CMSC 28100

Introduction to
Complexity Theory

Spring 2024
Instructor: William Hoza

Course Review

Which problems

can be solved

through computation?

Strings and languages

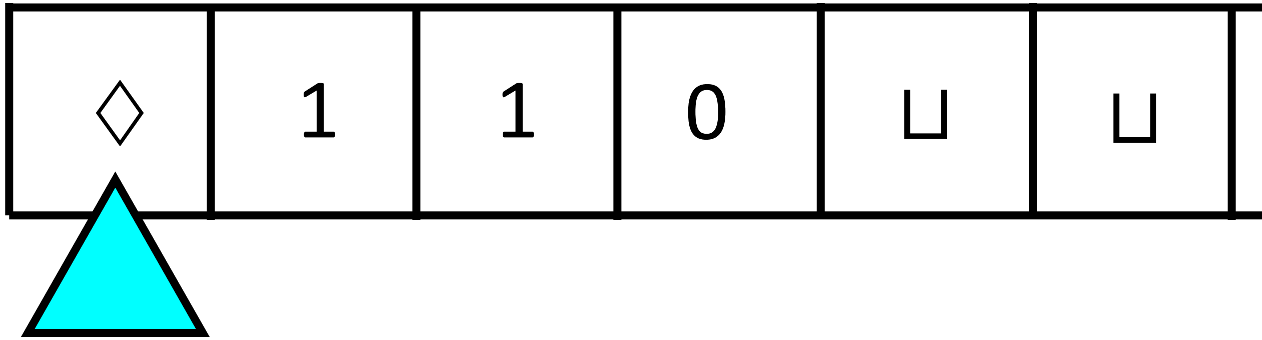
- Σ^* is the set of all strings over the alphabet Σ (of any finite length)
- A language is a subset $L \subseteq \Sigma^*$
- A language models a computational problem, namely, the problem of distinguishing strings in L from strings outside L
- To study other types of problems, we can often formulate a closely related language. (E.g., problem set 6: searching for large cliques)

Which problems

can be solved

through computation?

Turing machines



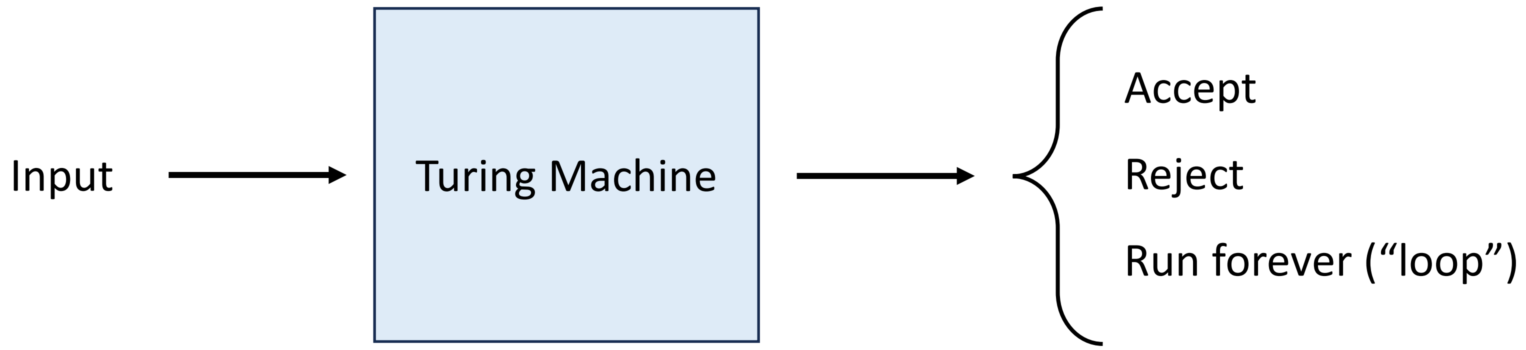
- A tape extends infinitely to the right
- The machine uses a head to read from and write to the tape
- The machine also has an internal state
- "Local evolution" of a Turing machine is described by the transition function $\delta: Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$

Which problems

can be solved

through computation?

Deciding a language



- Let M be a Turing machine and let L be a language
- We say that M decides L if M accepts every $w \in L$ and M rejects every $w \in \Sigma^* \setminus L$

The Church-Turing Thesis

- Let L be a language

Church-Turing Thesis:

The problem of deciding whether a given string is in L

can be "solved through computation" if and only if

there is a Turing machine that decides L.

Intuitive notion

Mathematically precise notion

The Physical Church-Turing Thesis

- Let L be a language

Physical Church-Turing Thesis:

It is physically possible to build a device that decides L

if and only if there is a Turing machine that decides L.

Code as data

- A Turing machine M represents an algorithm
- At the same time, a TM M can be encoded as a string $\langle M \rangle$
- This string $\langle M \rangle$ could be the input or output of a different algorithm!

Universal Turing machines

Theorem: There exists a Turing machine U such that for every Turing machine M and every input w:

- If M accepts w, then U accepts $\langle M, w \rangle$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on input w, then U loops on $\langle M, w \rangle$.

