Polynomial Time Reductions

Lecture 22
April 18
Part I

(Polynomial Time) Reductions
A reduction from Problem $X$ to Problem $Y$ means (informally) that if we have an algorithm for Problem $Y$, we can use it to find an algorithm for Problem $X$. 
A reduction from Problem $X$ to Problem $Y$ means (informally) that if we have an algorithm for Problem $Y$, we can use it to find an algorithm for Problem $X$.

Using Reductions

We use reductions to find algorithms to solve problems.
Reductions

A reduction from Problem $X$ to Problem $Y$ means (informally) that if we have an algorithm for Problem $Y$, we can use it to find an algorithm for Problem $X$.

Using Reductions

1. We use reductions to find algorithms to solve problems.
2. We also use reductions to show that we can’t find algorithms for some problems. (We say that these problems are hard.)
For languages $L_X, L_Y$, a reduction from $L_X$ to $L_Y$ is:

1. An algorithm . . .
2. Input: $w \in \Sigma^*$
3. Output: $w' \in \Sigma^*$
4. Such that:

$$w \in L_Y \iff w' \in L_X$$

(Actually, this is only one type of reduction, but this is the one we'll use most often.) There are other kinds of reductions.
For languages $L_X$, $L_Y$, a reduction from $L_X$ to $L_Y$ is:

1. An algorithm . . .
2. Input: $w \in \Sigma^*$
3. Output: $w' \in \Sigma^*$
4. Such that:

$$w \in L_Y \iff w' \in L_X$$

(Actually, this is only one type of reduction, but this is the one we’ll use most often.) There are other kinds of reductions.
For decision problems $X$, $Y$, a reduction from $X$ to $Y$ is:

1. An algorithm . . .
2. Input: $I_X$, an instance of $X$.
4. Such that:

   $I_Y$ is YES instance of $Y$ $\iff$ $I_X$ is YES instance of $X$
Using reductions to solve problems

1. $\mathcal{R}$: Reduction $X \rightarrow Y$

2. $A_Y$: algorithm for $Y$:

\[
\text{If } \mathcal{R} \text{ and } A_Y \text{ polynomial-time} \Rightarrow A_X \text{ polynomial-time.}
\]
Using reductions to solve problems

1. $\mathcal{R}$: Reduction $X \rightarrow Y$

2. $\mathcal{A}_Y$: algorithm for $Y$:

3. $\implies$ New algorithm for $X$:

$\mathcal{A}_X(I_X)$:

// $I_X$: instance of $X$.

$I_Y \leftarrow \mathcal{R}(I_X)$ \n
return $\mathcal{A}_Y(I_Y)$
Using reductions to solve problems

1. \( R \): Reduction \( X \rightarrow Y \)
2. \( A_Y \): algorithm for \( Y \):
   
3. \( \implies \) New algorithm for \( X \):
   
   \[
   A_X(I_X):
   \]
   
   // \( I_X \): instance of \( X \).
   
   \( I_Y \leftarrow R(I_X) \)
   
   return \( A_Y(I_Y) \)

If \( R \) and \( A_Y \) polynomial-time \( \implies \) \( A_X \) polynomial-time.
Comparing Problems

1. “Problem $X$ is no harder to solve than Problem $Y$”.

2. If Problem $X$ reduces to Problem $Y$ (we write $X \leq Y$), then $X$ cannot be harder to solve than $Y$.

3. $X \leq Y$:
   1. $X$ is no harder than $Y$, or
   2. $Y$ is at least as hard as $X$. 
Part II

Examples of Reductions
Independent Sets and Cliques

Given a graph $G$, a set of vertices $V'$ is:
Independent Sets andCliques

Given a graph $G$, a set of vertices $V'$ is:

1. **independent set**: no two vertices of $V'$ connected by an edge.
Independent Sets and Cliques

Given a graph $G$, a set of vertices $V'$ is:

1. **independent set**: no two vertices of $V'$ connected by an edge.
2. **clique**: every pair of vertices in $V'$ is connected by an edge of $G$. 
Independent Sets and Cliques

Given a graph $G$, a set of vertices $V'$ is:

1. **independent set**: no two vertices of $V'$ connected by an edge.
2. **clique**: every pair of vertices in $V'$ is connected by an edge of $G$. 

![Graph Example](image_url)
Independent Sets and Cliques

Given a graph $G$, a set of vertices $V'$ is:

1. **independent set**: no two vertices of $V'$ connected by an edge.
2. **clique**: every pair of vertices in $V'$ is connected by an edge of $G$. 

![Graph Image]
Independent Sets and Cliques

Given a graph $G$, a set of vertices $V'$ is:

1. **independent set**: no two vertices of $V'$ connected by an edge.
2. **clique**: every pair of vertices in $V'$ is connected by an edge of $G$. 

![Graph Diagram with independent set and clique examples]
Problem: **Independent Set**

Instance: A graph $G$ and an integer $k$.

