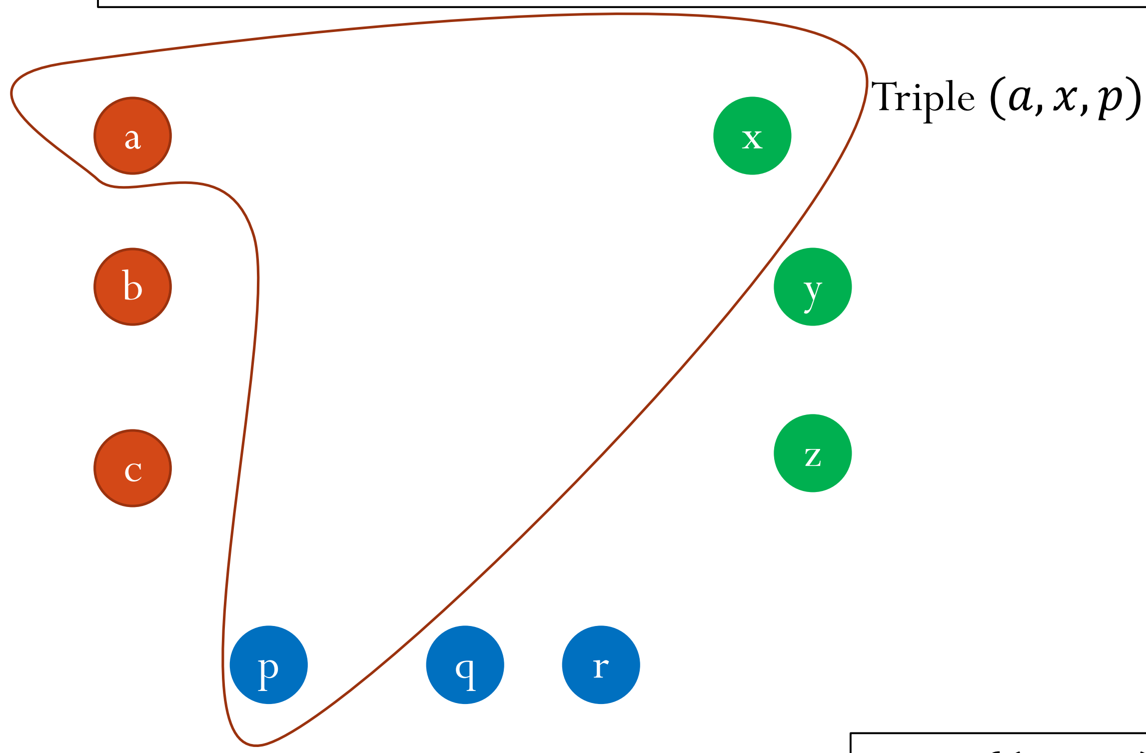
CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Computational Intractability: NP-complete problems

- Problem(3-D matching): Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.



$T = \{(a, x, p), (a, y, p), (b, y, q), (c, z, r)\}$

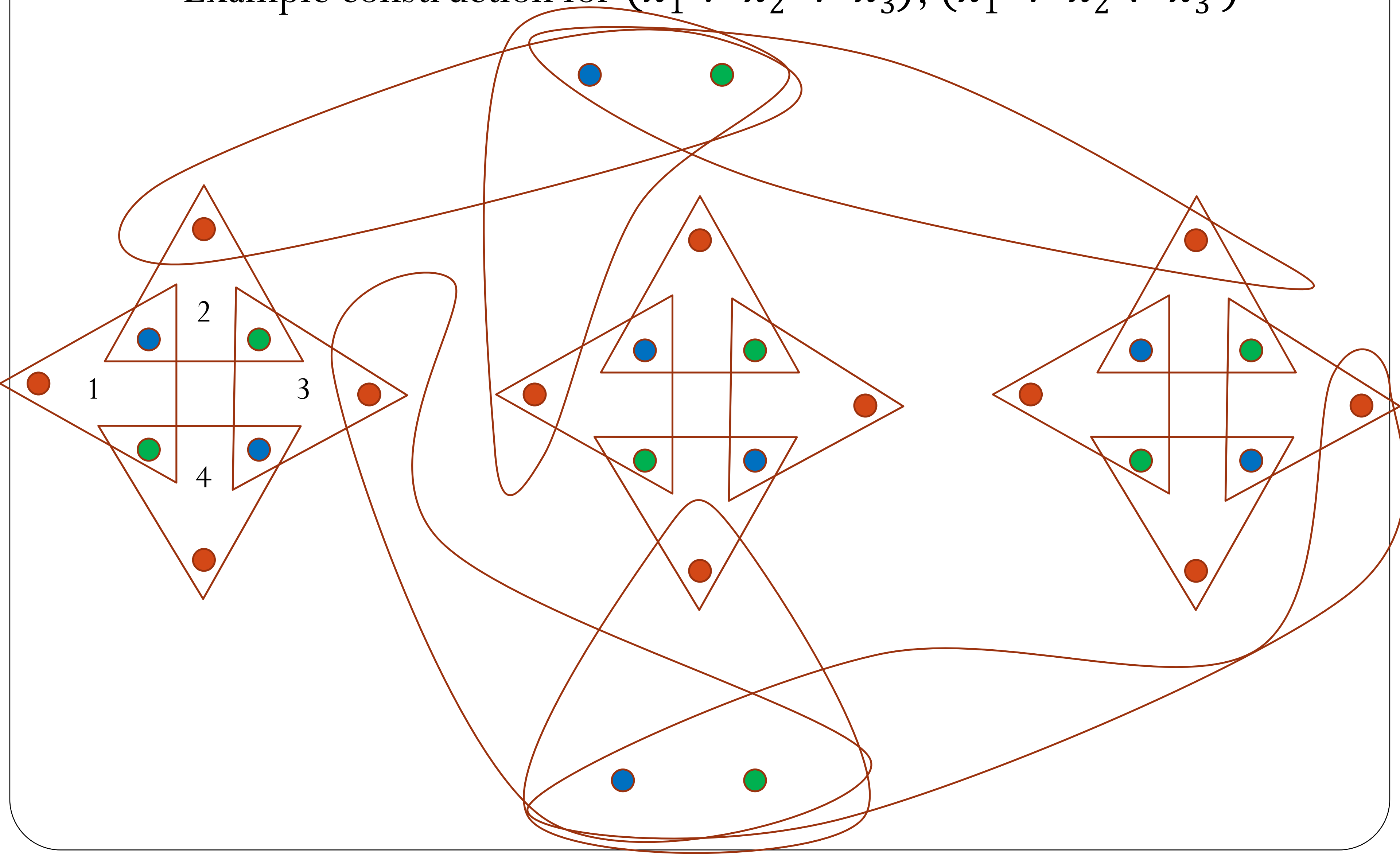
Computational Intractability: NP-complete problems

