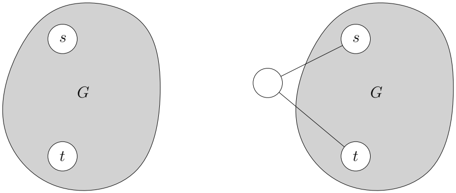
# Polynomial-Time Reduction: Example

## Hamiltonian-Path (HP) problem

**Input:** $G = (V, E)$ and $s, t \in V$

**Output:** whether there is a Hamiltonian path from $s$ to $t$ in $G$

**Lemma** HP $\leq_{\mathsf{P}}$ HC.




**Obs.** $G$ has a HP from $s$ to $t$ if and only if graph on right side has a HC.

# NP-Completeness

**Def.** A problem $X$ is called NP-complete if

1. $X \in \mathrm{NP}$, and
2. $Y \leq_{\mathrm{P}} X$ for every $Y \in \mathrm{NP}$.

# NP-Completeness

**Def.** A problem $X$ is called NP-hard if

2. $Y \leq_{\mathrm{P}} X$ for every $Y \in \mathrm{NP}$.

- NP-hard problems are at least as hard as NP-complete problems (a NP-hard problem is not required to be in NP)

# NP-Completeness

**Def.** A problem $X$ is called NP-complete if

1. $X \in \text{NP}$, and
2. $Y \leq_{\text{P}} X$ for every $Y \in \text{NP}$.

- NP-hard problems are at least as hard as NP-complete problems (a NP-hard problem is not required to be in NP)

# NP-Completeness

**Def.** A problem $X$ is called NP-complete if

1. $X \in \text{NP}$, and
2. $Y \leq_{\text{P}} X$ for every $Y \in \text{NP}$.

**Theorem** If $X$ is NP-complete and $X \in \text{P}$, then $\text{P} = \text{NP}$.

- NP-hard problems are at least as hard as NP-complete problems (a NP-hard problem is not required to be in NP)

# NP-Completeness

**Def.** A problem $X$ is called NP-complete if

1. $X \in \mathrm{NP}$, and
2. $Y \leq_{\mathrm{P}} X$ for every $Y \in \mathrm{NP}$.

**Theorem** If $X$ is NP-complete and $X \in \mathrm{P}$, then $\mathrm{P} = \mathrm{NP}$.

- NP-complete problems are the hardest problems in NP
- NP-hard problems are at least as hard as NP-complete problems (a NP-hard problem is not required to be in NP)

# Outline

**Def.** A problem $X$ is called NP-complete if

1. $X \in \mathsf{NP}$, and
2. $Y \leq_{\mathsf{P}} X$ for every $Y \in \mathsf{NP}$.

**Def.** A problem $X$ is called NP-complete if

1. $X \in$ NP, and
2. $Y \leq_{\mathsf{P}} X$ for every $Y \in$ NP.

- How can we find a problem $X \in$ NP such that every problem $Y \in$ NP is polynomial time reducible to $X$? Are we asking for too much?

**Def.** A problem $X$ is called NP-complete if

1. $X \in$ NP, and
2. $Y \leq_{\mathrm{P}} X$ for every $Y \in$ NP.

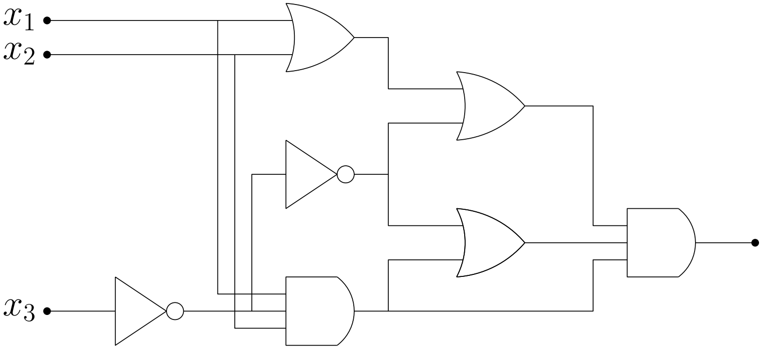
- How can we find a problem $X \in$ NP such that every problem $Y \in$ NP is polynomial time reducible to $X$? Are we asking for too much?
- No! There is indeed a large family of natural NP-complete problems

# The First NP-Complete Problem: Circuit-Sat

## Circuit Satisfiability (Circuit-Sat)

**Input:** a circuit

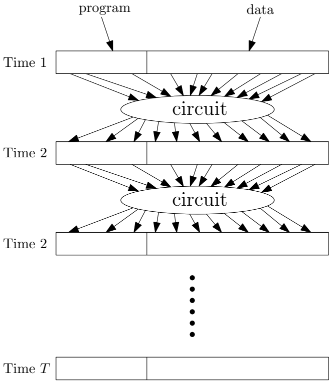
**Output:** whether the circuit is satisfiable



# Circuit-Sat is NP-Complete

- key fact: algorithms can be converted to circuits

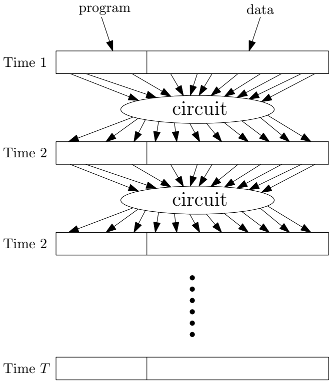
**Fact** Any algorithm that takes $n$ bits as input and outputs $0/1$ with running time $T(n)$ can be converted into a circuit of size $p(T(n))$ for some polynomial function $p(\cdot)$.



# Circuit-Sat is NP-Complete

- key fact: algorithms can be converted to circuits

**Fact** Any algorithm that takes $n$ bits as input and outputs $0/1$ with running time $T(n)$ can be converted into a circuit of size $p(T(n))$ for some polynomial function $p(\cdot)$.




- Then, we can show that any problem $Y \in$ NP can be reduced to Circuit-Sat.
- We prove HC $\leq_P$ Circuit-Sat as an example.

# HC $\leq_P$ Circuit-Sat

check-HC$(G, S)$

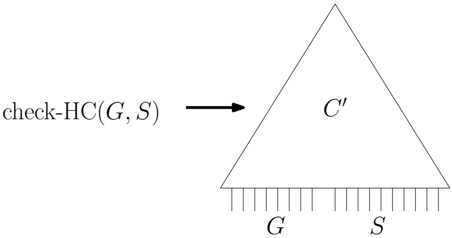
- Let check-HC$(G, S)$ be the certifier for the Hamiltonian cycle problem: check-HC$(G, S)$ returns 1 if $S$ is a Hamiltonian cycle is $G$ and $0$ otherwise.

# HC $\leq_P$ Circuit-Sat

check-HC$(G, S)$

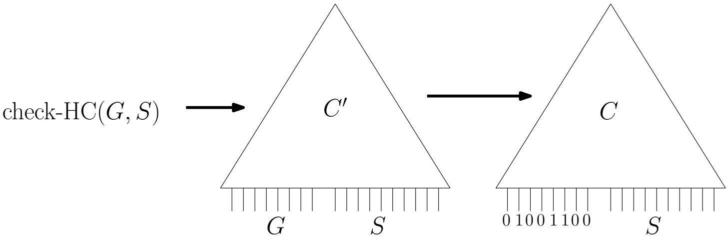
- Let check-HC$(G, S)$ be the certifier for the Hamiltonian cycle problem: check-HC$(G, S)$ returns 1 if $S$ is a Hamiltonian cycle is $G$ and $0$ otherwise.
- $G$ is a yes-instance if and only if there is an $S$ such that check-HC$(G, S)$ returns 1

# HC $\leq_P$ Circuit-Sat



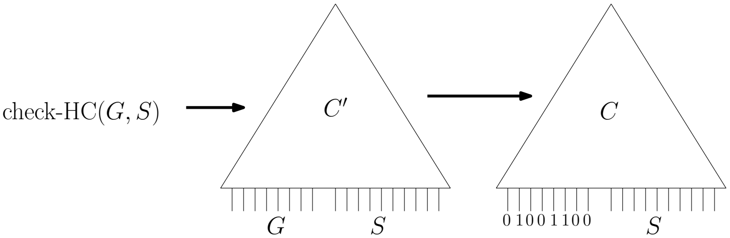

- Let check-HC$(G, S)$ be the certifier for the Hamiltonian cycle problem: check-HC$(G, S)$ returns 1 if $S$ is a Hamiltonian cycle is $G$ and $0$ otherwise.
- $G$ is a yes-instance if and only if there is an $S$ such that check-HC$(G, S)$ returns 1
- Construct a circuit $C'$ for the algorithm check-HC

