NP-Completeness

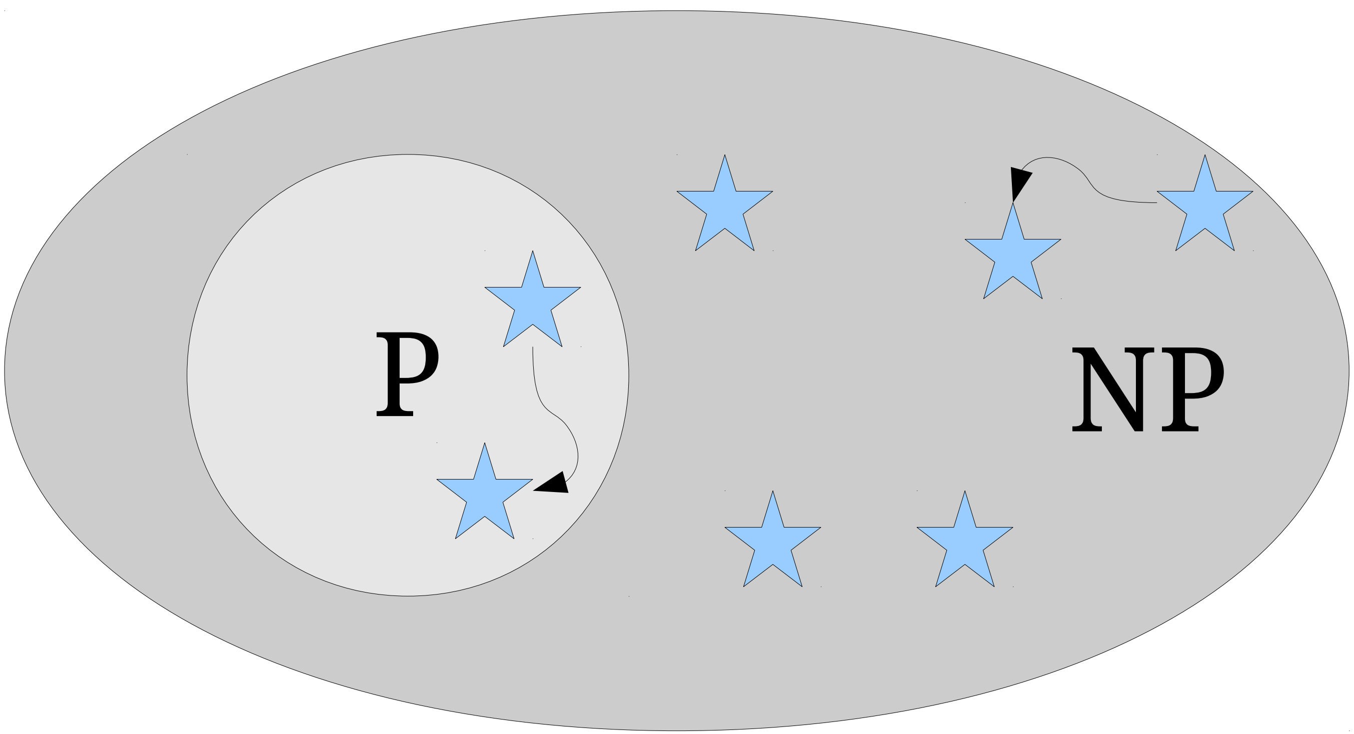
Part II

Outline for Today

- **Recap from Last Time**
  - What is **NP**-completeness again, anyway?
- **3SAT**
  - A simple, canonical **NP**-complete problem.
- **Independent Sets**
  - Discovering a new **NP**-complete problem.
- **Gadget-Based Reductions**
  - A common technique in **NP** reductions.
- **3-Colorability**
  - A more elaborate **NP**-completeness reduction.
- **Recent News**
  - Things happened! What were they?