- Problem(3-D matching): Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.

- Claim 1: 3-D matching is in **NP**.
- Claim 2: 3-D matching is **NP**-complete.
  - Claim 2.1: 3-SAT $\leq_p$ 3-D matching.
  - Proof: We will show an efficient many-one reduction.

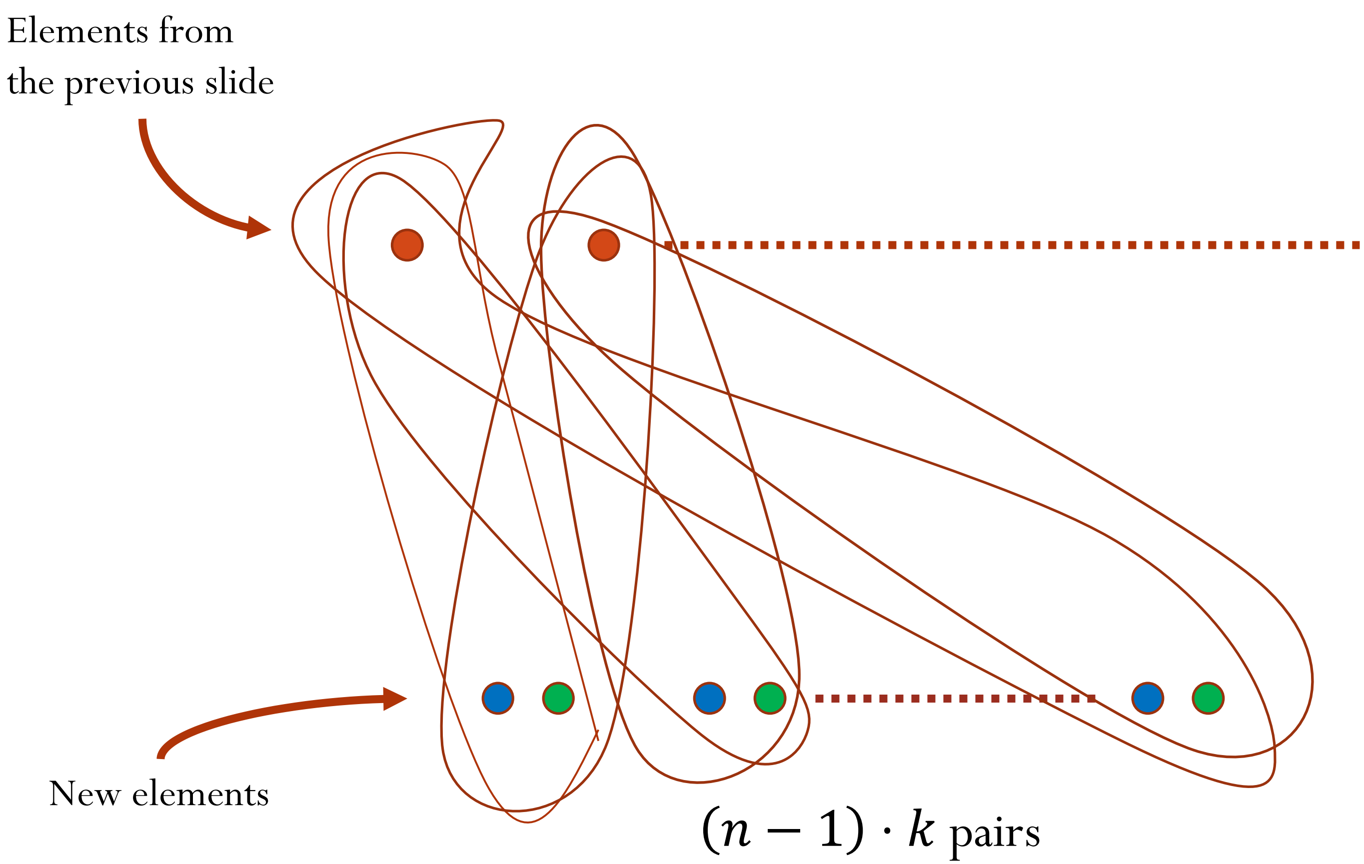
Computational Intractability: NP-complete problems

- Example construction for $(x_1 \lor x_2' \lor x_3), (x_1' \lor x_2 \lor x_3')$



Computational Intractability: NP-complete problems

- Example construction for $(x_1 \lor x_2' \lor x_3), (x_1' \lor x_2 \lor x_3')$



k denotes the number of Clauses

Computational Intractability

NP-Complete problems: Subset-sum

Computational Intractability: NP-complete problems

- Problem(Subset-sum): Given natural numbers $w_1, \ldots, w_n$ and a target number W, determine if there is a subset of $\{w_1, \ldots, w_n\}$ that adds up to precisely W.

