CSE 421 Winter 2025
Lecture 25:
NP-Complete 2

Nathan Brunelle

http://www.cs.uw.edu/421

Satisfiability

CNF formula example:

$$(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)$$

Defn: If there is some assignment of 0's and 1's to the variables that makes it true then we say the formula is satisfiable

- $(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)$ is satisfiable: $x_1 = x_3 = 1$
- $x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \neg x_3$ is not satisfiable.

3SAT: Given a CNF formula F with exactly 3 variables per clause, is F satisfiable?

More precise definition of NP

A decision problem A is in **NP** iff there is

- a polynomial time procedure VerifyA(.,.)

Such that:

- for every input x that is a **YES** for A there is a certificate t with $|t|$ polynomial in $|x|$ with VerifyA(x, t) = **YES**
- for every input x that is a **NO** for A there does not exist a certificate t with $|t|$ polynomial in $|x|$ with VerifyA(x, t) = **YES**

Steps for showing that a problem is in **NP**

1. Must be decision probem (**YES**/**NO**)
2. Describe what your certificates could look like for verification
3. Describe a verification algorithm VerifyA(x, t) where x is an instance of the problem and t is one of the certificates you just described, and show that it has these properties:
   1. For every given **YES** input x, there is at least one choice of t where VerifyA(x, t) is **YES**
   2. For any given **NO** input x, there is no choice of t where VerifyA(x, t) is **YES**
   3. VerifyA(x, t) runs in polynomial time

Verifying the certificate is efficient

3Color:

- Certificate: a coloring
- $Verify$ algorithm: Check that each vertex has one of only 3 colors and check that the endpoints of every edge have different colors
  - A valid coloring exists for any 3-colorable graph, but not for one that isn't 3-colorable

Independent-Set, **Clique**:

- Certificate: the set $\boldsymbol{U}$ of vertices
- $Verify$ algorithm: Check that $|\boldsymbol{U}| \geq \boldsymbol{k}$ and either no (**IS**) or all (**Clique**) edges on present on $\boldsymbol{U}$
  - A valid $\boldsymbol{U}$ only exists for yes instances

Vertex-Cover:

- Certificate: the set $\boldsymbol{W}$ of vertices
- $Verify$ algorithm: Check that $|\boldsymbol{W}| \leq \boldsymbol{k}$ and $\boldsymbol{W}$ touches every edge.
  - A valid $\boldsymbol{W}$ only exists for yes instances

- **3-SAT**:
  - Certificate: a truth assignment $\boldsymbol{\alpha}$ that makes the CNF formula $\boldsymbol{F}$ true.
  - $Verify$ algorithm: Evaluate $\boldsymbol{F}$ on the truth assignment $\boldsymbol{\alpha}$.
    - A valid truth assignment only exists for yes instances

$\mathbf{NP}$-hardness & $\mathbf{NP}$-completeness

Notion of hardness we **can** prove that is useful unless $\mathbf{P} = \mathbf{NP}$:

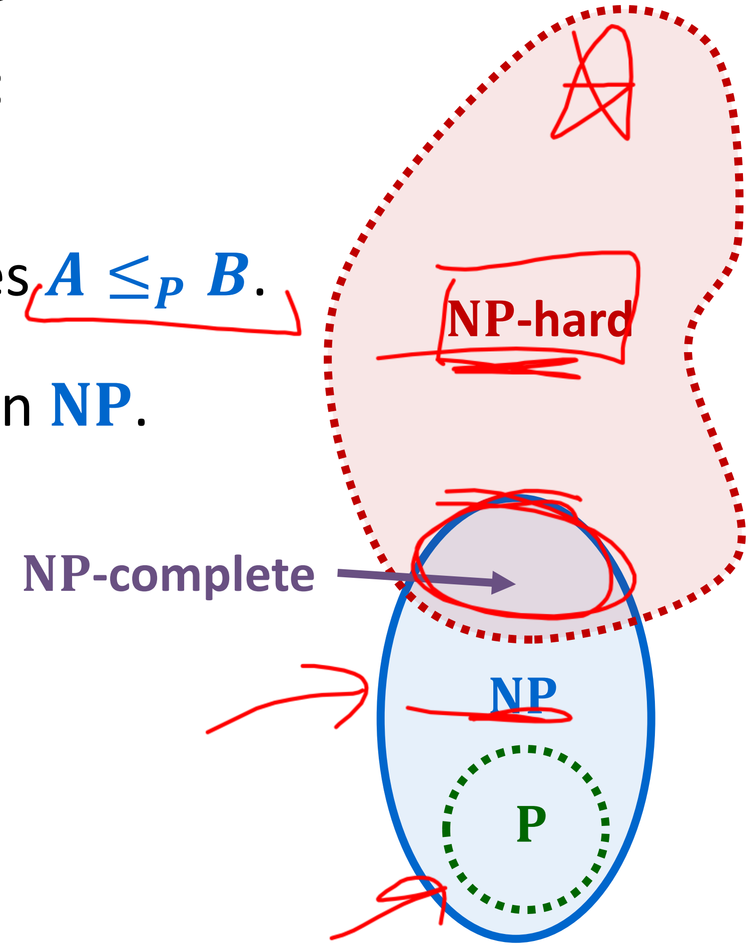
Defn: Problem $\boldsymbol{B}$ is $\mathbf{NP}$-hard iff **every** problem $\boldsymbol{A} \in \mathbf{NP}$ satisfies $\boldsymbol{A} \leq_{\boldsymbol{P}} \boldsymbol{B}$.

This means that $\boldsymbol{B}$ is at least as hard as every problem in $\mathbf{NP}$.

Defn: Problem $\boldsymbol{B}$ is $\mathbf{NP}$-complete iff

- $\boldsymbol{B} \in \mathbf{NP}$ and
- $\boldsymbol{B}$ is $\mathbf{NP}$-hard.

This means that $\boldsymbol{B}$ is a hardest problem in $\mathbf{NP}$.



Cook-Levin Theorem

Theorem [Cook 1971, Levin 1973]: **3SAT** is **NP**-complete

Proof: See CSE 431.

Corollary: If **3SAT** $\leq_P$ **B** then **B** is **NP**-hard.

Proof: Let **A** be an arbitrary problem in **NP**.
Since **3SAT** is **NP**-hard we have **A** $\leq_P$ **3SAT.**

Then **A** $\leq_P$ **3SAT** and **3SAT** $\leq_P$ **B** imply that **A** $\leq_P$ **B**.

Therefore every problem **A** in **NP** has **A** $\leq_P$ **B**
so **B** is **NP**-hard.

Cook & Levin did the hard work.

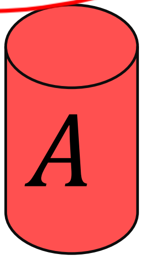
We only need to give one reduction to show that a problem is NP-hard!

What we know: 3Sat is NP-Hard

This reduction always exists!
(by definition of NP-Hard)

$O(n^p)$

Any NP problem



3Sat

$(x_1 \lor \neg x_3 \lor x_4) \land$
$(x_2 \lor \neg x_4 \lor x_3) \land$
$(x_2 \lor \neg x_1 \lor x_3)$