Question: Does $G$ has an independent set of size $\geq k$?
The **Independent Set** and **Clique** Problems

**Problem: Independent Set**

**Instance:** A graph $G$ and an integer $k$.

**Question:** Does $G$ have an independent set of size $\geq k$?

**Problem: Clique**

**Instance:** A graph $G$ and an integer $k$.

**Question:** Does $G$ have a clique of size $\geq k$?
Recall

For decision problems $X$, $Y$, a reduction from $X$ to $Y$ is:

1. An algorithm . . .
2. that takes $I_X$, an instance of $X$ as input . . .
3. and returns $I_Y$, an instance of $Y$ as output . . .
4. such that the solution (YES/NO) to $I_Y$ is the same as the solution to $I_X$. 
Reducing **Independent Set** to **Clique**

An instance of **Independent Set** is a graph $G$ and an integer $k$. 
Reducing **Independent Set** to **Clique**

An instance of **Independent Set** is a graph $G$ and an integer $k$. 

![Graph Diagram]
Reducing **Independent Set** to **Clique**

An instance of **Independent Set** is a graph $G$ and an integer $k$.

Reduction given $< G, k >$ outputs $< \overline{G}, k >$ where $\overline{G}$ is the complement of $G$. $\overline{G}$ has an edge $(u, v)$ if and only if $(u, v)$ is not an edge of $G$.
Reducing **Independent Set** to **Clique**

An instance of **Independent Set** is a graph $G$ and an integer $k$.

Reduction given $< G, k >$ outputs $< \overline{G}, k >$ where $\overline{G}$ is the *complement* of $G$. $\overline{G}$ has an edge $(u, v)$ if and only if $(u, v)$ is not an edge of $G$. 

![Diagram of a graph with four nodes connected to form a figure eight shape](attachment:image.png)
Reducing **Independent Set** to Clique

An instance of **Independent Set** is a graph \( G \) and an integer \( k \).

Reduction given \(< G, k >\) outputs \(< \overline{G}, k >\) where \( \overline{G} \) is the complement of \( G \). \( \overline{G} \) has an edge \((u, v)\) if and only if \((u, v)\) is not an edge of \( G \).
Reducing **Independent Set** to **Clique**

An instance of **Independent Set** is a graph $G$ and an integer $k$.

Reduction given $< G, k >$ outputs $< \overline{G}, k >$ where $\overline{G}$ is the complement of $G$. $\overline{G}$ has an edge $(u, v)$ if and only if $(u, v)$ is not an edge of $G$. 
Correctness of reduction

**Lemma**

$G$ has an independent set of size $k$ if and only if $\overline{G}$ has a clique of size $k$.

**Proof.**

Need to prove two facts:

1. $G$ has independent set of size at least $k$ implies that $\overline{G}$ has a clique of size at least $k$.
2. $\overline{G}$ has a clique of size at least $k$ implies that $G$ has an independent set of size at least $k$.

Easy to see both from the fact that $S \subseteq V$ is an independent set in $G$ if and only if $S$ is a clique in $\overline{G}$.
Independent Set and Clique

1. Independent Set $\leq$ Clique.

What does this mean?

If we have an algorithm for Clique, then we have an algorithm for Independent Set.

Clique is at least as hard as Independent Set.

Also...

Clique $\leq$ Independent Set. Why? Thus Clique and Independent Set are polynomial-time equivalent.
Independent Set and Clique

1. \textbf{Independent Set} $\leq$ \textbf{Clique}.
   What does this mean?

2. If have an algorithm for \textbf{Clique}, then we have an algorithm for \textbf{Independent Set}.
Independent Set and Clique

1. **Independent Set \( \leq \) Clique.** What does this mean?

2. If have an algorithm for **Clique**, then we have an algorithm for **Independent Set**.

3. **Clique** is *at least as hard as Independent Set*. 
Independent Set and Clique

1. **Independent Set ≤ Clique.** What does this mean?
2. If have an algorithm for **Clique**, then we have an algorithm for **Independent Set**.
3. **Clique** is *at least as hard as Independent Set*.
4. Also... **Clique ≤ Independent Set**. Why? Thus **Clique** and **Independent Set** are polynomial-time equivalent.
A DFA $M$ is universal if it accepts every string. That is, $L(M) = \Sigma^*$, the set of all strings.
A DFA $M$ is universal if it accepts every string. That is, $L(M) = \Sigma^*$, the set of all strings.

Problem (DFA universality)

Input: A DFA $M$.
Goal: Is $M$ universal?
A DFA $M$ is universal if it accepts every string. That is, $L(M) = \Sigma^*$, the set of all strings.

Problem (DFA universality)

**Input:** A DFA $M$.
**Goal:** Is $M$ universal?

How do we solve DFA Universality?
A **DFA** $M$ is universal if it accepts every string.
That is, $L(M) = \Sigma^*$, the set of all strings.

