Pre-lecture brain teaser

Does this graph have a hamiltonian cycle?

a Yes.
b No.
ECE-374-B: Lecture 22 - Lots of NP-Complete reductions

Instructor: Nickvash Kani
April 13, 2023
University of Illinois at Urbana-Champaign
Today

NP-Completeness of two problems:
  • Hamiltonian Cycle
  • 3-Color

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?

1. 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.

![Directed Graph Example](image-url)
Is the following graph Hamiltonian?

a. Yes.
b. No.
Directed Hamiltonian Cycle is NP-Complete

• Directed Hamiltonian Cycle is in \( NP \): exercise
• **Hardness:** We will show
  \[ 3\text{-SAT} \leq_p \text{Directed Hamiltonian Cycle} \]
Directed Hamiltonian Cycle is NP-Complete

- Directed Hamiltonian Cycle is in \( NP \): exercise
- **Hardness:** We will show
  \( 3\text{-SAT} \leq_P \text{Directed Hamiltonian Cycle} \)
Given 3-SAT formula $\varphi$ create a graph $G_\varphi$ such that

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

**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 clause 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 expression:

\[ f(x_1) = 1 \]  \hspace{1cm} (1)
Need to create a graph from any arbitrary boolean assignment. Consider the expression:

$$f(x_1) = 1$$ (1)

We create a cyclic graph that always has a hamiltonian:
Need to create a graph from any arbitrary boolean assignment. Consider the expression:

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

We create a cyclic graph that always has a hamiltonian:

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

$$f(x_1) = 1$$ \hspace{1cm} (2)

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 expression:

\[ f(x_1) = 1 \]  

(2)

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

If \( x_1 = 1 \)
How do we encode multiple variables?

\[ f(x_1, x_2) = 1 \] (3)

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

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (3)

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

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

(4)

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

\[ f(x_1, x_2) = 1 \] (4)

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

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

(5)

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} (6)

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

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

(6)

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

\[ f(x_1, x_2) = 1 \]  \hspace{1cm} (7)

This is how we encode variable assignments in a variable loop!
Reduction: Encoding idea III

How do we handle clauses?

\[ f(x_1) = x_1 \]  \hspace{1cm} (8)

Let's go back to our one variable graph:
How do we handle clauses?

\[ f(x_1) = x_1 \]  

Add node for clause:
How do we handle clauses?

\[ f(x_1, x_2) = (x_1 \lor \overline{x_2}) \] (10)

What do we do if the clause has two literals:

![Diagram](image-url)
How do we handle clauses?

\[ f(x_1, x_2) = (x_1 \lor \overline{x_2}) \quad (10) \]

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

How do we handle clauses?

\[ f(x_1, x_2) = (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_2) \quad (11) \]

What if the expression has multiple clauses:
• 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 $C_j$.  

\[ x_1 \lor \neg x_2 \lor x_4 \]
\[ \neg x_1 \lor \neg x_2 \lor \neg x_3 \]
Add vertex $c_j$ for clause $C_j$. $c_j$ has edge 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 $C_j$.  

\[ x_1 \lor \neg x_2 \lor x_4 \quad \neg x_1 \lor \neg x_2 \lor \neg x_3 \]
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 $C_j$. 

\[
\begin{align*}
x_1 \lor \neg x_2 \lor x_4 & \\
\neg x_1 \lor \neg x_2 \lor \neg x_3 & 
\end{align*}
\]
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 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 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 $C_j$. 

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

\[
\neg x_1 \lor \neg x_2 \lor \neg x_3
\]
Correctness Proof

**Theorem**
\( \varphi \) has a satisfying assignment iff \( G_\varphi \) has a Hamiltonian cycle.

Based on proving following two lemmas.

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

**Lemma**
If \( G_\varphi \) has a Hamilton cycle then \( \varphi \) 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 emphcheating.
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 cheating.

**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
Hamiltonian Cycle in Undirected Graphs

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)?
Theorem

*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
Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian Path iff $G'$ has Hamiltonian path

Reduction
Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian Path iff $G'$ has Hamiltonian path.

Reduction

- Replace each vertex $v$ by 3 vertices: $v_{in}$, $v$, and $v_{out}$.
Reduction Sketch

Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian Path 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 Path 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:
**Reduction Sketch Example**

Graph **with** cycle:

Graph **without** cycle:
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
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*
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
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
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 $k \geq 3$. 
Class Room Scheduling

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
Frequency Assignments in Cellular Networks

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
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 **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 $v$.

