



## 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?

# CS/ECE-374: Lecture 27 - More NP-Complete reductions

---

**Lecturer:** Nickvash Kani

**Chat moderator:** Samir Khan

April 29, 2021

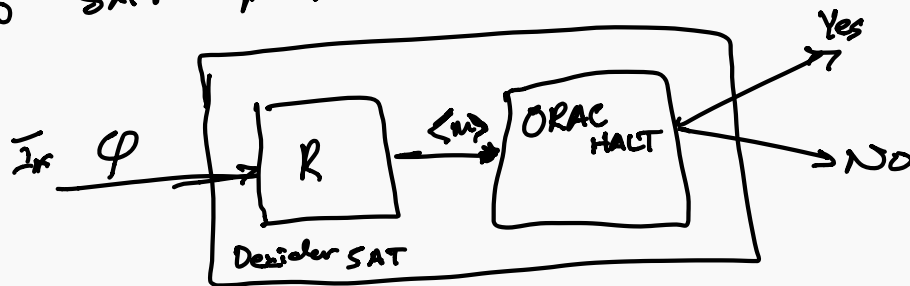
University of Illinois at Urbana-Champaign

# 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  $x \in NP$   $x \leq_p HALT$   
therefor  $SAT \leq_p HALT$

- Show  $SAT \leq_p HALT$



Reduction: Construct a TM  $\langle M \rangle$ :

- loop over all assignments to SAT
- halt if assignment is satisfied

*2 assignments*

# 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  $HC$  in directed graphs  $\longrightarrow$  NP-complete  
 $3SAT \leq HC_{directed}$

**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 □

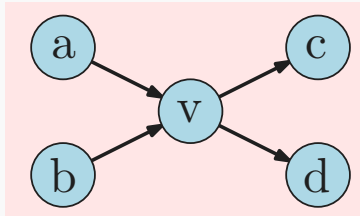


# Reduction Sketch

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

## Reduction

- 
- 

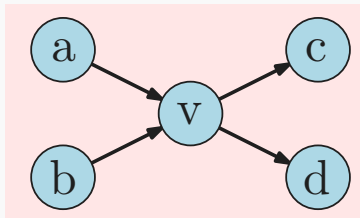


# Reduction Sketch

**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  $v_{out}$
- 

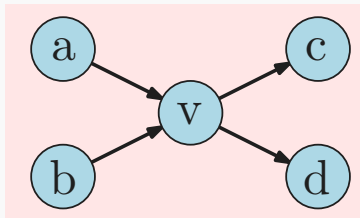


# Reduction Sketch

**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  $(a_{out}, b_{in})$

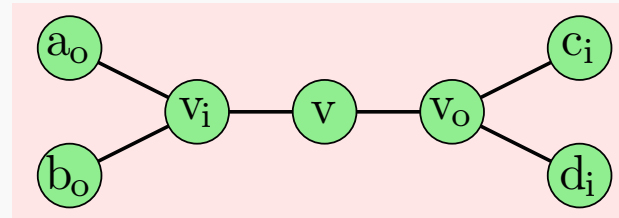
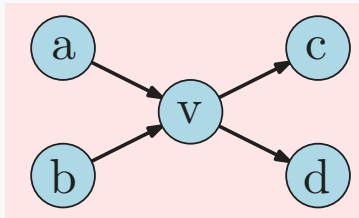


# Reduction Sketch

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

## Reduction

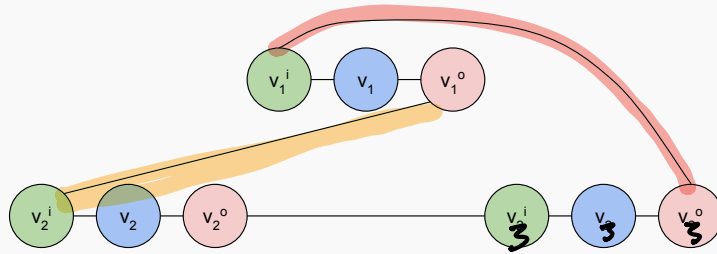
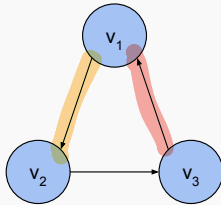
- Replace each vertex  $v$  by 3 vertices:  $v_{in}$ ,  $v$ , and  $v_{out}$
- A directed edge  $(a, b)$  is replaced by edge  $(a_{out}, b_{in})$



*if there is a HC in the directed graph then there is one in the undirected graph*

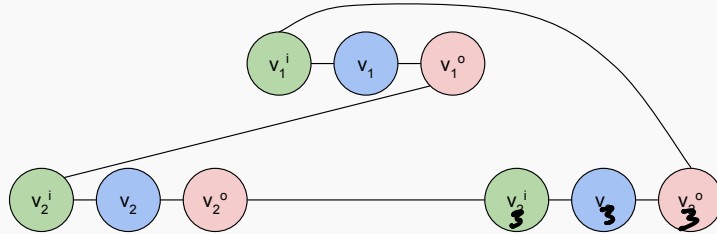
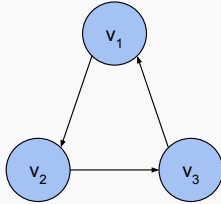
# Reduction Sketch Example

Graph *with* cycle:

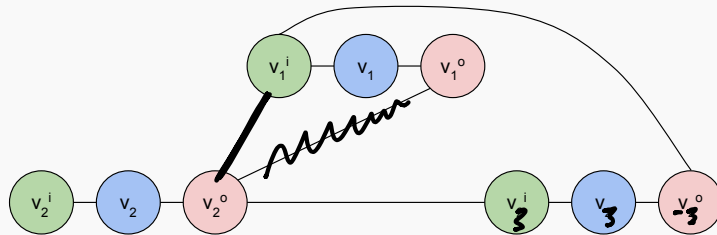
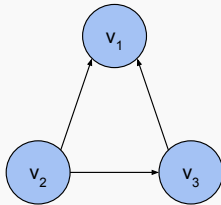


# Reduction Sketch Example

Graph *with* cycle:



Graph *without* cycle:



# 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?

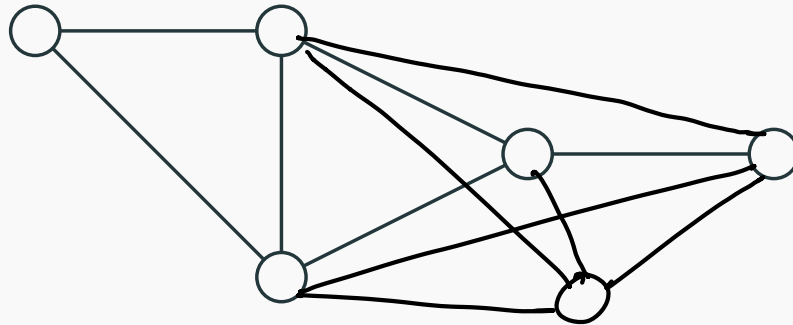


# Graph 3-Coloring

## 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?

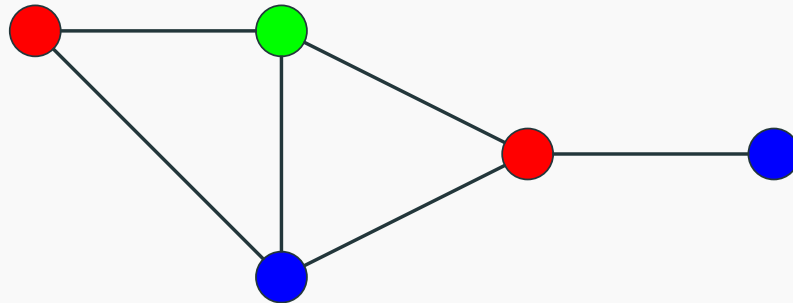


# Graph 3-Coloring

## 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?



# 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

---

# Graph Coloring and Register Allocation

## 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  $k \geq 3$   
*k - color*

# 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  $[a_0, b_0], [a_1, b_1], \dots, [a_k, b_k]$
- Each cell phone tower (simplifying) gets one band
- Constraint: nearby towers cannot be assigned same band, otherwise signals will interference

# 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  $[a_0, b_0], [a_1, b_1], \dots, [a_k, b_k]$
- Each cell phone tower (simplifying) gets one band
- Constraint: nearby towers cannot be assigned same band, otherwise signals will interference

**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. *Building a poly certifier  $\mathcal{O}(E)$*

*certificate* →

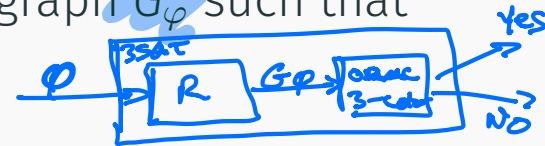
Non-deterministically guess a 3-coloring for each node

- Check if for each edge  $(u, v)$ , the color of  $u$  is different from that of  $v$ .

