CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

BPP as a model of tractability

- Because of the amplification lemma, languages in BPP should be considered "tractable"
- A mistake that occurs with probability $1/3^{100}$ can be safely ignored

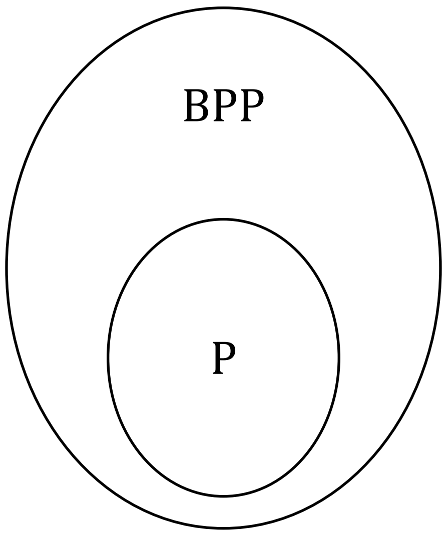
Extended Church-Turing Thesis

Extended Church-Turing Thesis:

For every $Y \subseteq \{0,1\}^*$, it is physically possible to build a device that decides Y in polynomial time if and only if $Y \in \mathrm{P}$.

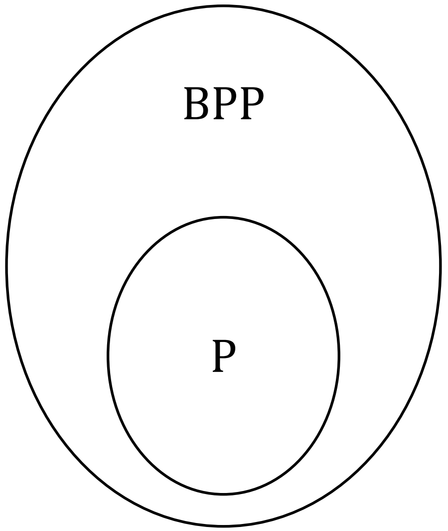
- Is PIT a counterexample?
- Not necessarily
- $\mathrm{PIT} \in \mathrm{BPP}$, but maybe $\mathrm{PIT} \in \mathrm{P}$ as well

P vs. BPP



- $\mathrm{P} \subseteq \mathrm{BPP}$
- **Open question:** Does $\mathrm{P} = \mathrm{BPP}$?
  - Is randomness helpful for computation?
- Profound question about the nature of efficient computation
- If $\mathrm{P} \neq \mathrm{BPP}$, then the extended Church-Turing thesis is false

P vs. BPP



- What would it take to prove $\mathrm{P} \neq \mathrm{BPP}$?
  - Define a language Y
  - Prove $Y \in \mathrm{BPP}$
  - Prove $Y \notin \mathrm{P}$
  - Good candidate: $Y = \mathrm{PIT}$
- What would it take to prove $\mathrm{P} = \mathrm{BPP}$?

Derandomization

- Suppose $Y \in \mathrm{BPP}$
- If we want to decide Y without randomness, what can we do?
- How can we convert a randomized algorithm into a deterministic algorithm?

Brute-force derandomization

- Let M be a randomized Turing machine that decides Y with error probability $1/3$ and time complexity n^k
- Deterministic algorithm that decides Y: Given $w \in \{0,1\}^n$:

1. For every $u \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and u on tape 2
   b) Keep a count of how many simulations accept
2. If more than half of the simulations accepted, then accept. Otherwise, reject

Brute-force derandomization: Correctness

1. For every $u \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and u on tape 2
   b) Keep a count of how many simulations accept
2. If more than half of the simulations accepted, then accept. Otherwise, reject

- If $w \in Y$, then at least
- If $w \notin Y$, then at most

What is the time complexity of the algorithm?