- **Hardness**: We will show $3$-SAT $\leq_P$ 3-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 $\bar{v}_i$ connected in a triangle with common Base
- If graph is 3-colored, either $v_i$ or $\bar{v}_i$ 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
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} (12)
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 start off with the simplest SAT we can think of:

\[ f(x_1, x_2) = (x_1 \lor x_2) \]  \hspace{1cm} (12)

Assume green=true and red=false,
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:
Reduction Idea 1 - 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

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
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

How do we do the same thing for 3 variables?:

$$f(x_1, x_2, x_3) = (x_1 \lor x_2 \lor x_3)$$ (13)
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?:

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

(13)

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

<table>
<thead>
<tr>
<th>FFF - BAD</th>
<th>FFT</th>
<th>FTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTT</td>
<td>TFF</td>
<td>TFT</td>
</tr>
<tr>
<td>TTT</td>
<td>TTF</td>
<td>TTT</td>
</tr>
</tbody>
</table>
Next we need a gadget that assigns literals. Our previously constructed gadget assumes:

- 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 $C_j = (a \vee b \vee c)$, create a small gadget graph

- 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
\[ \varphi = (u \lor \neg v \lor w) \land (v \lor x \lor \neg y) \]
$\phi$ is satisfiable implies $G_\phi$ is 3-colorable

- if $x_i$ is assigned True, color $v_i$ True and $\overline{v_i}$ 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.
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 \( \bar{v}_i \) 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 $\lor$, $\land$ or $\lnot$.
- 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.
Circuit SAT vs SAT

CNF formulas are a rather restricted form of Boolean formulas.

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

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: \(3\text{SAT} \leq_p \text{CSAT}\)

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

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

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

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

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

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

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

$$\varphi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \land (x_1 \lor x_3 \lor x_4)$$
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 0 0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td></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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>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>$z \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</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>
</tr>
<tr>
<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>
</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>
</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>0</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>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>
</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 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 0 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<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 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 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>

62
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>
<th>$\bar{z} \lor x \lor y$</th>
<th>$\bar{z} \lor x \lor \bar{y}$</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>
<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**

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

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

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

(A) Input circuit

Output: $\land$

\[ \neg \]

\[ \land \]

\[ \lor \]

\[ \land \]

\[ \lor \]

Inputs: 1, ?, ?, 0, ?

(B) Label the nodes.

Output: $\land, k$

\[ \neg, i \]

\[ \land, j \]

\[ \land, f \]

\[ \lor, g \]

\[ \lor, h \]

Inputs: 1, a, ?, b, ?, c, d, ?

\[ \land \]

\[ \lor \]
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.

Output: $\land, k \ x_k$

$x_i \ \neg, i

$x_f \ \land, f

$\lor, g \ x_g$

$\lor, h \ x_h$

 Inputs

$x_a \ 1, a \ x_b \ ? , b \ x_c \ ? , c \ x_d \ 0, d \ x_e \ ? , e$

$x_k$ (Demand a sat’ assignment!)

$x_k = x_i \land x_j$

$x_j = x_g \land x_h$

$x_i = \neg x_f$

$x_h = x_d \lor x_e$

$x_g = x_b \lor x_c$

$x_f = x_a \land x_b$

$x_d = 0$

$x_a = 1$

(D) Write a sub-formula for each variable that is true if the var is computed correctly.
### 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: \( \text{CSAT} \leq_p \text{SAT} \)

- 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

\[
x_v = \neg x_u \text{ is true } \iff (x_u \lor x_v) \land (\neg x_u \lor \neg x_v) \text{ both true.}
\]
Reduction: $CSAT \leq_P SAT$

- **Case $\lor$:** So $x_v = x_u \lor x_w$. In $SAT$ 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

$$\left( x_v = x_u \lor x_w \right) \text{ is true} \iff (x_v \lor \neg x_u), \quad (x_v \lor \neg x_w), \quad \text{all true.}$$

$$\left( \neg x_v \lor x_u \lor x_w \right)$$
Reduction: $\text{CSAT} \leq_p \text{SAT}$

- **Case $\land$:** So $x_v = x_u \land x_w$. In $\text{SAT}$ 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

$$x_v = x_u \land x_w \text{ is true } \iff (\neg x_v \lor x_u), \quad (\neg x_v \lor x_w), \quad \text{all true.} \quad (x_v \lor \neg x_u \lor \neg x_w)$$
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 $C$