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

# Reduction Idea

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



- need to establish truth assignment for  $x_1, \dots, x_n$  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

# 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:

$$\varphi = f(x_1, x_2) = (x_1 \vee x_2) \quad \{x_1, x_2\} = \begin{cases} [1, 1] \\ [1, 0] \\ [0, 1] \end{cases} \quad (1)$$

# 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

)  $G \neq \emptyset$

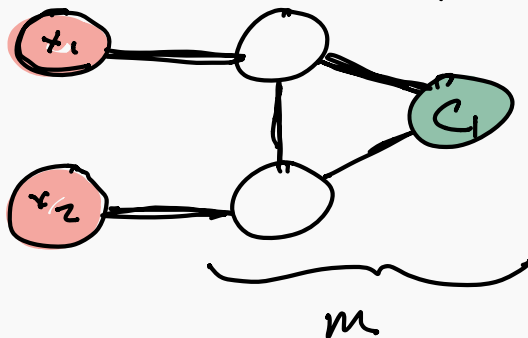
Let's start off with the simplest SAT we can think of:

$$f(x_1, x_2) = (x_1 \vee x_2)$$



Assume green=true and red=false,

$G$



OR Gate

# 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

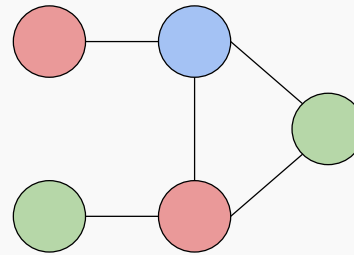
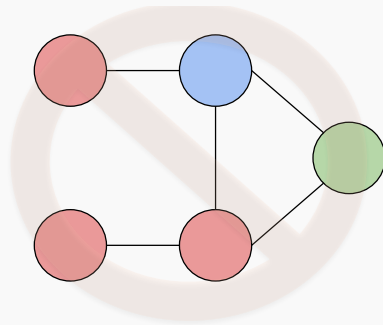
**Let's try some stuff:**

# Reduction Idea I - Simple 3-color gadget

We want to create a gadget that:

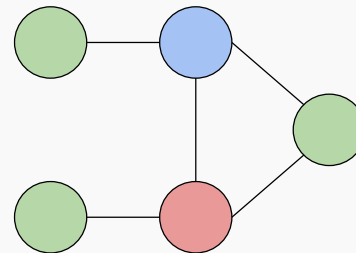
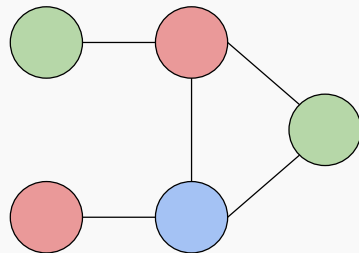
- Is 3 colorable if at least one of the literals is true
- Not 3-colorable if none of the literals are true

Seems to work:



*Prereqs:*

- Clause node green
- literals red/green





# 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

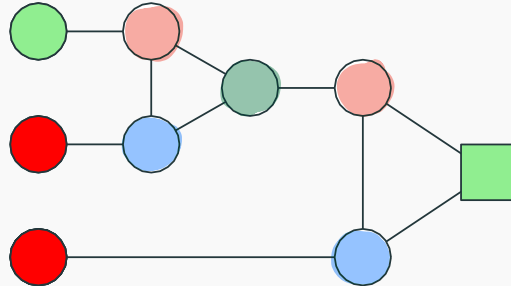
How do we do the same thing for 3 variables?:

$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \quad (2)$$

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).

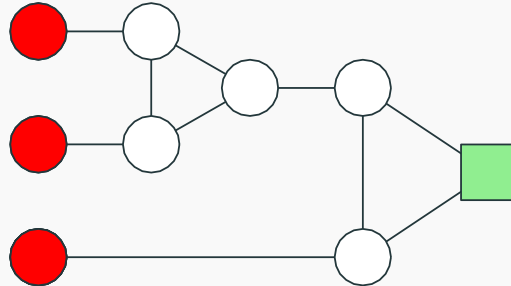


a Yes.

b No.

## 3 color this gadget.

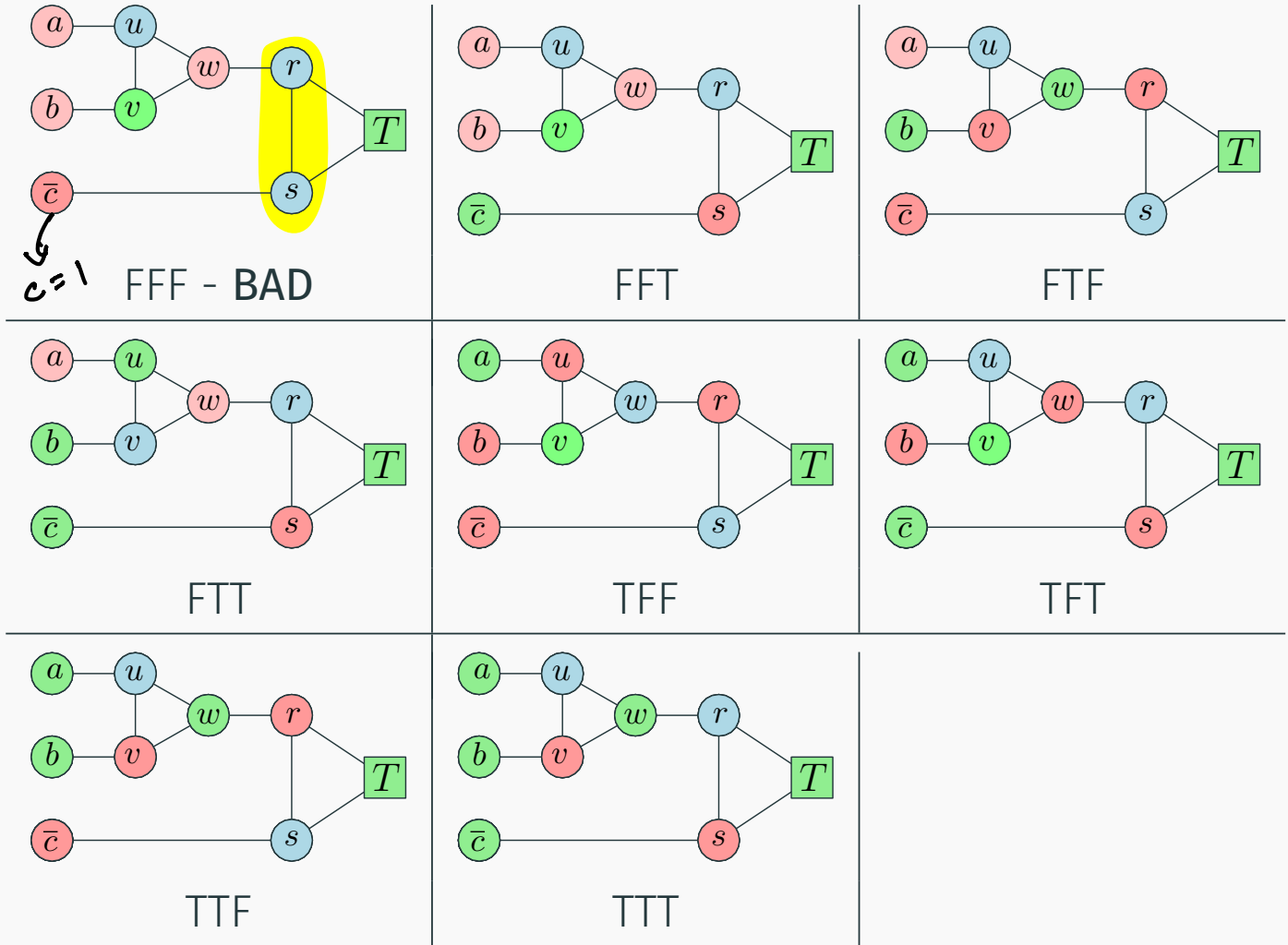
You are given three colors: red, green and blue. Can the following graph be three colored in a valid way (assuming that some of the nodes are already colored as indicated).



a Yes.

b No.

