NP-Completeness

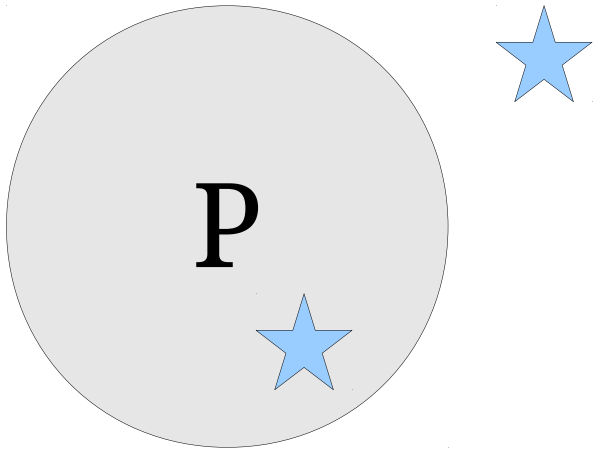
Part II

Outline for Today

- **Recap from Last Time**
  - What is **NP**-completeness again, anyway?
- **3SAT**
  - A simple, canonical **NP**-complete problem.
- **Independent Sets**
  - Discovering a new **NP**-complete problem.
- **Gadget-Based Reductions**
  - A common technique in **NP** reductions.
- **3-Colorability**
  - A more elaborate **NP**-completeness reduction.

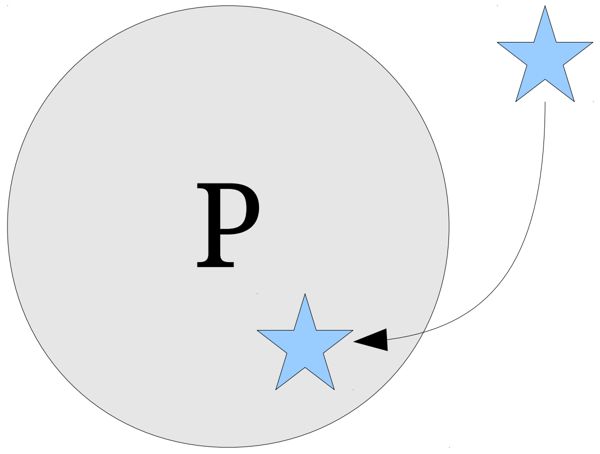
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.



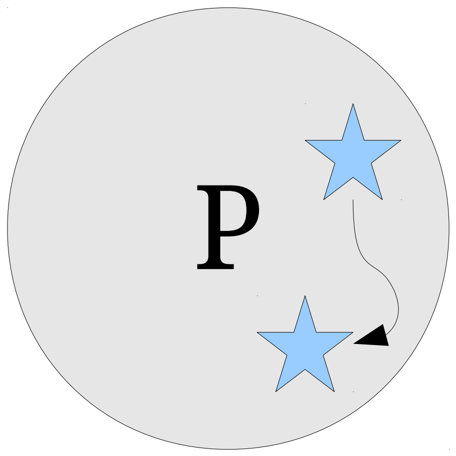
Polynomial-Time Reductions

- If $A \leq_{\text{P}} B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.



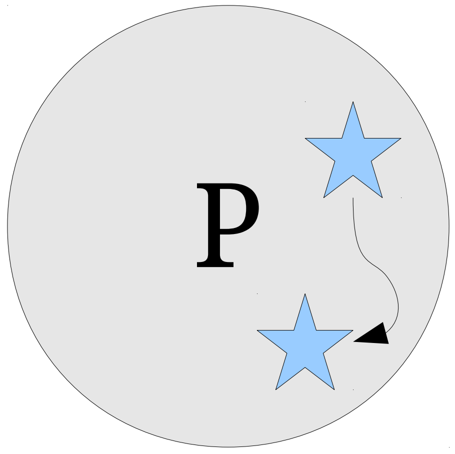
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.



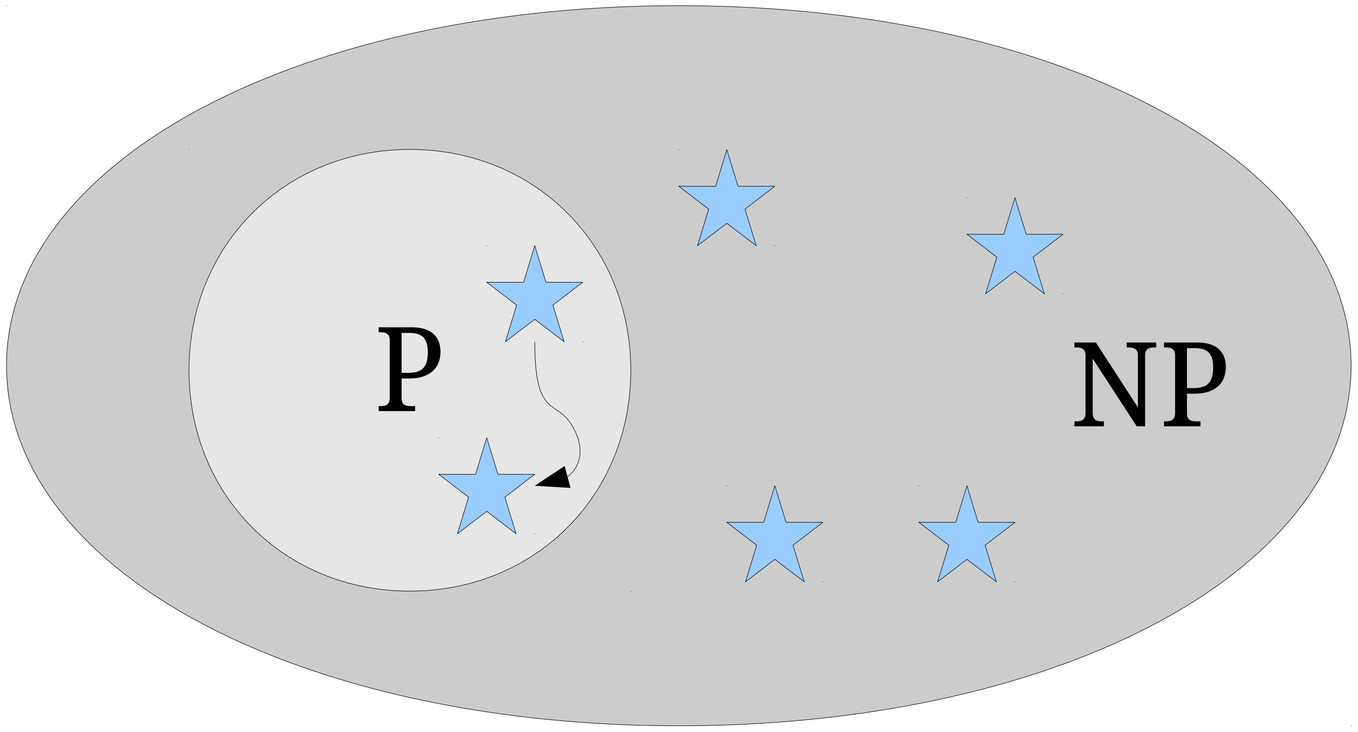
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_P B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



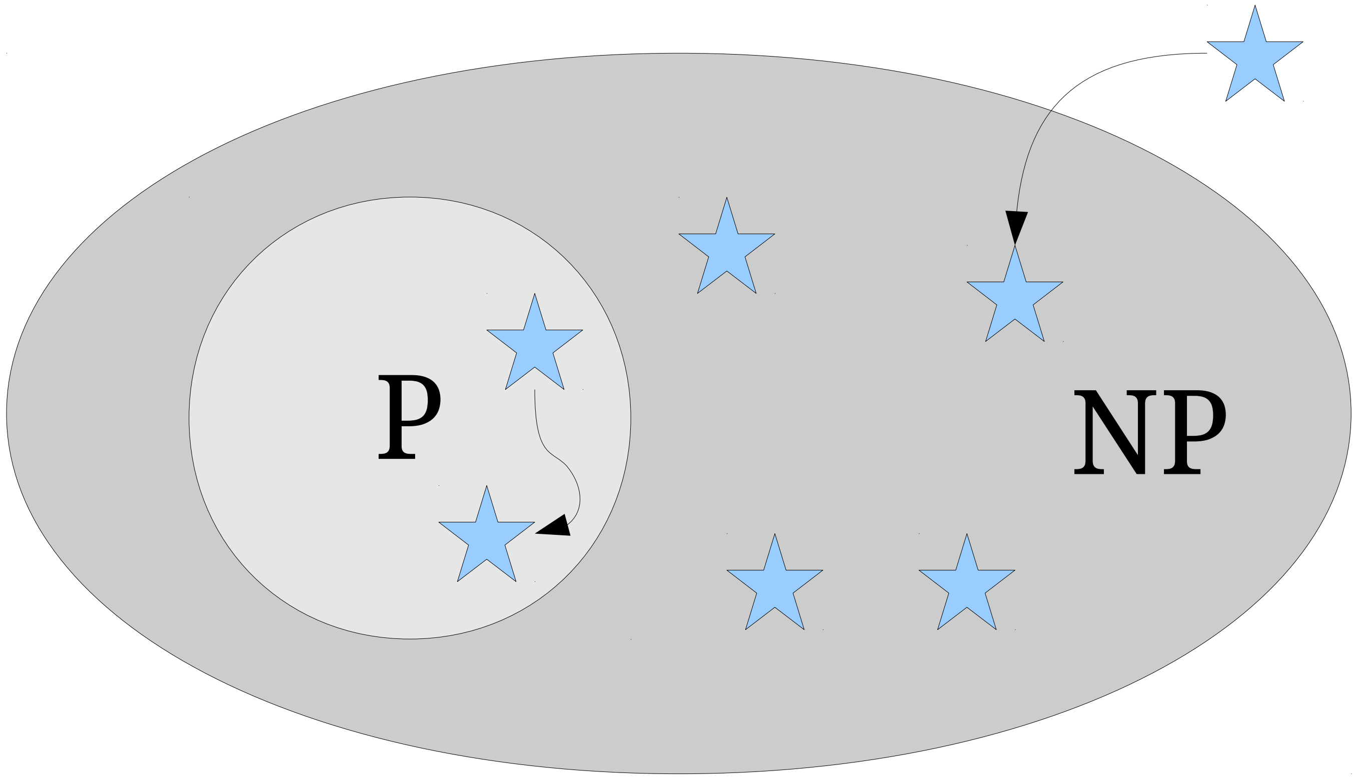
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_P B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



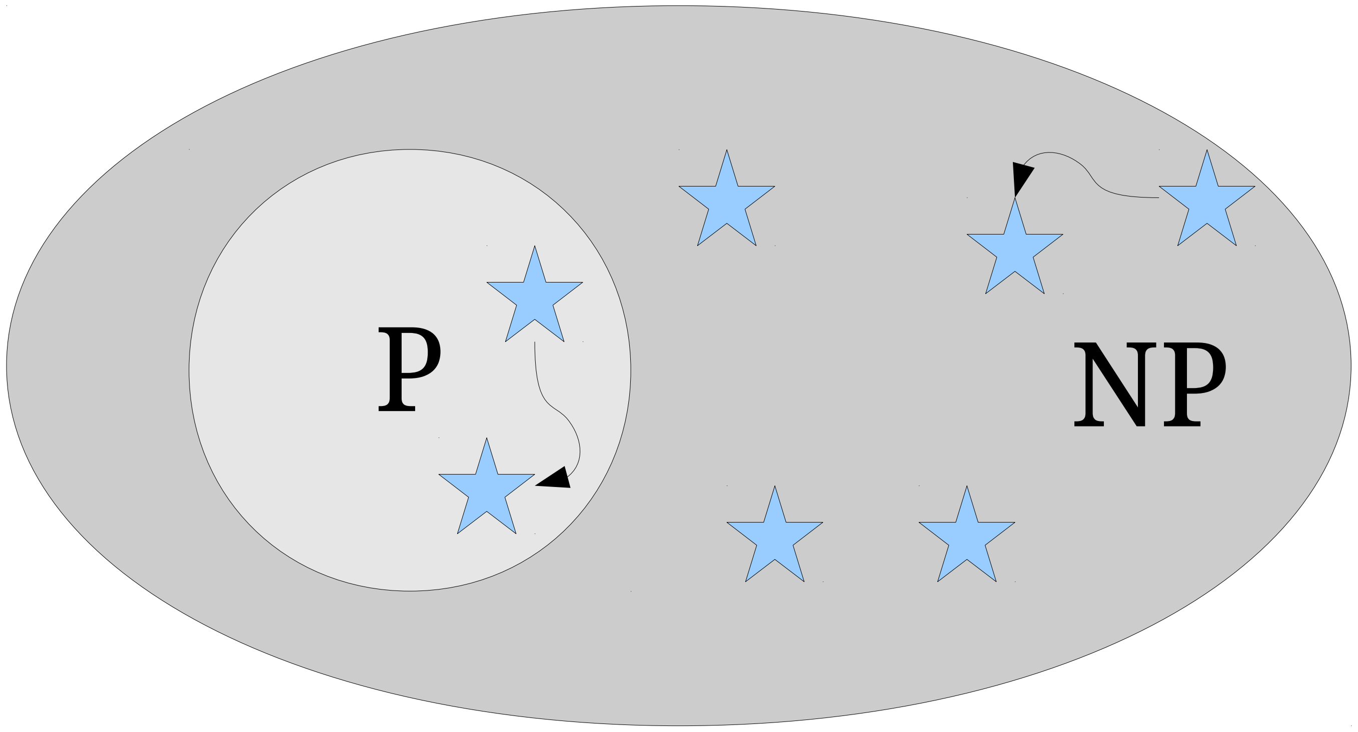
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_P B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



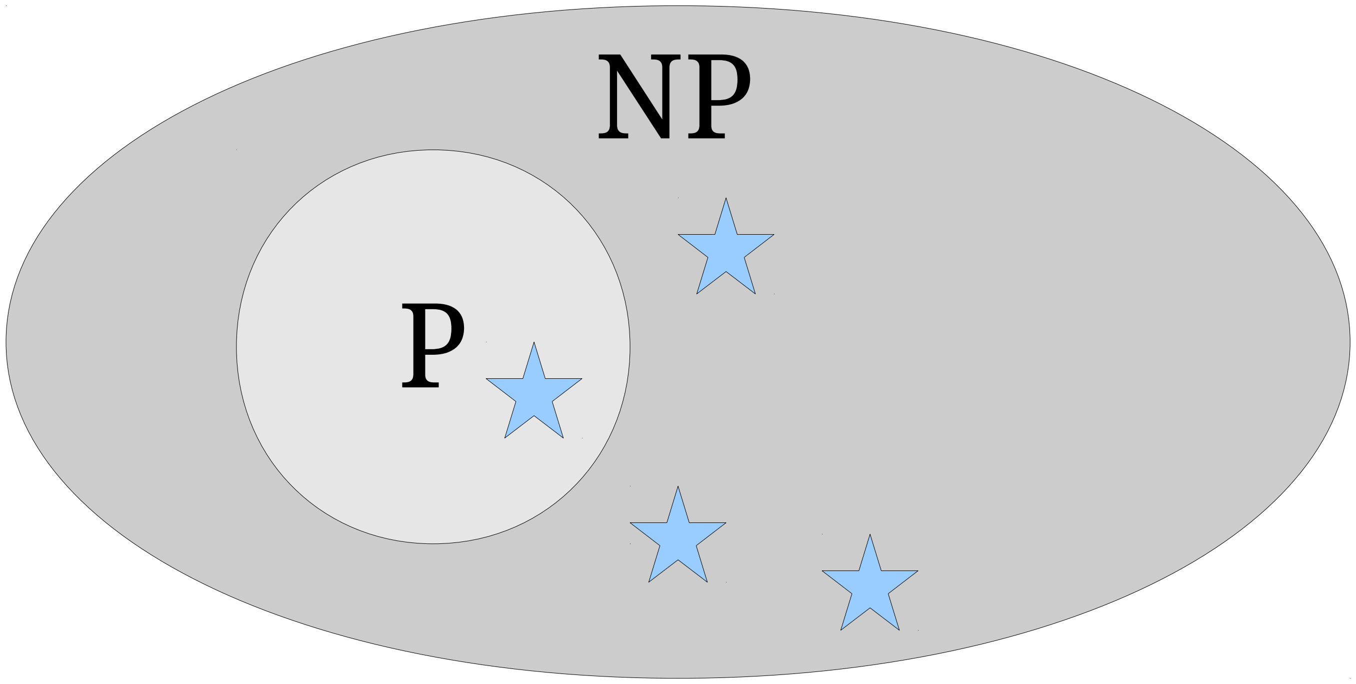
Polynomial-Time Reductions

- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_P B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



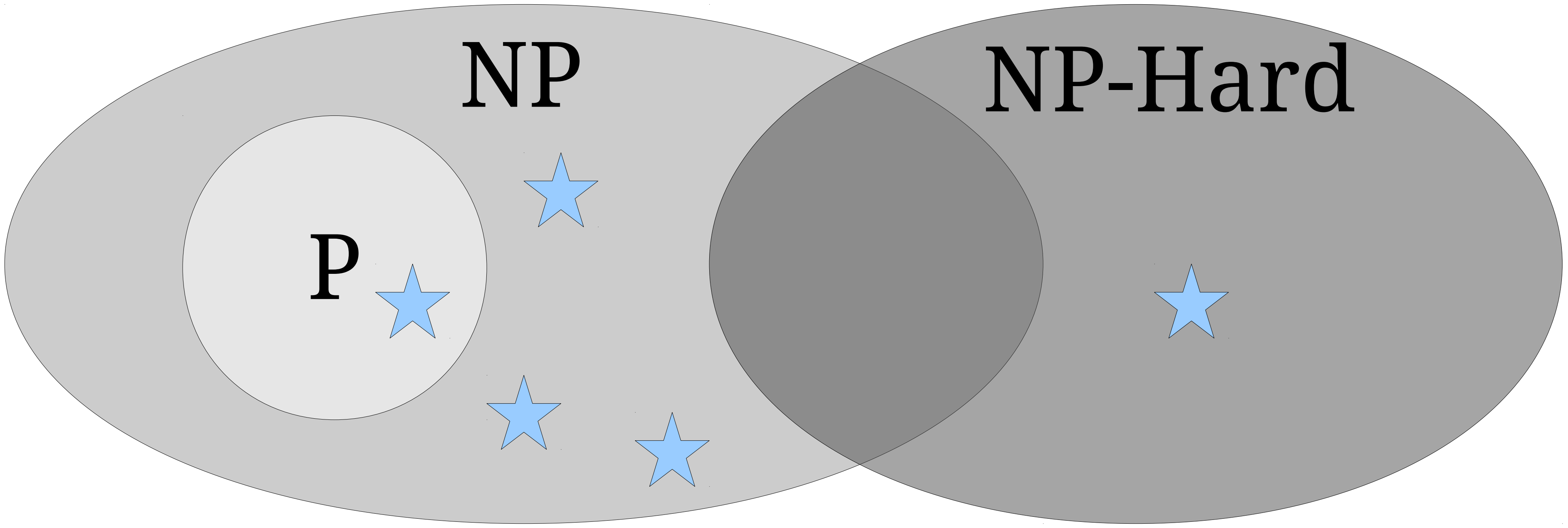
NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in \mathbf{NP}$, we have $A \leq_P L$.



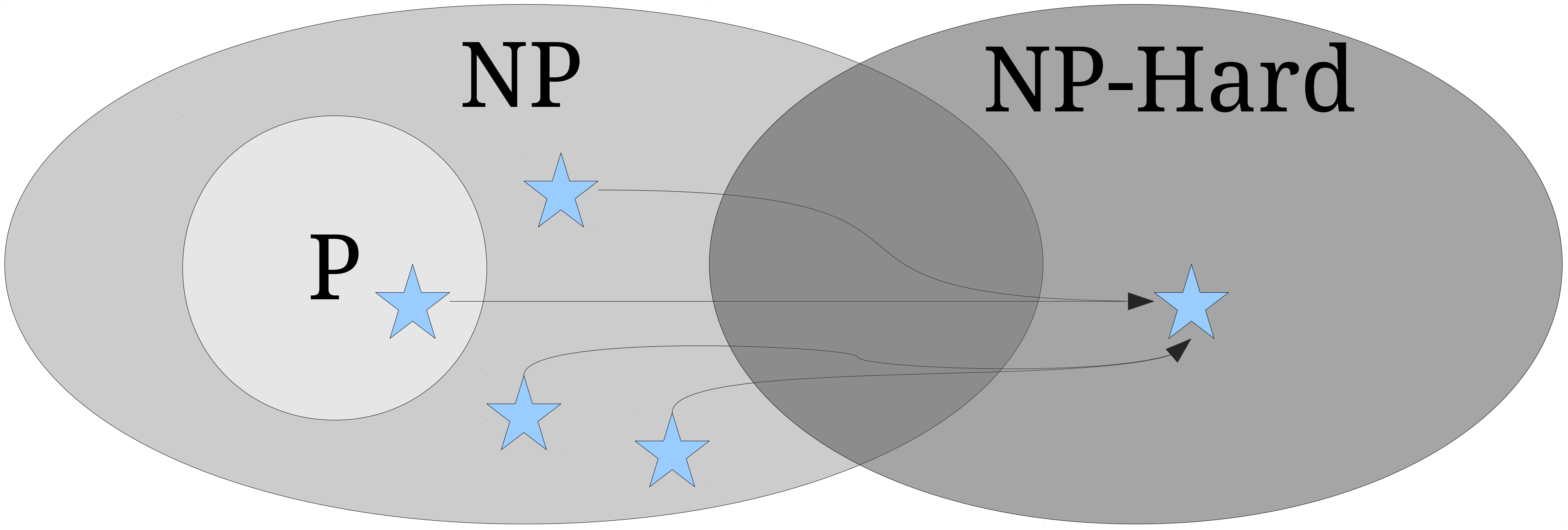
NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in \mathbf{NP}$, we have $A \leq_P L$.



NP-Hardness

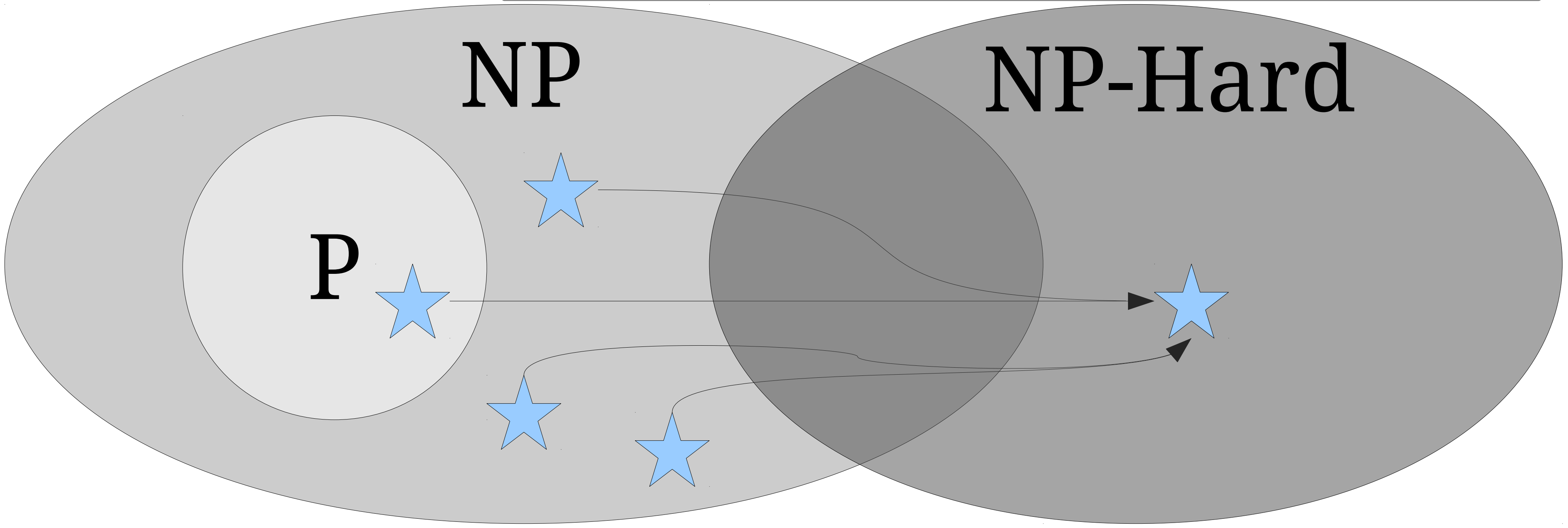
- A language L is called ***NP-hard*** if for *every* $A \in \mathbf{NP}$, we have $A \leq_P L$.



NP-Hardness

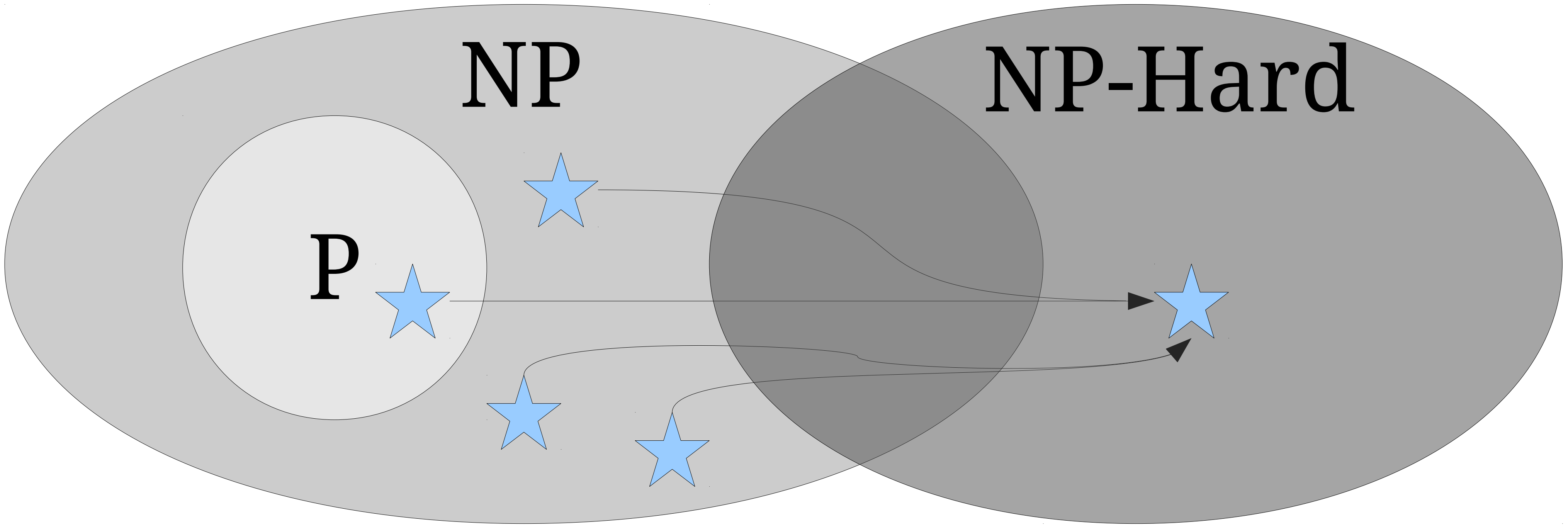
- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_P L$.

Intuitively: L has to be at least as hard as every problem in **NP**, since an algorithm for L can be used to decide all problems in **NP**.



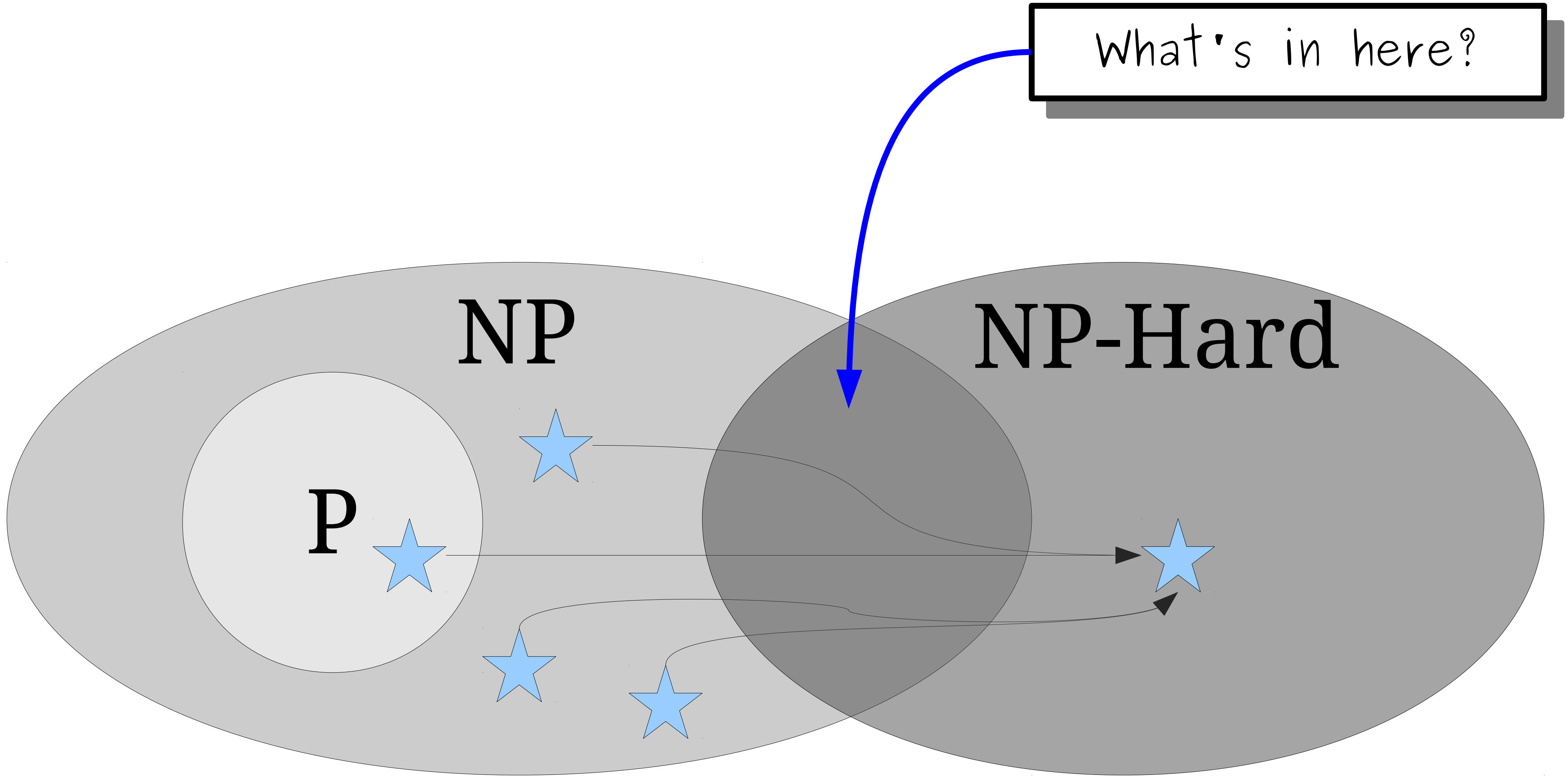
NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_{\mathrm{P}} L$.



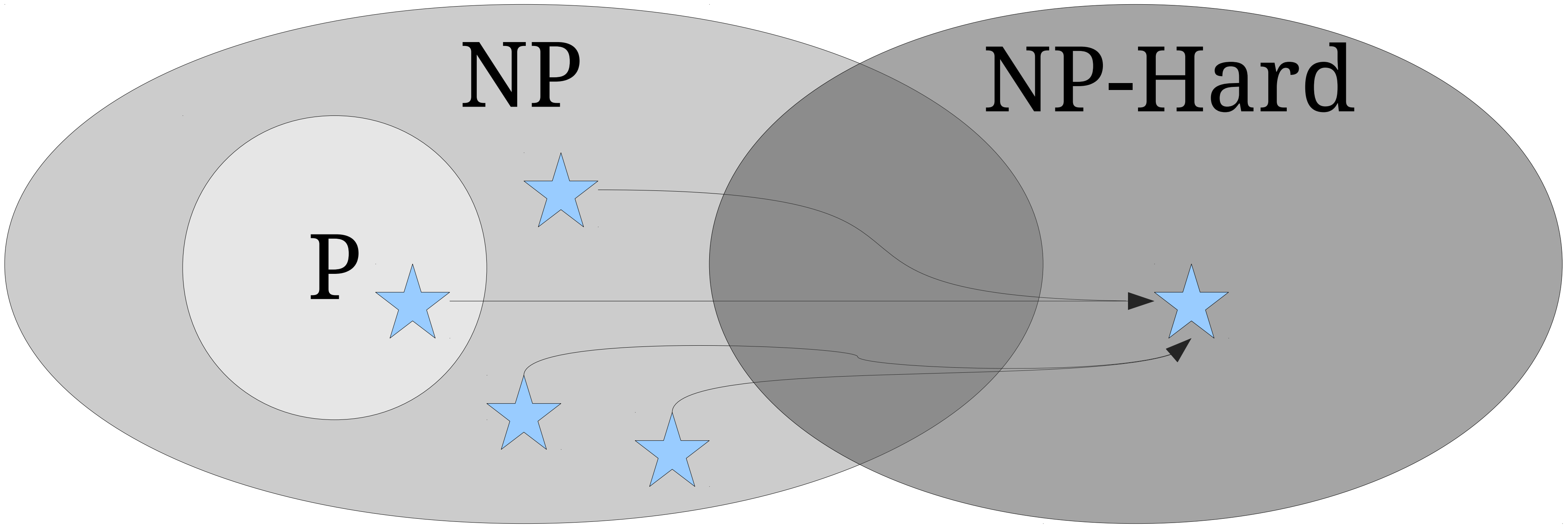
NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_P L$.



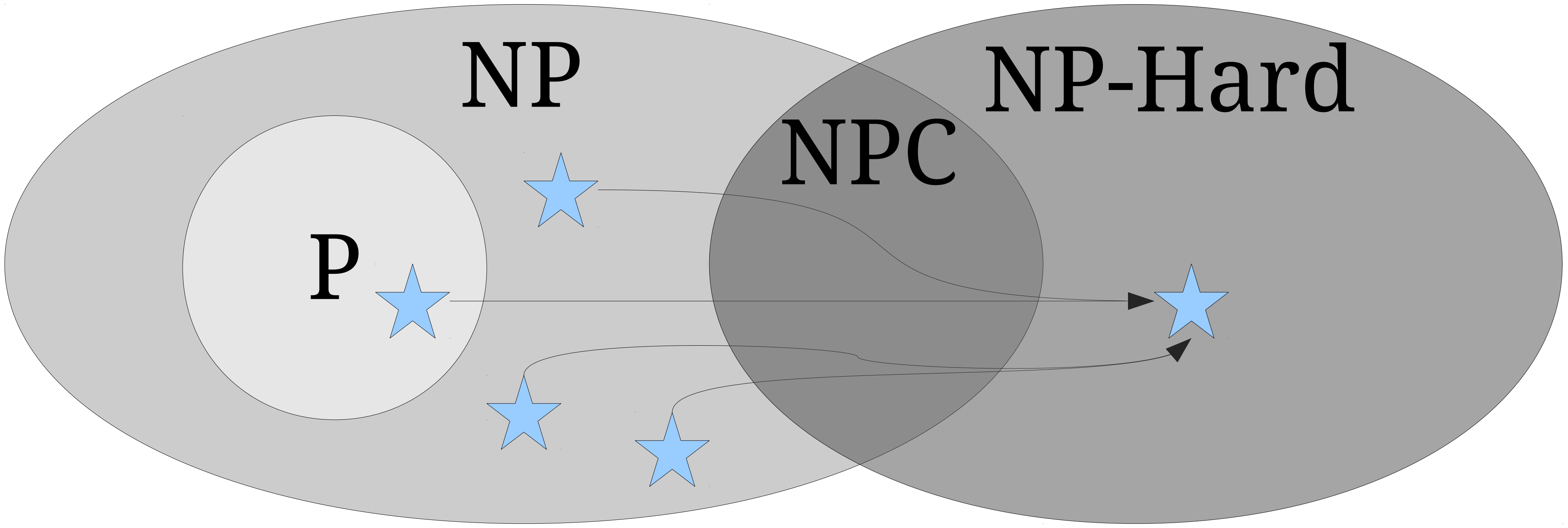
NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_{\mathrm{P}} L$.
- A language in L is called ***NP-complete*** if L is **NP**-hard and $L \in$ **NP**.
- The class ***NPC*** is the set of **NP**-complete problems.



NP-Hardness

- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_{\mathrm{P}} L$.
- A language in L is called ***NP-complete*** if L is **NP**-hard and $L \in$ **NP**.
- The class ***NPC*** is the set of **NP**-complete problems.

