Pre-lecture brain teaser

Does this graph have a Hamiltonian cycle?

a Yes.

b No.

Answer: B. The outer cycle has two entrances/exits that aren’t next to each other always leaving out one or two vertices.
NP-Completeness of the following two problems.

- Hamiltonian cycle
- 3-coloring

Important: Understanding the problems and that they are hard.

Proofs and reductions will be sketchy and mainly to give a flavor.
Reduction from 3SAT to Hamiltonian Cycle
Directed Hamiltonian Cycle

**Input**  Given a directed graph $G = (V, E)$ with $n$ vertices.

**Goal**  Does $G$ have a Hamiltonian cycle?

- A Hamiltonian cycle is a cycle in the graph that visits every vertex in $G$ exactly once.
Directed Hamiltonian Cycle

Input  Given a directed graph $G = (V, E)$ with $n$ vertices.

Goal  Does $G$ have a Hamiltonian cycle?

• A Hamiltonian cycle is a cycle in the graph that visits every vertex in $G$ exactly once.
Is the following graph Hamiltonianan?

a. Yes.
b. No.

Answer: B. The outer cycle has two entrances/exits that aren’t next to each other always leaving out one or two vertices.
Directed Hamiltonian Cycle is NP-Complete

(DHC)

- Directed Hamiltonian Cycle is in \(NP\): exercise.
- **Hardness:** We will show the following.

\[3\text{-SAT} \leq_P \text{Directed Hamiltonian Cycle}\]

\[3\text{SAT} \Rightarrow_P \text{DHC}\]
Directed Hamiltonian Cycle is NP-Complete

- Directed Hamiltonian Cycle is in \( NP \): exercise.
- **Hardness:** We will show the following.

\[
\begin{align*}
\text{3-SAT} & \leq_p \text{Directed Hamiltonian Cycle (DHC)}
\end{align*}
\]

We have the following reduction diagram.
Given 3-SAT formula $\varphi$ create a graph $G_\varphi$ such that the following hold.

- $G_\varphi$ has a Hamiltonian cycle if and only if $\varphi$ is satisfiable.
- $G_\varphi$ should be constructible from $\varphi$ by a polynomial time algorithm $\mathcal{A}$.

Notation: $\varphi$ has $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses $C_1, C_2, \ldots, C_m$. 
• Viewing SAT: Assign values to $n$ variables, and each clauses has 3 ways in which it can be satisfied.
• Construct graph with $2^n$ Hamiltonian cycles, where each cycle corresponds to some boolean assignment.
• Then add more graph structure to encode constraints on assignments imposed by the clauses.
Need to create a graph from any arbitrary boolean assignment. Consider the following expression.

\[
f(x_1) = 1 \tag{1}
\]

\[
\begin{array}{c|c}
x_1 & f(x_1) \\
\hline
1 & 1 \\
0 & 1 \\
\end{array}
\]
Need to create a graph from any arbitrary boolean assignment. Consider the following expression.

\[ f(x_1) = 1 \]  

(1)

We create a cyclic graph that always has a Hamiltonian cycle.
Need to create a graph from any arbitrary boolean assignment. Consider the following expression.

\[ f(x_1) = 1 \]  \hspace{1cm} (1)

We create a cyclic graph that always has a Hamiltonian cycle.

But how do we encode the variable?
Need to create a graph from any arbitrary boolean assignment. Consider the following expression.

\[ f(x_1) = 1 \]

Maybe we can encode the variable \( x_1 \) in terms of the cycle direction.
Need to create a graph from any arbitrary boolean assignment. Consider the following expression.

\[ f(x_1) = 1 \]

Maybe we can encode the variable \( x_1 \) in terms of the cycle direction.