# 3-coloring of the clause gadget

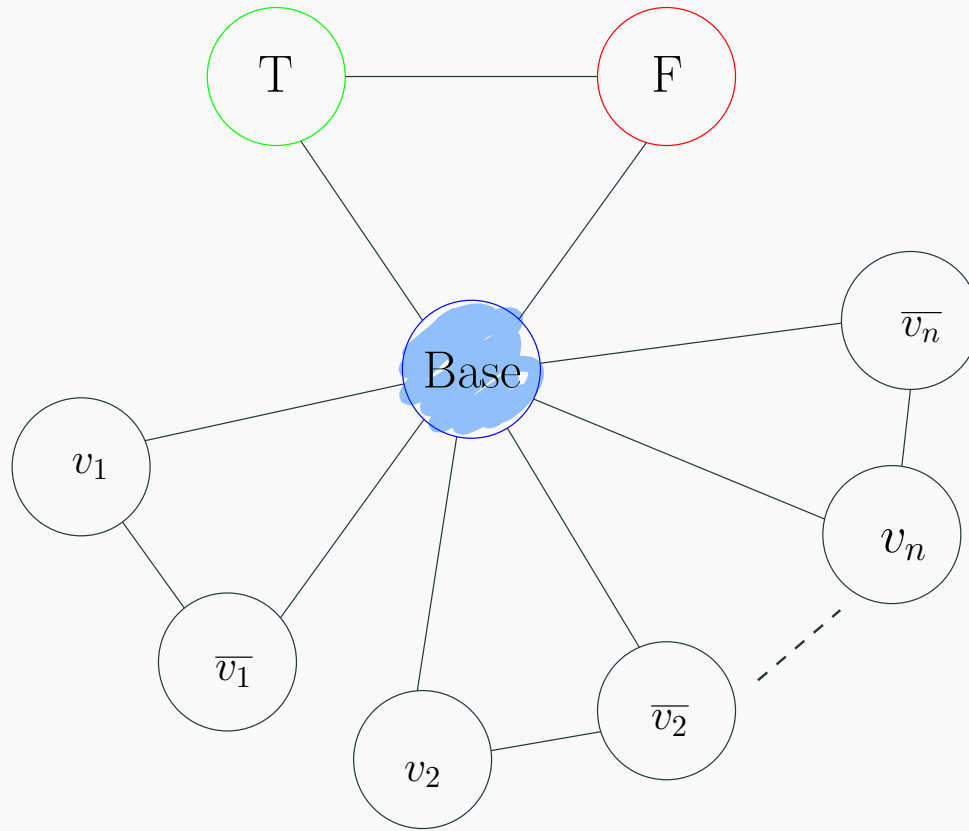


# Reduction Idea II - Literal Assignment I

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  $\bar{x}_1$  is green. Other must be red

# Reduction Idea II - Literal Assignment II

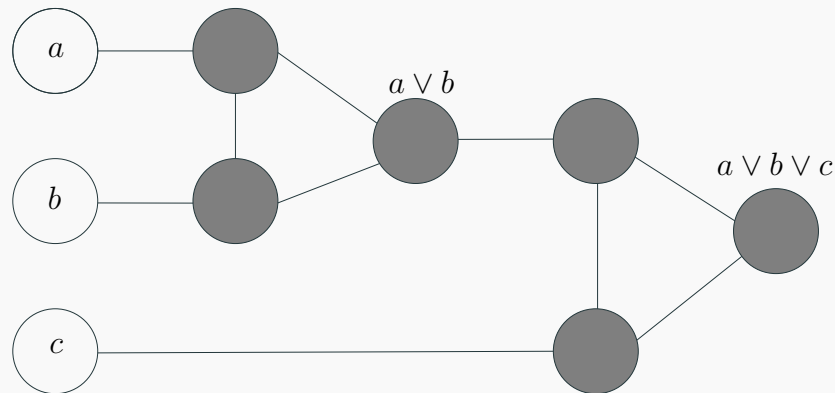


# Review Clause Satisfiability Gadget

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:



# OR-Gadget Graph

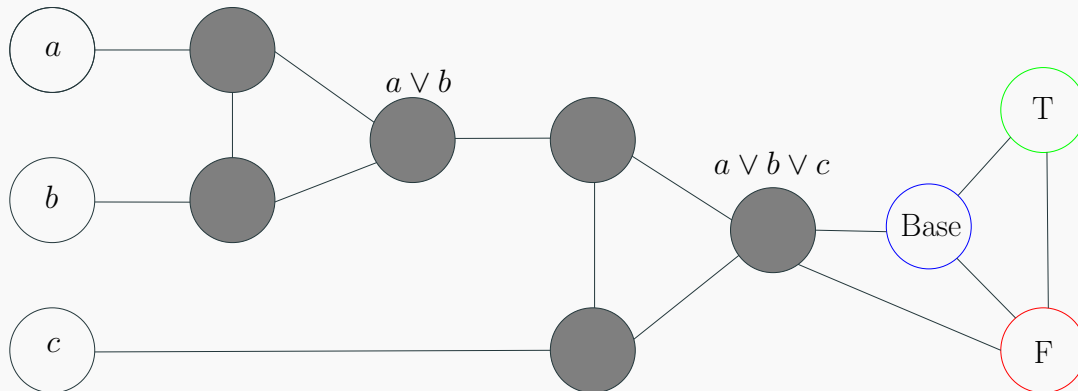
**Property:** if  $a, b, c$  are colored False in a 3-coloring then output node of OR-gadget has to be colored False.

**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.

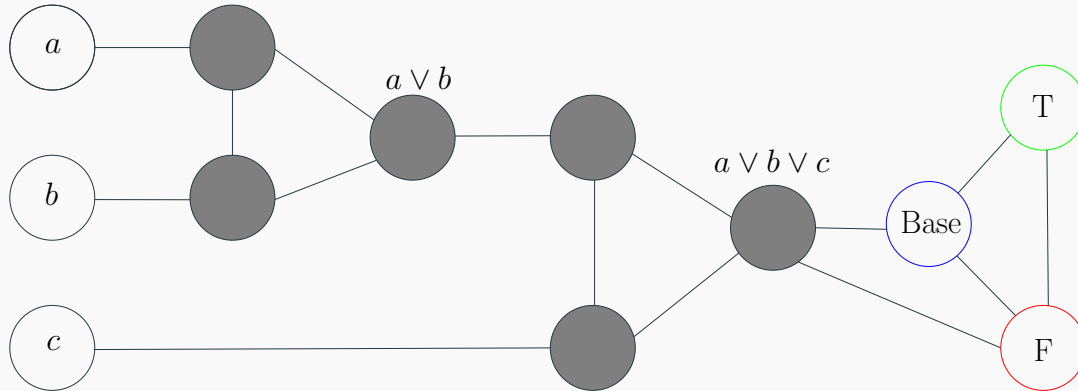


# Reduction

- 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  $C_j = (a \vee b \vee c)$ , add OR-gadget graph with input nodes  $a, b, c$  and connect output node of gadget to both False and Base