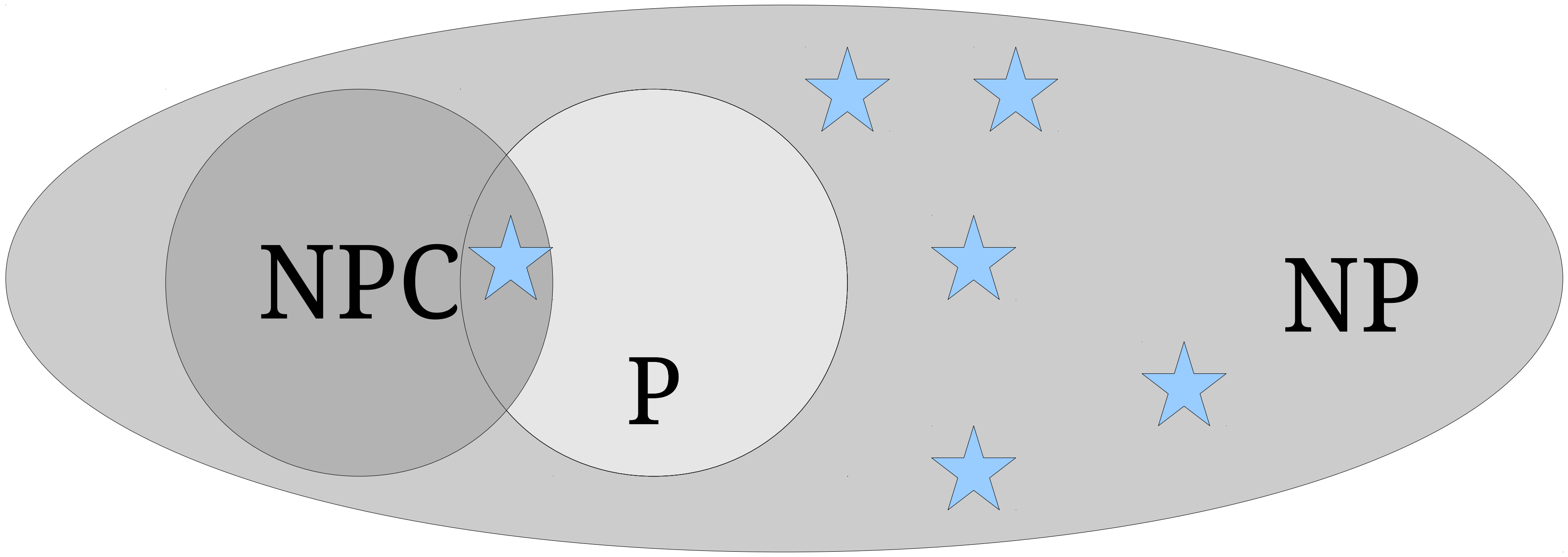


The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P** = **NP**.

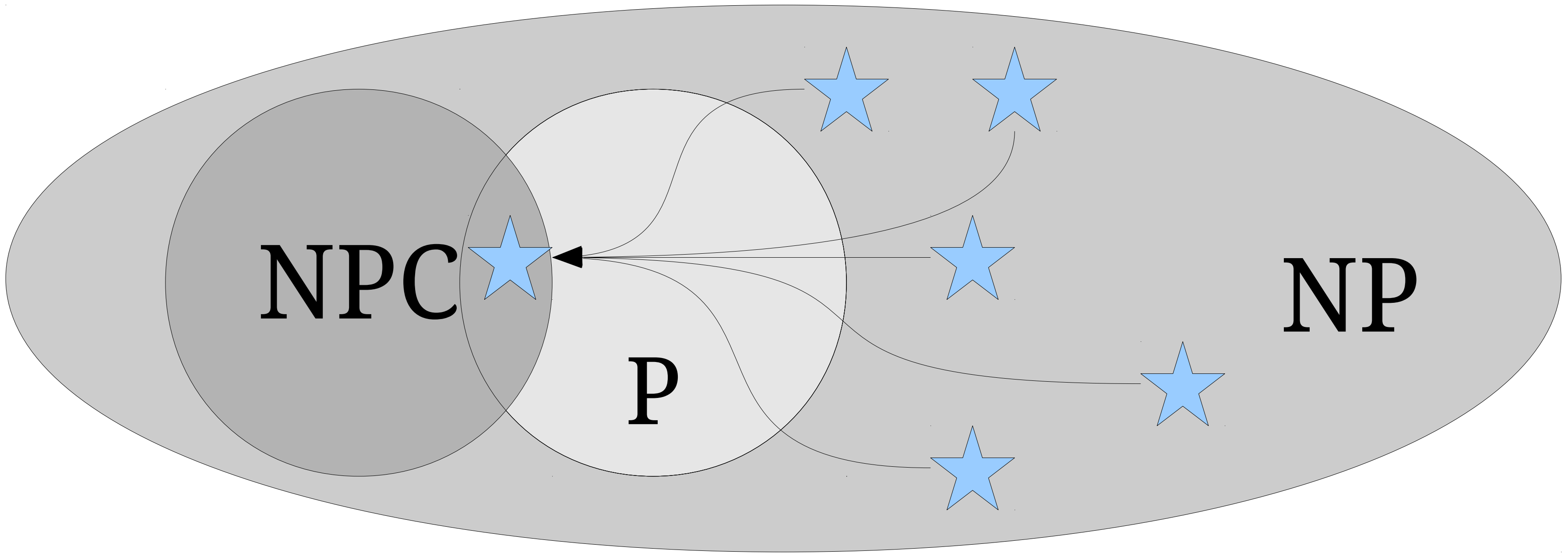
The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P = NP**.



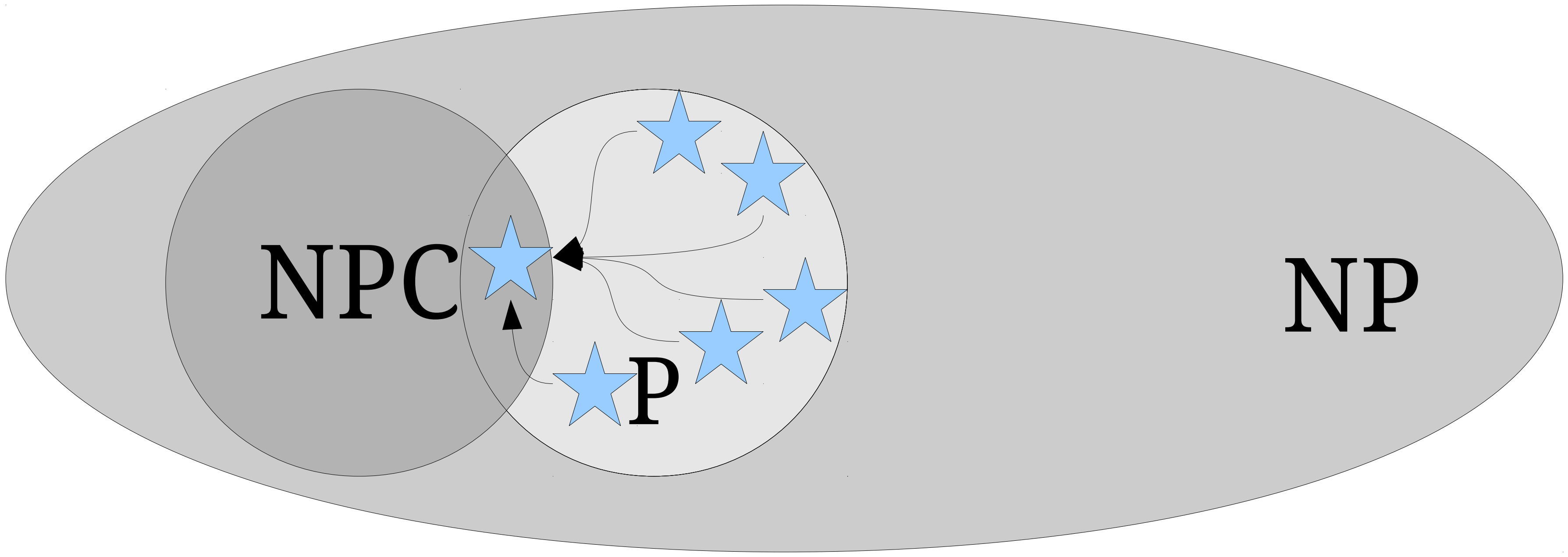
The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P** = **NP**.



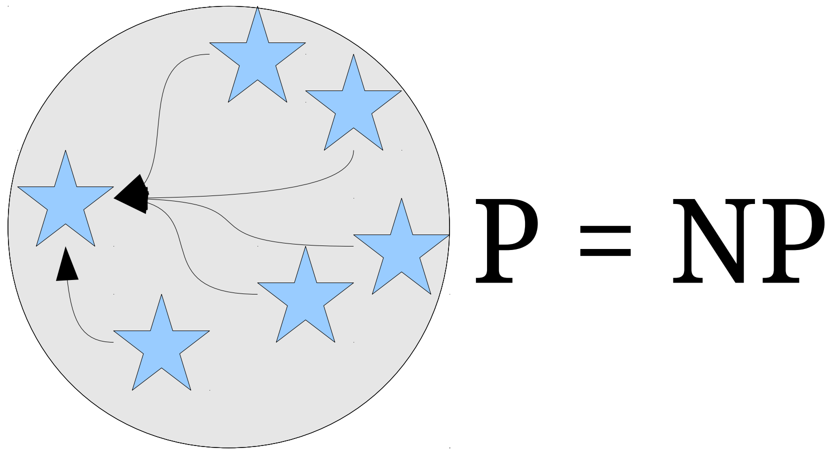
The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P** = **NP**.



The Tantalizing Truth

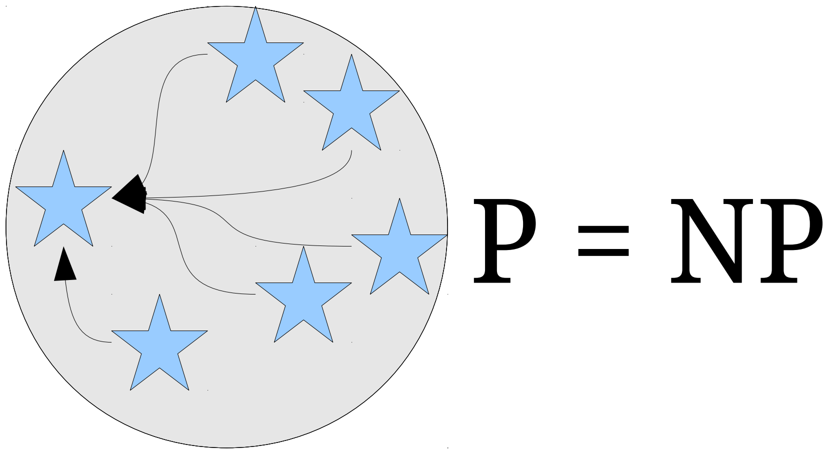
Theorem: If *any* **NP**-complete language is in **P**, then **P = NP**.



The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P** = **NP**.

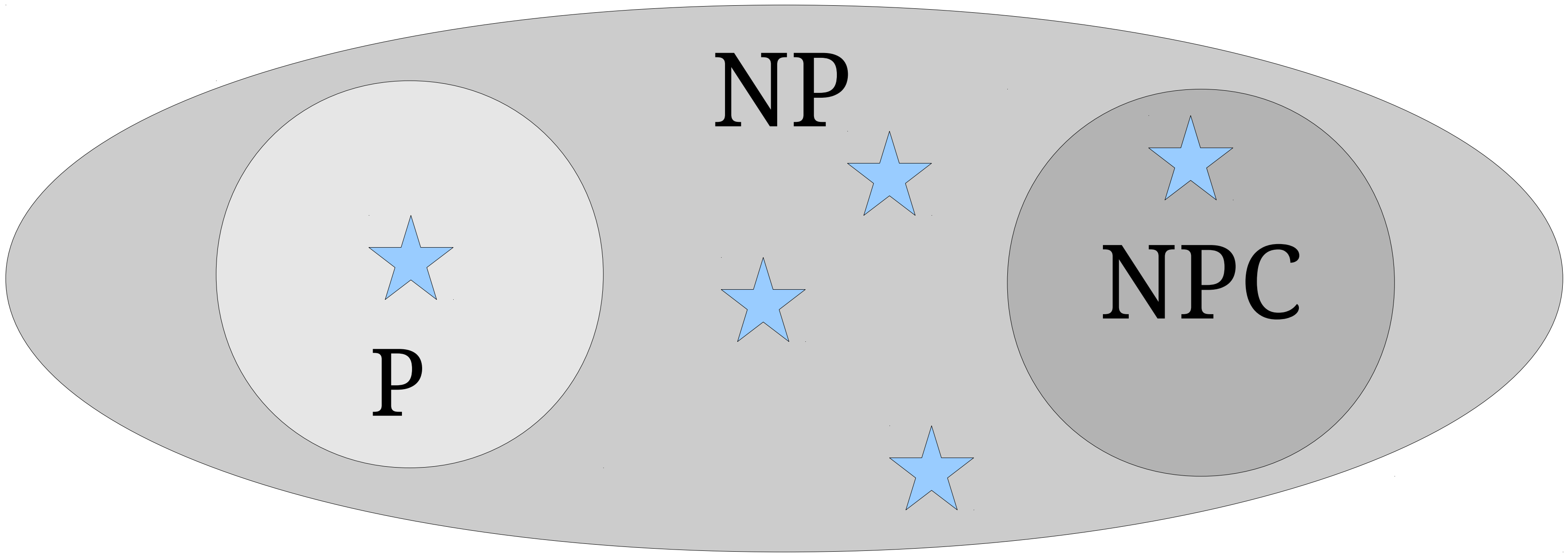
Proof: Suppose that L is **NP**-complete and $L \in$ **P**. Now consider any arbitrary **NP** problem A. Since L is **NP**-complete, we know that $A \leq_p L$. Since $L \in$ **P** and $A \leq_p L$, we see that $A \in$ **P**. Since our choice of A was arbitrary, this means that **NP** ⊆ **P**, so **P** = **NP**. ■



The Tantalizing Truth

Theorem: If *any* **NP**-complete language is not in **P**, then **P** ≠ **NP**.

Proof: Suppose that L is an **NP**-complete language not in **P**. Since L is **NP**-complete, we know that $L \in$ **NP**. Therefore, we know that $L \in$ **NP** and $L \notin$ **P**, so **P** ≠ **NP**. ■



How do we even know NP-complete problems exist in the first place?

Satisfiability

- A propositional logic formula φ is called ***satisfiable*** if there is some assignment to its variables that makes it evaluate to true.
  - $p \wedge q$ is satisfiable.
  - $p \wedge \neg p$ is unsatisfiable.
  - $p \to (q \wedge \neg q)$ is satisfiable.
- An assignment of true and false to the variables of φ that makes it evaluate to true is called a ***satisfying assignment***.

SAT

- The ***boolean satisfiability problem*** (***SAT***) is the following:

Given a propositional logic formula φ, is φ satisfiable?

- Formally:

SAT **= { ⟨φ⟩ | φ is a satisfiable PL formula }**

Theorem (Cook-Levin): SAT is **NP**-complete.

Proof: Read Sipser or take CS154!

New Stuff!

A Simpler **NP**-Complete Problem

Literals and Clauses

- A ***literal*** in propositional logic is a variable or its negation:
  - x
  - $\neg y$
  - But not $x \wedge y$.
- A ***clause*** is a many-way OR (*disjunction*) of literals.
  - $(\neg x \vee y \vee \neg z)$
  - (x)
  - But not $x \vee \neg(y \vee z)$

Conjunctive Normal Form

- A propositional logic formula φ is in ***conjunctive normal form*** (***CNF***) if it is the many-way AND (*conjunction*) of clauses.
  - $(x \lor y \lor z) \land (\neg x \lor \neg y) \land (x \lor y \lor z \lor \neg w)$
  - $(x \lor z)$
  - But not $(x \lor (y \land z)) \lor (x \lor y)$
- Only legal operators are ¬, ∨, ∧.
- No nesting allowed.

The Structure of CNF

$$\underbrace{(\ x \lor y \lor \neg z\ )}\land\underbrace{(\ \neg x \lor \neg y \lor z\ )}\land\underbrace{(\ \neg x \lor y \lor \neg z\ )}$$

For this formula to be satisfiable, each clause must have <u>at least</u> one true literal in it.

The Structure of CNF

$$\underbrace{(x \lor y \lor \boxed{\neg z})} \land \underbrace{(\neg x \lor \boxed{\neg y} \lor z)} \land \underbrace{(\neg x \lor y \lor \boxed{\neg z})}$$

We should pick at least one true literal from each clause...

The Structure of CNF

$$\underbrace{(\ x \lor y \lor \neg z\ )}_{} \land \underbrace{(\ \neg x \lor \neg y \lor z\ )}_{} \land \underbrace{(\ \neg x \lor y \lor \neg z\ )}_{}$$

... but never choose a literal and its negation

3-CNF

- A propositional formula is in ***3-CNF*** if
  - it is in CNF, and
  - every clause has *exactly* three literals.
- For example:
  - $(x \lor y \lor z) \land (\neg x \lor \neg y \lor z)$
  - $(x \lor x \lor x) \land (y \lor \neg y \lor \neg x) \land (x \lor y \lor \neg y)$
  - but not $(x \lor y \lor z \lor w) \land (x \lor y)$
- The language ***3SAT*** is defined as follows:

3SAT = { ⟨φ⟩ | φ is a satisfiable 3-CNF formula }

***Theorem**:* 3SAT is **NP**-Complete

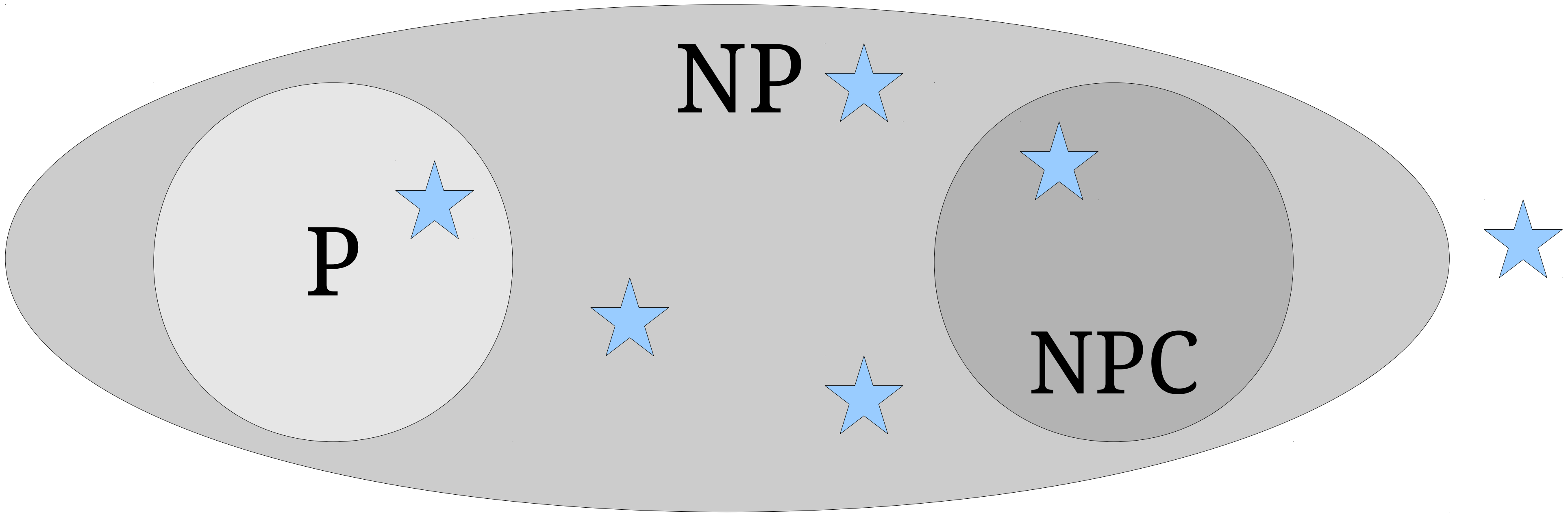
Finding Additional **NP**-Complete Problems

NP-Completeness

Theorem: Let A and B be languages. If $A \leq_P B$ and A is **NP**-hard, then B is **NP**-hard.

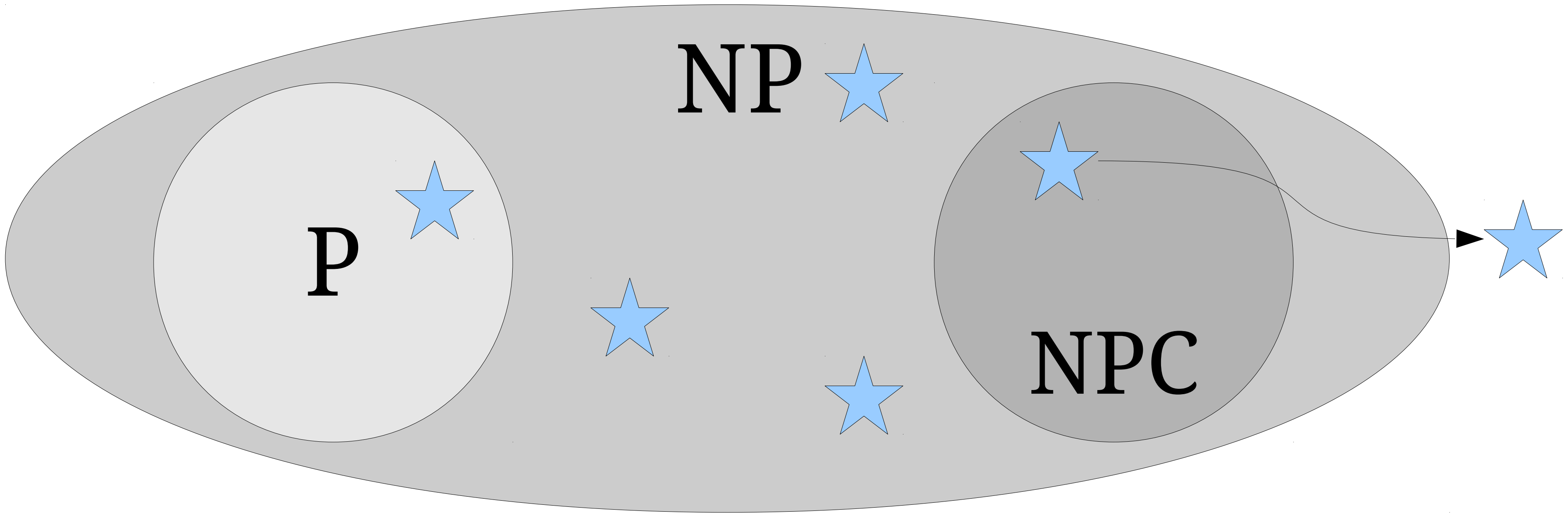
NP-Completeness

Theorem: Let A and B be languages. If $A \leq_{\mathrm{P}} B$ and A is **NP**-hard, then B is **NP**-hard.



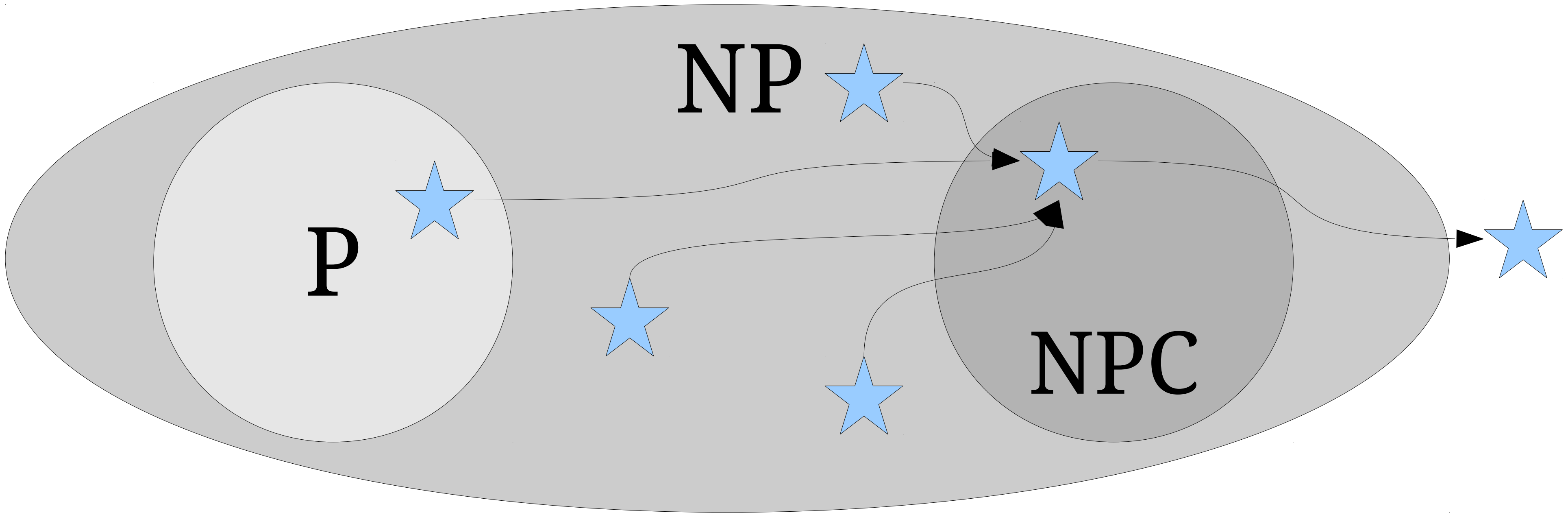
NP-Completeness

Theorem: Let A and B be languages. If $A \leq_{\text{P}} B$ and A is **NP**-hard, then B is **NP**-hard.



NP-Completeness

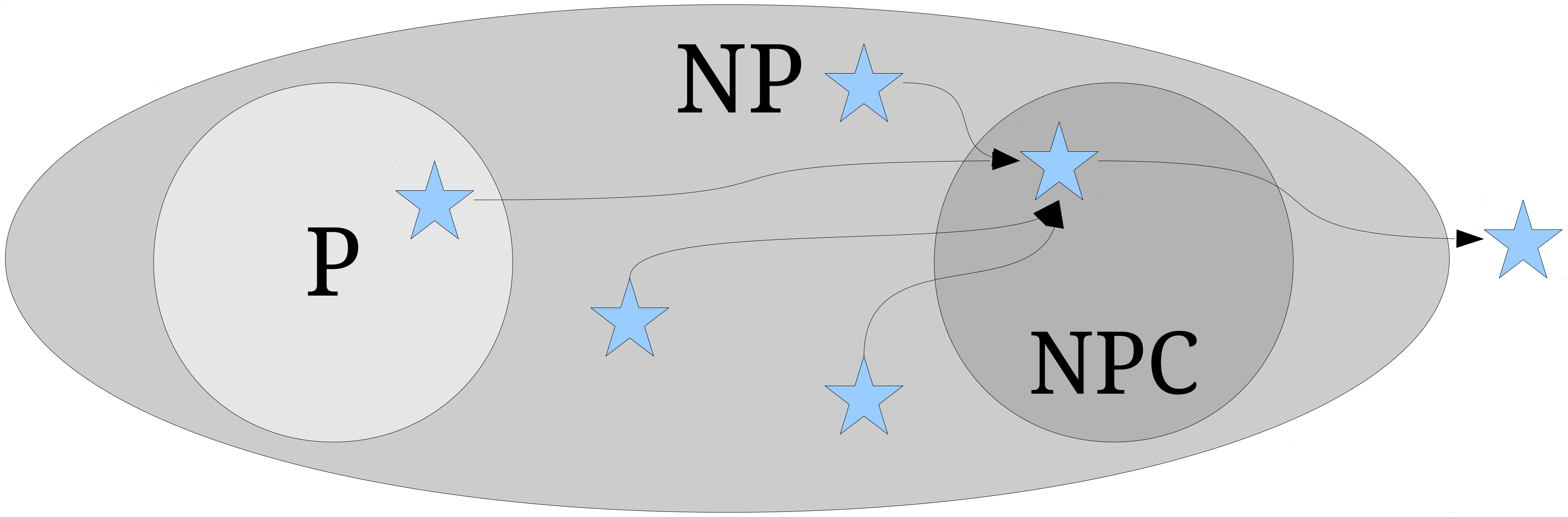
Theorem: Let A and B be languages. If $A \leq_P B$ and A is **NP**-hard, then B is **NP**-hard.



NP-Completeness

Theorem: Let A and B be languages. If $A \leq_P B$ and A is **NP**-hard, then B is **NP**-hard.

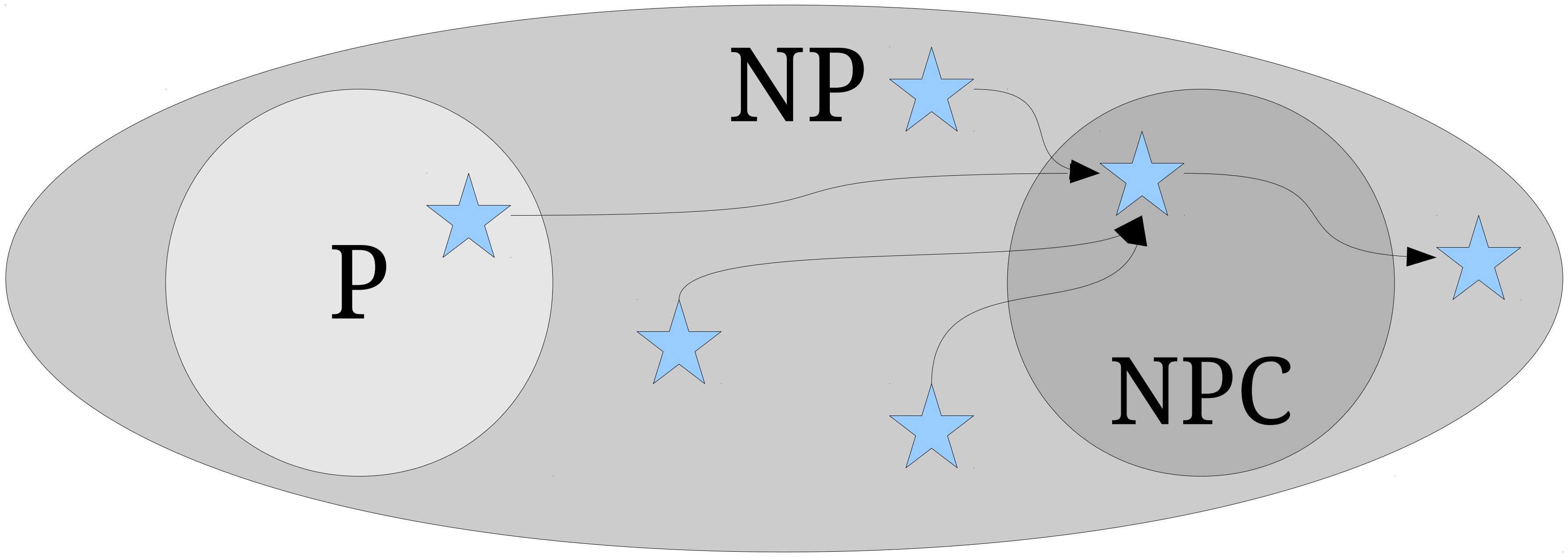
Theorem: Let A and B be languages where $A \in$ **NPC** and $B \in$ **NP**. If $A \leq_P B$, then $B \in$ **NPC**.



NP-Completeness

Theorem: Let A and B be languages. If $A \leq_P B$ and A is **NP**-hard, then B is **NP**-hard.

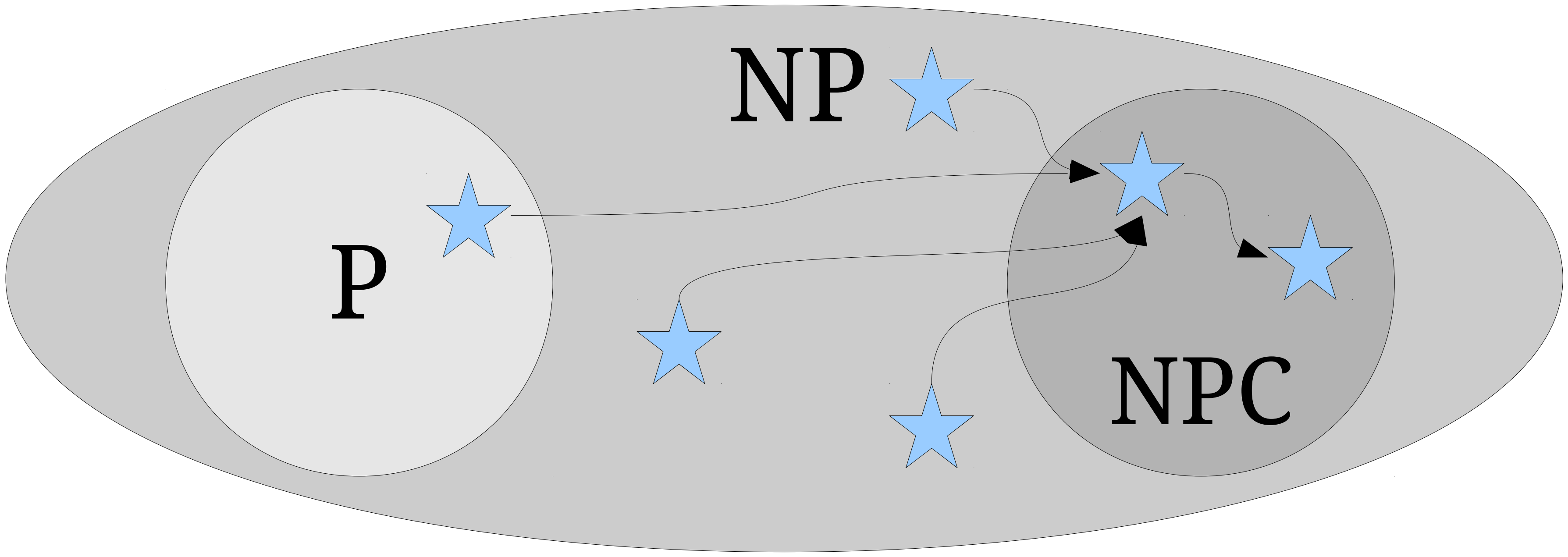
Theorem: Let A and B be languages where $A \in$ **NPC** and $B \in$ **NP**. If $A \leq_P B$, then $B \in$ **NPC**.



NP-Completeness

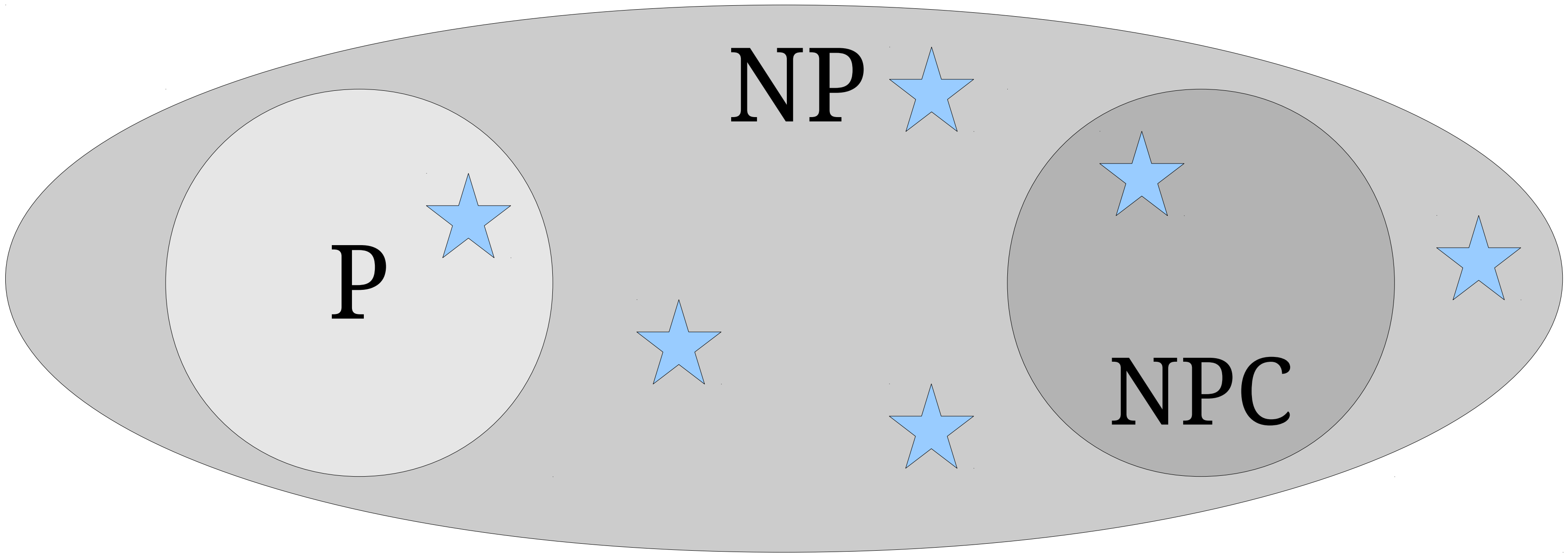
Theorem: Let A and B be languages. If $A \leq_P B$ and A is **NP**-hard, then B is **NP**-hard.

Theorem: Let A and B be languages where $A \in$ **NPC** and $B \in$ **NP**. If $A \leq_P B$, then $B \in$ **NPC**.



