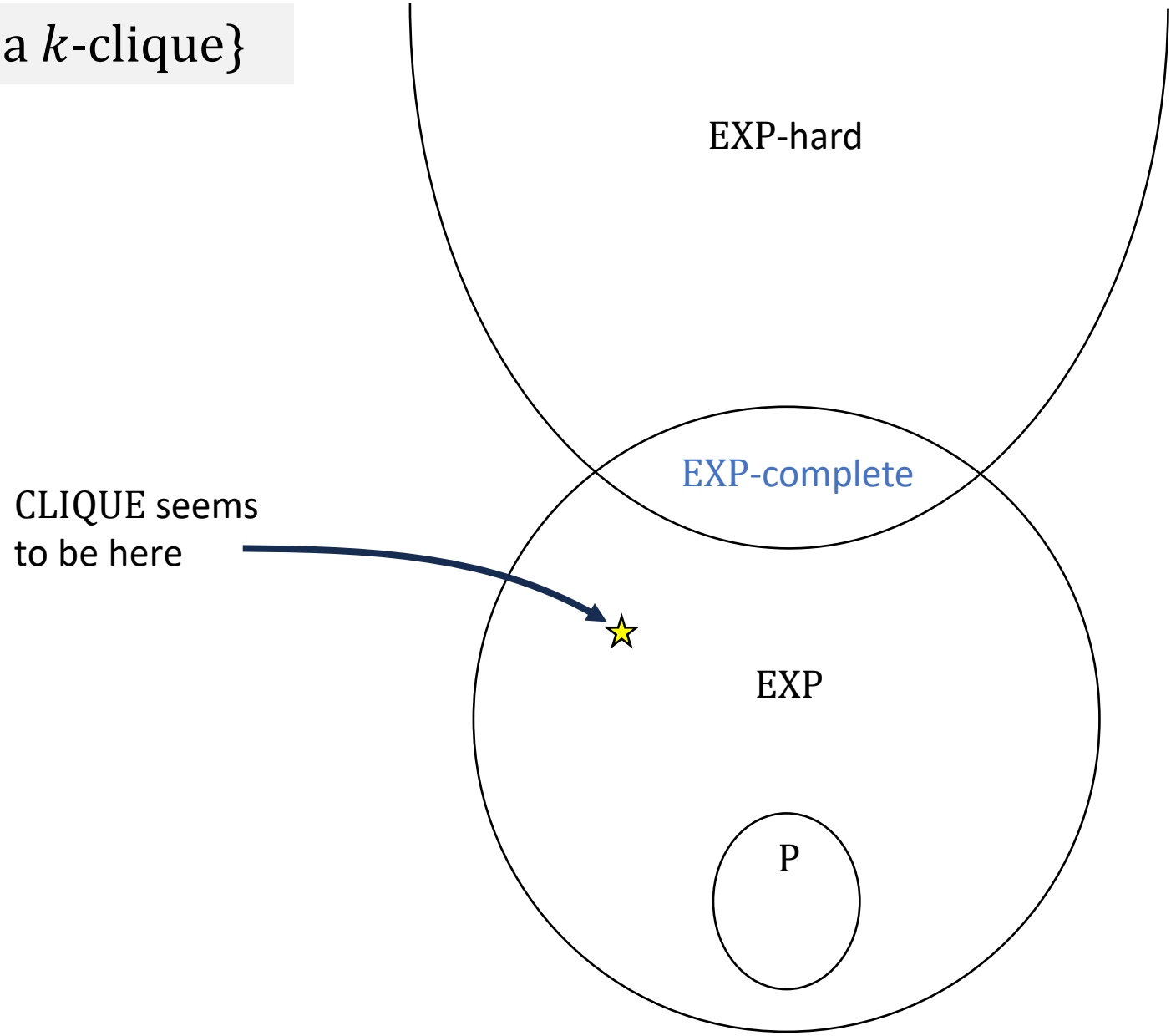
CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

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



Complexity of the clique problem

- Evidently, to understand the complexity of CLIQUE, we need

  new conceptual tools

Guessing and checking

- **Key insight:** There exists a polynomial-time randomized Turing machine M with the following properties.
  - If $\langle G, k \rangle \notin \text{CLIQUE}$, then $\Pr[M \text{ accepts } \langle G, k \rangle] = 0$.
  - If $\langle G, k \rangle \in \text{CLIQUE}$, then $\Pr[M \text{ accepts } \langle G, k \rangle] \neq 0$.