**Problem (DFA universality)**

**Input:** A **DFA** $M$.
**Goal:** Is $M$ universal?

How do we solve **DFA Universality**?
We check if $M$ has any reachable non-final state.
An NFA $N$ is said to be universal if it accepts every string. That is, $L(N) = \Sigma^*$, the set of all strings.

**Problem (NFA universality)**

**Input:** A NFA $M$.

**Goal:** Is $M$ universal?

How do we solve NFA Universality?
An NFA $N$ is said to be universal if it accepts every string. That is, $L(N) = \Sigma^*$, the set of all strings.

Problem (NFA universality)

Input: A NFA $M$.
Goal: Is $M$ universal?

How do we solve NFA Universality? Reduce it to DFA Universality?
An NFA $N$ is said to be universal if it accepts every string. That is, $L(N) = \Sigma^*$, the set of all strings.

**Problem (NFA universality)**

**Input:** A NFA $M$.

**Goal:** Is $M$ universal?

How do we solve NFA Universality?
Reduce it to DFA Universality?
Given an NFA $N$, convert it to an equivalent DFA $M$, and use the DFA Universality Algorithm.
An **NFA** $N$ is said to be **universal** if it accepts every string. That is, $L(N) = \Sigma^*$, the set of all strings.

**Problem (**NFA universality**)**

**Input:** A **NFA** $M$.

**Goal:** Is $M$ universal?

How do we solve **NFA Universality**?
Reduce it to **DFA Universality**?

Given an **NFA** $N$, convert it to an equivalent **DFA** $M$, and use the **DFA Universality** Algorithm.

The reduction takes exponential time!

**NFA Universality** is known to be PSPACE-Complete and we do not expect a polynomial-time algorithm.
Polynomial-time reductions

We say that an algorithm is **efficient** if it runs in polynomial-time.
We say that an algorithm is **efficient** if it runs in polynomial-time.

To find efficient algorithms for problems, we are only interested in polynomial-time reductions. Reductions that take longer are not useful.
Polynomial-time reductions

We say that an algorithm is **efficient** if it runs in polynomial-time. To find efficient algorithms for problems, we are only interested in polynomial-time reductions. Reductions that take longer are not useful.

If we have a polynomial-time reduction from problem $X$ to problem $Y$ (we write $X \leq_p Y$), and a poly-time algorithm $A_Y$ for $Y$, we have a polynomial-time/efficient algorithm for $X$. 
We say that an algorithm is **efficient** if it runs in polynomial-time.

To find efficient algorithms for problems, we are only interested in *polynomial-time* reductions. Reductions that take longer are not useful.

If we have a polynomial-time reduction from problem $X$ to problem $Y$ (we write $X \leq_P Y$), and a poly-time algorithm $A_Y$ for $Y$, we have a polynomial-time/efficient algorithm for $X$. 

\[ I_X \xrightarrow{R} I_Y \xrightarrow{A_Y} \begin{cases} \text{YES} \\ \text{NO} \end{cases} \]

\[ A_X \]
Polynomial-time Reduction

A polynomial time reduction from a decision problem $X$ to a decision problem $Y$ is an algorithm $A$ that has the following properties:

1. given an instance $I_X$ of $X$, $A$ produces an instance $I_Y$ of $Y$
2. $A$ runs in time polynomial in $|I_X|$.
3. Answer to $I_X$ YES iff answer to $I_Y$ is YES.

Proposition

If $X \leq_P Y$ then a polynomial time algorithm for $Y$ implies a polynomial time algorithm for $X$.

Such a reduction is called a Karp reduction. Most reductions we will need are Karp reductions. Karp reductions are the same as mapping reductions when specialized to polynomial time for the reduction step.
Let $X$ and $Y$ be two decision problems, such that $X$ can be solved in polynomial time, and $X \leq_p Y$. Then

(A) $Y$ can be solved in polynomial time.
(B) $Y$ can NOT be solved in polynomial time.
(C) If $Y$ is hard then $X$ is also hard.
(D) None of the above.
(E) All of the above.
For decision problems $X$ and $Y$, if $X \leq_P Y$, and $Y$ has an efficient algorithm, $X$ has an efficient algorithm.
Polynomial-time reductions and hardness

For decision problems $X$ and $Y$, if $X \leq_P Y$, and $Y$ has an efficient algorithm, $X$ has an efficient algorithm.

If you believe that Independent Set does not have an efficient algorithm, why should you believe the same of Clique?
For decision problems $X$ and $Y$, if $X \leq_P Y$, and $Y$ has an efficient algorithm, $X$ has an efficient algorithm.

If you believe that Independent Set does not have an efficient algorithm, why should you believe the same of Clique?

Because we showed Independent Set $\leq_P$ Clique. If Clique had an efficient algorithm, so would Independent Set!
Polynomial-time reductions and hardness

For decision problems $X$ and $Y$, if $X \leq_P Y$, and $Y$ has an efficient algorithm, $X$ has an efficient algorithm.

If you believe that Independent Set does not have an efficient algorithm, why should you believe the same of Clique?