# Reduction



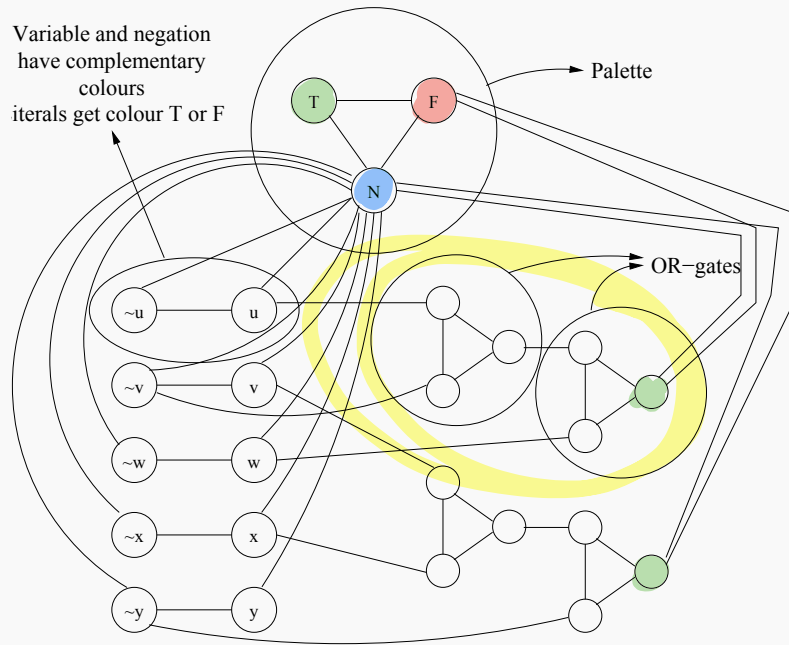
## Lemma

*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.*

# Reduction Outline

## Example

$$\varphi = (u \vee \neg v \vee w) \wedge (v \vee x \vee \neg y)$$



# 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

# 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  $C_j = (a \vee b \vee c)$  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  $\bar{v}_i$  False
- for each clause  $C_j = (a \vee b \vee c)$  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  $\bar{v}_i$  False
- for each clause  $C_j = (a \vee b \vee c)$  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

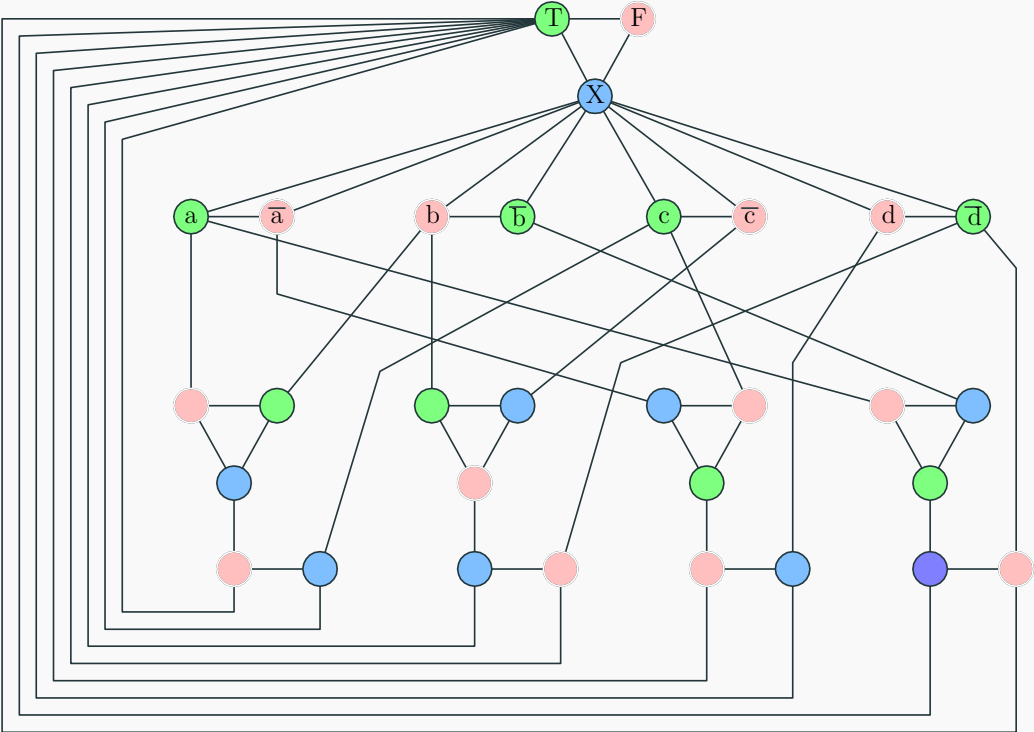
- if  $x_i$  is assigned True, color  $v_i$  True and  $\bar{v}_i$  False
- for each clause  $C_j = (a \vee b \vee c)$  at least one of  $a, b, c$  is colored True. OR-gadget for  $C_j$  can be 3-colored such that output is True.

$G_\varphi$  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  $C_j = (a \vee b \vee c)$ . 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

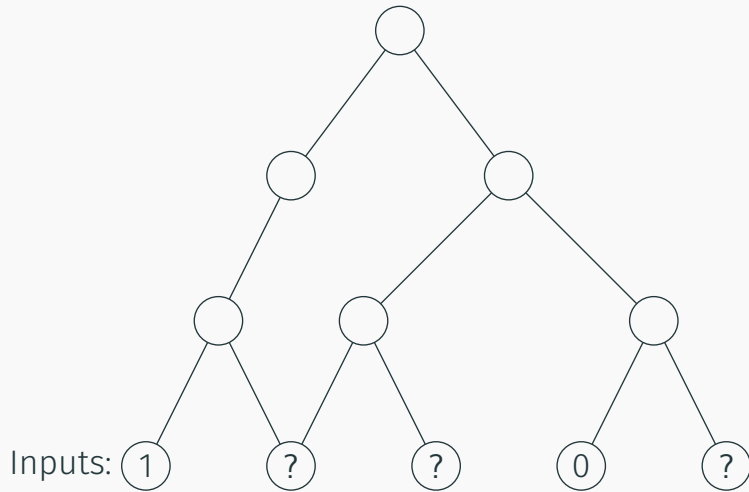


# Circuit-Sat Problem

---

# Circuits

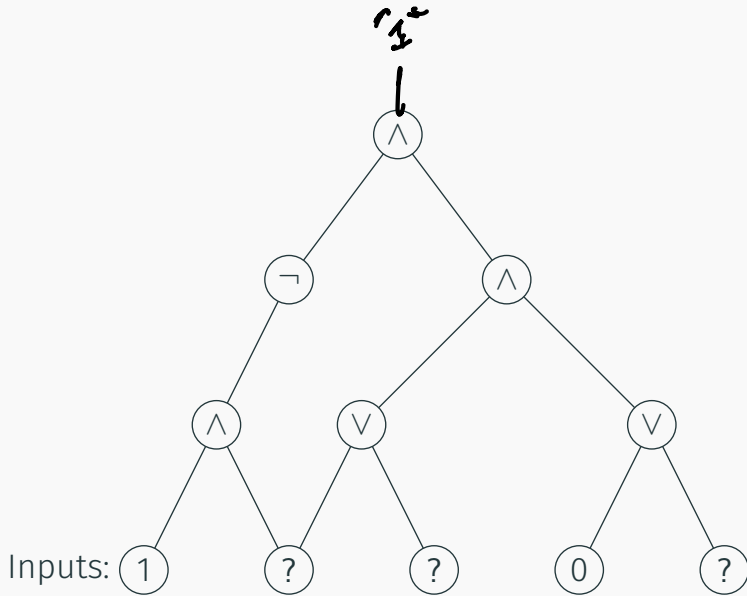
A circuit is a directed *acyclic* graph with



- **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.