If \( x_1 = 1 \)  

If \( x_1 = 0 \)
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (2)

Maybe two circles? Now we need to connect them so that we have a single Hamiltonian cycle.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (2)

Maybe two circles? Now we need to connect them so that we have a single Hamiltonian cycle.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  

Now we need to connect them so that we have a single Hamiltonian path
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  

(3)

Now we need to connect them so that we have a single Hamiltonian path
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  

Would be nice to have a single start/stop node.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  

Would be nice to have a single start/stop node.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (5)

Getting a bit messy. Let’s reorganize as follows.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \] (5)

Getting a bit messy. Let’s reorganize as follows.
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (6)

This is how we encode variable assignments in a variable loop!
How do we handle clauses?

\[ f(x_1) = x_1 \]  \hspace{1cm} (7)

Let's go back to our one variable graph.
Reduction: Encoding idea III

How do we handle clauses?

\[ f(x_1) = x_1 \]  (8)

Add node for clause.
How do we handle clauses?

\[ f(x_1, x_2) = (x_1 \lor \overline{x_2}) \] (9)

What do we do if the clause has two literals?
Reduction: Encoding idea III

How do we handle clauses?

$$f(x_1, x_2) = (x_1 \lor \overline{x_2})$$

What do we do if the clause has two literals?
How do we handle clauses?

\[ f(x_1, x_2) = (x_1 \lor \overline{x}_2) \land (x_1 \lor \overline{x}_2) \]  \hspace{1cm} (10)

What if the expression has multiple clauses:

\[
\begin{align*}
  v_{1_1} & \quad v_{1_2} & \quad v_{1_3} & \quad v_{1_4} & \quad \cdots & \quad v_{1_{n-2}} & \quad v_{1_{n-1}} & \quad v_{1_n} \\
  \text{C}_1 & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
  x_1 & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
  \text{C}_2 & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
  v_{1_1} & \quad v_{1_2} & \quad v_{1_3} & \quad v_{1_4} & \quad \cdots & \quad v_{1_{n-2}} & \quad v_{1_{n-1}} & \quad v_{1_n} \\
  x_2 & \quad & \quad & \quad & \quad & \quad & \quad & \quad
\end{align*}
\]
The Reduction: Review I

- Traverse path $i$ from left to right iff $x_i$ is set to true
- Each path has $3(m + 1)$ nodes where $m$ is number of clauses in $\varphi$; nodes numbered from left to right (1 to $3m + 3$)
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge from vertex $3j$ and to vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge from vertex $3j + 1$ and to vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[
\begin{align*}
 x_1 \lor \neg x_2 \lor x_4 \\
 \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge from vertex $3j$ and to vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge from vertex $3j + 1$ and to vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[
x_1 \vee \neg x_2 \vee x_4
\]
\[
\neg x_1 \vee \neg x_2 \vee \neg x_3
\]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge from vertex $3j$ and to vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge from vertex $3j + 1$ and to vertex $3j$ if $\neg x_i$ appears in $C_j$.
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge from vertex $3j$ and to vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge from vertex $3j + 1$ and to vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[
x_1 \lor \neg x_2 \lor x_4
\]

\[
\neg x_1 \lor \neg x_2 \lor \neg x_3
\]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge \textit{from} vertex $3j$ and \textit{to} vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge \textit{from} vertex $3j + 1$ and \textit{to} vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[ x_1 \lor \neg x_2 \lor x_4 \]

\[ \neg x_1 \lor \neg x_2 \lor \neg x_3 \]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge \textit{from} vertex $3j$ and \textit{to} vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge \textit{from} vertex $3j + 1$ and \textit{to} vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[
x_1 \lor \neg x_2 \lor x_4 \quad \quad \quad \quad \quad \neg x_1 \lor \neg x_2 \lor \neg x_3
\]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge *from* vertex $3j$ and *to* vertex $3j + 1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge *from* vertex $3j + 1$ and *to* vertex $3j$ if $\neg x_i$ appears in $C_j$. 

\[
\begin{align*}
x_1 \lor \neg x_2 \lor x_4 \\
\neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]
Theorem
ϕ has a satisfying assignment iff Gϕ has a Hamiltonian cycle.