# HC $\leq_P$ Circuit-Sat




- Let check-HC$(G, S)$ be the certifier for the Hamiltonian cycle problem: check-HC$(G, S)$ returns 1 if $S$ is a Hamiltonian cycle is $G$ and $0$ otherwise.
- $G$ is a yes-instance if and only if there is an $S$ such that check-HC$(G, S)$ returns 1
- Construct a circuit $C'$ for the algorithm check-HC
- hard-wire the instance $G$ to the circuit $C'$ to obtain the circuit $C$

# HC $\leq_P$ Circuit-Sat




- Let check-HC$(G, S)$ be the certifier for the Hamiltonian cycle problem: check-HC$(G, S)$ returns 1 if $S$ is a Hamiltonian cycle is $G$ and $0$ otherwise.
- $G$ is a yes-instance if and only if there is an $S$ such that check-HC$(G, S)$ returns 1
- Construct a circuit $C'$ for the algorithm check-HC
- hard-wire the instance $G$ to the circuit $C'$ to obtain the circuit $C$
- $G$ is a yes-instance if and only if $C$ is satisfiable

# $Y \leq_P$ Circuit-Sat, For Every $Y \in$NP

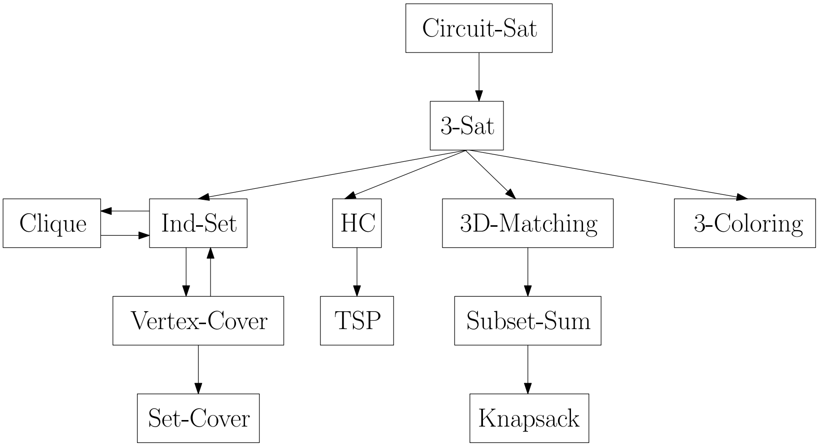
- Let check-Y$(s,t)$ be the certifier for problem $Y$: check-Y$(s,t)$ returns 1 if $t$ is a valid certificate for $s$.
- $s$ is a yes-instance if and only if there is a $t$ such that check-Y$(s,t)$ returns 1
- Construct a circuit $C'$ for the algorithm check-Y
- hard-wire the instance $s$ to the circuit $C'$ to obtain the circuit $C$
- $s$ is a yes-instance if and only if $C$ is satisfiable $\square$

# $Y \leq_P$ Circuit-Sat, For Every $Y \in$NP

- Let check-Y$(s,t)$ be the certifier for problem $Y$: check-Y$(s,t)$ returns 1 if $t$ is a valid certificate for $s$.
- $s$ is a yes-instance if and only if there is a $t$ such that check-Y$(s,t)$ returns 1
- Construct a circuit $C'$ for the algorithm check-Y
- hard-wire the instance $s$ to the circuit $C'$ to obtain the circuit $C$
- $s$ is a yes-instance if and only if $C$ is satisfiable □

**Theorem** Circuit-Sat is NP-complete.

# Reductions of NP-Complete Problems



# 3-Sat

3-CNF (conjunctive normal form) is a special case of formula:

# 3-Sat

3-CNF (conjunctive normal form) is a special case of formula:

- Boolean variables: $x_1, x_2, \cdots, x_n$

# 3-Sat

3-CNF (conjunctive normal form) is a special case of formula:

- Boolean variables: $x_1, x_2, \cdots, x_n$
- Literals: $x_i$ or $\neg x_i$

# 3-Sat

3-CNF (conjunctive normal form) is a special case of formula:

- Boolean variables: $x_1, x_2, \cdots, x_n$
- Literals: $x_i$ or $\neg x_i$
- Clause: disjunction ("or") of at most $3$ literals: $x_3 \vee \neg x_4$, $x_1 \vee x_8 \vee \neg x_9$, $\neg x_2 \vee \neg x_5 \vee x_7$

# 3-Sat

3-CNF (conjunctive normal form) is a special case of formula:

- Boolean variables: $x_1, x_2, \cdots, x_n$
- Literals: $x_i$ or $\neg x_i$
- Clause: disjunction ("or") of at most $3$ literals: $x_3 \vee \neg x_4$, $x_1 \vee x_8 \vee \neg x_9$, $\neg x_2 \vee \neg x_5 \vee x_7$
- 3-CNF formula: conjunction ("and") of clauses: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$

# 3-Sat

**3-Sat**

**Input:** a 3-CNF formula

**Output:** whether the 3-CNF is satisfiable

# 3-Sat

**3-Sat**

**Input:** a 3-CNF formula

**Output:** whether the 3-CNF is satisfiable

- To satisfy a 3-CNF, we need to satisfy all clauses

# 3-Sat

**3-Sat**

**Input:** a 3-CNF formula

**Output:** whether the 3-CNF is satisfiable

- To satisfy a 3-CNF, we need to satisfy all clauses
- To satisfy a clause, we need to satisfy at least 1 literal

# 3-Sat

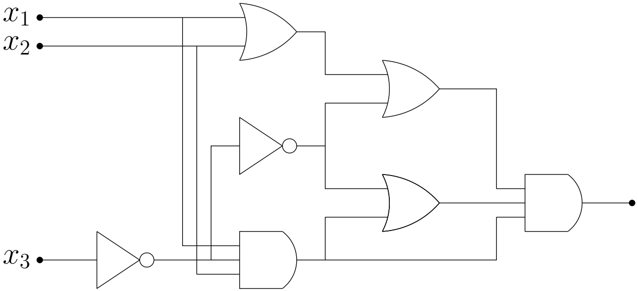
### 3-Sat

**Input:** a 3-CNF formula

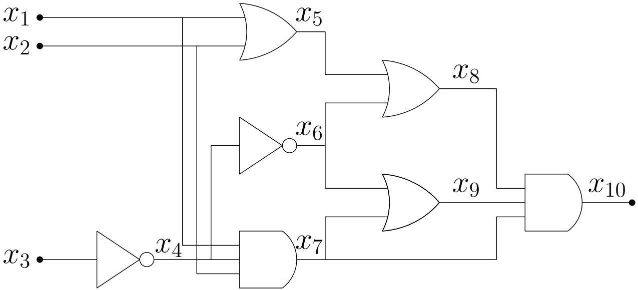
**Output:** whether the 3-CNF is satisfiable

- To satisfy a 3-CNF, we need to satisfy all clauses
- To satisfy a clause, we need to satisfy at least 1 literal
- Assignment $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$ satisfies $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$

# Circuit-Sat $\leq_P$ 3-Sat

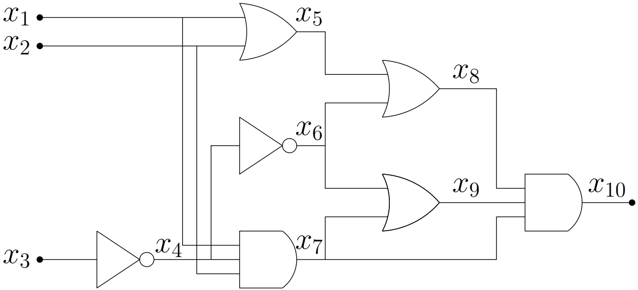


# Circuit-Sat $\leq_P$ 3-Sat




- Associate every wire with a new variable

# Circuit-Sat $\leq_P$ 3-Sat




- Associate every wire with a new variable
- The circuit is equivalent to the following formula:

$$
\begin{aligned}
&(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4) \\
&\wedge (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6) \\
&\wedge (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}
\end{aligned}
$$

