CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

Circuit satisfiability

- We say that a circuit C is satisfiable if there exists $x \in \{0,1\}^n$ such that $C(x) = 1$
- Let $\text{CIRCUIT-SAT} = \{\langle C \rangle : C \text{ is a satisfiable circuit}\}$

Theorem: CIRCUIT-SAT is NP-complete.

Proof that $\mathrm{CIRCUIT\text{-}SAT} \in \mathrm{NP}$

- Given $\langle C \rangle$, where C is an n-input 1-output circuit:
  1. Pick $x \in \{0,1\}^n$ at random
  2. Check whether $C(x) = 1$ (recall $\mathrm{CIRCUIT\text{-}VALUE} \in \mathrm{P}$)
  3. Accept if $C(x) = 1$; reject if $C(x) = 0$

Proof that $\mathrm{CIRCUIT\text{-}SAT}$ is NP-hard

- Let L be any language in NP
- Our job: Design a mapping reduction from L to $\mathrm{CIRCUIT\text{-}SAT}$
- Idea: Let's build a "verification circuit" for L

Proof that CIRCU

Given w, how do we efficiently construct $\langle C \rangle$?

A: Simulate M on w and output the corresponding circuit

B: Calculate t, then make a $t \times t$ grid of copies of C_0 (a fixed circuit)

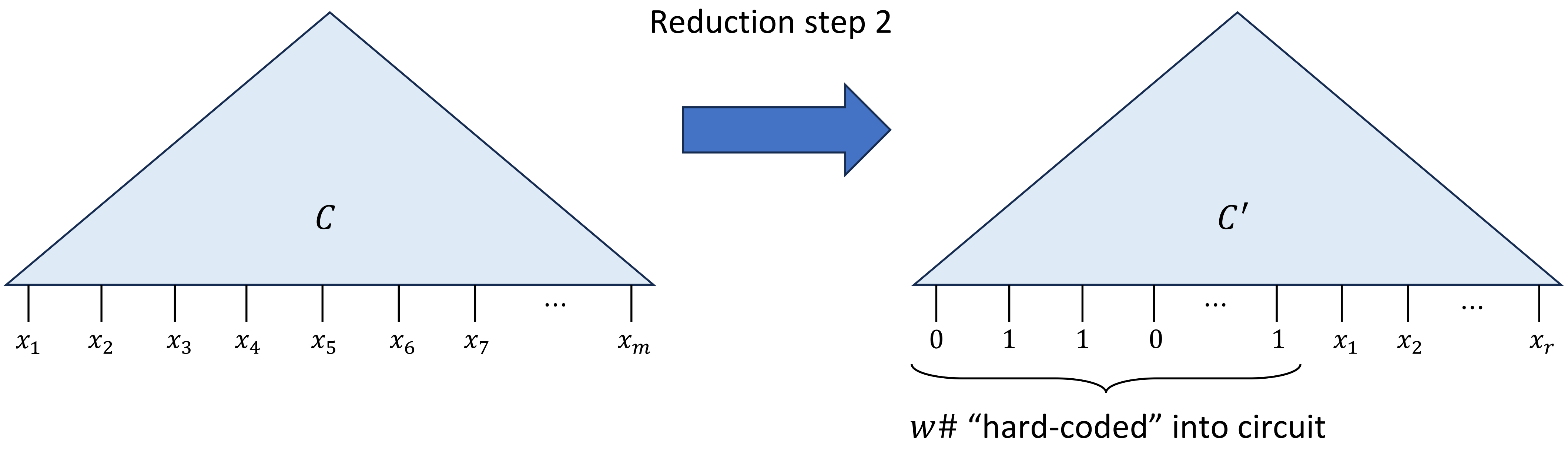
C: Inspect the transition function of M and implement it as a circuit

D: We don't; all that matters is the fact that $\langle C \rangle$ exists

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

- Let M be a nondeterminis
- Let $R = \{w\#u : M$ accepts w when initialized with u on tape $2\}$
- $R \in \mathrm{P} \subseteq \mathrm{PSIZE}$, and as discussed last time, the circuits are efficiently constructible
- Reduction step 1: Given w, construct $\langle C \rangle$, where C is a circuit that computes R_m for $m = |w| + 1 + T(|w|)$