Procedure for converting instances of A into 3CNF formulas

Algorithm for solving 3SAT

Use the same answer

Solution for $\boldsymbol{A}$

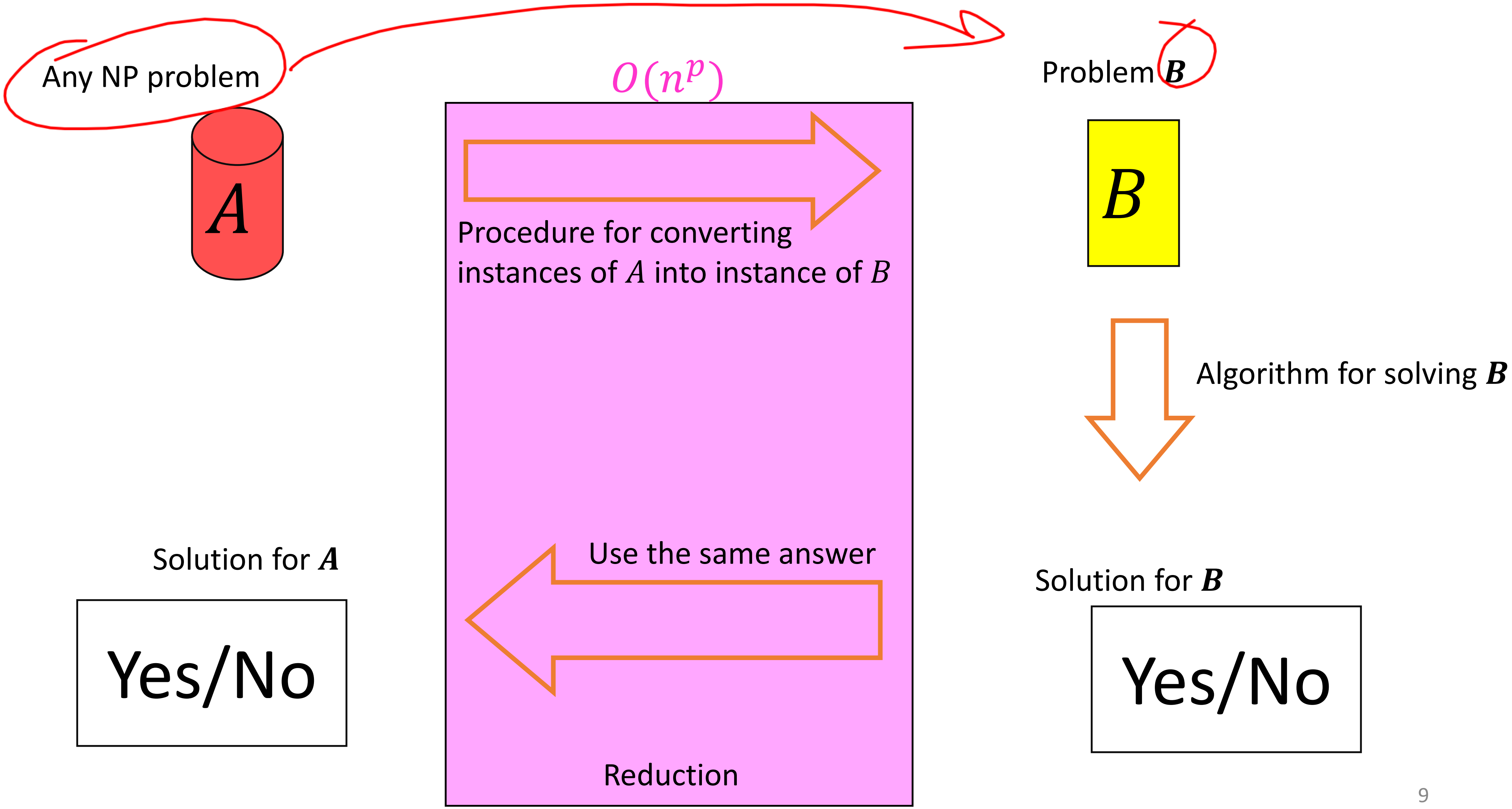
Yes/No

Reduction

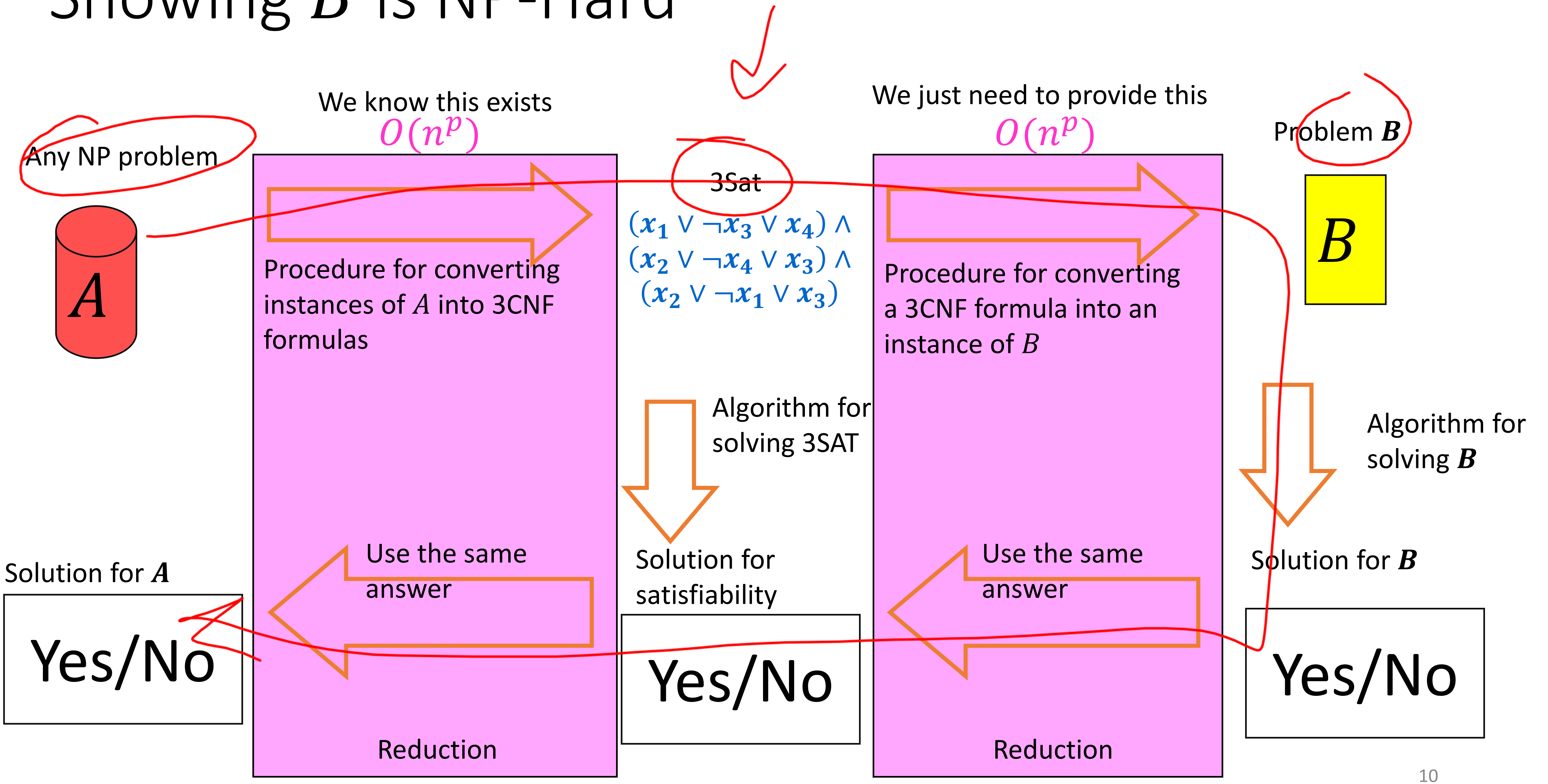
Solution for satisfiability

Yes/No

Goal: B is NP-Hard



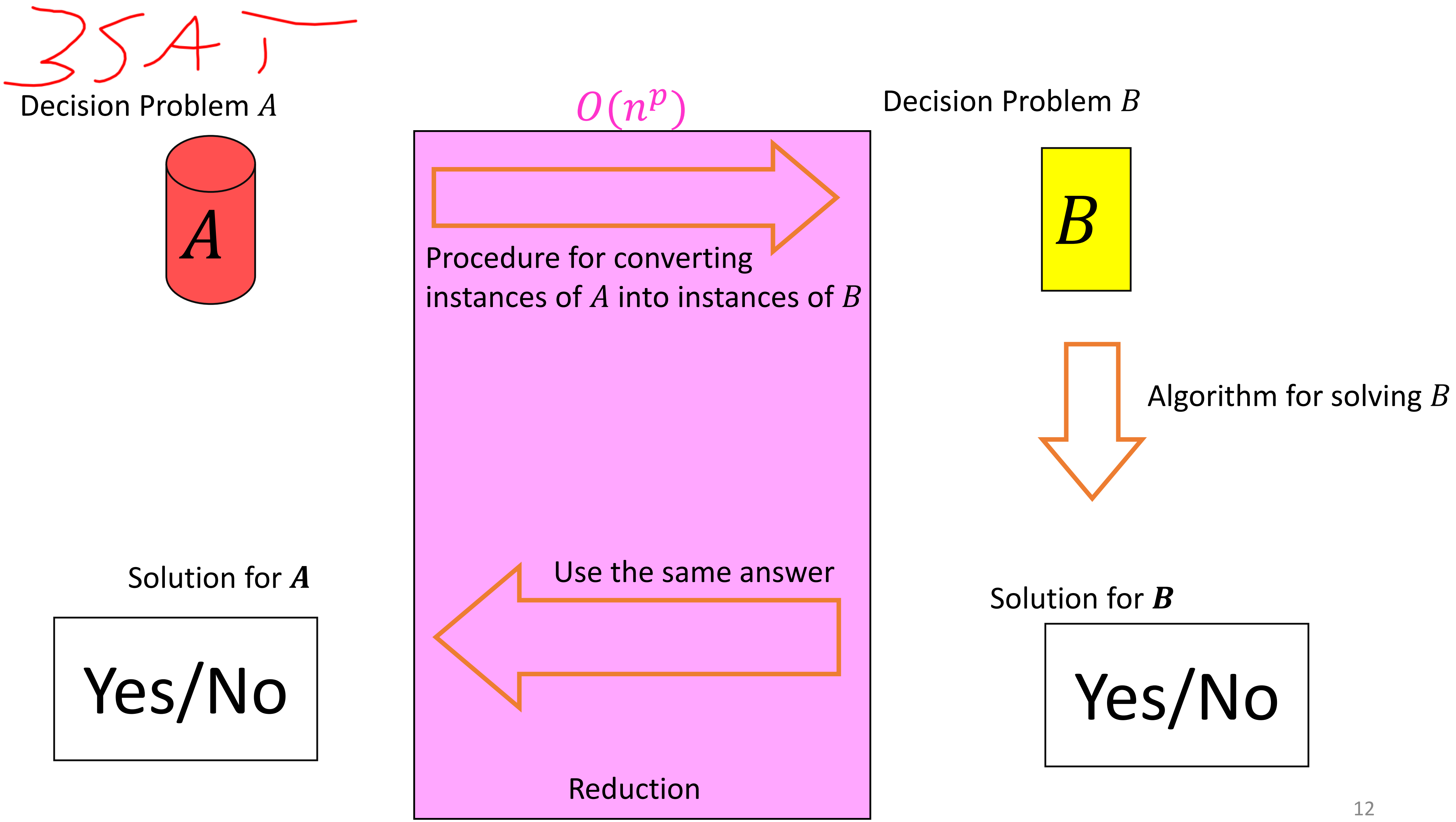
Showing B is NP-Hard



Steps to Proving Problem $\boldsymbol{B}$ is **NP**-complete

- Show $\boldsymbol{B}$ is in **NP**
  - State what the hint/certificate is.
  - Argue that it is polynomial-time to check and you won't get fooled.
- Show $\boldsymbol{B}$ is **NP**-hard:
  - State: "Reduction is from **NP**-hard Problem $\boldsymbol{A}$"
  - Show what the reduction function $\boldsymbol{f}$ is.
  - Argue that $\boldsymbol{f}$ is polynomial time.
  - Argue correctness in two directions:
    - $\boldsymbol{x}$ a **YES** for $\boldsymbol{A}$ implies $\boldsymbol{f(x)}$ is a **YES** for $\boldsymbol{B}$
      - Do this by showing how to convert a certificate for $\boldsymbol{x}$ being **YES** for $\boldsymbol{A}$ to a certificate for $\boldsymbol{f(x)}$ being a **YES** for $\boldsymbol{B}$.
    - $\boldsymbol{f(x)}$ a **YES** for $\boldsymbol{B}$ implies $\boldsymbol{x}$ is a **YES** for $\boldsymbol{A}$
      - … by converting certificates for $\boldsymbol{f(x)}$ to certificates for $\boldsymbol{x}$

Next up: Let's show Independent Set is NP-Hard



Showing Independent Set is NP-Hard

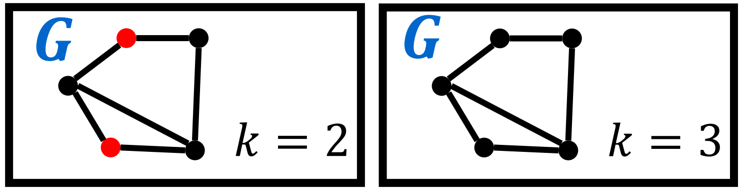
3Sat

$(x_1 \vee \neg x_3 \vee x_4) \wedge$
