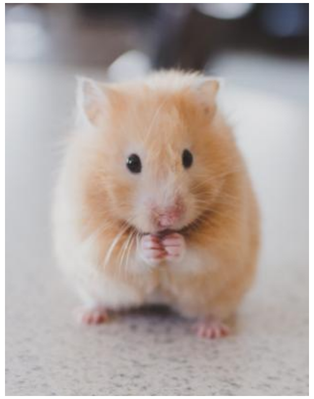
CMSC 28100

Introduction to Complexity Theory

Autumn 2025
Instructor: William Hoza

The complexity class NP

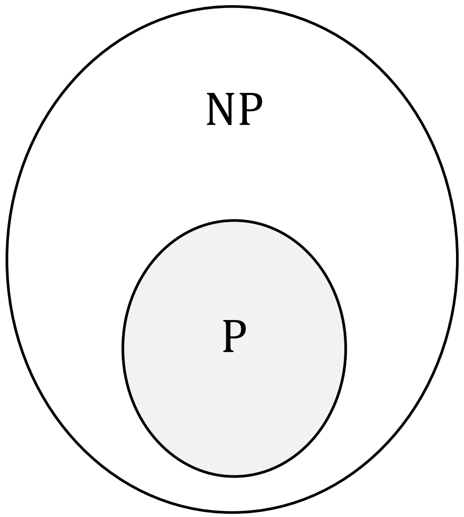


- Let $Y \subseteq \{0,1\}^*$

- **Definition:** $Y \in \mathrm{NP}$ if there exists a randomized polynomial-time Turing machine M such that $w \in Y \Leftrightarrow \Pr[M \text{ accepts } w] \neq 0$

- **Fact:** $Y \in \mathrm{NP}$ if and only if there exists a polynomial-time verifier for Y

