CMSC 28100

Introduction to Complexity Theory

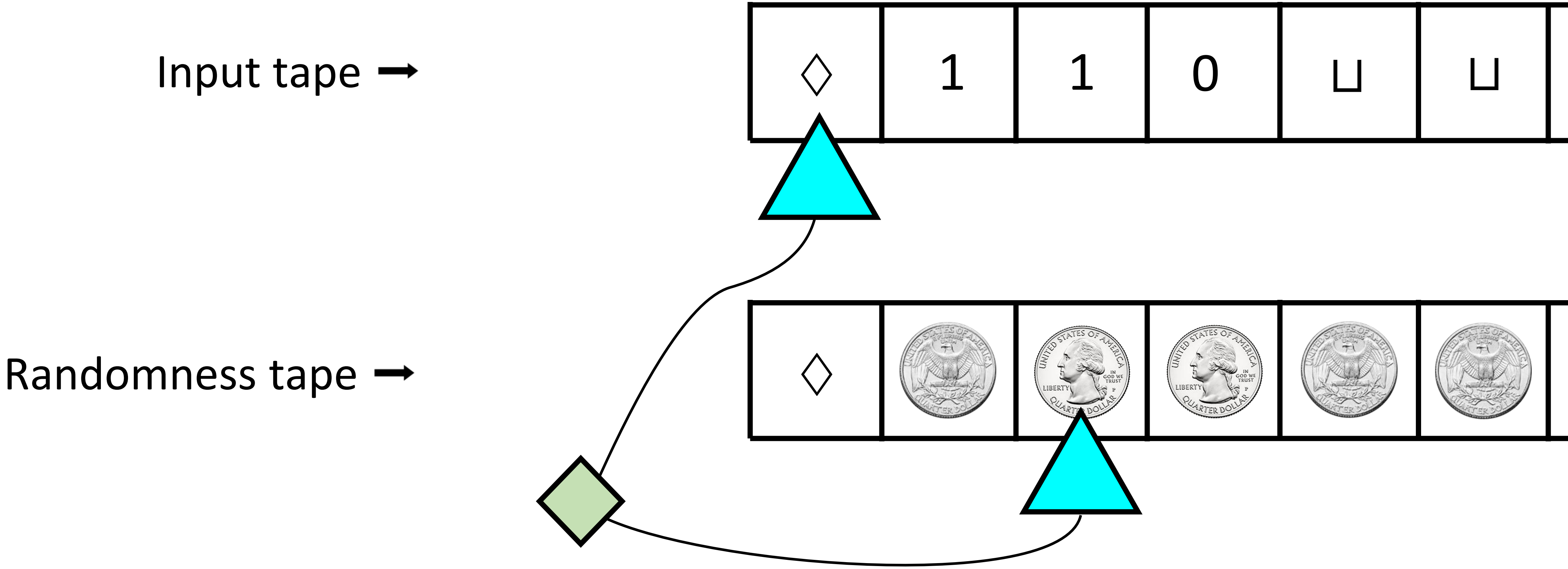
Spring 2024
Instructor: William Hoza

Which problems

can be solved

through computation?

Randomized Turing machines



The complexity class BPP

- Let $L \subseteq \Sigma^*$ be a language
- **Definition:** $L \in \mathrm{BPP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \Sigma^*$:
  - If $w \in L$, then $\Pr[M \text{ accepts } w] \geq 2/3$
  - If $w \notin L$, then $\Pr[M \text{ accepts } w] \leq 1/3$
- "<u>B</u>ounded-error <u>P</u>robabilistic <u>P</u>olynomial-time"

Amplification lemma

- Let $L \in \mathrm{BPP}$, and let $k \in \mathbb{N}$ be any constant

Amplification Lemma: There exists a randomized polynomial-time Turing machine M such that for every $n \in \mathbb{N}$ and every $w \in \Sigma^n$:

- If $w \in L$, then $\Pr[M \text{ accepts } w] \geq 1 - 1/2^{n^k}$
- If $w \notin L$, then $\Pr[M \text{ accepts } w] \leq 1/2^{n^k}$

- As $n \to \infty$, the error probability goes to 0 extremely rapidly!