Universal Turing machines

- If you are stranded on an alien planet and you are trying to build a computer, your job is to build a universal Turing machine
- A universal Turing machine can be "programmed" to do anything that is computationally possible

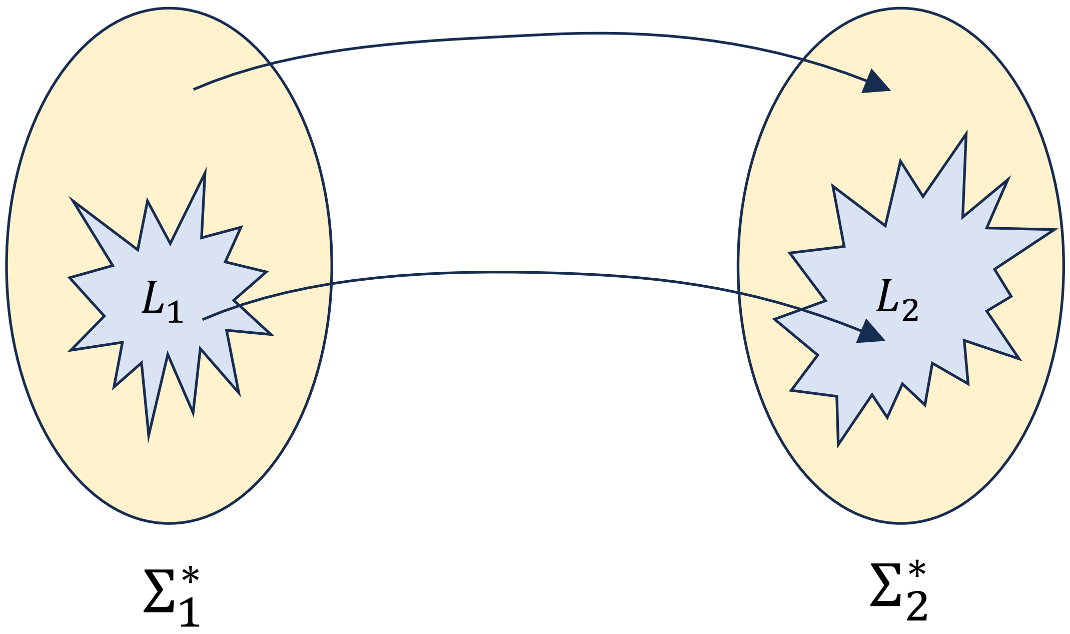
The halting problem

- $\mathrm{HALT} = \{\langle M, w\rangle : M \text{ is a Turing machine that halts on } w\}$

Theorem: HALT is undecidable

Reductions

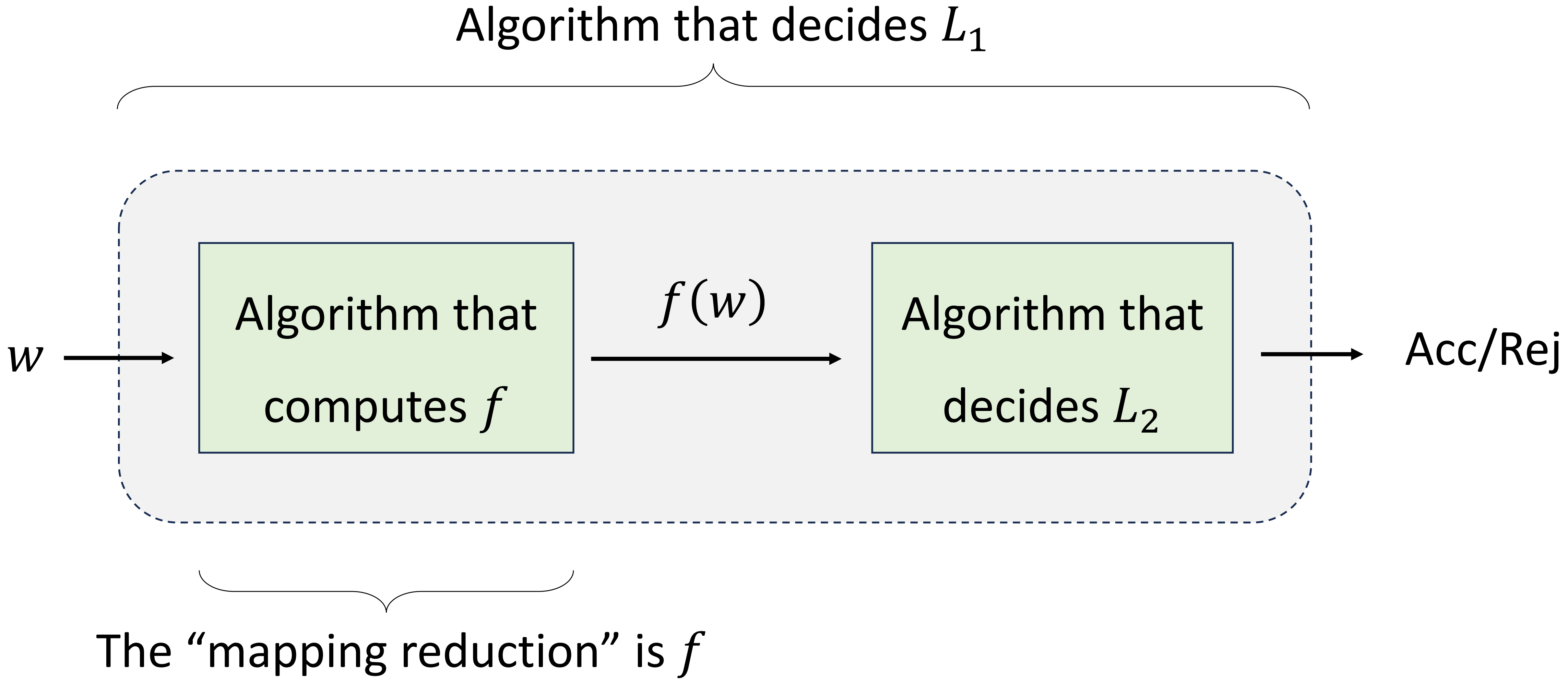
- A mapping reduction from L_1 to L_2 is a way of converting instances of L_1 into equivalent instances of L_2