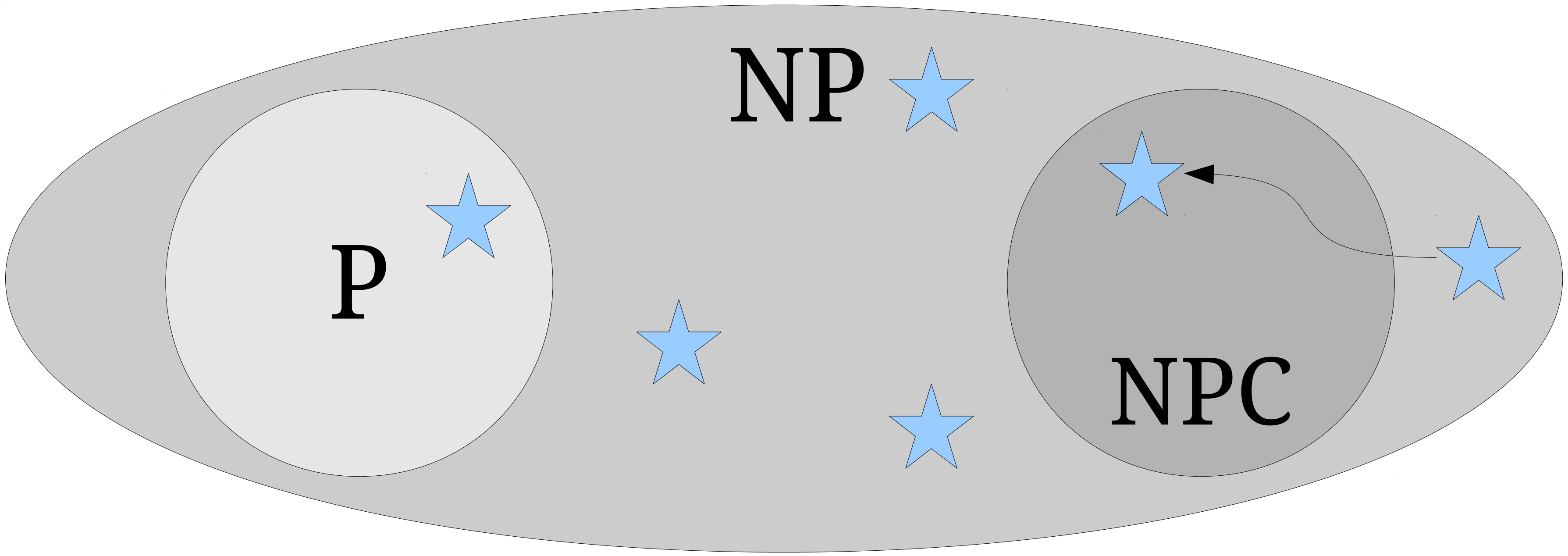
Be Careful!

- To prove that some language L is **NP**-complete, show that $L \in$ **NP**, then reduce some known **NP**-complete problem to L.
- **Do not** reduce L to a known **NP**-complete problem.
  - We already knew you could do this; *every* **NP** problem is reducible to any **NP**-complete problem!



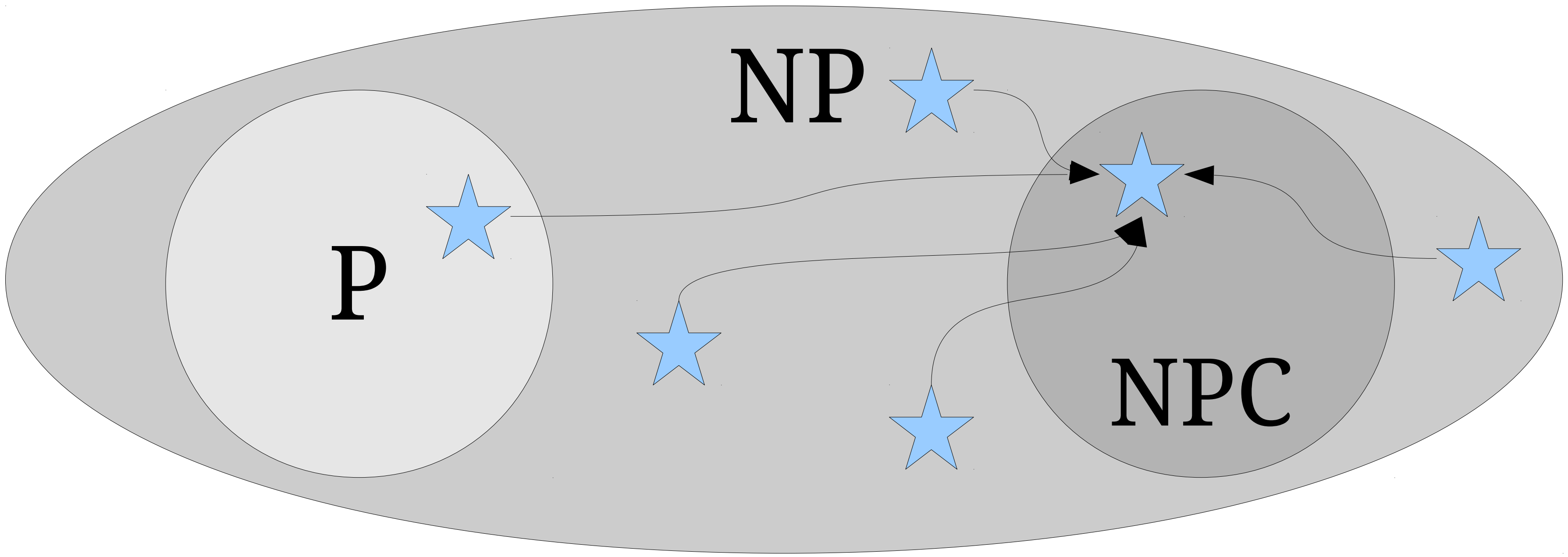
Be Careful!

- To prove that some language L is **NP**-complete, show that $L \in$ **NP**, then reduce some known **NP**-complete problem to L.
- **Do not** reduce L to a known **NP**-complete problem.
    - We already knew you could do this; *every* **NP** problem is reducible to any **NP**-complete problem!



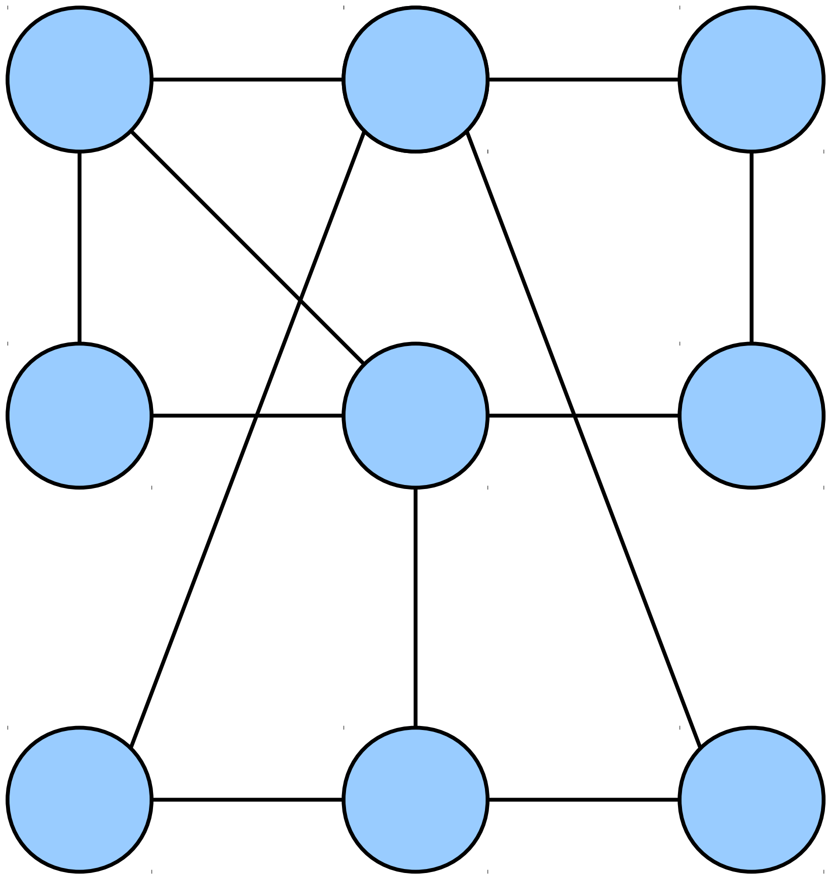
Be Careful!

- To prove that some language L is **NP**-complete, show that $L \in$ **NP**, then reduce some known **NP**-complete problem to L.
- **Do not** reduce L to a known **NP**-complete problem.
    - We already knew you could do this; *every* **NP** problem is reducible to any **NP**-complete problem!

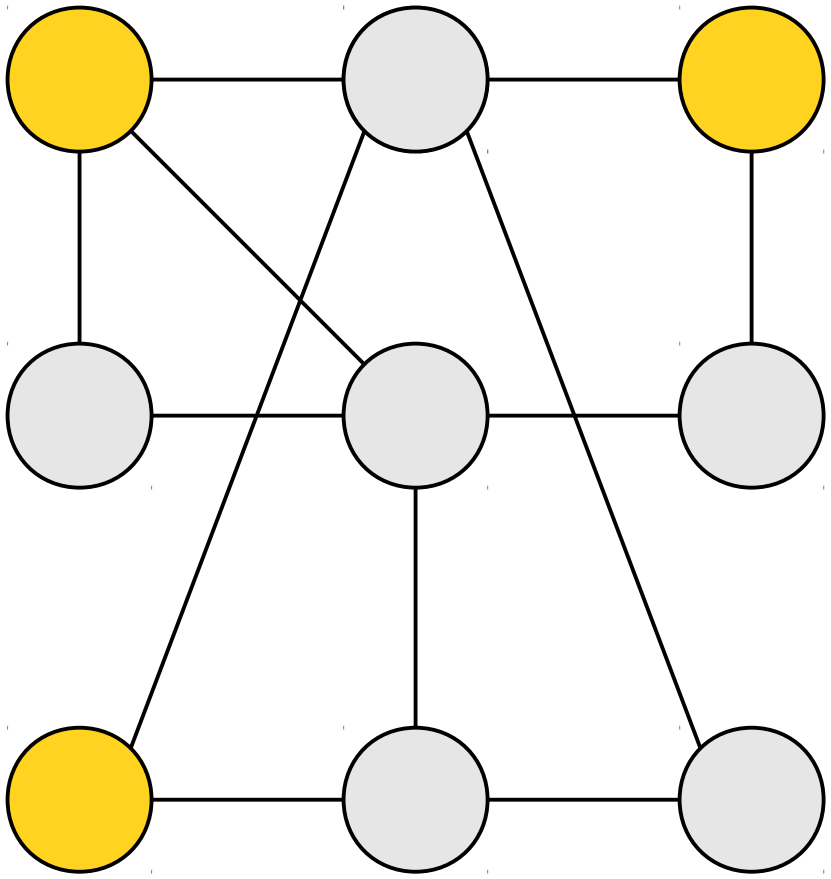


So what other problems are **NP**-complete?

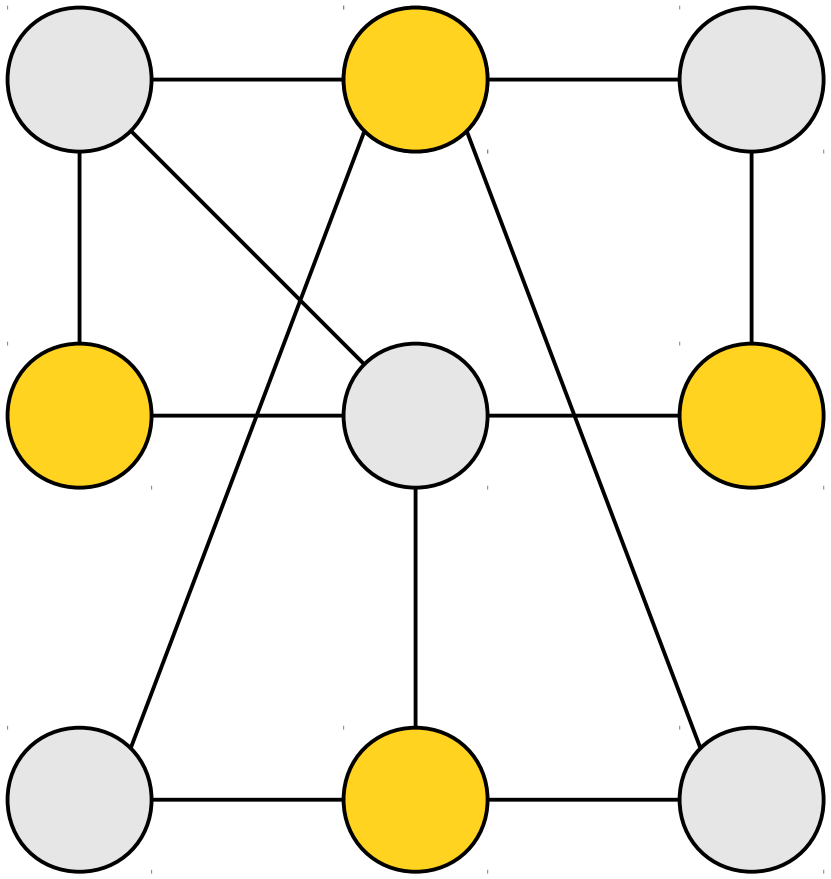
An ***independent set*** in an undirected graph
is a set of nodes that have no edges between them.



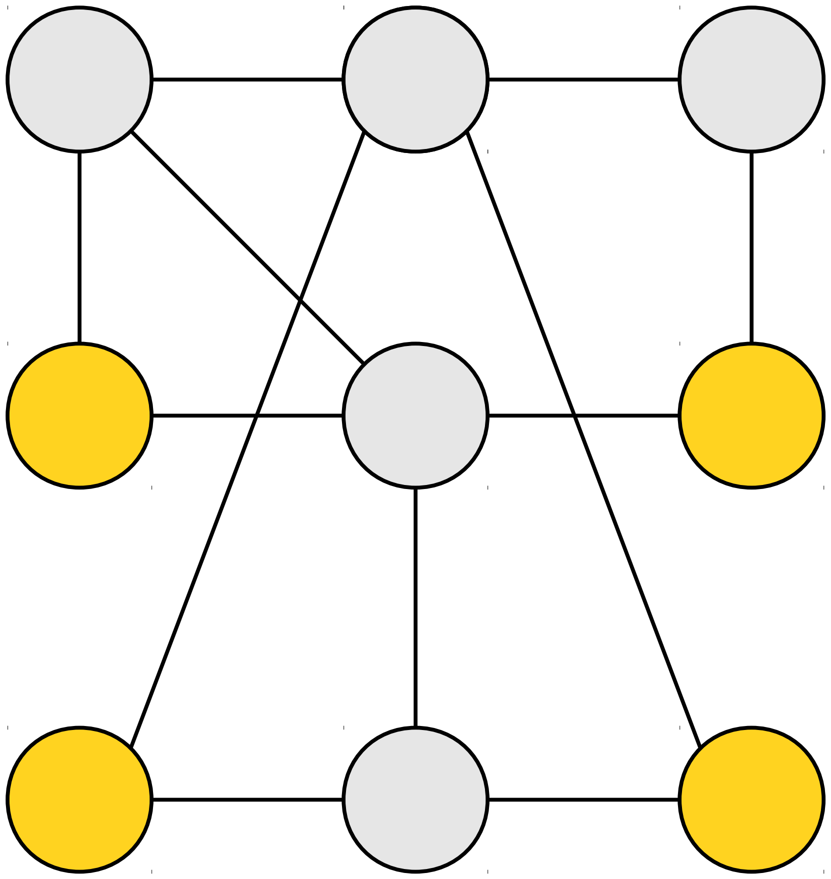
An ***independent set*** in an undirected graph is a set of nodes that have no edges between them.



An ***independent set*** in an undirected graph
is a set of nodes that have no edges between them.



An ***independent set*** in an undirected graph
is a set of nodes that have no edges between them.



An ***independent set*** in an undirected graph
is a set of nodes that have no edges between them.

The Independent Set Problem

- Given an undirected graph G and a natural number n, the ***independent set problem*** is

Does *G* contain an independent set of size at least *n*?

- As a formal language:

INDSET = { ⟨*G*, *n*⟩ | *G* is an undirected graph with an independent set of size at least *n* }

$INDSET \in \mathbf{NP}$

- The independent set problem is in **NP**.
- Here is a polynomial-time verifier that checks whether S is an n-element independent set:

```
V = “On input ⟨G, n, S⟩, where G is a graph,
        n ∈ ℕ, and S is a set of nodes in G:
            If |S| < n, reject.
            For each edge in G, if both
            endpoints are in S, reject.
            Otherwise, accept.”
```

INDSET ∈ **NPC**

- The *INDSET* problem is **NP**-complete.
- To prove this, we will find a polynomial-time reduction from 3SAT to *INDSET*.
- ***Goal:*** Given a 3CNF formula φ, build a graph *G* and number *n* such that φ is satisfiable iff *G* has an independent set of size *n*.
- How can we accomplish this?

The Structure of 3CNF

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$

The Structure of 3CNF

$$(\underbrace{x \lor y \lor \neg z}_{}) \land (\underbrace{\neg x \lor \neg y \lor z}_{}) \land (\underbrace{\neg x \lor y \lor \neg z}_{})$$

The Structure of 3CNF

$$\underbrace{(x \lor y \lor \neg z)}_{} \land \underbrace{(\neg x \lor \neg y \lor z)}_{} \land \underbrace{(\neg x \lor y \lor \neg z)}_{}$$

$\boxed{\text{Each clause must have \underline{at least} one true literal in it.}}$

The Structure of 3CNF

$$\underbrace{(\ x \lor y \lor \boxed{\neg z}\ )} \land \underbrace{(\ \neg x \lor \boxed{\neg y} \lor z\ )} \land \underbrace{(\ \neg x \lor y \lor \boxed{\neg z}\ )}$$

The Structure of 3CNF

$$\underbrace{(\ x \lor y \lor \neg z\ )}_{} \land \underbrace{(\ \neg x \lor \neg y \lor z\ )}_{} \land \underbrace{(\ \neg x \lor y \lor \neg z\ )}_{}$$

We should pick at least one true literal from each clause

The Structure of 3CNF

$$\underbrace{(\ x \lor \boxed{y} \lor \neg z\ )}\ \land\ \underbrace{(\ \boxed{\neg x} \lor \neg y \lor z\ )}\ \land\ \underbrace{(\ \boxed{\neg x} \lor y \lor \neg z\ )}$$

The Structure of 3CNF

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

The Structure of 3CNF

$$(\underbrace{x \lor y \lor \neg z}) \land (\underbrace{\neg x \lor \neg y \lor z}) \land (\underbrace{\neg x \lor y \lor \neg z})$$

The Structure of 3CNF

$$\underbrace{(x \lor y \lor \boxed{\neg z})} \land \underbrace{(\neg x \lor \neg y \lor \boxed{z})} \land \underbrace{(\neg x \lor y \lor \boxed{\neg z})}$$

The Structure of 3CNF

$$(\underbrace{x \lor y \lor \neg z}) \land (\underbrace{\neg x \lor \neg y \lor z}) \land (\underbrace{\neg x \lor y \lor \neg z})$$

… subject to the constraint that we never choose a literal and its negation

From 3SAT to INDSET

- To convert a 3SAT instance φ to an *INDSET* instance, we need to create a graph *G* and number *n* such that an independent set of size at least *n* in *G*
  - gives us a way to choose which literal in each clause of φ should be true,
  - doesn't simultaneously choose a literal and its negation, and
  - has size polynomially large in the length of the formula φ.

From 3SAT to INDSET

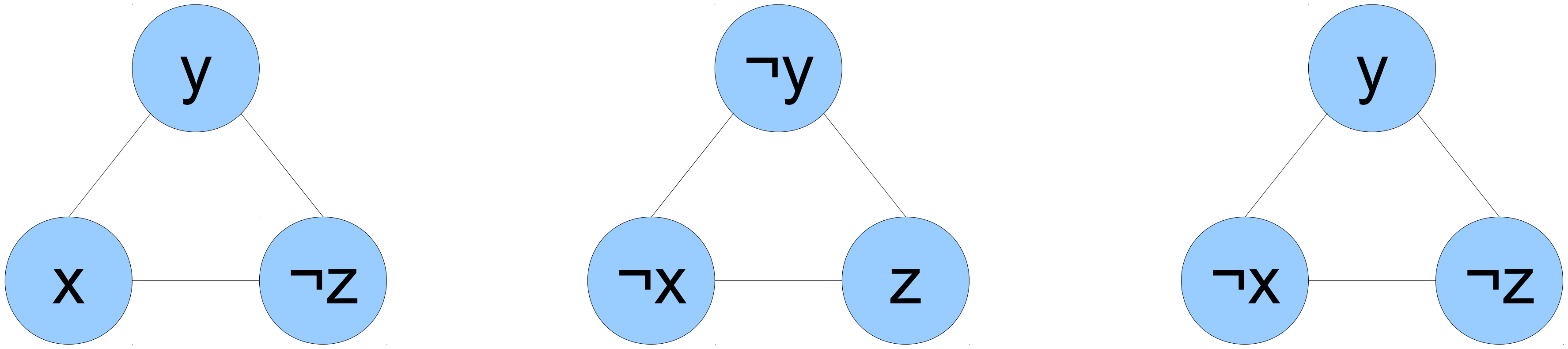
$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$

From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)}_{} \land \underbrace{(\neg x \lor \neg y \lor z)}_{} \land \underbrace{(\neg x \lor y \lor \neg z)}_{}$$

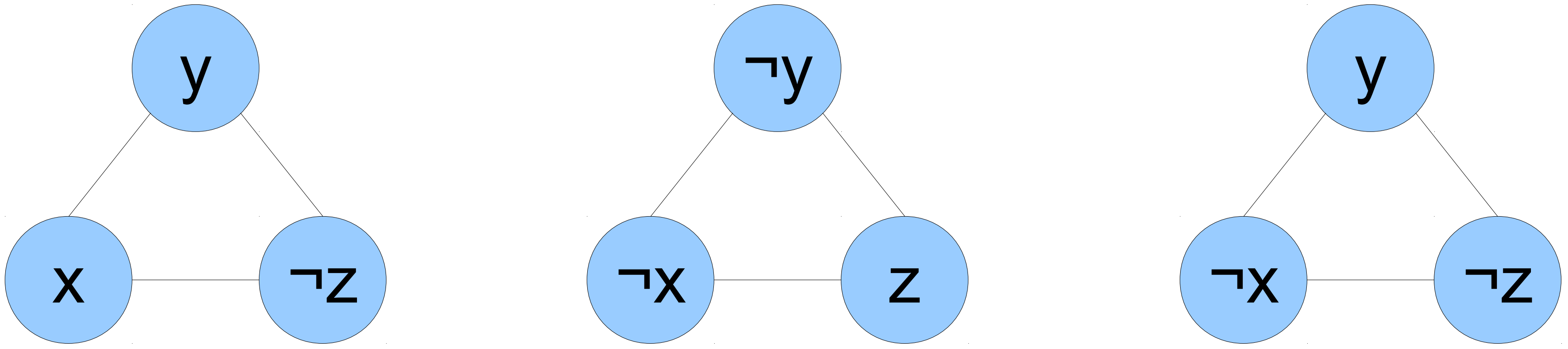
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



From 3SAT to INDSET

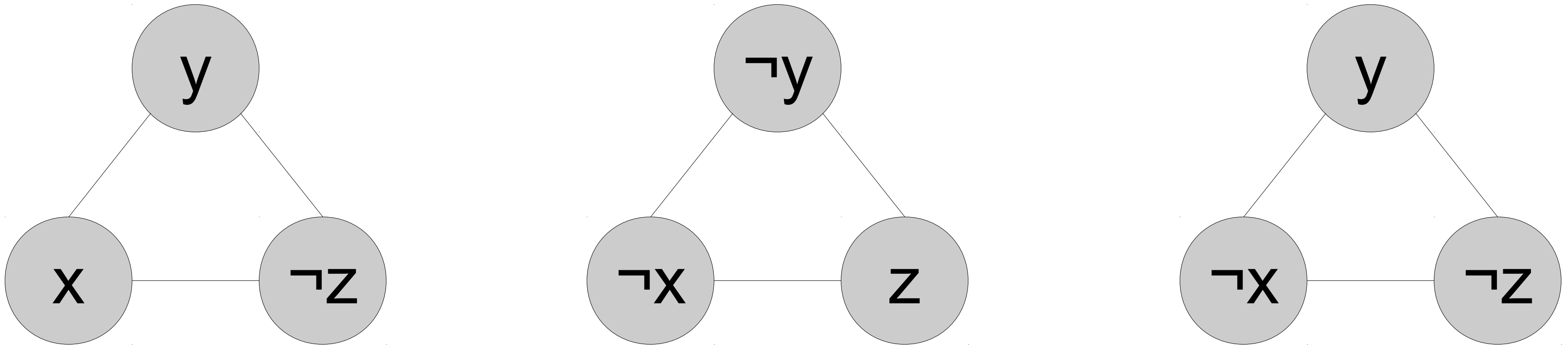
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

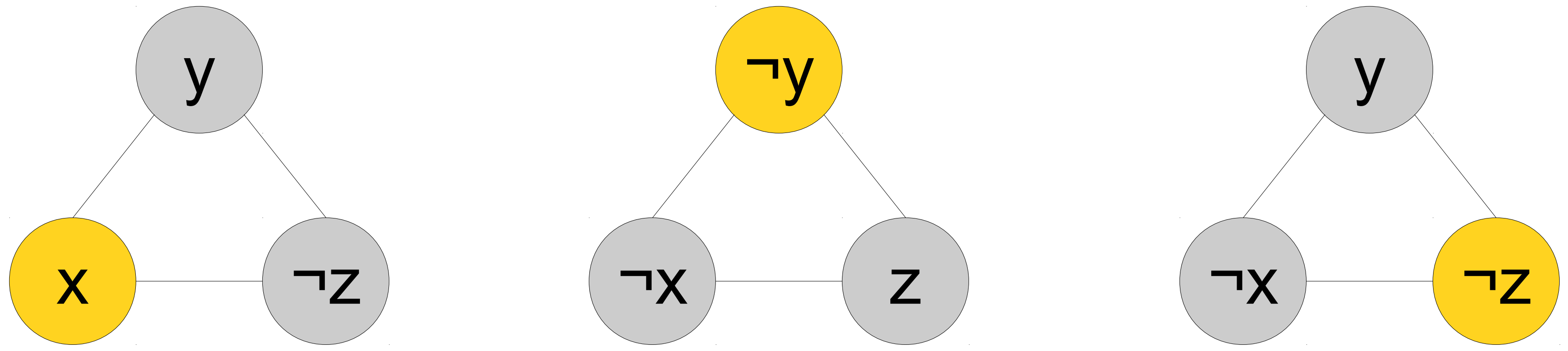
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses exactly one literal from each clause to be true.

From 3SAT to INDSET

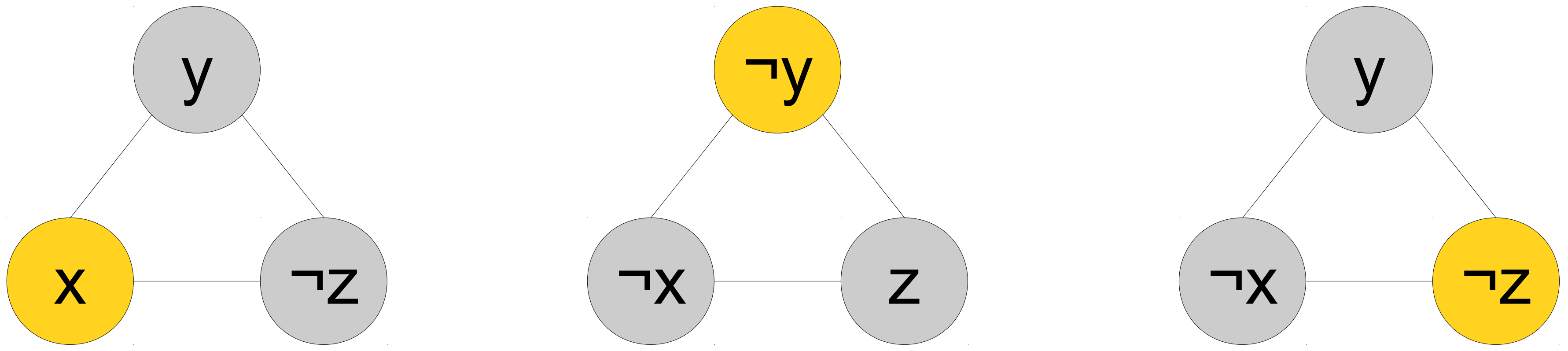
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

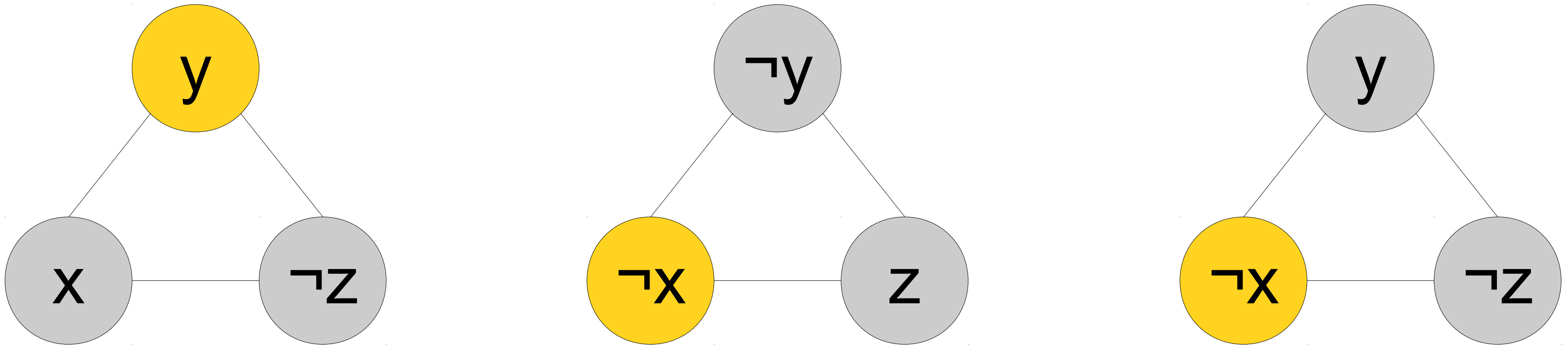
$$\underbrace{(\ x \lor y \lor \neg z\ )}_{} \land \underbrace{(\ \neg x \lor \neg y \lor z\ )}_{} \land \underbrace{(\ \neg x \lor y \lor \neg z\ )}_{}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

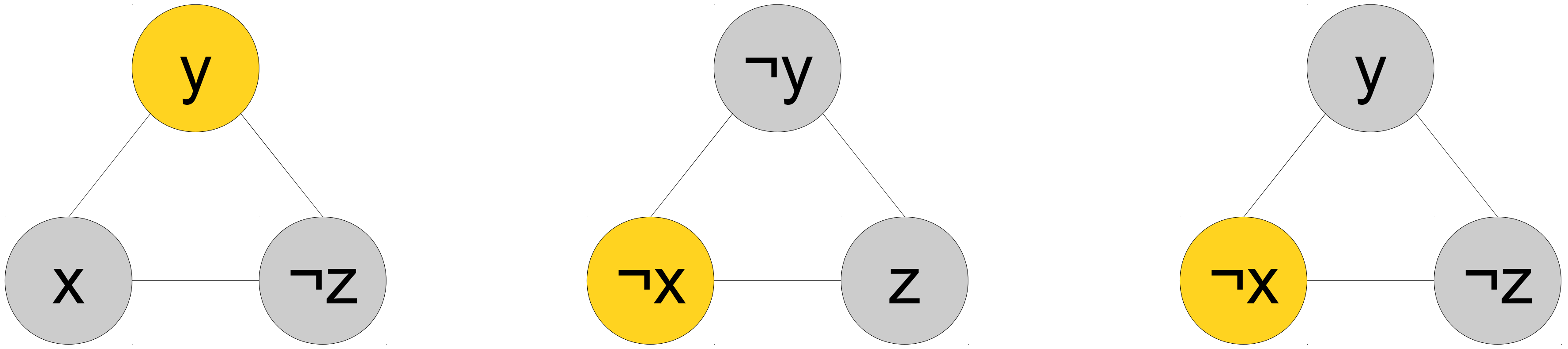
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses exactly one literal from each clause to be true.

From 3SAT to INDSET

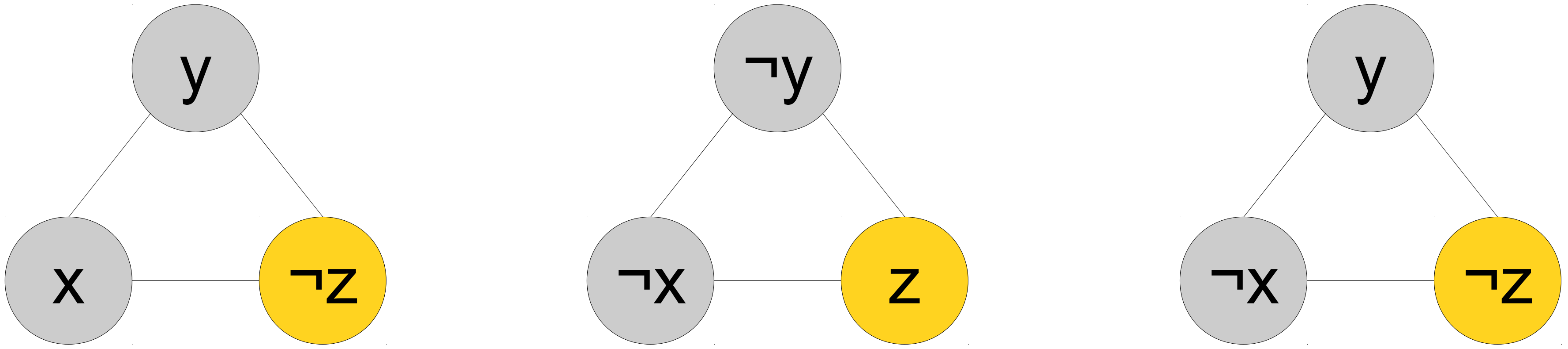
$$\underbrace{(\, x \lor y \lor \neg z \,)}\ \land\ \underbrace{(\, \neg x \lor \neg y \lor z \,)}\ \land\ \underbrace{(\, \neg x \lor y \lor \neg z \,)}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

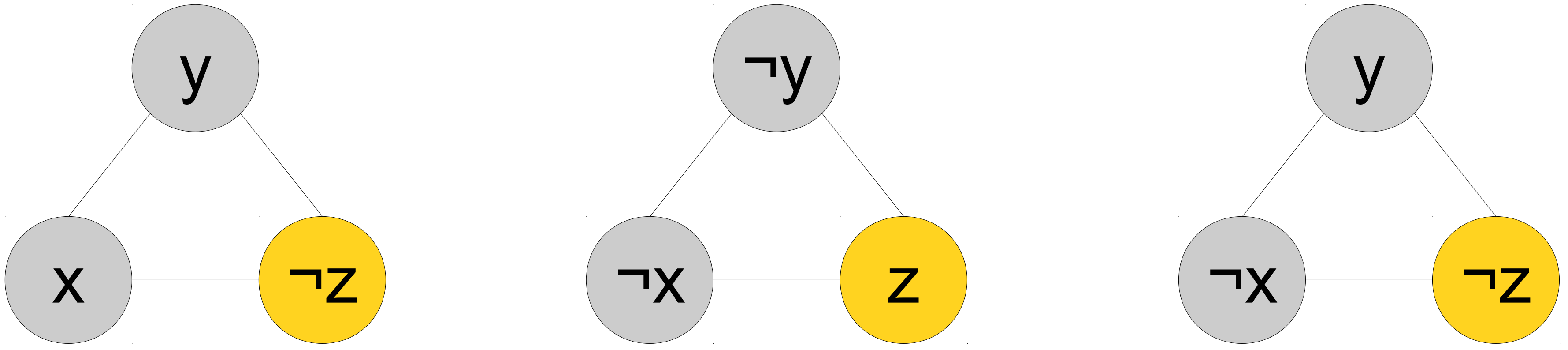
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses exactly one literal from each clause to be true.

From 3SAT to INDSET

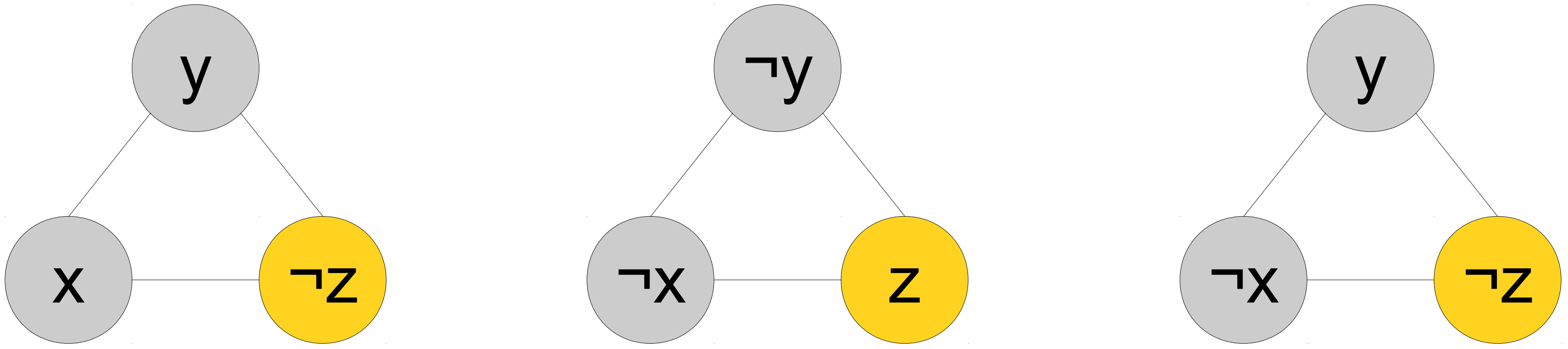
$$\underbrace{(x \lor y \lor \neg z)}\land\underbrace{(\neg x \lor \neg y \lor z)}\land\underbrace{(\neg x \lor y \lor \neg z)}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

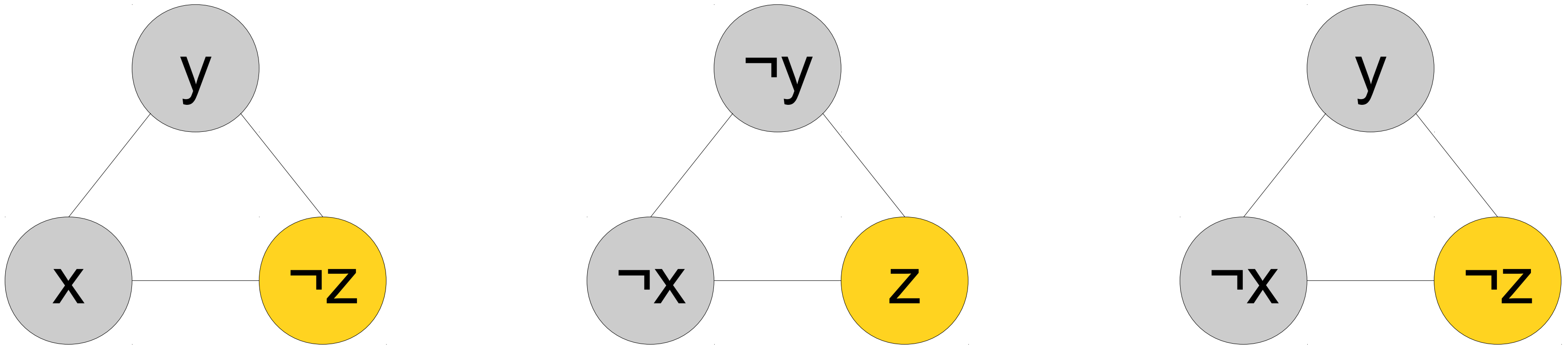
$$\underbrace{(\ x \lor y \lor \neg z\ )}\land\underbrace{(\ \neg x \lor \neg y \lor z\ )}\land\underbrace{(\ \neg x \lor y \lor \neg z\ )}$$



Any independent set in this graph chooses **exactly one** literal from each clause to be true.