Proof of the amplification lemma

- For simplicity, we will only prove the amplification lemma in a special case
- We will assume that there is a randomized poly-time Turing machine M_0 such that for every $w \in \Sigma^*$:
  - If $w \in L$, then $\Pr[M_0 \text{ accepts } w] \geq 2/3$
  - If $w \notin L$, then $\Pr[M_0 \text{ accepts } w] = 0$ ⬅️ No false positives!
- See the textbook for a proof of the general case

Proof of the amp

If M_0 uses $R(n)$ many random bits, then how many random bits does M use?

A: $R(n) + n^k$

B: $R(n) \cdot n^k$

C: $R(n)^k$

D: Not enough information

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

- Low-error algorithm M: C

  1) For $i = 1$ to n^k:
     a) Simulate M_0 on w using fresh random bits.
     b) If M_0 accepts, accept.
  2) Reject.

Polynomial time

- If $w \notin L$, then $\Pr[M \text{ accepts } w] = 0$. Still no false positives

- If $w \in L$, then $\Pr[M \text{ rejects } w] \leq (1/3)^{n^k} = 1/3^{n^k} < 1/2^{n^k}$

BPP as a model of tractability

- Because of the amplification lemma, languages in BPP should be considered "tractable"
- A mistake that occurs with probability $1/2^{100}$ can be safely ignored
- (Even if you use a deterministic algorithm, can you really be 100% certain that the computation was carried out correctly?)

Extended Church-Turing Thesis

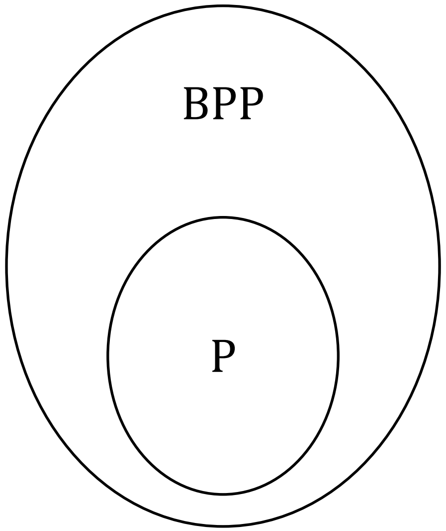
- Let L be a language

Extended Church-Turing Thesis:

It is physically possible to build a device that decides L in polynomial time if and only if $L \in \mathrm{P}$.

- Does the BPP model disprove the extended Church-Turing thesis?

P vs. BPP



- $\mathrm{P} \subseteq \mathrm{BPP}$
- Does $\mathrm{P} = \mathrm{BPP}$?
  - Is randomness helpful for computation?
- If $\mathrm{P} \neq \mathrm{BPP}$, then the extended Church-Turing thesis is false
- This is a profound question about the nature of efficient computation
- It's an open question! Nobody knows how to prove $\mathrm{P} = \mathrm{BPP}$ or $\mathrm{P} \neq \mathrm{BPP}$

P vs. BPP

- In communication complexity, randomness is powerful
- There are some languages in BPP that are not known to be in P
- These considerations might suggest $\mathrm{P} \neq \mathrm{BPP}$
- Surprisingly, there is a significant body of evidence favoring the opposite!

Conjecture: $\mathrm{P} = \mathrm{BPP}$

Extended Church-Turing Thesis

- Let L be a language

Extended Church-Turing Thesis:

It is physically possible to build a device that decides L in polynomial time if and only if $L \in \mathrm{P}$.

- Assuming $\mathrm{P} = \mathrm{BPP}$, the extended Church-Turing thesis survives the challenge posed by randomized computation!

Derandomization

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

Brute-force derandomization

- Let M be the randomized polynomial-time Turing machine guaranteed by the assumption $L \in \mathrm{BPP}$. Say M runs in time n^k
- Deterministic algorithm that decides L: Given $w \in \Sigma^n$:

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

What is the time complexity of the algorithm?

