CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

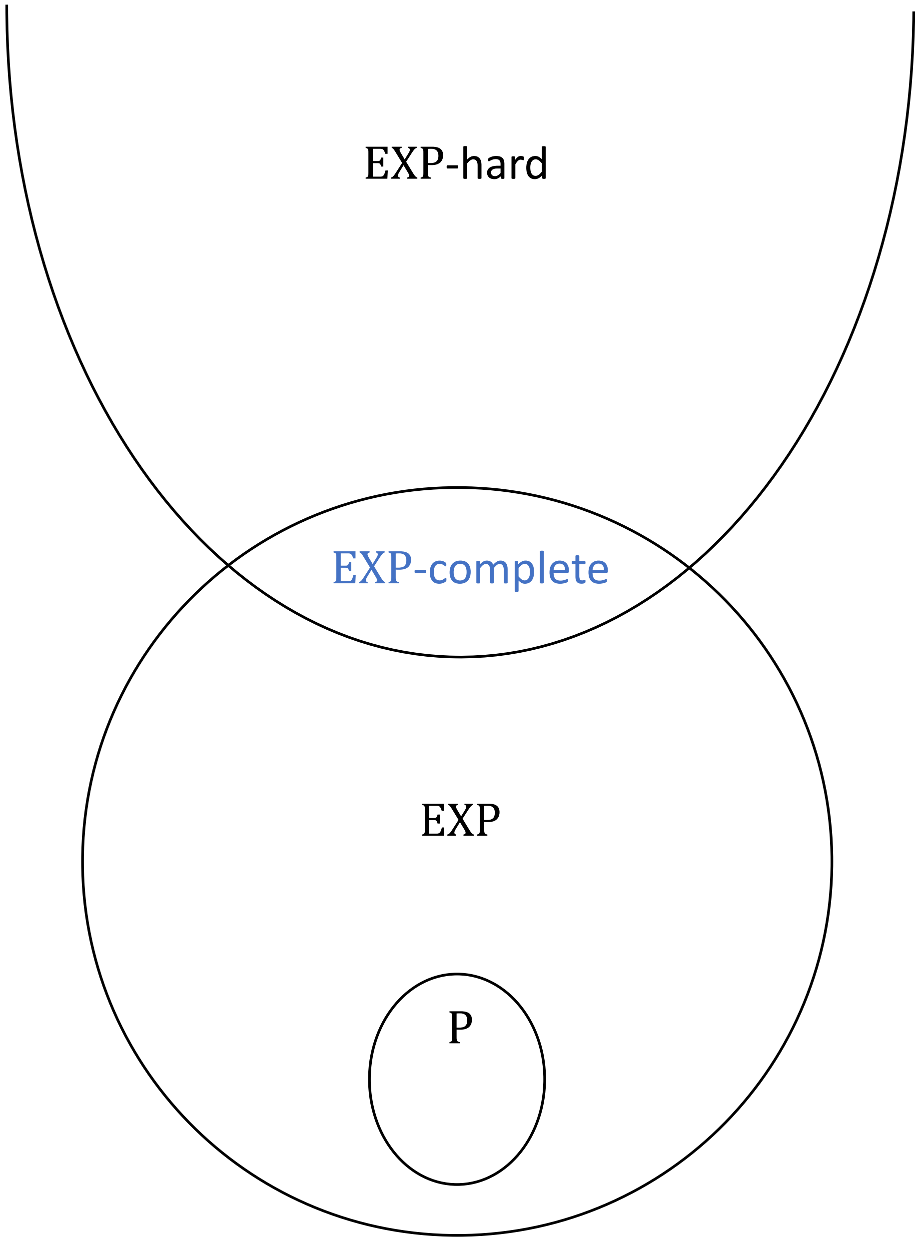
EXP-hardness

- **Definition:** Let L be a language. Suppose that for every $L' \in \mathrm{EXP}$, there is a poly-time mapping reduction from L' to L. In this case, we say that L is EXP-hard
- "L is EXP-hard" means "L is at least as hard as any language in EXP"