From 3SAT to INDSET

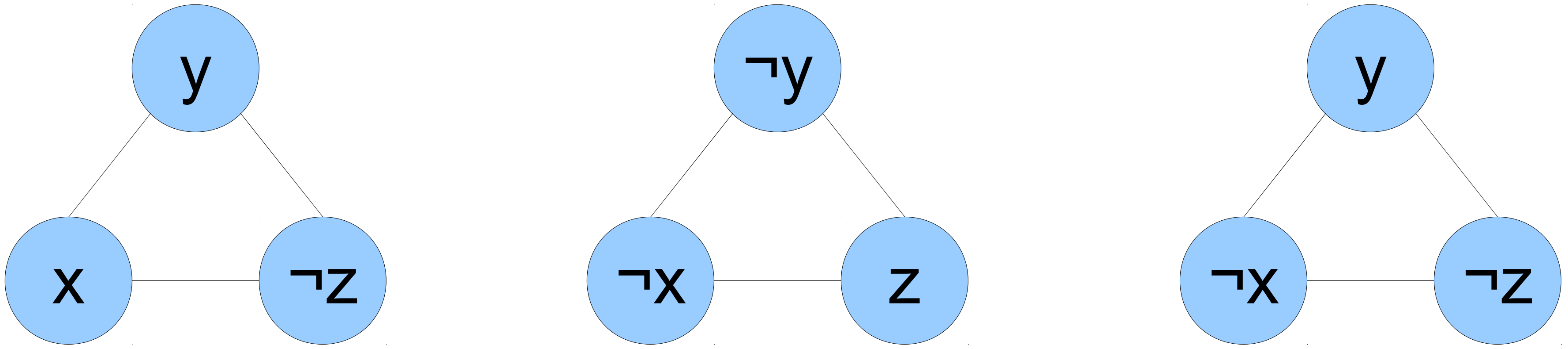
$$\underbrace{(x \lor y \lor \boxed{\neg z})} \land \underbrace{(\neg x \lor \neg y \lor \boxed{z})} \land \underbrace{(\neg x \lor y \lor \boxed{\neg z})}$$



We need a way to ensure we never pick a literal and its negation.

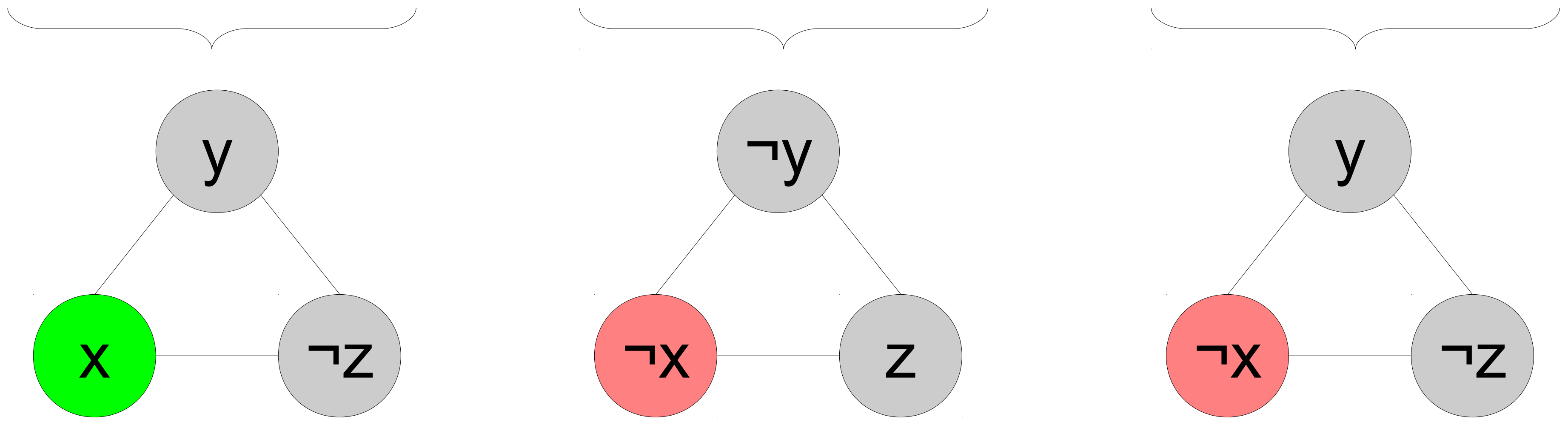
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



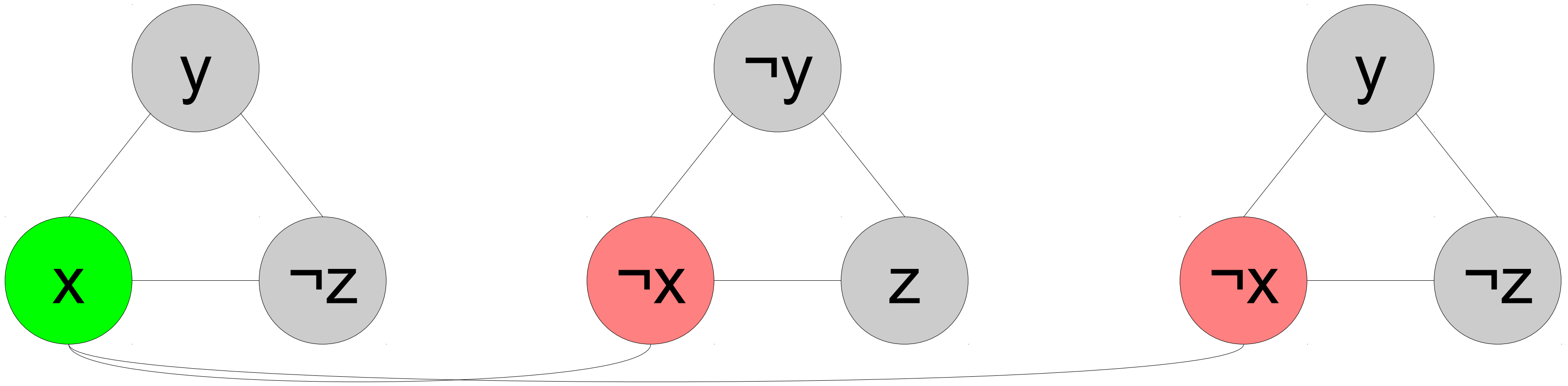
From 3SAT to INDSET

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$



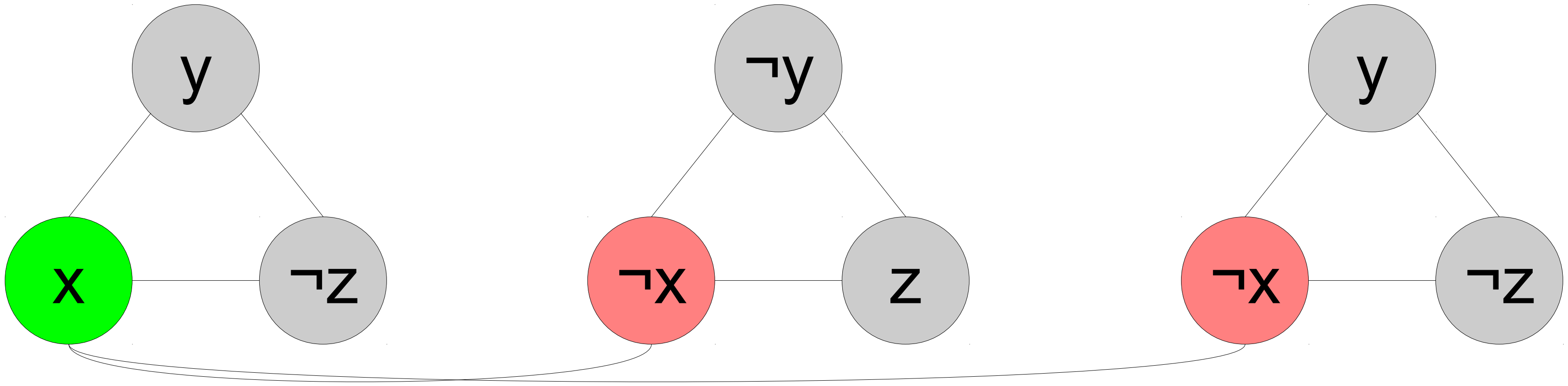
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



From 3SAT to INDSET

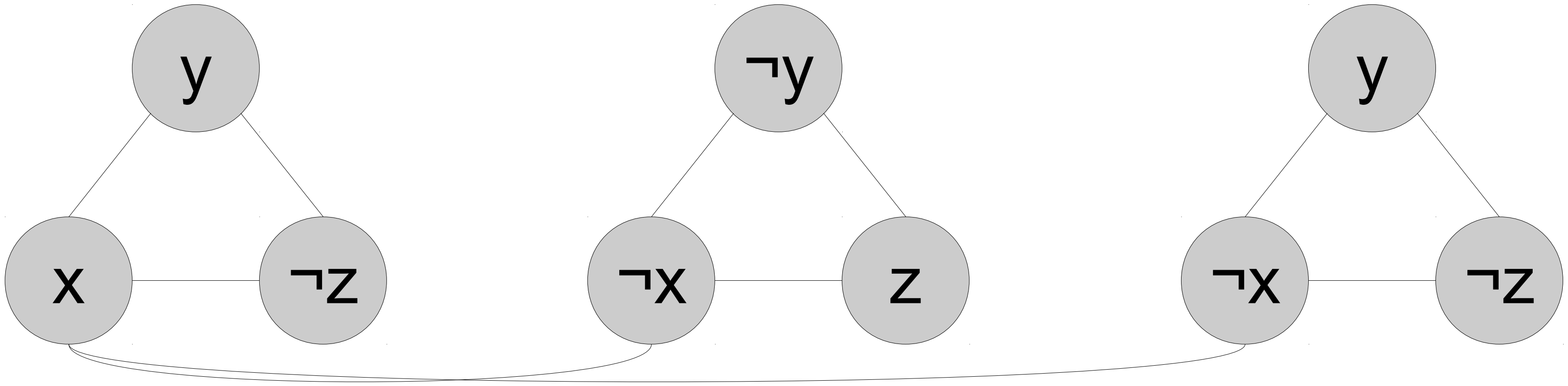
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



No independent set in this graph can choose two nodes labeled ***x*** and **¬*x***.

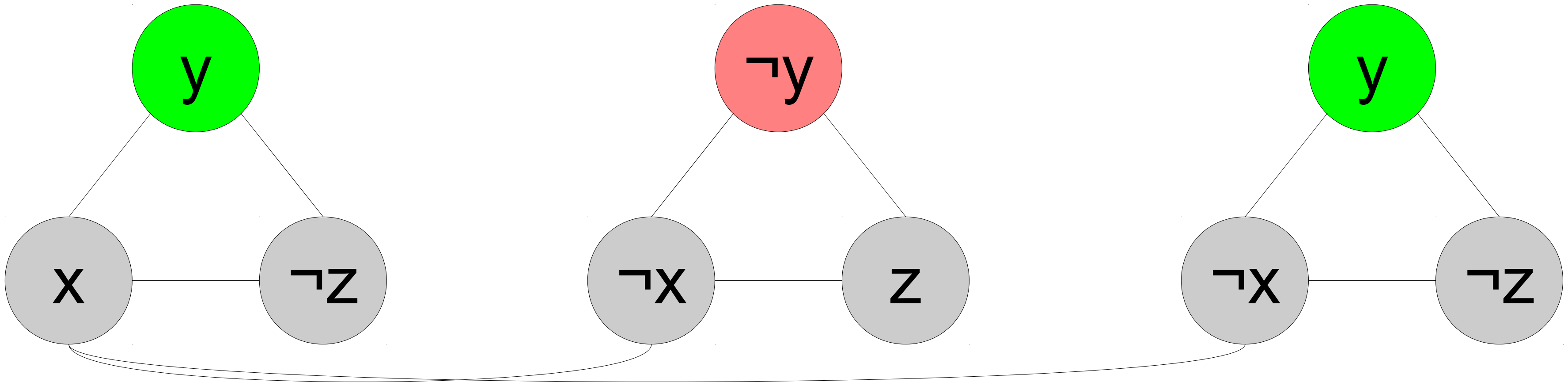
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



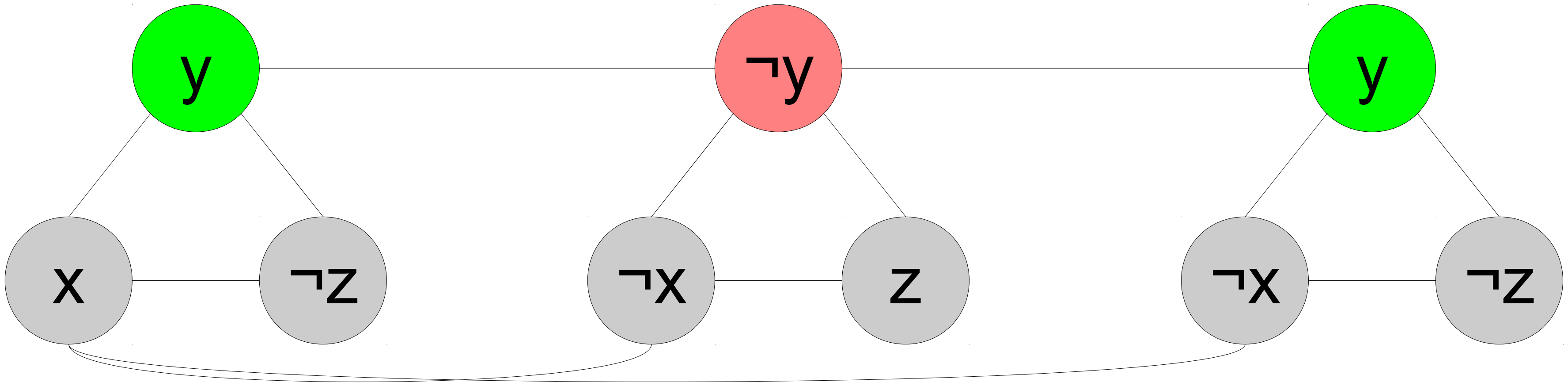
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

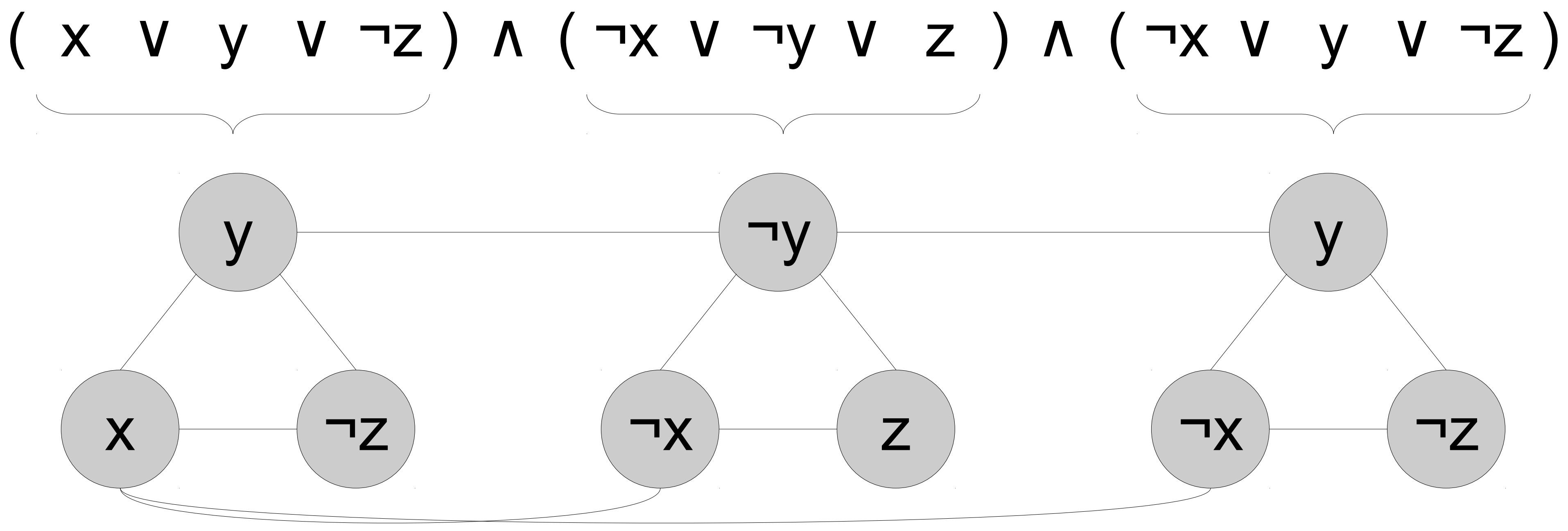


From 3SAT to INDSET

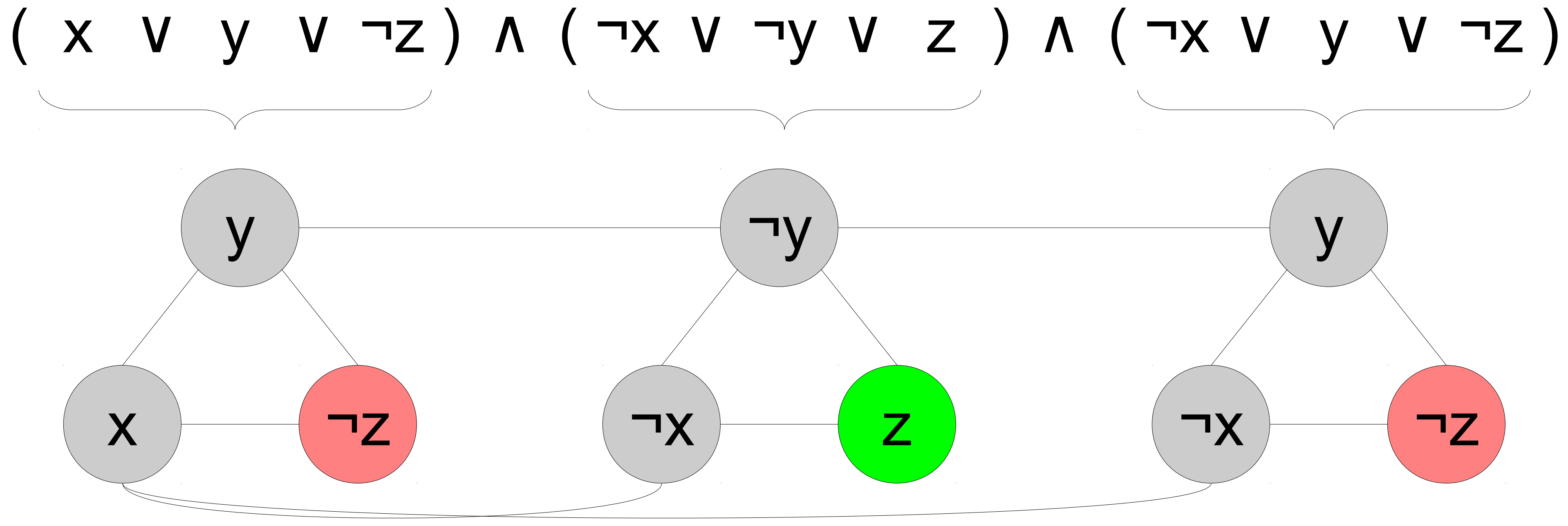
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



From 3SAT to INDSET

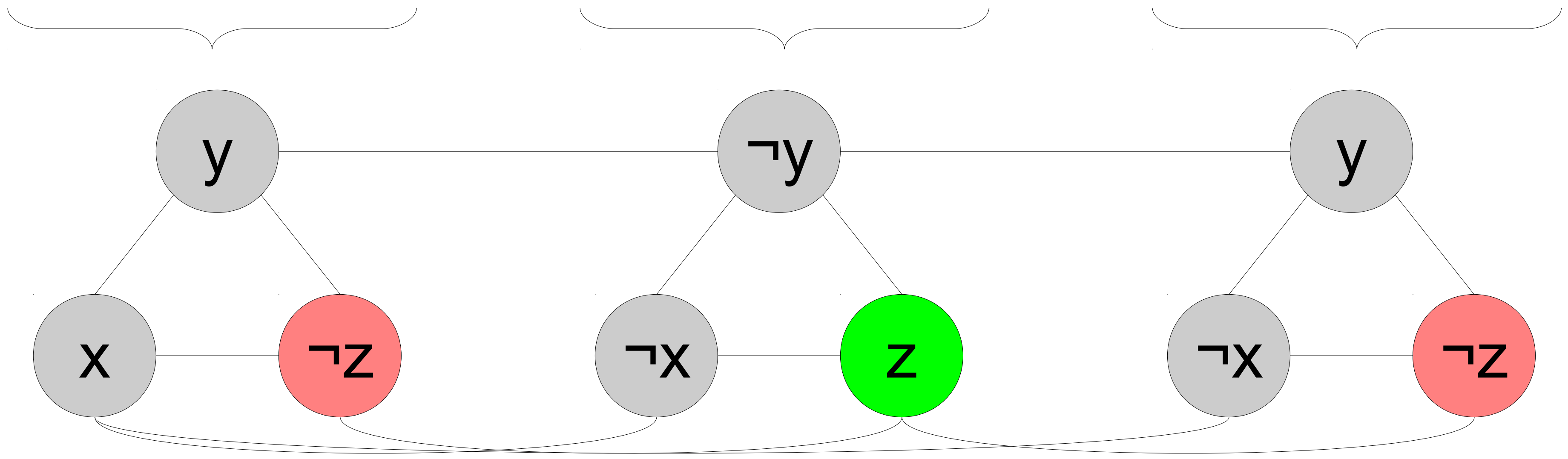


From 3SAT to INDSET



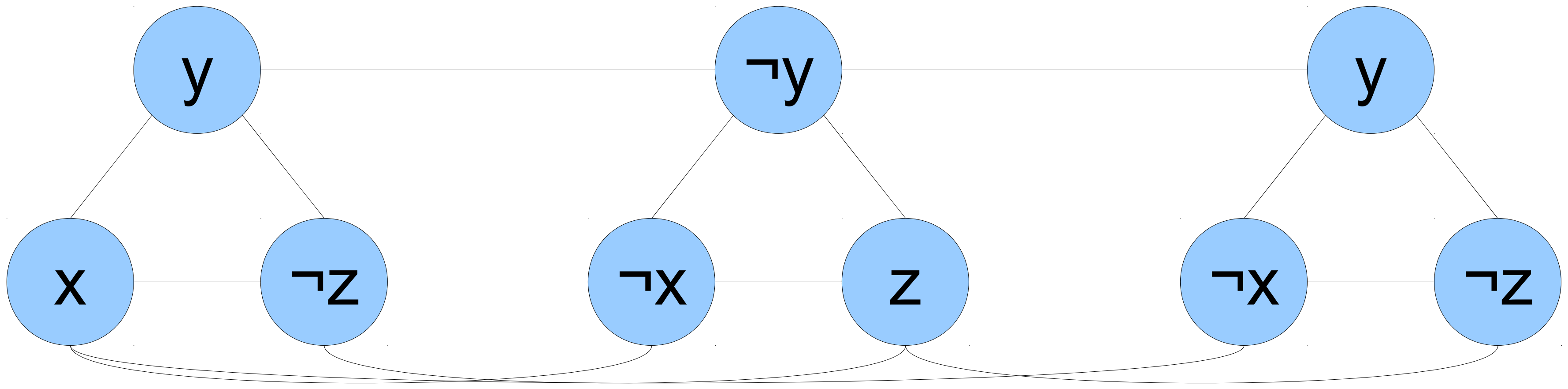
From 3SAT to INDSET

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$



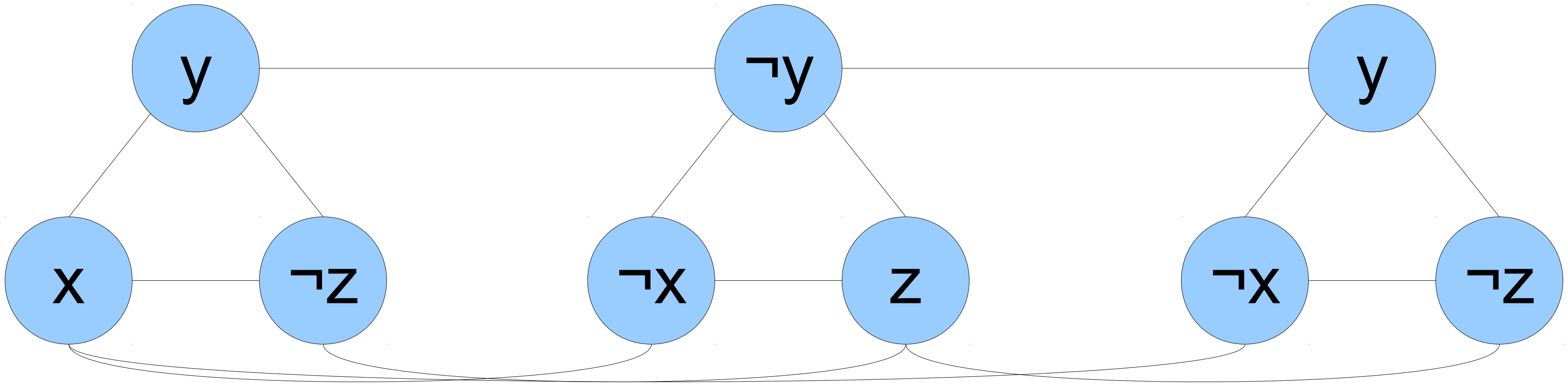
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



From 3SAT to INDSET

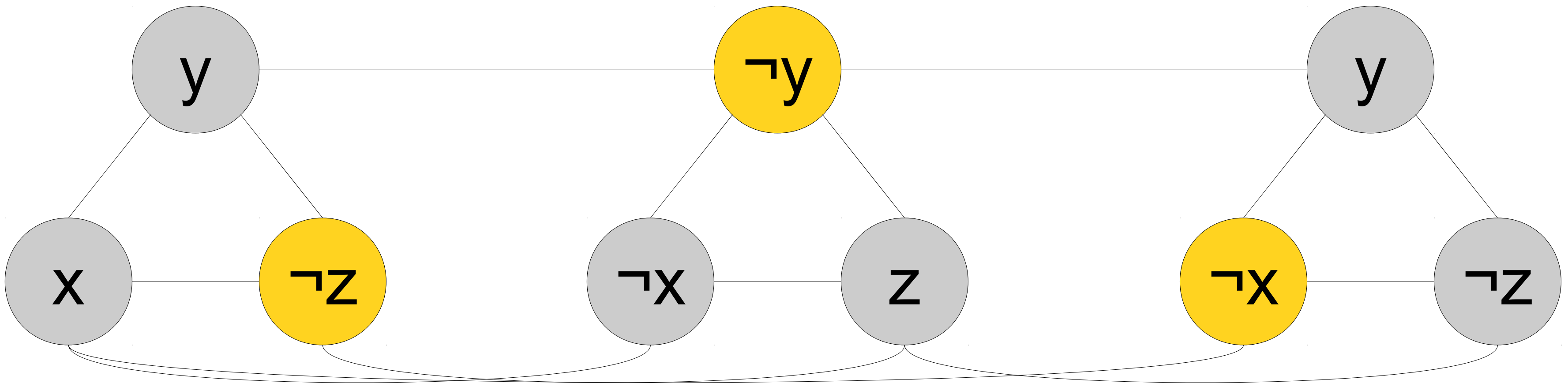
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

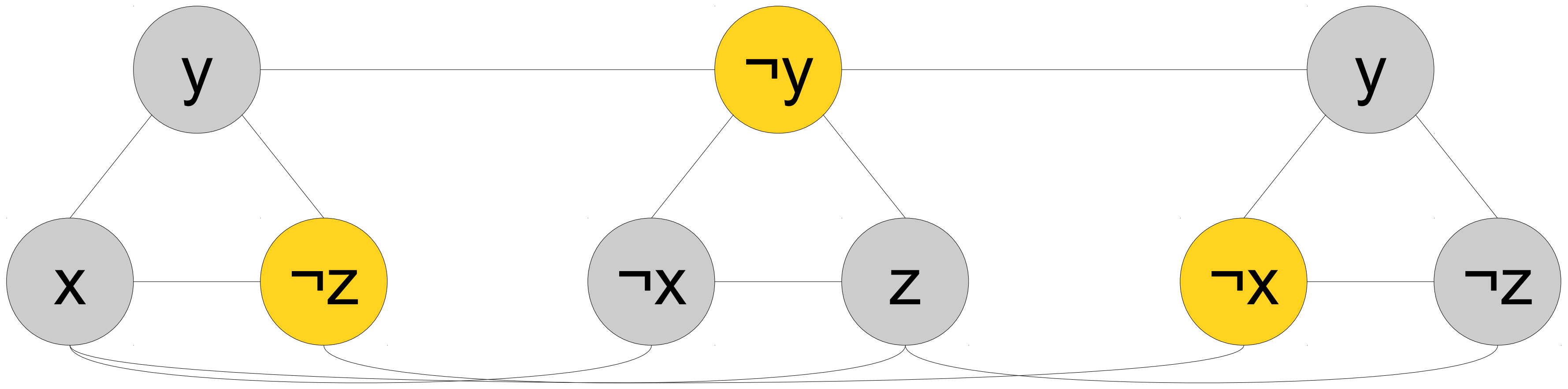
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

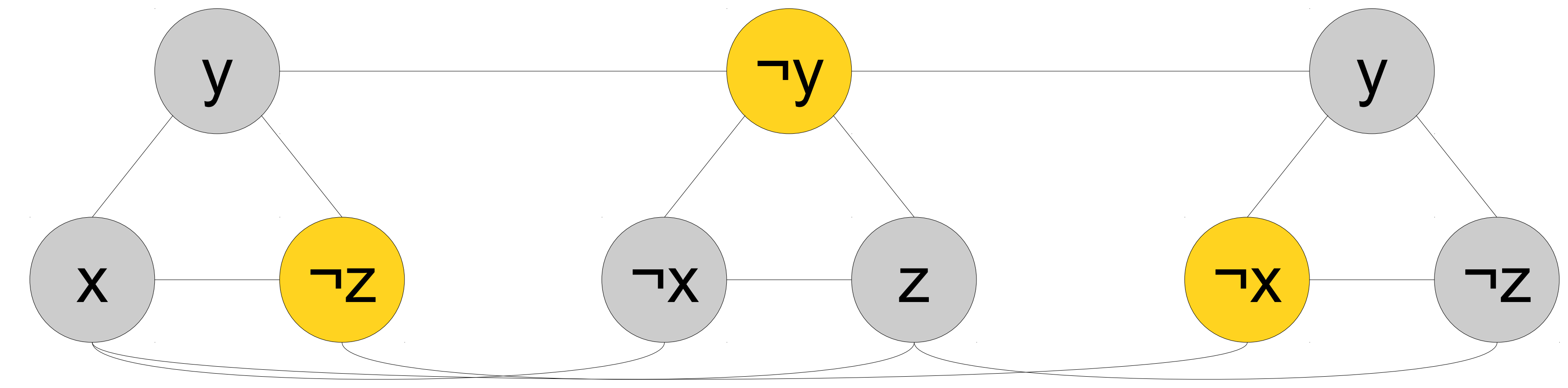


If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

x = false, y = false, z = false.

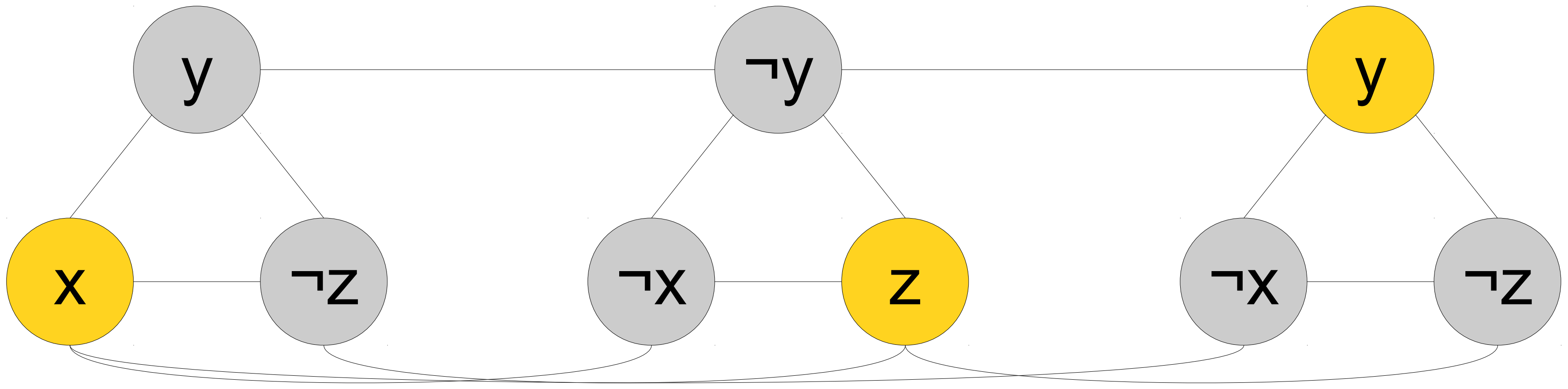
$$\underbrace{(\,x \lor y \lor \neg z\,)}\land\underbrace{(\,\neg x \lor \neg y \lor z\,)}\land\underbrace{(\,\neg x \lor y \lor \neg z\,)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

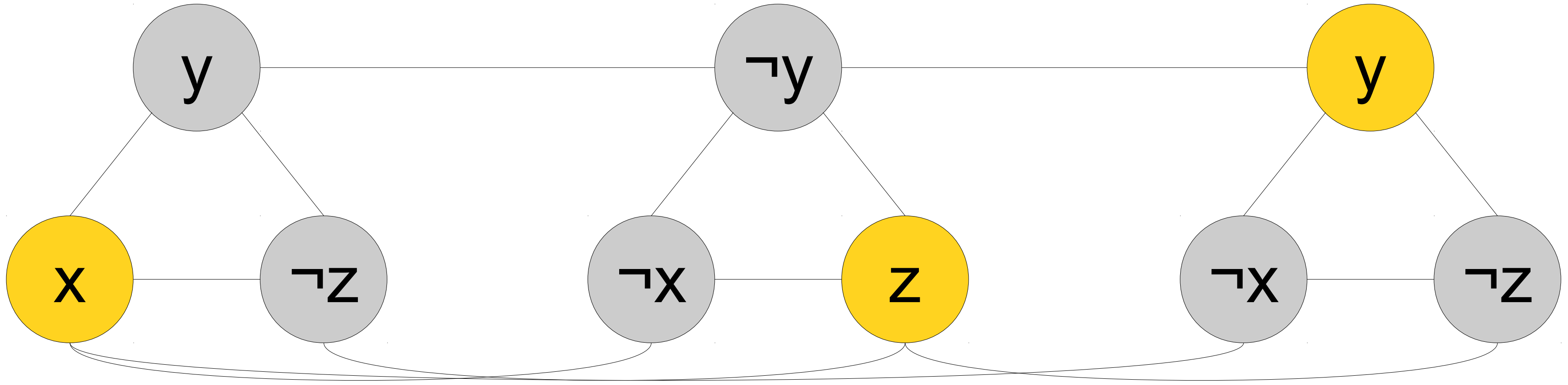
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

$$\underbrace{(\boxed{x} \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor \boxed{z})} \land \underbrace{(\neg x \lor \boxed{y} \lor \neg z)}$$

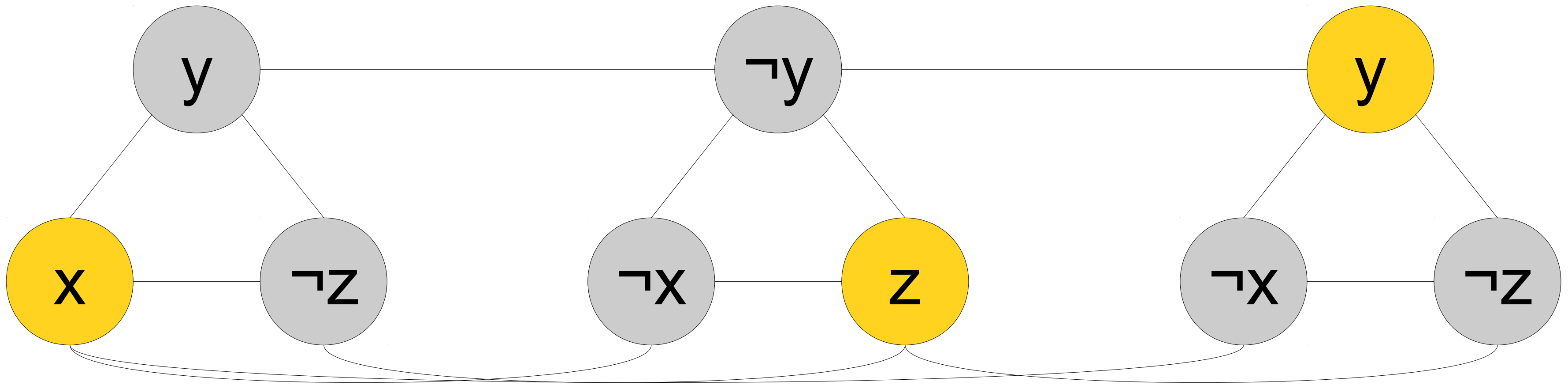


If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

x = true, y = true, z = true.