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\ (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\ (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$

$$(x_1 \vee \neg x_2 \vee x_5) \quad \wedge$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$

$$(x_1 \vee \neg x_2 \vee x_5) \quad \wedge$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$

$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$

$(x_1 \vee \neg x_2 \vee x_5) \quad \wedge$

$(\neg x_1 \vee x_2 \vee x_5) \quad \wedge$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$
$$(x_1 \vee \neg x_2 \vee x_5) \quad \wedge$$
$$(\neg x_1 \vee x_2 \vee x_5) \quad \wedge$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

$$(x_4 = \neg x_3) \wedge (x_5 = x_1 \vee x_2) \wedge (x_6 = \neg x_4)$$
$$\wedge\, (x_7 = x_1 \wedge x_2 \wedge x_4) \wedge (x_8 = x_5 \vee x_6)$$
$$\wedge\, (x_9 = x_6 \vee x_7) \wedge (x_{10} = x_8 \wedge x_9 \wedge x_7) \wedge x_{10}$$

Convert each clause to a 3-CNF

$$x_5 = x_1 \vee x_2 \quad \Leftrightarrow$$

$$(x_1 \vee x_2 \vee \neg x_5) \quad \wedge$$
$$(x_1 \vee \neg x_2 \vee x_5) \quad \wedge$$
$$(\neg x_1 \vee x_2 \vee x_5) \quad \wedge$$
$$(\neg x_1 \vee \neg x_2 \vee x_5)$$

| $x_1$ | $x_2$ | $x_5$ | $x_5 \leftrightarrow x_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\Longleftrightarrow$ Formula $\Longleftrightarrow$ 3-CNF

# Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\Longleftrightarrow$ Formula $\Longleftrightarrow$ 3-CNF
- The circuit is satisfiable if and only if the 3-CNF is satisfiable

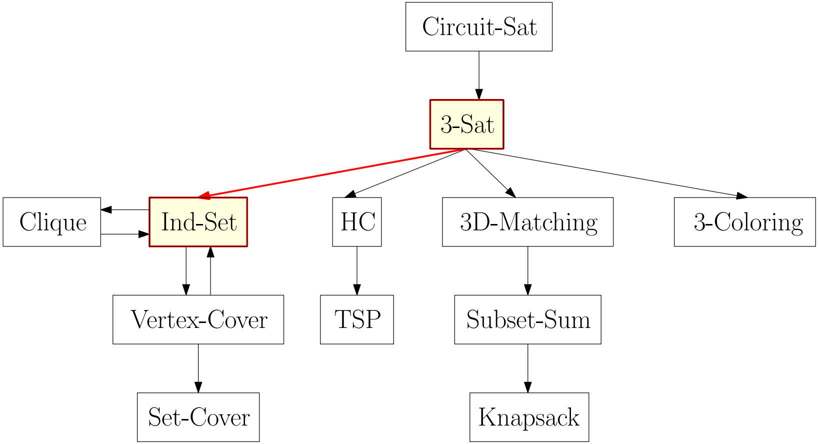
# Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\Longleftrightarrow$ Formula $\Longleftrightarrow$ 3-CNF
- The circuit is satisfiable if and only if the 3-CNF is satisfiable
- The size of the 3-CNF formula is polynomial (indeed, linear) in the size of the circuit

# Circuit-Sat $\leq_P$ 3-Sat

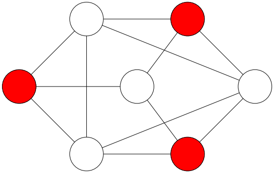
- Circuit $\Longleftrightarrow$ Formula $\Longleftrightarrow$ 3-CNF
- The circuit is satisfiable if and only if the 3-CNF is satisfiable
- The size of the 3-CNF formula is polynomial (indeed, linear) in the size of the circuit
- Thus, Circuit-Sat $\leq_P$ 3-Sat

# Reductions of NP-Complete Problems



# Recall: Independent Set Problem

**Def.** An independent set of $G = (V, E)$ is a subset $I \subseteq V$ such that no two vertices in $I$ are adjacent in $G$.



**Independent Set (Ind-Set) Problem**

**Input:** $G = (V, E), k$

**Output:** whether there is an independent set of size $k$ in $G$

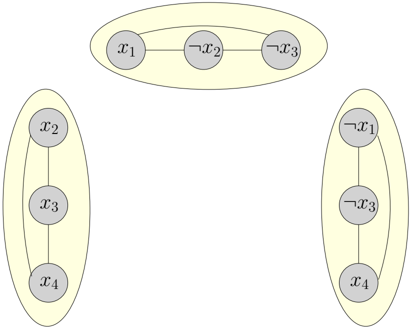
# 3-Sat $\leq_P$ Ind-Set

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

# 3-Sat $\leq_P$ Ind-Set

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

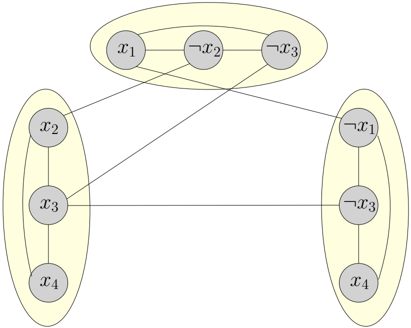
- A clause $\Rightarrow$ a group of $3$ vertices, one for each literal
- An edge between every pair of vertices in same group



# 3-Sat $\leq_P$ Ind-Set

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

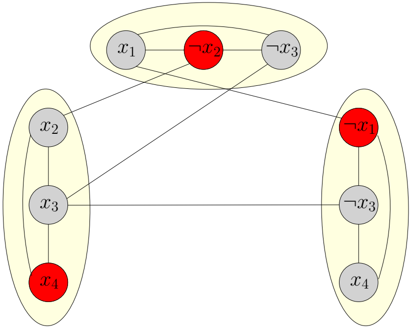
- A clause $\Rightarrow$ a group of $3$ vertices, one for each literal
- An edge between every pair of vertices in same group
- An edge between every pair of contradicting literals



# 3-Sat $\leq_P$ Ind-Set

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

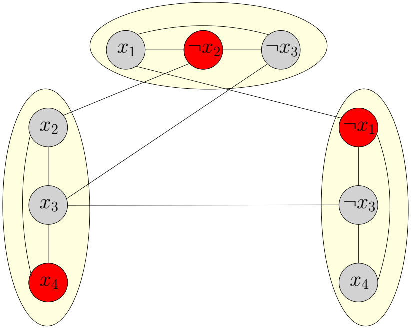
- A clause $\Rightarrow$ a group of $3$ vertices, one for each literal
- An edge between every pair of vertices in same group
- An edge between every pair of contradicting literals
- Problem: whether there is an IS of size $k = \#$clauses



# 3-Sat $\leq_P$ Ind-Set

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

- A clause $\Rightarrow$ a group of $3$ vertices, one for each literal
- An edge between every pair of vertices in same group
- An edge between every pair of contradicting literals
- Problem: whether there is an IS of size $k = \#$clauses

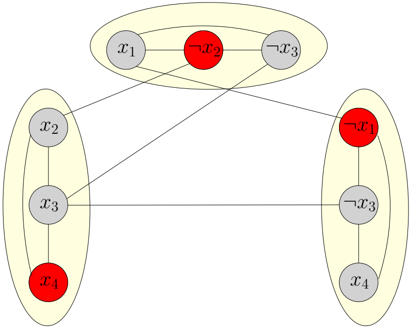



3-Sat instance is yes-instance $\Leftrightarrow$ Ind-Set instance is yes-instance:

# 3-Sat $\leq_P$ Ind-Set

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

- A clause $\Rightarrow$ a group of $3$ vertices, one for each literal
- An edge between every pair of vertices in same group
- An edge between every pair of contradicting literals
- Problem: whether there is an IS of size $k = \#$clauses