EXP-completeness

- **Definition:** Let L be a language. We say that L is EXP-complete if L is EXP-hard and $L \in \mathrm{EXP}$
- The EXP-complete languages are the hardest languages in EXP
- If L is EXP-complete, then the language L can be said to "capture" / "express" the entire complexity class EXP

EXP-completeness



EXP-complete languages are not in P

- **Claim:** If L is EXP-complete, then $L \notin \mathrm{P}$
- **Proof:** Since $\mathrm{P} \neq \mathrm{EXP}$, there exists $L_{\mathrm{HARD}} \in \mathrm{EXP} \setminus \mathrm{P}$
- Since L is EXP-hard, there is a poly-time mapping reduction from L_{HARD} to L
- Since $L_{\mathrm{HARD}} \notin \mathrm{P}$, this implies $L \notin \mathrm{P}$

$\mathrm{BOUNDED\text{-}HALT}$ is EXP-complete

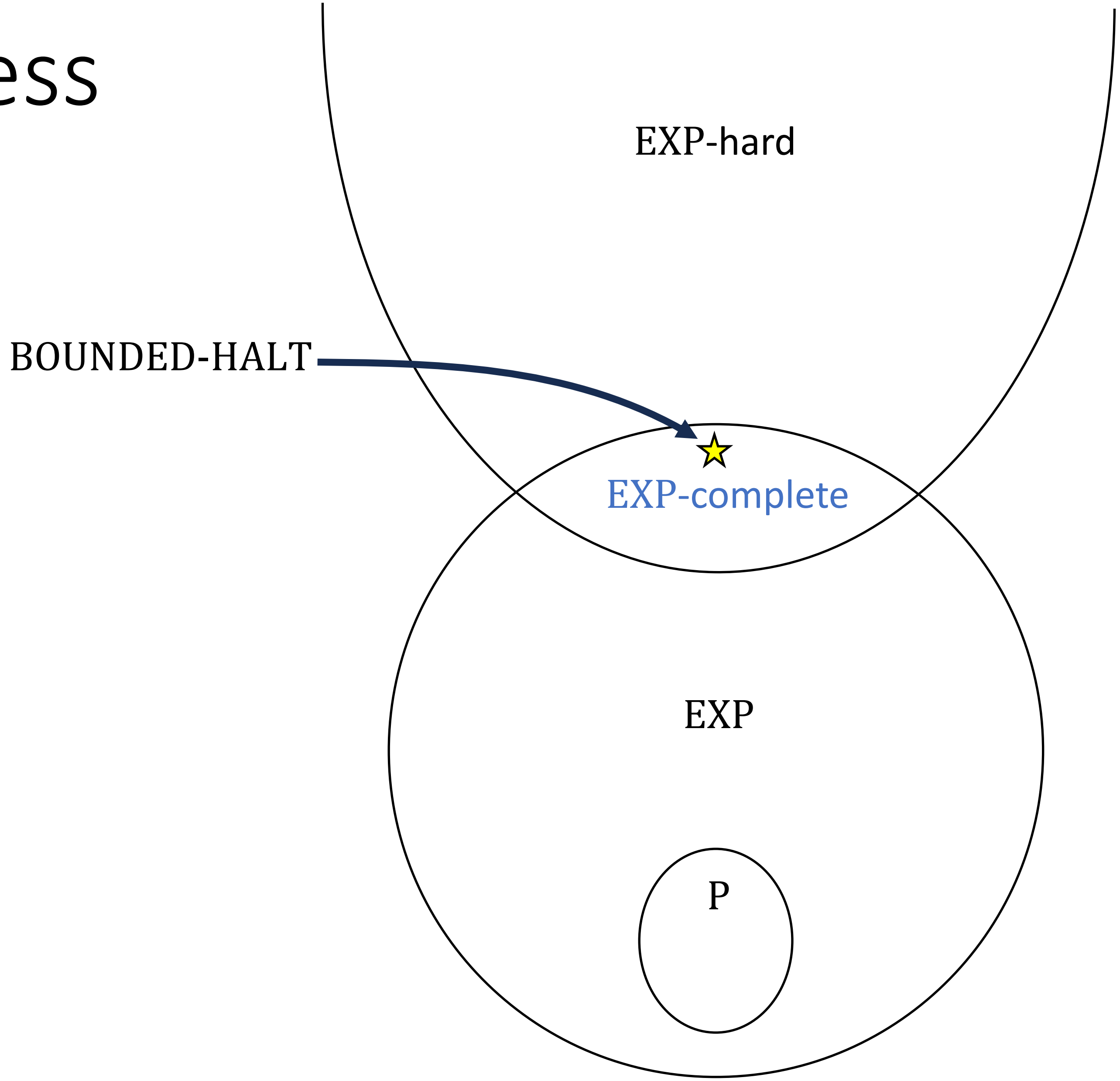
- Let $\mathrm{BOUNDED\text{-}HALT} = \{\langle M, w, T\rangle : M \text{ halts on } w \text{ within } T \text{ steps}\}$
- **Claim:** $\mathrm{BOUNDED\text{-}HALT}$ is EXP-complete
- **Proof:** First, let's show that $\mathrm{BOUNDED\text{-}HALT} \in \mathrm{EXP}$
- Algorithm: Given $\langle M, w, T\rangle$, we simulate M on w for T steps
- Exercise: This algorithm has time complexity