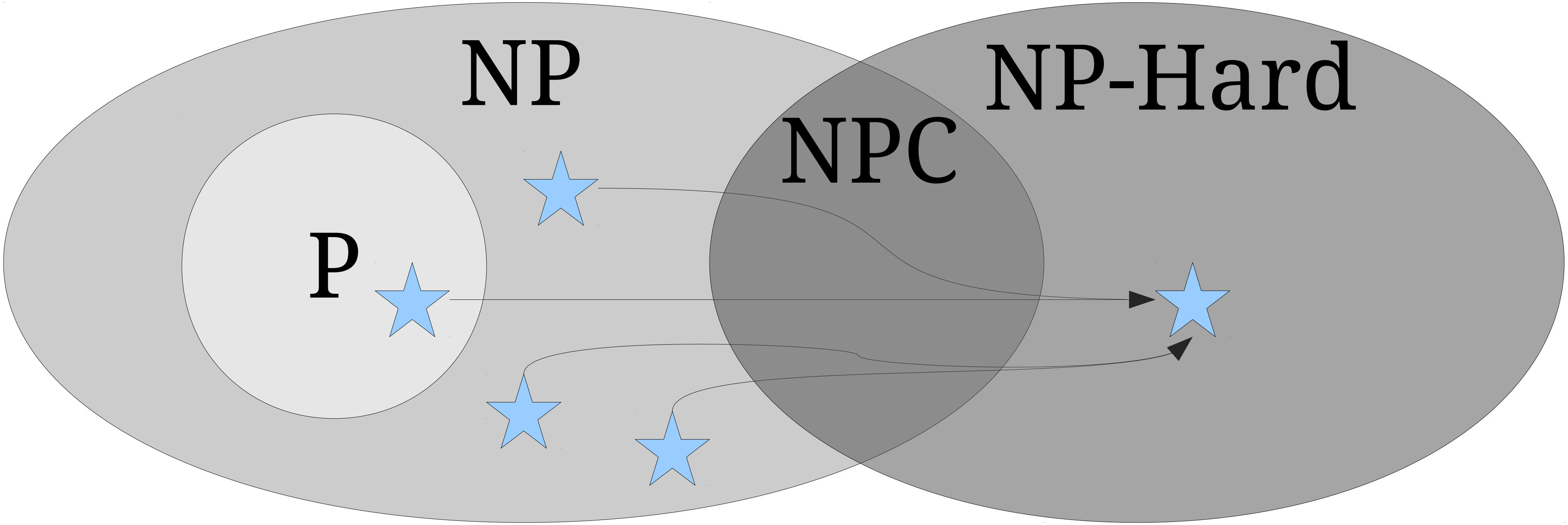
Polynomial-Time Reductions

- If $A \leq_{\text{P}} B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_{\text{P}} B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



NP-Hardness

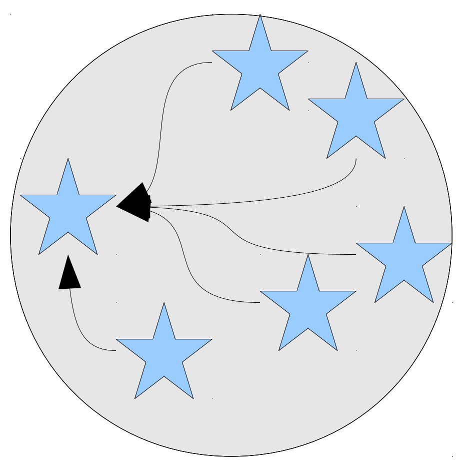
- A language L is called ***NP-hard*** if for *every* $A \in$ **NP**, we have $A \leq_P L$.
- A language in L is called ***NP-complete*** if L is **NP**-hard and $L \in$ **NP**.
- The class ***NPC*** is the set of **NP**-complete problems.



The Tantalizing Truth

Theorem: If *any* **NP**-complete language is in **P**, then **P** = **NP**.

Proof: Suppose that L is **NP**-complete and $L \in$ **P**. Now consider any arbitrary **NP** problem A. Since L is **NP**-complete, we know that $A \leq_p L$. Since $L \in$ **P** and $A \leq_p L$, we see that $A \in$ **P**. Since our choice of A was arbitrary, this means that **NP** $\subseteq$ **P**, so **P** = **NP**. ■