- Claim 1: Subset-sum is in **NP**.
- Claim: Subset-sum is **NP**-complete.
  - Claim 2.1: 3-D matching $\leq_p$ Subset-sum.
  - Proof idea: We will show an efficient many-one reduction. Given an instance (X, Y, Z, T) of the 3-D matching problem, we construct an instance of the Subset-sum problem.

Computational Intractability: NP-complete problems

- Claim 1: Subset-sum matching is in **NP**.
- Claim: Subset-sum is **NP**-complete.
  - Claim 2.1: 3-D matching $\leq_P$ Subset-sum.
  - Proof idea: We will show an efficient many-one reduction. Given an instance (X, Y, Z, T) of the 3-D matching problem, we construct an instance of the Subset-sum problem.
    - We construct a $3n$ bit vector. Given a triple $t_i = (x_1, y_3, z_5)$ we construct the following vector v_i:

1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

Computational Intractability: NP-complete problems

- Claim: Subset-sum is **NP**-complete.
  - Claim 2.1: 3-D matching $\leq_p$ Subset-sum.
  - Proof idea: Given an instance (X, Y, Z, T) of the 3-D matching problem, we construct an instance of the Subset-sum problem.
    - We construct a $3n$ bit vector. Given a triple $t_i = (x_1, y_3, z_5)$ we construct the following vector v_i:

1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

- Let w_i be the value of v_i in base $(|T| + 1)$ and let

$$W = \sum_{i=0}^{3n-1} (|T| + 1)^i$$

Computational Intractability: NP-complete problems

- Claim: Subset-sum is **NP**-complete.
  - Claim 2.1: 3-D matching $\leq_p$ Subset-sum.
  - Proof idea: Given an instance (X, Y, Z, T) of the 3-D matching problem, we construct an instance of the Subset-sum problem.
    - Let w_i be the value of v_i in base $(|T| + 1)$ and let
      $$W = \sum_{i=0}^{3n-1} (|T|+1)^i$$
    - Claim: There is a 3-D matching if and only if there is a subset of $\{w_1, \ldots, w_{|T|}\}$ that sums to W.

Computational Intractability

Complexity Classes

Computational Intractability: Complexity Classes

- **<u>NP</u>**: All problems X such that there is an efficient certifier for X.
  - *Efficient certifier*: There is an efficient certifier $B(.,.)$ such that
    - for all yes instances s of X, there is a short certificate t such that $B(s,t)$="yes" ,and
    - for all "no" instances s of X, there is *no* short string t such that $B(s,t)$="yes".
  - <u>Example 1</u>: Consider 3-SAT. If the formula is satisfiable, then there is a short certificate of this fact. Is there a short certificate showing that a formula is unsatisfiable?
  - <u>Example 2</u>: Consider 3-coloring. Is there a short certificate of the fact that there is no possible 3 coloring of the given graph?

Computational Intractability: Complexity Classes

- **co-NP**: A problem *X* is in **co-NP** if and only if the problem *X'* is in **NP**.
  - *X'*: Complement of *X*.
- Examples of **co-NP** problems:
  - UNSAT: Given a formula, determine if the formula is unsatisfiable.
  - TAUTOLOGY: Given a formula, determine if it is a tautology.
  - NO-Hamiltonian-cycle: Given a graph, determine if there is no hamiltonian cycle in the graph.

Computational Intractability: Complexity Classes

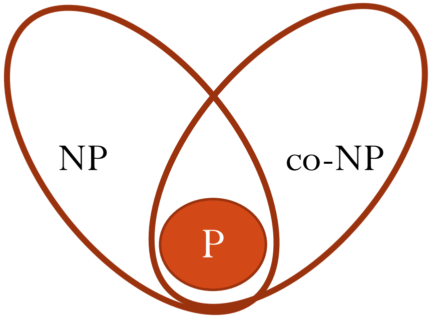
- Question: Is **NP** = **co-NP**?

Computational Intractability: Complexity Classes

- <u>Question</u>: Is **NP** = **co-NP**?
  - Not known but the popular belief is that they are not equal.
- <u>Theorem</u>: If **NP** ≠ **co-NP**, then **P** ≠ **NP**.

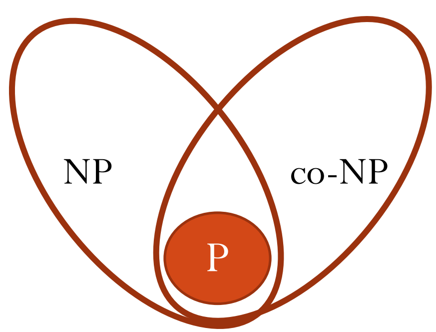
Computational Intractability: Complexity Classes

- <u>Question</u>: Is **NP** = **co-NP**?
  - Not known but the popular belief is that they are not equal.
- <u>Theorem</u>: If **NP** ≠ **co-NP**, then **P** ≠ **NP**.
  - <u>Proof idea</u>: **P** is closed under complementation.



