reduction f such that

$$x \in A^c \iff f(x) \in \mathrm{SAT}.$$

We can assume that $f(x)$ always encodes a formula. Then we claim f also reduces A to UNSAT:

$$x \in A \quad \iff \quad x \notin A^c$$

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT. Since $A^c \in \mathrm{NP}$, we know $A^c \leq_{\mathrm{P}} \mathrm{SAT}$. There is a reduction f such that

$$x \in A^c \iff f(x) \in \mathrm{SAT}.$$

We can assume that $f(x)$ always encodes a formula. Then we claim f also reduces A to UNSAT:

$$\begin{aligned} x \in A &\iff x \notin A^c \\ &\iff f(x) \notin \mathrm{SAT} \end{aligned}$$

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT. Since $A^c \in \mathrm{NP}$, we know $A^c \leq_{\mathrm{P}} \mathrm{SAT}$. There is a reduction f such that

$$x \in A^c \iff f(x) \in \mathrm{SAT}.$$

We can assume that $f(x)$ always encodes a formula. Then we claim f also reduces A to UNSAT:

$$\begin{aligned} x \in A &\iff x \notin A^c \\ &\iff f(x) \notin \mathrm{SAT} \\ &\iff f(x) \in \mathrm{UNSAT} \end{aligned}$$

Theorem

UNSAT *is* coNP*-complete.*

Proof. Let $A \in \mathrm{coNP}$. We want to reduce A to UNSAT. Since $A^c \in \mathrm{NP}$, we know $A^c \leq_{\mathrm{P}} \mathrm{SAT}$. There is a reduction f such that

$$x \in A^c \iff f(x) \in \mathrm{SAT}.$$

We can assume that $f(x)$ always encodes a formula. Then we claim f also reduces A to UNSAT:

$$\begin{aligned} x \in A &\iff x \notin A^c \\ &\iff f(x) \notin \mathrm{SAT} \\ &\iff f(x) \in \mathrm{UNSAT} \end{aligned}$$

Therefore $A \leq_{\mathrm{P}} \mathrm{UNSAT}$. Since $A \in \mathrm{coNP}$ was arbitrary, this shows that UNSAT is coNP-complete. $\square$

Theorem

TAUT *is* coNP*-complete.*

Theorem

TAUT *is* coNP*-complete.*

Proof. We will reduce UNSAT to TAUT. The reduction f is very simple. Given ϕ, we compute

$$f(\phi) = \neg(\phi).$$

Theorem

TAUT *is* coNP*-complete.*

Proof. We will reduce UNSAT to TAUT. The reduction f is very simple. Given ϕ, we compute

$$f(\phi) = \neg(\phi).$$

We have

$$\phi \in \mathrm{UNSAT} \iff (\forall \tau)\ \tau \text{ does not satisfy } \phi$$

Theorem

TAUT *is* coNP*-complete.*

Proof. We will reduce UNSAT to TAUT. The reduction f is very simple. Given ϕ, we compute

$$f(\phi) = \neg(\phi).$$

We have

$$\begin{aligned} \phi \in \mathrm{UNSAT} \quad &\iff \quad (\forall \tau)\ \tau \text{ does not satisfy } \phi \\ &\iff \quad (\forall \tau)\ \tau \text{ satisfies } \neg(\phi) \end{aligned}$$

Theorem

TAUT *is* coNP*-complete.*

Proof. We will reduce UNSAT to TAUT. The reduction f is very simple. Given ϕ, we compute

$$f(\phi) = \neg(\phi).$$

We have

$$\begin{aligned} \phi \in \mathrm{UNSAT} \quad &\Longleftrightarrow \quad (\forall \tau)\ \tau \text{ does not satisfy } \phi \\ &\Longleftrightarrow \quad (\forall \tau)\ \tau \text{ satisfies } \neg(\phi) \\ &\Longleftrightarrow \quad \neg(\phi) \in \mathrm{TAUT}. \quad \square \end{aligned}$$

Closure Under $\leq_{\mathrm{P}}$-Reductions

Definition

A complexity class $\mathcal{C}$ is closed under $\leq_{\mathrm{P}}$ reductions if $B \in \mathcal{C}$ and $A \leq_{\mathrm{P}} B$ implies $A \in \mathcal{C}$.