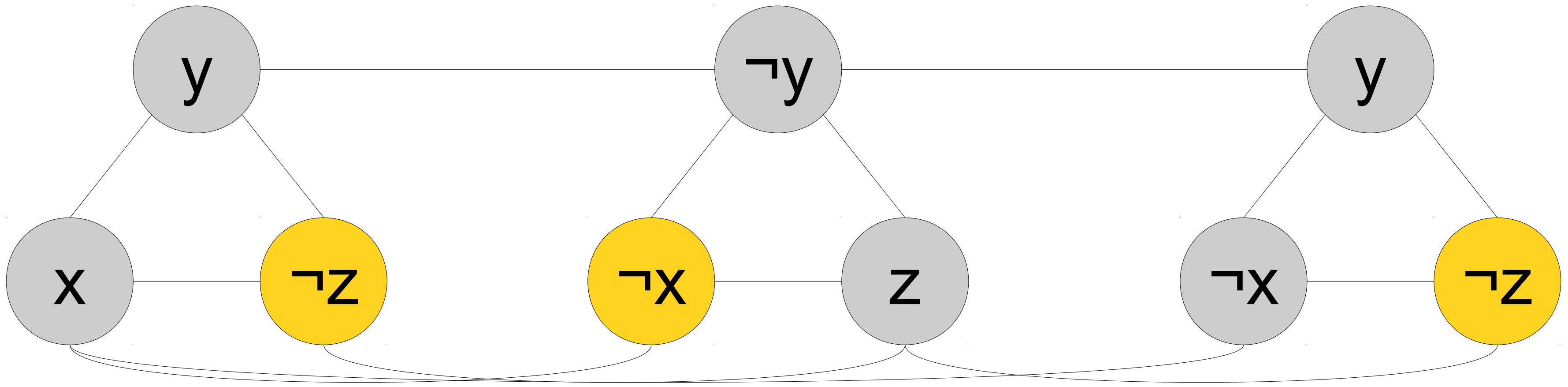
$$\underbrace{(\boxed{x} \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor \boxed{z})} \land \underbrace{(\neg x \lor \boxed{y} \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

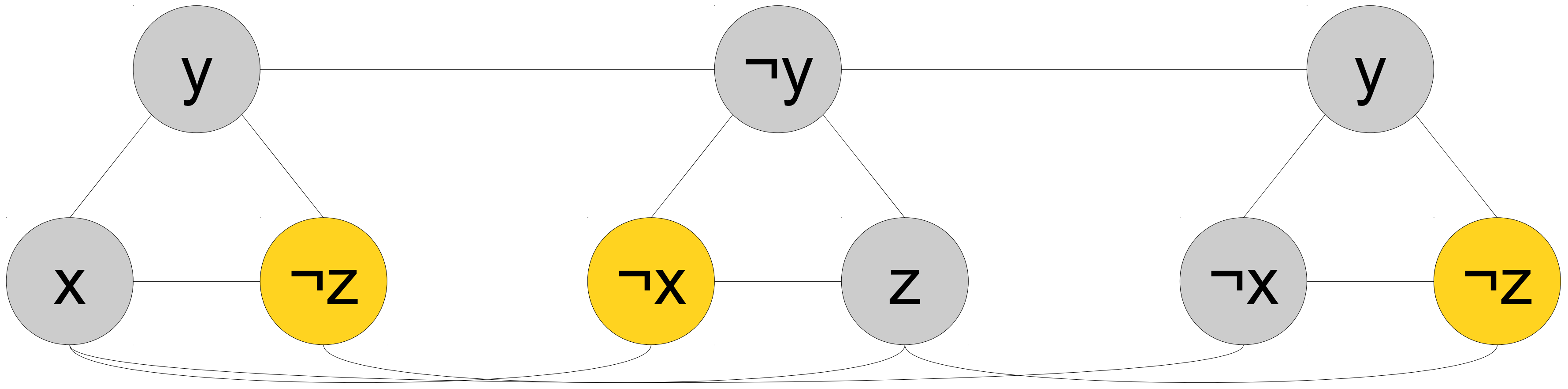
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

$$(\underbrace{x \lor y \lor \neg z}) \land (\underbrace{\neg x \lor \neg y \lor z}) \land (\underbrace{\neg x \lor y \lor \neg z})$$

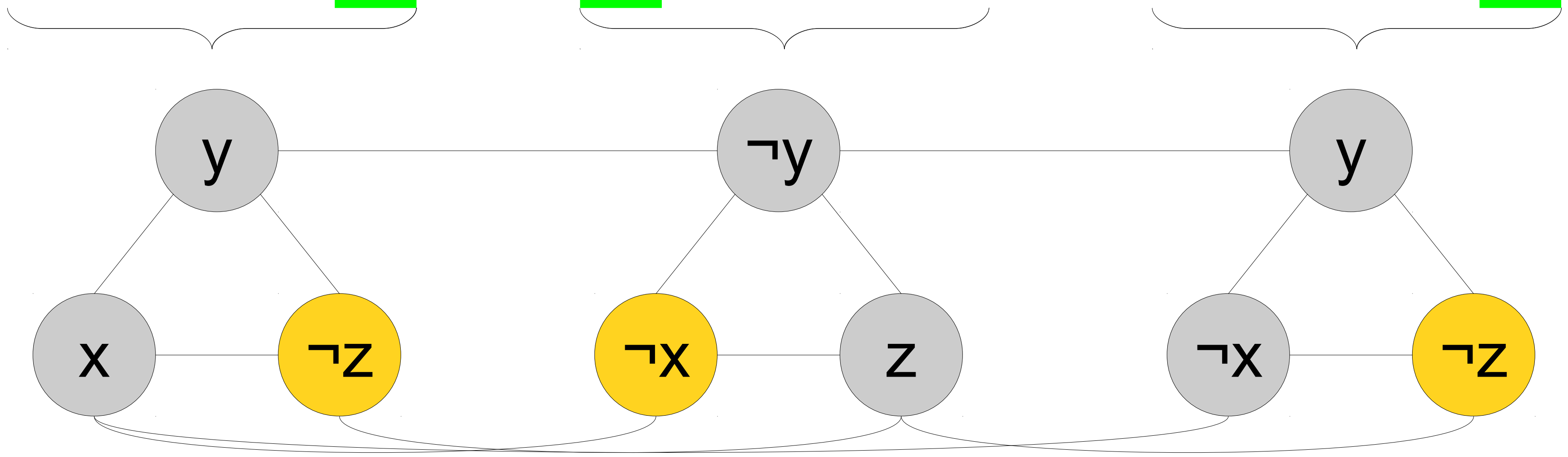


If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

x = false, y = ??, z = false.

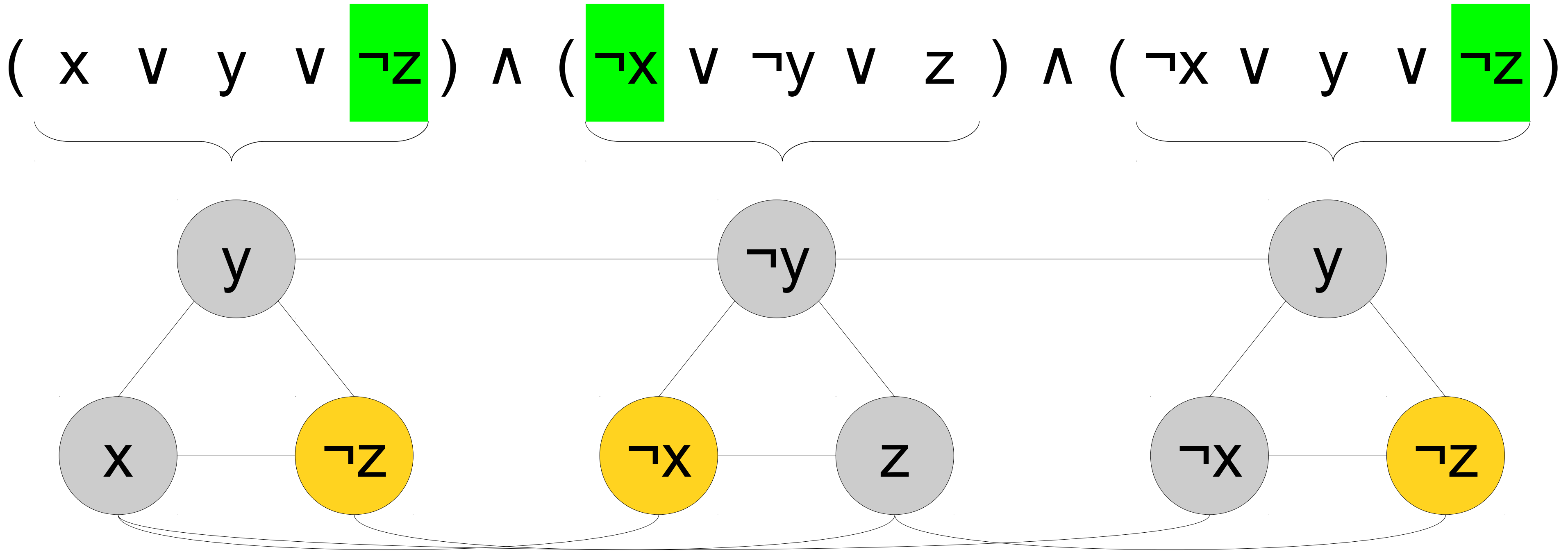
$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$



If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

x = false, y = true, z = false.

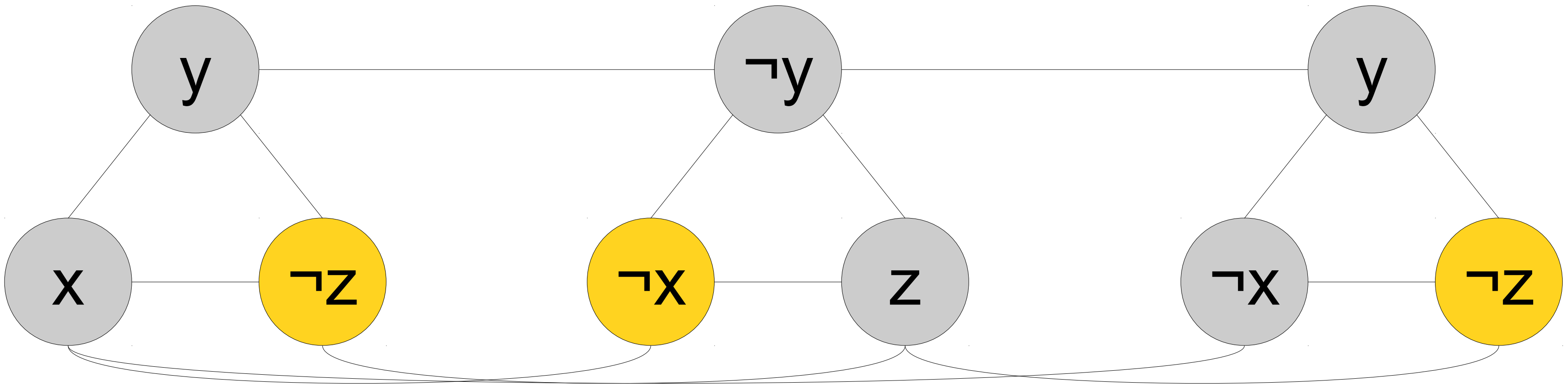


If this graph has an independent set of size three, the original formula is satisfiable.

From 3SAT to INDSET

x = false, y = false, z = false.

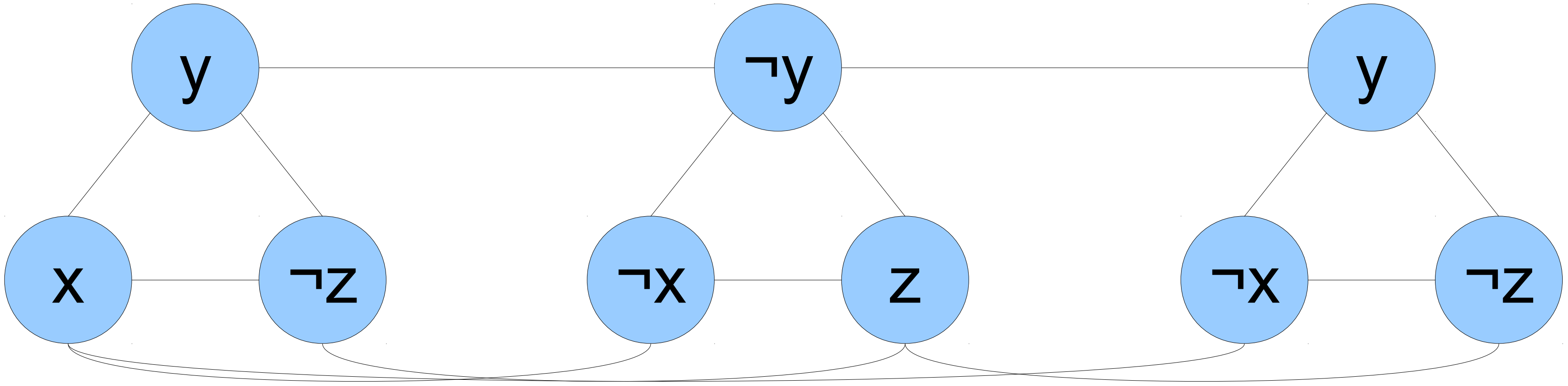
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If this graph has an independent set of size three, the original formula is satisfiable.

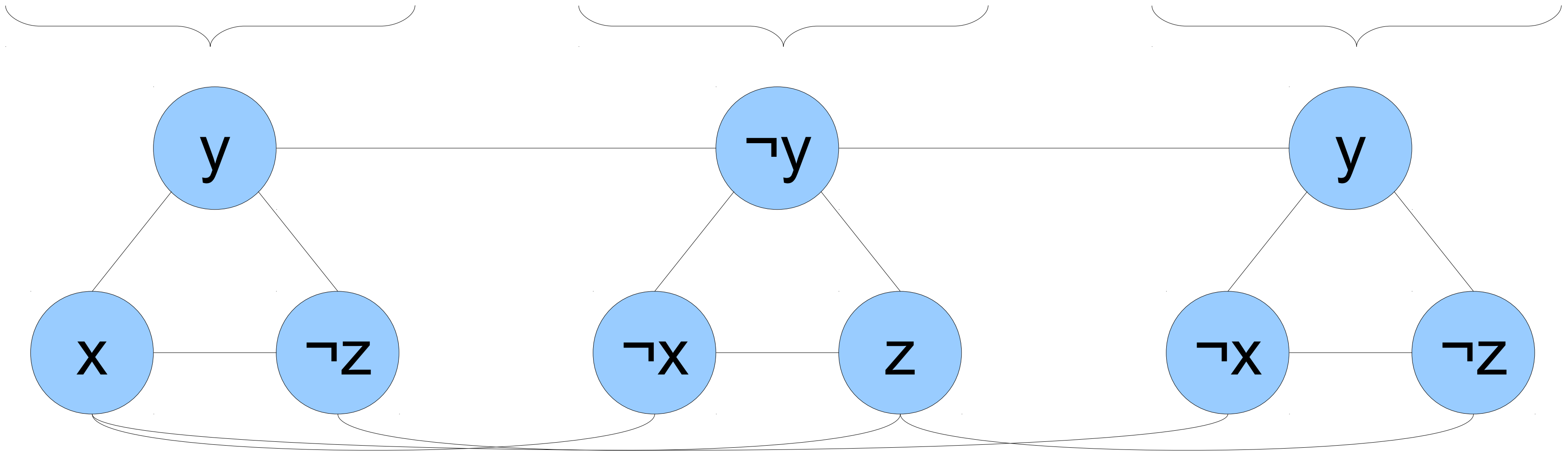
From 3SAT to INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



From 3SAT to INDSET

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$

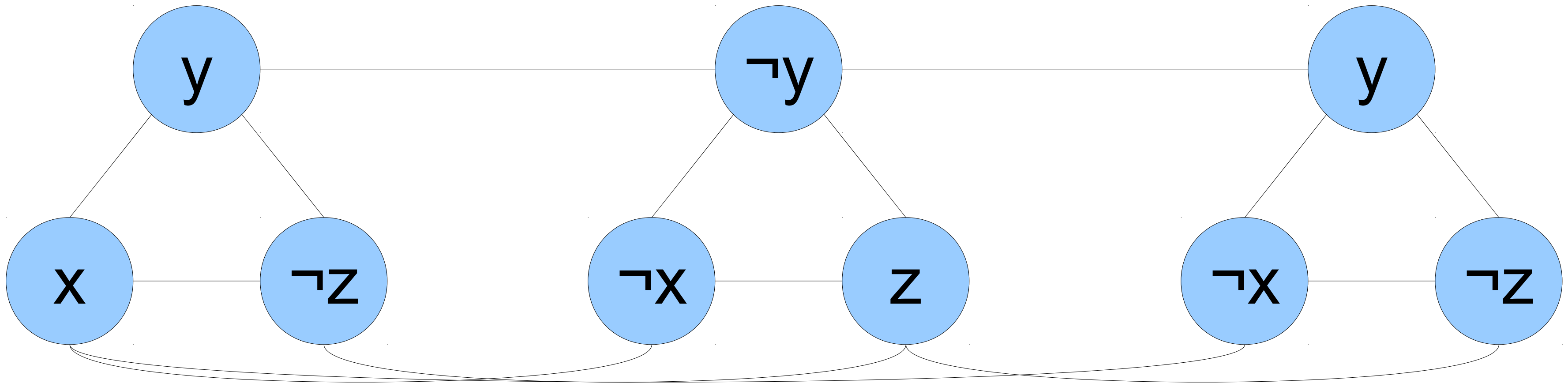


If the original formula is satisfiable,
this graph has an independent set of size three.

From 3SAT to INDSET

x = false, y = true, z = false.

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

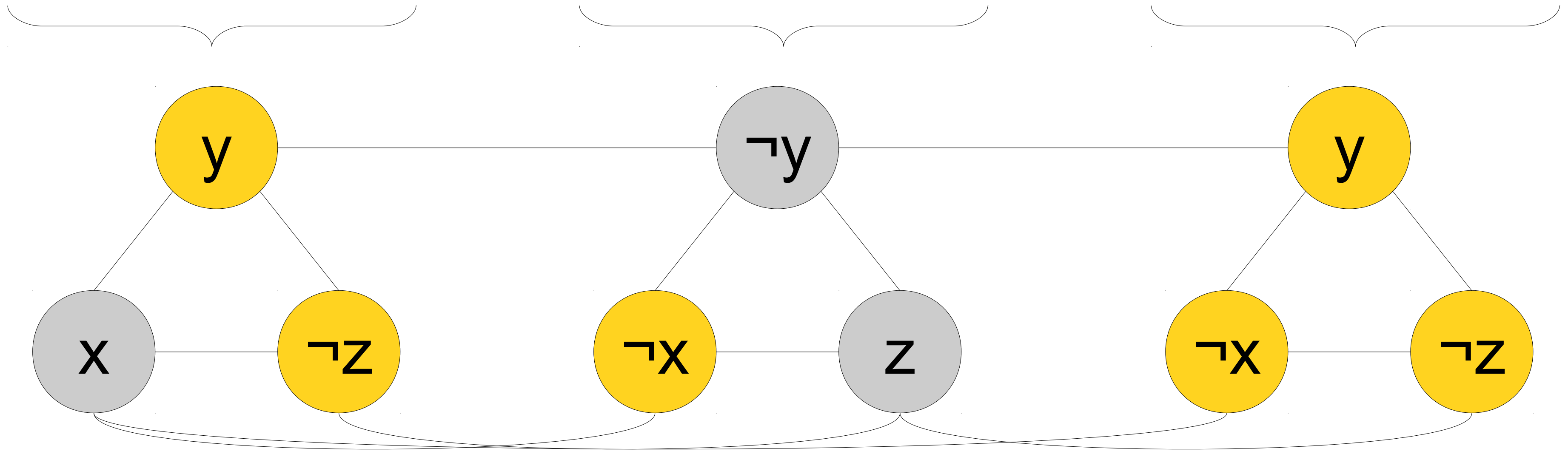


If the original formula is satisfiable,
this graph has an independent set of size three.

From 3SAT to INDSET

x = false, y = true, z = false.

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$

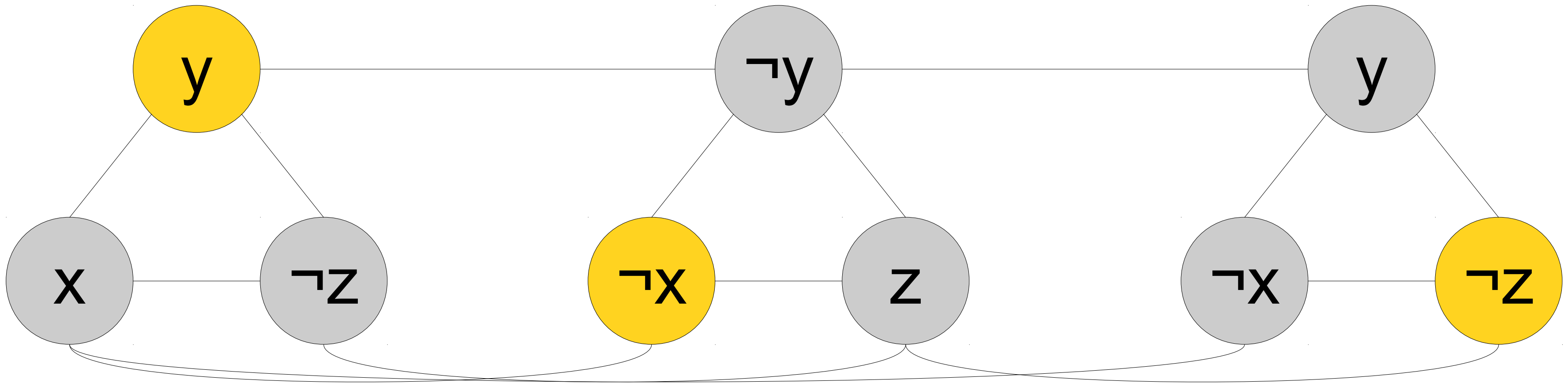


If the original formula is satisfiable,
this graph has an independent set of size three.

From 3SAT to INDSET

x = false, y = true, z = false.

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



If the original formula is satisfiable,
this graph has an independent set of size three.

From 3SAT to INDSET

- Let $\varphi = C_1 \land C_2 \land \ldots \land C_n$ be a 3-CNF formula.
- Construct the graph G as follows:
  - For each clause $C_\mathrm{i} = x_1 \lor x_2 \lor x_3$, where x_1, x_2, and x_3 are literals, add three new nodes into G with edges connecting them.
  - For each pair of nodes v_i and $\neg v_i$, where v_i is some variable, add an edge connecting v_i and $\neg v_i$. (Note that there are multiple copies of these nodes)
- ***Claim One:*** This reduction can be computed in polynomial time.
- ***Claim Two:*** G has an independent set of size n iff φ is satisfiable.

INDSET ∈ **NPC**

- ***Theorem:*** INDSET is **NP**-complete.
- ***Proof sketch:*** We just showed that INDSET ∈ **NP** and that 3SAT $\le_p$ INDSET. Therefore, INDSET is **NP**-complete. ■

Time-Out For Announcements!

Please evaluate this course in Axess.

Your feedback really makes a difference.

Final Exam Logistics

- The final exam will be one week from today: **Wednesday, December 9** from **3:30PM – 6:30PM** in **Cemex Auditorium**.
- As before, the exam is closed-book, closed-computer, and limited-note. You can have one 8.5” × 11”, double-sided sheet of notes with you when you take the exam.
- Topic coverage is cumulative with a slight focus on PS7 – PS9.

Extra Practice

- We've released three sets of extra practice problems and a set of challenge problems.
- Solutions to the extra practice problems are available in Gates.
- We will also release a practice final exam online (sorry, we didn't get a room for a practice exam). Solutions will be available in Gates starting on Friday.

Don Knuth Talk

- Every year around this time, Don Knuth gives a “Computer Musings” lecture on whatever topic seems most interesting.
- The next one is tomorrow night at 6PM in Nvidia Auditorium.
- Might be a lot of fun – check it out!

Your Questions

“Why didn't you major in math?”

Because I like CS!

☺

“I'm curious to know what you think about the lack of diversity among Stanford's faculty.”

This is a real problem. The School of Engineering is taking a lot of steps in the right direction, but still has a long ways to go.

I think we need to reframe the narrative. Everyone always talks about "lowering the bar" or "lowering standards" when getting more diverse hires. I'm amazed how often I hear these phrases thrown around. Really, we have a sourcing, networking, and mentoring problem. I don't see this as being much different from lots of Silicon Valley diversity issues.

“My understanding in this class seems to have gone completely off the rails once we started talking about **R** and **RE** languages and everything since. What's your advice for getting back on track and learning the material?”

Come stop by office hours and ask a lot of questions. Figure out where you no longer understand things, then chat with the course staff to see where things stopped making sense. There are a ton of concepts here and their interplay is pretty tricky, but I think you'll find that once you see how everything fits together, it's beautiful and makes a lot of sense.

Back to CS103!

Structuring **NP**-Completeness Reductions

The Shape of a Reduction

- Polynomial-time reductions work by solving one problem with a solver for a different problem.
- Most problems in **NP** have different pieces that must be solved simultaneously.
- For example, in 3SAT:
  - Each clause must be made true,
  - but no literal and its complement may be picked.
- In INDSET:
  - You can choose any nodes you want to put into the set,
  - but no two connected nodes can be added.

Reductions and Gadgets

- Many reductions used to show **NP**-completeness work by using ***gadgets***.
- Each piece of the original problem is translated into a “gadget” that handles some particular detail of the problem.
- These gadgets are then connected together to solve the overall problem.

Gadgets in INDSET

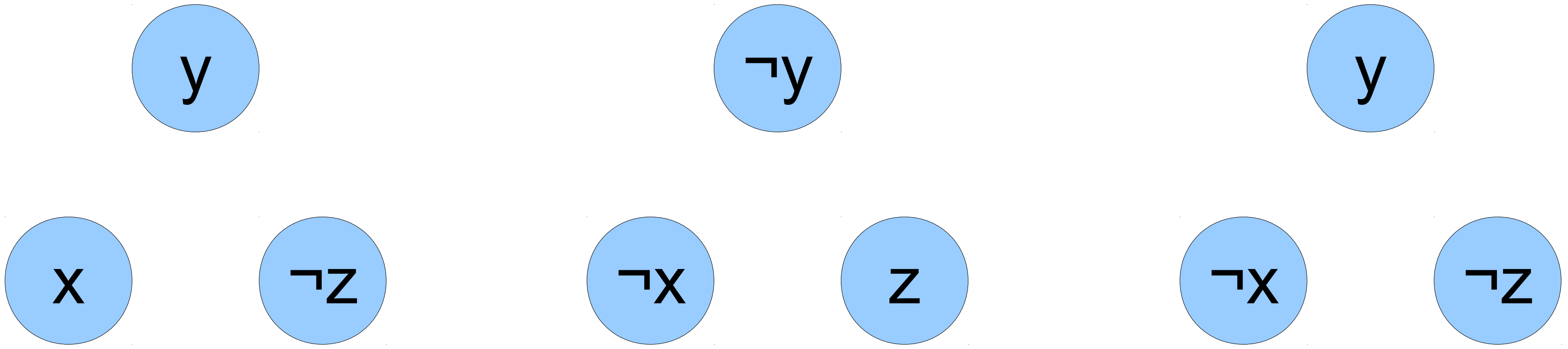
$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$

Gadgets in INDSET

$$(\underbrace{x \lor y \lor \neg z}_{}) \land (\underbrace{\neg x \lor \neg y \lor z}_{}) \land (\underbrace{\neg x \lor y \lor \neg z}_{})$$

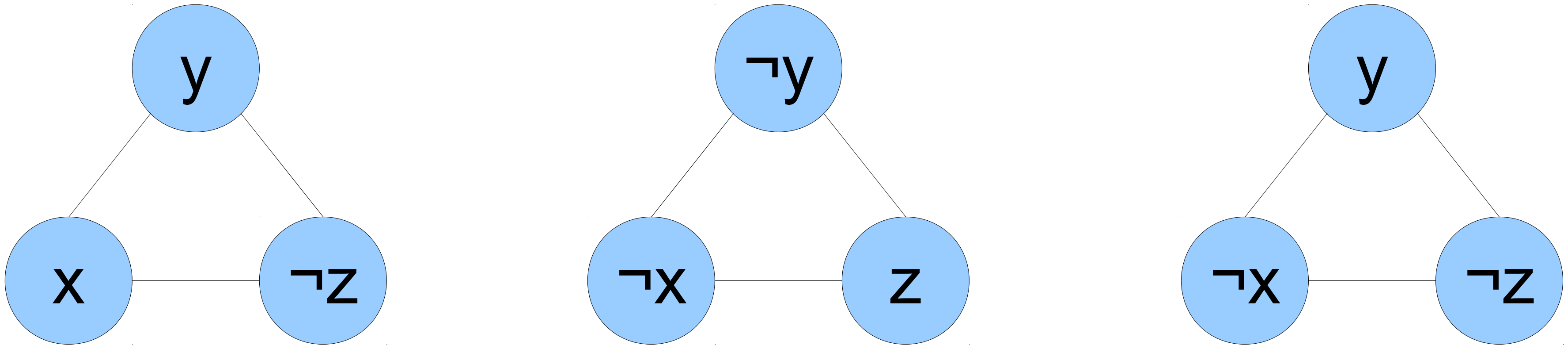
Gadgets in INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



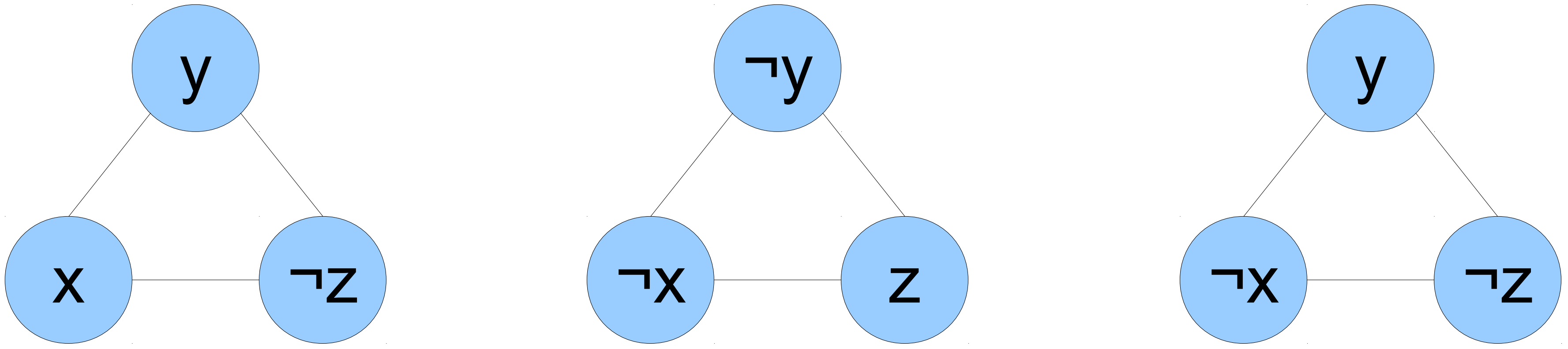
Gadgets in INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Gadgets in INDSET

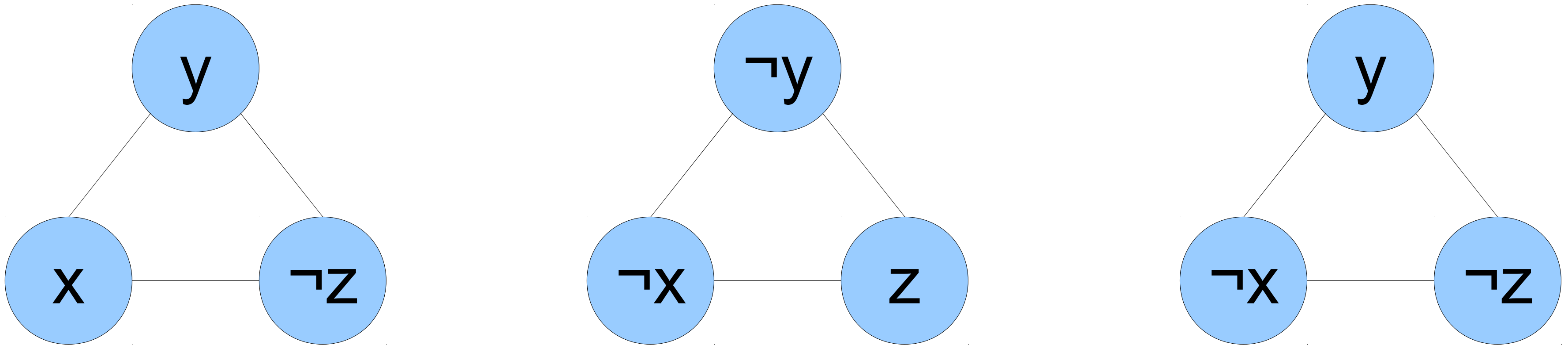
$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



Each of these gadgets is designed to solve one part of the problem: ensuring each clause is satisfied.

