We know that SAT is NP-complete which means that it is in NP-Hard. HALT is also in NP-Hard. Is SAT reducible to HALT?
CS/ECE-374: Lecture 27 - More NP-Complete reductions

Lecturer: Nickvash Kani
Chat moderator: Samir Khan
April 29, 2021

University of Illinois at Urbana-Champaign
Pre-lecture brain teaser

We know that SAT is NP-complete which means that it is in NP-Hard. HALT is also in NP-Hard. Is SAT reducible to HALT? Yes

- HALT is NP-Hard which for all $x \in NP$, $x \leq_p HALT$

Therefore $SAT \leq_p HALT$

- Show $SAT \leq_p HALT$

Reduction: Construct a TM $<M>$:
- Loop over all assignments to SAT
- halt if assignment is satisfied
Today

NP-Completeness of three problems:

- Undirected HC problem
- 3-Color Problem
- Circuit SAT

Important: understanding the problems and that they are hard.

Proofs and reductions will be sketchy and mainly to give a flavor
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)?
NP-Completeness

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

- 
- 
- 

![Graph Diagram](image-url)
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 $v_{out}$
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 $(a_{out}, b_{in})$

If there is a TC in the directed graph then there is one in the undirected graph.
Graph with cycle:
Graph with cycle:

Graph without cycle:
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?
Graph 3-Coloring

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?
Graph Coloring

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, $3\text{-COLOR} \leq_P k - \text{Register Allocation}$, 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
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

**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 intereference/conflict graph on towers.
Showing hardness of 3 COLORING
3-Coloring is NP-Complete

- **3-Coloring** is in NP. Building a poly certifier $O(E)$
  
  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 $\overline{v}_i$ 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)
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:

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

(1)
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} (1)

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:
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:
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?:

\[ f(x_1, x_2, x_3) = (x_1 \lor x_2 \lor x_3) \]  \hspace{1cm} (2)

Assume green=true and red=false,
3 color this gadget II

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

FFF - BAD

FFF

FFT

FTF

FTT

TFF

TFT

TTF

TTT
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.
Reduction Idea II - Literal Assignment II
For each clause $C_j = (a \lor b \lor c)$, create a small gadget graph

- gadget graph connects to nodes corresponding to $a, b, c$
- needs to implement OR

OR-gadget-graph:
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.
Reduction

- create triangle with nodes True, False, Base
- for each variable $x_i$ two nodes $v_i$ and $\bar{v}_i$ 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.

Palette

OR-gates

T
F
N
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
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 \( \overline{v_i} \) 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 $\lor$, $\land$ 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?
**CSAT:** Circuit Satisfaction

**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
\[ \text{SAT} \leq_p \text{3SAT} \leq_p \text{CSAT}. \]

Theorem
\[ \text{CSAT} \leq_p \text{SAT} \leq_p \text{3SAT}. \]
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

\[
\text{SAT} = (x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6)
\]
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: \(3\text{SAT} \leq_P \text{CSAT}\)

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

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

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

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

What will converting a circuit to a SAT formula prove?

\[
\text{Circuit-SAT} \leq_p \text{SAT}
\]

\[
\leq_{NP} \text{SAT} \leq_p \text{Circuit-SAT} \quad \text{shows} \quad \text{CSAT in NP-hard}
\]
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>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$z = x \lor y$

$[x, y, z] = [1, 0, 1]
[0, 0, 0]$
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>1</td>
</tr>
<tr>
<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>
</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>$z \lor \overline{x} \lor \overline{y}$</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>
<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>$z \lor \overline{x} \lor \overline{y}$</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>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>$z \lor \overline{x} \lor \overline{y}$</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>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor y$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor y$</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>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>0</td>
<td>1</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>0</td>
<td>1</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>0</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>
<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>
<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>$z \lor \bar{x} \lor \bar{y}$</th>
<th>$\bar{z} \lor x \lor y$</th>
<th>$\bar{z} \lor \bar{x} \lor \bar{y}$</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>

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

$\equiv$

$\left( z \lor \bar{x} \lor \bar{y} \right) \land \left( \bar{z} \lor x \lor y \right) \land \left( \bar{z} \lor x \lor \bar{y} \right) \land \left( \bar{z} \lor \bar{x} \lor y \right)$
Lemma

*The following identities hold:*

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

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

(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

\[
x_k \land (\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \\
\land (x_k \lor \neg x_i \lor \neg x_j) \land (\neg x_j \lor x_g) \\
\land (\neg x_j \lor x_h) \land (x_j \lor \neg x_g \lor \neg x_h) \\
\land (x_i \lor x_f) \land (\neg x_i \lor \neg x_f) \\
\land (x_i \lor x_f) \land (x_i \lor \neg x_f) \\
\land (x_h \lor \neg x_d) \land (x_h \lor \neg x_e) \\
\land (\neg x_h \lor x_d \lor x_e) \land (x_g \lor \neg x_b) \\
\land (x_g \lor \neg x_c) \land (\neg x_g \lor x_b \lor x_c) \\
\land (\neg x_f \lor x_a) \land (\neg x_f \lor x_a) \\
\land (x_f \lor \neg x_a \lor \neg x_b) \land (\neg x_d \lor x_a)
\]

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 $\text{SAT}$ formula generate, add clauses $(x_u \lor x_v), (\neg x_u \lor \neg x_v)$. Observe that

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

- **Case \( \forall \):** 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

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

\[
(\neg x_v \lor x_u \lor x_w)
\]
Reduction: $CSAT \leq_p SAT$

- **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

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

⇒ 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)

⇐ 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