Lemma

P, NP, coNP, PSPACE, *and* EXP *are closed under* $\leq_{\mathrm{P}}$*-reductions.*

Proof. We already proved this for P: if $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{P}$, then $A \in \mathrm{P}$. The proofs for PSPACE and EXP are similar.

Suppose that $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{NP}$. Let f be the polynomial-time reduction from A and let V be a polynomial-time verifier for B.

Suppose that $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{NP}$. Let f be the polynomial-time reduction from A and let V be a polynomial-time verifier for B. We define a new Verifier V':

V': on input $\langle x, w \rangle$:
  compute $f(x)$
  run V on input $\langle f(x), w \rangle$
    accept if V accepts
    reject if V rejects

Then V' runs in polynomial time as it is a composition of polynomial-time algorithms.

Suppose that $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{NP}$. Let f be the polynomial-time reduction from A and let V be a polynomial-time verifier for B. We define a new Verifier V':

> V': on input $\langle x, w \rangle$:
> compute $f(x)$
> run V on input $\langle f(x), w \rangle$
> accept if V accepts
> reject if V rejects

Then V' runs in polynomial time as it is a composition of polynomial-time algorithms. We have

$$x \in A \iff f(x) \in B$$

Suppose that $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{NP}$. Let f be the polynomial-time reduction from A and let V be a polynomial-time verifier for B. We define a new Verifier V':

V': on input $\langle x, w \rangle$:
    compute $f(x)$
    run V on input $\langle f(x), w \rangle$
        accept if V accepts
        reject if V rejects

Then V' runs in polynomial time as it is a composition of polynomial-time algorithms. We have

$$
\begin{aligned}
x \in A \quad &\iff \quad f(x) \in B \\
&\iff \quad (\exists w)\ V \text{ accepts } \langle f(x), w \rangle
\end{aligned}
$$

Suppose that $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{NP}$. Let f be the polynomial-time reduction from A and let V be a polynomial-time verifier for B. We define a new Verifier V':

V': on input $\langle x, w \rangle$:
  compute $f(x)$
  run V on input $\langle f(x), w \rangle$
    accept if V accepts
    reject if V rejects

Then V' runs in polynomial time as it is a composition of polynomial-time algorithms. We have

$$\begin{aligned} x \in A \quad &\iff \quad f(x) \in B \\ &\iff \quad (\exists w)\ V \text{ accepts } \langle f(x), w \rangle \\ &\iff \quad (\exists w)\ V' \text{ accepts } \langle x, w \rangle, \end{aligned}$$

so V' is a verifier for A. Therefore $A \in \mathrm{NP}$.

Now suppose $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{coNP}$. Then $A^c \leq_{\mathrm{P}} B^c$ via the same reduction and $B^c \in \mathrm{NP}$, so $A^c \in \mathrm{NP}$ by the above proof. Therefore $A \in \mathrm{coNP}$. $\square$

Theorem

The following are equivalent.

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{TAUT} \in \mathrm{NP}$.

Theorem

The following are equivalent.

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{TAUT} \in \mathrm{NP}$.

Proof. If $\mathrm{NP} = \mathrm{coNP}$, then $\mathrm{TAUT} \in \mathrm{NP}$ since $\mathrm{TAUT} \in \mathrm{coNP}$.

Theorem

The following are equivalent.

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{TAUT} \in \mathrm{NP}$.

Proof. If $\mathrm{NP} = \mathrm{coNP}$, then $\mathrm{TAUT} \in \mathrm{NP}$ since $\mathrm{TAUT} \in \mathrm{coNP}$.

Assume that $\mathrm{TAUT} \in \mathrm{NP}$.

- Then since NP is closed under $\leq_{\mathrm{P}}$-reductions and everything in coNP reduces to TAUT, we have $\mathrm{coNP} \subseteq \mathrm{NP}$.

Theorem

The following are equivalent.

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{TAUT} \in \mathrm{NP}$.

Proof. If $\mathrm{NP} = \mathrm{coNP}$, then $\mathrm{TAUT} \in \mathrm{NP}$ since $\mathrm{TAUT} \in \mathrm{coNP}$.

Assume that $\mathrm{TAUT} \in \mathrm{NP}$.

- Then since NP is closed under $\leq_{\mathrm{P}}$-reductions and everything in coNP reduces to TAUT, we have $\mathrm{coNP} \subseteq \mathrm{NP}$.