$$O(|\langle M\rangle| \cdot T^2) = O(n \cdot (2^n)^2) = 2^{O(n)}$$

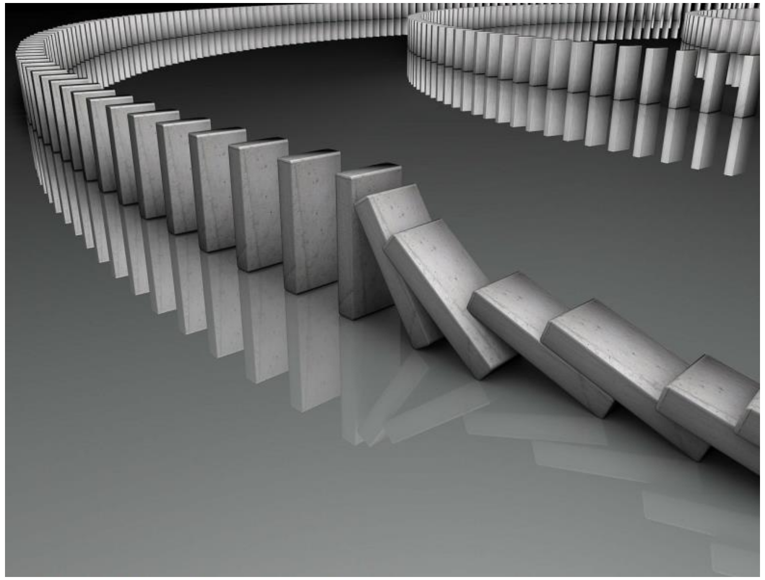
BOUNDED-HALT is EXP-complete

- Next, we need to show that BOUNDED-HALT is EXP-hard
- Fix any $L \in \text{EXP}$. Let M_L be a TM that decides L in time $C \cdot 2^{n^k}$
- Reduction from L to BOUNDED-HALT: $f(w) = \left\langle M_L', w, C \cdot 2^{n^k} \right\rangle$, where M_L' is a modified version of M_L in which q_{reject} has been replaced with looping
- Poly-time computable ✔ YES maps to YES ✔ NO maps to NO ✔

EXP-completeness



EXP-completeness



- EXP-completeness is a valuable technique for identifying languages outside P
- If L is EXP-complete, then $L \notin \mathrm{P}$
- There are many interesting EXP-complete languages

An EXP-complete problem that isn't about TMs

- Let $\mathrm{GENERALIZED\text{-}CHESS} = \{\langle P \rangle : P \text{ is an arrangement of chess pieces on an } N \times N \text{ board from which "white" can force a win}\}$
- (Exercise: Precisely define $\mathrm{GENERALIZED\text{-}CHESS}$)

Theorem: $\mathrm{GENERALIZED\text{-}CHESS}$ is EXP-complete.

