CMSC 28100

Introduction to Complexity Theory

Spring 2024

Instructor: William Hoza

k-CNF formulas

- A k-CNF formula is an AND of ORs of literals in which every clause has at most k literals
- Example of a 3-CNF formula with two clauses:

$$\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_6) \wedge (x_5 \vee x_1 \vee x_2)$$

The Cook-Levin Theorem

- Define $k\text{-SAT} = \{\langle\phi\rangle : \phi \text{ is a satisfiable } k\text{-CNF formula}\}$

The Cook-Levin Theorem: $3\text{-}SAT$ is NP-complete

- **Proof:** $3\text{-SAT} \in \text{NP}$ (guess a satisfying assignment)
- To show that 3-SAT is NP-hard, we will reduce from CIRCUIT-SAT

Gate gadgets

- Define the following Boolean functions:

$$\text{CHECK-NOT}(g, y) = \begin{cases} 1 & \text{if } g = \bar{y} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{CHECK-AND}(g, y, z) = \begin{cases} 1 & \text{if } g = (y \wedge z) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{CHECK-OR}(g, y, z) = \begin{cases} 1 & \text{if } g = (y \vee z) \\ 0 & \text{otherwise} \end{cases}$$

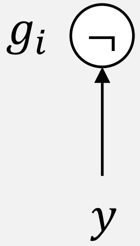
- Each can be represented by a 3-CNF formula. (Every function has a CNF representation!)

Reduction from CIRCUIT-SAT to 3-SAT

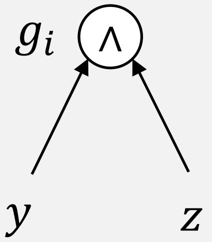
- Reduction: $f(\langle C \rangle) = \langle \phi \rangle$, where ϕ is a 3-CNF defined as follows
- Circuit C has variables $x_1, x_2, \ldots, x_n$ and AND/OR/NOT gates $g_1, \ldots, g_m$
- Assume without loss of generality that g_m is the output gate
- Formula ϕ has $n + m$ variables, which we denote $x_1, \ldots, x_n, g_1, \ldots, g_m$
- Note: In C, "g_i" is the name of a gate. In ϕ, "g_i" is the name of a variable

Reduction from CIRCUIT-SAT to 3-SAT

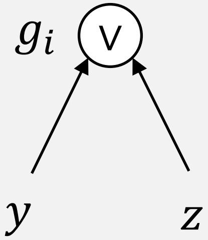
- For each AND/OR/NOT gate g_i in the circuit C, define a 3-CNF ϕ_i:



$\phi_i = \text{CHECK-NOT}(g_i, y)$



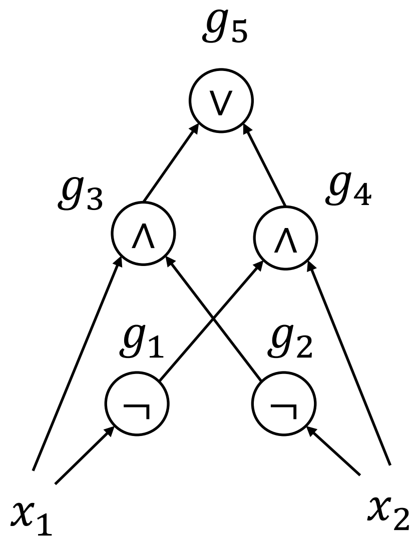
$\phi_i = \text{CHECK-AND}(g_i, y, z)$



$\phi_i = \text{CHECK-OR}(g_i, y, z)$

- Reduction produces $\phi := \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_m \wedge (g_m)$

Reduction example



- $\phi_1 = \text{CHECK-NOT}(g_1, x_1) = (g_1 \vee x_1) \wedge (\bar{g}_1 \vee \bar{x}_1)$
- $\phi_2 = \text{CHECK-NOT}(g_2, x_2) = (g_2 \vee x_2) \wedge (\bar{g}_2 \vee \bar{x}_2)$
- $\phi_3 = \text{CHECK-AND}(g_3, x_1, g_2) = (\bar{g}_3 \vee x_1) \wedge (\bar{g}_3 \vee g_2) \wedge (g_3 \vee \bar{x}_1 \vee \bar{g}_2)$
- $\phi_4 = \text{CHECK-AND}(g_4, g_1, x_2) = (\bar{g}_4 \vee g_1) \wedge (\bar{g}_4 \vee x_2) \wedge (g_4 \vee \bar{g}_1 \vee \bar{x}_2)$
- $\phi_5 = \text{CHECK-OR}(g_5, g_3, g_4) = (g_5 \vee \bar{g}_3) \wedge (g_5 \vee \bar{g}_4) \wedge (\bar{g}_5 \vee g_3 \vee g_4)$