"YES" maps to "YES"

"NO" maps to "NO"

Using reductions to prove decidability



Undecidability via reductions

- To prove that some language L is undecidable:
  - Identify a suitable language L_{HARD} that we previously proved is undecidable
  - Design a mapping reduction f from L_{HARD} to L
- Example: Post's Correspondence Problem

Theorem: PCP is undecidable

Asymptotic analysis

Notation	In words	Analogy
T is $o(f)$	$T(n)$ grows more slowly than $f(n)$	$<$
T is $O(f)$	$T(n)$ is at most $c \cdot f(n)$	$\leq$
T is $\Theta(f)$	$T(n)$ and $f(n)$ grow at the same rate	$=$
T is $\Omega(f)$	$T(n)$ is at least $c \cdot f(n)$	$\geq$
T is $\omega(f)$	$T(n)$ grows more quickly than $f(n)$	$>$

Polynomial-time computation

- We mainly focus on the distinction between polynomial-time algorithms and exponential-time algorithms
- We proved that $n^k = o(2^n)$ for every constant k
- Exponential-time algorithms are almost worthless
- Polynomial-time algorithms are usually usable

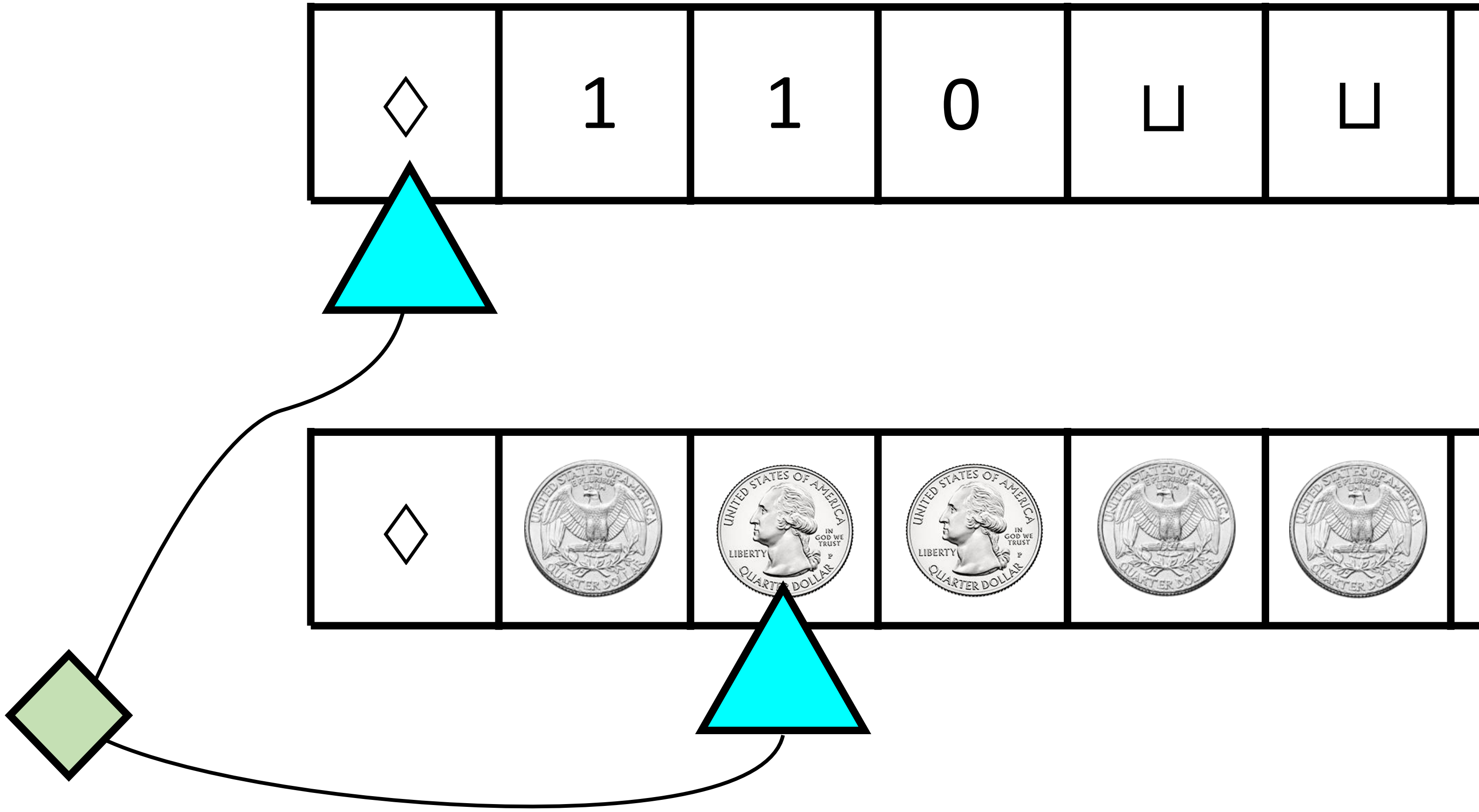
Complexity classes

- A complexity class is a set of languages
- A language is in P if it can be decided by a polynomial-time TM
- A language is in PSPACE if it can be decided by a polynomial-space TM
- A language is in EXP if it can be decided by a TM with time complexity $2^{\mathrm{poly}(n)}$