Consequently, $\mathrm{GENERALIZED\text{-}CHESS} \notin \mathrm{P}$.

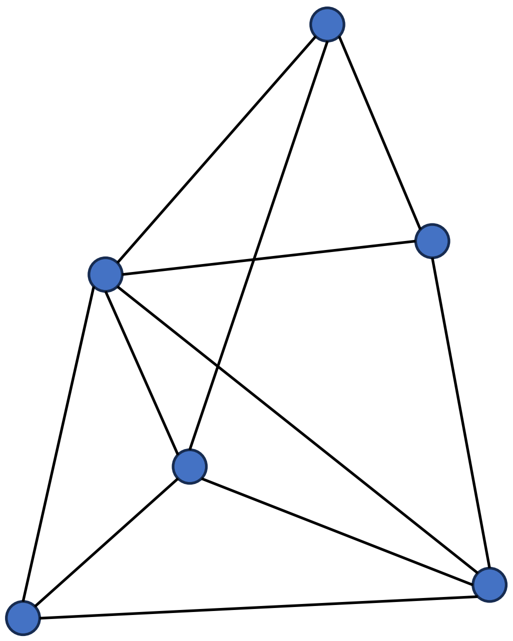
- (Proof omitted. This theorem will not be on psets/exams)

EXP-completeness

- EXP-completeness is a valuable tool for identifying intractability
- Is EXP-completeness the only tool we need for identifying intractability?

The clique problem

- A k-clique in a graph $G = (V, E)$ is a set $S \subseteq V$ such that $|S| = k$ and every two vertices in S are connected by an edge
- Example: This graph has a 4-clique



Which of the following statements is false?

A: Every vertex in a k-clique has degree at least $k - 1$

B: A single graph might have many k-cliques

C: If G has fewer than $\binom{k}{2}$ edges, then G does not have a k-clique

D: If every vertex has degree at least $k - 1$, then G has a k-clique

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

The clique problem

- Let $\text{CLIQUE} = \{\langle G, k \rangle : G \text{ has a } k\text{-clique}\}$
- Example: Let G be the graph with the following adjacency matrix
- Does G have a 4-clique?

	a	b	c	d	e	f	g
a	0	1	1	0	0	1	0
b	1	0	0	1	1	0	1
c	1	0	0	0	1	0	1
d	0	1	0	0	1	0	1
e	0	1	1	1	0	1	1
f	1	0	0	0	1	0	1
g	0	1	1	1	1	1	0