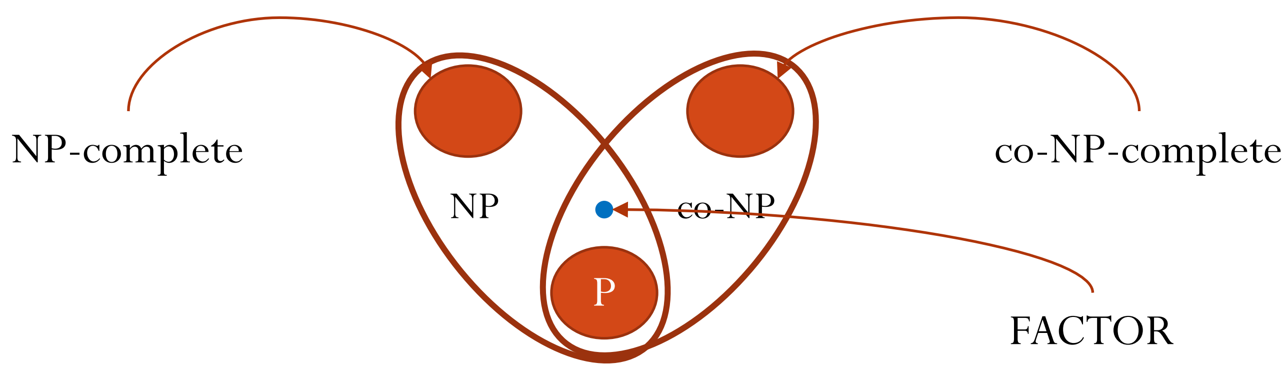
Computational Intractability: Complexity Classes

- <u>Question</u>: Is **NP** = **co-NP**?
  - Not known but the popular belief is that they are not equal.
- <u>Theorem</u>: If **NP** ≠ **co-NP**, then **P** ≠ **NP**.
  - <u>Proof idea</u>: **P** is closed under complementation.
- <u>Question</u>: Are there problems in **NP** intersect **co-NP** that are not in **P**?



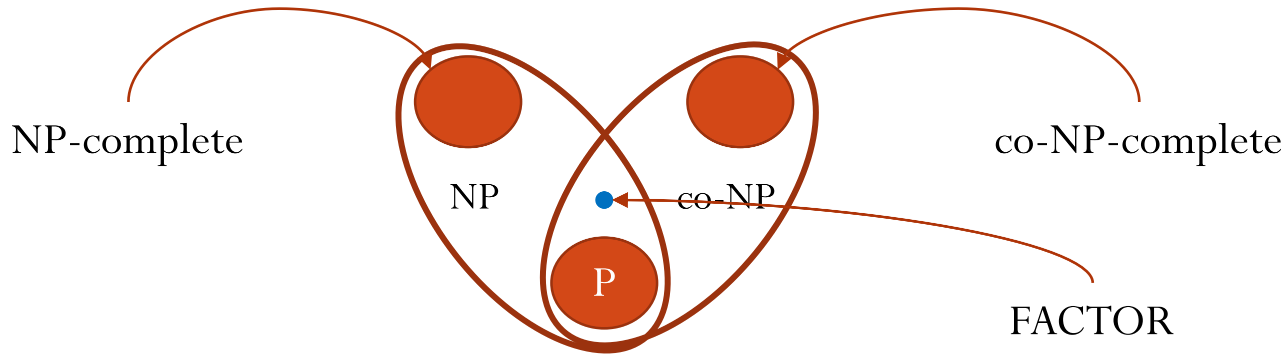
Computational Intractability: Complexity Classes

- Question: Is **NP** = **co-NP**?
  - Not known but the popular belief is that they are not equal.
- Theorem: If **NP** ≠ **co-NP**, then **P** ≠ **NP**.
  - Proof idea: **P** is closed under complementation.
- Question: Are there problems in **NP** intersect **co-NP** that are not in **P**?
  - FACTOR is in **NP** intersect **co-NP** but it is not known to be in **P**.
  - There is mixed feeling about this question.



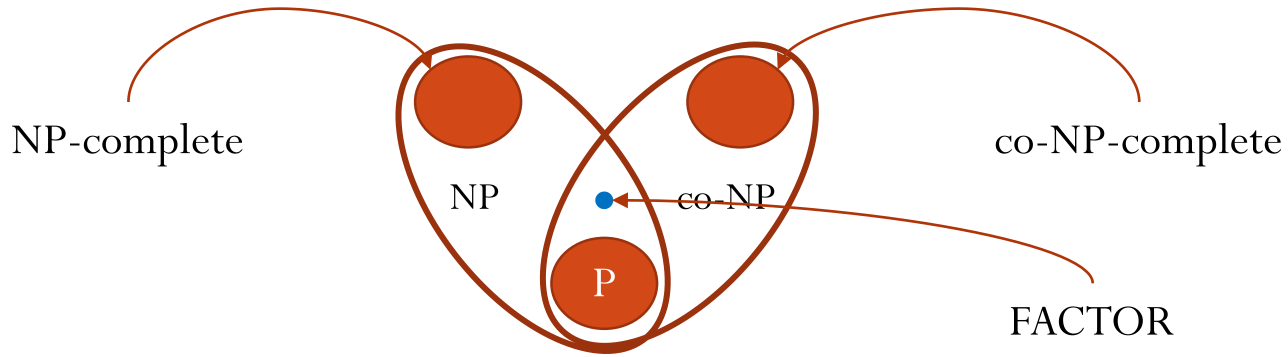
Computational Intractability: Complexity Classes

- <u>FACTOR</u>: Given two integers x and y, is there a non-trivial factor of x that is less than y.
  - <u>FACTOR is in **NP**</u>:
  - <u>FACTOR is in **co-NP**</u>:



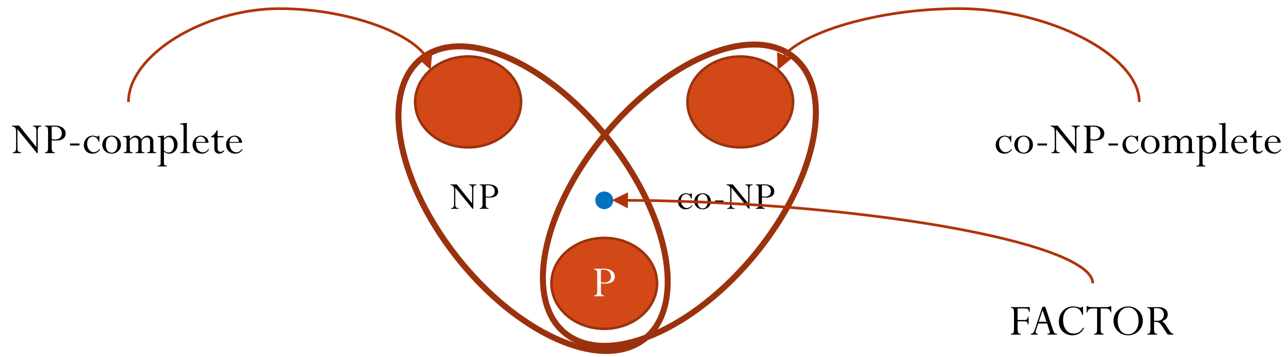
Computational Intractability: Complexity Classes

