CMSC 28100

Introduction to Complexity Theory

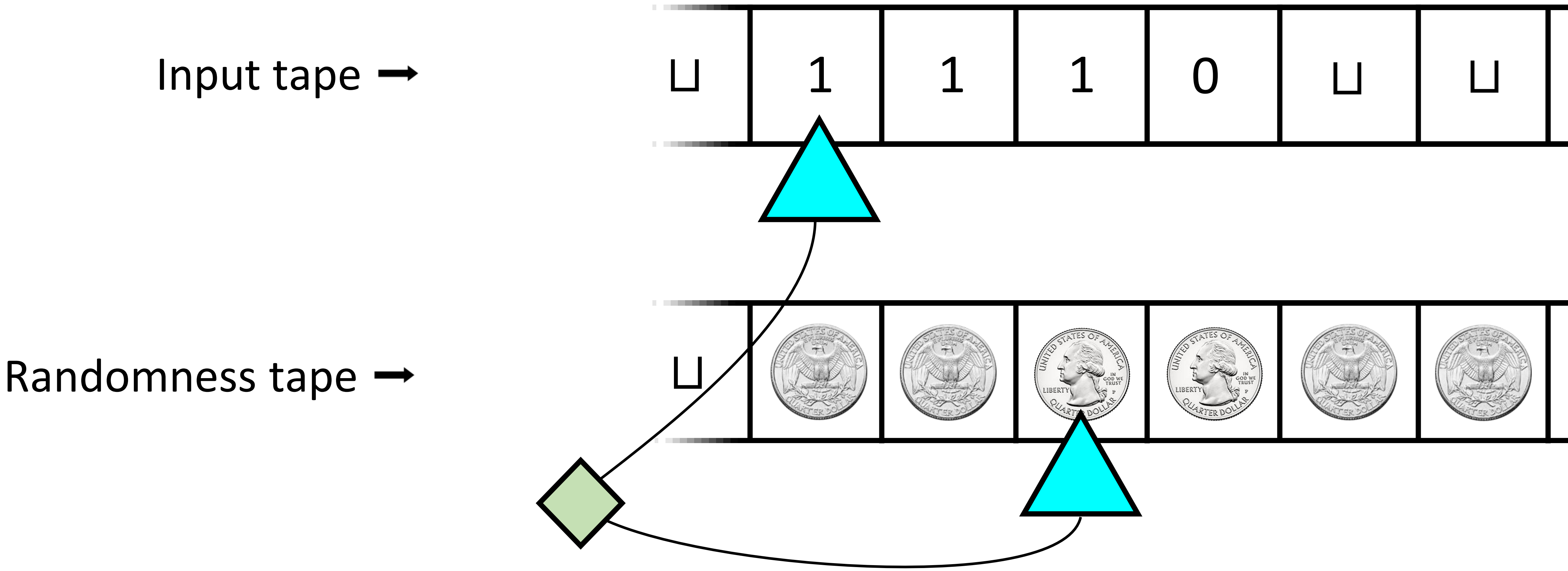
Autumn 2025
Instructor: William Hoza

Which problems

can be solved

through computation?