Randomized Turing machines

Input tape ➞

Randomness tape ➞

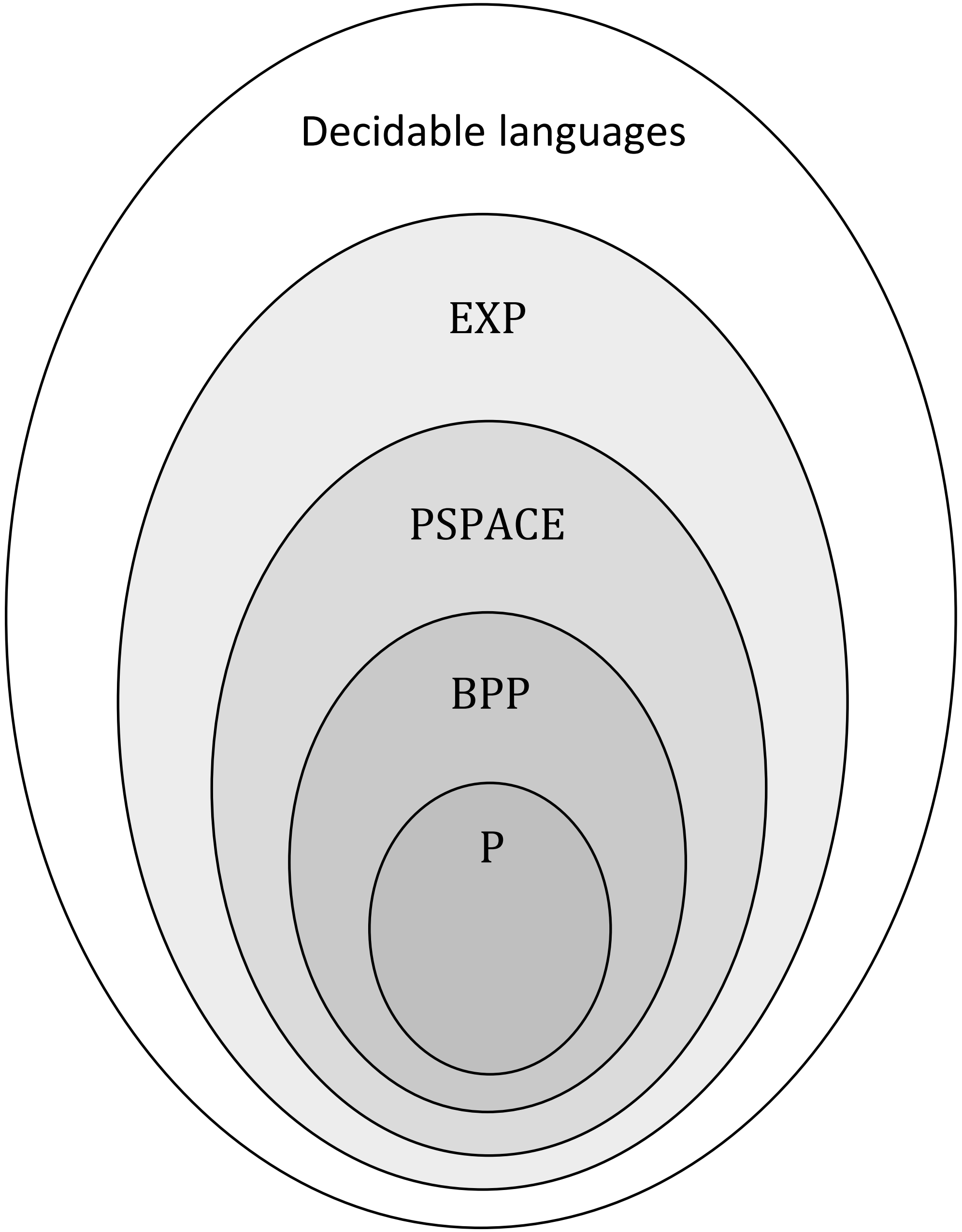


The complexity class BPP

- A language L is in BPP if there is a polynomial-time randomized Turing machine M such that:
  - For every $w \in L$, we have $\Pr[M \text{ accepts } w] \geq 2/3$
  - For every $w \notin L$, we have $\Pr[M \text{ rejects } w] \geq 2/3$
- **Amplification lemma:** We can replace $2/3$ with $1 - 1/2^{n^k}$

$\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$

- $\mathrm{P} \subseteq \mathrm{BPP}$ because we can elect to not use our random bits
- $\mathrm{BPP} \subseteq \mathrm{PSPACE}$ because we can deterministically try all possible settings of the random tape ("brute-force derandomization")
- $\mathrm{PSPACE} \subseteq \mathrm{EXP}$ because a polynomial-space algorithm that uses more than exponential time would repeat a configuration (problem set 2), hence it would get stuck in an infinite loop