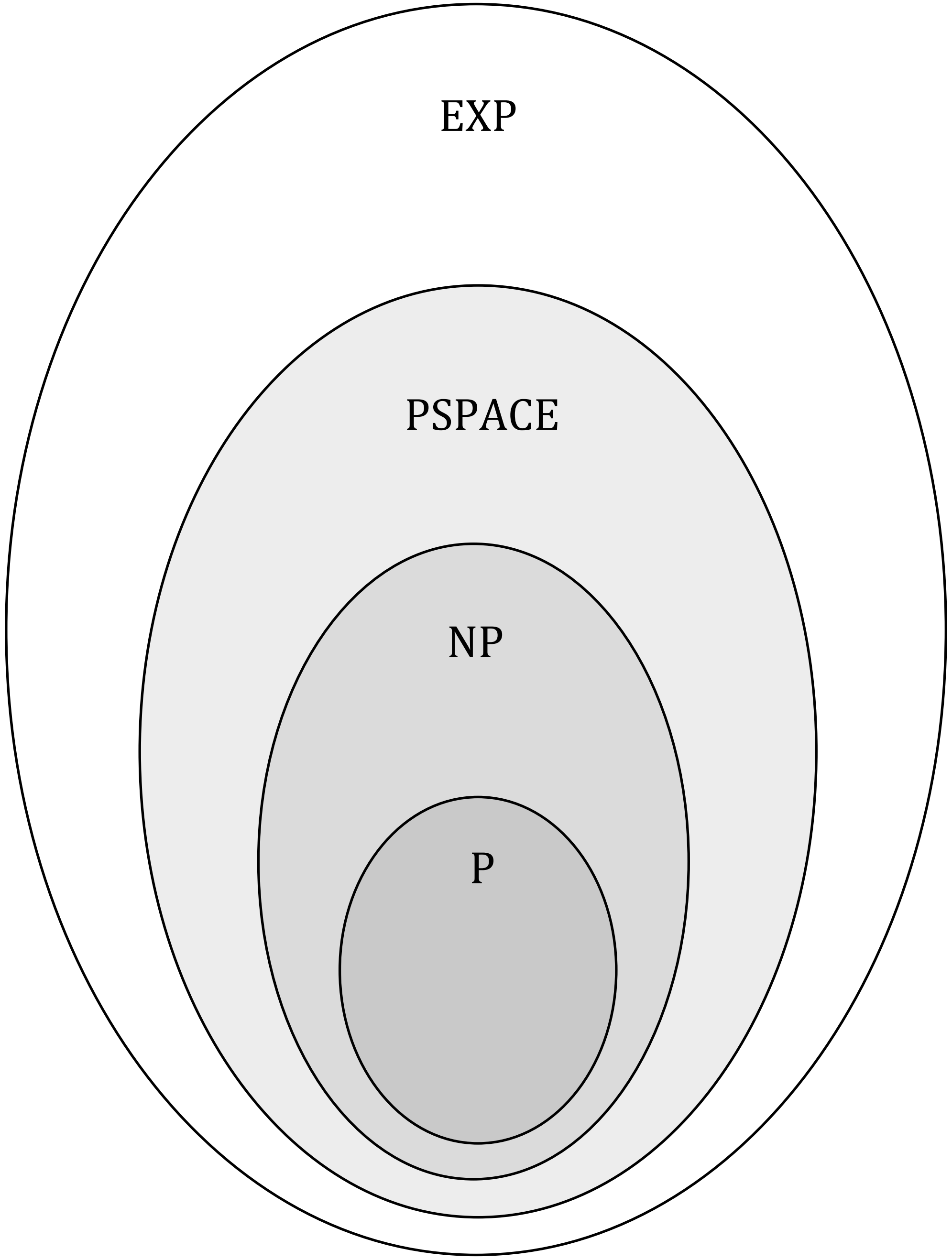
The P vs. NP problem



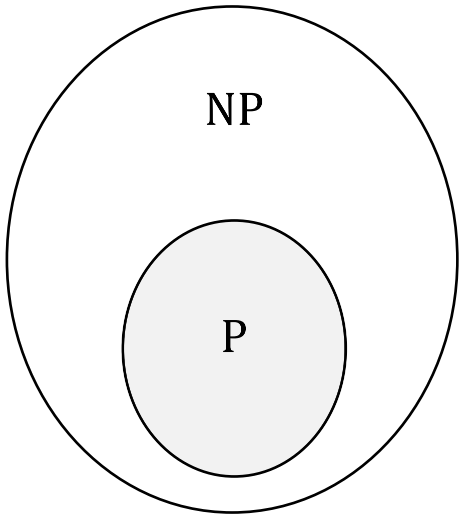
- $\mathrm{P} \subseteq \mathrm{NP}$ (why?)
- **Open question:** Does $\mathrm{P} = \mathrm{NP}$?
- The Clay Mathematics Institute will give you $1 million if you prove $\mathrm{P} = \mathrm{NP}$ or if you prove $\mathrm{P} \neq \mathrm{NP}$
- Let $Y \in \mathrm{NP}$. What can we do if we want to decide Y deterministically?

Solving problems in NP by brute force

- **Claim:** $\mathrm{NP} \subseteq \mathrm{PSPACE}$
- **Proof:** Let M be a time-n^k nondeterministic TM. Given $w \in \{0,1\}^n$:
  1. For every $x \in \{0,1\}^{n^k}$, simulate M, initialized with w on tape 1 and x on tape 2
  2. If we find some x such that M accepts, accept. Otherwise, reject
- NP can be informally defined as "the set of problems that can be solved by brute-force search"


EXP
PSPACE
NP
P

The P vs. NP problem



- "$\mathrm{P} = \mathrm{NP}$" would mean:
  - Brute-force search algorithms can always be converted into poly-time algorithms
  - Verifying someone else's solution is never significantly easier than solving a problem from scratch
- This would be counterintuitive!

Conjecture: $\mathrm{P} \neq \mathrm{NP}$

Comparing NP and BPP

- **Conjecture:** $\mathrm{P} \neq \mathrm{NP}$
  - It's hard to find a needle in a haystack
- **Conjecture:** $\mathrm{P} = \mathrm{BPP}$
  - It's easy to find hay in a haystack!

Complexity of CLIQUE

- Recall: $\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ has a } k\text{-clique}\}$
- Previously discussed: $\mathrm{CLIQUE} \in \mathrm{NP}$
- Consequence: If $\mathrm{P} = \mathrm{NP}$, then $\mathrm{CLIQUE} \in \mathrm{P}$
- **Plan:** We will prove that if $\mathrm{P} \neq \mathrm{NP}$, then $\mathrm{CLIQUE} \notin \mathrm{P}$
    - This will provide evidence that $\mathrm{CLIQUE} \notin \mathrm{P}$
- To prove it, we will use concepts of NP-hardness and NP-completeness

NP-hardness

- Let $Y \subseteq \{0, 1\}^*$
- **Definition:** Y is NP-hard if, for every $L \in \mathrm{NP}$, we have $L \leq_{\mathrm{P}} Y$
- Interpretation:
  - Y is at least as hard as any language in NP
  - Every problem in NP is basically a special case of Y