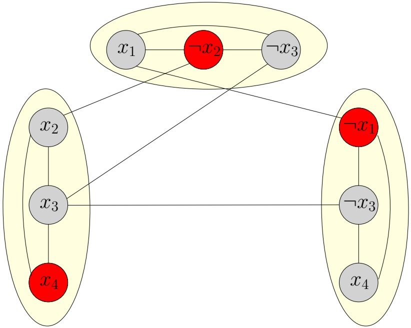



3-Sat instance is yes-instance $\Leftrightarrow$ Ind-Set instance is yes-instance:

- satisfying assignment $\Rightarrow$ independent set of size $k$
- independent set of size $k$ $\Rightarrow$ satisfying assignment

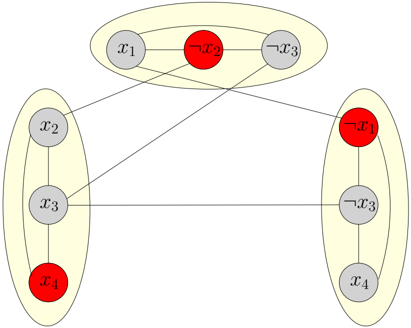
# Satisfying Assignment $\Rightarrow$ IS of Size $k$

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



# Satisfying Assignment $\Rightarrow$ IS of Size $k$

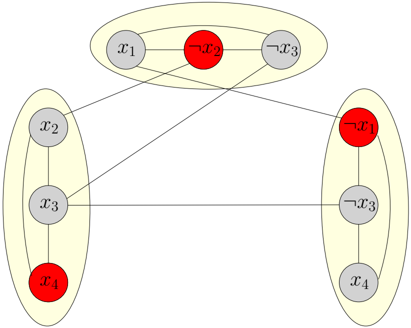
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$
- For every clause, at least 1 literal is satisfied



# Satisfying Assignment $\Rightarrow$ IS of Size $k$

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

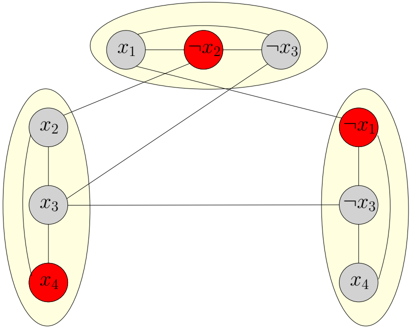
- For every clause, at least 1 literal is satisfied
- Pick the vertex correspondent the literal



# Satisfying Assignment $\Rightarrow$ IS of Size $k$

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

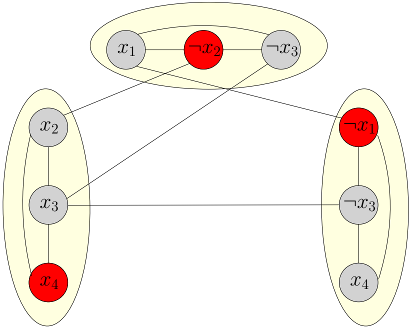
- For every clause, at least 1 literal is satisfied
- Pick the vertex correspondent the literal
- So, 1 literal from each group



# Satisfying Assignment $\Rightarrow$ IS of Size $k$

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

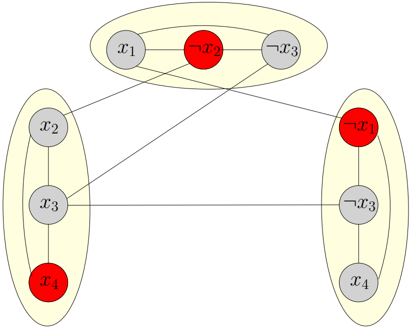
- For every clause, at least 1 literal is satisfied
- Pick the vertex correspondent the literal
- So, 1 literal from each group
- No contradictions among the selected literals



# Satisfying Assignment $\Rightarrow$ IS of Size $k$

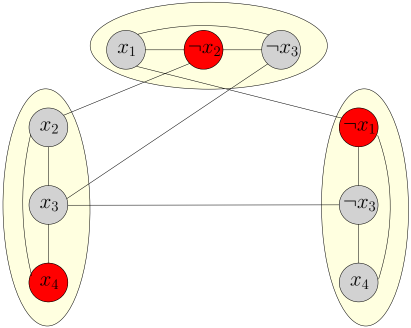
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$

- For every clause, at least 1 literal is satisfied
- Pick the vertex correspondent the literal
- So, 1 literal from each group
- No contradictions among the selected literals
- An IS of size $k$



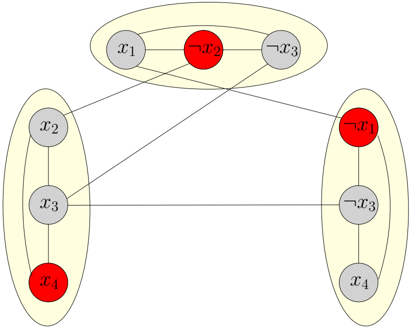
# IS of Size $k \Rightarrow$ Satisfying Assignment

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



# IS of Size $k \Rightarrow$ Satisfying Assignment

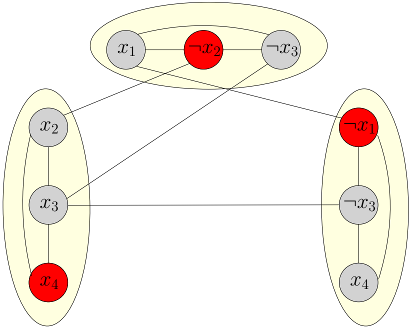
- $(x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor x_4)$
- For every group, exactly one literal is selected in IS



# IS of Size $k \Rightarrow$ Satisfying Assignment

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

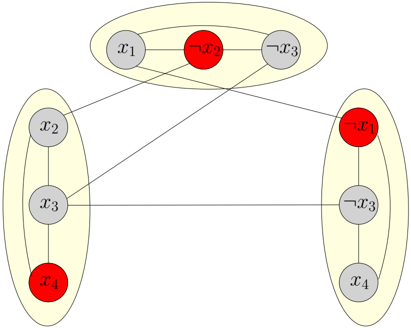
- For every group, exactly one literal is selected in IS
- No contradictions among the selected literals



# IS of Size $k \Rightarrow$ Satisfying Assignment

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

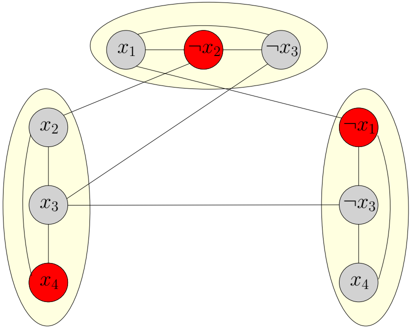
- For every group, exactly one literal is selected in IS
- No contradictions among the selected literals
- If $x_i$ is selected in IS, set $x_i = 1$



# IS of Size $k \Rightarrow$ Satisfying Assignment

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

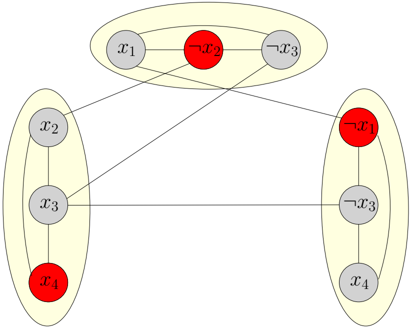
- For every group, exactly one literal is selected in IS
- No contradictions among the selected literals
- If $x_i$ is selected in IS, set $x_i = 1$
- If $\neg x_i$ is selected in IS, set $x_i = 0$