$$\begin{aligned}\phi = {} & (g_1 \vee x_1) \wedge (\bar{g}_1 \vee \bar{x}_1) \wedge (g_2 \vee x_2) \wedge (\bar{g}_2 \vee \bar{x}_2) \wedge (\bar{g}_3 \vee x_1) \wedge (\bar{g}_3 \vee g_2) \\ & \wedge (g_3 \vee \bar{x}_1 \vee \bar{g}_2) \wedge (\bar{g}_4 \vee g_1) \wedge (\bar{g}_4 \vee x_2) \wedge (g_4 \vee \bar{g}_1 \vee \bar{x}_2) \wedge (g_5 \vee \bar{g}_3) \\ & \wedge (g_5 \vee \bar{g}_4) \wedge (\bar{g}_5 \vee g_3 \vee g_4) \wedge (g_5)\end{aligned}$$

YES maps to YES

- **Claim:** If the circuit C is satisfiable, then the 3-CNF formula ϕ is also satisfiable
- **Proof:** We are assuming there is some $x \in \{0, 1\}^n$ such that $C(x) = 1$
- For each i, assign to g_i (the variable) the value that g_i (the gate) outputs when we evaluate C on x
- We claim that $\phi(x_1, \ldots, x_n, g_1, \ldots, g_m) = 1$. Indeed, each ϕ_i is satisfied because of the circuit structure, and $g_m = 1$ because $C(x) = 1$

NO maps to NO

- **Claim:** If C is not satisfiable, then ϕ is not satisfiable
- **Proof sketch:** We will prove the contrapositive: if ϕ is satisfiable, then C is satisfiable
- If $\phi(x_1, \ldots, x_n, g_1, \ldots, g_m) = 1$, then we claim $C(x_1, \ldots, x_n) = 1$
- Indeed, by induction on the circuit structure, g_i (the variable) must be equal to the value that g_i (the gate) outputs when we evaluate C on x. Furthermore, $g_m = 1$

Reduction efficiency

- Reduction is computable in polynomial time
- For each gate in the circuit, we write down $O(1)$ clauses, and it is straightforward to compute what they are

Let $\langle \phi \rangle = f(\langle C \rangle)$. Which of the following is <u>false</u>?

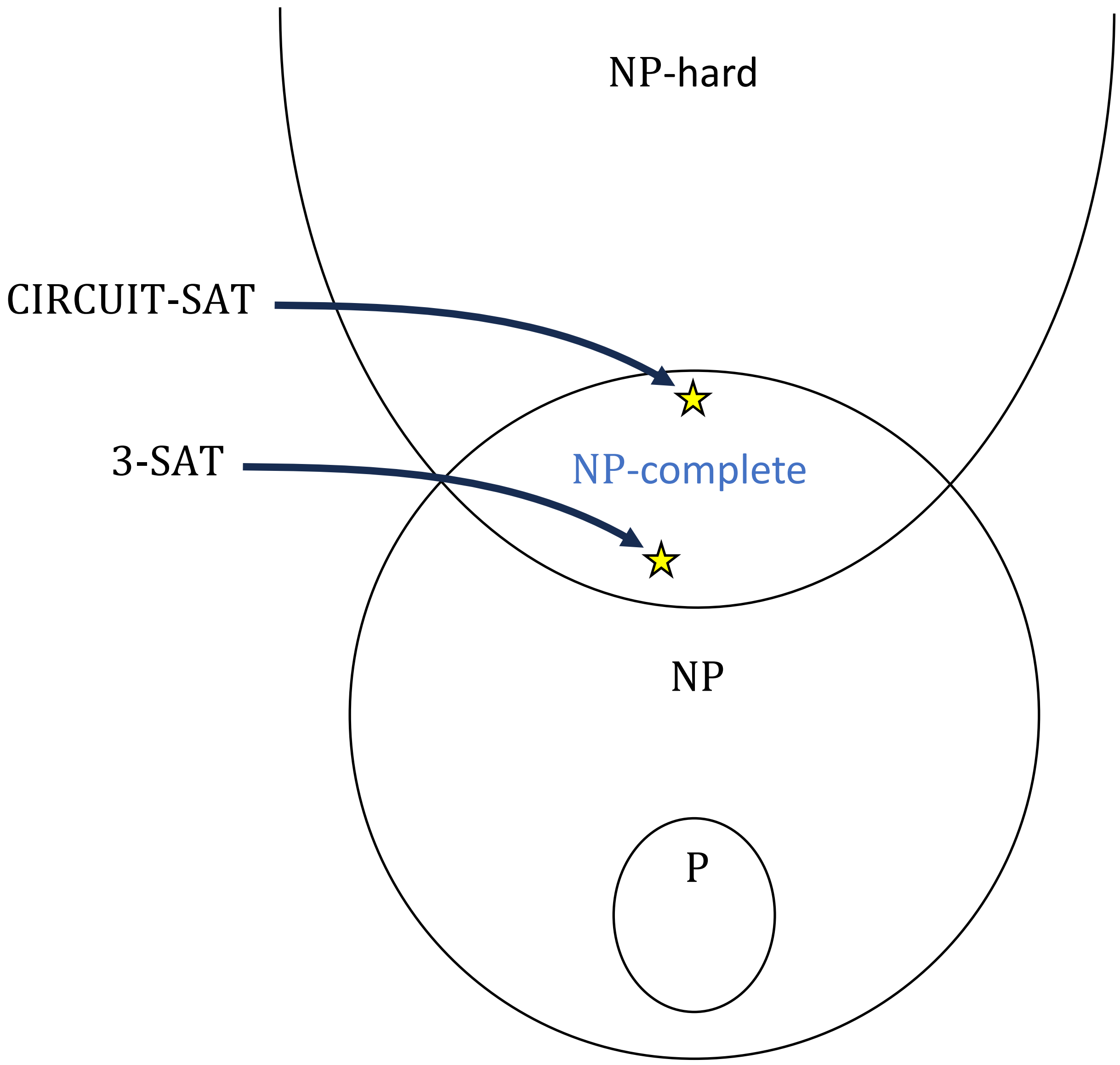
A: C is satisfiable if and only if ϕ is satisfiable