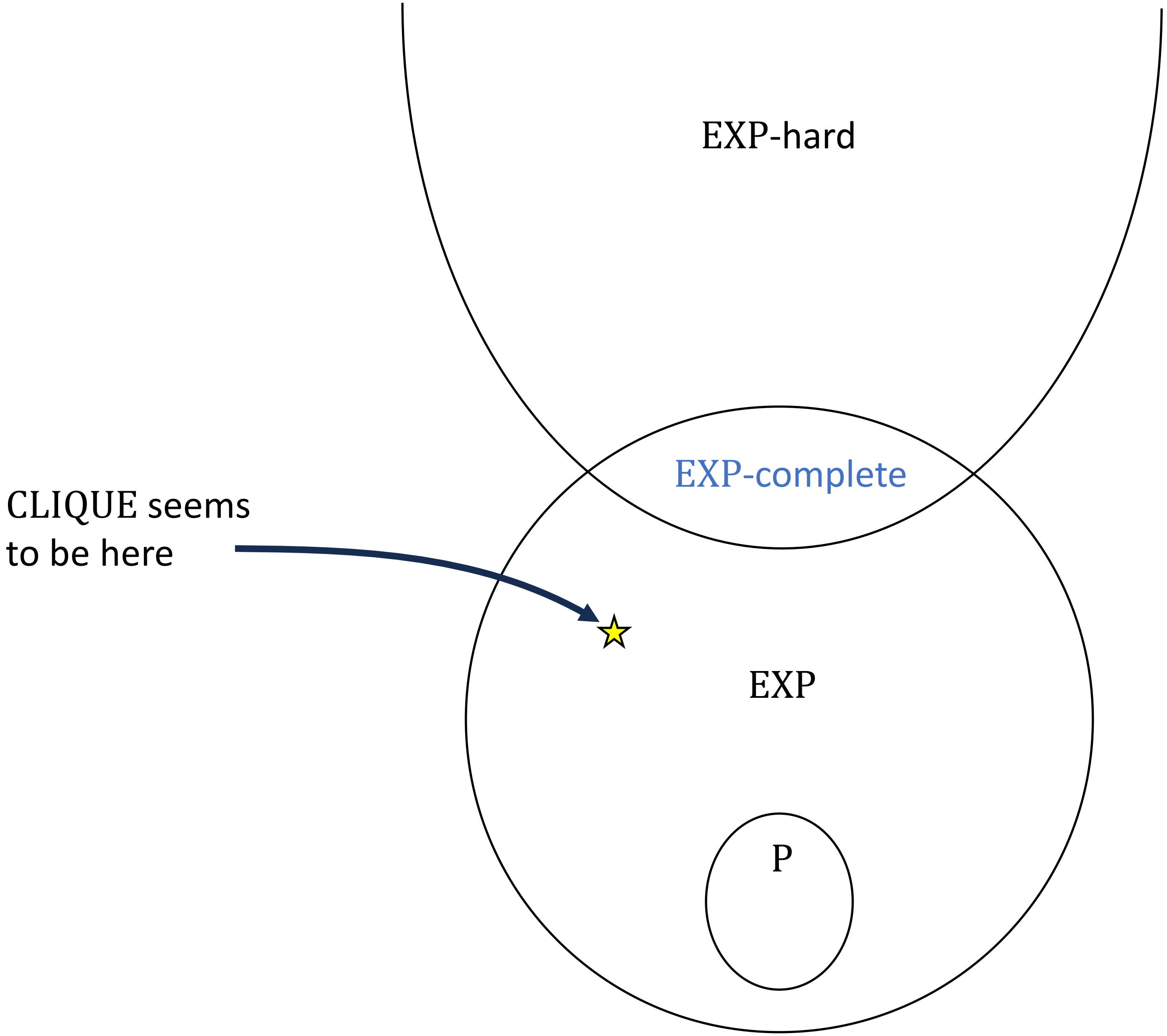
The clique problem

- Let $\text{CLIQUE} = \{\langle G, k \rangle : G \text{ has a } k\text{-clique}\}$
- Example: Let G be the graph with the following adjacency matrix
- Does G have a 4-clique?
- Yes! $S = \{\mathrm{b}, \mathrm{d}, \mathrm{e}, \mathrm{g}\}$

	a	b	c	d	e	f	g
a	0	1	1	0	0	1	0
b	1	0	0	1	1	0	1
c	1	0	0	0	1	0	1
d	0	1	0	0	1	0	1
e	0	1	1	1	0	1	1
f	1	0	0	0	1	0	1
g	0	1	1	1	1	1	0

Complexity of the clique problem

- Let $\text{CLIQUE} = \{\langle G, k\rangle : G \text{ has a } k\text{-clique}\}$
- $\text{CLIQUE} \in \text{EXP}$. (Why?)
- If you spend a while trying to design a good algorithm, eventually you might start to suspect that $\text{CLIQUE} \notin \text{P}$
- However, if you spend a while trying to design a good reduction, eventually you might start to suspect that CLIQUE is not EXP-complete either!