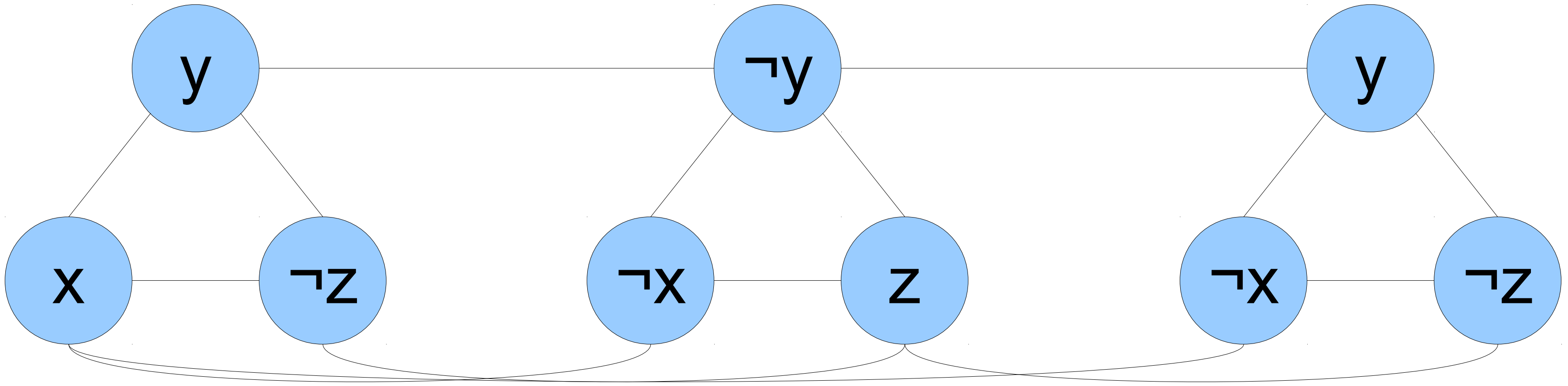
Gadgets in INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



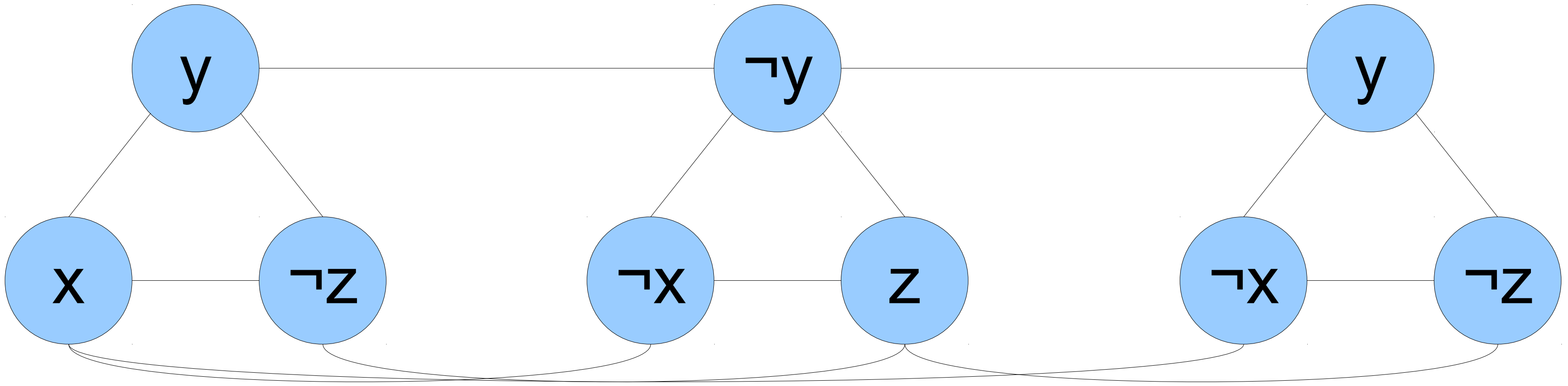
Gadgets in INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



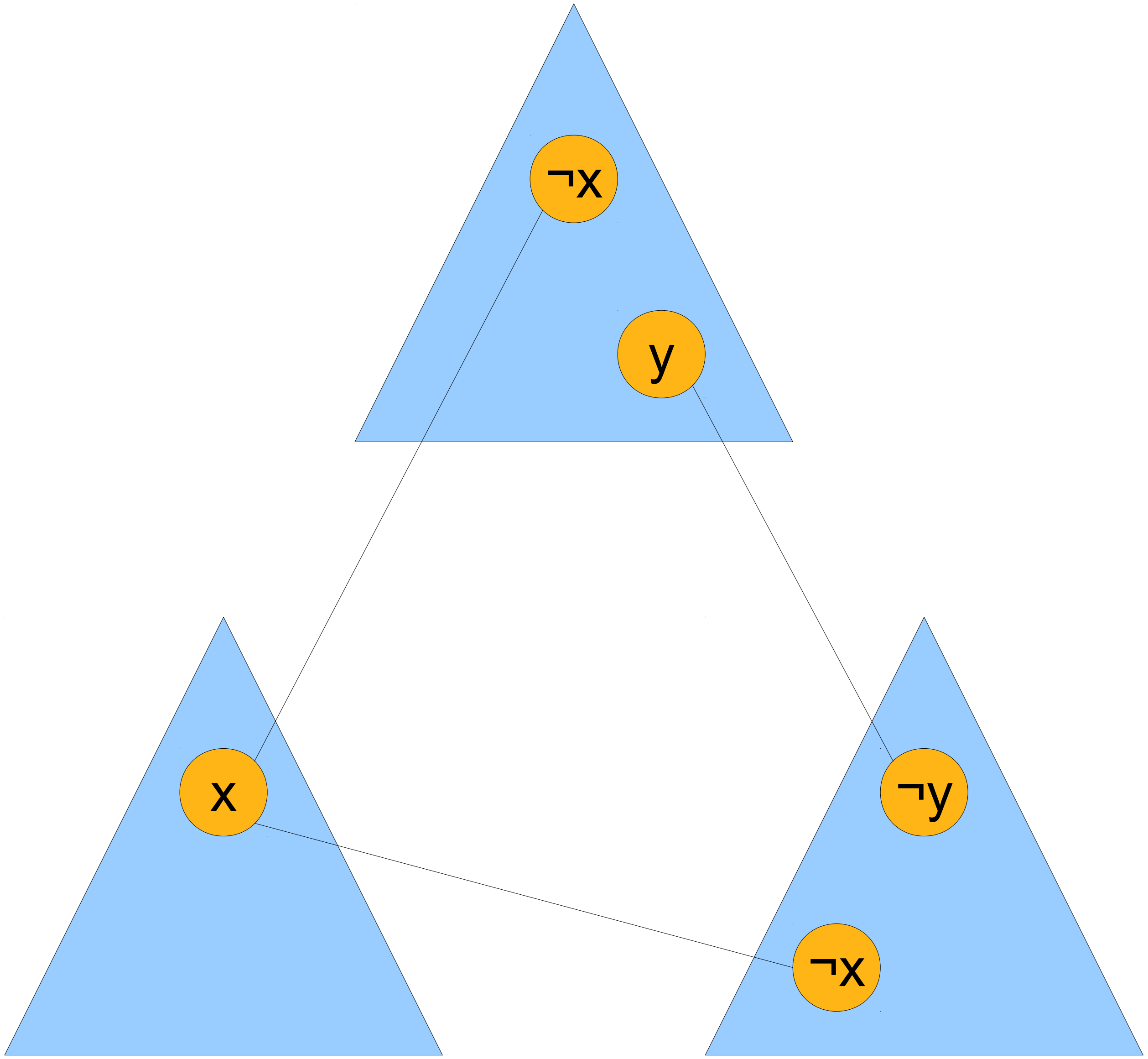
Gadgets in INDSET

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$



These connections ensure that the solutions to each gadget are linked to one another.

Gadgets in INDSET



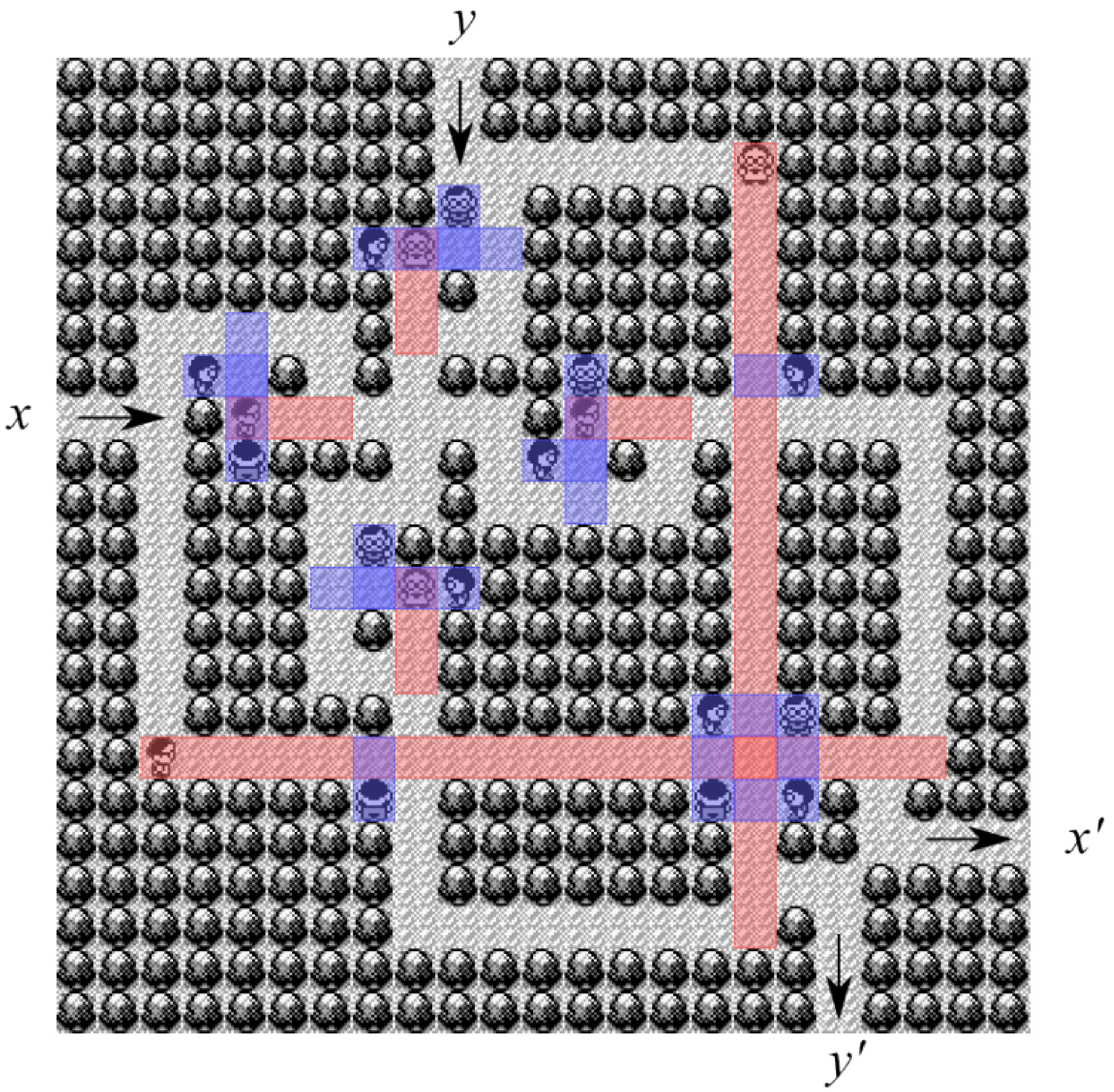
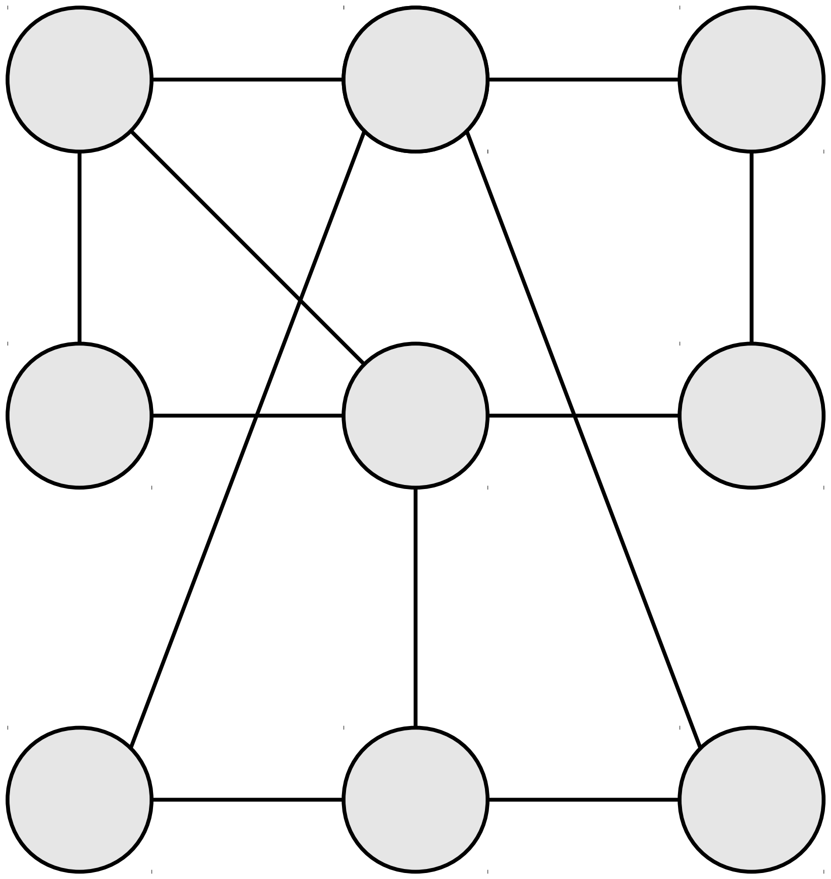


Figure 36: Crossover gadget for Pokémon

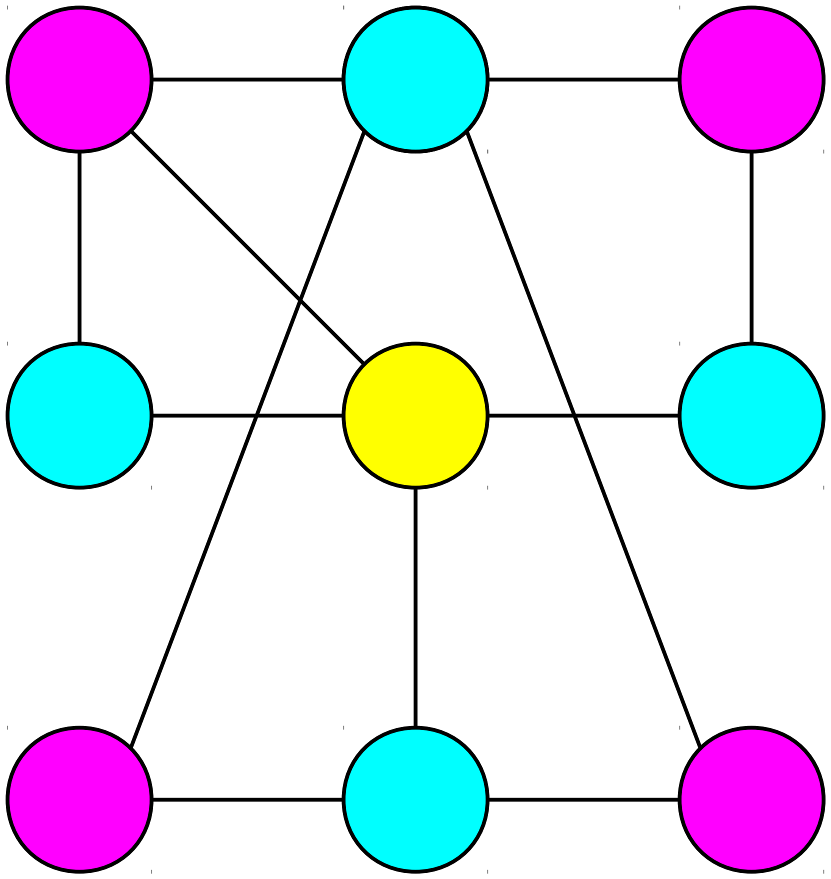
Aloupis et al. “Classic Nintendo Games are (Computationally Hard)”

A More Complex Reduction

A ***3-coloring*** of a graph is a way of coloring its nodes one of three colors such that no two connected nodes have the same color.



A ***3-coloring*** of a graph is a way of coloring its nodes one of three colors such that no two connected nodes have the same color.



A ***3-coloring*** of a graph is a way of coloring its nodes one of three colors such that no two connected nodes have the same color.

The 3-Coloring Problem

- The ***3-coloring problem*** is

  Given an undirected graph *G*, is there a legal 3-coloring of its nodes?

- As a formal language:

  3COLOR = { ⟨*G*⟩ | *G* is an undirected graph with a legal 3-coloring. }

- This problem is known to be **NP**-complete by a reduction from 3SAT.

3COLOR ∈ **NP**

- We can prove that 3COLOR ∈ **NP** by designing a polynomial-time verifier for 3COLOR.
- V = "On input $\langle G, C \rangle$, where G is a graph and C is a way of assigning colors to the nodes of G:
    - Check whether each edge's endpoints are different colors.
    - If so, accept; otherwise reject."

A Note on Terminology

- Although 3COLOR and 3SAT both have “3” in their names, the two are very different problems.
  - 3SAT means “there are three literals in every clause.” However, each literal can take on only one of two different values.
  - 3COLOR means “every node can take on one of three different colors.”
- **Key difference**:
  - In 3SAT variables have two choices of value.
  - In 3COLOR nodes have three choices of value.

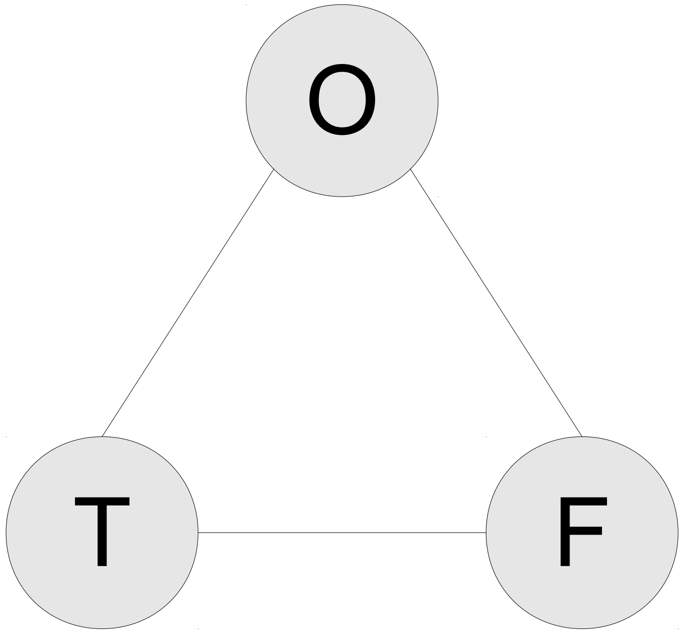
Why Not Two Colors?

- It would seem that 2COLOR (whether a graph has a 2-coloring) would be a better fit.
  - Every variable has one of two values.
  - Every node has one of two values.
- Interestingly, 2COLOR is known to be in **P** and is conjectured not to be **NP**-complete.
  - Though, if you can prove that it is, you've just won $1,000,000!

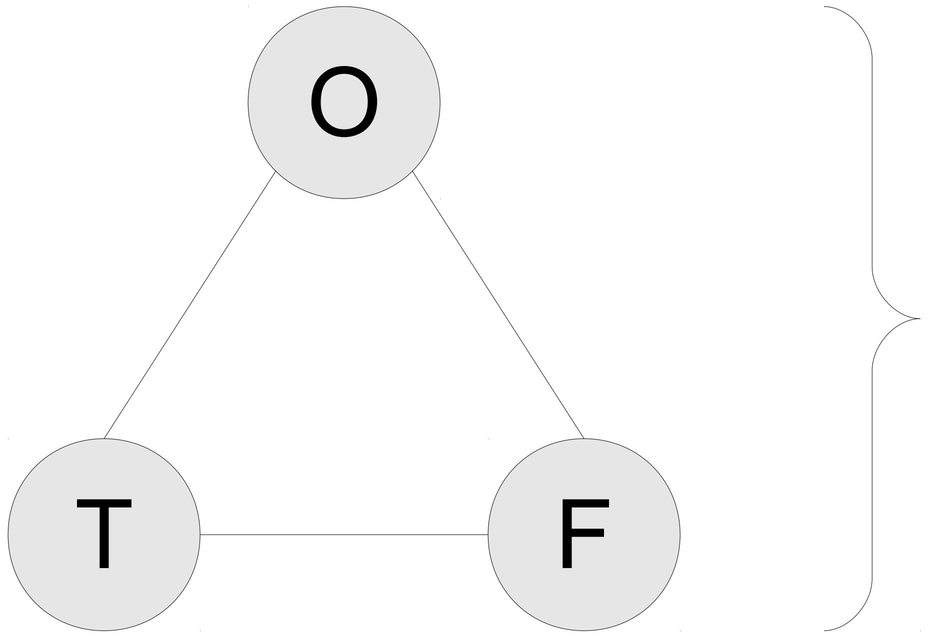
From 3SAT to 3COLOR

- In order to reduce 3SAT to 3COLOR, we need to somehow make a graph that is 3-colorable iff some 3-CNF formula φ is satisfiable.
- ***Idea:*** Use a collection of gadgets to solve the problem.
  - Build a gadget to assign two of the colors the labels “true” and “false.”
  - Build a gadget to force each variable to be either true or false.
  - Build a series of gadgets to force those variable assignments to satisfy each clause.

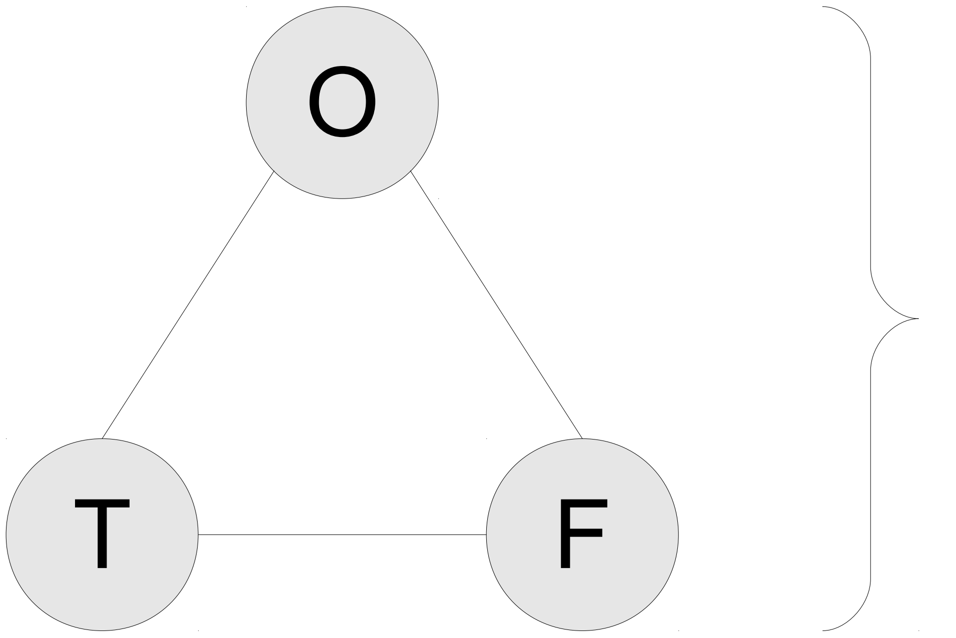
Gadget One: Assigning Meanings



Gadget One: Assigning Meanings



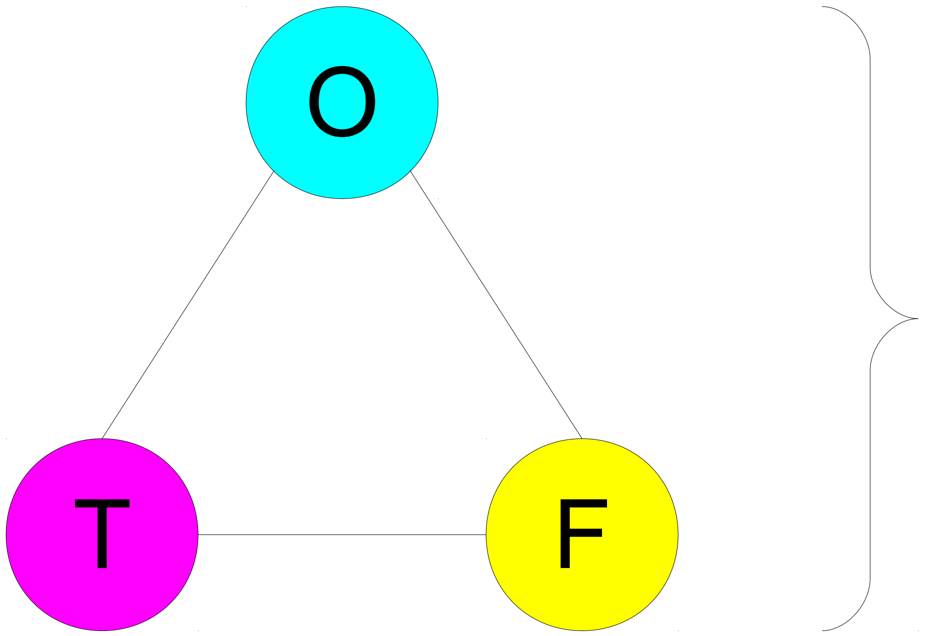
Gadget One: Assigning Meanings



These nodes must all have different colors.

The color assigned to T will be interpreted as "true."
The color assigned to F will be interpreted as "false."
We do not associate any special meaning with O.

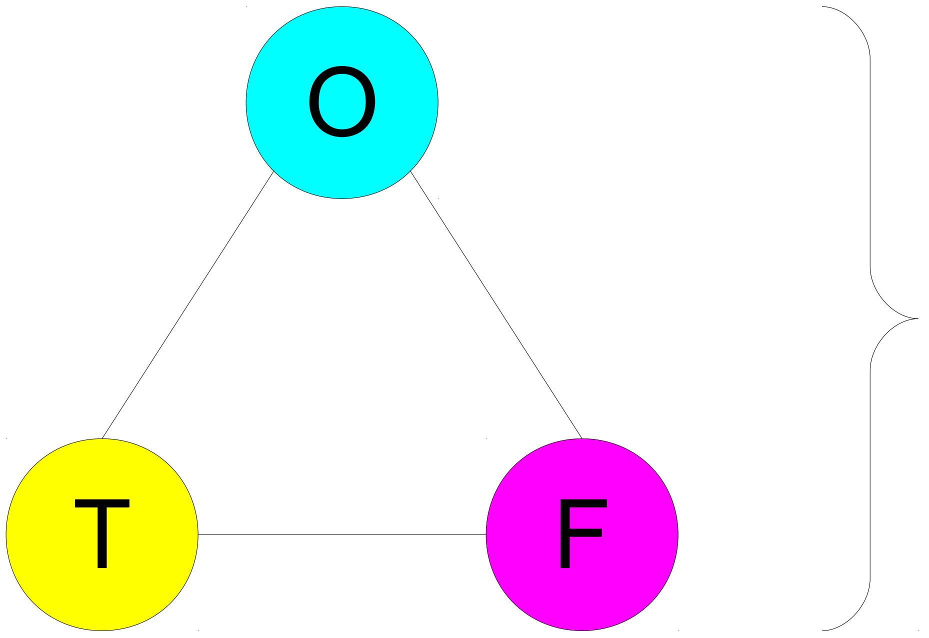
Gadget One: Assigning Meanings



These nodes must all have different colors.

The color assigned to T will be interpreted as "true."
The color assigned to F will be interpreted as "false."
We do not associate any special meaning with O.

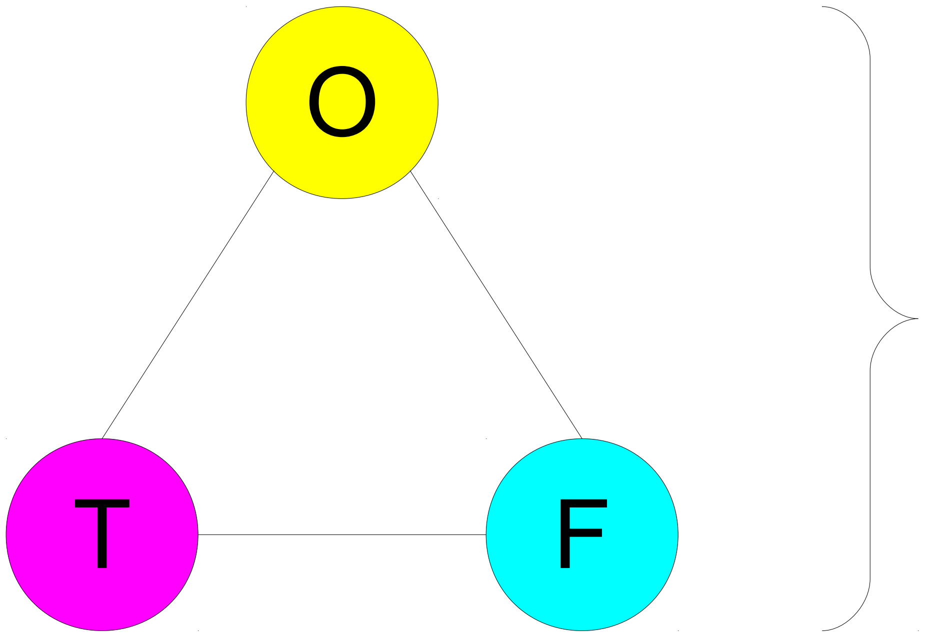
Gadget One: Assigning Meanings



These nodes must all have different colors.

The color assigned to T will be interpreted as "true."
The color assigned to F will be interpreted as "false."
We do not associate any special meaning with O.

Gadget One: Assigning Meanings



These nodes must all have different colors.

The color assigned to T will be interpreted as "true."
The color assigned to F will be interpreted as "false."
We do not associate any special meaning with O.

Gadget Two: Forcing a Choice

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$

Gadget Two: Forcing a Choice

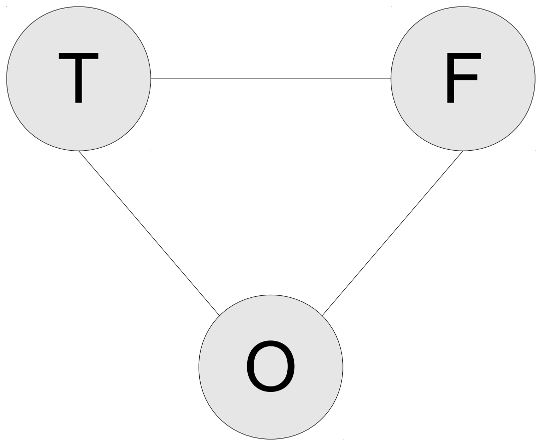
$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$

Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$

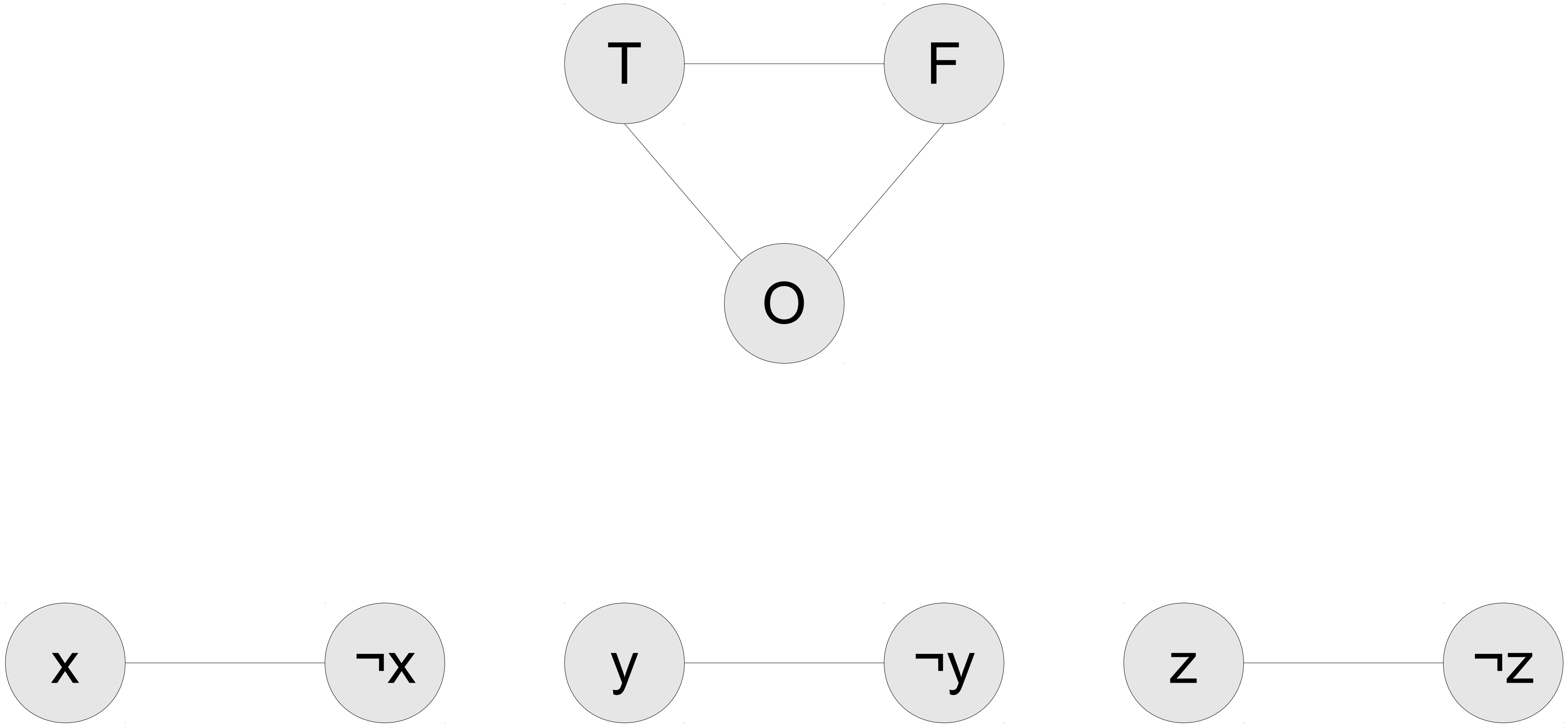
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



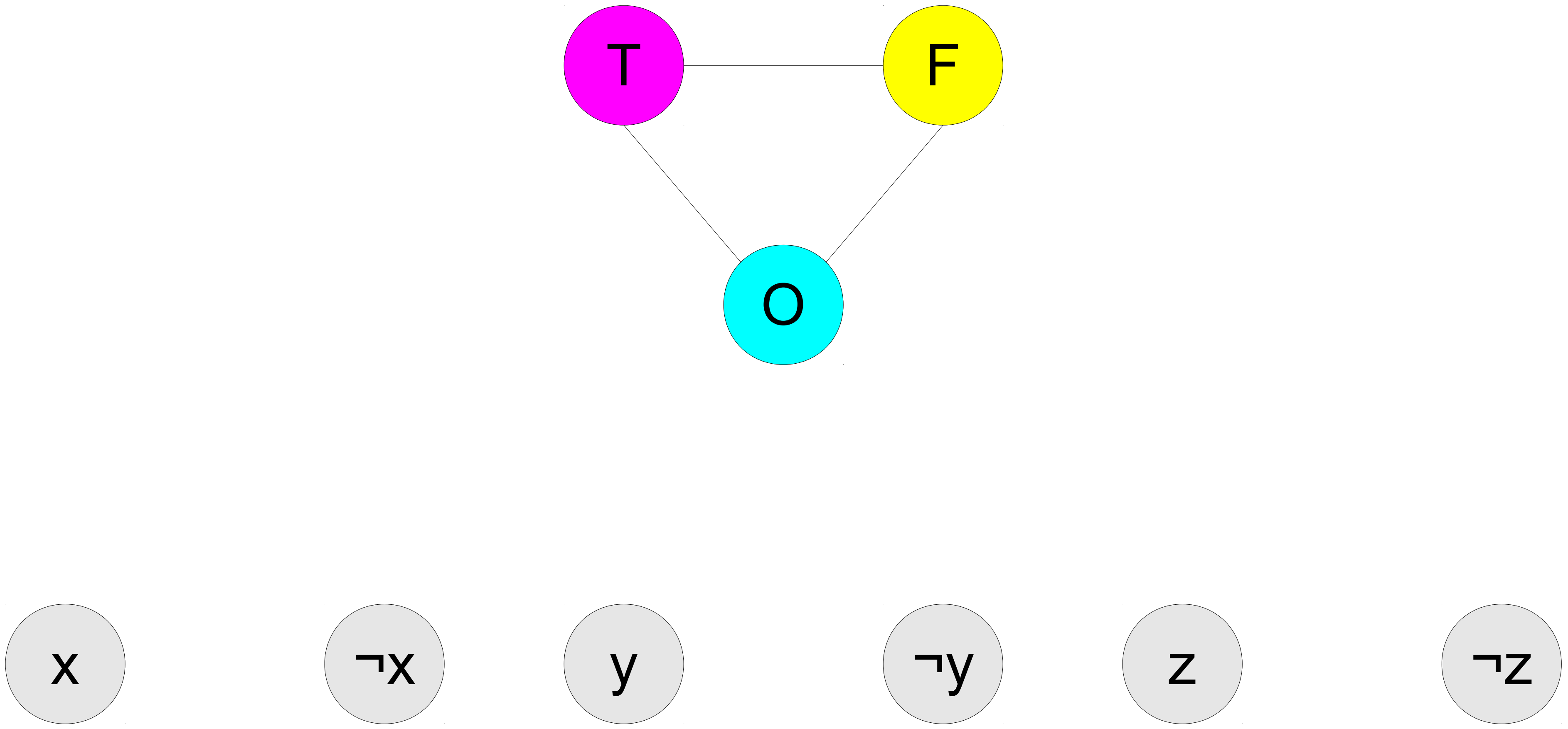
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



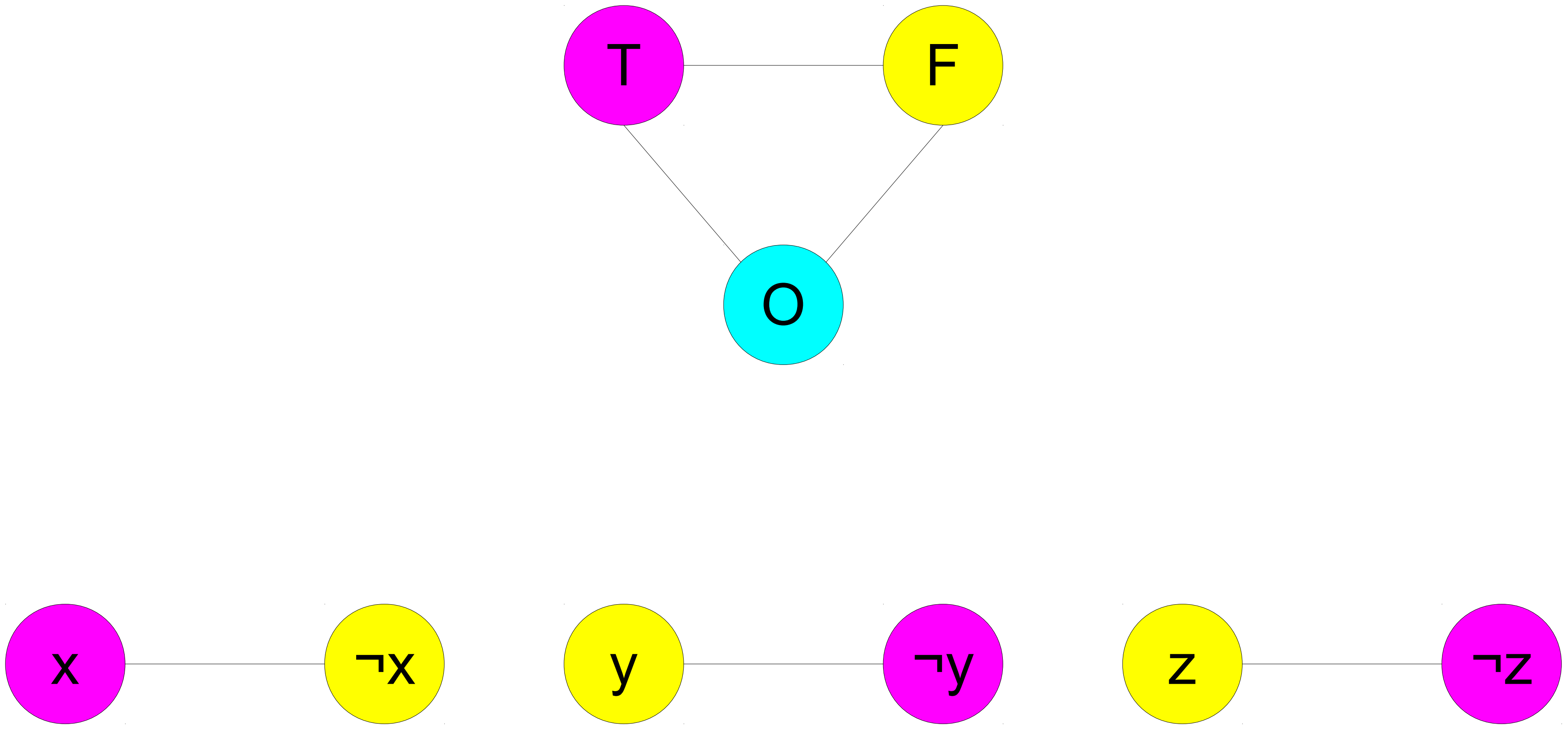
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