# Circuits

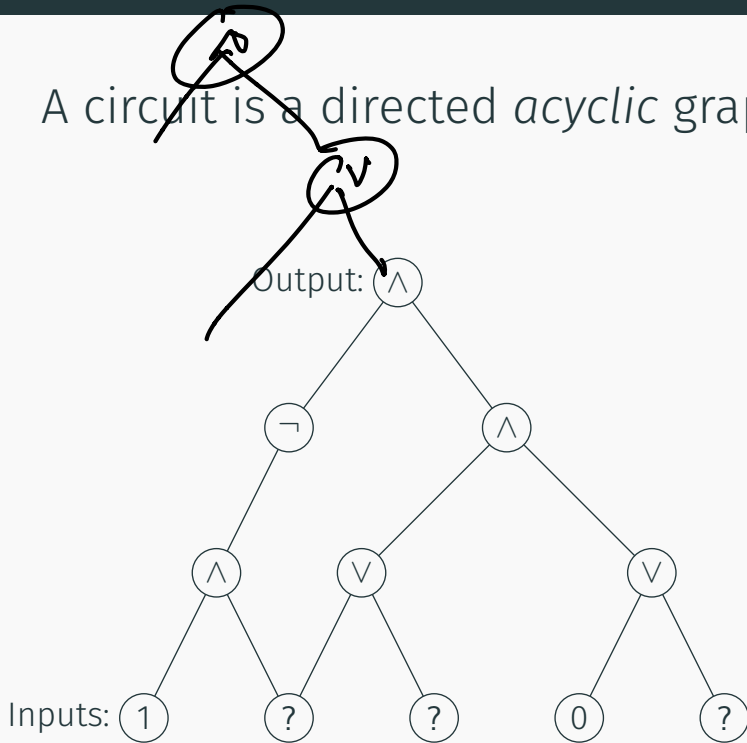
A circuit is a directed *acyclic* graph with



- **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.

# Circuits

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.

# 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?

# 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.

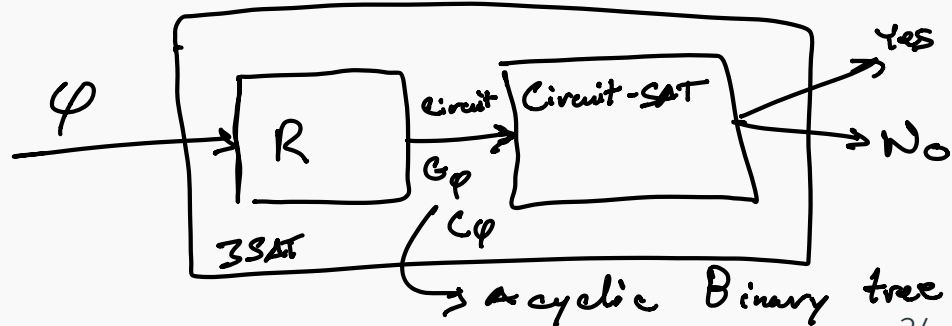
Circuits are a much more powerful (and hence easier) way to express Boolean formulas

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

Theorem  
 $SAT \leq_P 3SAT \leq_P CSAT.$

Theorem  
 $CSAT \leq_P SAT \leq_P 3SAT.$

$$EXP \leq 3SAT \leq_P \text{Circuit-SAT}$$



# Converting a CNF formula into a Circuit

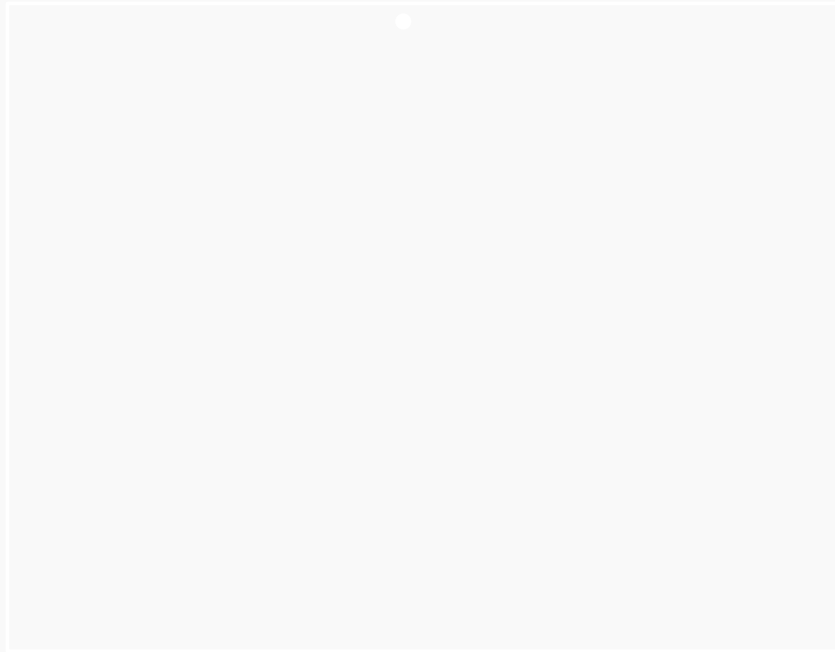
Given 3CNF formula  $\varphi$  with  $n$  variables and  $m$  clauses, create a Circuit  $C$ .

- Inputs to  $C$  are the  $n$  boolean variables  $x_1, x_2, \dots, x_n$
- Use NOT gate to generate literal  $\neg x_i$  for each variable  $x_i$
- For each clause  $(l_1 \vee l_2 \vee l_3)$  use two OR gates to mimic formula
- Combine the outputs for the clauses using AND gates to obtain the final output