Proof that CIRCUIT-SAT is NP-hard



- $w \in L$ if and only if there exists u such that $C(\langle w\#u \rangle) = 1$
- Reduction: $f(w) = \langle C' \rangle$, where $C'(\langle u \rangle) = C(\langle w\#u \rangle)$

Proof that CIRCUIT-SAT is NP-hard

- Reduction: $f(w) = \langle C' \rangle$, where $C'(\langle u \rangle) = C(\langle w\#u \rangle)$ and C computes R_m
- YES maps to YES:
  - If $w \in L$, then $\Pr[M \text{ accepts } w] \neq 0$
  - Therefore, there exists $u \in \{0,1\}^{T(|w|)}$ such that $w\#u \in R$
  - Therefore, $C(\langle w\#u \rangle) = 1$
  - Therefore, $C'(\langle u \rangle) = 1$, so C' is satisfiable ✓

Proof that CIRCUIT-SAT is NP-hard

- Reduction: $f(w) = \langle C' \rangle$, where $C'(\langle u \rangle) = C(\langle w\#u \rangle)$ and C computes R_m
- NO maps to NO:
  - Suppose C' is satisfiable, i.e., there exists $u \in \{0, 1\}^{T(|w|)}$ such that $C'(\langle u \rangle) = 1$
  - Then $C(\langle w\#u \rangle) = 1$, so $w\#u \in R$
  - Therefore, $\Pr[M \text{ accepts } w] \neq 0$
  - Therefore, $w \in L$ ✔

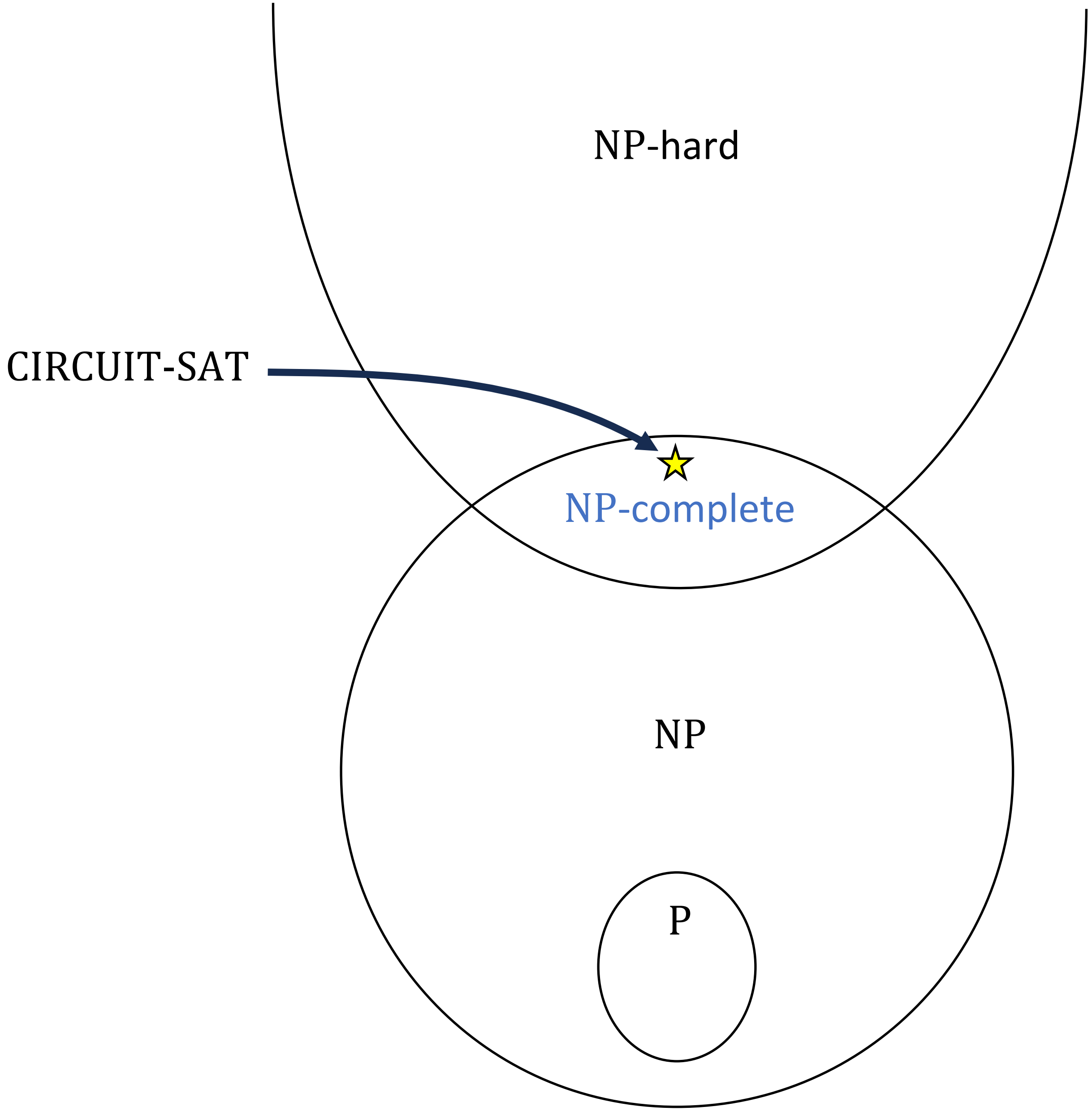
Proof that $\mathbf{CIRCUIT\text{-}SAT}$ is $\mathbf{NP}$-hard