A: $2^{\mathrm{poly}(n)}$

B: $\mathrm{poly}(n)$

C: $2^{2^{\Theta(n)}}$

D: ∞

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

Brute-force derandomization: Time complexity

1. For every $u \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and u on tape 2
   b) Keep a count of how many simulations accept
2. If more than half of the simulations accepted, then accept. Otherwise, reject

- Time complexity: $2^{\mathrm{poly}(n)}$ 😟
- This algorithm does not show that $\mathrm{P} = \mathrm{BPP}$, but it does show that even randomized algorithms have limitations. For example, $\mathrm{HALT} \notin \mathrm{BPP}$

The complexity class EXP

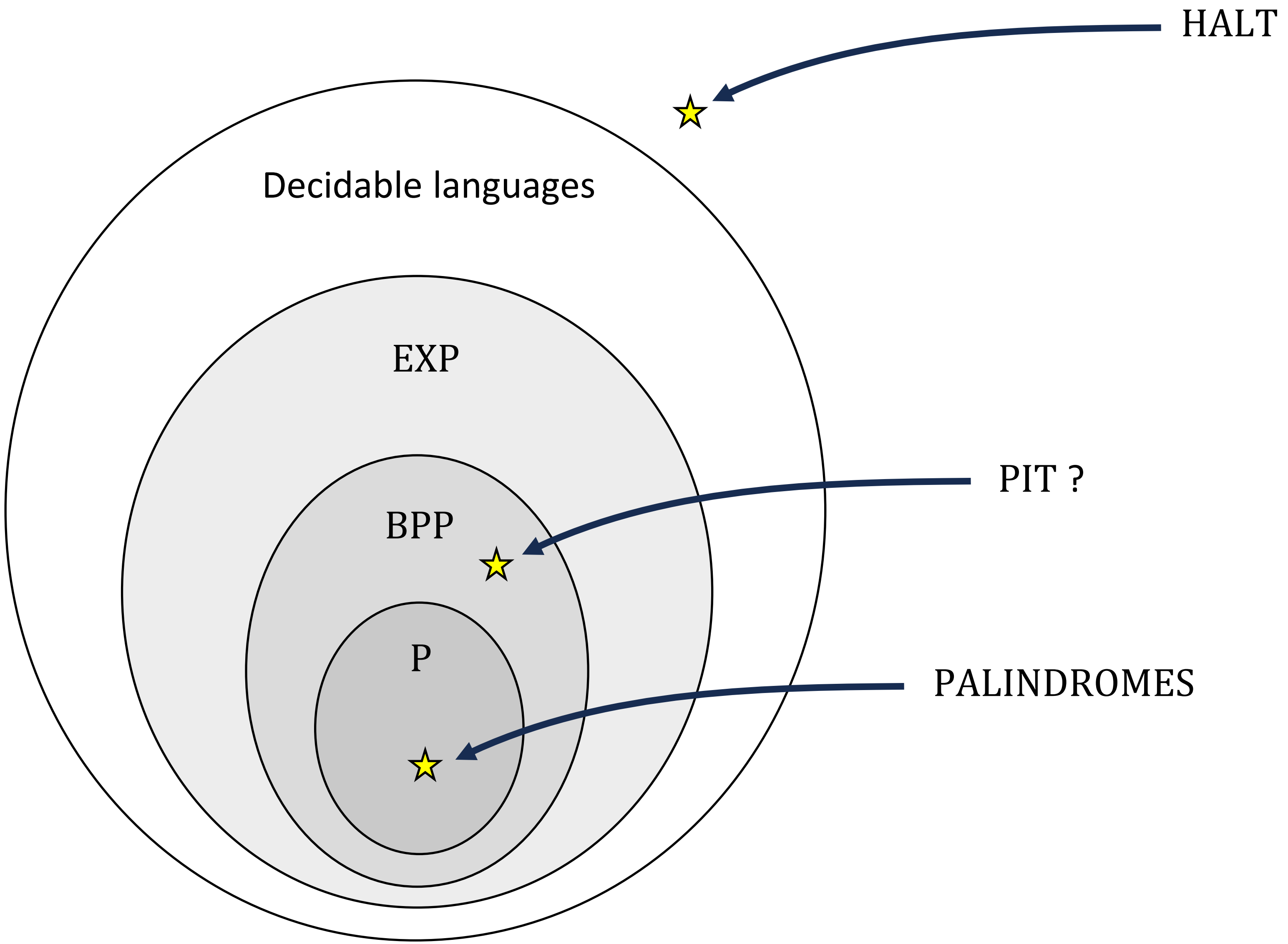
- **Definition:**

$$\mathrm{EXP} = \left\{ Y \subseteq \{0,1\}^* : Y \text{ can be decided in time } 2^{\mathrm{poly}(n)} \right\}$$

$$= \bigcup_{k=1}^{\infty} \mathrm{TIME}\left(2^{n^k}\right)$$

- Brute-force derandomization proves $\mathrm{BPP} \subseteq \mathrm{EXP}$

P ⊆ BPP ⊆ EXP



Brute-force derandomization: Space complexity

1. For every $u \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and u on tape 2
   b) Keep a count of how many simulations accept
2. If more than half of the simulations accepted, then accept. Otherwise, reject

What is the space complexity of the algorithm?