$$\text{SAT} = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$$

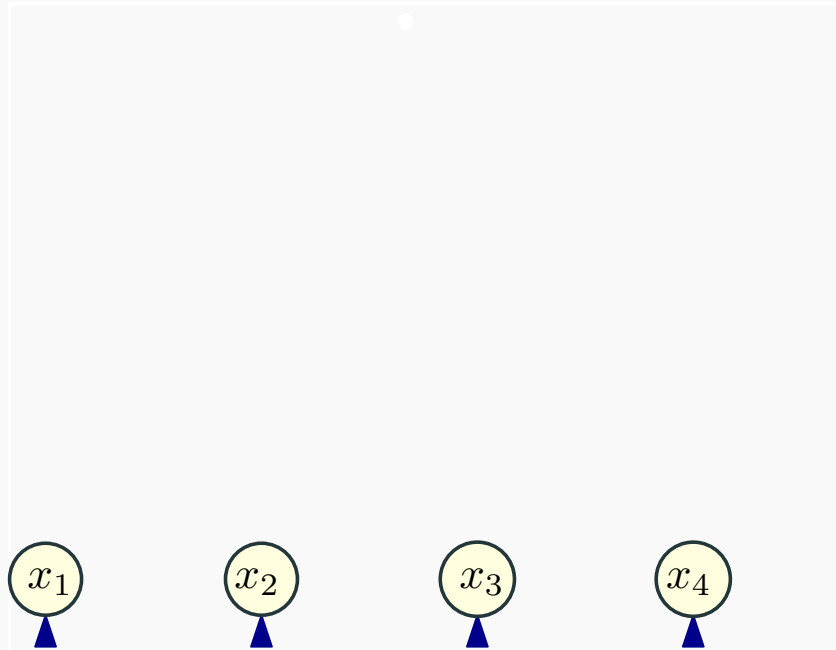
## Example: $3SAT \leq_p CSAT$

$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



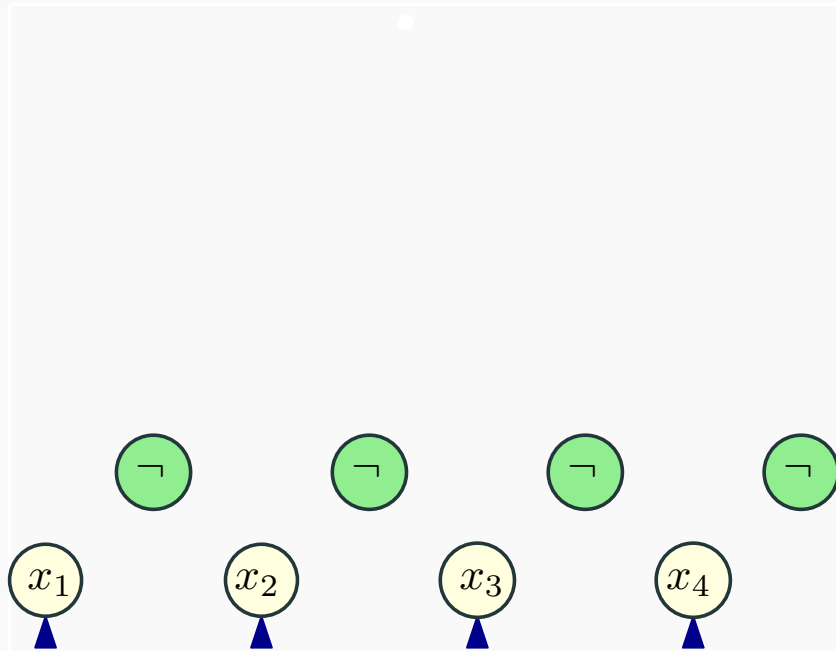
# Example: $3SAT \leq_p CSAT$

$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



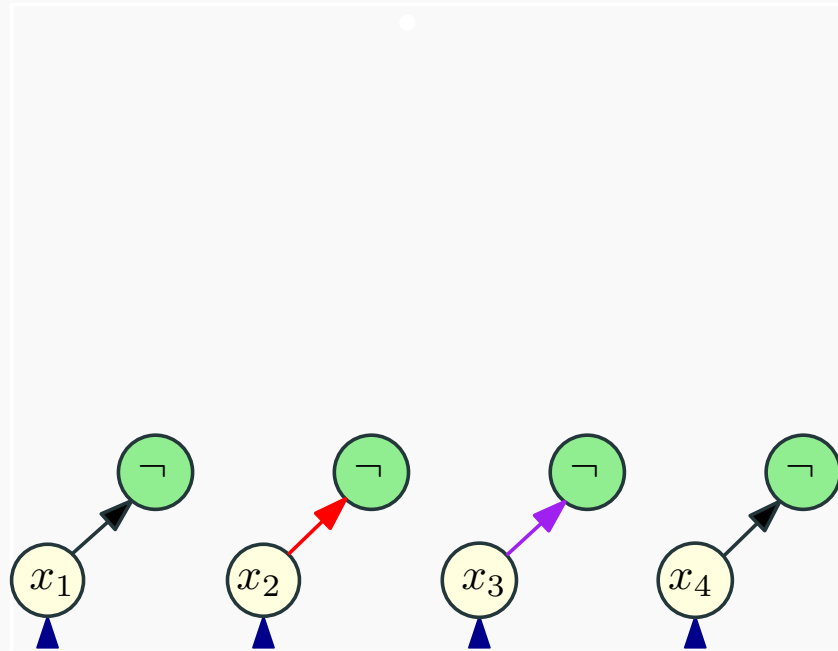
# Example: $3SAT \leq_p CSAT$

$$\varphi = (x_1 \vee \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



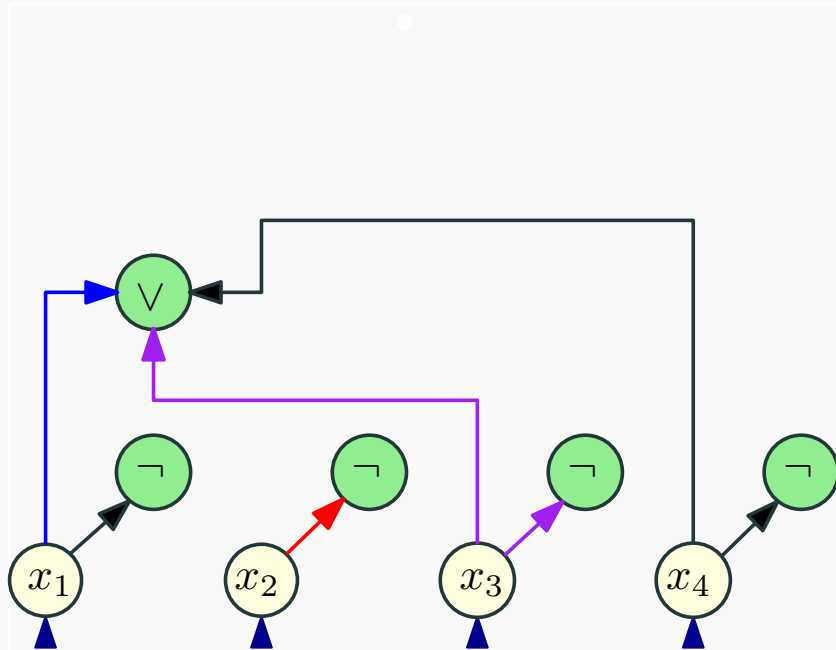
# Example: $3SAT \leq_p CSAT$

$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



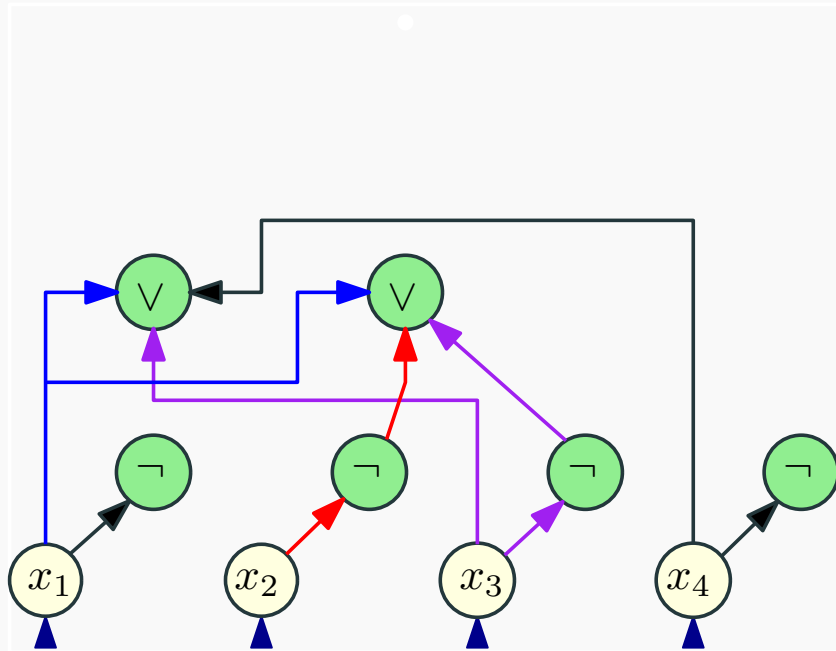
# Example: $3SAT \leq_p CSAT$

$$\varphi = (x_1 \vee \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



# Example: $3SAT \leq_p CSAT$

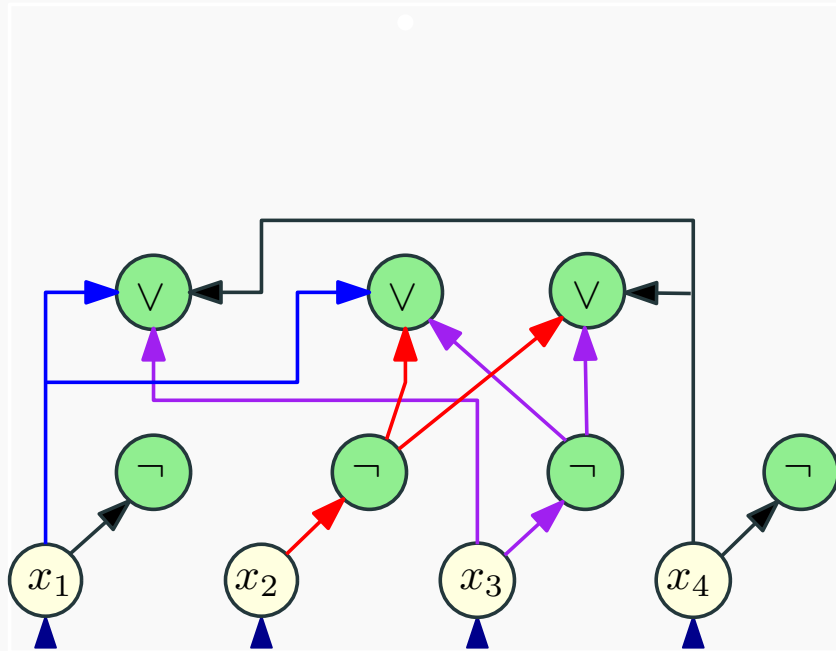
$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$