- Reduction: $f(w) = \langle C' \rangle$, where $C'(\langle u \rangle) = C(\langle w\#u \rangle)$ and C computes R_m

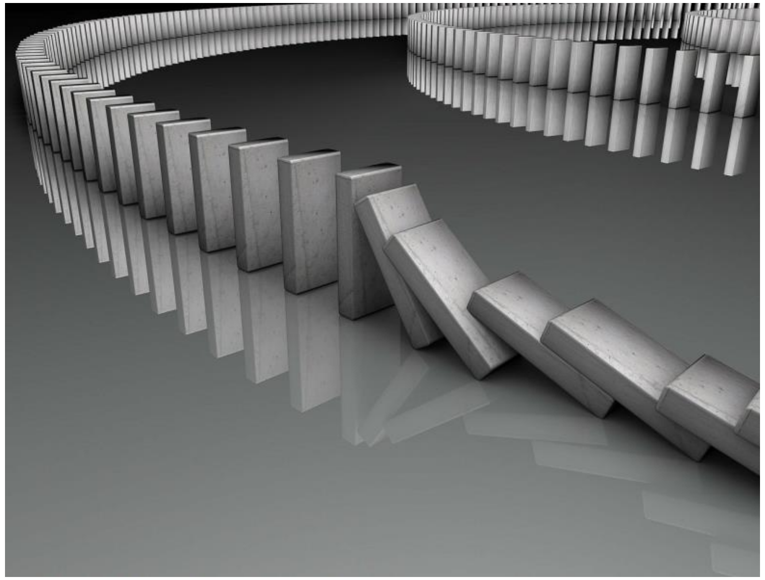
- Polynomial-time computable:

  1. Compute $\langle C \rangle$. This takes $\mathrm{poly}(m) = \mathrm{poly}(n)$ time ✓
  2. Plug in $w\#$. This takes $\mathrm{poly}(n)$ time ✓

NP-completeness



What else is NP-complete?