A: $2^{\Theta(n^k)}$

B: $\mathrm{poly}(n)$

C: $2^{2^{\Theta(n)}}$

D: ∞

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

The complexity class $\mathbf{PSPACE}$

- **Definition:**

$$\mathrm{PSPACE} = \{Y \subseteq \{0,1\}^* : Y \text{ can be decided in space poly}(n)\}$$

- Brute-force derandomization proves that $\mathrm{BPP} \subseteq \mathrm{PSPACE}$

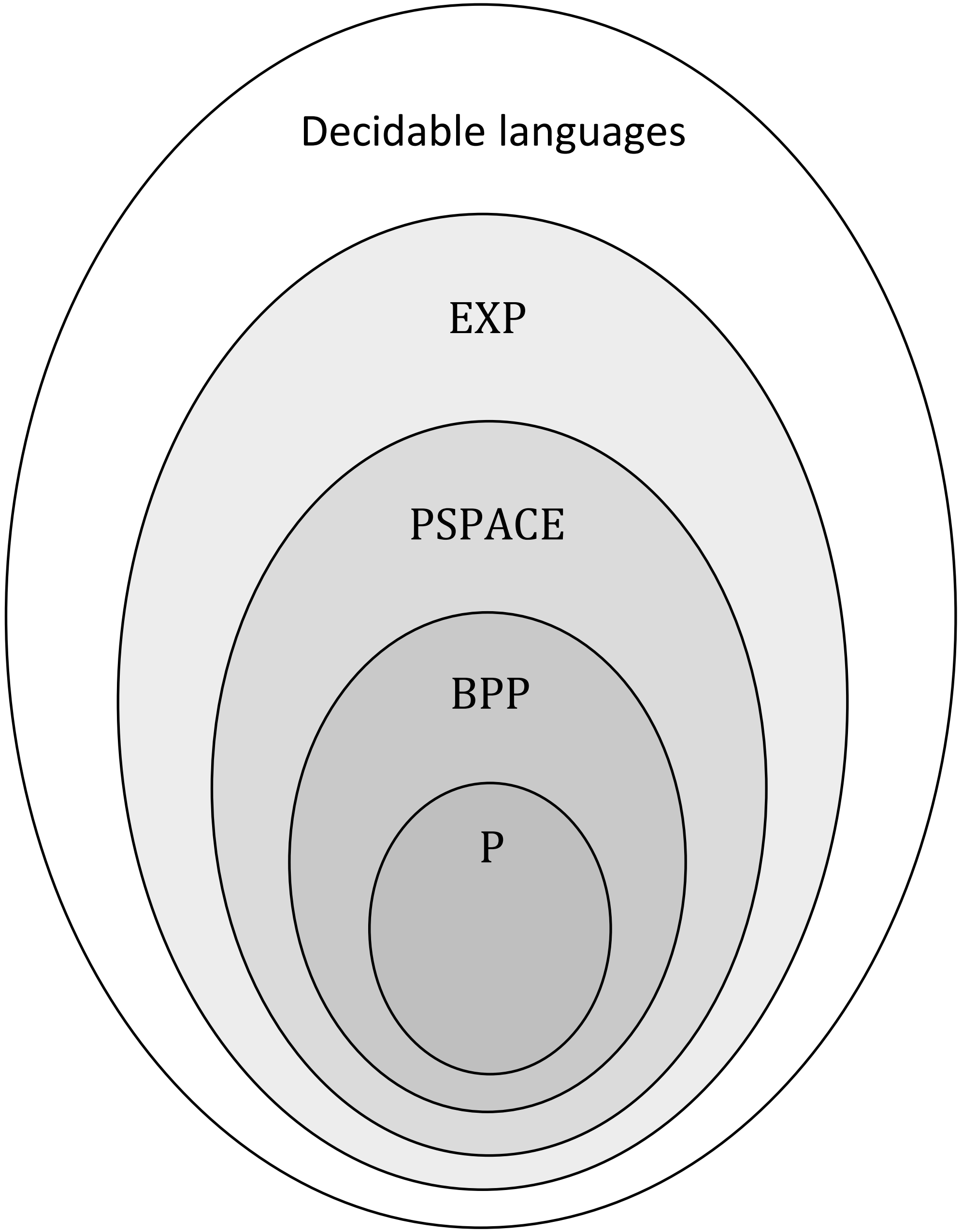
PSPACE vs. EXP

- **Theorem 1:** $\mathrm{BPP} \subseteq \mathrm{EXP}$
- **Theorem 2:** $\mathrm{BPP} \subseteq \mathrm{PSPACE}$
- Which theorem is stronger?
- How does PSPACE compare to EXP?

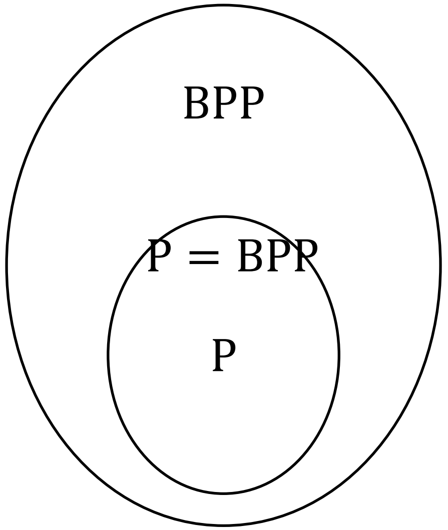
Theorem: $\text{PSPACE} \subseteq \text{EXP}$

- **Proof (1 slide):** Let M be a Turing machine that decides a language Y
- Exercise 5: For each input, $\text{Time} \leq C^{\text{Space}+1}$, where C depends only on M
- When $\text{Space} = \text{poly}(n)$, we get

$$\text{Time} \leq C^{\text{poly}(n)} = \left(2^{\log C}\right)^{\text{poly}(n)} = 2^{(\log C)\cdot\text{poly}(n)} = 2^{\text{poly}(n)}$$


Decidable languages
EXP
PSPACE
BPP
P

Beyond brute-force derandomization



- There are other derandomization methods that are more sophisticated
  - We will see an example later in the course