# Example: $3SAT \leq_p CSAT$

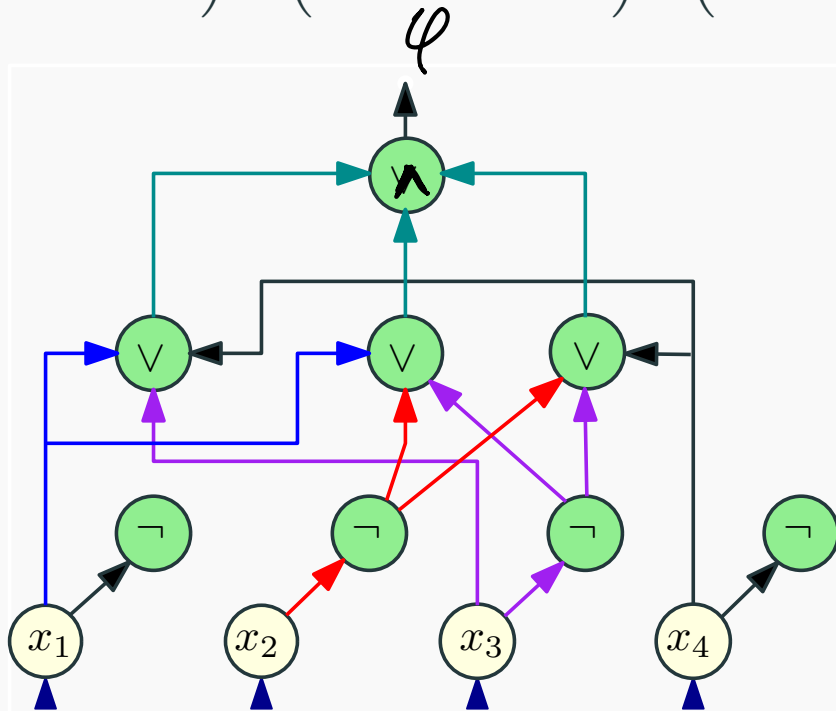
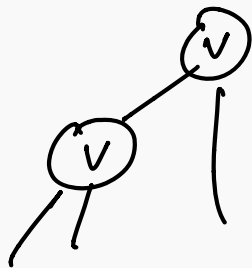
$$\varphi = (x_1 \vee \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



Example: 3SAT  $\leq_P$  CSAT

$\rightarrow$  NP-complete

$$\varphi = (x_1 \vee \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



# Converting a circuit to a SAT formula

What will converting a circuit to a SAT formula prove?

$$\text{Circuit-SAT} \leq_p \text{SAT}$$

$$\overset{\text{ENP}}{X} \leq 3 \text{SAT} \leq_p \text{Circuit-SAT} \quad \text{show we C SAT in NP-hard}$$

# 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 $z = x \wedge y$ to 3SAT

z	x	y					
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

$$z = x \vee y$$

$$[x, y, z] = [1, 0, 1]$$
$$[0, 0, 0]$$

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$				
0	0	0	1				
0	0	1	1				
0	1	0	1				
0	1	1	0				
1	0	0	0				
1	0	1	0				
1	1	0	0				
1	1	1	1				

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$				
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

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee e\bar{y}$			
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



# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee \bar{y}$	$\bar{z} \vee x \vee y$		
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

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee \bar{y}$	$\bar{z} \vee x \vee y$	$\bar{z} \vee x \vee \bar{y}$	
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

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee e\bar{e}\bar{y}$	$\bar{z} \vee x \vee y$	$\bar{z} \vee x \vee \bar{y}$	$\bar{z} \vee \bar{x} \vee y$
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

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee e\bar{e}\bar{y}$	$\bar{z} \vee x \vee y$	$\bar{z} \vee x \vee \bar{y}$	$\bar{z} \vee \bar{x} \vee y$
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

# Converting $z = x \wedge y$ to 3SAT

$z$	$x$	$y$	$z = x \wedge y$	$z \vee \bar{x} \vee \bar{y}$	$\bar{z} \vee x \vee y$	$\bar{z} \vee x \vee \bar{y}$	$\bar{z} \vee \bar{x} \vee y$
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

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

$\equiv$

$$(z \vee \bar{x} \vee \bar{y}) \wedge (\bar{z} \vee x \vee y) \wedge (\bar{z} \vee x \vee \bar{y}) \wedge (\bar{z} \vee \bar{x} \vee y)$$

# Summary of formulas we derived

## Lemma

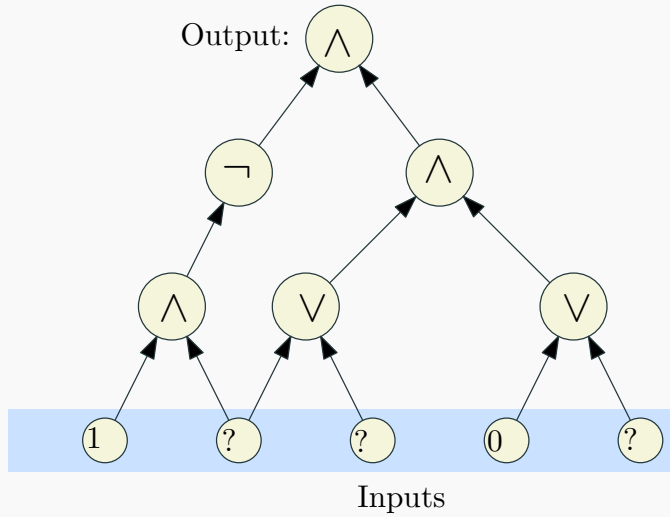
*The following identities hold:*

$$\bullet z = \bar{x} \quad \equiv \quad (z \vee x) \wedge (\bar{z} \vee \bar{x}).$$

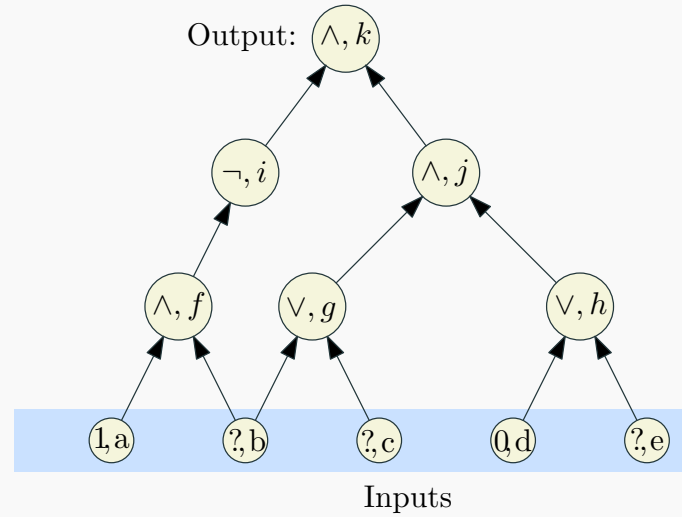
$$\bullet (z = x \vee y) \quad \equiv \quad (z \vee \bar{y}) \wedge (z \vee \bar{x}) \wedge (\bar{z} \vee x \vee y)$$

$$\bullet (z = x \wedge y) \quad \equiv \quad (z \vee \bar{x} \vee \bar{y}) \wedge (\bar{z} \vee x) \wedge (\bar{z} \vee y)$$