Because we showed Independent Set $\leq_P$ Clique. If Clique had an efficient algorithm, so would Independent Set!

If $X \leq_P Y$ and $X$ does not have an efficient algorithm, $Y$ cannot have an efficient algorithm!
Proposition

Let \( R \) be a polynomial-time reduction from \( X \) to \( Y \). Then for any instance \( I_X \) of \( X \), the size of the instance \( I_Y \) of \( Y \) produced from \( I_X \) by \( R \) is polynomial in the size of \( I_X \).
Polynomial-time reductions and instance sizes

Proposition

Let \( R \) be a polynomial-time reduction from \( X \) to \( Y \). Then for any instance \( I_X \) of \( X \), the size of the instance \( I_Y \) of \( Y \) produced from \( I_X \) by \( R \) is polynomial in the size of \( I_X \).

Proof.

\( R \) is a polynomial-time algorithm and hence on input \( I_X \) of size \( |I_X| \) it runs in time \( p(|I_X|) \) for some polynomial \( p() \).

\( I_Y \) is the output of \( R \) on input \( I_X \).

\( R \) can write at most \( p(|I_X|) \) bits and hence \( |I_Y| \leq p(|I_X|) \).
Proposition

Let \( R \) be a polynomial-time reduction from \( X \) to \( Y \). Then for any instance \( I_X \) of \( X \), the size of the instance \( I_Y \) of \( Y \) produced from \( I_X \) by \( R \) is polynomial in the size of \( I_X \).

Proof.

\( R \) is a polynomial-time algorithm and hence on input \( I_X \) of size \( |I_X| \) it runs in time \( p(|I_X|) \) for some polynomial \( p() \).

\( I_Y \) is the output of \( R \) on input \( I_X \).

\( R \) can write at most \( p(|I_X|) \) bits and hence \( |I_Y| \leq p(|I_X|) \).

Note: Converse is not true. A reduction need not be polynomial-time even if output of reduction is of size polynomial in its input.
A polynomial time reduction from a *decision* problem $X$ to a *decision* problem $Y$ is an *algorithm* $A$ that has the following properties:

2. $A$ runs in time polynomial in $|I_X|$. This implies that $|I_Y|$ (size of $I_Y$) is polynomial in $|I_X|$.
3. Answer to $I_X$ YES *iff* answer to $I_Y$ is YES.

**Proposition**

If $X \leq_p Y$ then a polynomial time algorithm for $Y$ implies a polynomial time algorithm for $X$. 
**Transitivity of Reductions**

**Proposition**

\[ X \leq_P Y \text{ and } Y \leq_P Z \implies X \leq_P Z. \]

**Note:** \( X \leq_P Y \) does not imply that \( Y \leq_P X \) and hence it is very important to know the FROM and TO in a reduction.

To prove \( X \leq_P Y \) you need to show a reduction FROM \( X \) TO \( Y \). That is, show that an algorithm for \( Y \) implies an algorithm for \( X \).
Vertex Cover

Given a graph $G = (V, E)$, a set of vertices $S$ is:
Vertex Cover

Given a graph $G = (V, E)$, a set of vertices $S$ is:

1. A **vertex cover** if every $e \in E$ has at least one endpoint in $S$. 
Vertex Cover

Given a graph $G = (V, E)$, a set of vertices $S$ is:

1. A vertex cover if every $e \in E$ has at least one endpoint in $S$. 
Given a graph $G = (V, E)$, a set of vertices $S$ is:

1. A **vertex cover** if every $e \in E$ has at least one endpoint in $S$. 

![Graph Diagram]

Miller, Hassanieh (UIUC)  
CS374  24  24 / 54  
Spring 2020
Vertex Cover

Given a graph $G = (V, E)$, a set of vertices $S$ is:

1. A **vertex cover** if every $e \in E$ has at least one endpoint in $S$. 
The **Vertex Cover** Problem

Problem (**Vertex Cover**)

**Input:** A graph $G$ and integer $k$.

**Goal:** Is there a vertex cover of size $\leq k$ in $G$?
The **Vertex Cover** Problem

**Problem (Vertex Cover)**

**Input:** A graph $G$ and integer $k$.

**Goal:** Is there a vertex cover of size $\leq k$ in $G$?

Can we relate **Independent Set** and **Vertex Cover**?
Proposition

Let \( G = (V, E) \) be a graph. \( S \) is an independent set if and only if \( V \setminus S \) is a vertex cover.

Proof.

\((\Rightarrow)\) Let \( S \) be an independent set

1. Consider any edge \( uv \in E \).
2. Since \( S \) is an independent set, either \( u \notin S \) or \( v \notin S \).
3. Thus, either \( u \in V \setminus S \) or \( v \in V \setminus S \).
4. \( V \setminus S \) is a vertex cover.