Based on proving the following two lemmas.

Lemma
If ϕ has a satisfying assignment then Gϕ has a Hamilton cycle.

Lemma
If Gϕ has a Hamilton cycle then ϕ has a satisfying assignment.
Lemma
If \( \varphi \) has a satisfying assignment then \( G_\varphi \) has a Hamilton cycle.

Proof.

Let \( a \) be the satisfying assignment for \( \varphi \). Define Hamiltonian cycle as follows.

- If \( a(x_i) = 1 \) then traverse path \( i \) from left to right.
- If \( a(x_i) = 0 \) then traverse path \( i \) from right to left.
- For each clause, path of at least one variable is in the “right” direction to splice in the node corresponding to clause.
Suppose $\Pi$ is a Hamiltonian cycle in $G_\varphi$.

**Definition**
We say $\Pi$ is *canonical* if for each clause vertex $c_j$ the edge of $\Pi$ entering $c_j$ and edge of $\Pi$ leaving $c_j$ are from the same path corresponding to some variable $x_i$. Otherwise $\Pi$ is *non-canonical* or emph cheating.
Suppose \( \Pi \) is a Hamiltonian cycle in \( G_\varphi \).

**Definition**
We say \( \Pi \) is *canonical* if for each clause vertex \( c_j \) the edge of \( \Pi \) entering \( c_j \) and edge of \( \Pi \) leaving \( c_j \) are from the same path corresponding to some variable \( x_i \). Otherwise \( \Pi \) is *non-canonical* or emphcheating.

**Lemma**
*Every Hamilton cycle in \( G_\varphi \) is canonical.*
Lemma
Every Hamilton cycle in $G_\varphi$ is canonical.

- If $\Pi$ enters $c_j$ (vertex for clause $C_j$) from vertex $3j$ on path $i$ then it must leave the clause vertex on edge to $3j + 1$ on the same path $i$.
  - If not, then only unvisited neighbor of $3j + 1$ on path $i$ is $3j + 2$.
  - Thus, we don’t have two unvisited neighbors (one to enter from, and the other to leave) to have a Hamiltonian Cycle.
- Similarly, if $\Pi$ enters $c_j$ from vertex $3j + 1$ on path $i$ then it must leave the clause vertex $c_j$ on edge to $3j$ on path $i$. 
Lemma
Any canonical Hamilton cycle in $G_\varphi$ corresponds to a satisfying truth assignment to $\varphi$.

Consider a canonical Hamilton cycle $\Pi$.

- For every clause vertex $c_j$, vertices visited immediately before and after $c_j$ are connected by an edge on same path corresponding to some variable $x_i$.
- We can remove $c_j$ from cycle, and get Hamiltonian cycle in $G - c_j$.
- Hamiltonian cycle from $\Pi$ in $G - \{c_1, \ldots c_m\}$ traverses each path in only one direction, which determines truth assignment.
- Easy to verify that this truth assignment satisfies $\varphi$. 
Hamiltonian cycle in undirected graph
Problem

**Input**  Given *undirected* graph $G = (V, E)$.

**Goal**  Does $G$ have a *Hamiltonian* cycle? That is, is there a cycle that visits every vertex exactly one (except start and end vertex)?
NP-Completeness

Theorem (UHC)

Hamiltonian cycle problem for undirected graphs is NP-Complete.

Proof.

• The problem is in NP; proof left as exercise.
• Hardness proved by reducing Directed Hamiltonian Cycle to this problem.

\[3SAT \not\leq_p \ DHC \not\leq_p \ UHC\]

We want to do now!

Already done!
Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian cycle iff $G'$ has Hamiltonian path.

Reduction

\[
\begin{array}{c}
\text{a} \\
v \\
\text{b} \\
\text{c} \\
\text{d}
\end{array}
\]
**Goal:** Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian cycle iff $G'$ has Hamiltonian path.