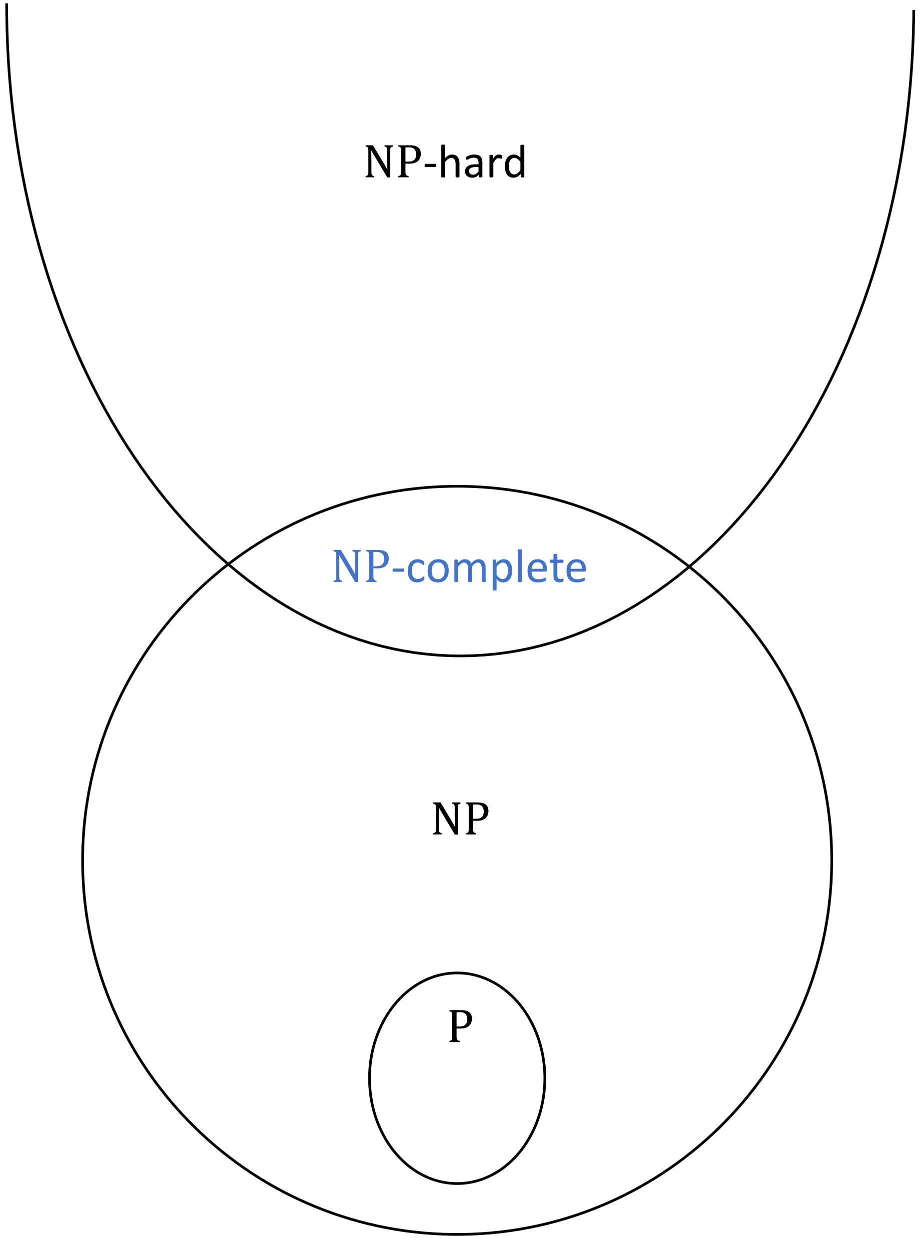
NP-completeness

- Let $Y \subseteq \{0, 1\}^*$
- **Definition:** Y is NP-complete if Y is NP-hard and $Y \in \mathrm{NP}$
- The NP-complete languages are the hardest languages in NP
- If Y is NP-complete, then the language Y can be said to “capture” / “express” the entire complexity class NP
- Example: We will eventually prove that CLIQUE is NP-complete

NP-complete languages are probably not in P

- Let Y be an NP-complete language
- **Claim:** $Y \in \mathrm{P}$ if and only if $\mathrm{P} = \mathrm{NP}$
- **Proof:**
  - ($\Leftarrow$) This holds because $Y \in \mathrm{NP}$ ✔
  - ($\Rightarrow$) This holds because Y is NP-hard ✔