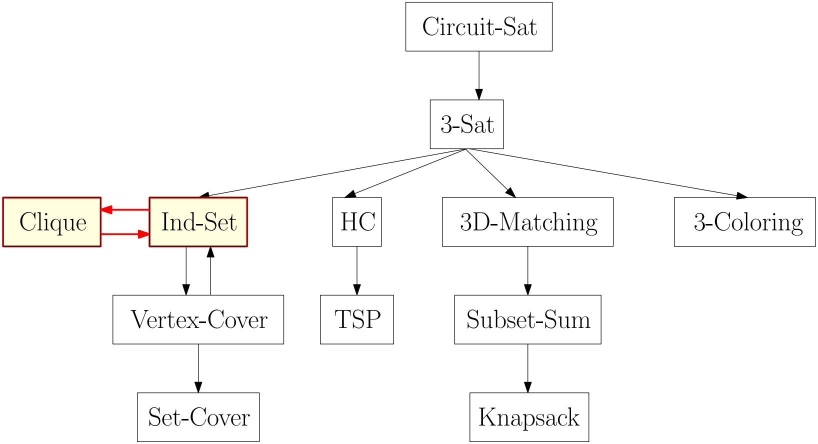
# IS of Size $k \Rightarrow$ Satisfying Assignment

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

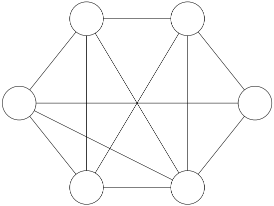
- For every group, exactly one literal is selected in IS
- No contradictions among the selected literals
- If $x_i$ is selected in IS, set $x_i = 1$
- If $\neg x_i$ is selected in IS, set $x_i = 0$
- Otherwise, set $x_i$ arbitrarily



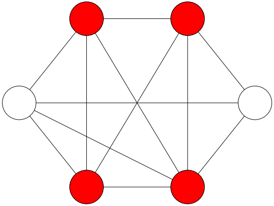
# Reductions of NP-Complete Problems



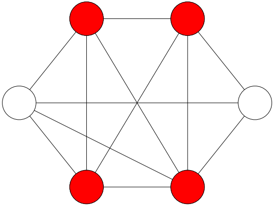
**Def.** A clique in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that $\forall u, v \in S$ we have $(u, v) \in E$



**Def.** A clique in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that $\forall u, v \in S$ we have $(u, v) \in E$



**Def.** A clique in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that $\forall u, v \in S$ we have $(u, v) \in E$

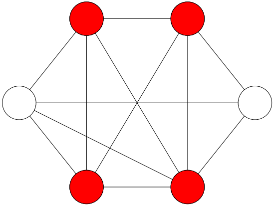


## Clique Problem

**Input:** $G = (V, E)$ and integer $k > 0$,

**Output:** whether there exists a clique of size $k$ in $G$

**Def.** A clique in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that $\forall u, v \in S$ we have $(u, v) \in E$



## Clique Problem

**Input:** $G = (V, E)$ and integer $k > 0$,

**Output:** whether there exists a clique of size $k$ in $G$

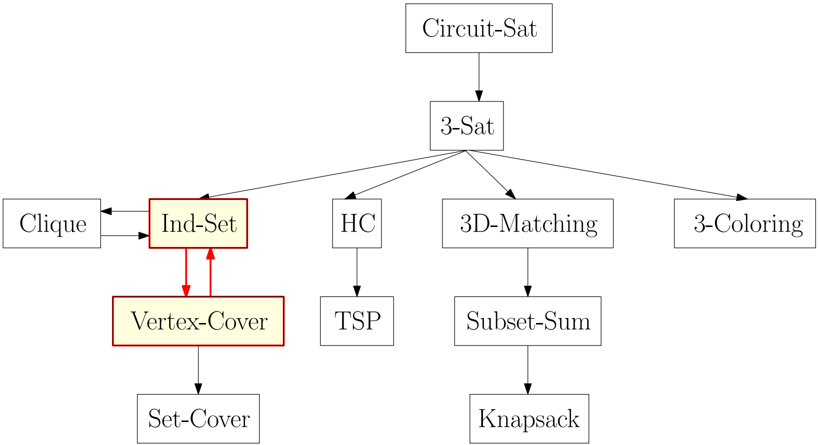
- What is the relationship between Clique and Ind-Set?

# Clique $=_P$ Ind-Set

**Def.** Given a graph $G = (V, E)$, define $\overline{G} = (V, \overline{E})$ be the graph such that $(u, v) \in \overline{E}$ if and only if $(u, v) \notin E$.

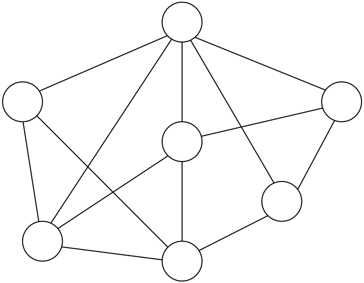
**Obs.** $S$ is an independent set in $G$ if and only if $S$ is a clique in $\overline{G}$.

# Reductions of NP-Complete Problems



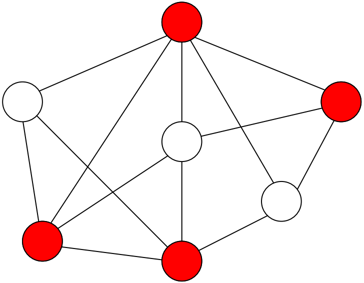
# Vertex-Cover

**Def.** Given a graph $G = (V, E)$, a vertex cover of $G$ is a subset $S \subseteq V$ such that for every $(u, v) \in E$ then $u \in S$ or $v \in S$ .



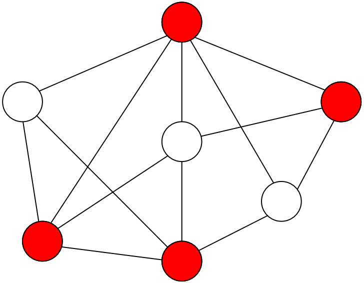
# Vertex-Cover

**Def.** Given a graph $G = (V, E)$, a vertex cover of $G$ is a subset $S \subseteq V$ such that for every $(u, v) \in E$ then $u \in S$ or $v \in S$ .



# Vertex-Cover

**Def.** Given a graph $G = (V, E)$, a vertex cover of $G$ is a subset $S \subseteq V$ such that for every $(u, v) \in E$ then $u \in S$ or $v \in S$ .



## Vertex-Cover Problem

**Input:** $G = (V, E)$ and integer $k$

**Output:** whether there is a vertex cover of $G$ of size at most $k$

# Vertex-Cover $=_P$ Ind-Set

# Vertex-Cover $=_P$ Ind-Set

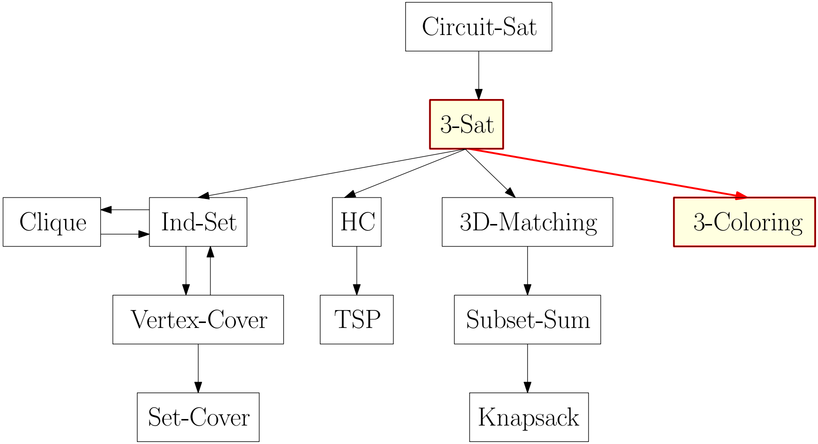
**Q:** What is the relationship between Vertex-Cover and Ind-Set?

# Vertex-Cover $=_P$ Ind-Set

**Q:** What is the relationship between Vertex-Cover and Ind-Set?

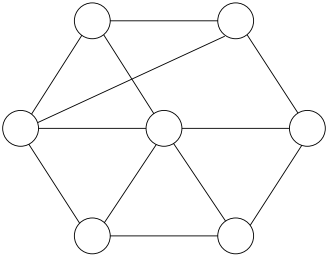
**A:** $S$ is a vertex-cover of $G = (V, E)$ if and only if $V \setminus S$ is an independent set of $G$.