**Reduction**

- Replace each vertex $v$ by 3 vertices: $v_{in}$, $v$, and $v_{out}$.
Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian cycle iff $G'$ has Hamiltonian path.

Reduction

- Replace each vertex $v$ by 3 vertices: $v_{in}$, $v$, and $v_{out}$.
- A directed edge $(a, b)$ is replaced by edge $(a_{out}, b_{in})$. 
Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian cycle iff $G'$ has Hamiltonian path.

Reduction

- Replace each vertex $v$ by 3 vertices: $v_{in}$, $v$, and $v_{out}$.
- A directed edge $(a, b)$ is replaced by edge $(a_{out}, b_{in})$. 
Graph with cycle:

3 nodes
3 edges

3.3 nodes : 9 nodes

(2.3) + 3 edges : 9 edges
Reduction Sketch Example

Graph with cycle:

Graph without cycle:
Reduction: Wrapup

- The reduction is polynomial time: exercise.
- The reduction is correct: exercise.
Hamiltonian Path

Input  Given a graph $G = (V, E)$ with $n$ vertices

Goal  Does $G$ have a Hamiltonian path?

• A Hamiltonian path is a path in the graph that visits every vertex in $G$ exactly once.

Theorem

Directed Hamiltonian Path and Undirected Hamiltonian Path are NP-Complete.

Easy to modify the reduction from 3-SAT to Hamiltonian Cycle or do a reduction from Hamiltonian Cycle.

Implies that Longest Simple Path in a graph is NP-Complete.
Hamiltonian Path

**Input**  Given a graph \( G = (V, E) \) with \( n \) vertices

**Goal**  Does \( G \) have a Hamiltonian path?

- A Hamiltonian path is a path in the graph that visits every vertex in \( G \) exactly once.

**Theorem**  
*Directed Hamiltonian Path* and *Undirected Hamiltonian Path* are NP-Complete.

Easy to modify the reduction from 3-SAT to Hamiltonian Cycle or do a reduction from Hamiltonian Cycle.
Hamiltonian Path

Input  Given a graph $G = (V, E)$ with $n$ vertices

Goal  Does $G$ have a Hamiltonian path?

- A Hamiltonian path is a path in the graph that visits every vertex in $G$ exactly once.

Theorem

*Directed Hamiltonian Path* and *Undirected Hamiltonian Path* are NP-Complete.

Easy to modify the reduction from *3-SAT* to *Halitonian Cycle* or do a reduction from *Halitonian Cycle*.

Implies that *Longest Simple Path* in a graph is NP-Complete.
NP-Completeness of Graph Coloring
Problem: **Graph Coloring**

**Instance:** $G = (V, E)$: Undirected graph, integer $k$.

**Question:** Can the vertices of the graph be colored using $k$ colors so that vertices connected by an edge do not get the same color?
Problem: **3 Coloring**

**Instance:** $G = (V, E)$: Undirected graph.

**Question:** Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?
Problem: 3 Coloring

**Instance:** $G = (V, E)$: Undirected graph.

**Question:** Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?
**Observation:** If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$. Thus, $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

Graph 2-Coloring can be decided in polynomial time.

$G$ is 2-colorable iff $G$ is bipartite! There is a linear time algorithm to check if $G$ is bipartite using breadth first search.
Problems related to graph coloring (RIY)
Register Allocation
Assign variables to (at most) $k$ registers such that variables needed at the same time are not assigned to the same register.

Interference Graph
Vertices are variables, and there is an edge between two vertices, if the two variables are “live” at the same time.

Observations

• [Chaitin] Register allocation problem is equivalent to coloring the interference graph with $k$ colors.
• Moreover, 3-COLOR $\leq_P k$ – Register Allocation, for any $k \geq 3$. 
Given $n$ classes and their meeting times, are $k$ rooms sufficient?

Reduce to Graph $k$-Coloring problem.

Create graph $G$

- a node $v_i$ for each class $i$.
- an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict.

Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.
Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA).

- Break up a frequency range $[a, b]$ into disjoint bands of frequencies $[a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]$.
- Each cell phone tower (simplifying) gets one band.
- Constraint: nearby towers cannot be assigned the same band, otherwise signals will interfere.
Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA).

- Breakup a frequency range \([a, b]\) into disjoint bands of frequencies \([a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]\).
- Each cell phone tower (simplifying) gets one band.
- Constraint: nearby towers cannot be assigned same band, otherwise signals will interfere.

**Problem:** given \(k\) bands and some region with \(n\) towers, is there a way to assign the bands to avoid interference?

Can reduce to \(k\)-coloring by creating interference/conflict graph on towers.
Showing hardness of 3 COLORING
3-Coloring is NP-Complete

- **3-Coloring is in $\text{NP}$**.
  - Non-deterministically guess a 3-coloring for each node
  - Check if for each edge $(u, v)$, the color of $u$ is different from that of $v$. *(Exercise!)*

- **Hardness**: We will show $3\text{-SAT} \leq_p 3\text{-Coloring}$. 
Reduction Idea

Start with 3SAT formula (i.e., 3CNF formula) \( \varphi \) with \( n \) variables \( x_1, \ldots, x_n \) and \( m \) clauses \( C_1, \ldots, C_m \). Create graph \( G_\varphi \) such that \( G_\varphi \) is 3-colorable iff \( \varphi \) is satisfiable.

- need to establish truth assignment for \( x_1, \ldots, x_n \) via colors for some nodes in \( G_\varphi \).
- create triangle with node True, False, Base.
- for each variable \( x_i \) two nodes \( v_i \) and \( \overline{v}_i \) connected in a triangle with common Base.
- If graph is 3-colored, either \( v_i \) or \( \overline{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \).
- Need to add constraints to ensure clauses are satisfied (next phase).
Reduction Idea I - Simple 3-color gadget

We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.

\[ \Phi : C_1 \land C_2 \land \ldots \land C_m \]

E.g.: \[ C_1 = x_1 \lor x_2 \lor x_3 \]

True if at least one of the literals is true!
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.

Let’s start off with the simplest SAT we can think of:

\[ f(x_1, x_2) = (x_1 \lor x_2) \]  \hspace{1cm} (11)
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.

Let’s start off with the simplest SAT we can think of:

\[ f(x_1, x_2) = (x_1 \lor x_2) \]  \hspace{1cm} (11)

Assume green=true and red=false,
Reduction Idea I - Simple 3-color gadget

We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true
- Not 3-colorable if none of the literals are true

Let’s try some stuff:
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true
- Not 3-colorable if none of the literals are true

Seems to work:
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.

How do we do the same thing for 3 variables?

\[ f(x_1, x_2, x_3) = (x_1 \lor x_2 \lor x_3) \] (12)
We want to create a gadget that:

- Is 3 colorable if at least one of the literals is true.
- Not 3-colorable if none of the literals are true.

How do we do the same thing for 3 variables?

\[ f(x_1, x_2, x_3) = (x_1 \lor x_2 \lor x_3) \]  

Assume green=true and red=false.
You are given three colors: red, green and blue. Can the following graph be three colored in a valid way (assuming that some of the nodes are already colored as indicated).

a Yes.
b No.
3 color this gadget.

You are given three colors: red, green and blue. Can the following graph be three colored in a valid way (assuming that some of the nodes are already colored as indicated).

a Yes.
b No.
3-coloring of the clause gadget

(i)

(ii)

(iii)

FFF - BAD

FTF

TTF

TFF

TFT

TTT
Next we need a gadget that assigns literals. Our previously constructed gadget assumes the following.

- All literals are either red or green.
- Need to limit graph so only $x_1$ or $\overline{x_1}$ is green. Other must be red.
Review Clause Satisfiability Gadget

For each clause $C_j = (a \lor b \lor c)$, create a small gadget.

