4 Possibilities of Relationships

Notice that $X \in \text{NP} \iff \overline{X} \in \text{Co-NP}$ and $P \subseteq \text{NP} \cap \text{Co-NP}$

- $P = \text{NP} = \text{Co-NP}$
- $\text{NP} = \text{Co-NP}$
- $\text{NP} \subseteq \text{NP} \cap \text{Co-NP} \subseteq \text{Co-NP}$
- $\text{NP} \subseteq \text{NP} \cap \text{Co-NP}$

- People commonly believe we are in the 4th scenario
Outline

1. Some Hard Problems
2. P, NP and Co-NP
3. Polynomial Time Reductions and NP-Completeness
4. NP-Complete Problems
5. Dealing with NP-Hard Problems
6. Summary
**Def.** Given a black box algorithm $A$ that solves a problem $X$, if any instance of a problem $Y$ can be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to $A$, then we say $Y$ is polynomial-time reducible to $X$, denoted as $Y \leq_P X$.
Def. Given a black box algorithm $A$ that solves a problem $X$, if any instance of a problem $Y$ can be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to $A$, then we say $Y$ is polynomial-time reducible to $X$, denoted as $Y \leq_P X$.

To prove positive results:

Suppose $Y \leq_P X$. If $X$ can be solved in polynomial time, then $Y$ can be solved in polynomial time.
Def. Given a black box algorithm $A$ that solves a problem $X$, if any instance of a problem $Y$ can be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to $A$, then we say $Y$ is polynomial-time reducible to $X$, denoted as $Y \leq_P X$.

To prove positive results:

Suppose $Y \leq_P X$. If $X$ can be solved in polynomial time, then $Y$ can be solved in polynomial time.

To prove negative results:

Suppose $Y \leq_P X$. If $Y$ cannot be solved in polynomial time, then $X$ cannot be solved in polynomial time.
## 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 \( G \)
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**  $\text{HP} \leq_P \text{HC}$. 
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 $\leq_P$ HC.
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 $\text{HP} \leq_{P} \text{HC}$. 

![Diagram of graphs demonstrating the reduction from HP to HC](image)
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** $\text{HP} \leq_{P} \text{HC}$.

**Obs.** $G$ has a HP from $s$ to $t$ if and only if graph on right side has a HC.
Def. A problem $X$ is called **NP-complete** if

1. $X \in \text{NP}$, and
2. $Y \leq_P X$ for every $Y \in \text{NP}$.
NP-Completeness

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

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

- 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. $Y \leq_P X$ 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)
**Def.** A problem $X$ is called **NP-complete** if

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

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

- NP-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

1. Some Hard Problems
2. P, NP and Co-NP
3. Polynomial Time Reductions and NP-Completeness
4. NP-Complete Problems
5. Dealing with NP-Hard Problems
6. Summary
Def. A problem $X$ is called NP-complete if

1. $X \in \text{NP}$, and
2. $Y \leq_{P} X$ for every $Y \in \text{NP}$.
Def. A problem $X$ is called **NP-complete** if

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

How can we find a problem $X \in \text{NP}$ such that every problem $Y \in \text{NP}$ is polynomial time reducible to $X$? Are we asking for too much?
Def. A problem $X$ is called \textbf{NP-complete} if

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

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

**Input:** a circuit

**Output:** whether the circuit is satisfiable
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 $Y \in \text{NP}$ can be reduced to Circuit-Sat.
- We prove $\text{HC} \leq_P \text{Circuit-Sat}$ as an example.
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 \text{Circuit-Sat}

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

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

Construct a circuit $C'$ for the algorithm $\text{check-HC}$

hard-wire the instance $G$ to the circuit $C'$ to obtain the circuit $C$
Let \( \text{check-HC}(G, S) \) be the certifier for the Hamiltonian cycle problem: \( \text{check-HC}(G, S) \) returns 1 if \( S \) is a Hamiltonian cycle in \( G \) and 0 otherwise.

\( G \) is a yes-instance if and only if there is an \( S \) such that \( \text{check-HC}(G, S) \) 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.
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
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

- Circuit-Sat
  - 3-Sat
    - Clique
    - Ind-Set
      - Vertex-Cover
        - Set-Cover
    - HC
    - 3D-Matching
    - 3-Coloring
    - Knapsack
    - Subset-Sum
    - TSP
3-CNF (conjunctive normal form) is a special case of formula:
3-CNF (conjunctive normal form) is a special case of formula:

- **Boolean variables:** $x_1, x_2, \ldots, x_n$
3-Sat

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

- **Boolean variables**: \( x_1, x_2, \ldots, x_n \)
- **Literals**: \( x_i \) or \( \neg x_i \)
3-Sat

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

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

Assign: $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$ satisfies $(x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)$
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

**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
  \[
  (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)
  \]
Associate every wire with a new variable. The circuit is equivalent to the following formula:

\((x_4 = \neg x_3)^\land (x_5 = x_1 \land x_2)^\land (x_6 = \neg x_4)^\land (x_7 = x_1 \land x_2 \land x_4)^\land (x_8 = x_5 \land x_6)^\land (x_9 = x_6 \land x_7)^\land (x_{10} = x_8 \land x_9 \land x_7)\)
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:

\[
(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4) \\
\land (x_7 = x_4 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6) \\
\land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}
\]
Circuit-Sat $\leq_P$ 3-Sat