NP-completeness

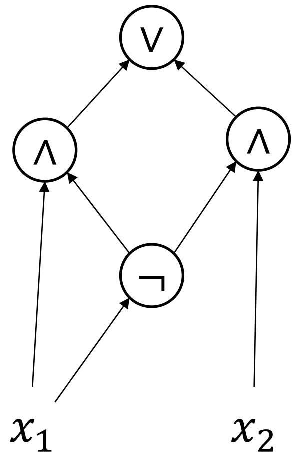


Proving $\mathbf{NP}$-completeness

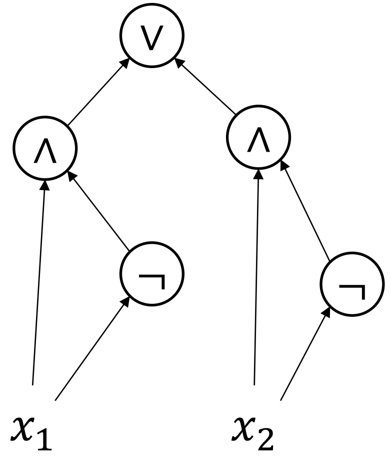
- We will prove that several interesting languages, including CLIQUE, are NP-complete
- This will provide evidence that these languages are intractable
- First example: The circuit satisfiability problem

Circuit satisfiability

- Let C be an n-input 1-output circuit
- We say that C is satisfiable if there exists $x \in \{0,1\}^n$ such that $C(x) = 1$



Satisfiable ✔



Unsatisfiable ❌

Circuit satisfiability is NP-complete

- Let $\mathrm{CIRCUIT\text{-}SAT} = \{\langle C \rangle : C \text{ is a satisfiable circuit}\}$

Theorem: $\mathrm{CIRCUIT\text{-}SAT}$ is NP-complete.

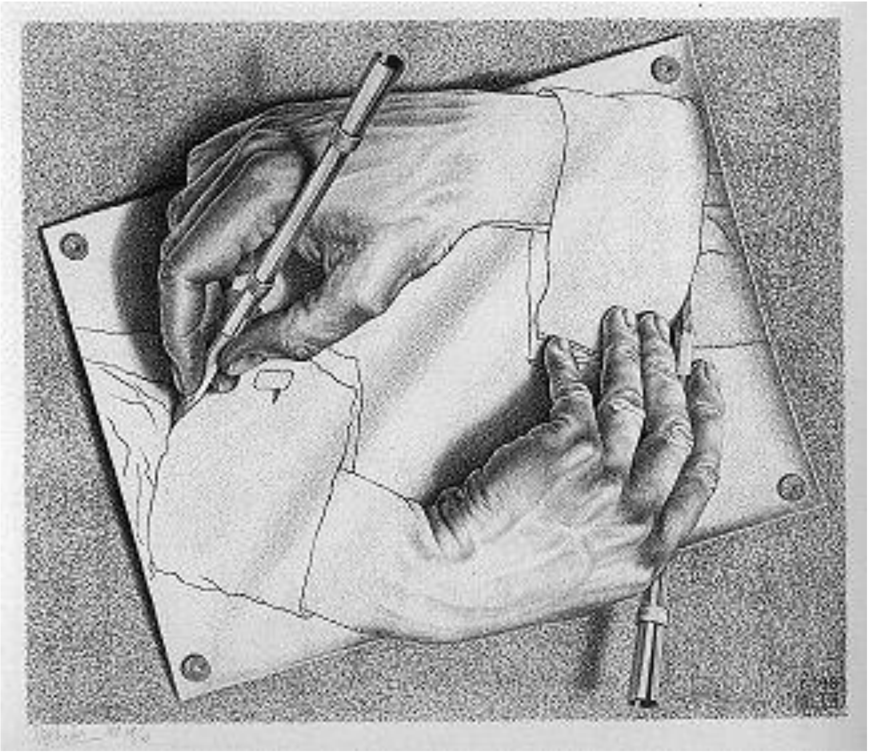
- Proof: Next 6 slides

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

- 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 $\text{CIRCUIT-VALUE} \in \text{P}$)
    3. Accept if $C(x) = 1$; reject if $C(x) = 0$

Code as data IV

- Let $Y \in \mathrm{NP}$
- To prove that $\mathrm{CIRCUIT\text{-}SAT}$ is NP-hard, we need to prove $Y \leq_{\mathrm{P}} \mathrm{CIRCUIT\text{-}SAT}$
- Given $w \in \{0, 1\}^*$, we will construct a circuit that is satisfiable if and only if $w \in Y$
- Idea: Build a "verification circuit"