- Gadget graph connects to nodes corresponding to $a$, $b$, $c$.
- Needs to implement OR.

OR-gadget-graph:
Property: If $a, b, c$ are colored False in a 3-coloring then output node of OR-gadget has to be colored False.

Property: If one of $a, b, c$ is colored True then OR-gadget can be 3-colored such that output node of OR-gadget is colored True.
• Create triangle with nodes True, False, Base.
• For each variable $x_i$ two nodes $v_i$ and $\overline{v}_i$ connected in a triangle with common Base.
• For each clause $C_j = (a \lor b \lor c)$, add OR-gadget graph with input nodes $a, b, c$ and connect output node of gadget to both False and Base.
Lemma
No legal 3-coloring of above graph (with coloring of nodes $T, F, B$ fixed) in which $a, b, c$ are colored False. If any of $a, b, c$ are colored True then there is a legal 3-coloring of above graph.
Example

\[ \varphi = (u \lor \neg v \lor w) \land (v \lor x \lor \neg y) \]

Variable and negation have complementary colours. Literals get colour T or F.
\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- If \( x_i \) is assigned True, color \( v_i \) True and \( \overline{v}_i \) False.

G \( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- If \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment.

- Consider any clause \( C_j = (a \lor b \lor c) \). It cannot be that all \( a \), \( b \), \( c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- If \( x_i \) is assigned True, color \( v_i \) True and \( \overline{v}_i \) False.
- For each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- If \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment.
- Consider any clause \( C_j = (a \lor b \lor c) \). It cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Correctness of Reduction

$\varphi$ is satisfiable implies $G_\varphi$ is 3-colorable

- If $x_i$ is assigned True, color $v_i$ True and $\overline{v_i}$ False.
- For each clause $C_j = (a \lor b \lor c)$ at least one of $a, b, c$ is colored True. OR-gadget for $C_j$ can be 3-colored such that output is True.
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- If \( x_i \) is assigned True, color \( v_i \) True and \( \overline{v}_i \) False.
- For each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- If \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment.
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- If \( x_i \) is assigned True, color \( v_i \) True and \( \overline{v_i} \) False.
- For each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- If \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment.
- Consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Graph generated in reduction from 3SAT to 3COLOR
Circuit-Sat Problem
A circuit is a directed *acyclic* graph with

- **Input** vertices (without incoming edges) labeled with 0, 1 or a distinct variable.
- Every other vertex is labeled \(\lor, \land\) or \(\neg\).
- Single node **output** vertex with no outgoing edges.
A circuit is a directed *acyclic* graph with

- **Input** vertices (without incoming edges) labeled with 0, 1 or a distinct variable.
- Every other vertex is labeled $\lor$, $\land$ or $\neg$.
- Single node **output** vertex with no outgoing edges.
A circuit is a directed *acyclic* graph with

- **Input** vertices (without incoming edges) labeled with 0, 1 or a distinct variable.
- Every other vertex is labeled $\vee$, $\wedge$ or $\neg$.
- Single node **output** vertex with no outgoing edges.
**Definition (Circuit Satisfaction (CSAT).)**
Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?
Definition (Circuit Satisfaction (CSAT).)
Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

Lemma
CSAT is in NP.

- **Certificate**: Assignment to input variables.
- **Certifier**: Evaluate the value of each gate in a topological sort of DAG and check the output gate value.
CNF formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas.
CNF formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas.

However they are equivalent in terms of polynomial-time solvability.

Theorem
\[ SAT \leq_p 3SAT \leq_p CSAT. \]

Theorem
\[ CSAT \leq_p SAT \leq_p 3SAT. \]
Converting a CNF formula into a Circuit

Given 3CNF formula $\varphi$ with $n$ variables and $m$ clauses, create a Circuit $C$.