  } "Nondeterministic TM"

- **Proof:** M picks a random subset of the vertices, accepts if it is a k-clique, and rejects otherwise.

The complexity class $\mathbf{NP}$

- Let $Y \subseteq \{0,1\}^*$
- **Definition:** $Y \in \mathrm{NP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \{0,1\}^*$:
  - If $w \in Y$, then $\Pr[M \text{ accepts } w] \neq 0$
  - If $w \notin Y$, then $\Pr[M \text{ accepts } w] = 0$
- "Nondeterministic Polynomial-time"

Another example of a language in $\mathbf{NP}$

- $\mathrm{FACTOR} = \{\langle K, M\rangle : K \text{ has a prime factor } p \leq M\}$
- **Claim:** $\mathrm{FACTOR} \in \mathrm{NP}$
- **Proof:**
  1. Pick $R \in \{2, 3, 4, \ldots, M\}$ uniformly at random
  2. Check whether K/R is an integer (long division)
  3. If it is, accept; if it isn't, reject

How to interpret NP

- NP is not intended to model the concept of tractability
- A nondeterministic polynomial-time algorithm is not a practical way to solve a problem
- Instead, NP is a conceptual tool for reasoning about computation

“Verification of certificates” perspective

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

“Verifier”

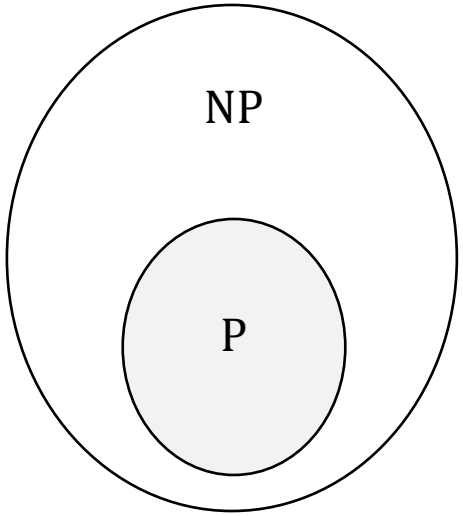
- **Claim:** $Y \in \mathrm{NP}$ iff there exists $k \in \mathbb{N}$ and a deterministic poly-time TM V such that:

“Certificate” / “Witness”

  - For every $w \in Y$, there exists x such that $|x| \leq |w|^k$ and V accepts $\langle w, x \rangle$
  - For every $w \notin Y$, for every x, the machine V rejects $\langle w, x \rangle$

- **Proof:** ($\Rightarrow$) Given $\langle w, x \rangle$, V simulates M with w on tape 1 and x on tape 2
- ($\Leftarrow$) Pick x at random and simulate V on $\langle w, x \rangle$