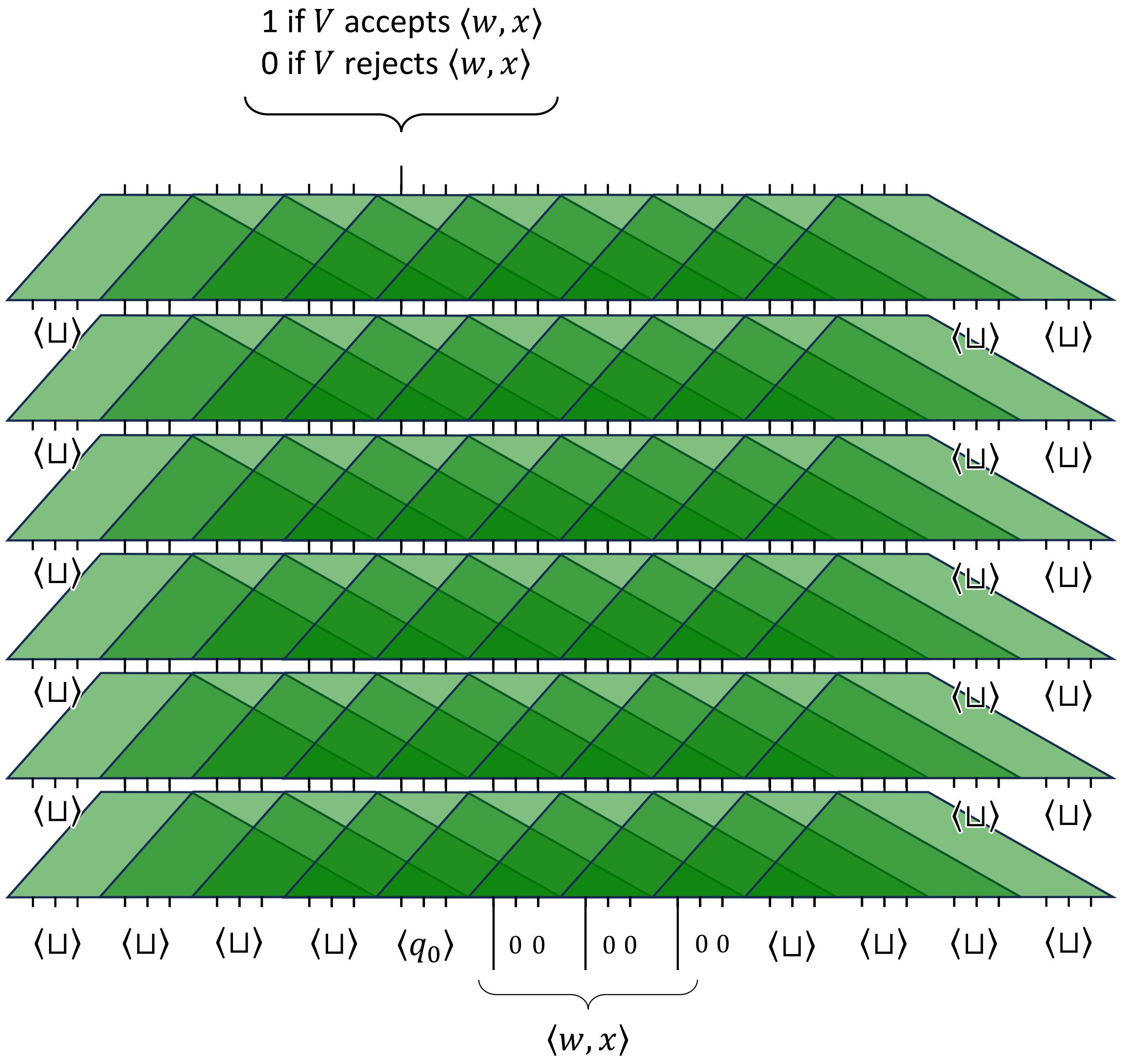
"Drawing Hands."
(1948 lithograph by M. C. Escher)

Constructing the verification circuit

- Let V be a poly-time verifier for Y with certificates of length n^k
- Let $w \in \{0,1\}^n$
- $w \in Y$ if and only if there exists $x \in \{0,1\}^{n^k}$ such that V accepts $\langle w, x \rangle$