- Inputs to $C$ are the $n$ boolean variables $x_1, x_2, \ldots, x_n$
- Use NOT gate to generate literal $\neg x_i$ for each variable $x_i$
- For each clause $(\ell_1 \lor \ell_2 \lor \ell_3)$ use two OR gates to mimic formula
- Combine the outputs for the clauses using AND gates to obtain the final output
Example: $3\text{SAT} \leq^p \text{CSAT}$

\[ \varphi = (x_1 \lor \lnot x_3 \lor x_4) \land (x_1 \lor \lnot x_2 \lor \lnot x_3) \land (\lnot x_2 \lor \lnot x_3 \lor x_4) \]
Example: $3\text{SAT} \leq_p \text{CSAT}$
Example: \(3\text{SAT} \leq_p \text{CSAT}\)

\[
\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)
\]
Example: $3\text{SAT} \leq_P \text{CSAT}$

$$\varphi = (x_1 \lor \neg x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4)$$
Example: $3\text{SAT} \leq_p \text{CSAT}$

\[ \varphi = \left( x_1 \lor \neg x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right) \]
Example: $3\text{SAT} \leq_p \text{CSAT}$

$$\varphi = \left( x_1 \lor \neg x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
Example: $3\text{SAT} \leq_p \text{CSAT}$

$$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
Example: $3\text{SAT} \leq_{P} \text{CSAT}$

$$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
Converting a circuit to a SAT formula

What will converting a circuit to a SAT formula prove?
Converting a circuit to a SAT formula

What will converting a circuit to a SAT formula prove?

But first we need to look back at a gadget!
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>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>1</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>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>

The truth table shows the values of $z$ for all possible combinations of $x$ and $y$.
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>$z \lor \bar{x} \lor \bar{y}$</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>$z \lor \overline{x} \lor y$</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>
<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>$\overline{z} \lor x \lor \overline{y}$</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>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor y$</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>
<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>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor y$</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>
<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

\[
\begin{array}{cccc|cccc}
Z & X & Y & z = x \land y & z \lor \overline{x} \lor \overline{y} & \overline{z} \lor x \lor y & \overline{z} \lor x \lor \overline{y} & \overline{z} \lor \overline{x} \lor y \\
\hline
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\left( z = x \land y \right) \\
\equiv \\
\left( z \lor \overline{x} \lor \overline{y} \right) \land \left( \overline{z} \lor x \lor y \right) \land \left( \overline{z} \lor x \lor \overline{y} \right) \land \left( \overline{z} \lor \overline{x} \lor y \right)
\]
Lemma
The following identities hold:

\[
\begin{align*}
\text{• } z = \overline{x} & \equiv (z \lor x) \land (\overline{z} \lor \overline{x}). \\
\text{• } (z = x \lor y) & \equiv (z \lor \overline{y}) \land (z \lor \overline{x}) \land (\overline{z} \lor x \lor y) \\
\text{• } (z = x \land y) & \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)
\end{align*}
\]
Converting a circuit into a CNF formula

(A) Input circuit

(B) Label the nodes.
Converting a circuit into a CNF formula

(B) Label the nodes.
(C) Introduce var for each node.
Converting a circuit into a CNF formula

(C) Introduce var for each node.

(D) Write a sub-formula for each variable that is true if the var is computed correctly.

\[
x_k \quad \text{(Demand a sat' assignment!)} \\
x_k = x_i \land x_j \\
x_j = x_g \land x_h \\
x_i = \neg x_f \\
x_h = x_d \lor x_e \\
x_g = x_b \lor x_c \\
x_f = x_a \land x_b \\
x_d = 0 \\
x_a = 1
\]
Converting a circuit into a CNF formula