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

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

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, $\mathrm{HALT} \notin \mathrm{BPP}$

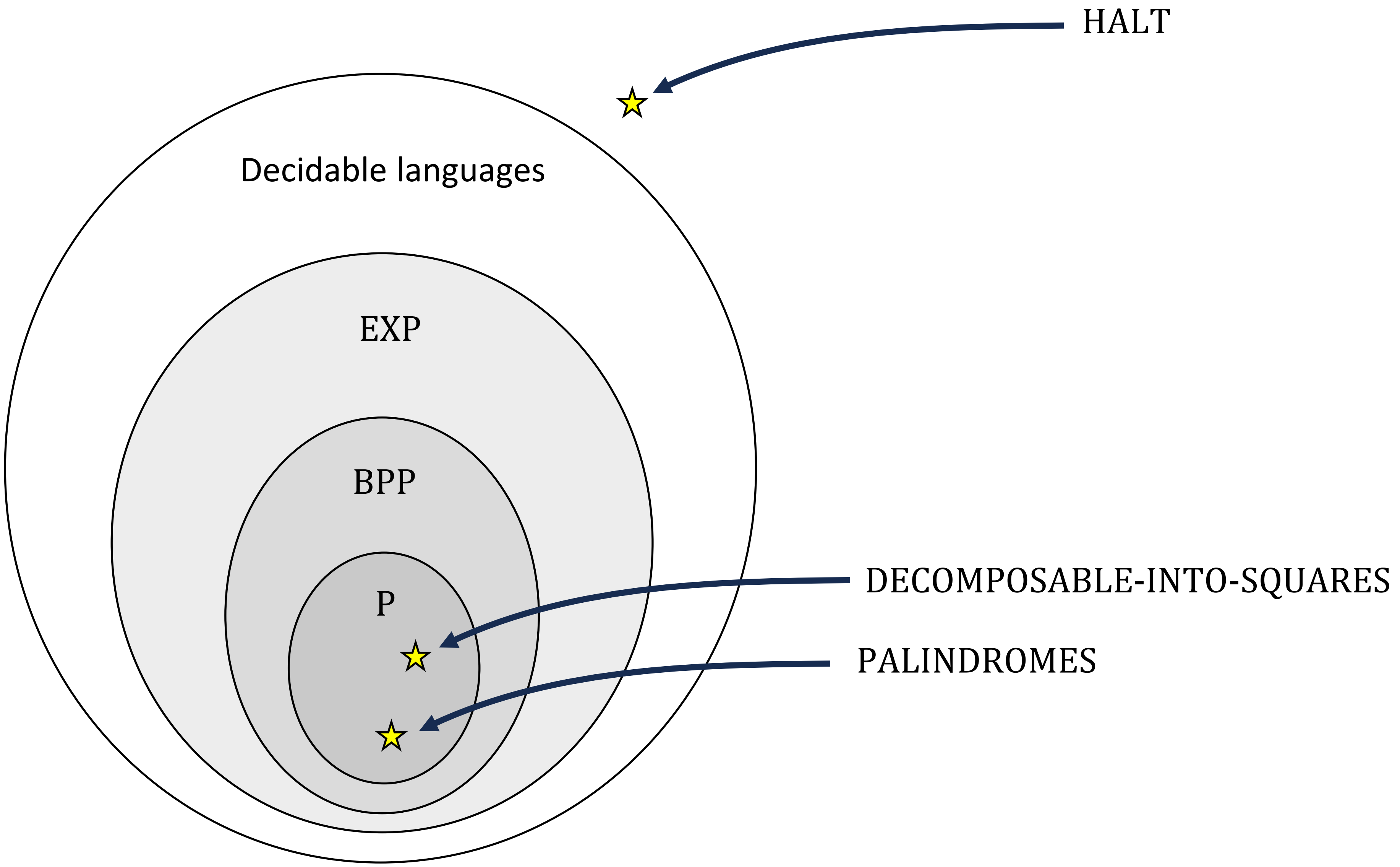
The complexity class EXP

- **Definition:** EXP is the class of languages that can be decided in time $2^{\mathrm{poly}(n)}$:

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

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

$P \subseteq BPP \subseteq EXP$



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 PSPACE

- Let L be a language
- **Definition:** $L \in \mathrm{PSPACE}$ if there exists a Turing machine M that decides L with space complexity $O(n^k)$ for some constant $k \in \mathbb{N}$
- Brute-force derandomization proves that $\mathrm{BPP} \subseteq \mathrm{PSPACE}$