- Because of these other methods, most experts conjecture $\mathrm{P} = \mathrm{BPP}$!

BPP and the Extended Church-Turing Thesis

Extended Church-Turing Thesis:

For every $Y \subseteq \{0, 1\}^*$, it is physically possible to build a device that decides Y in polynomial time if and only if $Y \in \mathrm{P}$.

- If experts are correct that $\mathrm{P} = \mathrm{BPP}$, then the Extended Church-Turing Thesis survives the challenge posed by randomization

BPP and the Extended Church-Turing Thesis

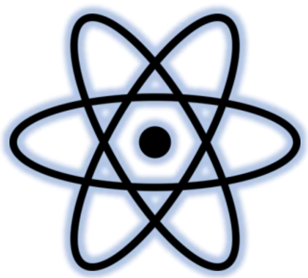
- Just in case, the thesis is sometimes revised to allow randomization:

Extended Church-Turing Thesis, version 2:

For every $Y \subseteq \{0,1\}^*$, it is physically possible to build a device that decides Y in polynomial time if and only if $Y \in \mathrm{BPP}$.

- This version is immune to the challenge posed by randomization
- However, there is a bigger threat: Quantum Computation

Quantum computing



- Properly studying quantum computing is beyond the scope of this course
- We will briefly circle back to it later
- For now, let's focus on P
- P is probably not the ultimate model of efficient computation…
- but it is still a valuable model

Which problems can be solved through computation?