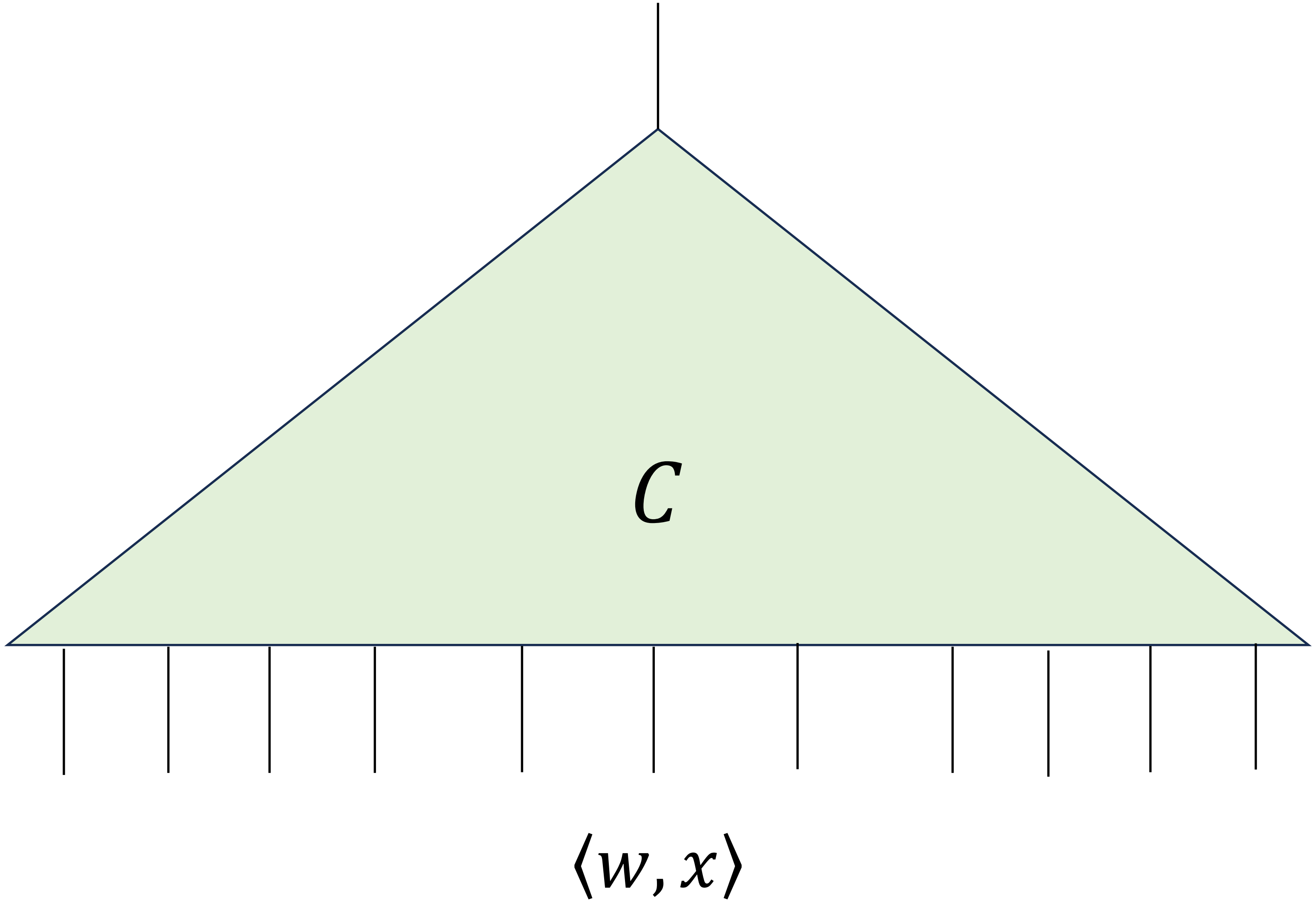
TM ⇒ Circuit

- Step 1: Construct a circuit C that simulates the verifier V
- (Recall $\mathrm{P} \subseteq \mathrm{PSIZE}$ proof)