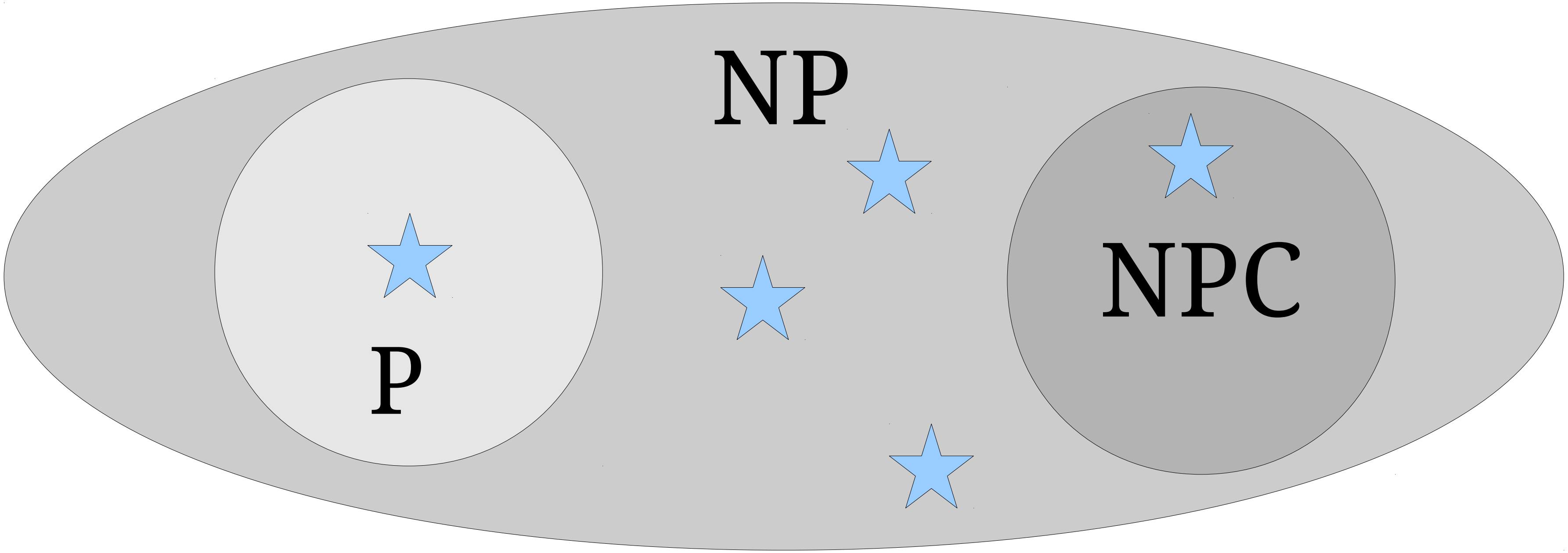


P = NP

The Tantalizing Truth

Theorem: If *any* **NP**-complete language is not in **P**, then $\mathbf{P} \neq \mathbf{NP}$.

Proof: Suppose that L is an **NP**-complete language not in **P**. Since L is **NP**-complete, we know that $L \in \mathbf{NP}$. Therefore, we know that $L \in \mathbf{NP}$ and $L \notin \mathbf{P}$, so $\mathbf{P} \neq \mathbf{NP}$. ■



How do we even know NP-complete problems exist in the first place?

Satisfiability

- A propositional logic formula φ is called ***satisfiable*** if there is some assignment to its variables that makes it evaluate to true.
  - $p \land q$ is satisfiable.
  - $p \land \neg p$ is unsatisfiable.
  - $p \to (q \land \neg q)$ is satisfiable.
- An assignment of true and false to the variables of φ that makes it evaluate to true is called a ***satisfying assignment***.

SAT

- The ***boolean satisfiability problem*** (***SAT***) is the following:

Given a propositional logic formula φ, is φ satisfiable?

- Formally:

$$\boldsymbol{SAT = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable PL formula} \}}$$

Theorem (Cook-Levin): SAT is **NP**-complete.

Proof: Read Sipser or take CS154!

New Stuff!

A Simpler **NP**-Complete Problem

Literals and Clauses

- A ***literal*** in propositional logic is a variable or its negation:
  - x
  - $\neg y$
  - But not $x \wedge y$.
- A ***clause*** is a many-way OR (*disjunction*) of literals.
  - $(\neg x \vee y \vee \neg z)$
  - (x)
  - But not $x \vee \neg(y \vee z)$

Conjunctive Normal Form

- A propositional logic formula φ is in ***conjunctive normal form*** (***CNF***) if it is the many-way AND (*conjunction*) of clauses.
  - $(x \vee y \vee z) \wedge (\neg x \vee \neg y) \wedge (x \vee y \vee z \vee \neg w)$
  - $(x \vee z)$
  - But not $(x \vee (y \wedge z)) \vee (x \vee y)$
- Only legal operators are ¬, ∨, ∧.
- No nesting allowed.

The Structure of CNF

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

For this formula to be satisfiable, each clause must have at least one true literal in it.

The Structure of CNF

$$\underbrace{(x \lor y \lor \neg z)} \land \underbrace{(\neg x \lor \neg y \lor z)} \land \underbrace{(\neg x \lor y \lor \neg z)}$$

We should pick at least one true literal from each clause…

The Structure of CNF

$$(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor z) \land (\neg x \lor y \lor \neg z)$$

… but never choose a literal and its negation

3-CNF

- A propositional formula is in ***3-CNF*** if
  - it is in CNF, and
  - every clause has *exactly* three literals.
- For example:
  - $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$
  - $(x \vee x \vee x) \wedge (y \vee \neg y \vee \neg x) \wedge (x \vee y \vee \neg y)$
  - but not $(x \vee y \vee z \vee w) \wedge (x \vee y)$
- The language ***3SAT*** is defined as follows:

3SAT = { ⟨φ⟩ | φ is a satisfiable 3-CNF formula }

Theorem: 3SAT is **NP**-Complete

Finding Additional **NP**-Complete Problems

NP-Completeness

Theorem: Let A and B be languages. If $A \leq_{\text{P}} B$ and A is **NP**-hard, then B is **NP**-hard.

Theorem: Let A and B be languages where $A \in \mathbf{NPC}$ and $B \in \mathbf{NP}$. If $A \leq_{\text{P}} B$, then $B \in \mathbf{NPC}$.