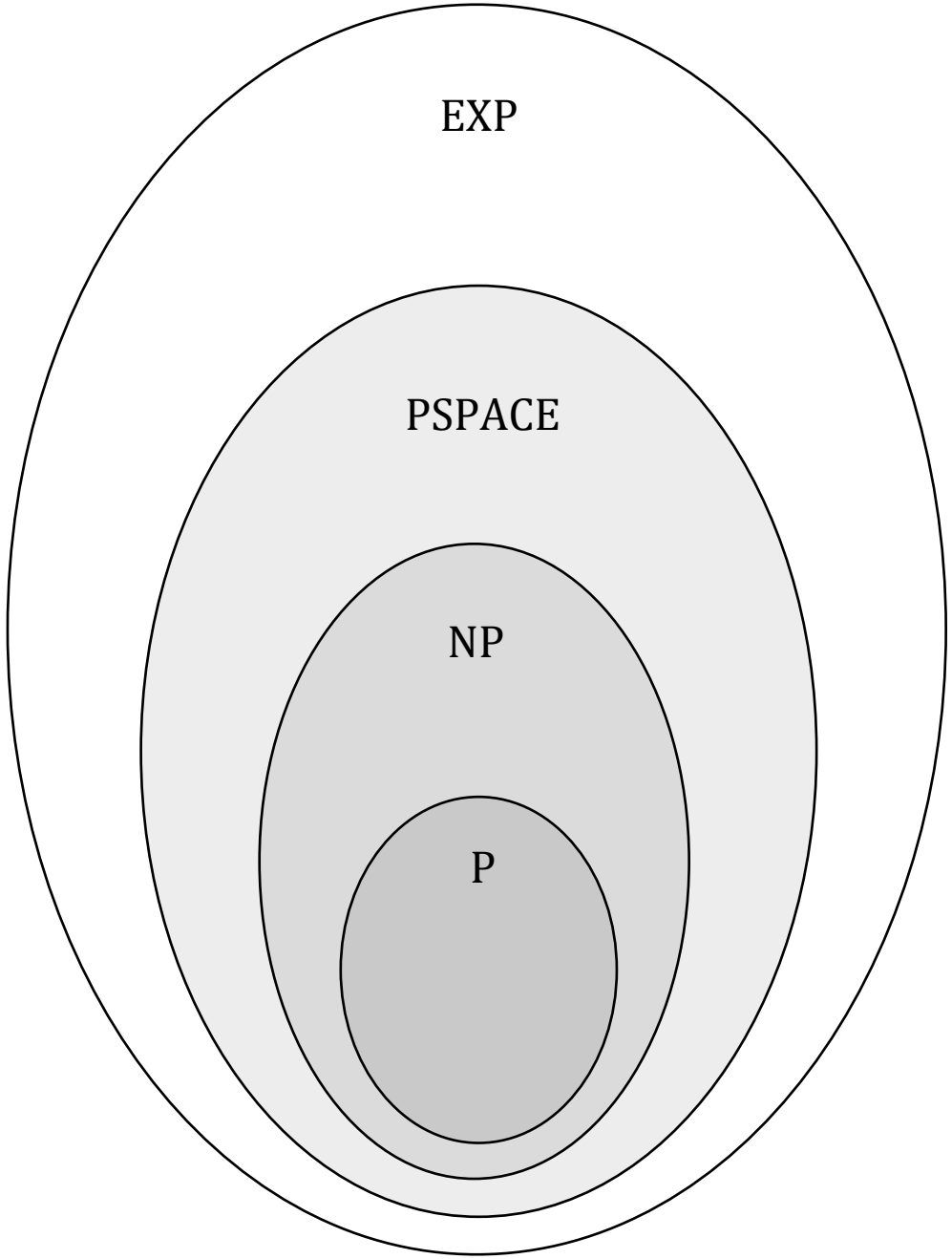
The P vs. NP problem



- $\mathrm{P} \subseteq \mathrm{NP}$ (why?)
- **Open question:** Does $\mathrm{P} = \mathrm{NP}$?
- Let $Y \in \mathrm{NP}$
- What can we do if we want to decide Y deterministically?