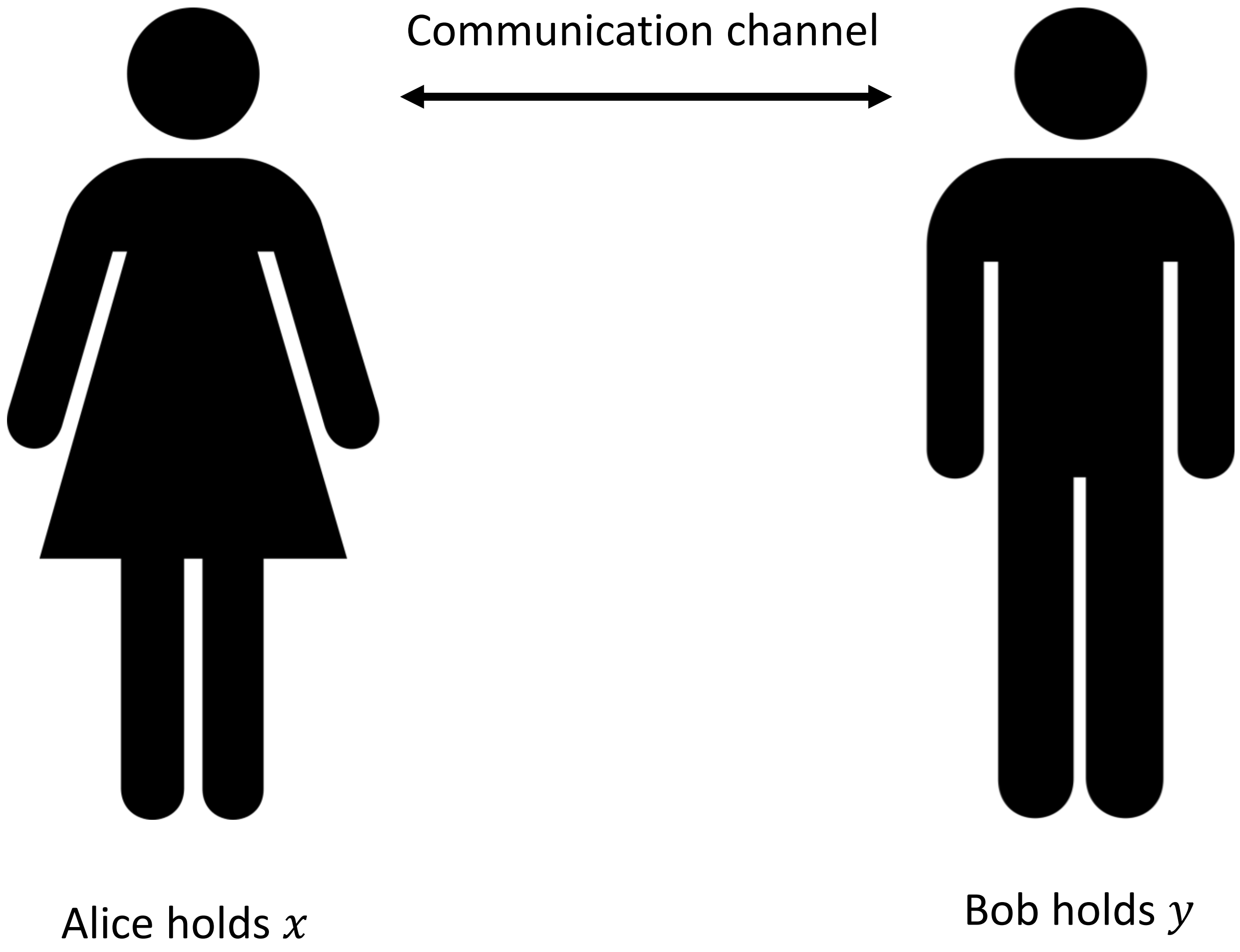

Decidable languages
EXP
PSPACE
BPP
P

P vs. BPP

- To prove that certain problems are intractable, we need a mathematical model of tractability
- P and BPP are both reasonable models of tractability
- **Open Question:** Does $\mathrm{P} = \mathrm{BPP}$?

Communication complexity

- Goal: Compute $f(x, y)$ using as little communication as possible



Communication complexity of $\mathbf{EQ}_n$

- $\mathrm{EQ}_n(x, y) = 1 \Leftrightarrow x = y$

Theorem: Every deterministic communication protocol that computes EQ_n has cost at least $n + 1$

Theorem: There is a randomized communication protocol with cost $O(\log n)$ that computes EQ_n with high probability

P vs. BPP

- Communication complexity might suggest $\mathrm{P} \neq \mathrm{BPP}$
- However, we gathered some more "evidence" about the P vs. BPP question by studying Boolean logic: AND / OR / NOT operations

Conjunctive normal form

- A literal is a Boolean variable or its negation (x_i or $\bar{x}_i$)
- A clause is a disjunction (OR) of literals
- A conjunctive normal form (CNF) formula is a conjunction (AND) of clauses
- In other words, a CNF formula is an AND of ORs of literals