- <u>FACTOR</u>: Given two integers x and y, is there a non-trivial factor of x that is less than y.
  - <u>FACTOR is in **NP**</u>: The non-trivial factor of x less than y acts as a certificate.
  - <u>FACTOR is in **co-NP**</u>: The prime factorization of x acts as a certificate.



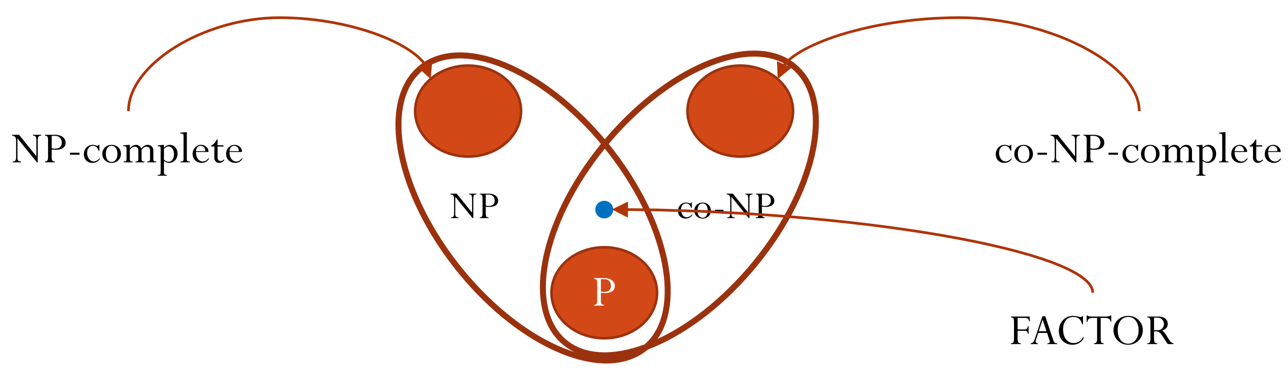
Computational Intractability: Complexity Classes

- **PSPACE**: The set of all problems that can be solved using polynomial amount of *space*.



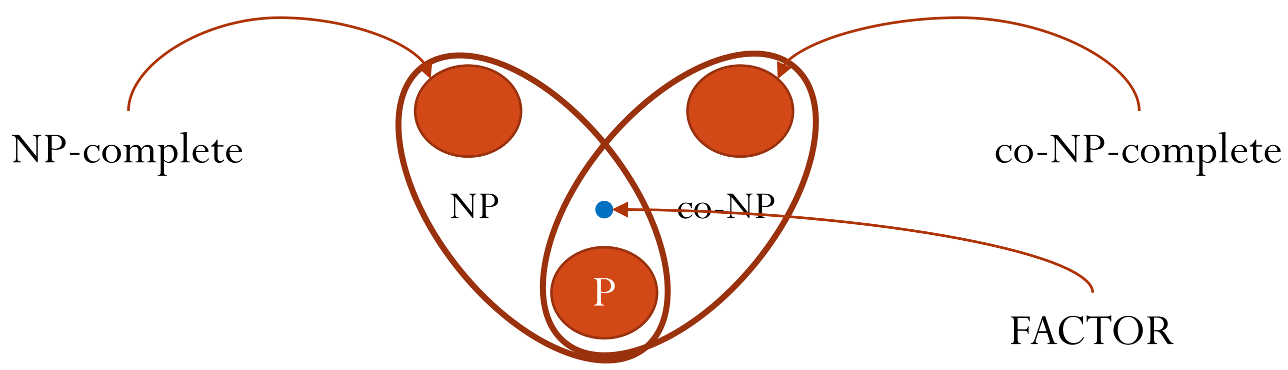
Computational Intractability: Complexity Classes

- **PSPACE**: The set of all problems that can be solved using polynomial amount of *space*.
- Theorem: **PSPACE** contains **P**.



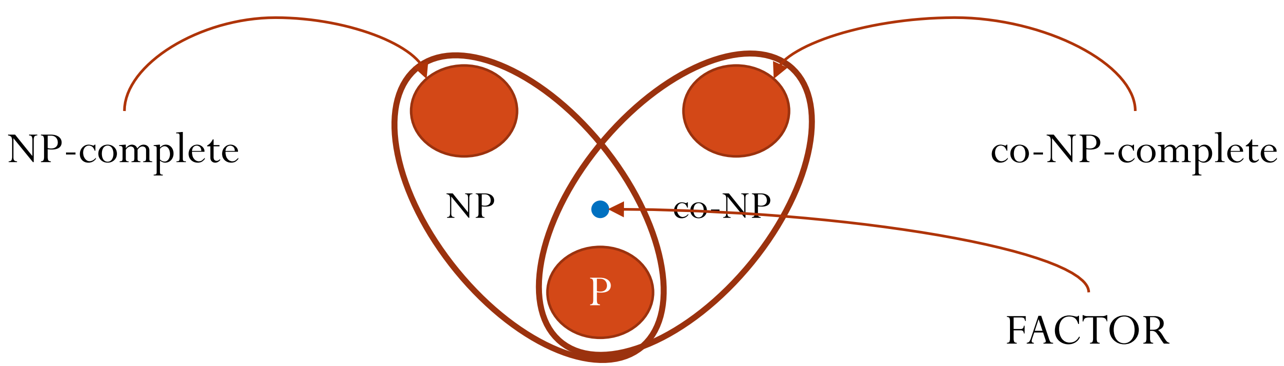
Computational Intractability: Complexity Classes

- **<u>PSPACE</u>**: The set of all problems that can be solved using polynomial amount of *space*.
- <u>Theorem</u>: **PSPACE** contains **P**.
  - <u>Proof idea</u>: You cannot use more than polynomial space in polynomial time.