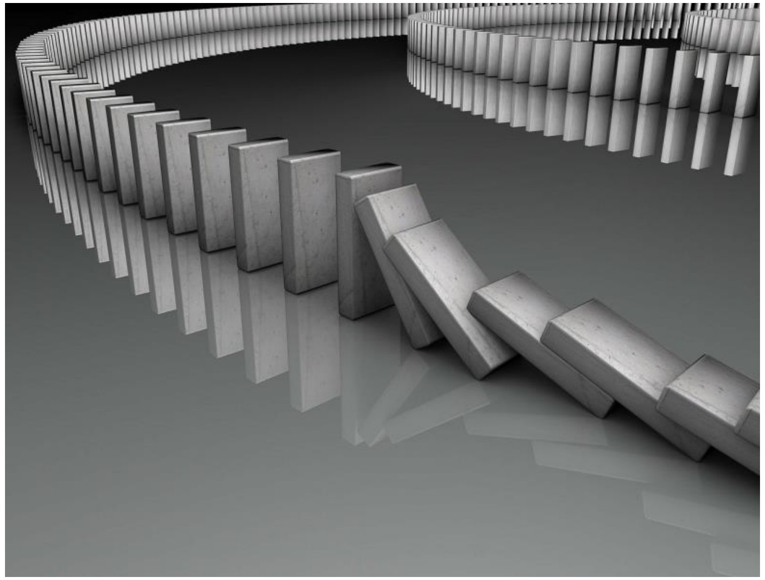


- We showed that CIRCUIT-SAT is NP-complete
- It turns out that a huge number of natural, interesting, and important languages are NP-complete!
- For example, we still want to show that CLIQUE is NP-complete…
- To prove NP-hardness, we can chain reductions together

Chaining reductions together

- **Claim:** Suppose L_{HARD} is NP-hard and there is a polynomial-time mapping reduction f from L_{HARD} to L_{NEW}. Then L_{NEW} is NP-hard
- **Proof:** Let L be any language in NP
- There is a polynomial-time mapping reduction g from L to L_{HARD}
- Reduction from L to L_{NEW}: $h(w) = f\big(g(w)\big)$
- Poly-time computable ✔ YES maps to YES ✔ NO maps to NO ✔

Chaining reductions together



- Consequence: To prove that CLIQUE is NP-complete, there is no need to reduce from an arbitrary language in NP
- We “merely” need to do a reduction from the one language CIRCUIT-SAT
- That’s nice, but it’s still not clear how to reduce CIRCUIT-SAT to CLIQUE…
- Plan: We will reduce CIRCUIT-SAT to “3-SAT” and “3-SAT” to CLIQUE

k-CNF formulas

- Recall: A CNF formula is an "AND of ORs of literals"
- **Definition:** A k-CNF formula is a CNF formula 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-SAT is NP-complete

- **Proof:** We need to show two things.

1. We need to show $3\text{-SAT} \in \text{NP}$. What is the certificate?

2. We need to show that 3-SAT is NP-hard. Reduction from CIRCUIT-SAT

Gate gadgets

- Define the following Boolean functions:

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

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

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

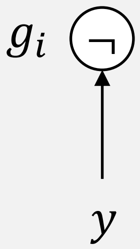
- Each can be represented by a 3-CNF formula. (Every function has a CNF representation!)

Reduction from CIRCUIT-SAT to 3-SAT

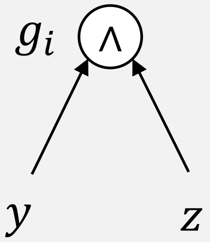
- Reduction: $f(\langle C \rangle) = \langle \phi \rangle$, where ϕ is a 3-CNF defined as follows
- Circuit C has variables $x_1, x_2, \ldots, x_n$ and AND/OR/NOT gates $g_1, \ldots, g_m$
- Assume without loss of generality that g_m is the output gate
- Formula ϕ has $n + m$ variables, which we denote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

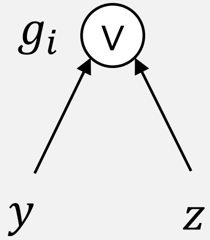
- For each AND/OR/NOT gate g_i in the circuit C, define a 3-CNF ϕ_i:



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



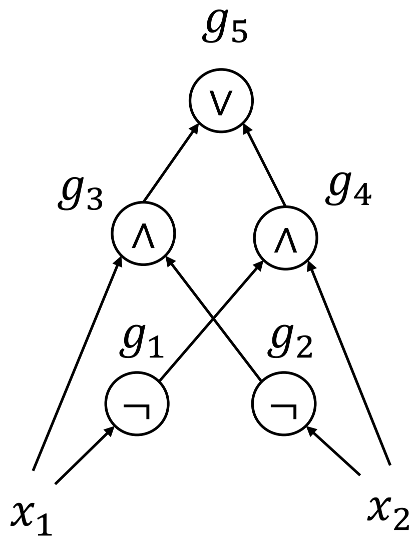
$\phi_i = \text{CHECK-AND}(g_i, y, z)$



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

- 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