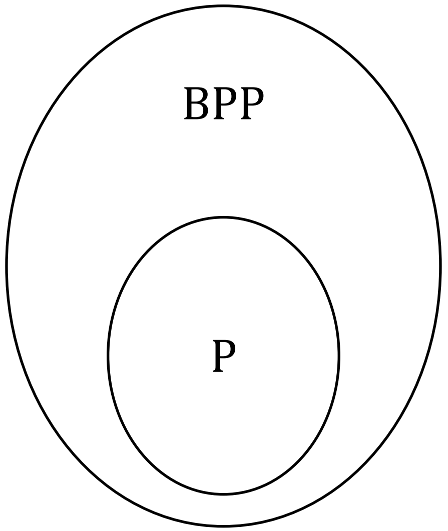
Randomized Turing machines



Identity testing: Recap

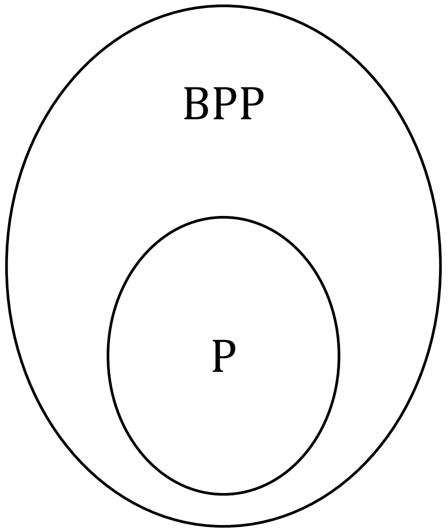
- We proved $\mathrm{IDENTICALLY\text{-}ZERO} \in \mathrm{BPP}$
- Therefore, we should consider $\mathrm{IDENTICALLY\text{-}ZERO}$ to be tractable
- Is this a counterexample to the idea that P is the set of tractable problems?
- Not necessarily. Maybe $\mathrm{IDENTICALLY\text{-}ZERO} \in \mathrm{P}$

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

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: $Y = \text{IDENTICALLY-ZERO}$
- 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 $x \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and x 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 $x \in \{0,1\}^{n^k}$:
    a) Simulate M, initialized with w on tape 1 and x 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: $2^{\text{poly}(n)}$

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 $x \in \{0,1\}^{n^k}$:
    a) Simulate M, initialized with w on tape 1 and x 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