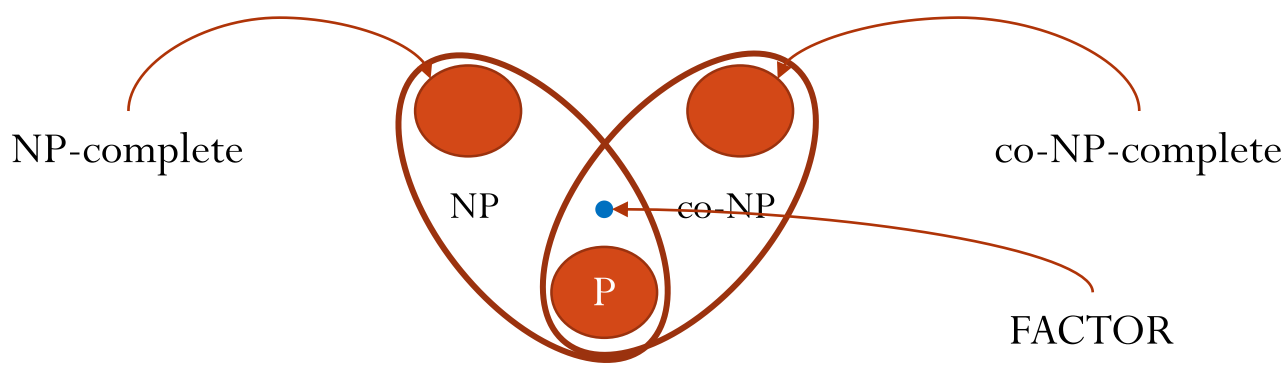
Computational Intractability: Complexity Classes

- **<u>PSPACE</u>**: The set of all problems that can be solved using polynomial amount of *space*.
- <u>Theorem</u>: **PSPACE** contains **P**.
- <u>Theorem</u>: **PSPACE** contains **NP**.



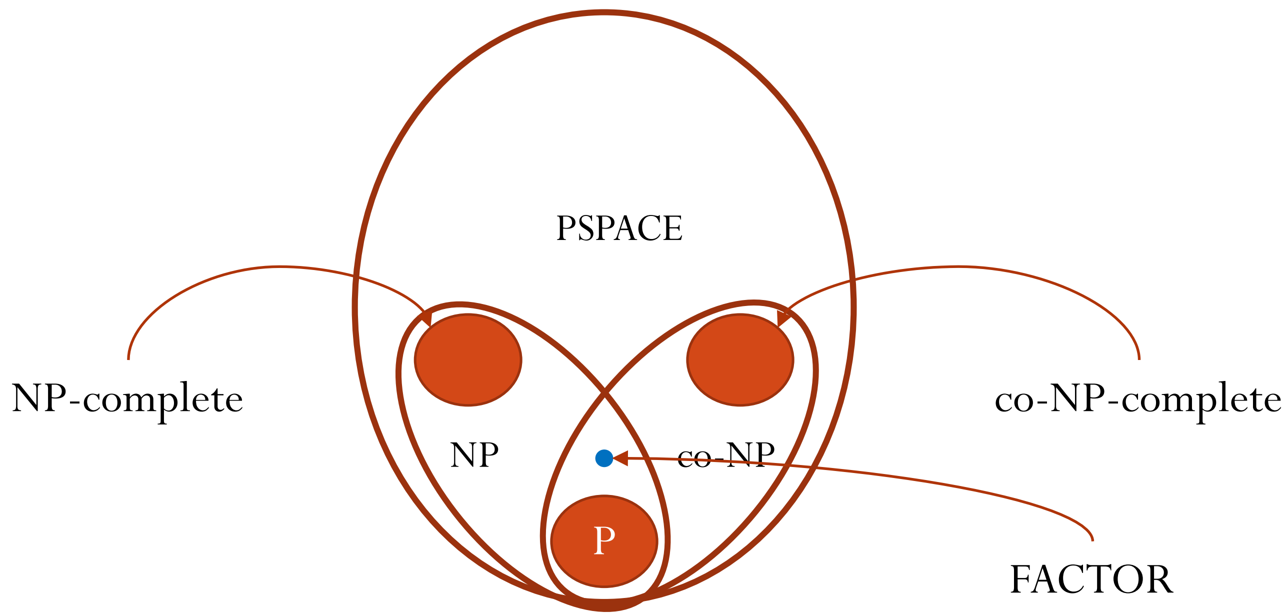
Computational Intractability: Complexity Classes

- **PSPACE**: The set of all problems that can be solved using polynomial amount of *space*.
- Theorem: **PSPACE** contains **P**.
- Theorem: **PSPACE** contains **NP**.
  - Proof idea: 3-SAT can be solved in polynomial space and for all problems X in **NP**, $X \leq_p$ 3-SAT.