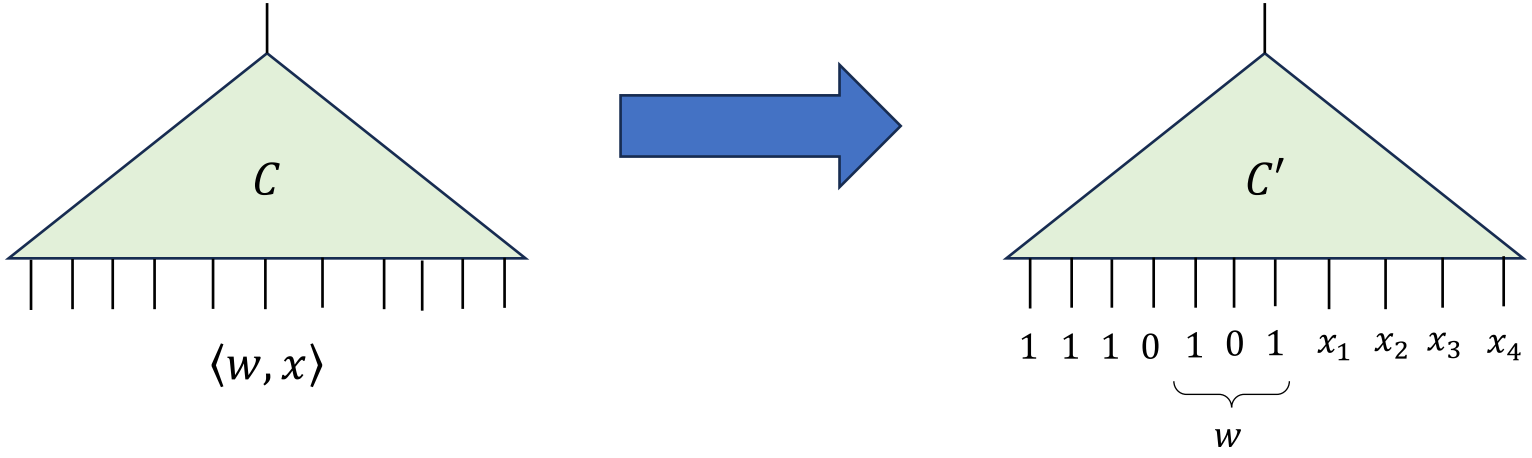


TM $\Rightarrow$ Circuit

- Step 1: Construct a circuit C that simulates the verifier V
- (Recall $\mathrm{P} \subseteq \mathrm{PSIZE}$ proof)



Step 2: Hard-coding



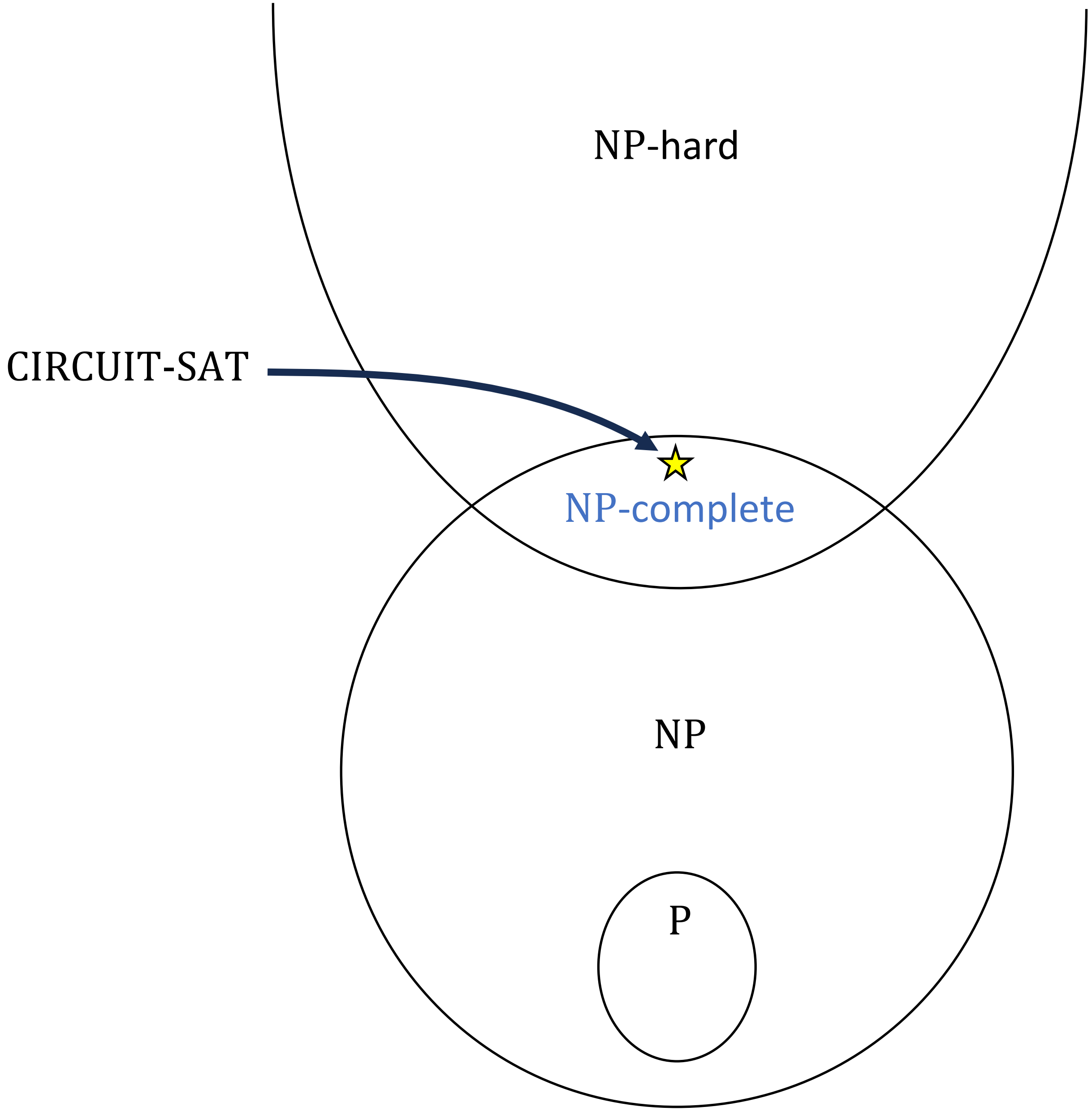
- Poly-time computable ✓
- YES maps to YES: If $w \in Y$, then C' is satisfiable ✓
- NO maps to NO: If $w \notin Y$, then C' is not satisfiable ✓