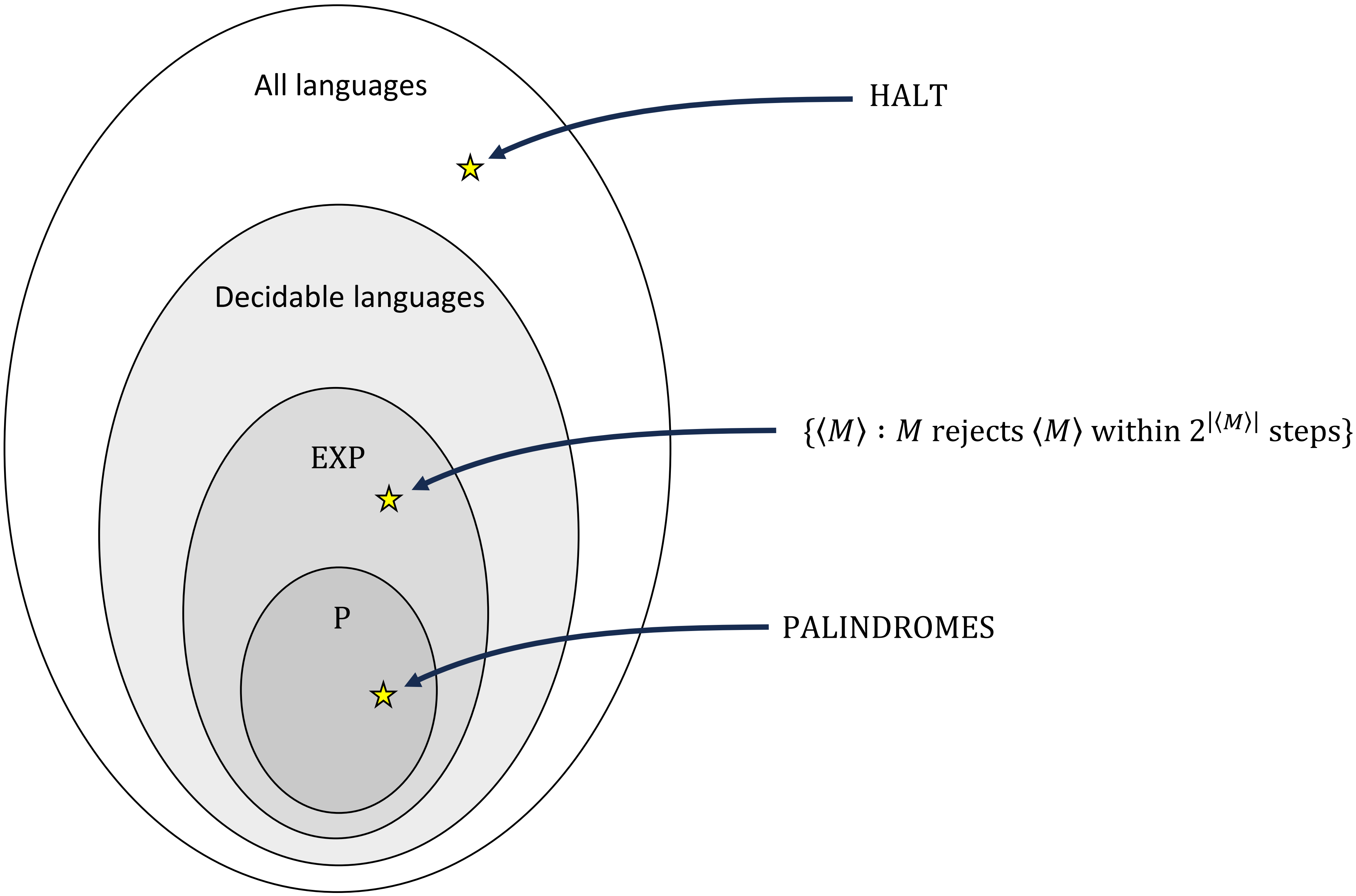
CLASSICAL

Which languages are in $\mathbf{P}$?

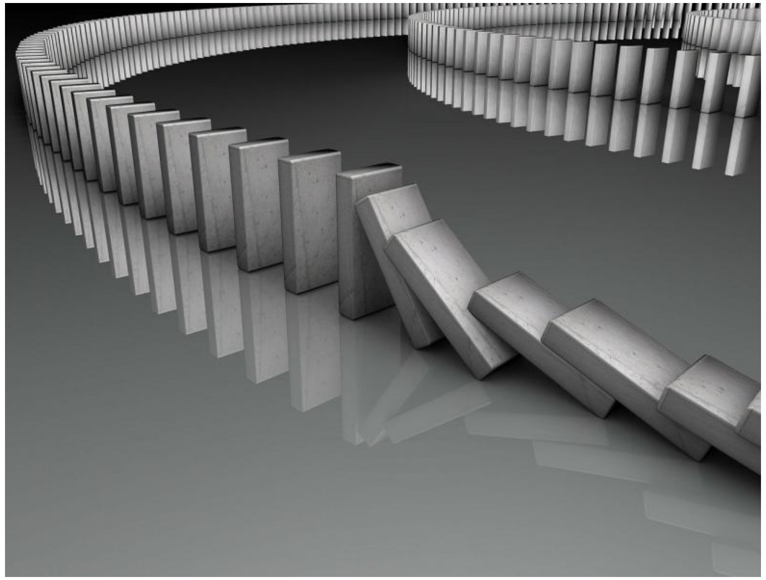
Which languages are not in $\mathbf{P}$?