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

Convert each clause to a 3-CNF
Circuit-Sat $\leq_P$ 3-Sat

\[(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4) \]
\[\land (x_7 = x_1 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6) \]
\[\land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}\]

Convert each clause to a 3-CNF

\[x_5 = x_1 \lor x_2 \quad \iff \quad x_5 \leftrightarrow x_1 \lor x_2\]

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>(x_5 \leftrightarrow x_1 \lor x_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P$ 3-Sat

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

Convert each clause to a 3-CNF

$x_5 = x_1 \lor x_2$  $\iff$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>$x_5 \iff x_1 \lor x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P 3$-Sat

$$(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4)$$

$$\land (x_7 = x_1 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6)$$

$$\land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}$$

Convert each clause to a 3-CNF

$x_5 = x_1 \lor x_2 \iff$

$$(x_1 \lor x_2 \lor \neg x_5) \land$$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>$x_5 \leftrightarrow x_1 \lor x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P$ 3-Sat

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

Convert each clause to a 3-CNF

$$x_5 = x_1 \lor x_2 \iff$$

$$(x_1 \lor x_2 \lor \neg x_5) \land$$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>$x_5 \iff x_1 \lor x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P$ 3-Sat

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

Convert each clause to a 3-CNF

$x_5 = x_1 \lor x_2$  $\Leftrightarrow$

$(x_1 \lor x_2 \lor \neg x_5) \land$

$(x_1 \lor \neg x_2 \lor x_5) \land$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>$x_5 \Leftrightarrow x_1 \lor x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P$ 3-Sat

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

Convert each clause to a 3-CNF

$x_5 = x_1 \lor x_2 \iff$

$$(x_1 \lor x_2 \lor \neg x_5) \land$$
$$(x_1 \lor \neg x_2 \lor x_5) \land$$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_5$</th>
<th>$x_5 \iff x_1 \lor x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq_P$ 3-Sat

\[
(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4) \\
\land (x_7 = x_1 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6) \\
\land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}
\]

Convert each clause to a 3-CNF

\[
x_5 = x_1 \lor x_2 \iff
\]

\[
(x_1 \lor x_2 \lor \neg x_5) \land \\
(x_1 \lor \neg x_2 \lor x_5) \land \\
(\neg x_1 \lor x_2 \lor x_5)
\]

\[
\begin{array}{c|c|c|c|c}
  x_1 & x_2 & x_5 & x_5 \leftrightarrow x_1 \lor x_2 \\
 0 & 0 & 0 & 1 \\
 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 \\
 1 & 0 & 0 & 0 \\
 1 & 0 & 1 & 1 \\
 1 & 1 & 0 & 0 \\
 1 & 1 & 1 & 1 \\
\end{array}
\]
Circuit-Sat \( \leq_P \) 3-Sat

\[(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4) \land (x_7 = x_1 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6) \land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}\]

Convert each clause to a 3-CNF

\[x_5 = x_1 \lor x_2 \iff\]

\[(x_1 \lor x_2 \lor \neg x_5) \land (x_1 \lor \neg x_2 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_5)\]

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_5)</th>
<th>(x_5 \iff x_1 \lor x_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Circuit-Sat $\leq P$ 3-Sat

\[
(x_4 = \neg x_3) \land (x_5 = x_1 \lor x_2) \land (x_6 = \neg x_4) \\
\land (x_7 = x_1 \land x_2 \land x_4) \land (x_8 = x_5 \lor x_6) \\
\land (x_9 = x_6 \lor x_7) \land (x_{10} = x_8 \land x_9 \land x_7) \land x_{10}
\]

Convert each clause to a 3-CNF

\[
x_5 = x_1 \lor x_2 \quad \Leftrightarrow \\
(x_1 \lor x_2 \lor \neg x_5) \land \]
\[
(x_1 \lor \neg x_2 \lor x_5) \land \\
(\neg x_1 \lor x_2 \lor x_5) \land \\
(\neg x_1 \lor \neg x_2 \lor x_5)
\]

\[
\begin{array}{ccc|c}
 x_1 & x_2 & x_5 & x_5 \leftrightarrow x_1 \lor x_2 \\
\hline
 0 & 0 & 0 & 1 \\
 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 \\
 1 & 0 & 0 & 0 \\
 1 & 0 & 1 & 1 \\
 1 & 1 & 0 & 0 \\
 1 & 1 & 1 & 1 \\
\end{array}
\]
Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\iff$ Formula $\iff$ 3-CNF
Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\iff$ Formula $\iff$ 3-CNF
- The circuit is satisfiable if and only if the 3-CNF is satisfiable
Circuit-Sat $\leq_P$ 3-Sat

- Circuit $\iff$ Formula $\iff$ 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 $\iff$ Formula $\iff$ 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

- Circuit-Sat
- 3-Sat
- 3-Coloring
- 3D-Matching
- Subset-Sum
- Knapsack
- TSP
- HC
- Ind-Set
- Vertex-Cover
- Clique
- Set-Cover