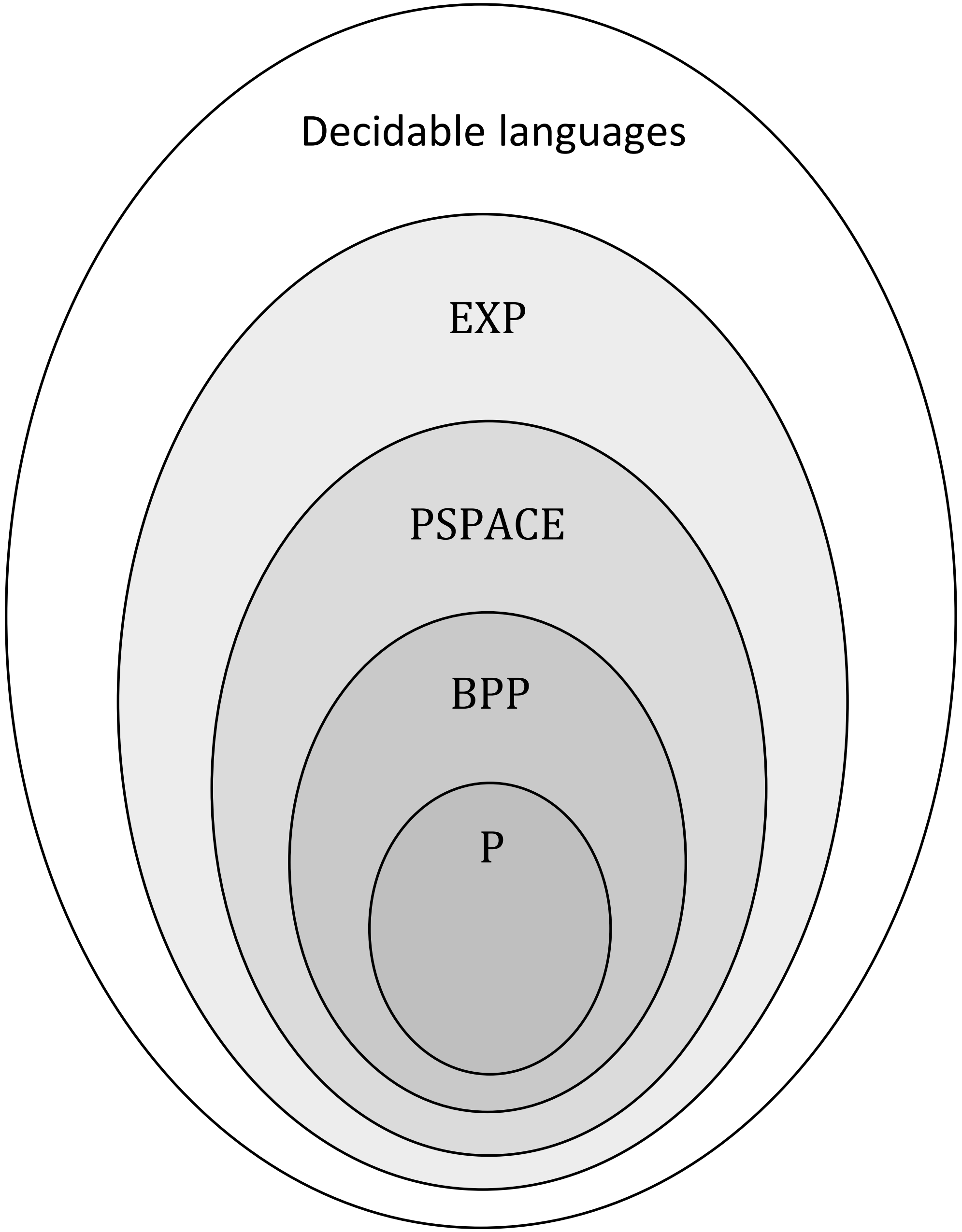
PSPACE vs. EXP

- We have proven two upper bounds on the power of BPP:
  - $\mathrm{BPP} \subseteq \mathrm{EXP}$
  - $\mathrm{BPP} \subseteq \mathrm{PSPACE}$
- Which theorem is stronger?
- How does PSPACE compare to EXP?

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

- **Proof:** Let M be a Turing machine that decides a language L
- Let T, S be the amounts of time/space that M uses on some input w
- Problem set 2: $T \leq C^{S+1}$, where C depends only on M
- When $S = \mathrm{poly}(n)$, we get

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


Decidable languages
EXP
PSPACE
BPP
P