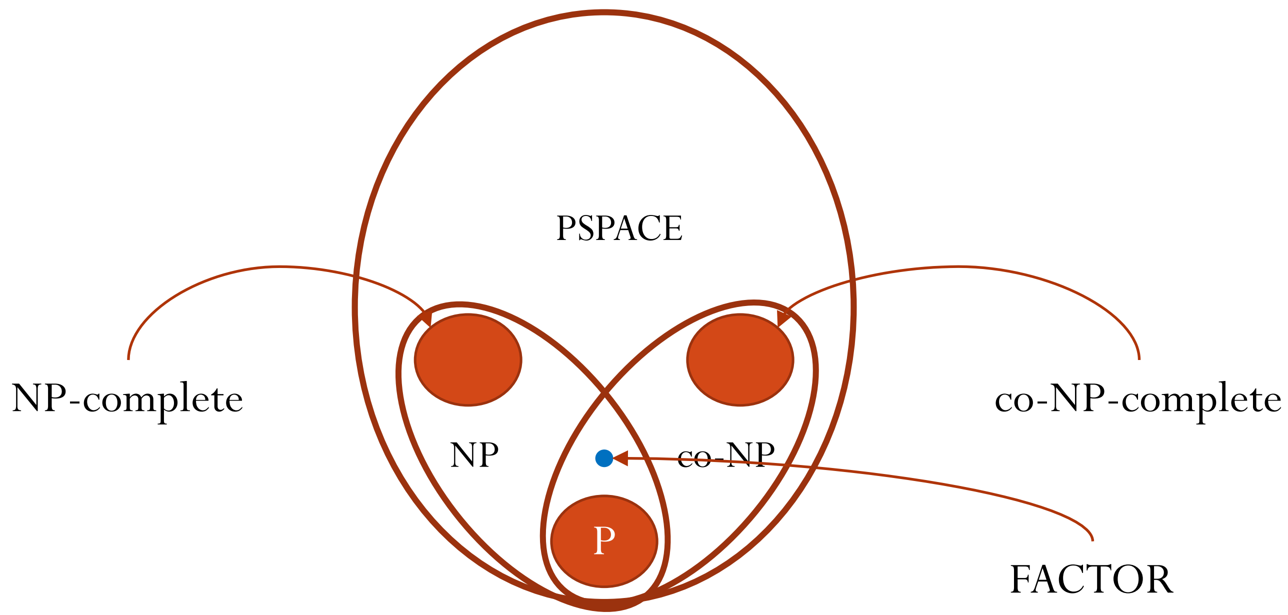
Computational Intractability: Complexity Classes

- **PSPACE**: The set of all problems that can be solved using polynomial amount of *space*.
- Theorem: **PSPACE** contains **P.**
- Theorem: **PSPACE** contains **NP**.
- Theorem: **PSPACE** contains **co-NP**.



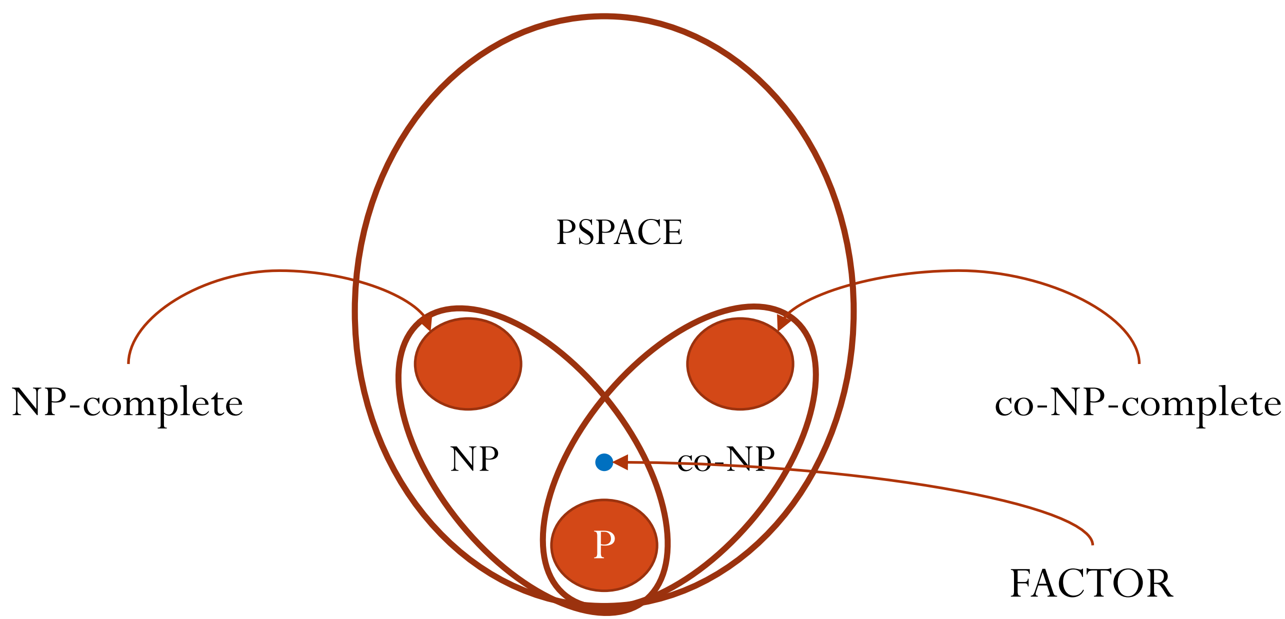
Computational Intractability: Complexity Classes

- Question: Are there problems in **PSPACE** that are not in simpler classes?
  - There are **PSPACE**-complete problems that are the hardest problems in **PSPACE** are not known to be in simpler classes.