\((\Leftarrow)\) Let \( V \setminus S \) be some vertex cover:

1. Consider \( u, v \in S \)
2. \( uv \) is not an edge of \( G \), as otherwise \( V \setminus S \) does not cover \( uv \).
3. \( \implies S \) is thus an independent set.
Independent Set $\leq_P$ Vertex Cover

1. Let $G$: graph with $n$ vertices, and an integer $k$ be an instance of the Independent Set problem.
Independent Set $\leq_p$ Vertex Cover

1. $G$: graph with $n$ vertices, and an integer $k$ be an instance of the Independent Set problem.

2. $G$ has an independent set of size $\geq k$ iff $G$ has a vertex cover of size $\leq n - k$
Independent Set $\leq_p$ Vertex Cover

1. $G$: graph with $n$ vertices, and an integer $k$ be an instance of the Independent Set problem.

2. $G$ has an independent set of size $\geq k$ iff $G$ has a vertex cover of size $\leq n - k$

3. $(G, k)$ is an instance of Independent Set, and $(G, n - k)$ is an instance of Vertex Cover with the same answer.
Independent Set $\leq_P$ Vertex Cover

1. $G$: graph with $n$ vertices, and an integer $k$ be an instance of the **Independent Set** problem.

2. $G$ has an independent set of size $\geq k$ iff $G$ has a vertex cover of size $\leq n - k$

3. $(G, k)$ is an instance of **Independent Set**, and $(G, n - k)$ is an instance of **Vertex Cover** with the same answer.

4. Therefore, **Independent Set $\leq_P$ Vertex Cover**. Also **Vertex Cover $\leq_P$ Independent Set**.
To prove that \( X \leq_P Y \) you need to give an algorithm \( A \) that:

1. Transforms an instance \( I_X \) of \( X \) into an instance \( I_Y \) of \( Y \).
2. Satisfies the property that answer to \( I_X \) is YES iff answer to \( I_Y \) is YES.
   - typical easy direction to prove: answer to \( I_Y \) is YES if answer to \( I_X \) is YES
   - typical difficult direction to prove: answer to \( I_X \) is YES if answer to \( I_Y \) is YES (equivalently answer to \( I_X \) is NO if answer to \( I_Y \) is NO).
3. Runs in \textit{polynomial} time.
Part III

The Satisfiability Problem (SAT)
Propositional Formulas

Definition

Consider a set of boolean variables $x_1, x_2, \ldots x_n$.

1. A **literal** is either a boolean variable $x_i$ or its negation $\neg x_i$.
2. A **clause** is a disjunction of literals. For example, $x_1 \lor x_2 \lor \neg x_4$ is a clause.
3. A **formula in conjunctive normal form** (CNF) is a propositional formula which is a conjunction of clauses.

   \[
   (x_1 \lor x_2 \lor \neg x_4) \land (x_2 \lor \neg x_3) \land x_5 \text{ is a CNF formula.}
   \]
Propositional Formulas

Definition

Consider a set of boolean variables $x_1, x_2, \ldots, x_n$.

1. A **literal** is either a boolean variable $x_i$ or its negation $\neg x_i$.
2. A **clause** is a disjunction of literals.
   For example, $x_1 \lor x_2 \lor \neg x_4$ is a clause.
3. A **formula in conjunctive normal form (CNF)** is a propositional formula which is a conjunction of clauses
   - $(x_1 \lor x_2 \lor \neg x_4) \land (x_2 \lor \neg x_3) \land x_5$ is a CNF formula.
4. A formula $\varphi$ is a **3CNF**:
   A CNF formula such that every clause has **exactly** 3 literals.
   - $(x_1 \lor x_2 \lor \neg x_4) \land (x_2 \lor \neg x_3 \lor x_1)$ is a 3CNF formula, but
   - $(x_1 \lor x_2 \lor \neg x_4) \land (x_2 \lor \neg x_3) \land x_5$ is not.
Satisfiability

Problem: **SAT**

**Instance:** A CNF formula $\varphi$.

**Question:** Is there a truth assignment to the variable of $\varphi$ such that $\varphi$ evaluates to true?

Problem: **3SAT**

**Instance:** A 3CNF formula $\varphi$.

**Question:** Is there a truth assignment to the variable of $\varphi$ such that $\varphi$ evaluates to true?
Satisfiability

**SAT**
Given a **CNF** formula $\varphi$, is there a truth assignment to variables such that $\varphi$ evaluates to true?

**Example**
1. $(x_1 \lor x_2 \lor \neg x_4) \land (x_2 \lor \neg x_3) \land x_5$ is satisfiable; take $x_1, x_2, \ldots, x_5$ to be all true
2. $(x_1 \lor \neg x_2) \land (\neg x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land (x_1 \lor x_2)$ is not satisfiable.

**3SAT**
Given a **3CNF** formula $\varphi$, is there a truth assignment to variables such that $\varphi$ evaluates to true?