B: $|\langle \phi \rangle| \leq \mathrm{poly}(|\langle C \rangle|)$

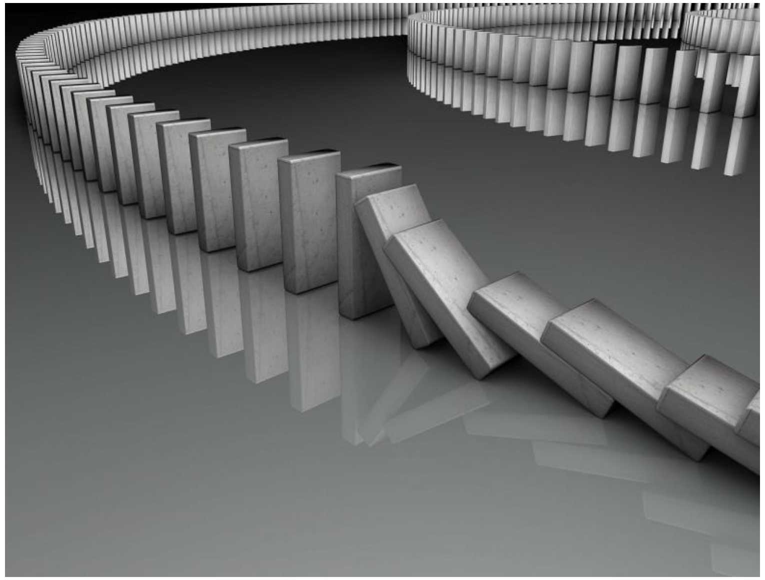
C: The number of clauses in ϕ is $\Theta(\text{size of } C)$

D: C and ϕ compute the same Boolean function

Respond at PollEv.com/whoza or text "whoza" to 22333


NP-hard
CIRCUIT-SAT
3-SAT
NP-complete
NP
P

Chaining reductions together



- 3-SAT is the starting point for many NP-hardness proofs
- We are finally ready to use the hardness of 3-SAT to prove that CLIQUE is NP-complete

CLIQUE is NP-complete

- Recall $\text{CLIQUE} = \{\langle G, k \rangle : G \text{ contains a } k\text{-clique}\}$

Theorem: CLIQUE is NP-complete

- **Proof:** We showed $\text{CLIQUE} \in \text{NP}$ in a previous class
- To prove that CLIQUE is NP-hard, we will do a reduction from 3-SAT