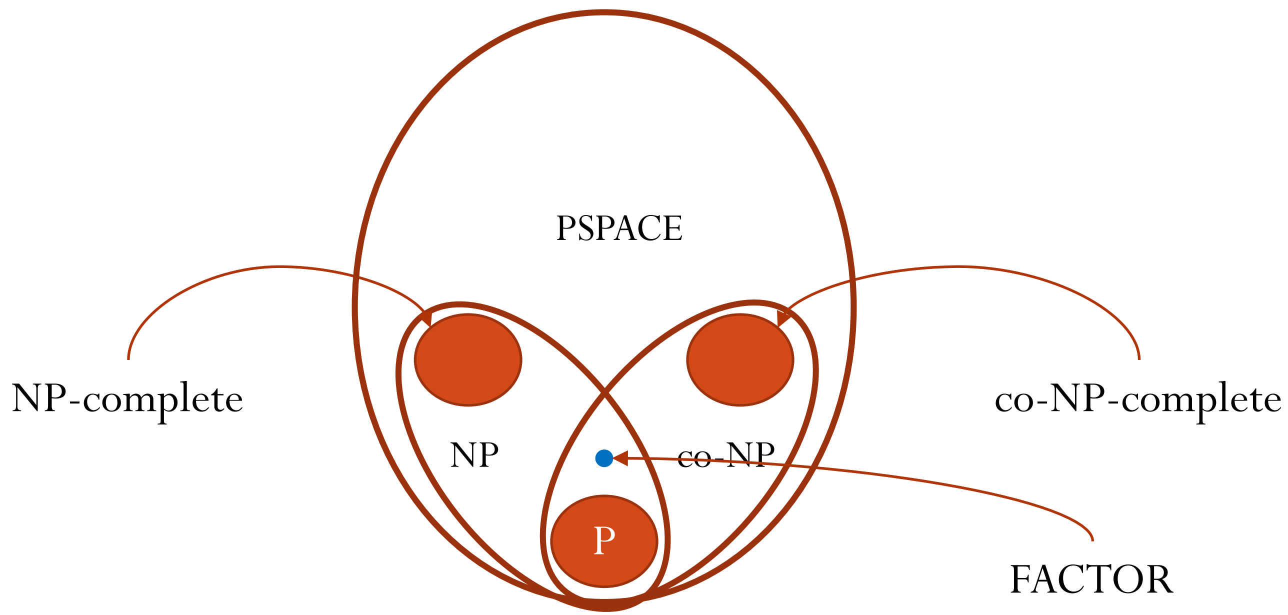
Computational Intractability: Complexity Classes

- **PSPACE-complete**: A problem X is **PSPACE**-complete if
  1. X is in **PSPACE**.
  2. For all Y in PSPACE, $Y \leq_p X$.
- Question: Are there natural problems that are **PSPACE**-complete?



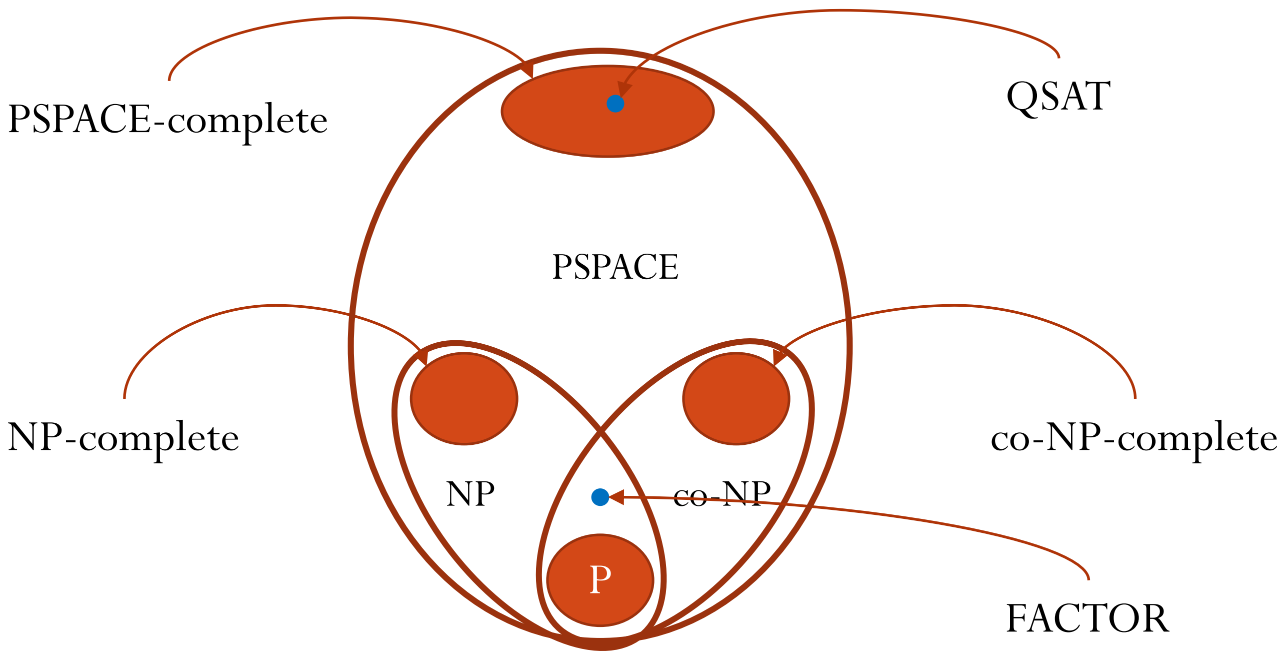
Computational Intractability: Complexity Classes

- Problem(Quantified-SAT): Given a quantified boolean formula $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \forall x_{n-1} \exists x_n \ \varphi(x_1, \ldots, x_n)$, determine if the formula is true.
  - This captures 2-player games.
- Theorem: QSAT is **PSPACE**-complete.



Computational Intractability: Complexity Classes

- Problem(Quantified-SAT): Given a quantified boolean formula $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \forall x_{n-1} \exists x_n \; \varphi(x_1, \ldots, x_n)$, determine if the formula is true.
  - This captures 2-player games.
- Theorem: QSAT is **PSPACE**-complete.



End