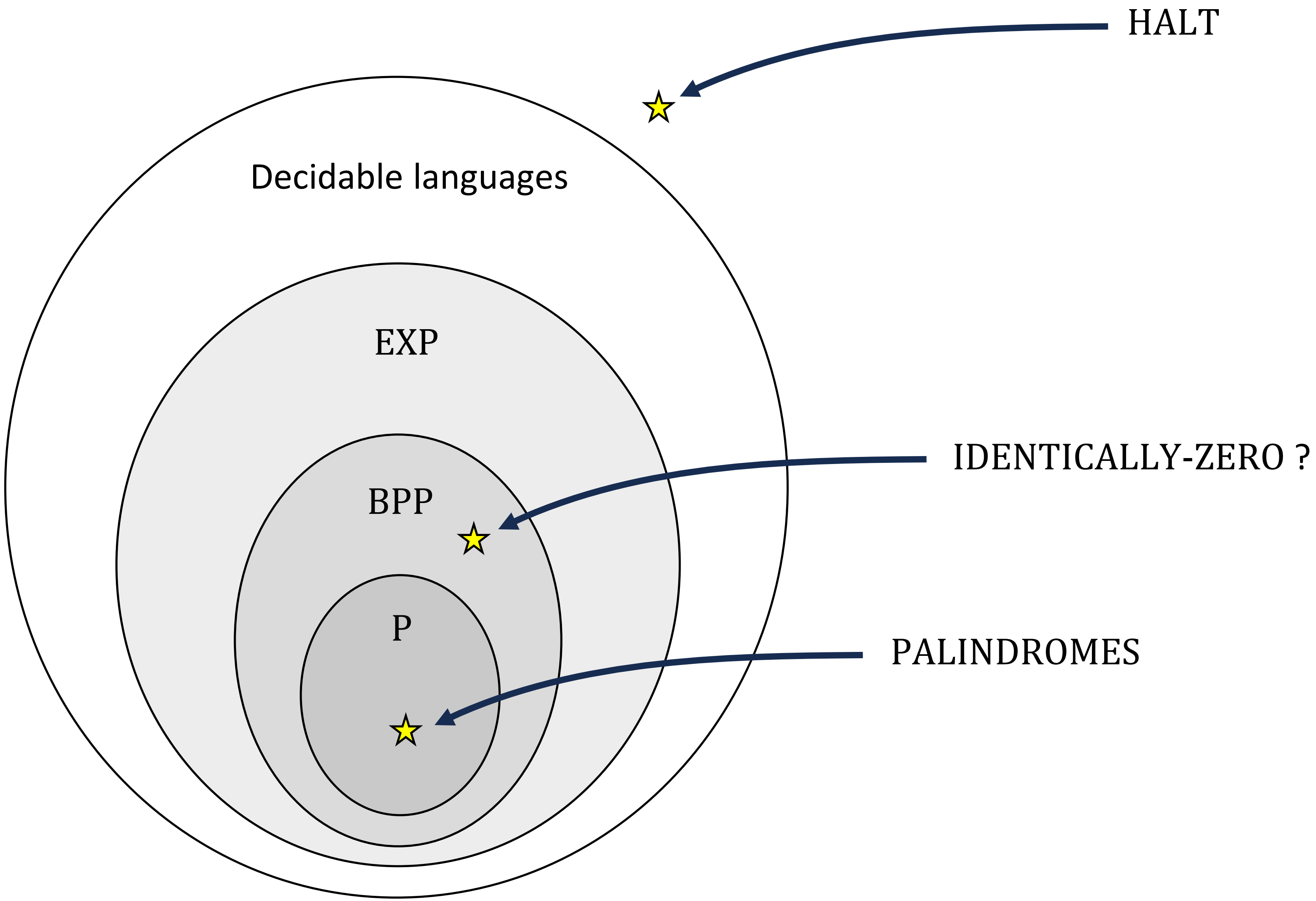
- **Definition:**

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

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

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

$P \subseteq BPP \subseteq EXP$



Brute-force derandomization: Space complexity

1. For every $x \in \{0,1\}^{n^k}$:
   a) Simulate M, initialized with w on tape 1 and x 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