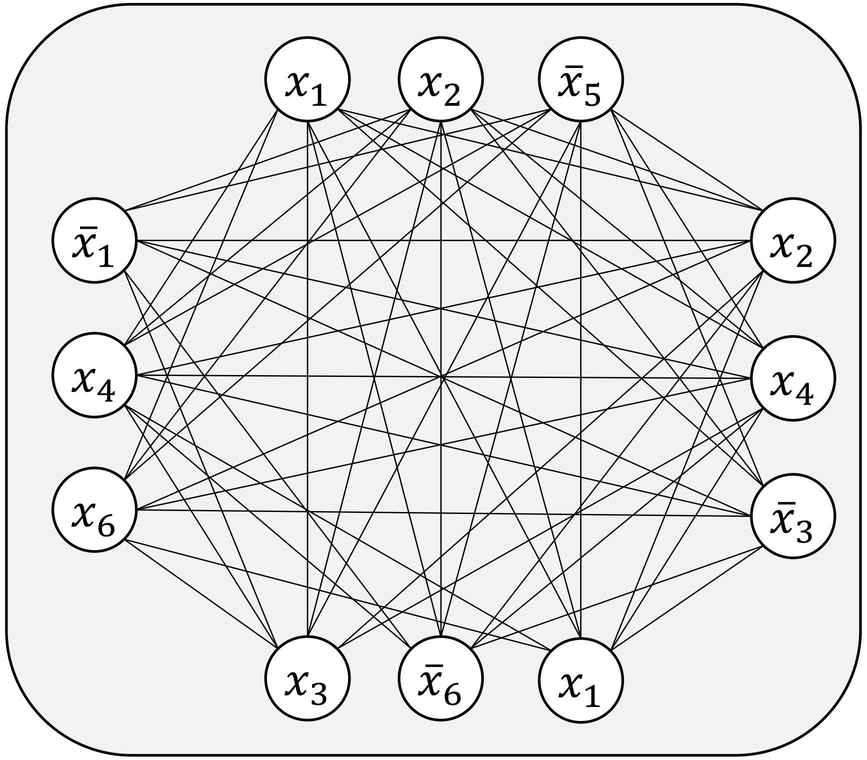
Reduction from $\mathrm{3\text{-}SAT}$ to CLIQUE

- Let ϕ be a 3-CNF formula (an instance of $\mathrm{3\text{-}SAT}$)
- Reduction: $f(\langle\phi\rangle) = \langle G, k\rangle$
  - k is the number of clauses in ϕ
  - G is a graph on $\leq 3k$ vertices defined as follows

Reduction from **3-SAT** to **CLIQUE**

- For each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$, create a "group" of three vertices labeled ℓ_1, ℓ_2, ℓ_3
  - (If the clause only has one or two literals, then only use one or two vertices)
- Put an edge $\{u, v\}$ if u and v are in different groups and u and v do not have contradictory labels (x_i and $\bar{x}_i$)

- E.g., $\phi = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_6) \wedge (x_2 \vee x_4 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_6 \vee x_1)$