Solving problems in $\mathbf{NP}$ by brute force

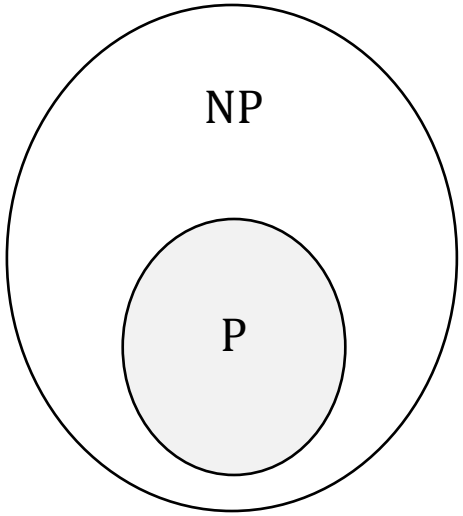
- **Claim:** $\mathrm{NP} \subseteq \mathrm{PSPACE}$
- **Proof:** Let M be a time-n^k nondeterministic TM. Given $w \in \{0,1\}^n$:
  1. For every $x \in \{0,1\}^{n^k}$, simulate M, initialized with w on tape 1 and x on tape 2
  2. If we find some x such that M accepts, accept. Otherwise, reject
- NP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


EXP
PSPACE
NP
P

P vs. NP vs. PSPACE vs. EXP

- $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$
- What we expect: All of these containments are strict
- What we can prove: At least one of these containments is strict. (Why?)

The P vs. NP problem



- "P = 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: $P \neq NP$

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!

The P vs. NP problem

- P vs. NP is one of the most important open questions in theoretical computer science and mathematics
- The Clay Mathematics Institute will give you $1 million if you prove $\mathrm{P} \neq \mathrm{NP}$ (or if you prove $\mathrm{P} = \mathrm{NP}$)

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:** We say that 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