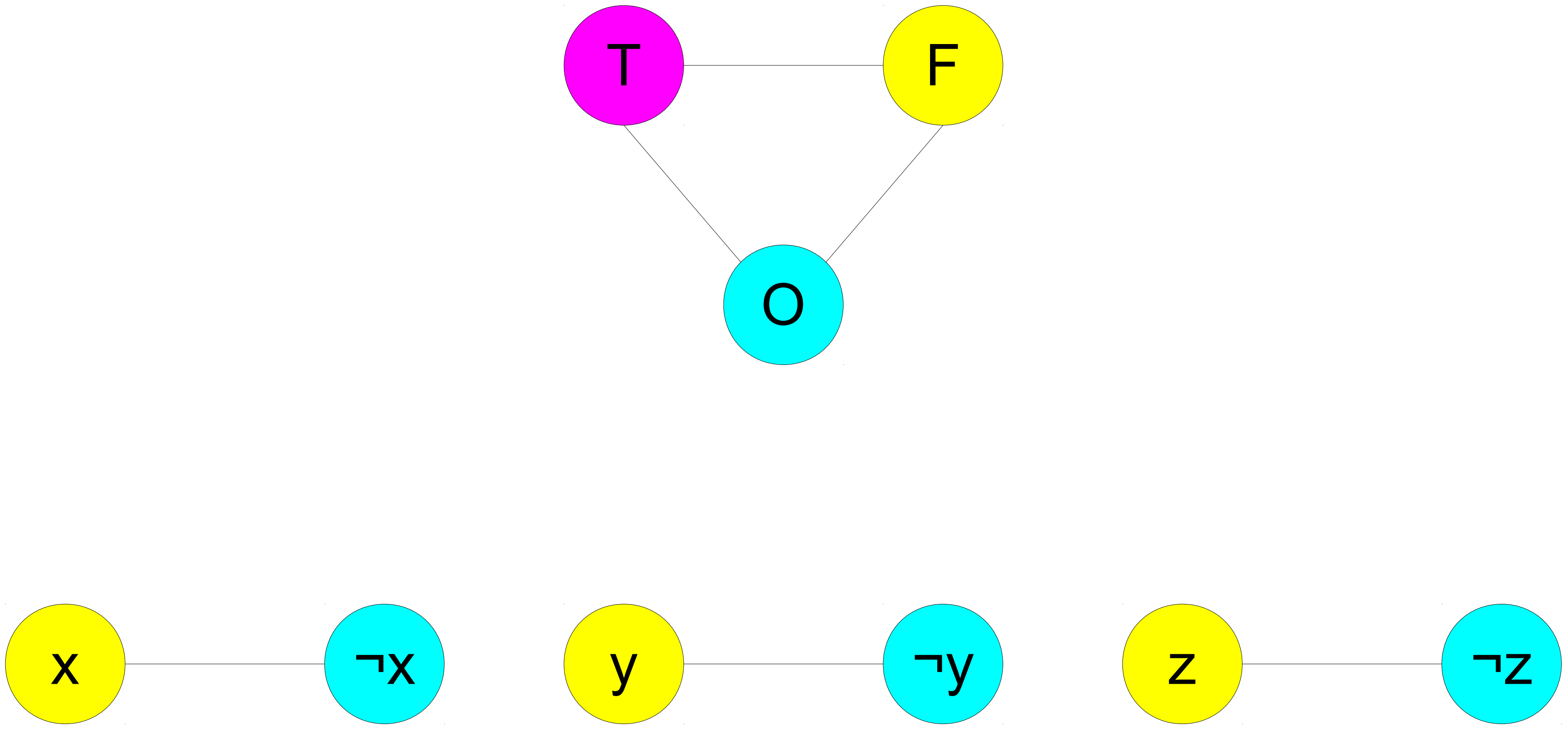
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



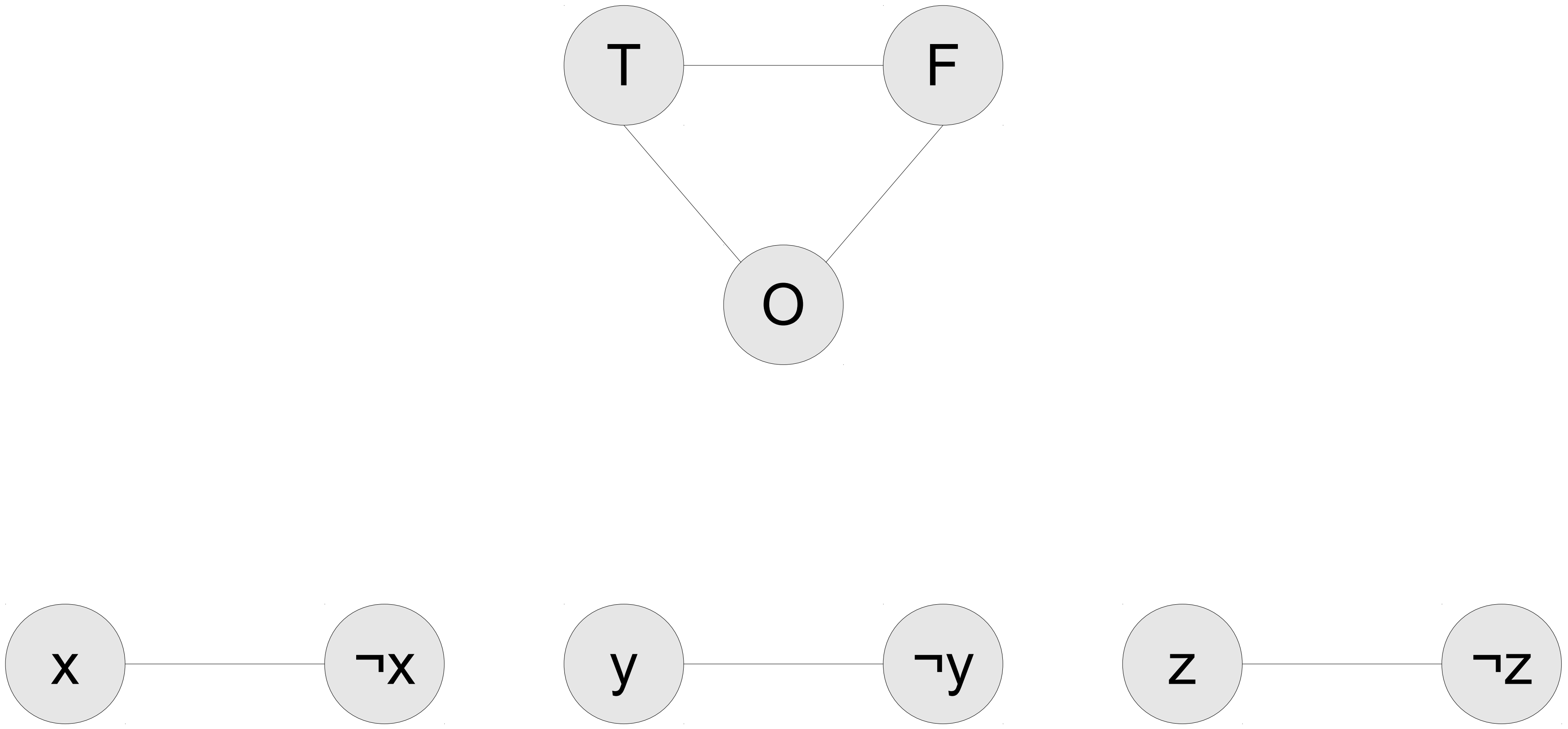
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



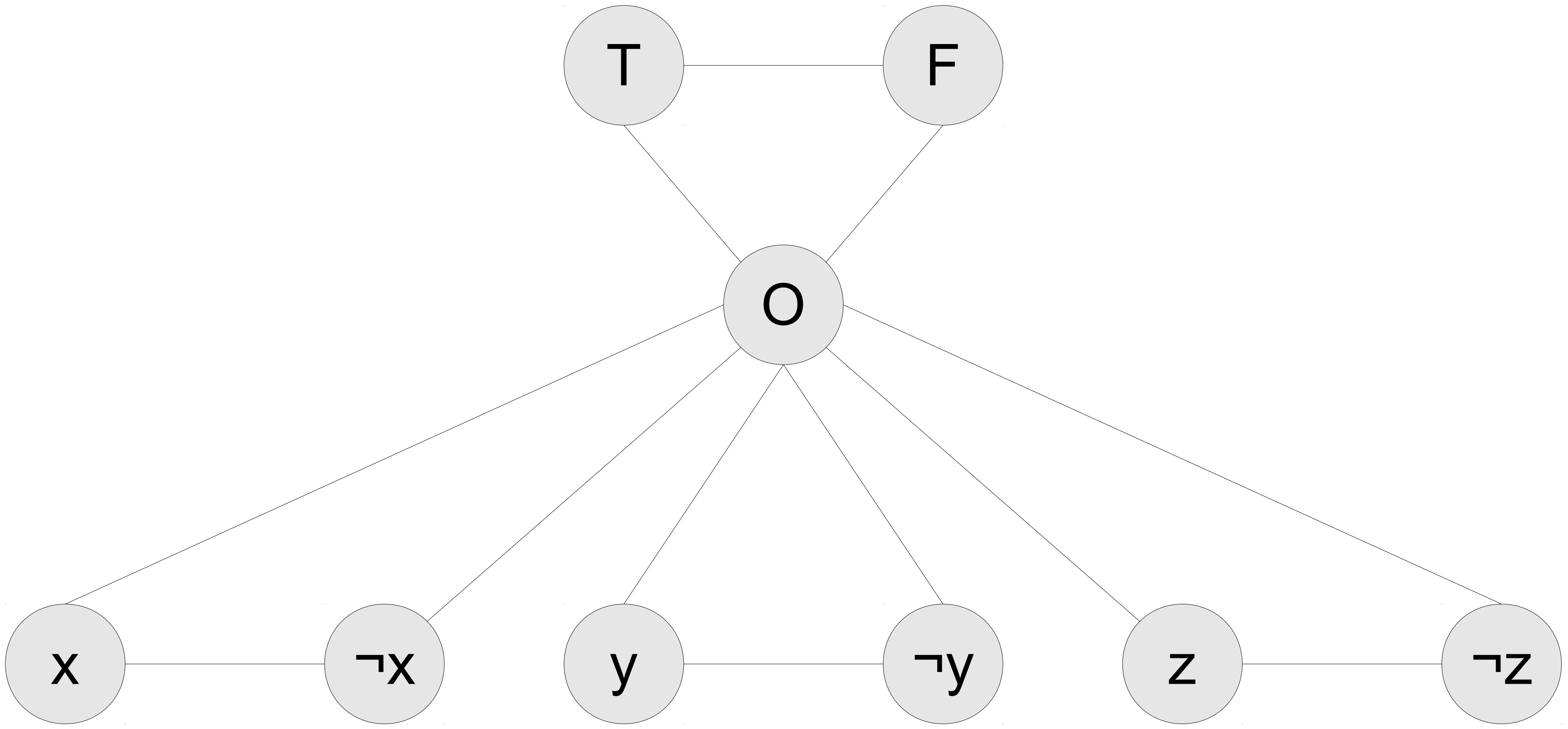
Gadget Two: Forcing a Choice

(x ∨ y ∨ ¬z) ∧ (¬x ∨ ¬y ∨ z) ∧ (¬x ∨ y ∨ ¬z)



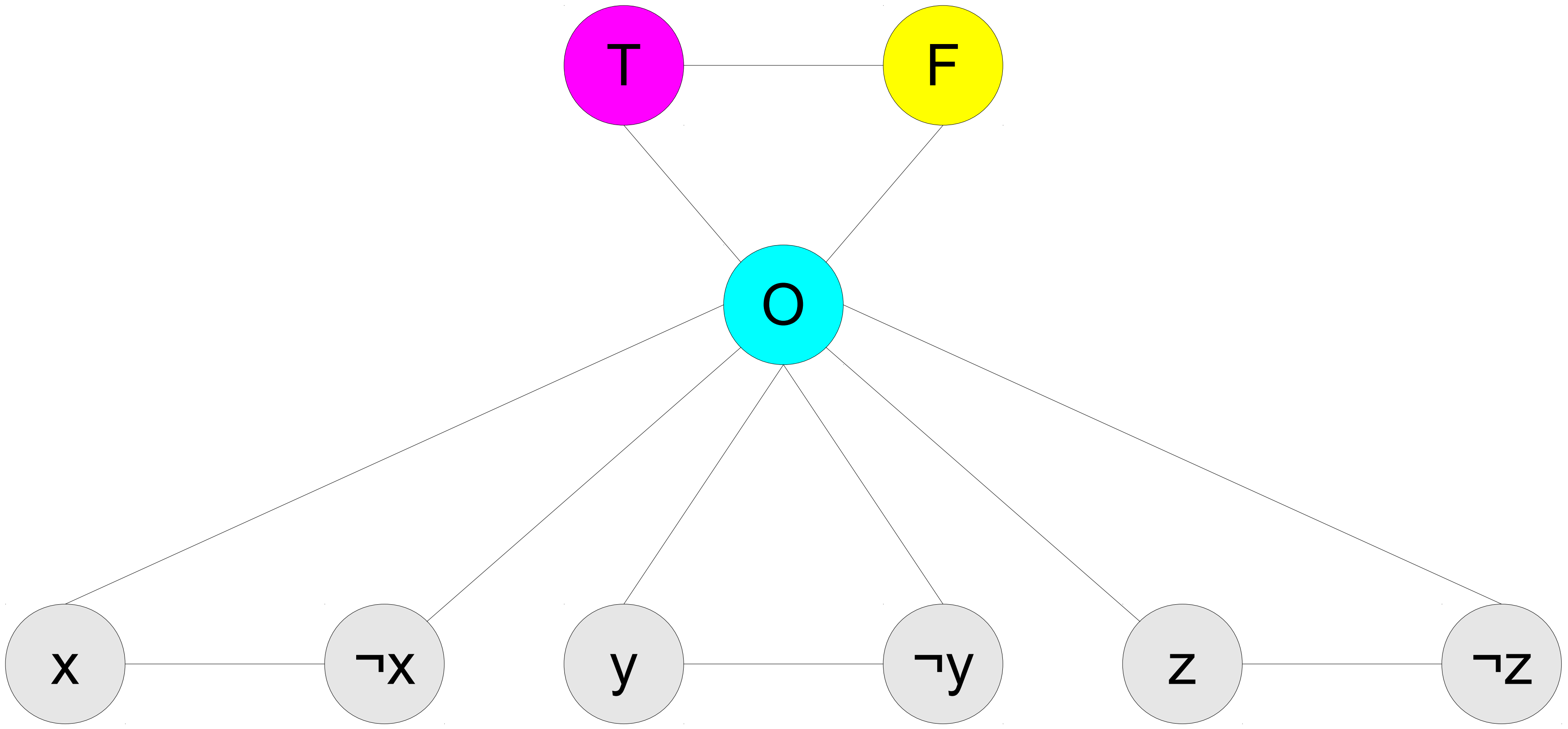
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



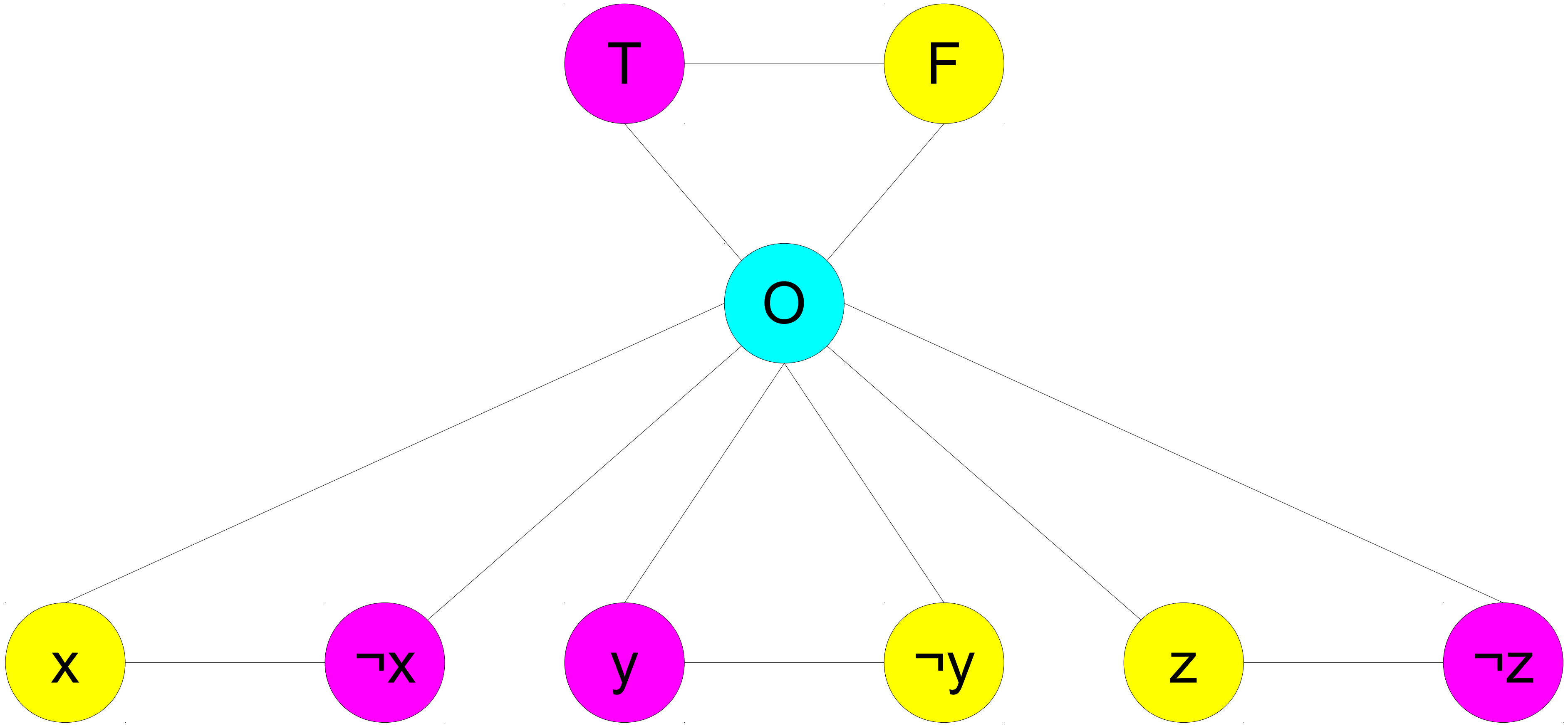
Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$



Gadget Two: Forcing a Choice

$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$

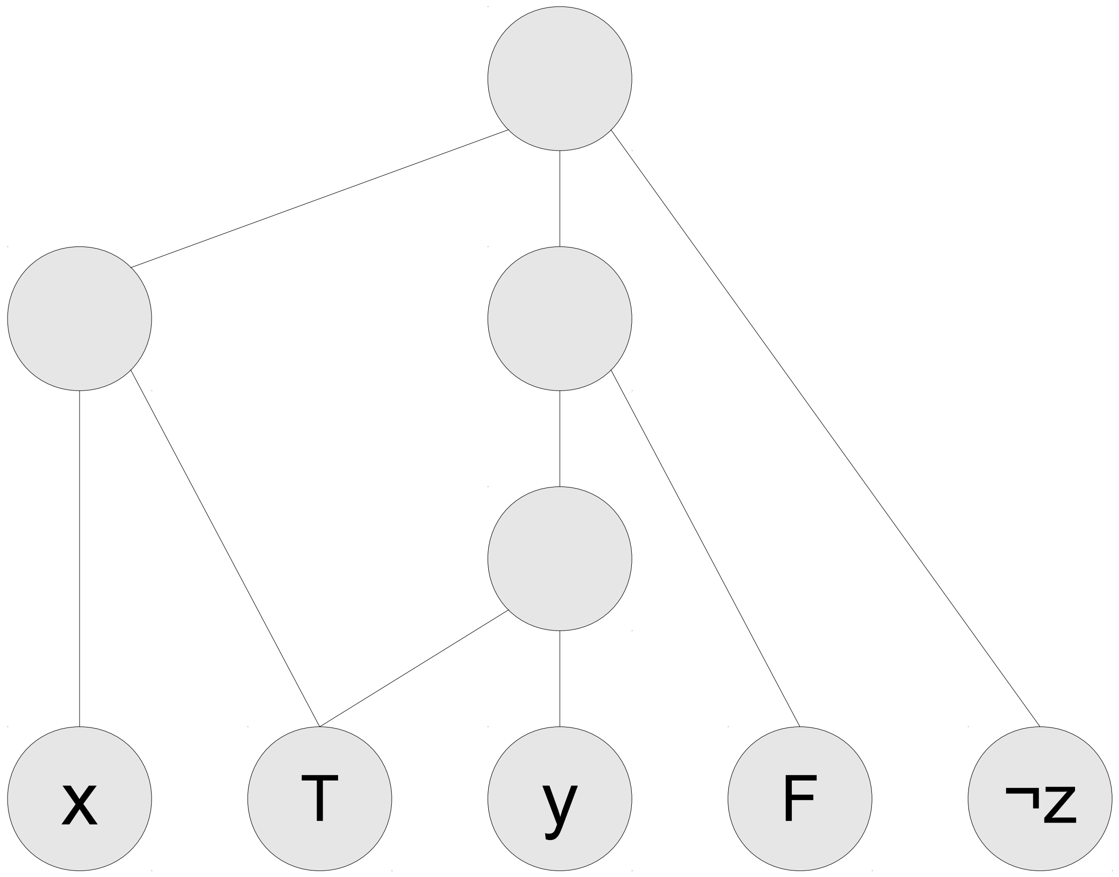


Gadget Three: Clause Satisfiability

$$(x \lor y \lor \neg z)$$

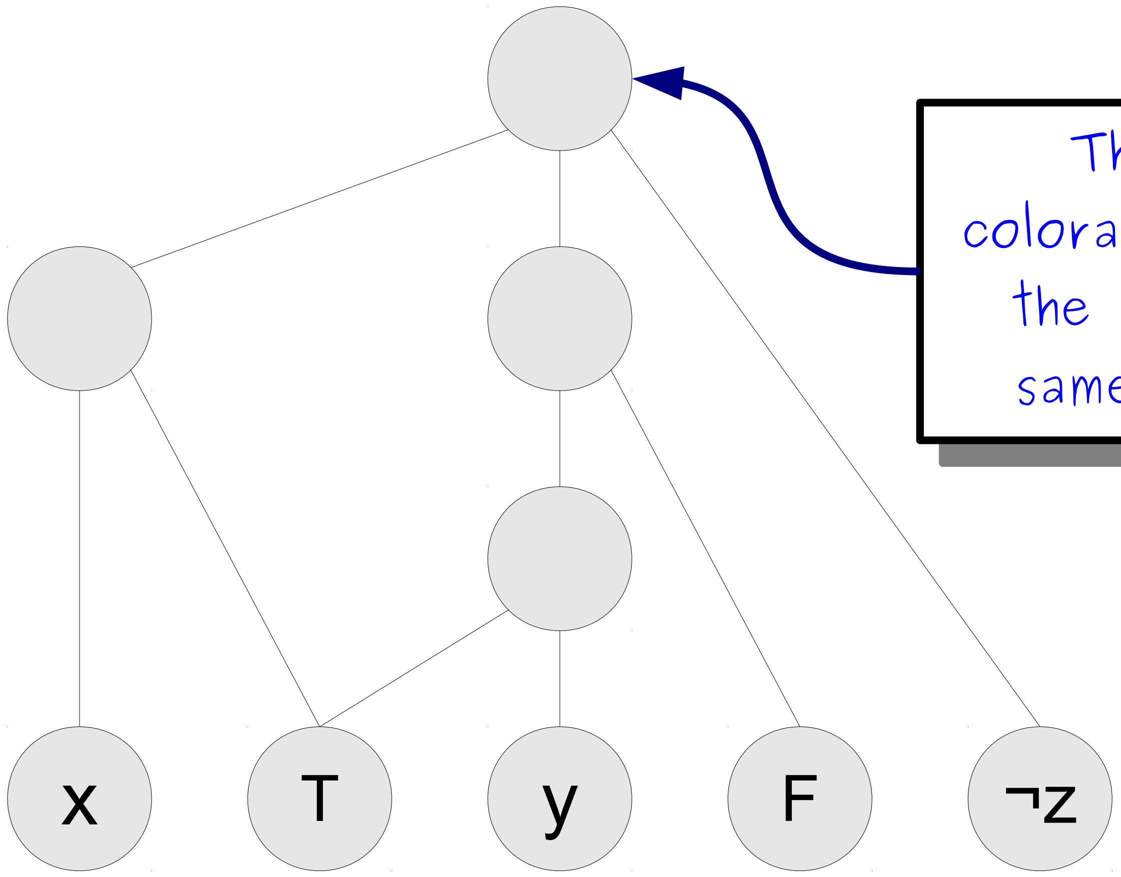
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



Gadget Three: Clause Satisfiability

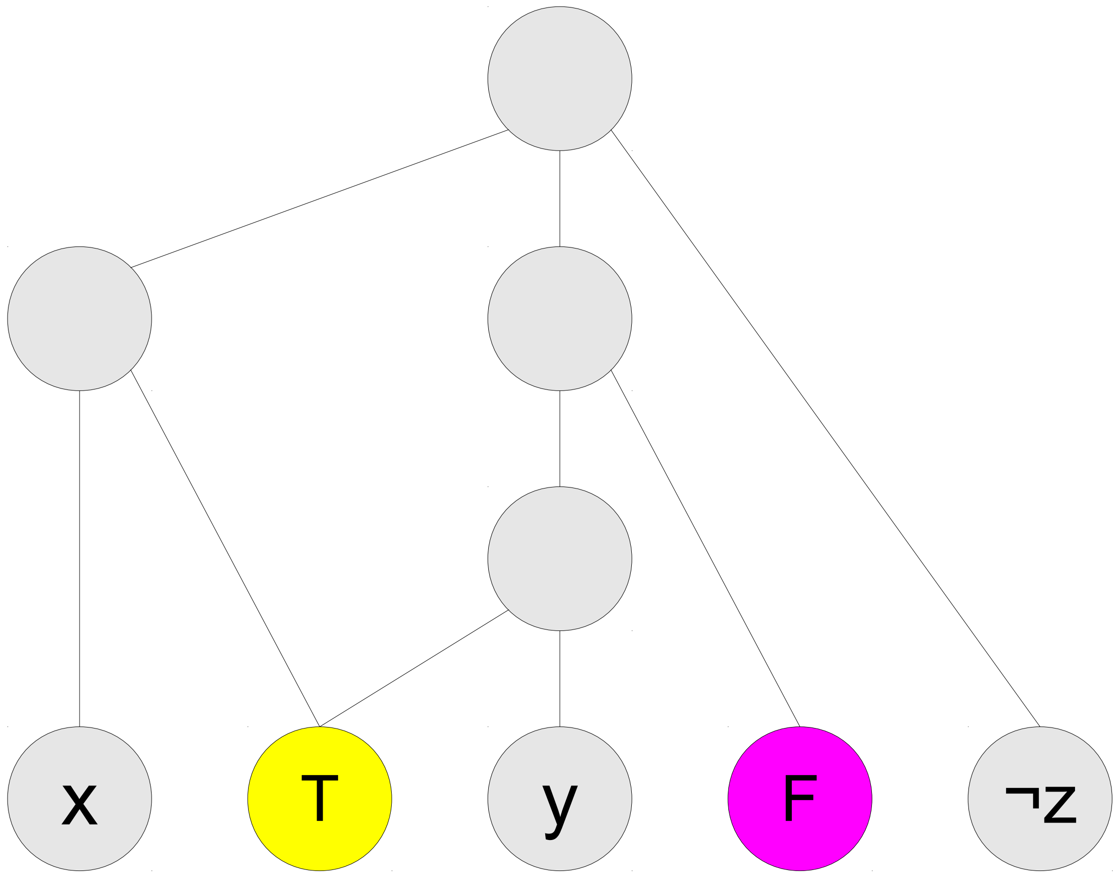
$(x \lor y \lor \neg z)$



This node is colorable iff one of the inputs is the same color as T

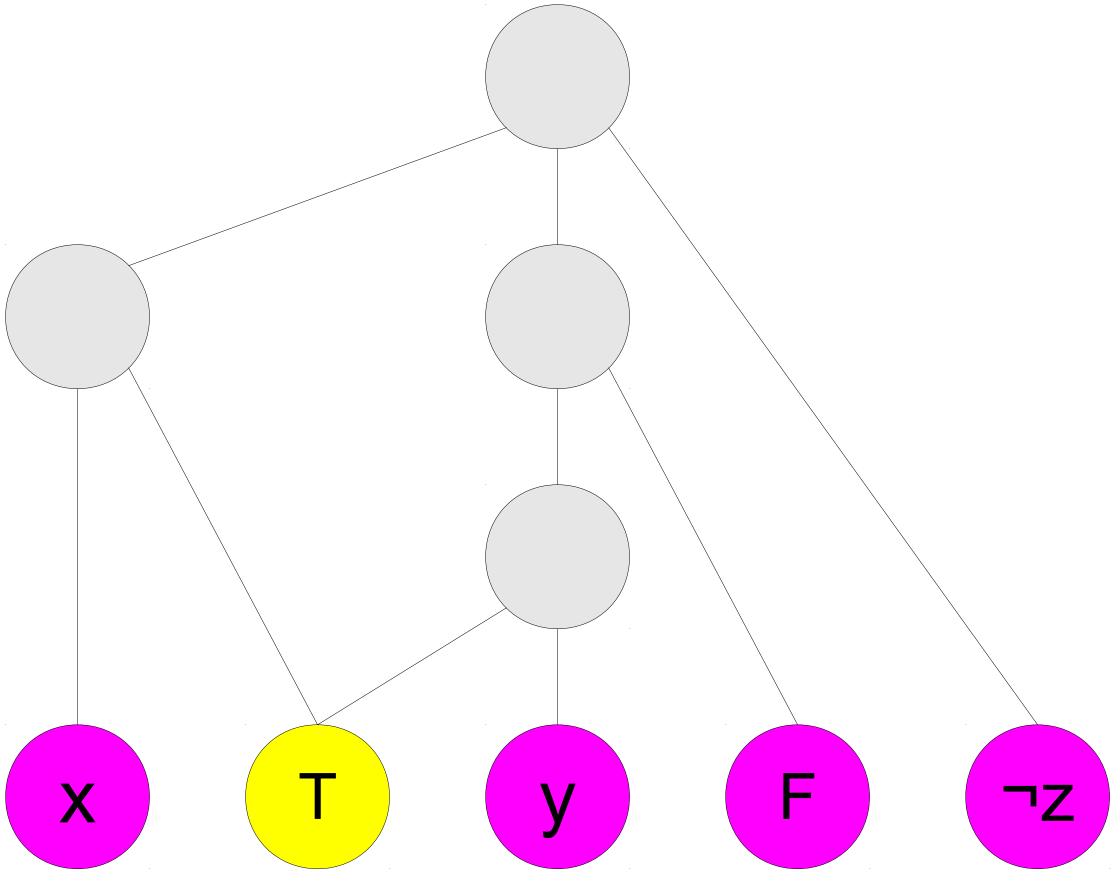
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



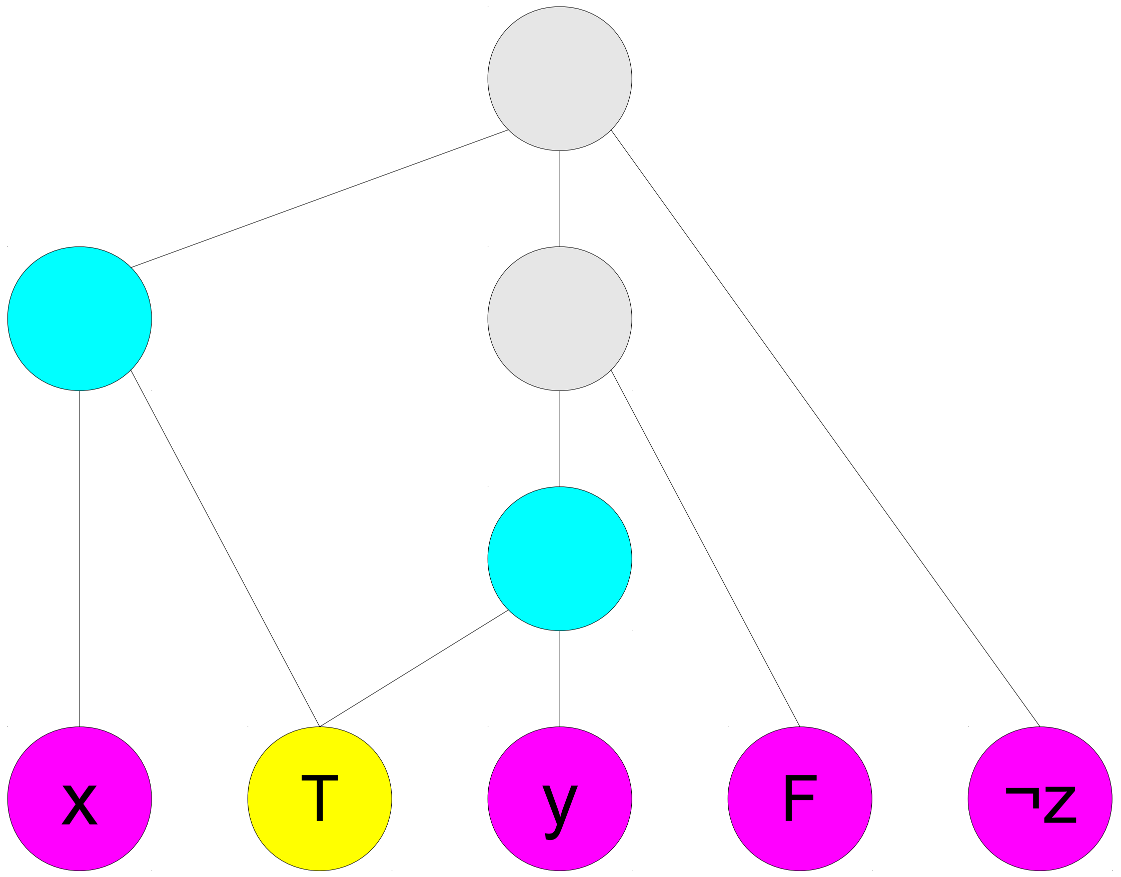
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



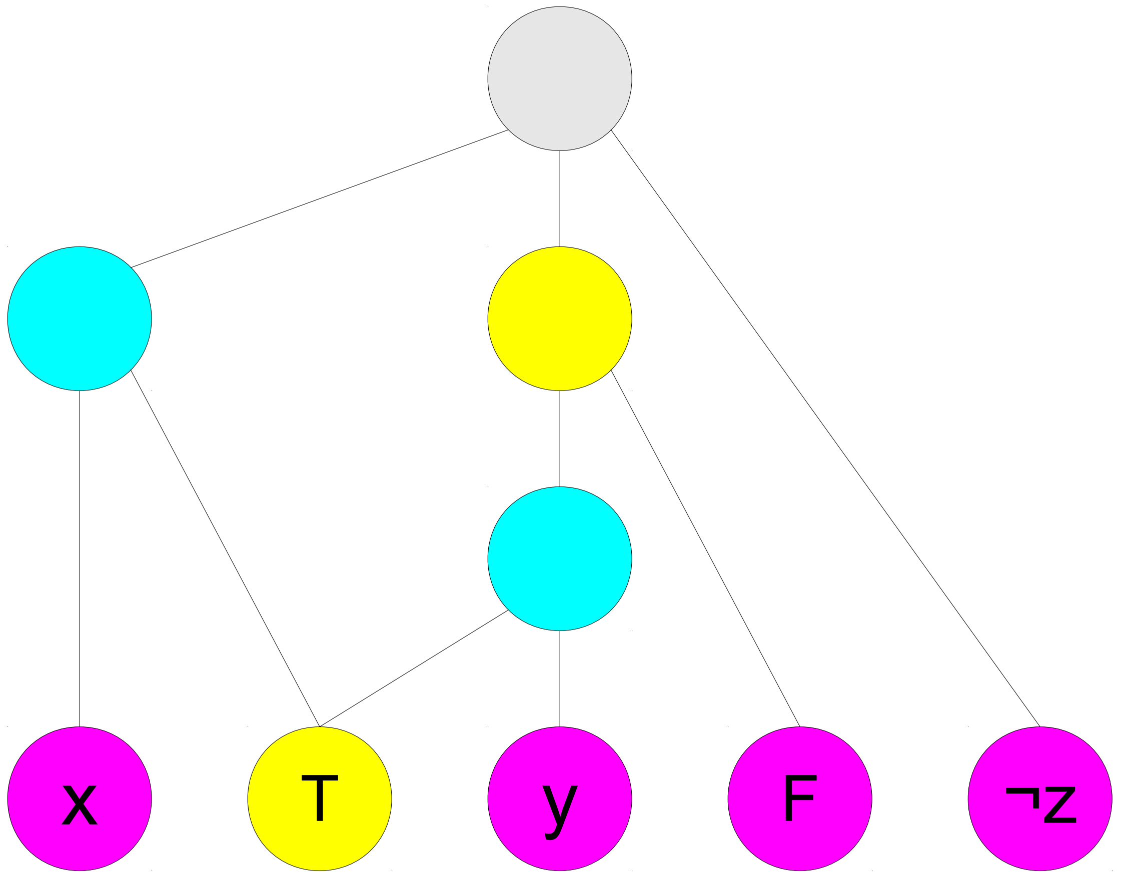
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