EXP-hard
EXP-complete
CLIQUE seems
to be here
EXP
P

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 $\langle G, k \rangle \notin \mathrm{CLIQUE}$, then $\Pr[M \text{ accepts } \langle G, k \rangle] = 0$.
  - If $\langle G, k \rangle \in \mathrm{CLIQUE}$, 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 NP

- Let $L \subseteq \Sigma^*$ be a language
- **Definition:** $L \in \mathrm{NP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \Sigma^*$:
  - If $w \in L$, then $\Pr[M \text{ accepts } w] \neq 0$
  - If $w \notin L$, then $\Pr[M \text{ accepts } w] = 0$
- "<u>N</u>ondeterministic <u>P</u>olynomial-time"

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

- Let $\text{FACTOR} = \{\langle N, K\rangle : N \text{ has a prime factor } p \leq K\}$
- **Claim:** $\text{FACTOR} \in \text{NP}$
- **Proof:**
  1. Pick $M \in \{2, 3, 4, \dots, K\}$ uniformly at random
  2. Check whether N is a multiple of M by 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

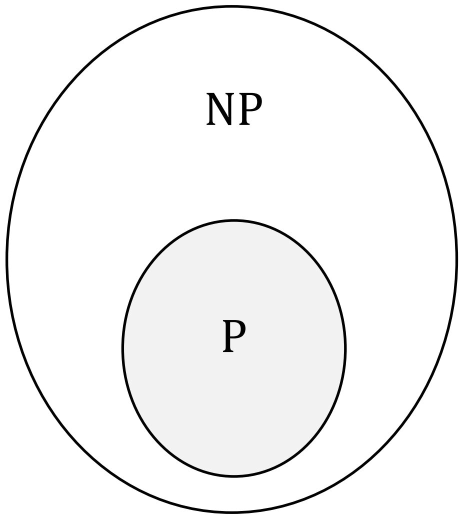
“Verificatio

Terminology:

- A poly-time TM that decides R is called a verifier for L
- If $\langle w, x \rangle \in R$, then x is called a certificate/witness for w

- Let $L \subseteq \Sigma^*$ be a language
- **Claim:** $L \in \mathrm{NP}$ if and only if there exists $k \in \mathbb{N}$ and $R \in \mathrm{P}$ such that
  - For every $w \in L$, there exists x such that $|x| \leq |w|^k$ and $\langle w, x \rangle \in R$
  - For every $w \notin L$, for every x, we have $\langle w, x \rangle \notin R$
- **Proof:** ($\Rightarrow$) Let $R = \{\langle w, x \rangle : M \text{ accepts } w \text{ when } x \text{ is on tape } 2\}$
- ($\Leftarrow$) Pick x at random. Accept if $\langle w, x \rangle \in R$ and reject otherwise

The P vs. NP problem



- $\mathrm{P} \subseteq \mathrm{NP}$ (why?)
- Does $\mathrm{P} = \mathrm{NP}$?
- "$\mathrm{P} = \mathrm{NP}$" would mean that finding a solution is never significantly harder than verifying someone else's solution
  - This would be counterintuitive!