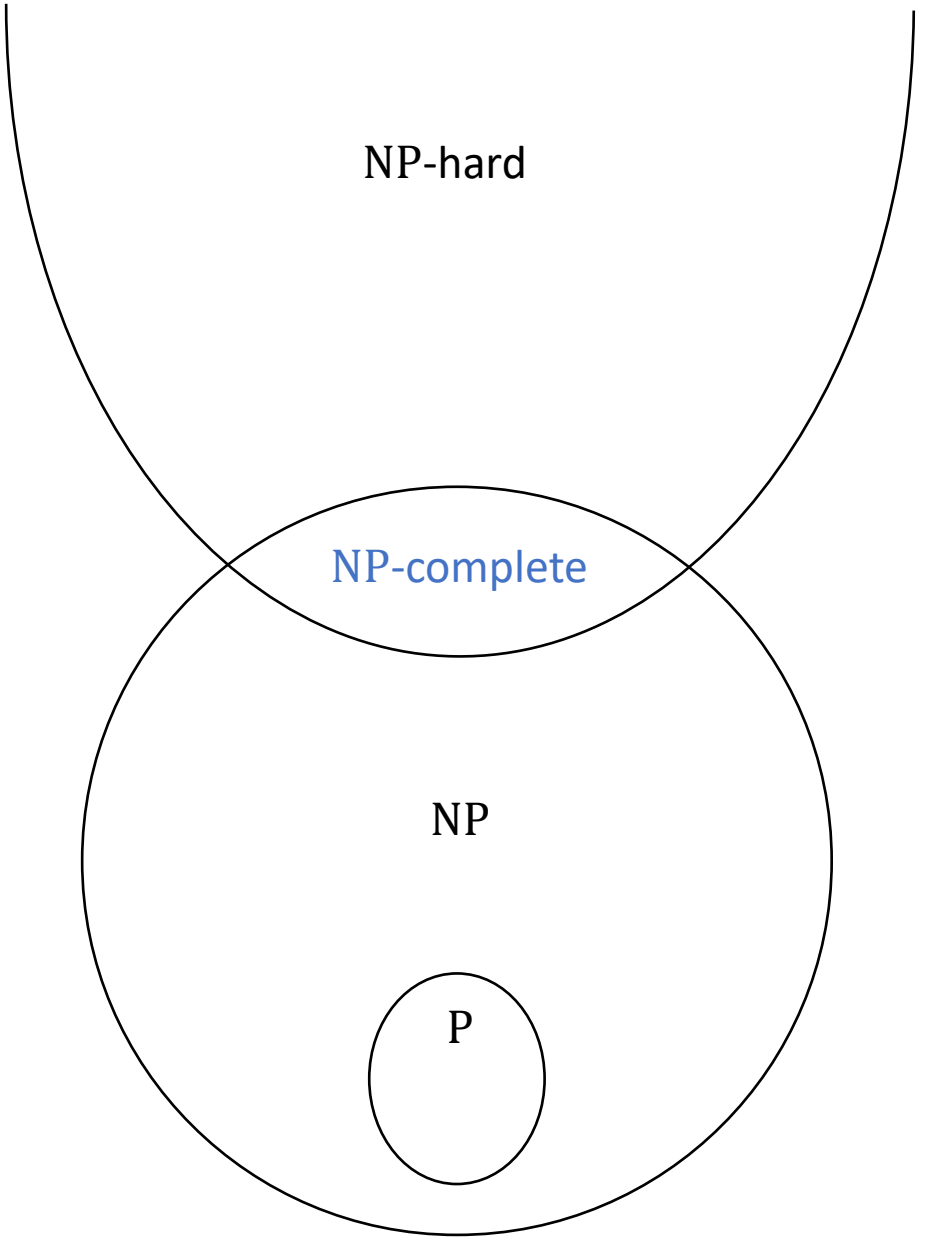
NP-completeness

- Let $Y \subseteq \{0,1\}^*$
- **Definition:** We say that 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 NP-completeness

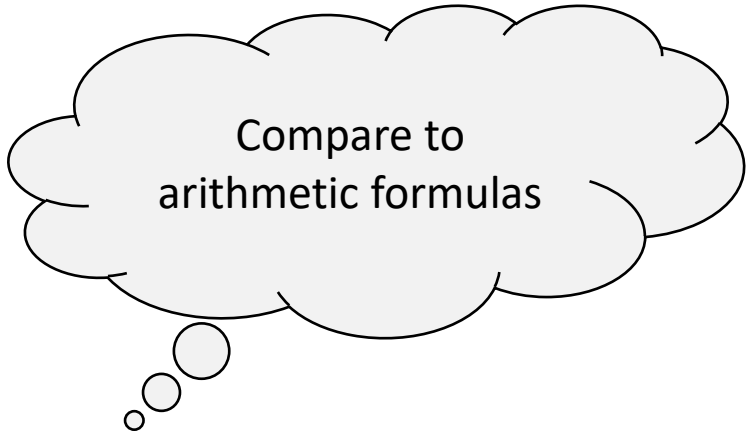
- How can we prove that a language like CLIQUE is NP-complete?
- How can we use graph theory to simulate Turing machines?
- Plan:

$$\text{Turing Machines} \Rightarrow \text{Logic Gates} \Rightarrow \text{Graph Theory}$$