$(x_2 \vee \neg x_4 \vee x_3) \wedge$
$(x_2 \vee \neg x_1 \vee x_3)$

$O(n^p)$

Covert a 3CNF formula F into a graph G and a number k such that G has an independent set of size k if and only if F has a satisfying assignment

Independent Set



Algorithm for solving Independent Set

Solution for the instance of Independent Set

Yes/No

Use the same answer

Solution for the instance of 3Sat

Yes/No

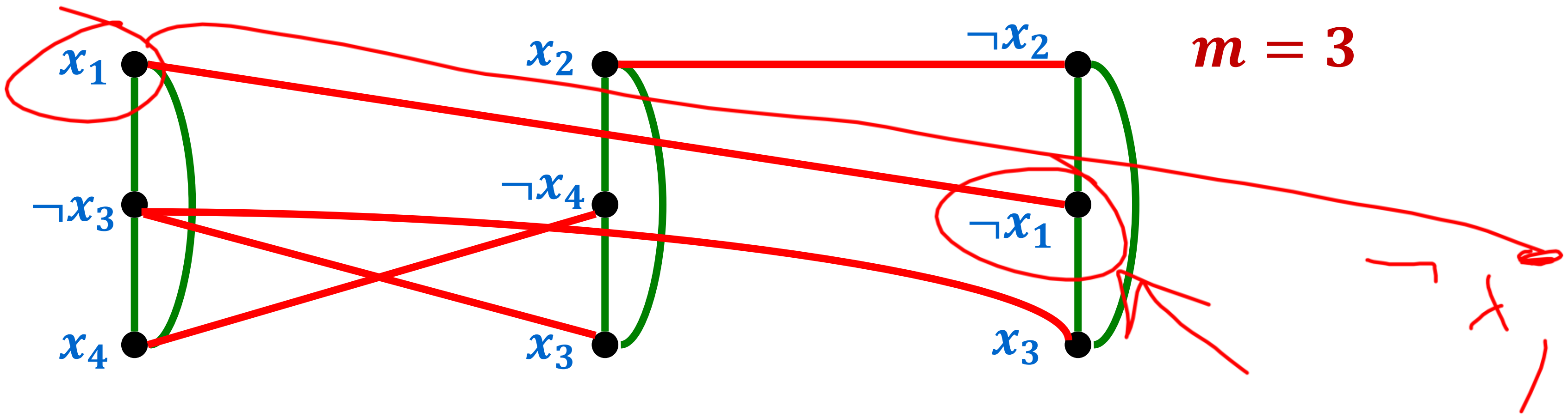
Reduction

Another $\mathbf{NP}$-complete problem: 3SAT $\le_{P}$ Independent-Set

1. The reduction:
   - Map CNF formula $\boldsymbol{F}$ to a graph $\boldsymbol{G}$ and integer $\boldsymbol{k}$
   - Let $\boldsymbol{m}$ = # of clauses of $\boldsymbol{F}$
   - Create a vertex in $\boldsymbol{G}$ for each literal occurrence in $\boldsymbol{F}$
     - $3m$ total vertices
   - Join two vertices $\boldsymbol{u}, \boldsymbol{v}$ in $\boldsymbol{G}$ by an edge iff
     - $\boldsymbol{u}$ and $\boldsymbol{v}$ correspond to literals in the same clause of $\boldsymbol{F}$ or
     - $\boldsymbol{u}$ and $\boldsymbol{v}$ correspond to literals $\boldsymbol{x}$ and $\neg\boldsymbol{x}$ (or vice versa) for some variable $\boldsymbol{x}$ (i.e. they contradict).
   - Set $\boldsymbol{k} = \boldsymbol{m}$
2. Clearly polynomial-time computable

Another **NP**-complete problem: 3SAT $\leq_P$ Independent-Set

$F = (x_1 \lor \neg x_3 \lor x_4) \land (x_2 \lor \neg x_4 \lor x_3) \land (\neg x_2 \lor \neg x_1 \lor x_3)$



$m = 3$

G has both kinds of edges.
The color is just to show why the edges were included.