- Hard-code the original input w, so the input to C' is x (certificate)
  - (Recall $\mathrm{P/poly} \subseteq \mathrm{PSIZE}$ proof)
  - Technical detail: Use the encoding $\langle w, x \rangle = 1^{|w|}0wx$
- Reduction: $\Psi(w) = \langle C' \rangle$

Theorem: CIRCUIT-SAT is NP-complete.

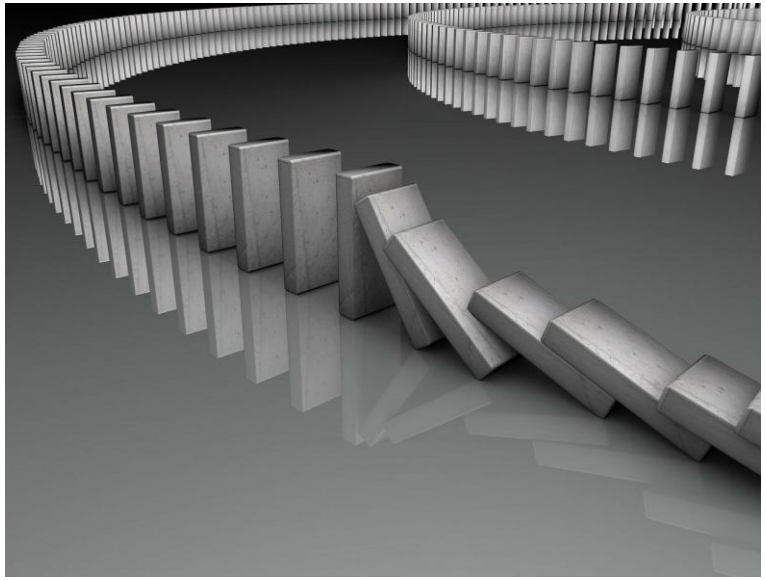
- Make sure you thoroughly understand this theorem and its proof!
- A ton of key concepts from this course come together here!

NP-completeness

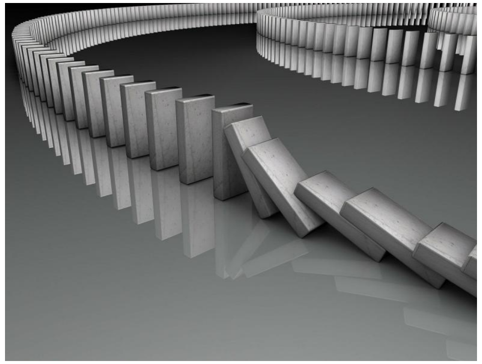


What else is NP-complete?

- We showed that $\mathrm{CIRCUIT\text{-}SAT}$ is NP-complete
- This will help us to prove that other problems, such as CLIQUE, are also NP-complete
- Idea: Chain reductions together



Chaining reductions together



- **Claim:** If $Y_1 \leq_{\mathrm{P}} Y_2 \leq_{\mathrm{P}} Y_3$, then $Y_1 \leq_{\mathrm{P}} Y_3$
- **Proof:** Let $\Psi_{1\to 2}$ and $\Psi_{2\to 3}$ be the mapping reductions
- Reduction from Y_1 to Y_3 is $\Psi(w) = \Psi_{2\to 3}\left(\Psi_{1\to 2}(w)\right)$
  - YES maps to YES ✔
  - NO maps to NO ✔
  - Poly-time computable, because $|\Psi_{1\to 2}(w)| \leq \mathrm{poly}(|w|)$

Chaining reductions together