(More on **2SAT** in a bit...)
Importance of **SAT** and **3SAT**

1. **SAT** and **3SAT** are basic constraint satisfaction problems.
2. Many different problems can reduced to them because of the simple yet powerful expressively of logical constraints.
3. Arise naturally in many applications involving hardware and software verification and correctness.
4. As we will see, it is a fundamental problem in theory of **NP-Completeness**.
Given two bits $x, z$ which of the following SAT formulas is equivalent to the formula $z = \overline{x}$:

(A) $(\overline{z} \lor x) \land (z \lor \overline{x})$.

(B) $(z \lor x) \land (\overline{z} \lor \overline{x})$.

(C) $(\overline{z} \lor x) \land (\overline{z} \lor \overline{x}) \land (\overline{z} \lor \overline{x})$.

(D) $z \oplus x$.

(E) $(z \lor x) \land (\overline{z} \lor \overline{x}) \land (z \lor \overline{x}) \land (\overline{z} \lor x)$. 
Given three bits $x, y, z$ which of the following SAT formulas is equivalent to the formula $z = x \land y$:

(A) $(\bar{z} \lor x \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(B) $(\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(C) $(\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(D) $(z \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(E) $(z \lor x \lor y) \land (z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y) \land (\bar{z} \lor x \lor \bar{y}) \land (\bar{z} \lor \bar{x} \lor y) \land (\bar{z} \lor \bar{x} \lor \bar{y})$.
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</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>0</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>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</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>

Miller, Hassanieh (UIUC)
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor y$</th>
<th>$\overline{z} \lor x \lor y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting \( z = x \land y \) to 3SAT

<table>
<thead>
<tr>
<th>( z )</th>
<th>( x )</th>
<th>( y )</th>
<th>( z = x \land y )</th>
<th>( z \lor \bar{x} \lor y )</th>
<th>( \bar{z} \lor x \lor y )</th>
<th>( \bar{z} \lor \bar{x} \lor y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( \textbf{0} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<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>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor y$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<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>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$(z = x \land y)$$

$$\equiv$$

$$(z \lor \overline{x} \lor y) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor \overline{y})$$
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</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>0</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>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \overline{x} \lor \overline{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\overline{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\overline{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\overline{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \bar{x} \lor \bar{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$$\left( z = x \land y \right)$$

$$\equiv$$

$$\left( z \lor \bar{x} \lor \bar{y} \right) \land \left( \bar{z} \lor x \lor y \right) \land \left( \bar{z} \lor x \lor \bar{y} \right) \land \left( \bar{z} \lor \bar{x} \lor y \right)$$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $\left( (x \lor y) \land (x \lor \overline{y}) \right) = x$, we have that:
   
   1. $\left( \overline{z} \lor x \lor u \right) \land \left( \overline{z} \lor x \lor \overline{y} \right) = \left( \overline{z} \lor x \right)$
   2. $\left( \overline{z} \lor x \lor y \right) \land \left( \overline{z} \lor \overline{x} \lor y \right) = \left( \overline{z} \lor y \right)$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:

   1. $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   2. $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula

   $$\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)$$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   
   1. $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   2. $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula
   
   $\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)$

   is equivalent to

   $\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1. $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   2. $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula
   \[
   \psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)
   \]
   is equivalent to
   \[
   \psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)
   \]

**Lemma**

\[
(z = x \land y) \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)
\]
Given three bits $x, y, z$ which of the following SAT formulas is equivalent to the formula $z = x \lor y$:

(A) $(\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(B) $(\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(C) $(z \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y})$.

(D) $(z \lor x \lor y) \land (z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y) \land (\bar{z} \lor x \lor \bar{y}) \land (\bar{z} \lor \bar{x} \lor y) \land (\bar{z} \lor \bar{x} \lor \bar{y})$.

(E) $(\bar{z} \lor x \lor y) \land (z \lor \bar{x} \lor y) \land (z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor \bar{y})$. 
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>z</th>
<th>x</th>
<th>y</th>
<th>$z = x \lor y$</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>0</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$z \lor x \lor \overline{y}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$z \lor \overline{x} \lor y$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \overline{x} \lor \overline{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\overline{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting \( z = x \lor y \) to 3SAT

<table>
<thead>
<tr>
<th>( z )</th>
<th>( x )</th>
<th>( y )</th>
<th>( z = x \lor y )</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( z \lor x \lor \overline{y} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( z \lor \overline{x} \lor y )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( z \lor \overline{x} \lor \overline{y} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \overline{z} \lor x \lor y )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[
\left( z = x \lor y \right) \\
\equiv \\
\left( z \lor x \lor \overline{y} \right) \land \left( z \lor \overline{x} \lor y \right) \land \left( z \lor \overline{x} \lor \overline{y} \right) \land \left( \overline{z} \lor x \lor y \right)
\]
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

\[
(z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y)
\]

Using that \((x \lor y) \land (x \lor \overline{y}) = x\), we have that:

\[
(z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y)
\]
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

$$
(z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y)
$$

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1. $(z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) = z \lor \overline{y}$.
   2. $(z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{x}$
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

\[ (z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \]

1. Using that \((x \lor y) \land (x \lor \overline{y}) = x\), we have that:
   1. \((z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) = z \lor \overline{y}\).
   2. \((z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{x}\)

2. Using the above two observation, we have the following.
Converting \( z = x \lor y \) to 3SAT

Simplify further if you want to

\[
(z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y)
\]

1. Using that \((x \lor y) \land (x \lor \overline{y}) = x\), we have that:

1. \((z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) = z \lor \overline{y}\).