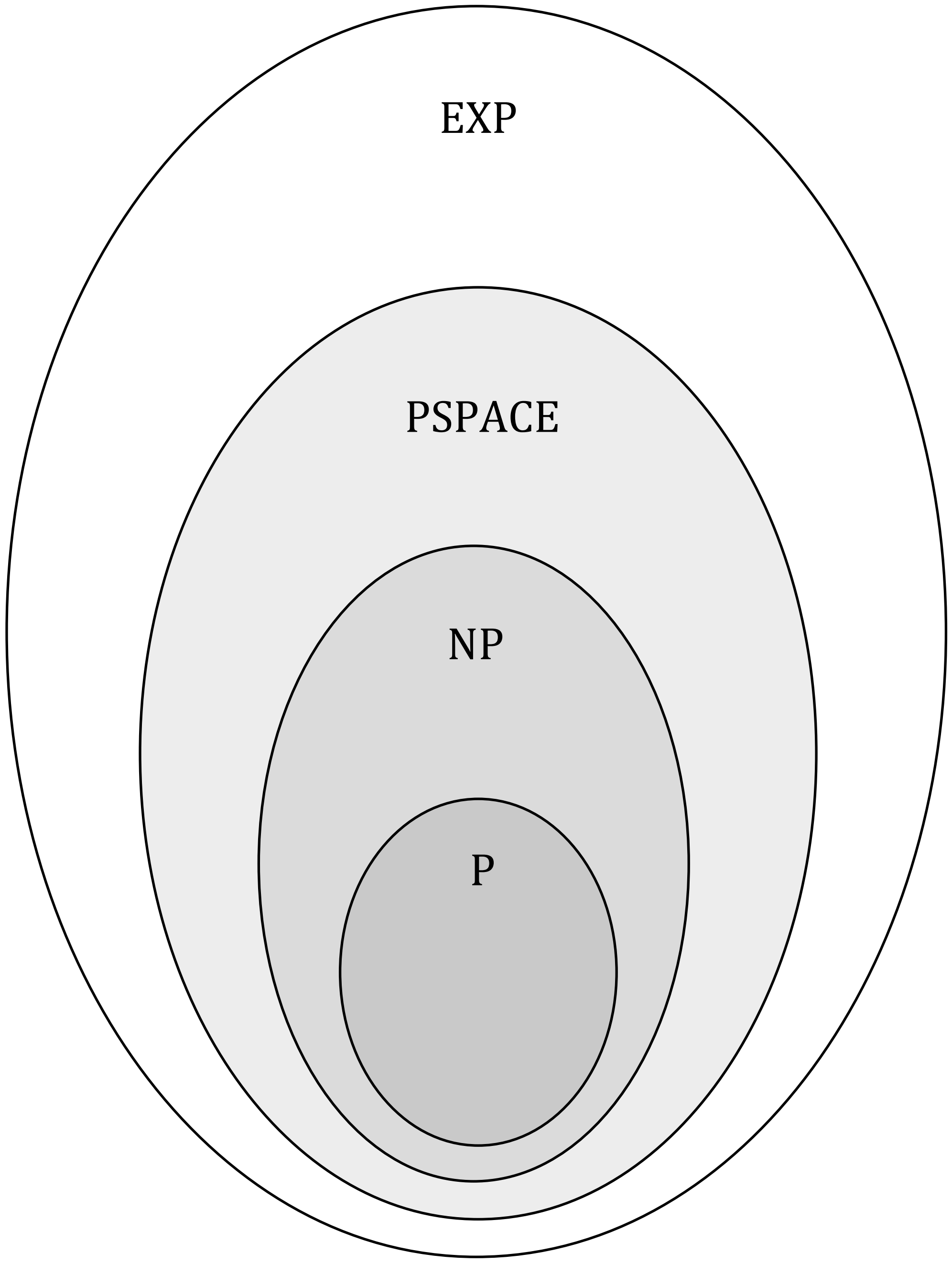
Conjecture: $\mathrm{P} \neq \mathrm{NP}$

The P vs. NP problem

- Nobody knows how to prove that $\mathrm{P} \neq \mathrm{NP}$
- The question of whether $\mathrm{P} = \mathrm{NP}$ 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 $\mathrm{P} = \mathrm{NP}$)

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

- **Claim:** $\mathrm{NP} \subseteq \mathrm{PSPACE}$
- **Proof:** Let M be a nondeterministic TM that runs in time n^k. Given $w \in \Sigma^n$:
  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