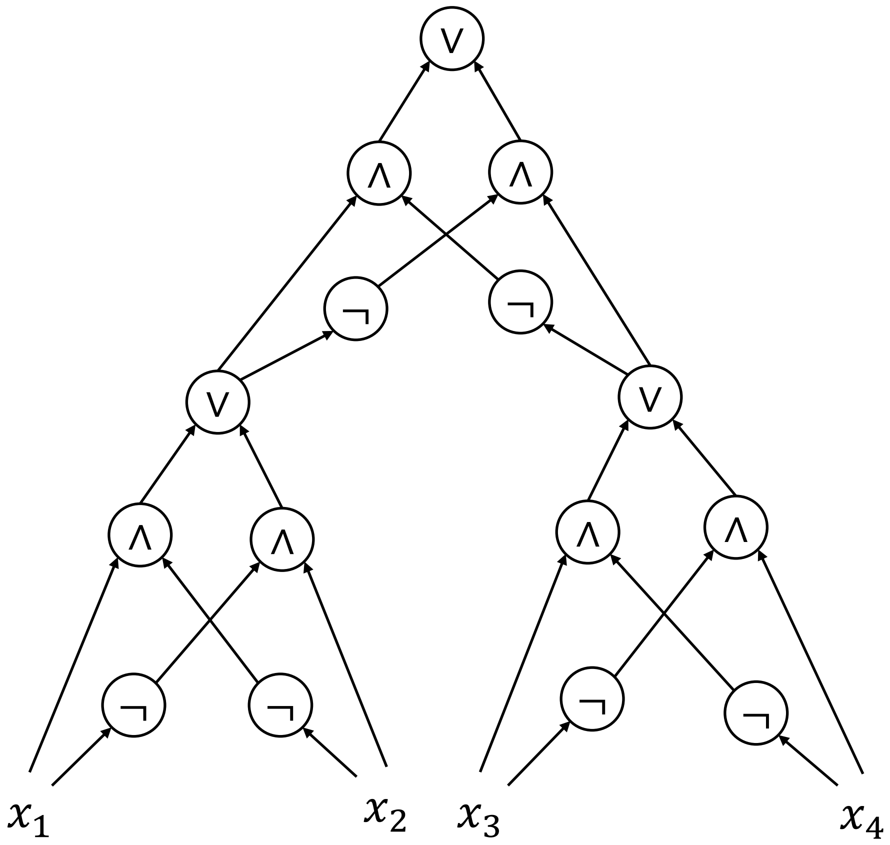
Conjunctive normal form

Lemma: Every function $f: \{0,1\}^n \to \{0,1\}$ can be represented by a CNF formula in which

- There are at most 2^n clauses
- Each clause has at most n literals

Boolean circuits

- A “circuit” is a network of AND/OR/NOT gates applied to Boolean variables



Circuit complexity

- CNF representation $\Rightarrow$ Every function $f: \{0,1\}^n \to \{0,1\}^m$ can be computed by a circuit of size $O(2^n \cdot n \cdot m)$
- In your homework, you showed that there exists a function $f: \{0,1\}^n \to \{0,1\}$ with circuit complexity $\Omega(2^n/n)$

Polynomial-size circuits

- A language L is in PSIZE if for every n, there is a circuit of size $\mathrm{poly}(n)$ that decides L restricted to inputs of length n

Theorem: $\mathrm{P} \subseteq \mathrm{PSIZE}$.

- Polynomial-Time Algorithm $\Rightarrow$ Polynomial-Size Circuits
- In your homework, you showed that $\mathrm{P} \neq \mathrm{PSIZE}$

Adleman’s theorem

Adleman’s Theorem: $\mathrm{BPP} \subseteq \mathrm{PSIZE}$

- Adleman’s theorem is tantalizingly similar to the statement “$\mathrm{P} = \mathrm{BPP}$”
- **Conjecture:** $\mathrm{P} = \mathrm{BPP}$

The 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 $L \in \mathrm{P}$.

- The Extended Church-Turing thesis is probably false because of quantum computing

The Time Hierarchy Theorem

- Let $T: \mathbb{N} \to \mathbb{N}$ be any "reasonable" (time-constructible) function

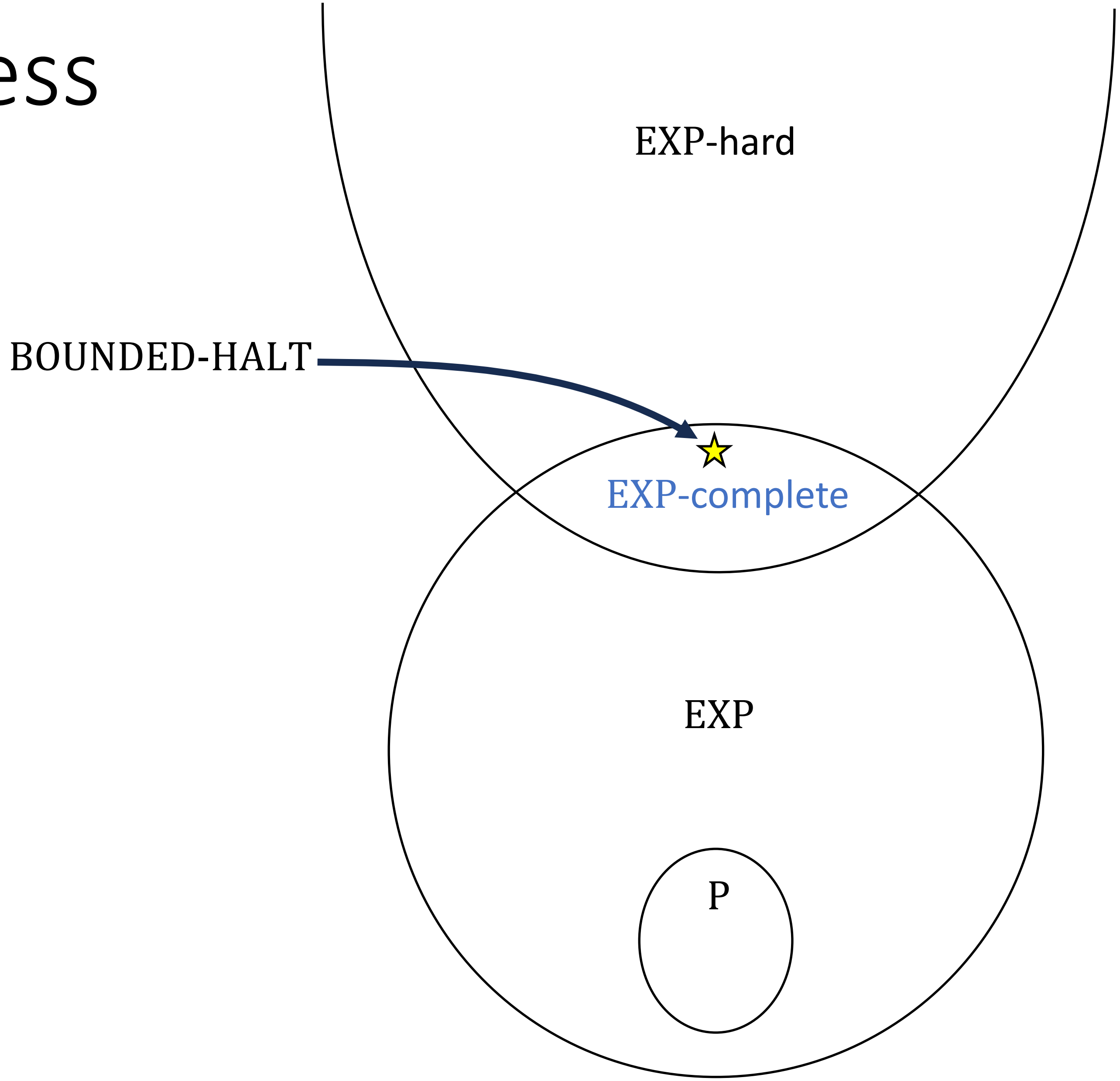
Time Hierarchy Theorem: $\mathrm{TIME}\big(o(T)\big) \neq \mathrm{TIME}(T^3)$