2. \((z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{x}\)

2. Using the above two observation, we have the following.

Lemma

The formula \( z = x \lor y \) is equivalent to the CNF formula

\[
(z = x \lor y) \equiv (z \lor \overline{y}) \land (z \lor \overline{x}) \land (\overline{z} \lor x \lor y)
\]
**SAT \leq_p 3SAT**

How **SAT** is different from **3SAT**?

In **SAT** clauses might have arbitrary length: 1, 2, 3, \ldots variables:

\[
(x \lor y \lor z \lor w \lor u) \land (\neg x \lor \neg y \lor \neg z \lor w \lor u) \land (\neg x)
\]

In **3SAT** every clause must have exactly 3 different literals.
**SAT \leq_P 3SAT**

How **SAT** is different from **3SAT**?

In **SAT** clauses might have arbitrary length: 1, 2, 3, ... variables:

\[
(x \lor y \lor z \lor w \lor u) \land (\neg x \lor \neg y \lor \neg z \lor w \lor u) \land (\neg x)
\]

In **3SAT** every clause must have **exactly** 3 different literals.

To reduce from an instance of **SAT** to an instance of **3SAT**, we must make all clauses to have exactly 3 variables...

**Basic idea**

1. Pad short clauses so they have 3 literals.
2. Break long clauses into shorter clauses.
3. Repeat the above till we have a 3CNF.
3SAT \leq_p SAT

1. 3SAT \leq_p SAT.

2. Because...
   A 3SAT instance is also an instance of SAT.
Claim

\[ \text{SAT} \leq_P \text{3SAT}. \]
Claim

\[
\text{SAT} \leq_P \text{3SAT}.
\]

Given \( \varphi \) a SAT formula we create a 3SAT formula \( \varphi' \) such that

1. \( \varphi \) is satisfiable iff \( \varphi' \) is satisfiable.
2. \( \varphi' \) can be constructed from \( \varphi \) in time polynomial in \( |\varphi| \).
Claim

\( \text{SAT} \leq_p 3\text{SAT}. \)

Given \( \varphi \) a SAT formula we create a 3SAT formula \( \varphi' \) such that

1. \( \varphi \) is satisfiable iff \( \varphi' \) is satisfiable.
2. \( \varphi' \) can be constructed from \( \varphi \) in time polynomial in \(|\varphi|\).

Idea: if a clause of \( \varphi \) is not of length 3, replace it with several clauses of length exactly 3.
Reduction Ideas: clause with 2 literals

1. Case clause with 2 literals: Let $c = \ell_1 \lor \ell_2$. Let $u$ be a new variable. Consider

$$c' = (\ell_1 \lor \ell_2 \lor u) \land (\ell_1 \lor \ell_2 \lor \neg u).$$

2. Suppose $\varphi = \psi \land c$. Then $\varphi' = \psi \land c'$ is satisfiable iff $\varphi$ is satisfiable.
Reduction Ideas: clause with 1 literal

1. **Case clause with one literal:** Let \( c \) be a clause with a single literal (i.e., \( c = \ell \)). Let \( u, v \) be new variables. Consider

\[
c' = (\ell \lor u \lor v) \land (\ell \lor u \lor \neg v) \\
    \land (\ell \lor \neg u \lor v) \land (\ell \lor \neg u \lor \neg v).
\]

2. Suppose \( \varphi = \psi \land c \). Then \( \varphi' = \psi \land c' \) is satisfiable iff \( \varphi \) is satisfiable.
SAT $\leq_P$ 3SAT
A clause with more than 3 literals

Reduction Ideas: clause with more than 3 literals

1. Case clause with five literals: Let $c = \ell_1 \lor \ell_2 \lor \ell_3 \lor \ell_4 \lor \ell_5$. Let $u$ be a new variable. Consider

$$c' = (\ell_1 \lor \ell_2 \lor \ell_3 \lor u) \land (\ell_4 \lor \ell_5 \lor \neg u).$$

2. Suppose $\varphi = \psi \land c$. Then $\varphi' = \psi \land c'$ is satisfiable iff $\varphi$ is satisfiable.
**Reduction Ideas: clause with more than 3 literals**

1. **Case clause with $k > 3$ literals:** Let $c = l_1 \lor l_2 \lor \ldots \lor l_k$. Let $u$ be a new variable. Consider

   $$c' = (l_1 \lor l_2 \ldots l_{k-2} \lor u) \land (l_{k-1} \lor l_k \lor \neg u).$$