YES maps to YES

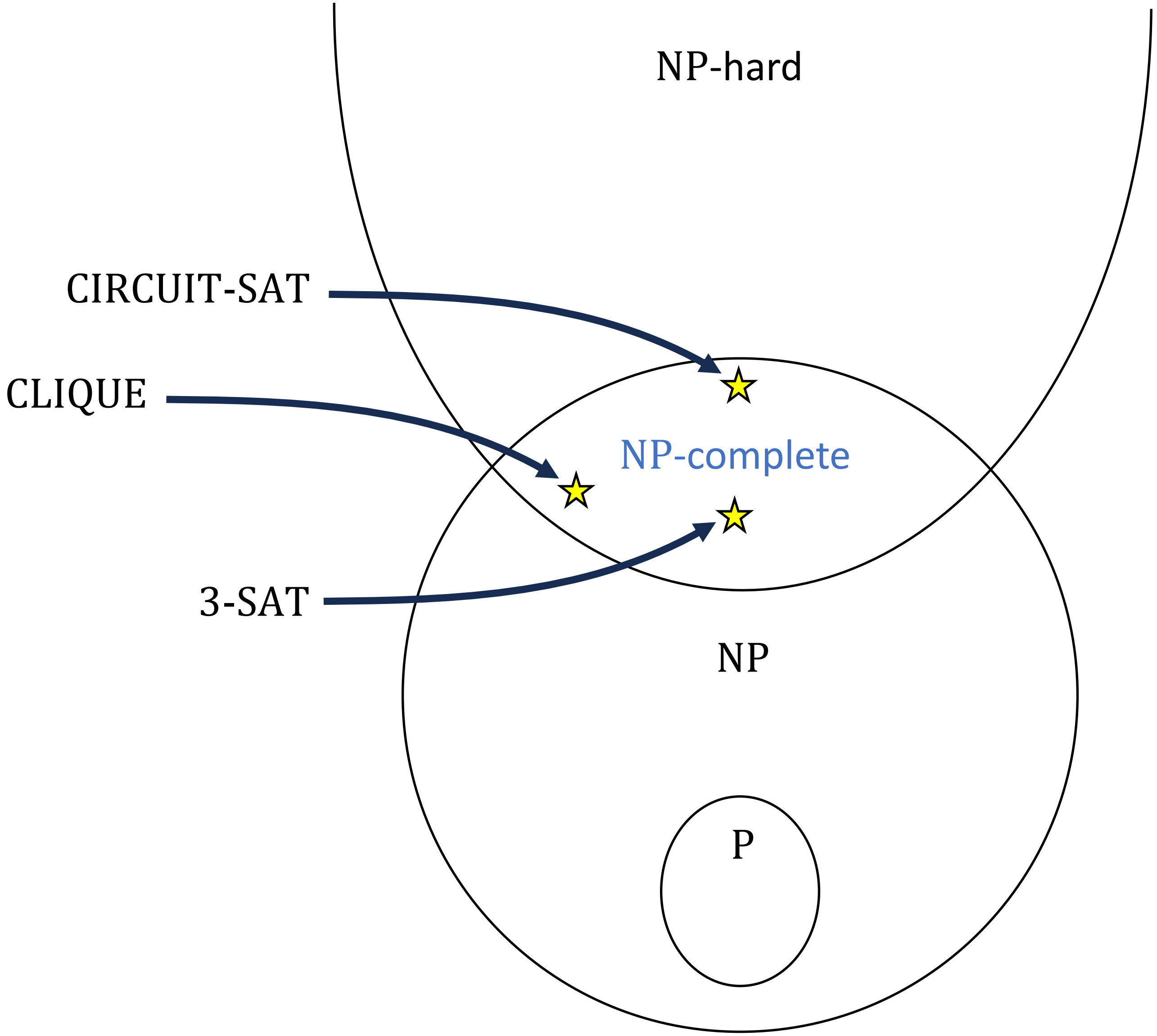
- Suppose ϕ is satisfiable, i.e., there exists a satisfying assignment x
- In each clause, at least one literal is satisfied by x
- Therefore, in each group, at least one vertex is "satisfied by x," i.e., it is labeled by a literal that is satisfied by x
- Let S be a set consisting of one satisfied vertex from each group
- Then S is a k-clique (vertices in S cannot have contradictory labels)

NO maps to NO

- Suppose G has a k-clique S
- Let x be an assignment that satisfies each vertex in S (this exists because no two vertices in S have contradictory labels)
- S cannot contain two vertices from a single group, and $|S| = k$, so S must contain one vertex from each group
- Therefore, x satisfies at least one literal in each clause, i.e., x satisfies ϕ

Poly-time computable

- Hopefully it is clear that the reduction $f(\langle \phi \rangle) = \langle G, k \rangle$ can be computed in polynomial time


NP-hard
CIRCUIT-SAT
CLIQUE
NP-complete
3-SAT
NP
P

NP-completeness is everywhere

- There are thousands of known NP-complete problems!
- These problems come from many different areas of study
  - Logic, graph theory, number theory, scheduling, optimization, economics, physics, chemistry, biology, …

Proving that L is $\mathbf{NP}$-complete ("cheat sheet")

1. Prove that $L \in \text{NP}$
   - What is the certificate? How can you verify a purported certificate in polynomial time?
2. Prove that L is NP-hard
   - Which NP-complete language L_{HARD} are you reducing from?
   - What is the reduction? Is it polynomial-time computable?
   - YES maps to YES: Assume there is a certificate x showing $w \in L_{\text{HARD}}$. In terms of x, construct a certificate y showing that $f(w) \in L$.
   - NO maps to NO: (Contrapositive) Assume there is a certificate y showing $f(w) \in L$. In terms of y, construct a certificate x showing that $w \in L_{\text{HARD}}$.

NP-complete languages stand or fall together

- If you design a poly-time algorithm for one NP-complete language, then $\mathrm{P} = \mathrm{NP}$, so all NP-complete languages can be decided in poly time!

- If you prove that one NP-complete language cannot be decided in poly time, then $\mathrm{P} \neq \mathrm{NP}$, so no NP-complete language can be decided in poly time!

Final exam cutoff point

- Final exam will be **Wednesday, May 22 from 10am to noon** in this room (STU 105)
- The exam is cumulative
- To prepare for the final exam, you only need to study the material up to this point (including problem set 7)