- Consequence: $\mathrm{P} \neq \mathrm{EXP}$

EXP-completeness

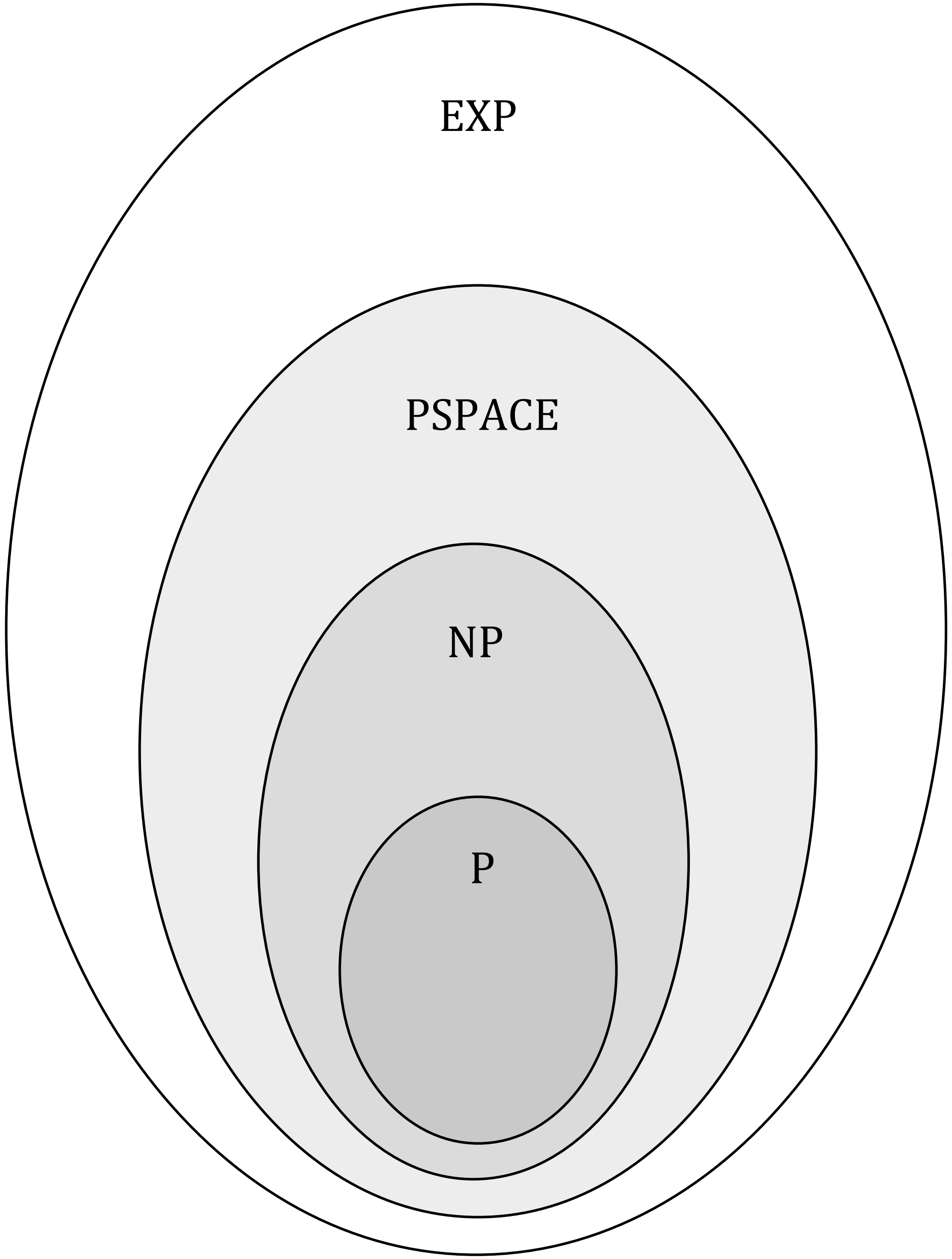
- If every language in EXP reduces to L in polynomial time, then we say that L is EXP-hard
- If L is EXP-hard and $L \in \mathrm{EXP}$, then we say that L is EXP-complete
- EXP-complete languages are not in P
- $\mathrm{BOUNDED\text{-}HALT}$ is EXP-complete