# Reductions of NP-Complete Problems



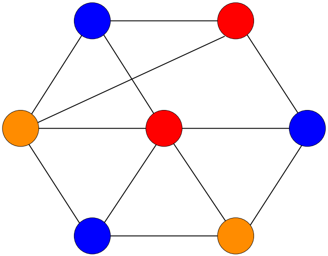
# $k$-coloring problem

**Def.** A $k$-coloring of $G = (V, E)$ is a function $f : V \to \{1, 2, 3, \cdots, k\}$ so that for every edge $(u, v) \in E$, we have $f(u) \neq f(v)$. $G$ is $k$-colorable if there is a $k$-coloring of $G$.



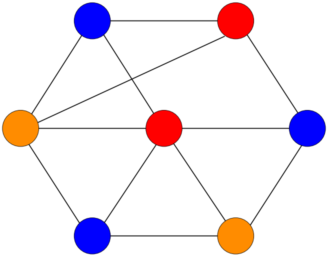
# $k$-coloring problem

**Def.** A $k$-coloring of $G = (V, E)$ is a function $f : V \to \{1, 2, 3, \cdots, k\}$ so that for every edge $(u, v) \in E$, we have $f(u) \neq f(v)$. $G$ is $k$-colorable if there is a $k$-coloring of $G$.



# $k$-coloring problem

**Def.** A $k$-coloring of $G = (V, E)$ is a function $f : V \to \{1, 2, 3, \cdots, k\}$ so that for every edge $(u, v) \in E$, we have $f(u) \neq f(v)$. $G$ is $k$-colorable if there is a $k$-coloring of $G$.



**$k$-coloring problem**

**Input:** a graph $G = (V, E)$

**Output:** whether $G$ is $k$-colorable or not

# 2-Coloring Problem

**Obs.** A graph $G$ is 2-colorable if and only if it is bipartite.

**Q:** How do we check if a graph $G$ is 2-colorable?

# 2-Coloring Problem

**Obs.** A graph $G$ is 2-colorable if and only if it is bipartite.

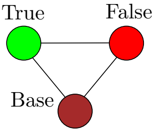
**Q:** How do we check if a graph $G$ is 2-colorable?

**A:** We check if $G$ is bipartite.

# 3-SAT $\leq_P$ 3-Coloring

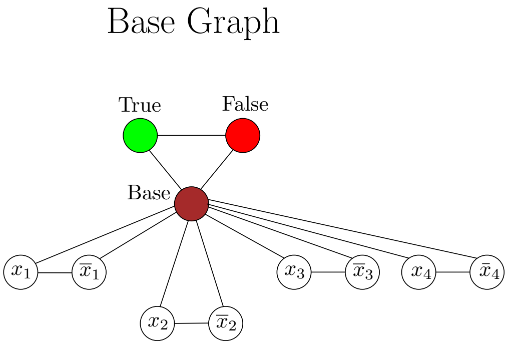
- Construct the base graph

Base Graph



# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph

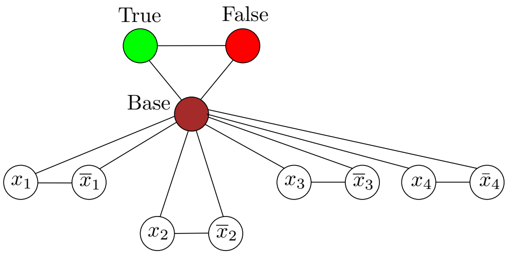


# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph

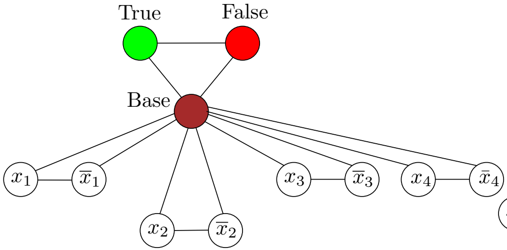
$x_1 \vee \neg x_2 \vee x_3$



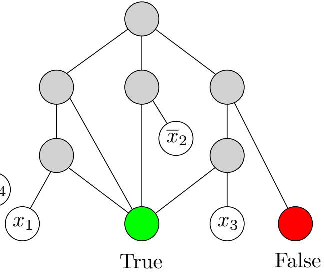
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph

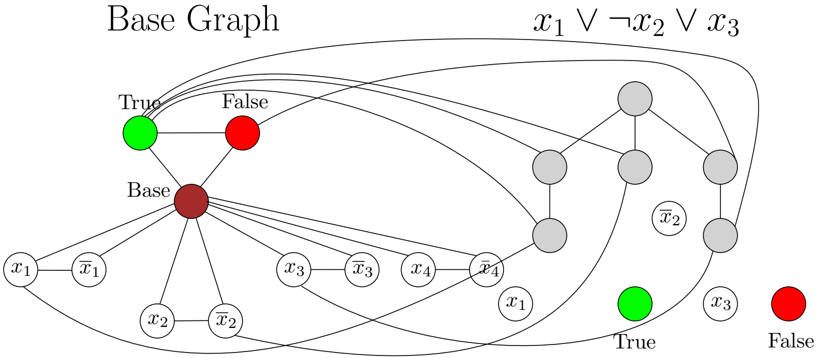



$x_1 \vee \neg x_2 \vee x_3$



# 3-SAT $\leq_P$ 3-Coloring

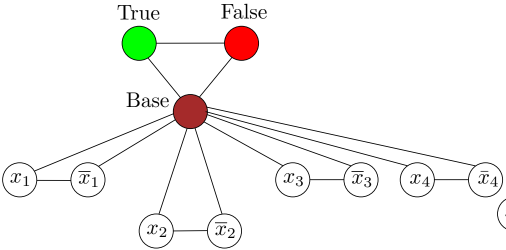
- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.



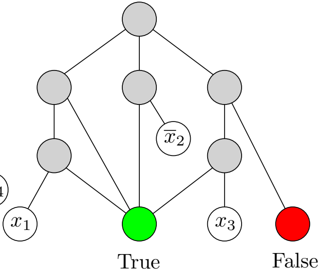
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



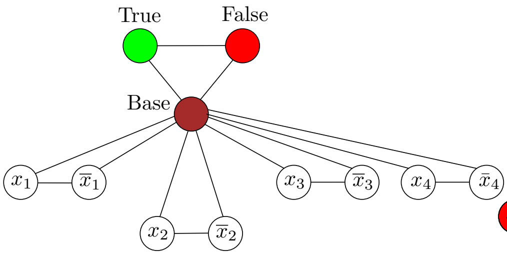

$x_1 \vee \neg x_2 \vee x_3$



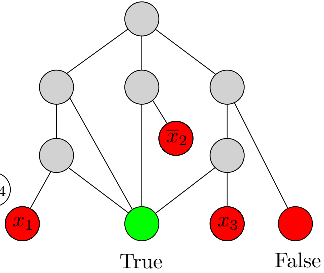
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



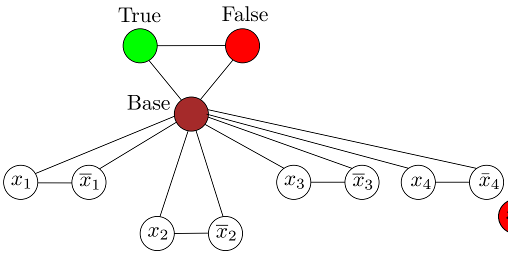

$x_1 \vee \neg x_2 \vee x_3$



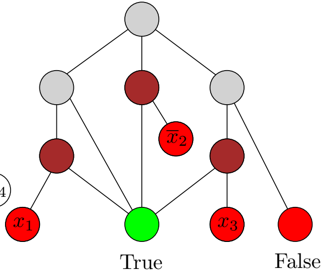
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



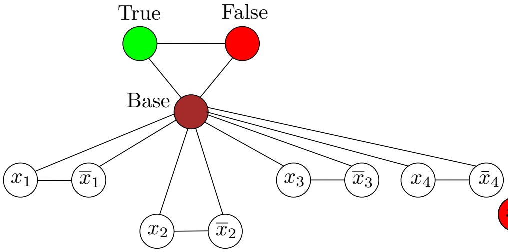

$x_1 \vee \neg x_2 \vee x_3$



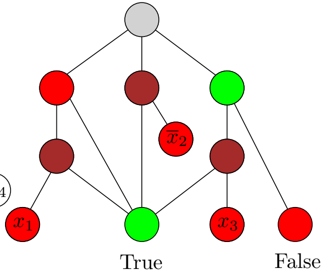
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