$k = m$

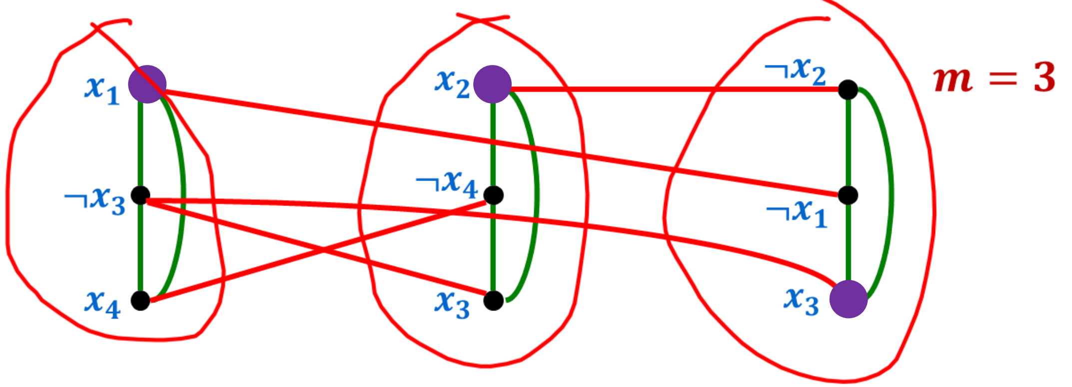
Correctness (⇒)

Suppose that F is satisfiable (**YES** for **3SAT**)

- Let α be a satisfying assignment; it satisfies at least one literal in each clause.
- Choose the set U in G to correspond to the **first satisfied literal in each clause**.
  - $|U| = m$
  - Since U has $\mathbf{1}$ vertex per clause, no same-clause edges inside U.
  - A truth assignment never satisfies both x and $\neg x$, so no contradicting-variable edges inside U.
  - Therefore U is an independent set of size m

Therefore (G, m) is a **YES** for **Independent-Set**.

$$F = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3) \wedge (\neg x_2 \vee \neg x_1 \vee x_3)$$



Satisfying assignment α:

$$\alpha(x_1) = \alpha(x_2) = \alpha(x_3) = \alpha(x_4) = 1$$

Set U marked in purple is independent.

Correctness (⇐)

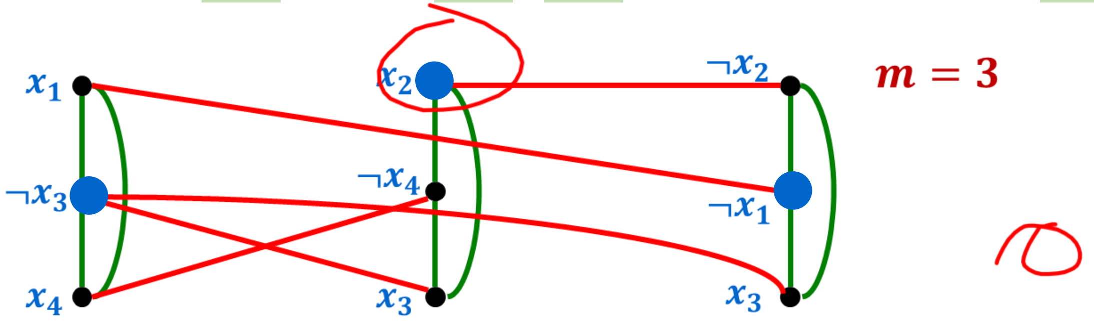
Suppose that G has an independent set of size m

((G, m) is a **YES** for **Independent-Set**)

- Let U be the independent set of size m;
- U must have one vertex per column (same-clause edges)
- Because of contraidict-variable edges, U doesn't have vertex labels with conflicting literals.
- Set all literals labelling vertices in U to true
- This may not be a total assignment but just extend arbitrarily to a total assignment α.
    - This assignment satisfies F since it makes at least one literal per clause true.

Therefore F is satisfiable and a **YES** for **3SAT**.

$F = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3) \wedge (\neg x_2 \vee \neg x_1 \vee x_3)$



Given independent set U of size m

Satisfying assignment α: Part defined by U:

$\alpha(x_1) = 0, \alpha(x_2) = 1, \alpha(x_3) = 0$

Set $\alpha(x_4) = 0$.

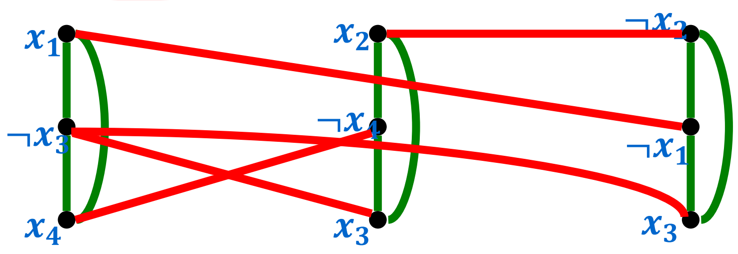
Showing Independent Set is NP-Hard

3Sat

$(x_1 \lor \neg x_3 \lor x_4) \land$
$(x_2 \lor \neg x_4 \lor x_3) \land$
$(x_2 \lor \neg x_1 \lor x_3)$

$O(n^p)$

Independent Set



$k = 3$

Make one node per literal, connect each to other nodes in the same clause, connect literals with their negations, set k to be the number of clauses

Algorithm for solving Independent Set

Solution for the instance of 3Sat

Yes/No

Use the same answer

Reduction

Solution for the instance of Independent Set

Yes/No

Many **NP**-complete problems

Since **3SAT** $\leq_P$ **Independent-Set**, **Independent-Set** is **NP**-hard.

We already showed that **Independent-Set** is in **NP**.

⇒ **Independent-Set** is **NP**-complete

Corollary: **Clique** and **Vertex-Cover** are also **NP**-complete.

Proof: We already showed that all are in **NP**.

We also showed that **Independent-Set** polytime reduces to all of them.

Combining this with **3SAT** $\leq_P$ **Independent-Set** we get that all are **NP**-hard.

NP-complete problems so far

So far:

3SAT → Independent-Set → Clique

$$\downarrow$$

Vertex-Cover

4-Satisfiability

CNF formula example:

$$(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)$$

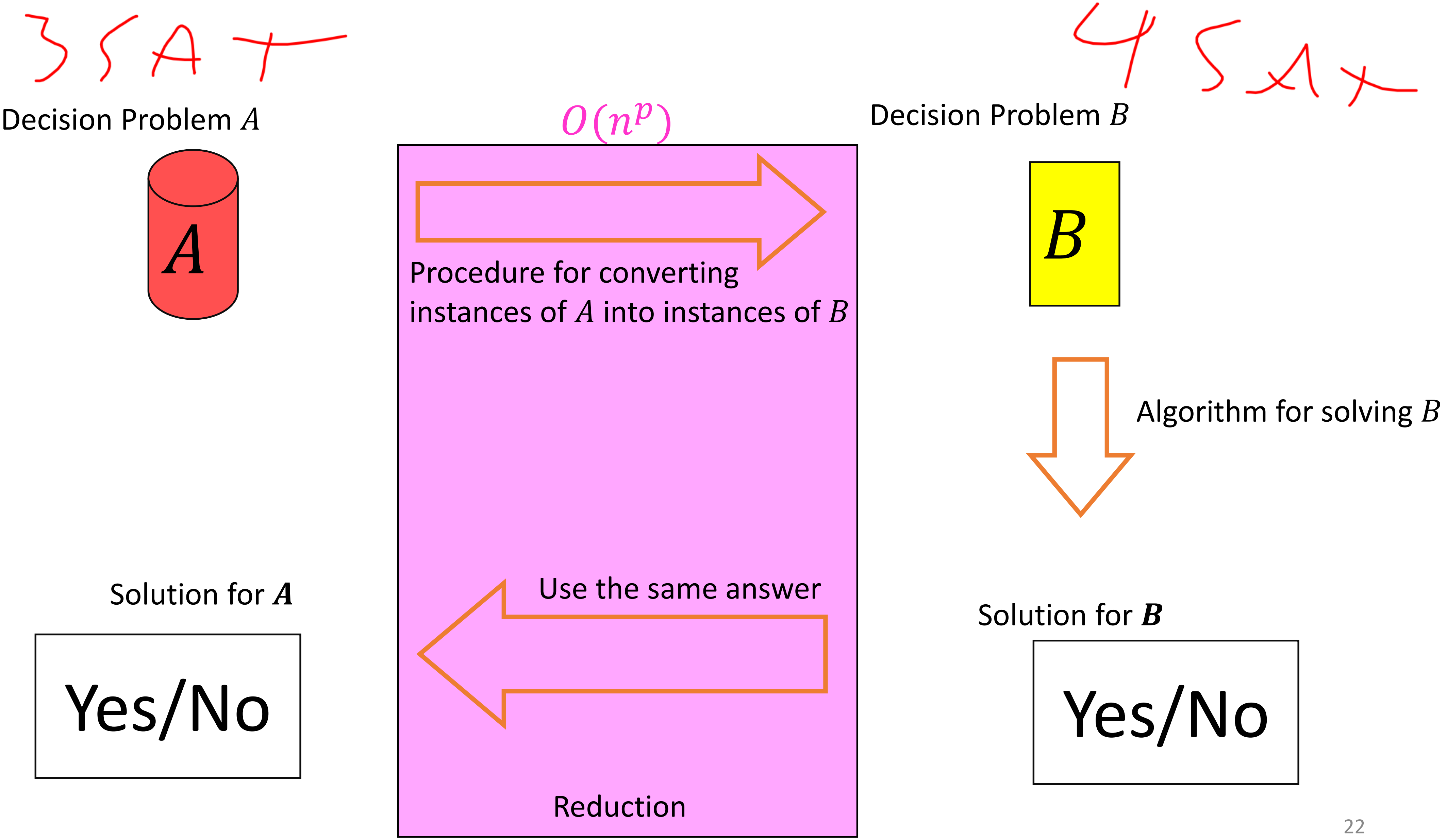
Defn: If there is some assignment of 0's and 1's to the variables that makes it true then we say the formula is satisfiable