EXP-completeness



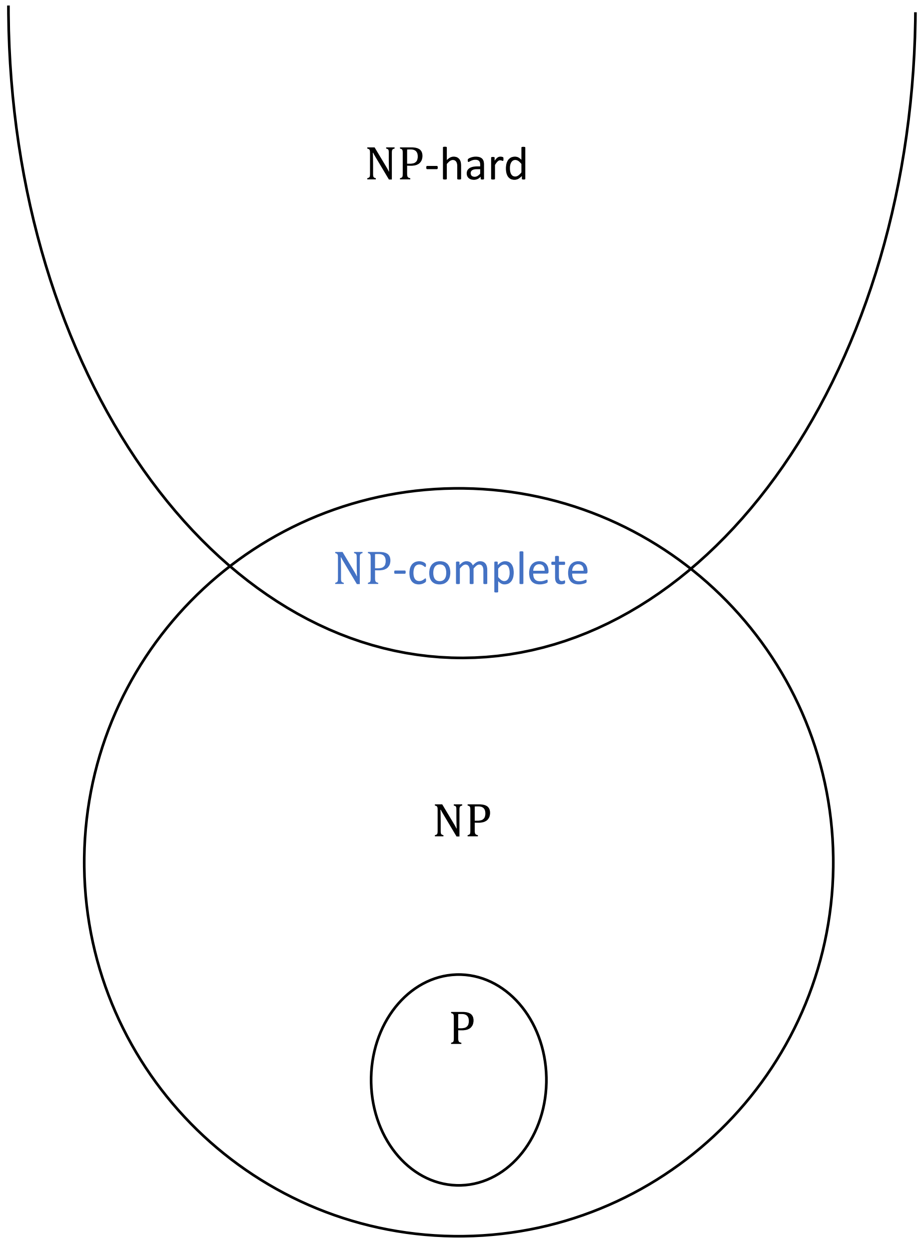
The complexity class NP

- A language L is in NP if there is a polynomial-time randomized Turing machine M such that:
  - For every $w \in L$, we have $\Pr[M \text{ accepts } w] \neq 0$
  - For every $w \notin L$, we have $\Pr[M \text{ accepts } w] = 0$
- Equivalent: Every $w \in L$ has a certificate of membership, and certificates can be verified in (deterministic) polynomial time


EXP
PSPACE
NP
P

NP-completeness

- A language L is NP-complete if $L \in \mathrm{NP}$ and every language in NP reduces to L in polynomial time



Circuit satisfiability

- $\text{CIRCUIT-SAT} = \{\langle C \rangle : C \text{ is a satisfiable circuit}\}$

Theorem: $\textbf{CIRCUIT-SAT}$ is NP-complete

- Key idea: If $L \in \text{P}$, then not only does L have polynomial-size circuits ($L \in \text{PSIZE}$), but in fact we can efficiently construct the circuits

The Cook-Levin Theorem

- **Definition:** A k-CNF formula is an AND of ORs of at most k literals
- **Definition:** $k\text{-SAT} = \{\langle\phi\rangle : \phi \text{ is a satisfiable } k\text{-CNF formula}\}$

The Cook-Levin Theorem: 3-SAT is NP-complete

- Using this theorem, we also proved that CLIQUE is NP-complete
- On your homework, you showed, e.g., that 3-COLORABLE is NP-complete

The P vs. NP problem

- We conjecture that $\mathrm{P} \neq \mathrm{NP}$: Solving and verifying are different
- A proof that $\mathrm{P} = \mathrm{NP}$ would change the world
  - *Assuming the proof gives us truly practical algorithms
- We could solve countless important problems in polynomial time 🥹
- Hackers could break our encryption schemes in polynomial time 🥺

Lessons

- Computation has intrinsic limitations
- Mathematics and computer science form a powerful combination
- Complexity theory enables us to formulate and study profound questions
  - Questions about the human condition
  - Questions about the physical universe

Thank you!

- Teaching you has been a privilege
- I hope you've enjoyed taking the course as much as I've enjoyed teaching it
- Please fill out the College Course Feedback Form using My.UChicago (deadline is May 26)
- See you next week for office hours and the final exam!