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$x_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k = x_i \land x_j$</td>
<td>$(\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \land (x_k \lor \neg x_i \lor \neg x_j)$</td>
</tr>
<tr>
<td>$x_j = x_g \land x_h$</td>
<td>$(\neg x_j \lor x_g) \land (\neg x_j \lor x_h) \land (x_j \lor \neg x_g \lor \neg x_h)$</td>
</tr>
<tr>
<td>$x_i = \neg x_f$</td>
<td>$(x_i \lor x_f) \land (\neg x_i \lor \neg x_f)$</td>
</tr>
<tr>
<td>$x_h = x_d \lor x_e$</td>
<td>$(x_h \lor \neg x_d) \land (x_h \lor \neg x_e) \land (\neg x_h \lor x_d \lor x_e)$</td>
</tr>
<tr>
<td>$x_g = x_b \lor x_c$</td>
<td>$(x_g \lor \neg x_b) \land (x_g \lor \neg x_c) \land (\neg x_g \lor x_b \lor x_c)$</td>
</tr>
<tr>
<td>$x_f = x_a \land x_b$</td>
<td>$(\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (x_f \lor \neg x_a \lor \neg x_b)$</td>
</tr>
<tr>
<td>$x_d = 0$</td>
<td>$\neg x_d$</td>
</tr>
<tr>
<td>$x_a = 1$</td>
<td>$x_a$</td>
</tr>
</tbody>
</table>
Converting a circuit into a CNF formula

We got a CNF formula that is satisfiable if and only if the original circuit is satisfiable.
Reduction: \( CSAT \leq_p SAT \)

- For each gate (vertex) \( v \) in the circuit, create a variable \( x_v \)
- Case \( \neg \): \( v \) is labeled \( \neg \) and has one incoming edge from \( u \) (so \( x_v = \neg x_u \)). In SAT formula generate, add clauses \((x_u \lor x_v), (\neg x_u \lor \neg x_v)\). Observe that

\[
x_v = \neg x_u \text{ is true } \iff \begin{align*}
(x_u & \lor x_v) \\
(\neg x_u & \lor \neg x_v)
\end{align*} \text{ both true.}
\]
Reduction: \( \text{CSAT} \leq_P \text{SAT} \)

- **Case \( \lor \):** So \( x_v = x_u \lor x_w \). In \( \text{SAT} \) formula generated, add clauses \((x_v \lor \neg x_u), (x_v \lor \neg x_w)\), and \((\neg x_v \lor x_u \lor x_w)\). Again, observe that

\[
(x_v = x_u \lor x_w) \text{ is true} \iff (x_v \lor \neg x_u), \quad (x_v \lor \neg x_w), \quad \text{all true.}
\]

\[
(\neg x_v \lor x_u \lor x_w)
\]
Reduction: \( \text{CSAT} \leq_p \text{SAT} \)

- Case \( \land \): So \( x_v = x_u \land x_w \). In SAT formula generated, add clauses \((\neg x_v \lor x_u), (\neg x_v \lor x_w)\), and \((x_v \lor \neg x_u \lor \neg x_w)\). Again observe that

\[
x_v = x_u \land x_w \text{ is true } \iff \ (\neg x_v \lor x_u), (\neg x_v \lor x_w), (x_v \lor \neg x_u \lor \neg x_w) \text{ all true.}
\]
Reduction: \( CSAT \leq_p SAT \)

- If \( v \) is an input gate with a fixed value then we do the following. If \( x_v = 1 \) add clause \( x_v \). If \( x_v = 0 \) add clause \( \neg x_v \)
- Add the clause \( x_v \) where \( v \) is the variable for the output gate
Correctness of Reduction

Need to show circuit $C$ is satisfiable iff $\varphi_C$ is satisfiable

$\Rightarrow$ Consider a satisfying assignment $a$ for $C$
  - Find values of all gates in $C$ under $a$
  - Give value of gate $v$ to variable $x_v$; call this assignment $a'$
  - $a'$ satisfies $\varphi_C$ (exercise)

$\Leftarrow$ Consider a satisfying assignment $a$ for $\varphi_C$
  - Let $a'$ be the restriction of $a$ to only the input variables
  - Value of gate $v$ under $a'$ is the same as value of $x_v$ in $a$
  - Thus, $a'$ satisfies $C$