# Converting a circuit into a CNF formula

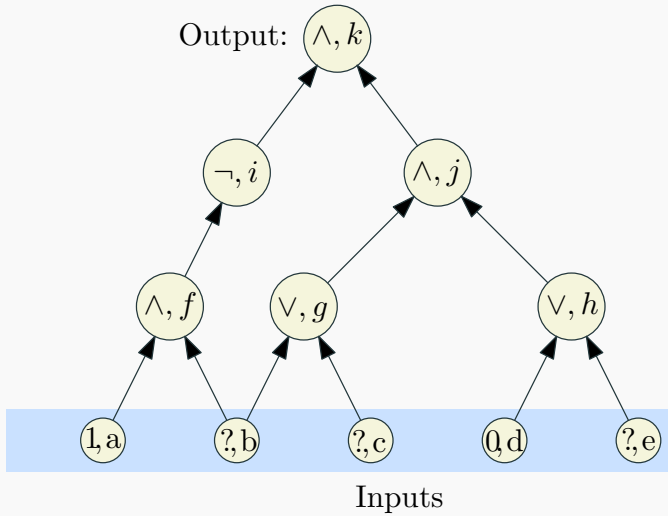


(A) Input circuit

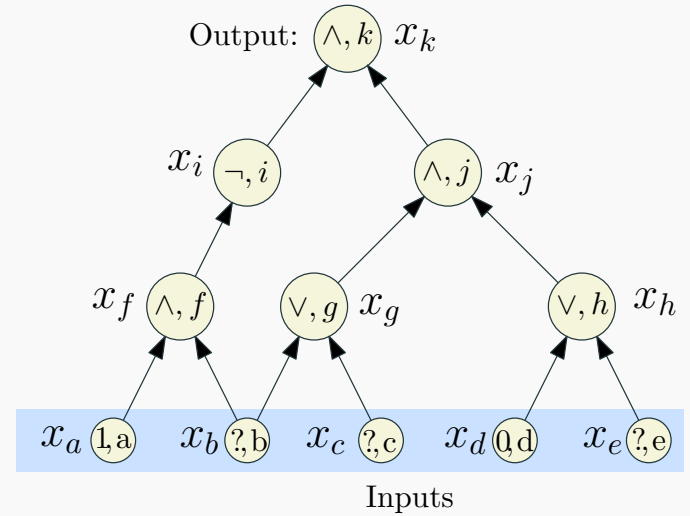


(B) Label the nodes.

# Converting a circuit into a CNF formula



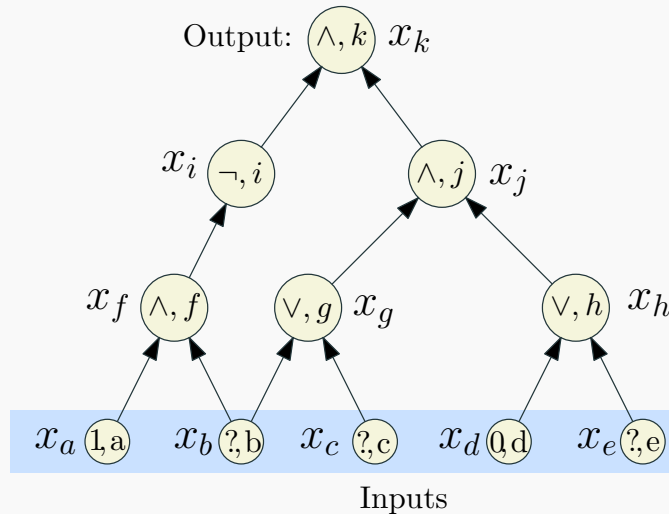
(B) Label the nodes.



(C) Introduce var for each node.



# Converting a circuit into a CNF formula



(C) Introduce var for each node.

$x_k$  (Demand a sat' assignment!)

$$x_k = x_i \wedge x_j$$

$$x_j = x_g \wedge x_h$$

$$x_i = \neg x_f$$

$$x_h = x_d \vee x_e$$

$$x_g = x_b \vee x_c$$

$$x_f = x_a \wedge x_b$$

$$x_d = 0$$

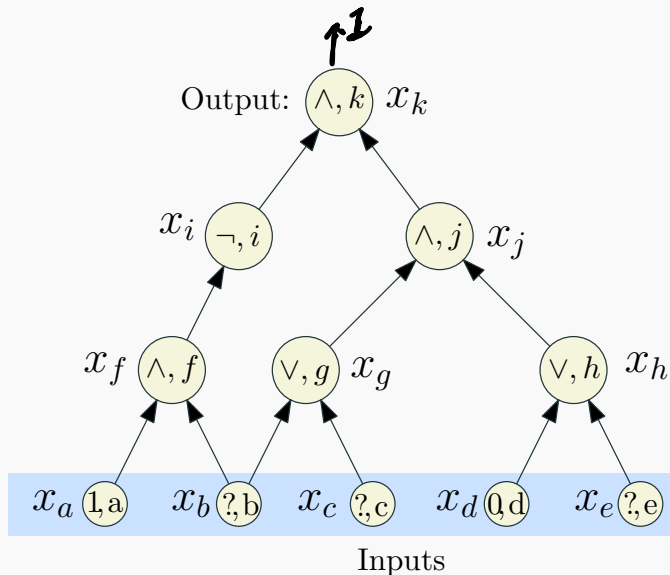
$$x_a = 1$$

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

# Converting a circuit into a CNF formula

$X_k$	$X_k$
$X_k = X_i \wedge X_j$	$(\neg X_k \vee X_i) \wedge (\neg X_k \vee X_j) \wedge (X_k \vee \neg X_i \vee \neg X_j)$
$X_j = X_g \wedge X_h$	$(\neg X_j \vee X_g) \wedge (\neg X_j \vee X_h) \wedge (X_j \vee \neg X_g \vee \neg X_h)$
$X_i = \neg X_f$	$(X_i \vee X_f) \wedge (\neg X_i \vee \neg X_f)$
$X_h = X_d \vee X_e$	$(X_h \vee \neg X_d) \wedge (X_h \vee \neg X_e) \wedge (\neg X_h \vee X_d \vee X_e)$
$X_g = X_b \vee X_c$	$(X_g \vee \neg X_b) \wedge (X_g \vee \neg X_c) \wedge (\neg X_g \vee X_b \vee X_c)$
$X_f = X_a \wedge X_b$	$(\neg X_f \vee X_a) \wedge (\neg X_f \vee X_b) \wedge (X_f \vee \neg X_a \vee \neg X_b)$
$X_d = 0$	$\neg X_d$
$X_a = 1$	$X_a$

# Converting a circuit into a CNF formula



$$\begin{aligned}
 & x_k \wedge (\neg x_k \vee x_i) \wedge (\neg x_k \vee x_j) \\
 & \wedge (x_k \vee \neg x_i \vee \neg x_j) \wedge (\neg x_j \vee x_g) \\
 & \wedge (\neg x_j \vee x_h) \wedge (x_j \vee \neg x_g \vee \neg x_h) \\
 & \wedge (x_i \vee x_f) \wedge (\neg x_i \vee \neg x_f) \\
 & \wedge (x_h \vee \neg x_d) \wedge (x_h \vee \neg x_e) \\
 & \wedge (\neg x_h \vee x_d \vee x_e) \wedge (x_g \vee \neg x_b) \\
 & \wedge (x_g \vee \neg x_c) \wedge (\neg x_g \vee x_b \vee x_c) \\
 & \wedge (\neg x_f \vee x_a) \wedge (\neg x_f \vee x_b) \\
 & \wedge (x_f \vee \neg x_a \vee \neg x_b) \wedge (\neg x_d) \wedge x_a
 \end{aligned}$$

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  $(x_u \vee x_v)$ ,  $(\neg x_u \vee \neg x_v)$ . Observe that

$$x_v = \neg x_u \text{ is true} \iff \begin{array}{l} (x_u \vee x_v) \\ (\neg x_u \vee \neg x_v) \end{array} \text{ both true.}$$

# Reduction: $CSAT \leq_p SAT$

- **Case V:** So  $x_V = x_U \vee x_W$ . In **SAT** formula generated, add clauses  $(x_V \vee \neg x_U)$ ,  $(x_V \vee \neg x_W)$ , and  $(\neg x_V \vee x_U \vee x_W)$ . Again, observe that

$$\left(x_V = x_U \vee x_W\right) \text{ is true} \iff \begin{array}{l} (x_V \vee \neg x_U), \\ (x_V \vee \neg x_W), \\ (\neg x_V \vee x_U \vee x_W) \end{array} \text{ all true.}$$

# Reduction: $CSAT \leq_P SAT$

- **Case  $\wedge$ :** So  $x_V = x_U \wedge x_W$ . In **SAT** formula generated, add clauses  $(\neg x_V \vee x_U)$ ,  $(\neg x_V \vee x_W)$ , and  $(x_V \vee \neg x_U \vee \neg x_W)$ . Again observe that

$$x_V = x_U \wedge x_W \text{ is true} \iff \begin{array}{l} (\neg x_V \vee x_U), \\ (\neg x_V \vee x_W), \\ (x_V \vee \neg x_U \vee \neg x_W) \end{array} \text{ all true.}$$

## 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  $C$