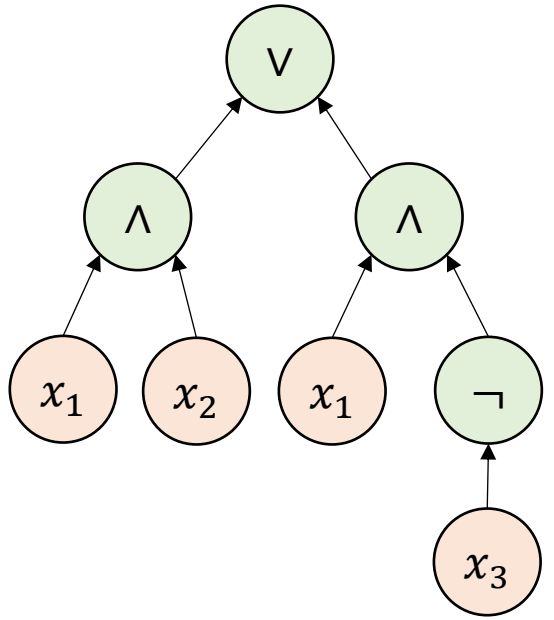
Logic gates

- AND: $a \wedge b$
- OR: $a \vee b$
- NOT: $\neg a$

Boolean formulas

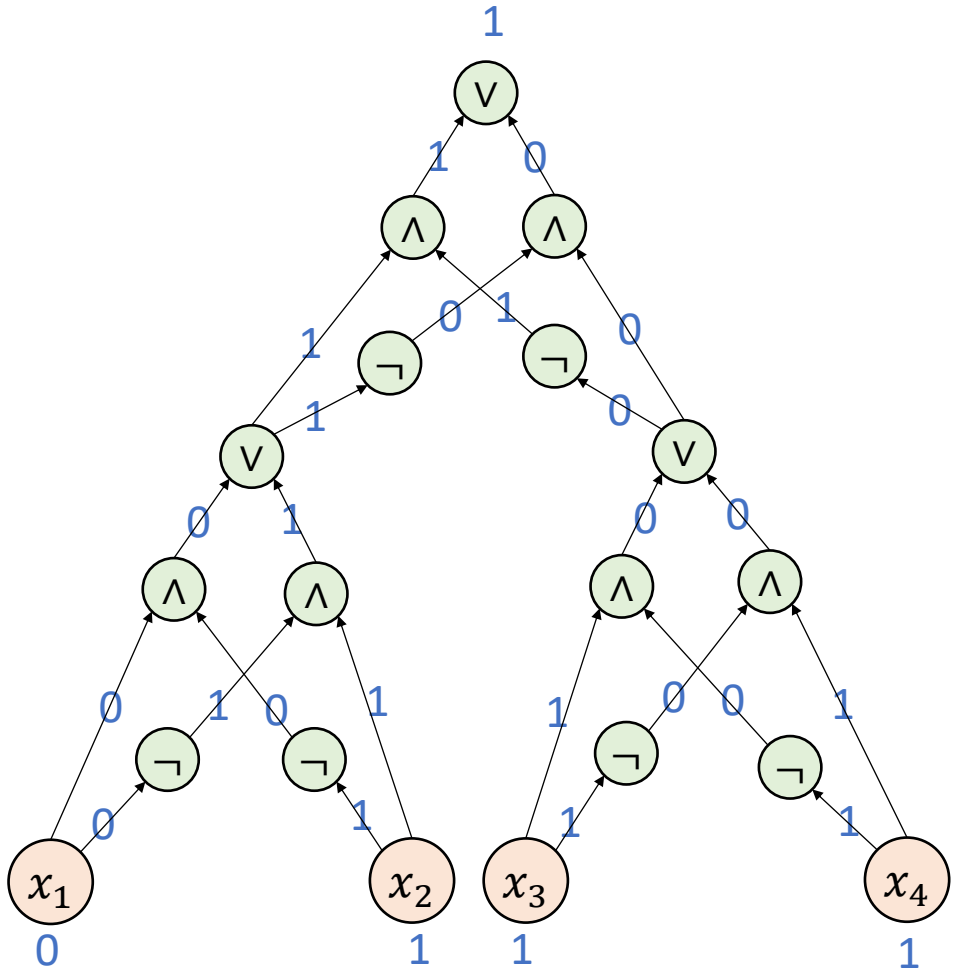


- **Definition:** An n-variate Boolean formula is a rooted tree
  - Each internal node is labeled ∨ or ∨ (two children) or ¬ (one child)
  - Each leaf is labeled with 0, 1, or a variable among $x_1, \ldots, x_n$
- It computes $f: \{0,1\}^n \to \{0,1\}$
- E.g., $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_3)$
  - Notation: $\bar{x}_i$ is another way of writing $\neg x_i$



Boolean circuits

- A Boolean circuit is like a Boolean formula, except that we permit vertices to have multiple outgoing wires