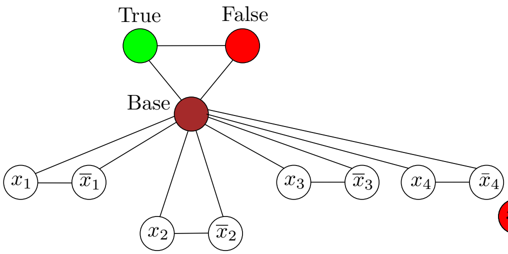

$x_1 \vee \neg x_2 \vee x_3$



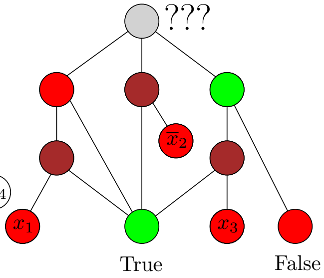
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph

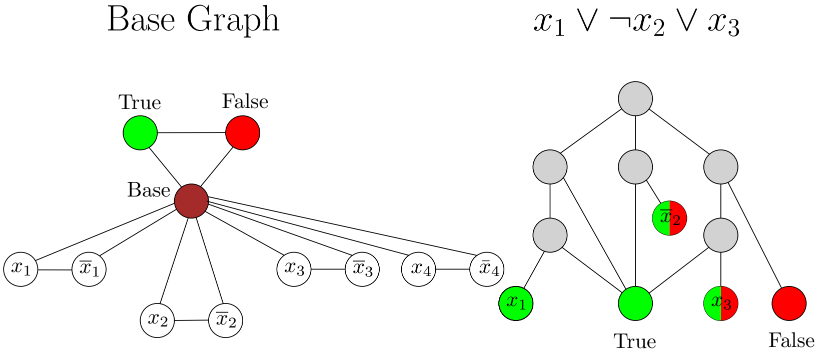



$x_1 \vee \neg x_2 \vee x_3$



# 3-SAT $\leq_P$ 3-Coloring

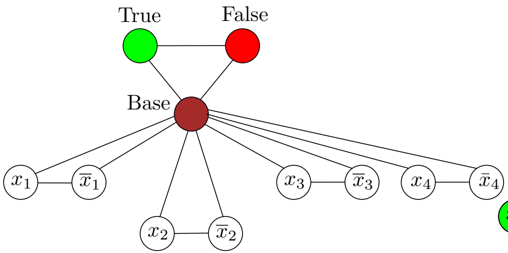
- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.



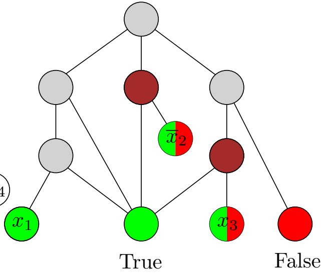
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph

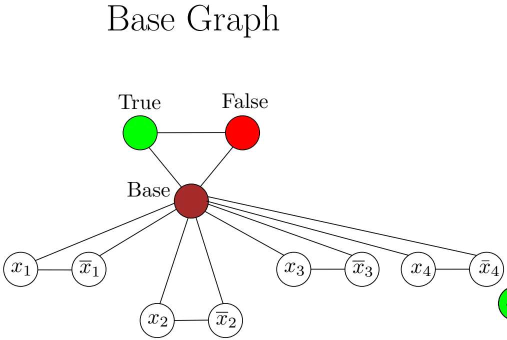


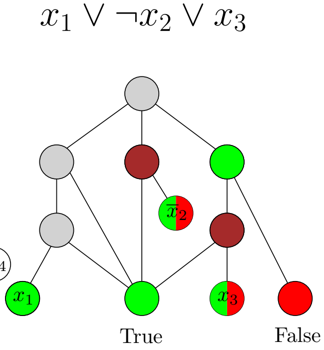

$x_1 \vee \neg x_2 \vee x_3$



# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

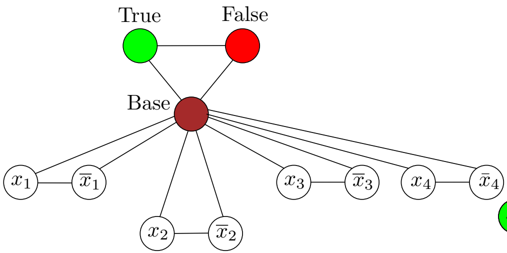


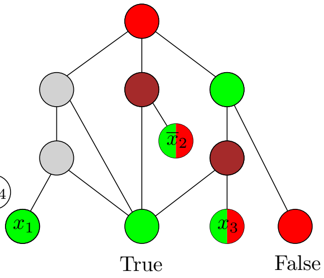
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



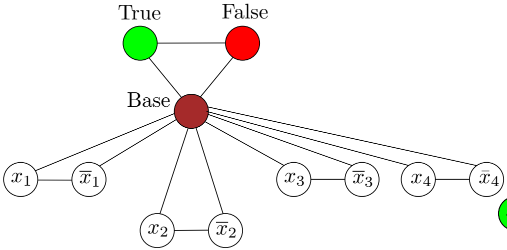

$x_1 \vee \neg x_2 \vee x_3$



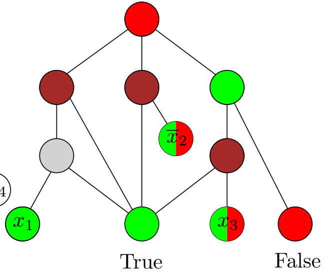
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



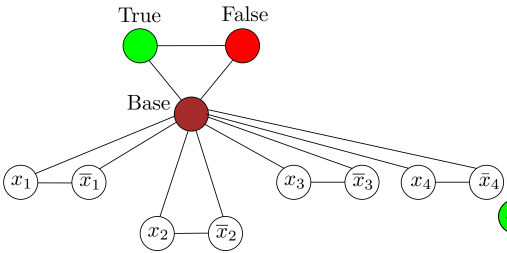

$x_1 \vee \neg x_2 \vee x_3$



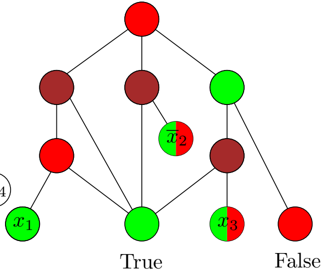
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



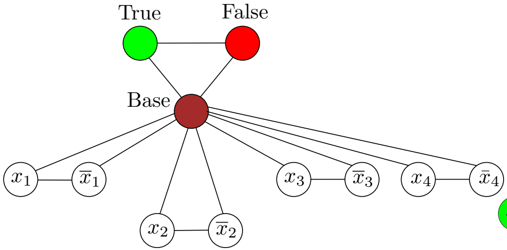

$x_1 \vee \neg x_2 \vee x_3$



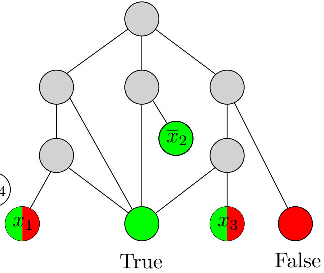
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



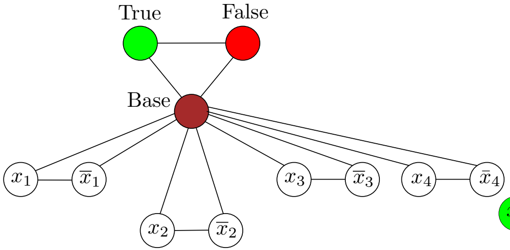

$x_1 \vee \neg x_2 \vee x_3$



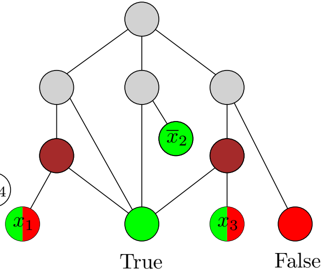
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph

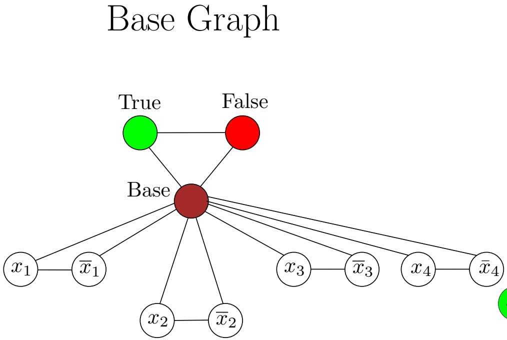


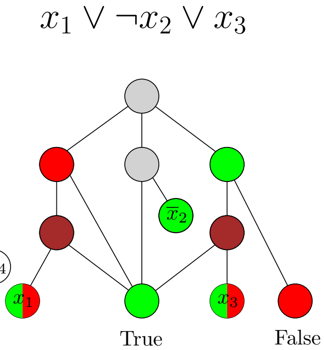

$x_1 \vee \neg x_2 \vee x_3$



# 3-SAT $\leq_P$ 3-Coloring

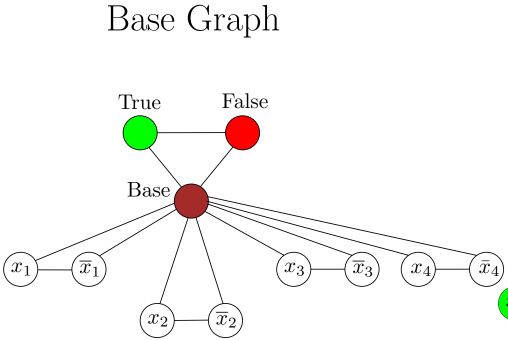
- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

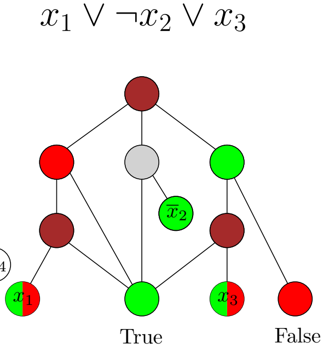


# 3-SAT $\leq_P$ 3-Coloring

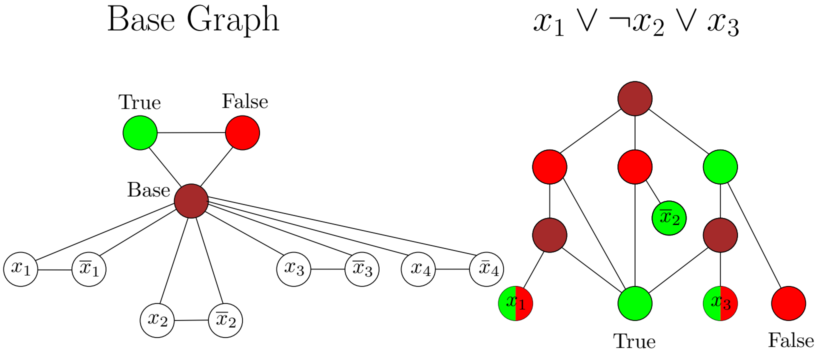
- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.



# 3-SAT $\leq_P$ 3-Coloring

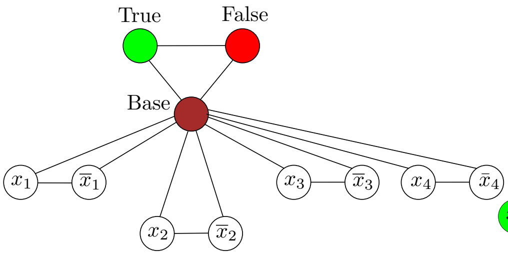
- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.



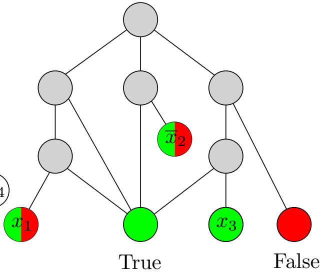
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



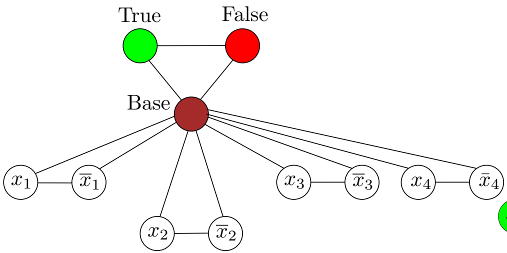

$x_1 \vee \neg x_2 \vee x_3$



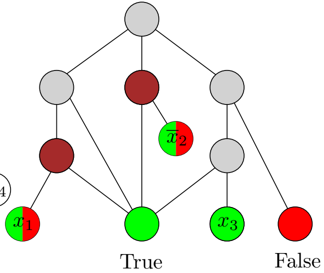
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



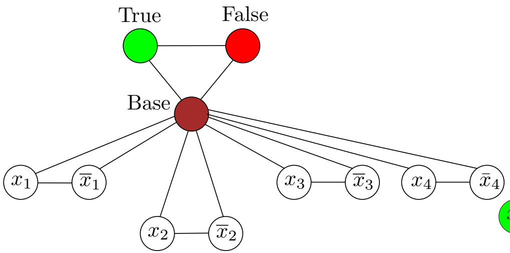

$x_1 \vee \neg x_2 \vee x_3$



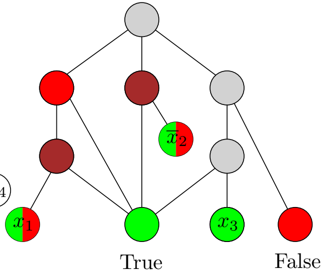
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



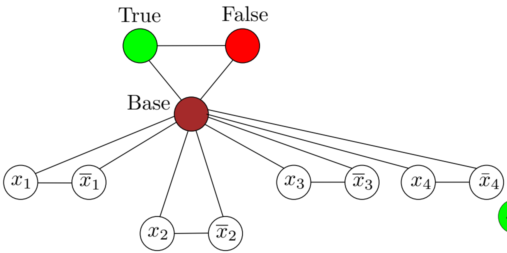

$x_1 \vee \neg x_2 \vee x_3$



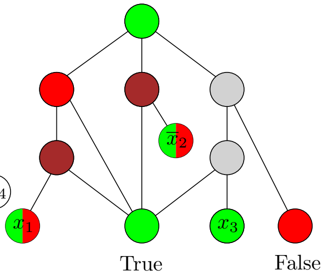
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



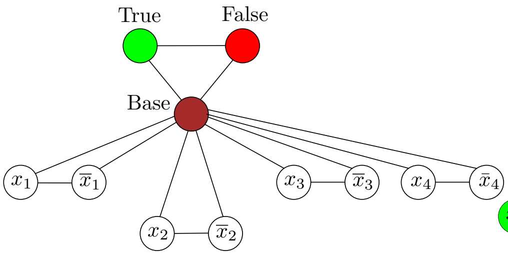

$x_1 \vee \neg x_2 \vee x_3$



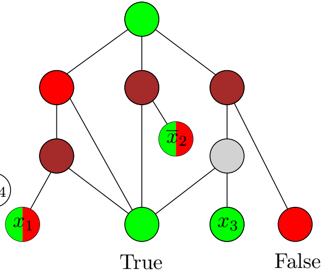
# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph



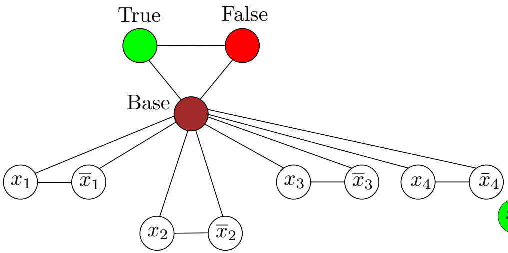

$x_1 \vee \neg x_2 \vee x_3$



# 3-SAT $\leq_P$ 3-Coloring

- Construct the base graph
- Construct a gadget from each clause: gadget is 3-colorable if and only if the clause is satisfied.

Base Graph




$x_1 \vee \neg x_2 \vee x_3$