- $(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)$ is satisfiable: $x_1 = x_3 = 1$
- $x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \neg x_3$ is not satisfiable.

3SAT: Given a CNF formula F with exactly 3 variables per clause, is F satisfiable?

4SAT: Given a CNF formula F with exactly 4 variables per clause, is F satisfiable?

Let’s show 4Sat is NP-Hard



Showing Independent Set is NP-Hard

3Sat

$(x_1 \vee \neg x_3 \vee x_4) \wedge$
$(x_2 \vee \neg x_4 \vee x_3) \wedge$
$(x_2 \vee \neg x_1 \vee x_3)$

$O(n^p)$

Covert a 3CNF formula F into a 4CNF formula F' such that F' has a satisfying assignment if and only if F has a one

4Sat

$(x_1 \vee \neg x_3 \vee x_4 \vee x_5) \wedge$
$(x_2 \vee \neg x_4 \vee x_3 \vee \neg x_5) \wedge$
$(x_2 \vee \neg x_1 \vee x_3 \vee x_6)$

Algorithm for solving 4Sat

Solution for the instance of Independent Set

Yes/No

Use the same answer

Solution for the instance of 3Sat

Yes/No

Reduction

3Sat $\leq_P$ 4Sat: A False Start (pun intended)

Goal: Covert a 3CNF formula F into a 4CNF formula F' such that F' has a satisfying assignment if and only if F has a one

Idea: Given a 3CNF formula, add one more variable per clause without changing its satisfiability

$$F = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)$$

This almost works: Add “false” to each clause

$$F' = (x_1 \vee \neg x_3 \vee x_4 \vee \mathit{false}) \wedge (x_2 \vee \neg x_4 \vee x_3 \vee \mathit{false}) \wedge (x_2 \vee \neg x_1 \vee x_3 \vee \mathit{false})$$

The resulting formula is logically equivalent to the original

Problem: This violates the definition of a CNF formula. The definition doesn’t allow for Boolean constants, only variables

3Sat $\leq_P$ 4Sat: The Reduction

Goal: Covert a 3CNF formula $\boldsymbol{F}$ into a 4CNF formula $\boldsymbol{F'}$ such that $\boldsymbol{F'}$ has a satisfying assignment if and only if $\boldsymbol{F}$ has a one

Idea: Given a 3CNF formula, add one more variable per clause without changing its satisfiability

$$\boldsymbol{F = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3) \wedge (x_2 \vee \neg x_1 \vee x_3)}$$

Solution: Add the same variable to each clause, then add one or more clauses to guarantee that variable must be false

$$\boldsymbol{F' = (x_1 \vee \neg x_3 \vee x_4 \vee y) \wedge (x_2 \vee \neg x_4 \vee x_3 \vee y) \wedge (x_2 \vee \neg x_1 \vee x_3 \vee y) \wedge (\neg y \vee \neg y \vee \neg y \vee \neg y)}$$

Showing Independent Set is NP-Hard

3Sat

$(x_1 \vee \neg x_3 \vee x_4) \wedge$
$(x_2 \vee \neg x_4 \vee x_3) \wedge$
$(x_2 \vee \neg x_1 \vee x_3)$

$O(n^p)$

Add variable y to each clause of F, then add the new clause $(\neg y \vee \neg y \vee \neg y \vee \neg y)$

4Sat

$(x_1 \vee \neg x_3 \vee x_4 \vee y)$
$\wedge (x_2 \vee \neg x_4 \vee x_3 \vee y)$
$\wedge (x_2 \vee \neg x_1 \vee x_3 \vee y)$
$\wedge (\neg y \vee \neg y \vee \neg y \vee \neg y)$

Algorithm for solving 4Sat

Solution for the instance of Independent Set

Yes/No

Use the same answer

Reduction

Solution for the instance of 3Sat

Yes/No

Correctness

- $\boldsymbol{F}$ satisfiable $\Rightarrow$ $\boldsymbol{F'}$ satisfiable

Let a be an assignment of true/false to each variable that satisfies $\boldsymbol{F}$. In this case, since all except one clause of $\boldsymbol{F'}$ share variables with clauses from $\boldsymbol{F}$, a satisfies all clauses of $\boldsymbol{F'}$ except that one new clause. This new clause can be satisfied by $y = false$.

- $\boldsymbol{F'}$ satisfiable $\Rightarrow$ $\boldsymbol{F}$ satisfiable

Let a' be an assignment of true/false to each variable that satisfies $\boldsymbol{F'}$. Because a' must satisfy all clauses, and the only way to satisfy the new clause is $y = false$. Since $y = false$ in a', all other clauses are logically equivalent to the original clauses from $\boldsymbol{F}$.

Recall: Graph Colorability

Defn: A undirected graph $G = (V, E)$ is k-colorable iff
we can assign one of k colors to each vertex of V s.t.
for every edge (u, v) has different colored endpoints, $\chi(u) \neq \chi(v)$.
"edges are not monochromatic"

Theorem: 3Color is $\mathbf{NP}$-complete

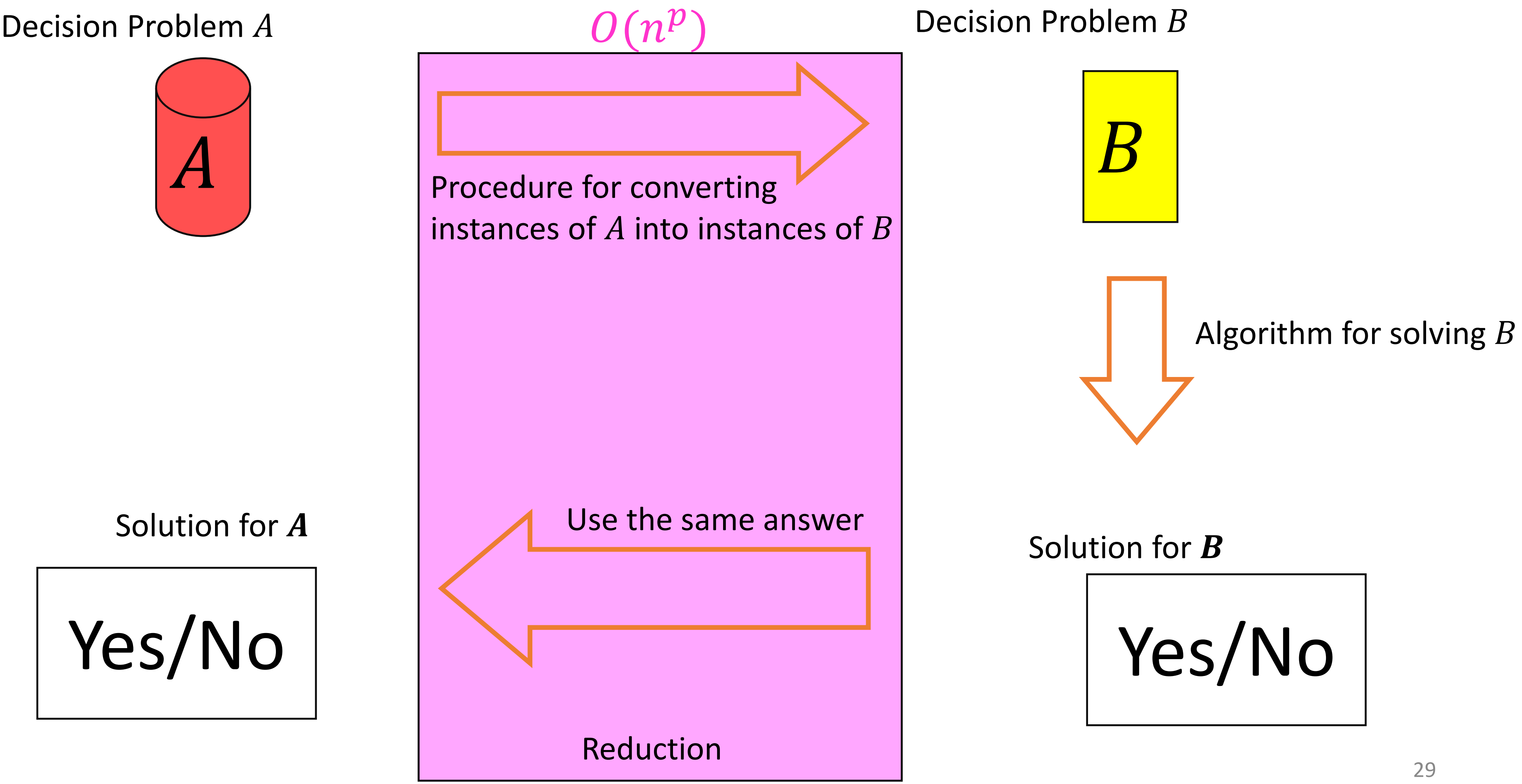
Proof:

1. **3Color** is in $\mathbf{NP}$:
   - We already showed this; the certificate was the coloring.
2. **3Color** is $\mathbf{NP}$-hard:

   Claim: 3SAT$\leq_P$3Color

   We need to find a function f that maps a 3CNF formula F to a graph G s.t.
   F is satisfiable $\Leftrightarrow$ G is 3-colorable.

Next up: Let’s show 3Color is NP-Hard



Showing 3Color is NP-Hard

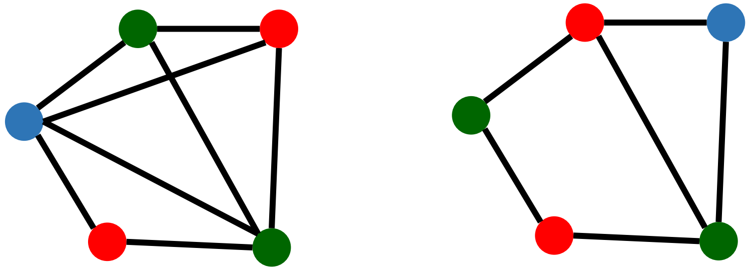
3Sat

$$(x_1 \lor \neg x_3 \lor x_4) \land$$
$$(x_2 \lor \neg x_4 \lor x_3) \land$$
$$(x_2 \lor \neg x_1 \lor x_3)$$

$O(n^p)$

Covert a 3CNF formula F into a graph G such that G is 3 colorable if and only if F has a satisfying assignment

3Color



Algorithm for solving Independent Set

Solution for the instance of Independent Set

Yes/No

Use the same answer

Reduction

Solution for the instance of 3Sat

Yes/No

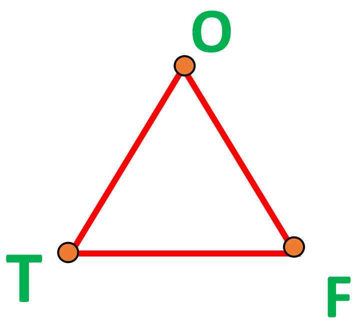
3SAT $\leq_P$ 3Color

Start with a base triangle with vertices **T**, **F**, and **O**.

We can assume that **T**, **F**, and **O** are the three colors used.

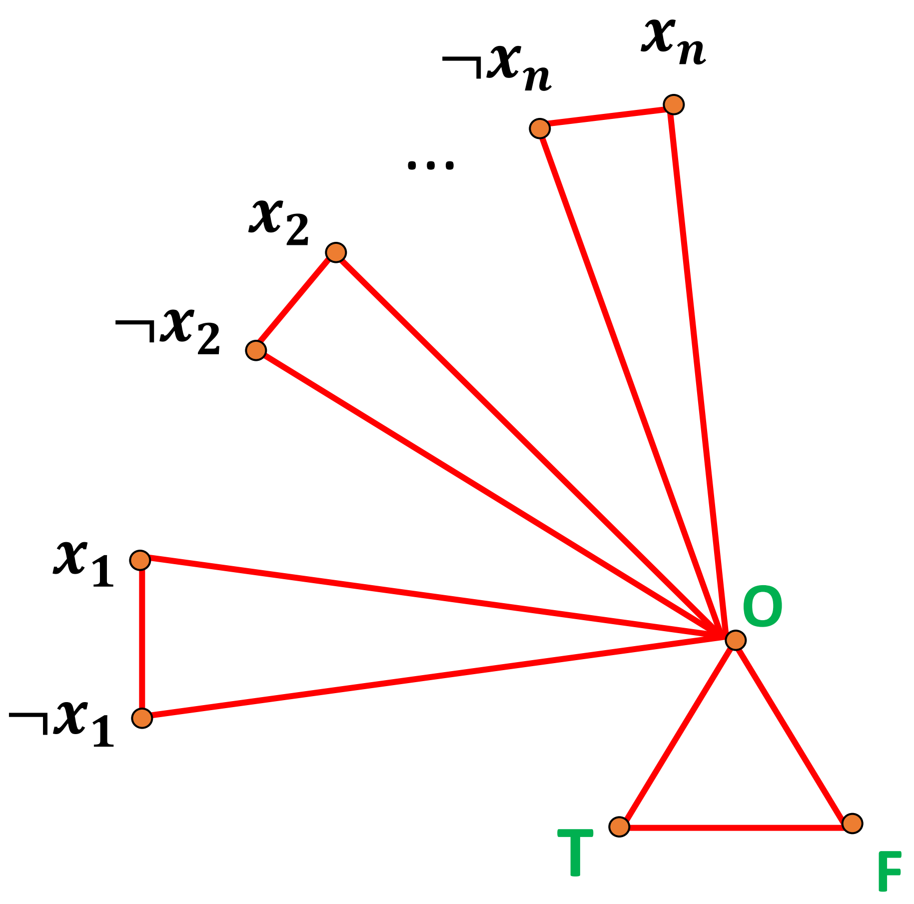
- Intuition: **T** and **F** will stand for *true* and *false*; **O** will stand for *other*.

To represent the properties of the 3CNF formula $\boldsymbol{F}$ we will need both a Boolean variable part and a clause part.



Base Triangle

3SAT $\leq_P$ 3Color



Base Triangle

Boolean variable part:

- For each Boolean variable add a triangle with two nodes labelled by literals as shown.
- Since both nodes are joined to node **O** and to each other, they must have opposite colors **T** and **F** in any 3-coloring.
- So, any 3-coloring corresponds to a unique truth assignment.

3SAT $\leq_P$ 3Color

Idea:

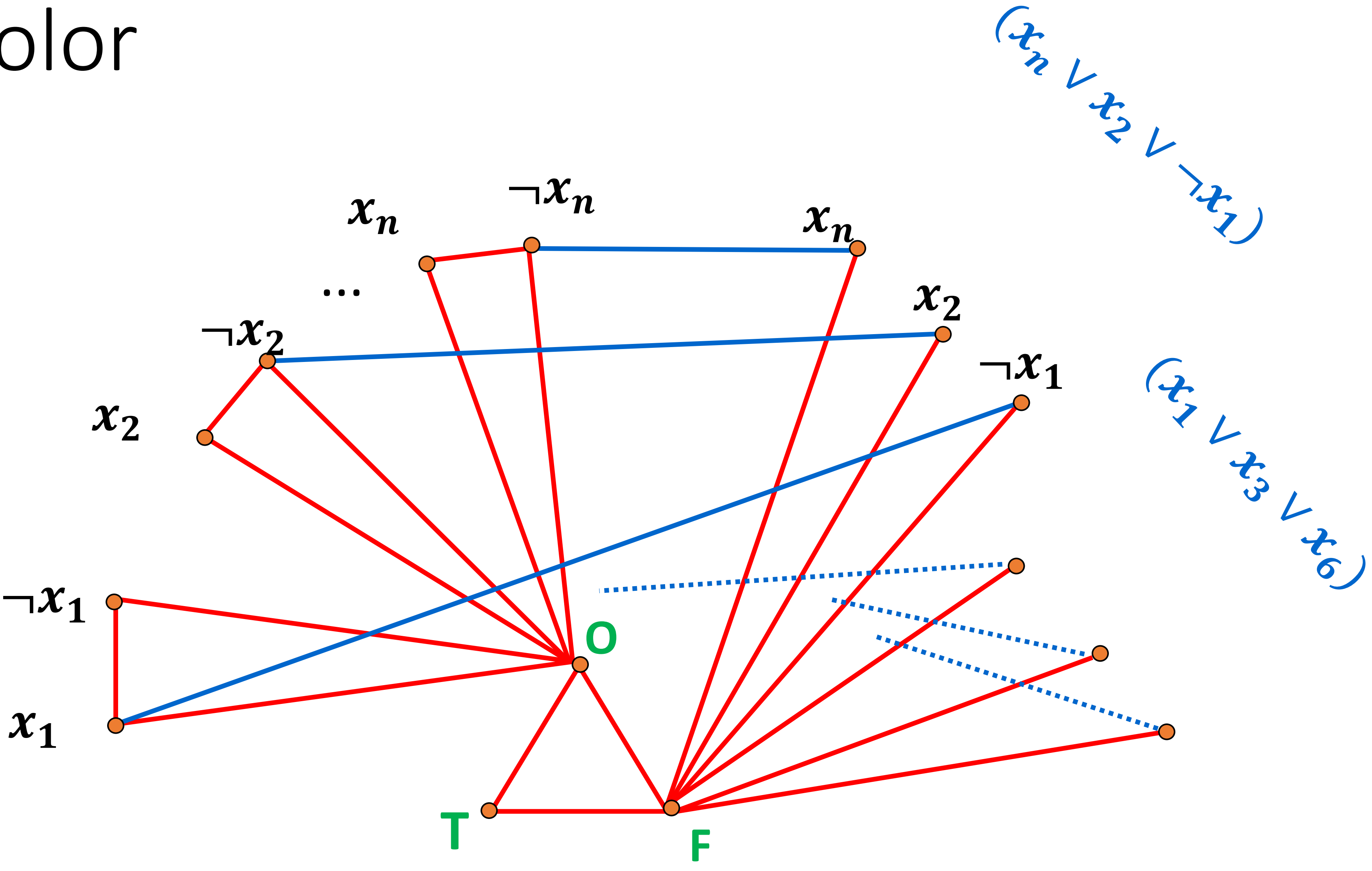
Create a "middle" node per literal for each clause, we will consider a T-colored middle node to satisfy a clause.

In the graph:

For each clause of F add 3 "middle" nodes. Then:

- Join each middle node to it opposite literal node
- Join each middle node to F

Now each middle node must be either T or O, and any connect to something T-colored must be O-colored



3SAT $\leq_P$ 3Color

Idea:

Force at least one middle node per clause to be T-colored.

In the graph:

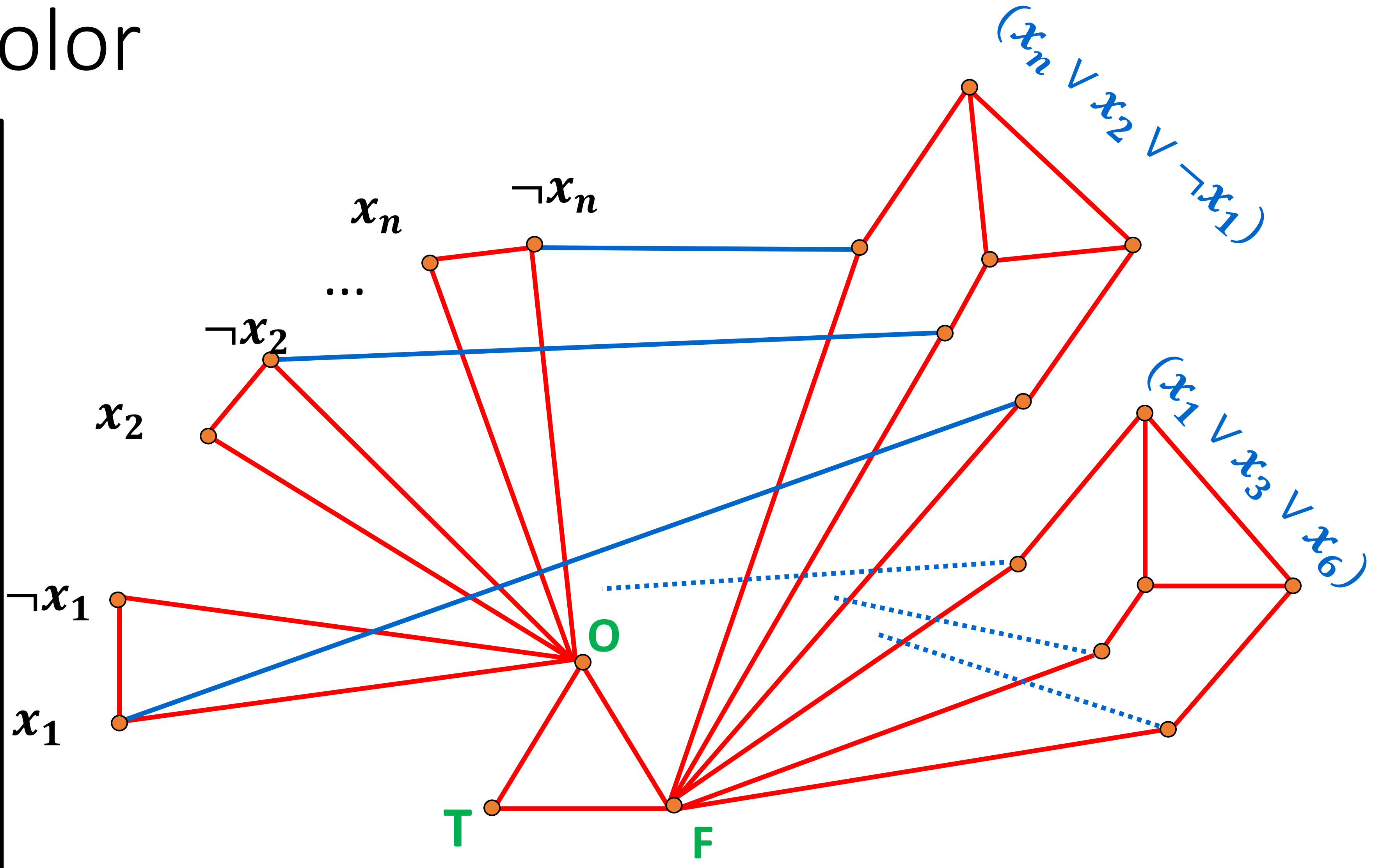
For each clause of F add an outer triangle.

- Join each middle node a vertex in the triangle

No middle node can be F-colored (all connect to F)

Not all middle nodes are O-colored (because something in the outer triangle must be)