Boolean circuits: Rigorous definition

- **Definition:** An n-input m-output circuit is a directed acyclic graph
  - We refer to the edges as "wires"
  - Each node is labeled with one of the following:
    - $\wedge$ or $\vee$ (two incoming wires)
    - $\neg$ (one incoming wire)

    } "gates"

    - 0, 1, or a variable among $x_1, \ldots, x_n$ (zero incoming wires)
  - m of the nodes are additionally labeled as "output 1", "output 2", …, "output m"

Boolean circuits: Rigorous definition

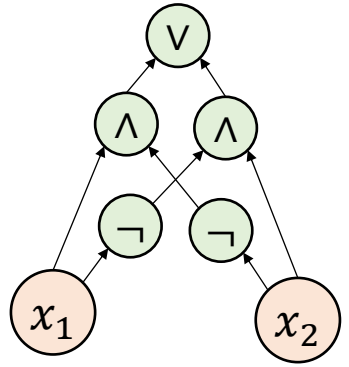
- Each node g computes a function $g: \{0,1\}^n \to \{0,1\}$ defined inductively:
  - If g is labeled x_i, then $g(x) =$ the i-th bit of x
  - If g is labeled $\neg$ and its incoming wire comes from f, then $g(x) = \neg f(x)$
  - If g is labeled $\wedge$ and its incoming wires come from f and h, then $g(x) = f(x) \wedge h(x)$
  - If g is labeled $\vee$ and its incoming wires come from f and h, then $g(x) = f(x) \vee h(x)$

Boolean circuits

- Let the output nodes be $g_1, \ldots, g_m$
- As a whole, the circuit computes $C: \{0,1\}^n \to \{0,1\}^m$ defined by

$$C(x) = \big(g_1(x), \ldots, g_m(x)\big)$$

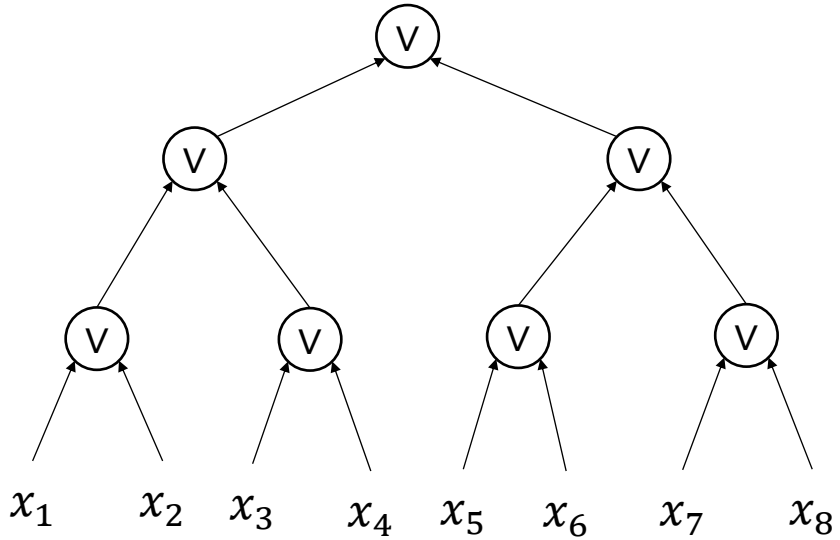
Circuit complexity



- The size of the circuit is the total number of AND/OR/NOT gates
  - How much "work" does the circuit do?
- Let $f: \{0,1\}^n \to \{0,1\}^m$
- The circuit complexity of f is the size of the smallest circuit that computes f
  - How much work is required to compute f?

Circuit complexity example 1

- Let $f(x) = x_1 \vee x_2 \vee \cdots \vee x_n$
- Circuit complexity: $\Theta(n)$



Circuit complexity example 2

- Let $f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$

What is the circuit complexity of f?

A: $\Theta(n^2)$

B: $O(1)$

C: $\Theta(n)$

D: $\Theta(2^n)$

Respond at PollEv.com/whoza or text “whoza” to 22333