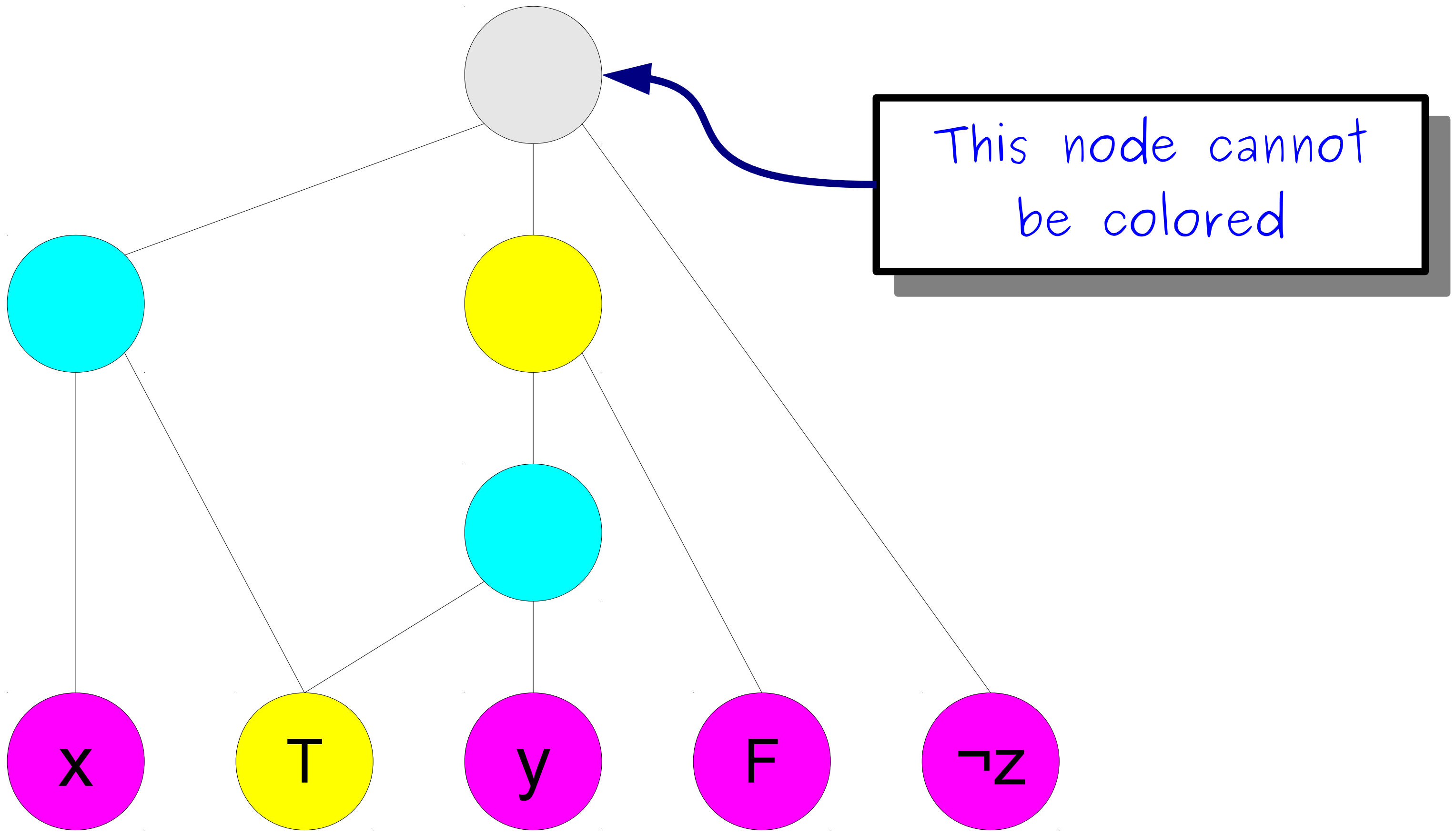
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



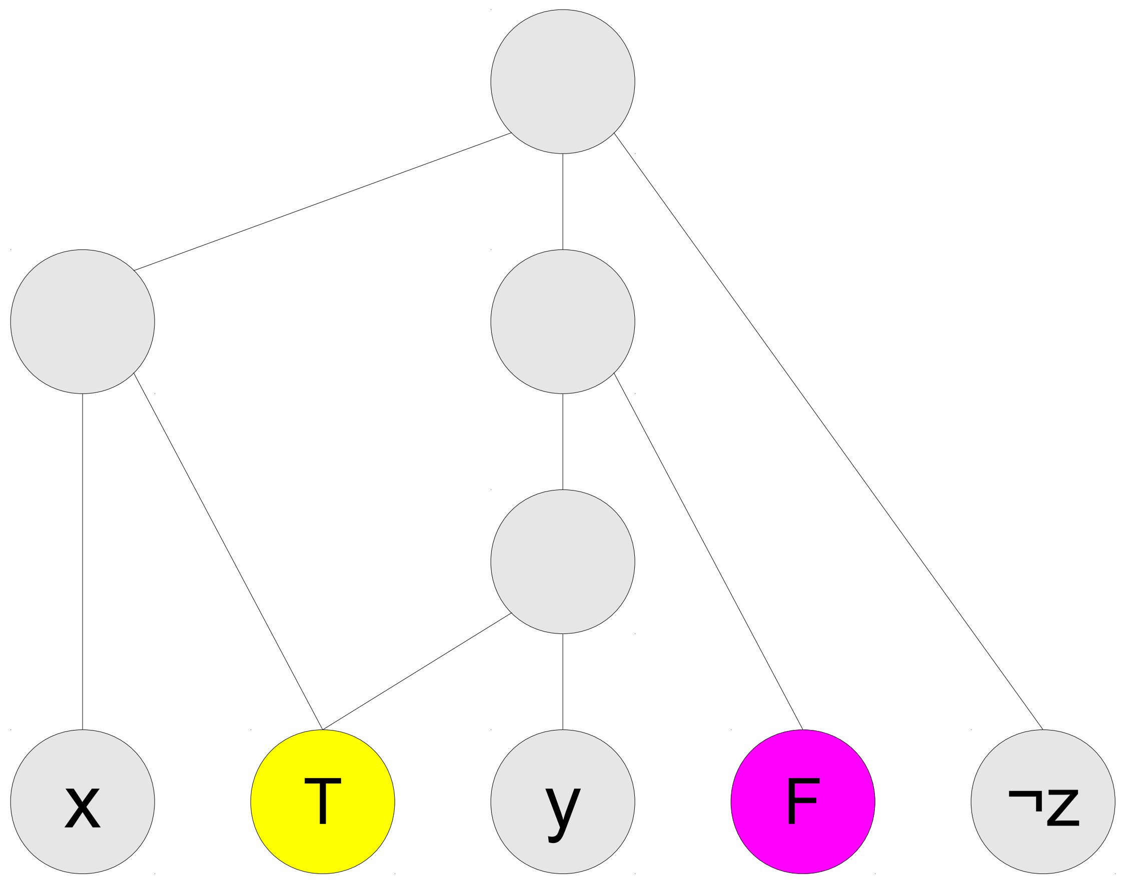
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



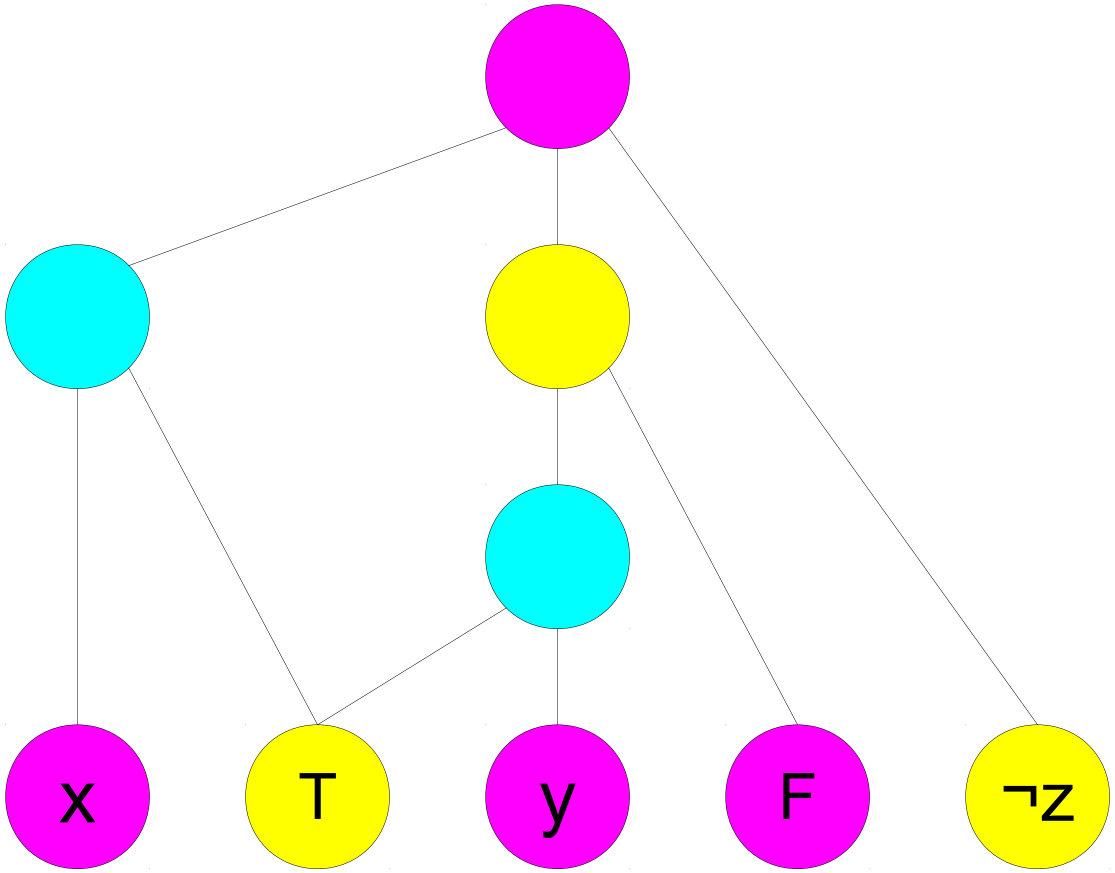
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)

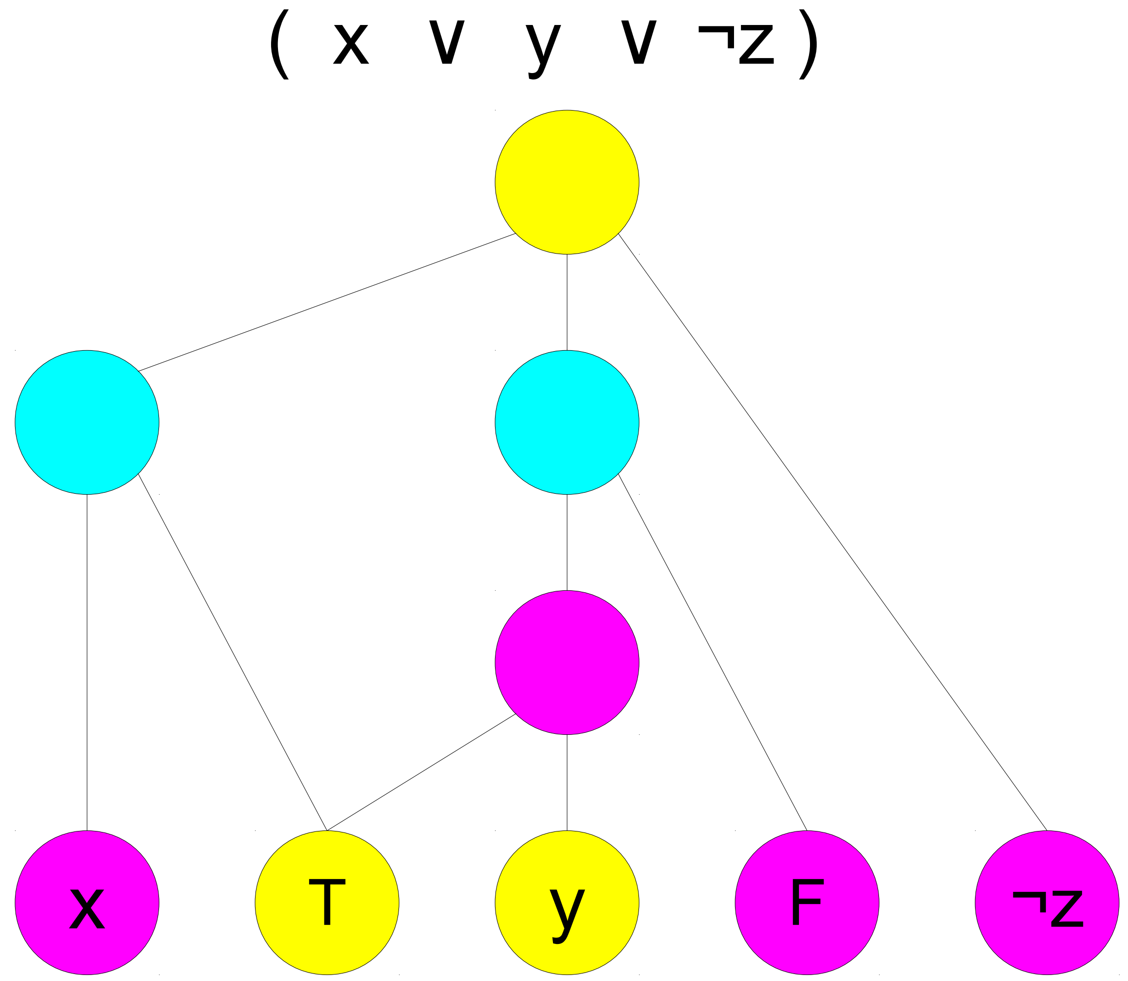


Gadget Three: Clause Satisfiability

$(x \lor y \lor \neg z)$

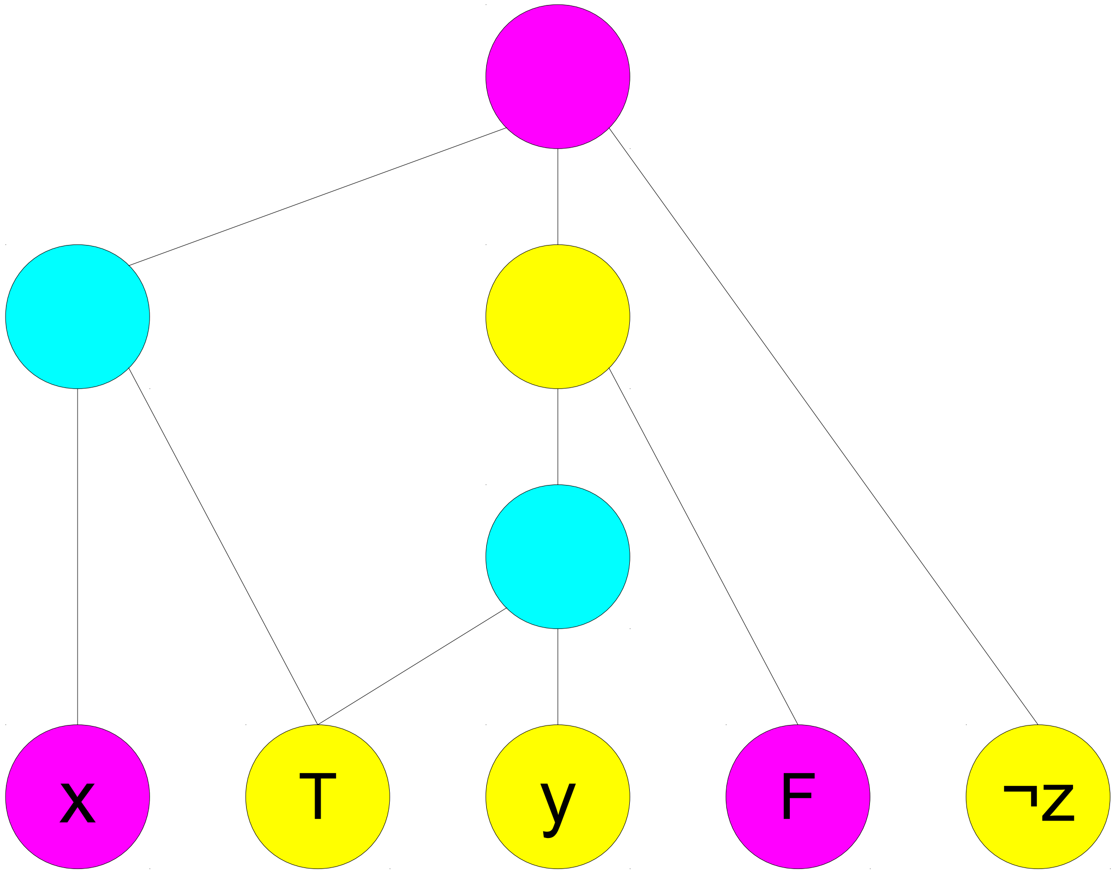


Gadget Three: Clause Satisfiability

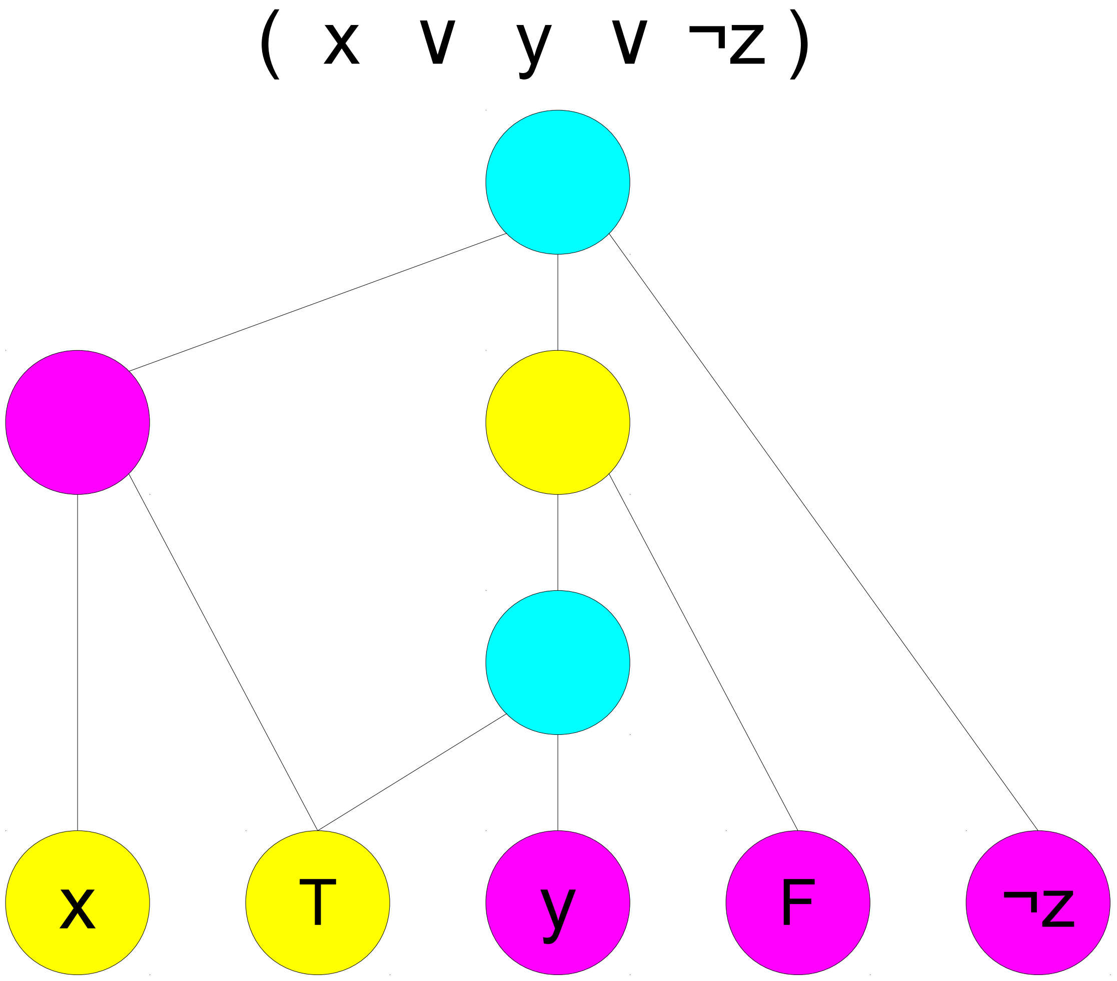


Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)

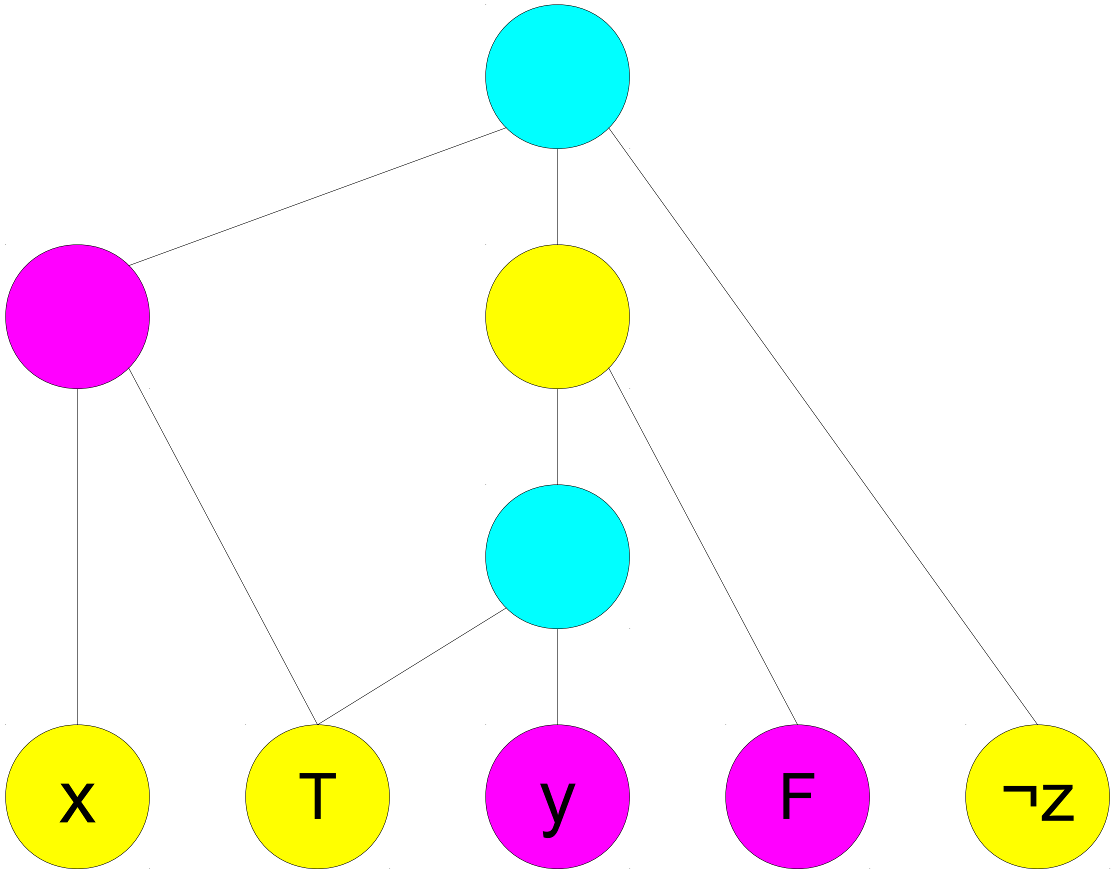


Gadget Three: Clause Satisfiability



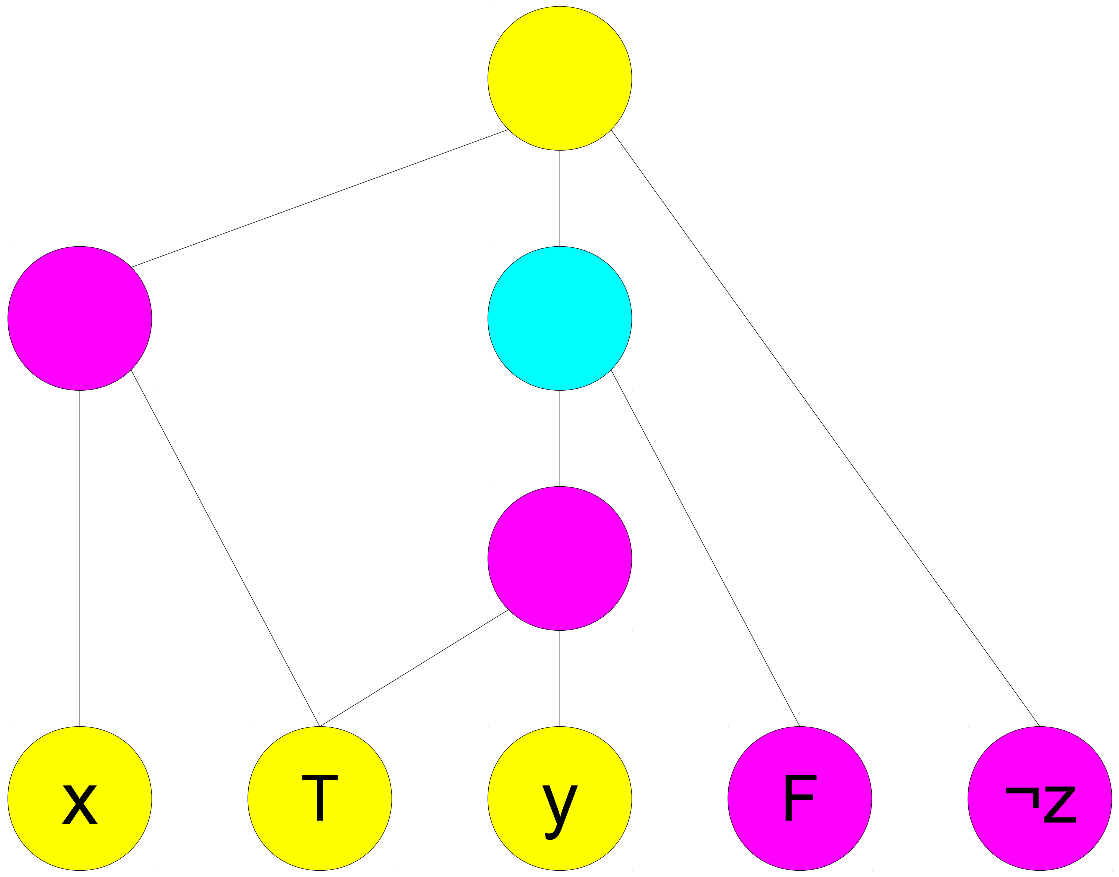
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



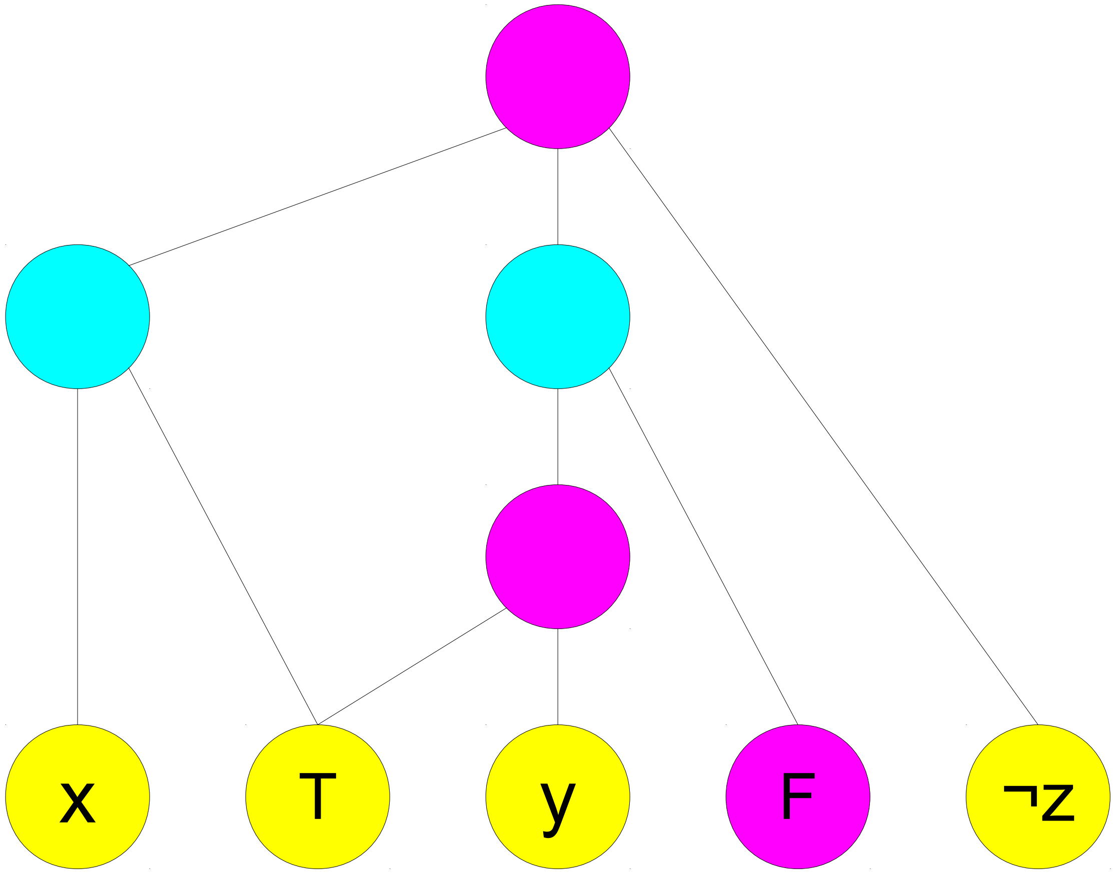
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



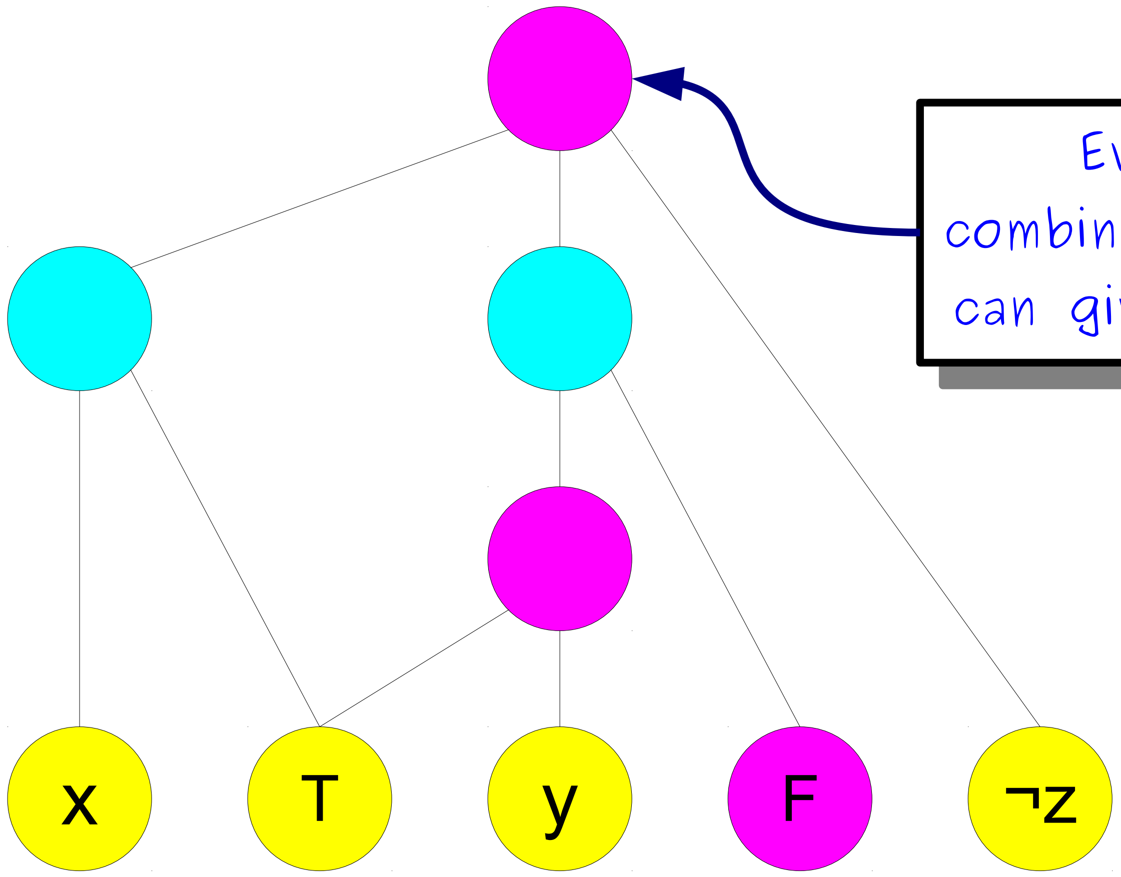
Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)



Gadget Three: Clause Satisfiability

(x ∨ y ∨ ¬z)

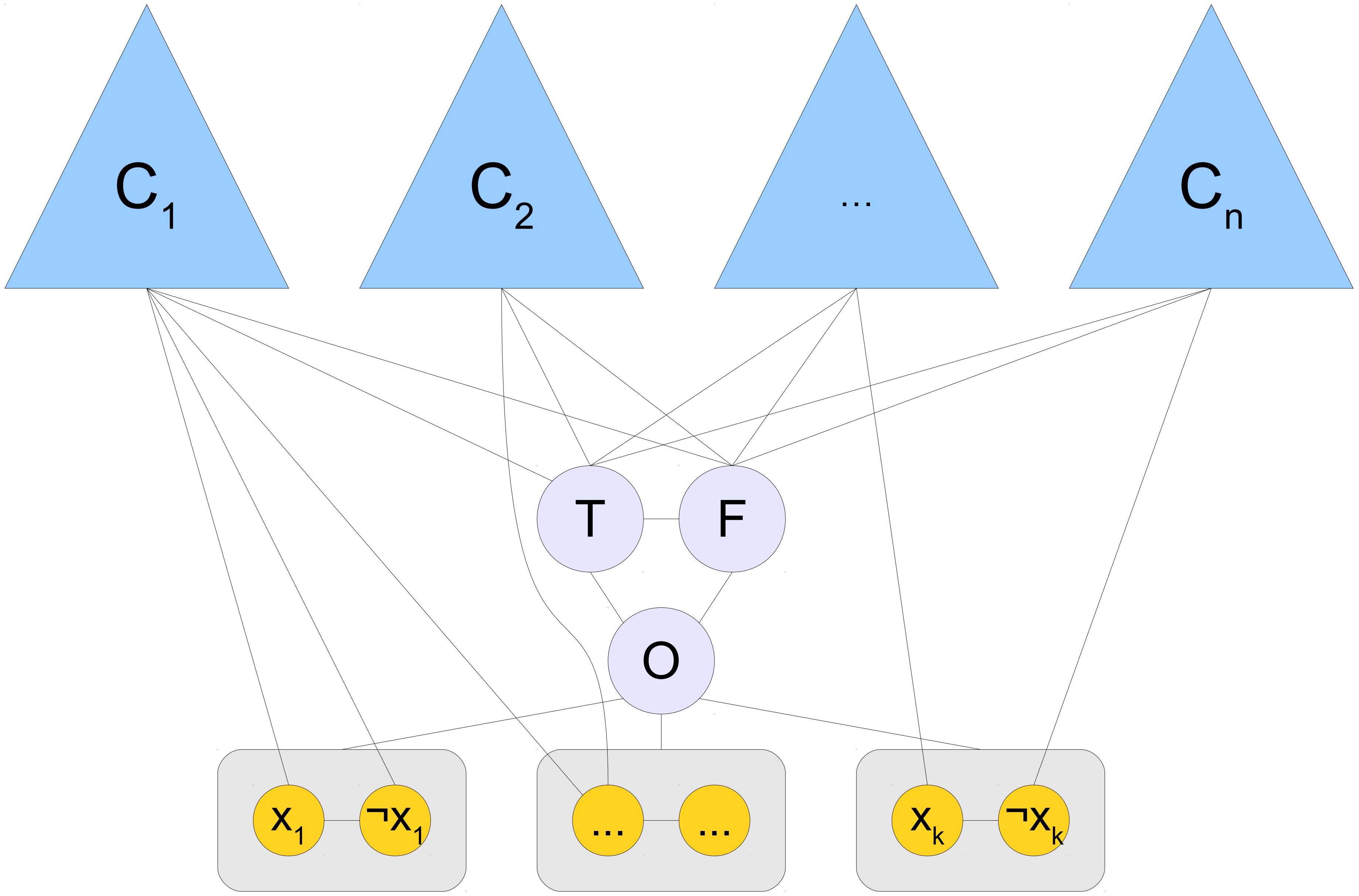


Every other combination of inputs can give this a color

Putting It All Together

- Construct the first gadget so we have a consistent definition of true and false.
- For each variable *v*:
    - Construct nodes *v* and ¬*v*.
    - Add an edge between *v* and ¬*v*.
    - Add an edge between *v* and O and between ¬*v* and O.
- For each clause *C*:
    - Construct the earlier gadget from *C* by adding in the extra nodes and edges.

Putting It All Together



Next Time

- **The Big Picture**
  - How do all of our results relate to one another?
- **Where to Go from Here**
  - What's next in CS theory?