So at least one is T-colored



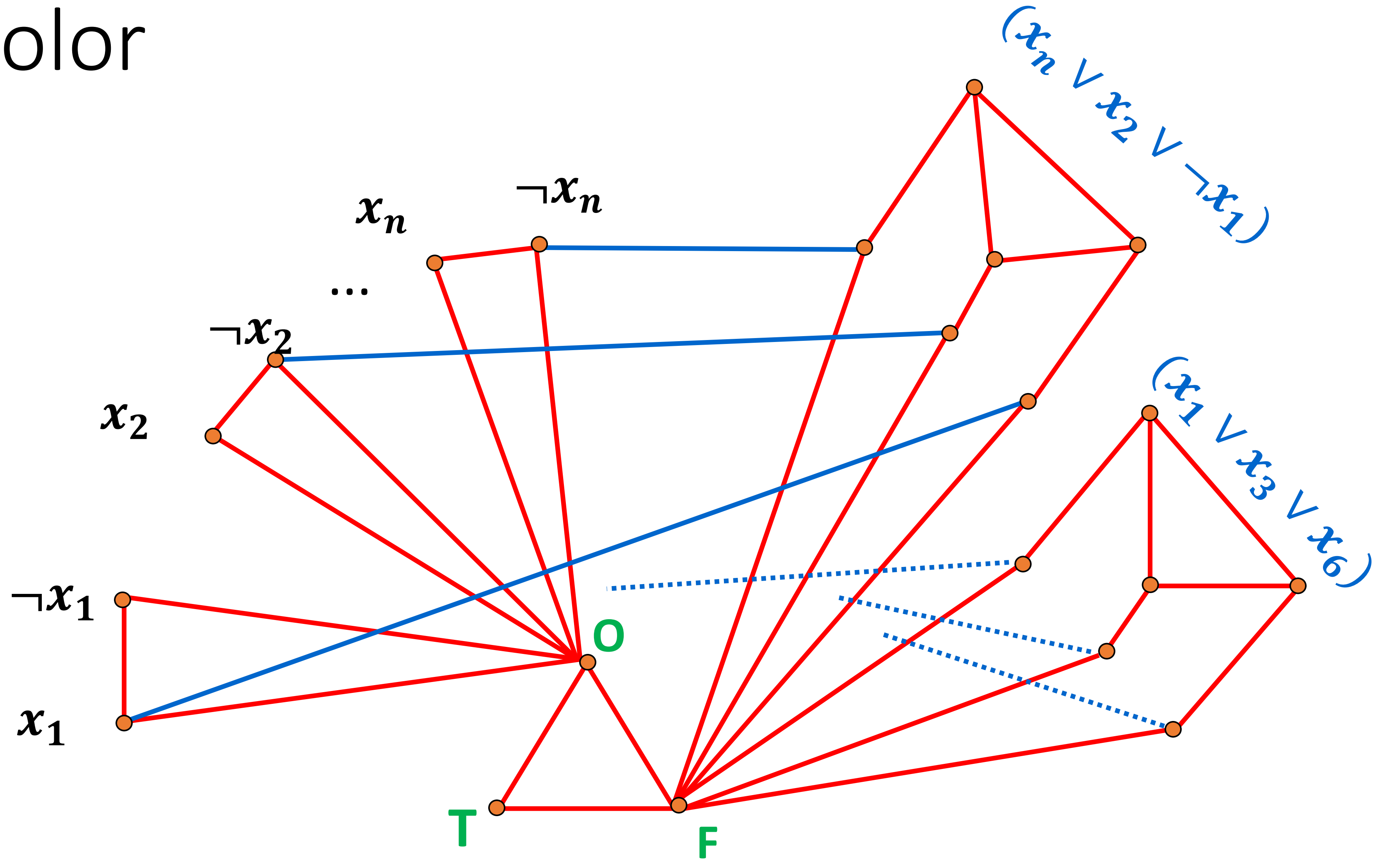
3SAT $\leq_P$ 3Color

Key property:

In any 3-coloring:

outer nodes either **T** or **O**

inner triangle must use **O**



Showing 3Color is NP-Hard

3Sat

$(x_1 \lor \neg x_3 \lor x_4) \land$
$(x_2 \lor \neg x_4 \lor x_3) \land$
$(x_2 \lor \neg x_1 \lor x_3)$

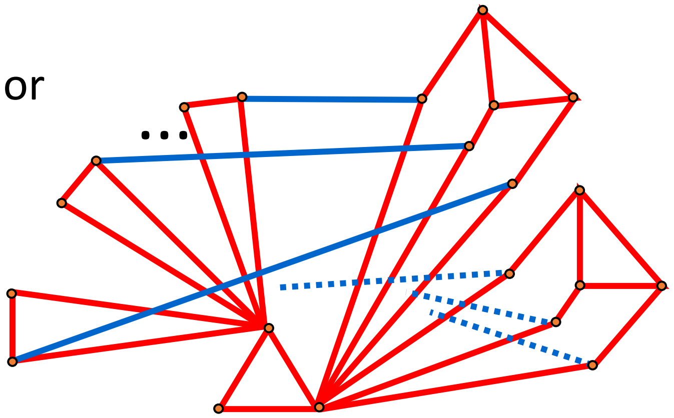
$O(n^p)$

Create “base triangle” and one node per variable and negation. Connect each variable node to the “false color” node. Per clause, create a triangle and one middle node per literal. Connect each middle to the triangle, false, and the opposite variable

Use the same answer

Reduction

3Color



Algorithm for solving 3Color

Solution for the instance of 3Sat

Yes/No

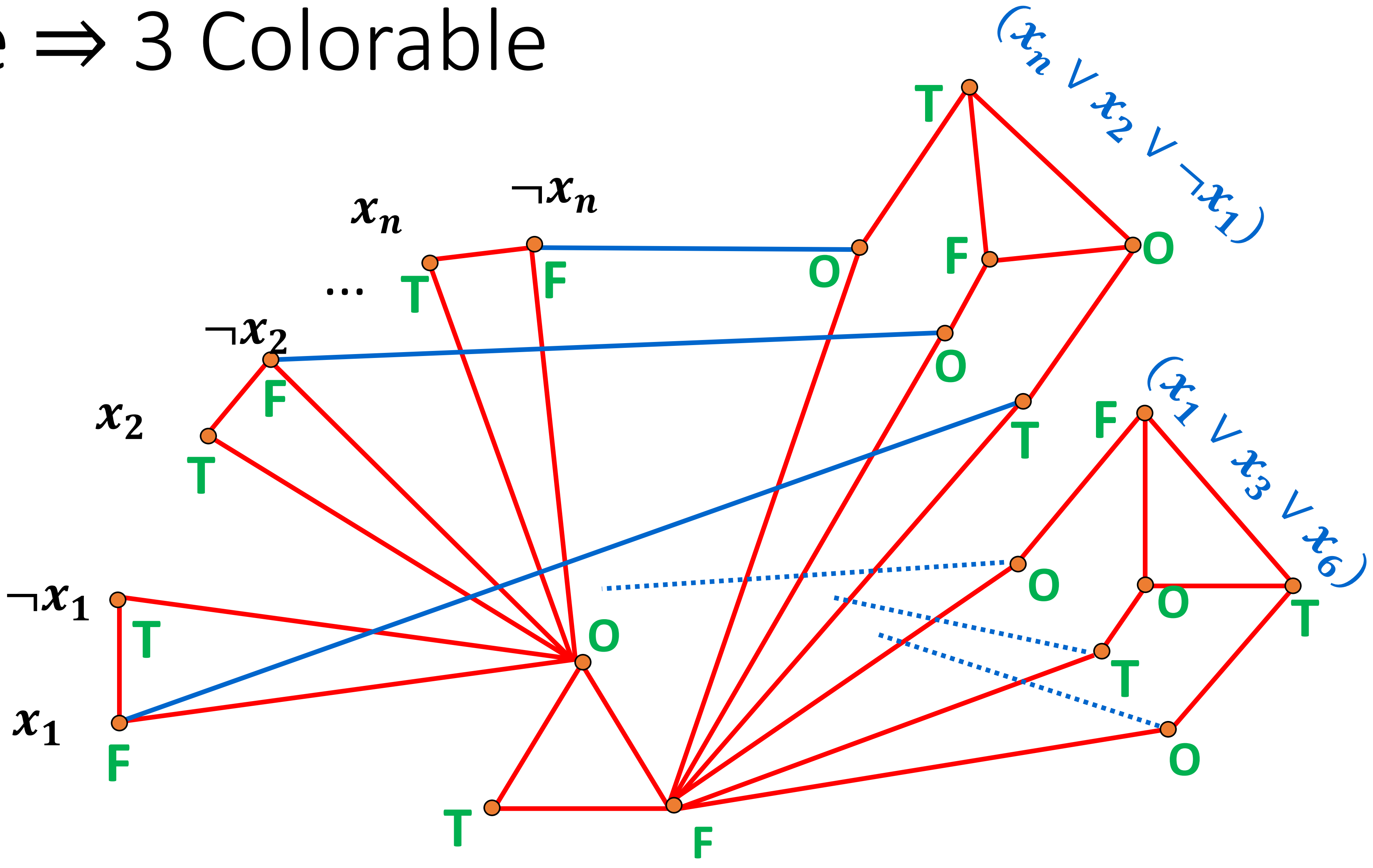
Solution for the instance of 3Color

Yes/No

F satisfiable ⇒ 3 Colorable

Suppose F is satisfiable. We can then 3-Color the graph by:

- Make each True literal node T-colored
- Make each False literal node F-colored
- Make one True middle node per clause T-colored
- Make the remaining middle nodes O-colored
- Color each outer triangle (node connect to the T-colored middle node will be O-colored, the others can be either T-colored or F-colored)



3 Colorable ⇒ F satisfiable

Suppose the graph is 3-colorable. We can satisfy F by:

- Making each T-colored literal node True and each F-colored literal node False
  - No nodes are O-colored, so this will work out
- We know this satisfies F because:
  - Each clause will have one T-colored middle node (connected to the O-colored outer triangle node) which matches the color of its equivalent literal