P vs. EXP

- **Time Hierarchy Theorem:** For every time-constructible $T: \mathbb{N} \to \mathbb{N}$, there exists a language $Y \in \mathrm{TIME}(T^4)$ such that $Y \notin \mathrm{TIME}\big(o(T)\big)$
- **Corollary:** $\mathrm{P} \neq \mathrm{EXP}$
  - **Proof:** $\mathrm{P} = \bigcup_{k=1}^{\infty} \mathrm{TIME}\big(n^k\big) \subseteq \mathrm{TIME}\big(o(2^n)\big) \subsetneq \mathrm{TIME}(2^{4n}) \subseteq \mathrm{EXP}$
  - Interpretation: There are some exponential-time algorithms that cannot be converted into polynomial-time algorithms


All languages
HALT
Decidable languages
EXP
$\{\langle M \rangle : M \text{ rejects } \langle M \rangle \text{ within } 2^{|\langle M \rangle|} \text{ steps}\}$
P
PALINDROMES

Contrived vs. natural



- The language

$$\{\langle M \rangle : M \text{ rejects } \langle M \rangle \text{ within } 2^{|\langle M \rangle|} \text{ steps}\}$$

is rather contrived

- Are there languages in $\mathrm{EXP} \setminus \mathrm{P}$ that are interesting / natural / well-motivated?

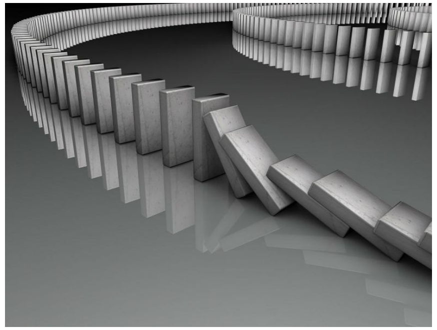
The bounded halting problem

- Let $\text{BOUNDED-HALT} = \{\langle M, w, T\rangle : M \text{ halts on } w \text{ within } T \text{ steps}\}$
- Exercise: Can decide in time $O(|\langle M\rangle|^2 \cdot |w|^2 \cdot T^2)$

Polynomial time?

- ⚠️ The input size is $n = |\langle M, w, T\rangle| \approx |\langle M\rangle| + |\langle w\rangle| + \log T$
- $\text{BOUNDED-HALT} \in \text{TIME}(n^4 \cdot 2^{2n}) \subseteq \text{EXP}$

The bounded halting problem



- $\text{BOUNDED-HALT} = \{\langle M, w, T \rangle : M \text{ halts on } w \text{ within } T \text{ steps}\}$

Theorem: $\text{BOUNDED-HALT} \notin \text{P}$

- Proof strategy: We'll show that if BOUNDED-HALT were in P, then it would follow that $\text{P} = \text{EXP}$

Proof that $\mathrm{BOUNDED\text{-}HALT} \notin \mathrm{P}$

- Assume B is a poly-time TM deciding $\mathrm{BOUNDED\text{-}HALT}$
- Let $Y \in \mathrm{EXP}$. There is a TM M that $\begin{cases} \text{accepts } w \text{ within } 2^{|w|^k} \text{ steps} & \text{if } w \in Y \\ \text{loops} & \text{if } w \notin Y \end{cases}$
- We will construct a poly-time TM R that decides Y

R:

> Given $w \in \{0,1\}^*$:
>
> 1. Simulate B on $\left\langle M, w, 2^{|w|^k} \right\rangle$
> 2. If B accepts, accept. If B rejects, reject.

- Polynomial time ✓
- If $w \in Y$, then M accepts w within $2^{|w|^k}$ steps, so R accepts w ✓
- If $w \notin Y$, then M loops on w, so R rejects w ✓