- **Definition:**

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

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

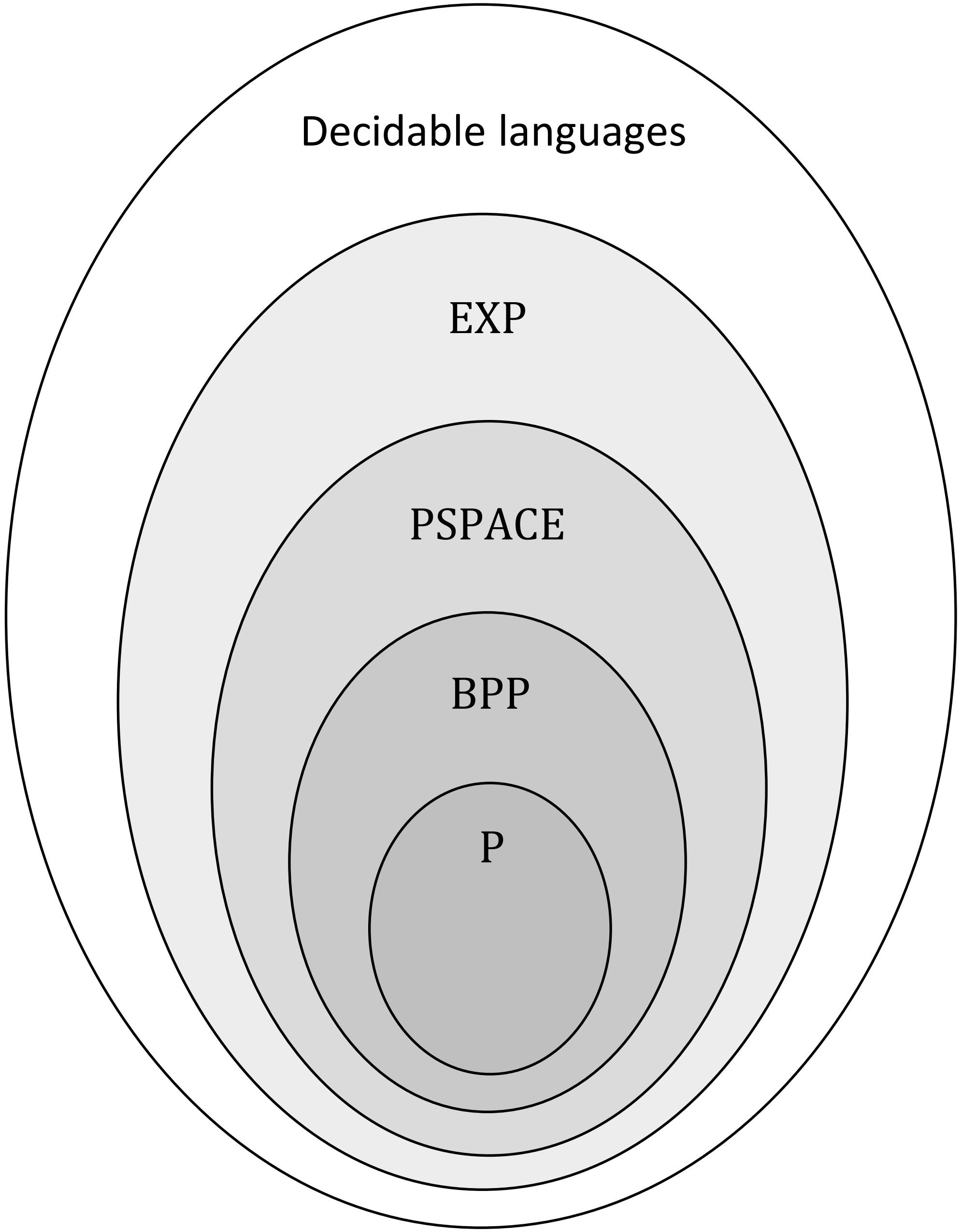
PSPACE vs. EXP

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

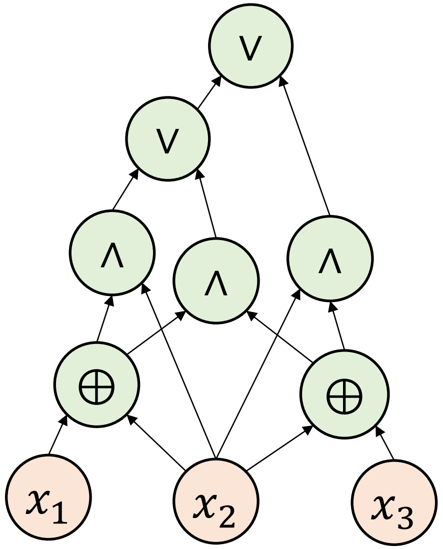
Theorem: $\text{PSPACE} \subseteq \text{EXP}$

- **Proof (1 slide):** Let M be a Turing machine that decides a language Y
- Exercise 4: For each input, $\text{Time} \leq C^{\text{Space}+1}$, where C depends only on M
- When $\text{Space} = \text{poly}(n)$, we get

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

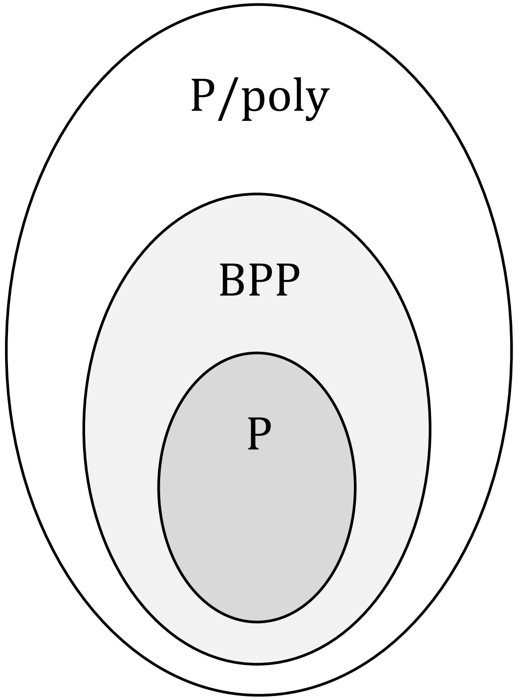

Decidable languages
EXP
PSPACE
BPP
P

Derandomization beyond brute force



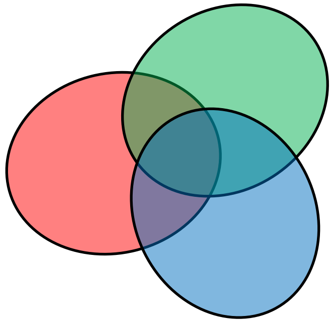
Adleman's Theorem: $\mathrm{BPP} \subseteq \mathrm{P/poly}$

Tantalizingly similar to "$\mathrm{P} = \mathrm{BPP}$"



- Recall $\mathrm{P/poly} = \mathrm{PSIZE}$
- Note: There is no randomness in the definitions of $\mathrm{P/poly}$ and PSIZE!
- Proof of Adleman's theorem: Next 5 slides