2. Suppose $\varphi = \psi \land c$. Then $\varphi' = \psi \land c'$ is satisfiable iff $\varphi$ is satisfiable.
For any boolean formulas $X$ and $Y$ and $z$ a new boolean variable. Then

$$X \lor Y \text{ is satisfiable}$$

if and only if, $z$ can be assigned a value such that

$$\left( X \lor z \right) \land \left( Y \lor \neg z \right) \text{ is satisfiable}$$

(with the same assignment to the variables appearing in $X$ and $Y$).
Let $c = \ell_1 \lor \cdots \lor \ell_k$. Let $u_1, \ldots, u_{k-3}$ be new variables. Consider

$$c' = \left(\ell_1 \lor \ell_2 \lor u_1\right) \land \left(\ell_3 \lor \neg u_1 \lor u_2\right) \land \left(\ell_4 \lor \neg u_2 \lor u_3\right) \land \cdots \land \left(\ell_{k-2} \lor \neg u_{k-4} \lor u_{k-3}\right) \land \left(\ell_{k-1} \lor \ell_k \lor \neg u_{k-3}\right).$$

Claim

$\varphi = \psi \land c$ is satisfiable iff $\varphi' = \psi \land c'$ is satisfiable.

Another way to see it — reduce size of clause by one:

$$c' = \left(\ell_1 \lor \ell_2 \ldots \lor \ell_{k-2} \lor u_{k-3}\right) \land \left(\ell_{k-1} \lor \ell_k \lor \neg u_{k-3}\right).$$
Example

\( \varphi = (\neg x_1 \lor \neg x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4 \lor x_1) \land (x_1) . \)

Equivalent form:

\( \psi = (\neg x_1 \lor \neg x_4 \lor z) \land (\neg x_1 \lor \neg x_4 \lor \neg z) \)
An Example

Example

\[ \varphi = \left( \neg x_1 \lor \neg x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \lor x_1 \right) \land \left( x_1 \right). \]

Equivalent form:

\[ \psi = \left( \neg x_1 \lor \neg x_4 \lor z \right) \land \left( \neg x_1 \lor \neg x_4 \lor \neg z \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right). \]
An Example

Example

\[ \varphi = \left( \neg x_1 \lor \neg x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \lor x_1 \right) \land \left( x_1 \right) . \]

Equivalent form:

\[ \psi = \left( \neg x_1 \lor \neg x_4 \lor z \right) \land \left( \neg x_1 \lor \neg x_4 \lor \neg z \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor y_1 \right) \land \left( x_4 \lor x_1 \lor \neg y_1 \right) . \]
An Example

Example

\[ \varphi = \left( \neg x_1 \lor \neg x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \lor x_1 \right) \land \left( x_1 \right) . \]

Equivalent form:

\[ \psi = \left( \neg x_1 \lor \neg x_4 \lor z \right) \land \left( \neg x_1 \lor \neg x_4 \lor \neg z \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor y_1 \right) \land \left( x_4 \lor x_1 \lor \neg y_1 \right) \land \left( x_1 \lor u \lor v \right) \land \left( x_1 \lor u \lor \neg v \right) \land \left( x_1 \lor \neg u \lor v \right) \land \left( x_1 \lor \neg u \lor \neg v \right) . \]
**ReduceSATTo3SAT(ϕ):**

// ϕ: CNF formula.
for each clause c of ϕ do
  if c does not have exactly 3 literals then
    construct c′ as before
  else
    c′ = c

ψ is conjunction of all c′ constructed in loop
return Solver3SAT(ψ)

**Correctness (informal)**

ϕ is satisfiable iff ψ is satisfiable because for each clause c, the new 3CNF formula c′ is logically equivalent to c.
What about 2SAT?

2SAT can be solved in polynomial time! (specifically, linear time!)

No known polynomial time reduction from SAT (or 3SAT) to 2SAT. If there was, then SAT and 3SAT would be solvable in polynomial time.

Why the reduction from 3SAT to 2SAT fails?

Consider a clause \((x \lor y \lor z)\). We need to reduce it to a collection of 2CNF clauses. Introduce a face variable \(\alpha\), and rewrite this as

\[
(x \lor y \lor \alpha) \land (\neg\alpha \lor z) \quad \text{(bad! clause with 3 vars)}
\]

or

\[
(x \lor \alpha) \land (\neg\alpha \lor y \lor z) \quad \text{(bad! clause with 3 vars)}
\]

(In animal farm language: 2SAT good, 3SAT bad.)
What about \textbf{2SAT}?

A challenging exercise: Given a \textbf{2SAT} formula show to compute its satisfying assignment...

(Hint: Create a graph with two vertices for each variable (for a variable $x$ there would be two vertices with labels $x = 0$ and $x = 1$). For every \textbf{2CNF} clause add two directed edges in the graph. The edges are implication edges: They state that if you decide to assign a certain value to a variable, then you must assign a certain value to some other variable.

Now compute the strong connected components in this graph, and continue from there...)