- To see that $\mathrm{NP} \subseteq \mathrm{coNP}$, let $A \in \mathrm{NP}$. Then $A^c \in \mathrm{coNP}$, so $A^c \in \mathrm{NP}$ because we just proved $\mathrm{coNP} \subseteq \mathrm{NP}$. Hence $A \in \mathrm{coNP}$, so $\mathrm{NP} \subseteq \mathrm{coNP}$.

Theorem

The following are equivalent.

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{TAUT} \in \mathrm{NP}$.

Proof. If $\mathrm{NP} = \mathrm{coNP}$, then $\mathrm{TAUT} \in \mathrm{NP}$ since $\mathrm{TAUT} \in \mathrm{coNP}$.

Assume that $\mathrm{TAUT} \in \mathrm{NP}$.

- Then since NP is closed under $\leq_{\mathrm{P}}$-reductions and everything in coNP reduces to TAUT, we have $\mathrm{coNP} \subseteq \mathrm{NP}$.
- To see that $\mathrm{NP} \subseteq \mathrm{coNP}$, let $A \in \mathrm{NP}$. Then $A^c \in \mathrm{coNP}$, so $A^c \in \mathrm{NP}$ because we just proved $\mathrm{coNP} \subseteq \mathrm{NP}$. Hence $A \in \mathrm{coNP}$, so $\mathrm{NP} \subseteq \mathrm{coNP}$.

Therefore $\mathrm{NP} = \mathrm{coNP}$. $\square$

Proposition

Let $A \subseteq \{0,1\}^*$. The following are equivalent.

1. *A is NP-complete.*
2. *A^c is coNP-complete.*

Proposition

Let $A \subseteq \{0,1\}^*$. The following are equivalent.

1. *A is NP-complete.*
2. *A^c is coNP-complete.*

Proof. Assume A is NP-complete. Let $B \in \mathrm{coNP}$. Then $B^c \in \mathrm{NP}$, so $B^c \leq_{\mathrm{P}} A$ via some reduction f.

Proposition

Let $A \subseteq \{0,1\}^*$. The following are equivalent.

1. *A is NP-complete.*
2. *A^c is coNP-complete.*

Proof. Assume A is NP-complete. Let $B \in \mathrm{coNP}$. Then $B^c \in \mathrm{NP}$, so $B^c \leq_{\mathrm{P}} A$ via some reduction f. Then

$$x \in B \iff x \notin B^c \iff f(x) \notin A \iff f(x) \in A^c,$$

so $B \leq_{\mathrm{P}} A^c$.

Proposition

Let $A \subseteq \{0,1\}^*$. The following are equivalent.

1. *A is NP-complete.*
2. *A^c is coNP-complete.*

Proof. Assume A is NP-complete. Let $B \in \mathrm{coNP}$. Then $B^c \in \mathrm{NP}$, so $B^c \leq_{\mathrm{P}} A$ via some reduction f. Then

$$x \in B \iff x \notin B^c \iff f(x) \notin A \iff f(x) \in A^c,$$

so $B \leq_{\mathrm{P}} A^c$. Therefore A^c is coNP-complete.

Proposition

Let $A \subseteq \{0,1\}^*$. The following are equivalent.

1. *A is NP-complete.*
2. *A^c is coNP-complete.*

Proof. Assume A is NP-complete. Let $B \in \mathrm{coNP}$. Then $B^c \in \mathrm{NP}$, so $B^c \leq_{\mathrm{P}} A$ via some reduction f. Then

$$x \in B \iff x \notin B^c \iff f(x) \notin A \iff f(x) \in A^c,$$

so $B \leq_{\mathrm{P}} A^c$. Therefore A^c is coNP-complete.
The other direction is similar. $\square$

coNP-Complete Examples

For a graph G, let $\omega(G)$ denote the size of a largest clique in G. We can restate the $\textsc{Clique}$ problem as

$$\textsc{Clique} = \{\langle G, k\rangle \mid \omega(G) \geq k\}.$$

The complementary problem

$$\{\langle G, k\rangle \mid \omega(G) < k\}$$

coNP-Complete Examples

For a graph G, let $\omega(G)$ denote the size of a largest clique in G. We can restate the CLIQUE problem as

$$\textsc{Clique} = \{\langle G, k\rangle \mid \omega(G) \geq k\}.$$

The complementary problem

$$\begin{aligned}&\{\langle G, k\rangle \mid \omega(G) < k\}\\ &= \{\langle G, k\rangle \mid \text{every clique in } G \text{ has size} < k\}\end{aligned}$$

is coNP-complete.