The union bound



- Key fact from probability theory:

The Union Bound: For any events $E_1, E_2, \dots, E_k$, we have

$$\Pr[E_1 \text{ or } E_2 \text{ or} \dots \text{or } E_k] \leq \Pr[E_1] + \Pr[E_2] + \cdots + \Pr[E_k]$$

- Example: If we pick two cards from a deck, then

$$\Pr[\text{card 1 is a queen or card 2 is a queen}] \leq \frac{1}{13} + \frac{1}{13} = \frac{2}{13}$$

Adleman proof step 1: Amplification

- Let $Y \in \mathrm{BPP}$
- By the amplification lemma, there exists a poly-time randomized Turing machine M such that for every $n \in \mathbb{N}$ and every $w \in \{0,1\}^n$:
  - If $w \in Y$, then $\Pr[M \text{ accepts } w] > 1 - 1/2^n$
  - If $w \notin Y$, then $\Pr[M \text{ accepts } w] < 1/2^n$

Adleman proof step 2: Good random bits

- Let $w, x \in \{0, 1\}^*$
- **Definition:** x is good relative to w if:
  - $w \in Y$ and M accepts when tape 1 is initialized with w and tape 2 is initialized with x, or
  - $w \notin Y$ and M rejects when tape 1 is initialized with w and tape 2 is initialized with x

Adleman proof step 2: Good random bits

Lemma: For every n, there exists $x_* \in \{0,1\}^{n^k}$ that is good relative to every $w \in \{0,1\}^n$

- **Proof:** Pick $x \in \{0,1\}^{n^k}$ uniformly at random. Then

Union Bound

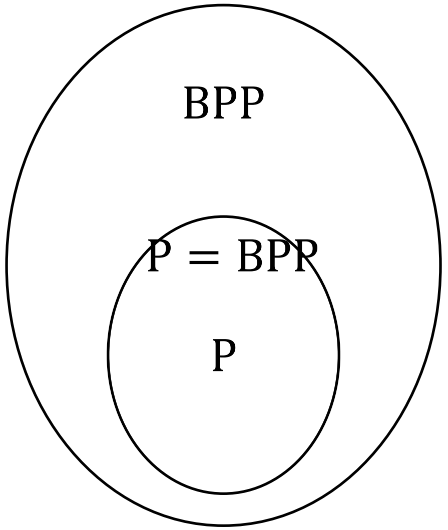
$$\Pr\left[\begin{array}{c}\text{there exists } w \in \{0,1\}^n \\ \text{relative to which } x \text{ is bad}\end{array}\right] \leq \sum_{w \in \{0,1\}^n} \Pr[x \text{ is bad relative to } w] < 2^n \cdot \frac{1}{2^n} = 1$$

- The claim follows!

Adleman proof step 3: Advice

- Use the "good random bits" x_* as advice
- Given w and x_*, simulate M with tape 1 initialized with w and tape 2 initialized with x_*
- This shows $Y \in \mathrm{P/poly}$

P vs. BPP



- We have seen two derandomization methods:
  - Brute-force
  - Adleman’s theorem
- There are other methods that are more sophisticated
  - (Beyond the scope of this course)
- Because of these other methods, most experts conjecture $\mathrm{P} = \mathrm{BPP}$!

Is $\mathbf{P}$ a good model of tractability?

Robustness of P, revisited

- Let $Y \subseteq \{0, 1\}^*$. If $Y \notin \mathrm{P}$, then Y cannot be decided by…
  - A poly-time one-tape Turing machine
  - A poly-time multi-tape Turing machine
  - A poly-time word RAM program
  - A poly-time randomized Turing machine (assuming $\mathrm{P} = \mathrm{BPP}$)
- **OBJECTION:** "This still leaves open the possibility that I could somehow build a device that decides Y in polynomial time."

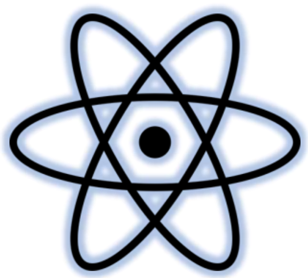
Extended Church-Turing Thesis

~~**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 it were true, the thesis would justify studying P
- But the thesis is probably false!
- Key challenge: 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