coNP-Complete Examples

Similarly, the complementary problem of

$$\mathrm{VERTEX\text{-}COVER} = \{\langle G, k\rangle \,|\, G \text{ has a vertex cover of size } k\}$$

is coNP-complete:

$$\{\langle G, k\rangle \mid \text{every vertex cover of } G \text{ has size} > k\}.$$

coNP-Complete Examples

The complementary Hamiltonian cycle problem:

$$\{G \mid G \text{ does not have a Hamiltonian cycle}\}$$

The complementary TSP problem:

$$\left\{ \langle n, c, k \rangle \;\middle|\; \begin{array}{l} c \text{ is a cost function and every} \\ \text{tour has cost at least } k \end{array} \right\}$$

The Complexity Class $\mathrm{NP} \cap \mathrm{coNP}$

$$\begin{tikzcd} \boxed{\mathrm{NP}} & & \boxed{\mathrm{coNP}} \\ & \boxed{\mathrm{NP} \cap \mathrm{coNP}} \arrow[ul] \arrow[ur] & \\ & \boxed{\mathsf{P}} \arrow[u] & \end{tikzcd}$$

The intersection of the classes NP and coNP gives the complexity class $\mathrm{NP} \cap \mathrm{coNP}$.

- For a problem A to be in $\mathrm{NP} \cap \mathrm{coNP}$, we must have $A \in \mathrm{NP}$ and $A \in \mathrm{coNP}$.

- Equivalently, $A \in \mathrm{NP} \cap \mathrm{coNP}$ if A and A^c are both in NP. That is, there are polynomial-time verifiable witnesses for membership in A and in A^c. For any string x, there is a proof that either $x \in A$ or that $x \notin A$.

Strong Nondeterminism

Another definition of $\mathrm{NP} \cap \mathrm{coNP}$ uses *strong nondeterministic computation*.

- Every computation path of a strong nondeterministic TM produces output in $\{0, 1, ?\}$.

Strong Nondeterminism

Another definition of $\mathrm{NP} \cap \mathrm{coNP}$ uses *strong nondeterministic computation*.

- Every computation path of a strong nondeterministic TM produces output in $\{0, 1, ?\}$.
- At least one computation path does not output ?.
- The computation is accepting if all outputs are in $\{1, ?\}$.
- The computation is rejecting if all outputs are in $\{0, ?\}$.

Strong Nondeterminism

Another definition of $\mathrm{NP} \cap \mathrm{coNP}$ uses *strong nondeterministic computation*.

- Every computation path of a strong nondeterministic TM produces output in $\{0, 1, ?\}$.
- At least one computation path does not output ?.
- The computation is accepting if all outputs are in $\{1, ?\}$.
- The computation is rejecting if all outputs are in $\{0, ?\}$.

Polynomial-time strong nondeterministic TMs accept exactly the $\mathrm{NP} \cap \mathrm{coNP}$ problems:

$$\mathrm{NP} \cap \mathrm{coNP} = \left\{ L(N) \,\middle|\, \begin{array}{l} N \text{ is a polynomial-time} \\ \text{strong nondeterministic TM} \end{array} \right\}.$$

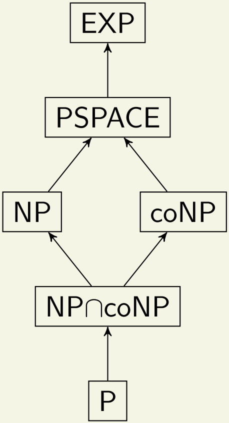
Problems in $\mathrm{NP} \cap \mathrm{coNP}$

Problems in $\mathrm{NP} \cap \mathrm{coNP}$:

- Factoring
- Discrete Logarithm
- Lattice Problems
- Parity Games
- Stochastic Games

Under a derandomization hypothesis, Graph Isomorphism and other isomorphism problems are in $\mathrm{NP} \cap \mathrm{coNP}$.

Open Problems



Does $\mathrm{NP} = \mathrm{coNP}$?

Does $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$?