CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

Course Review

Which problems

can be solved

through computation?

Strings and languages

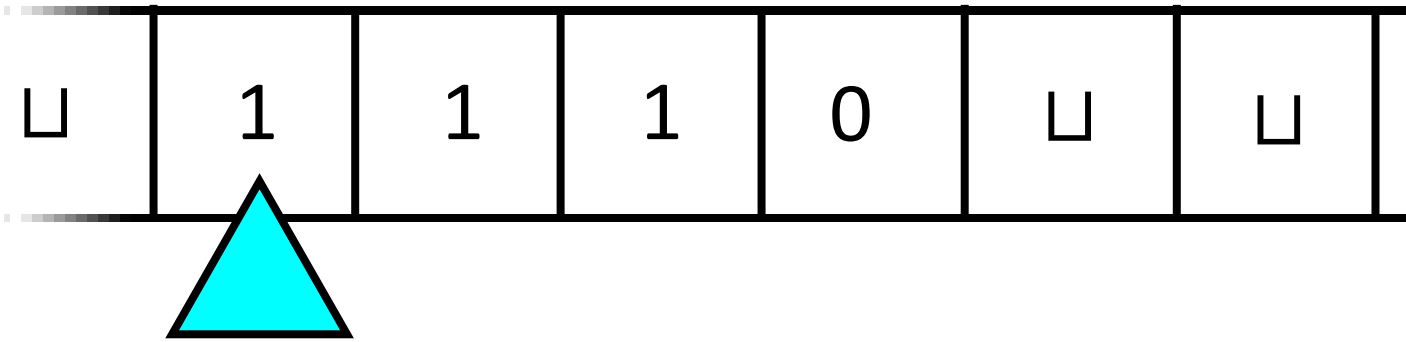
- Σ^* is the set of all strings over the alphabet Σ (of any finite length)
- A binary language is a subset $Y \subseteq \{0,1\}^*$
- Corresponding problem: Given $w \in \{0,1\}^*$, figure out whether $w \in Y$
  - To study other types of problems, we can often formulate a closely related language
  - E.g., Exercise 14: Searching for large cliques

Which problems

can be solved

through computation?

Turing machines



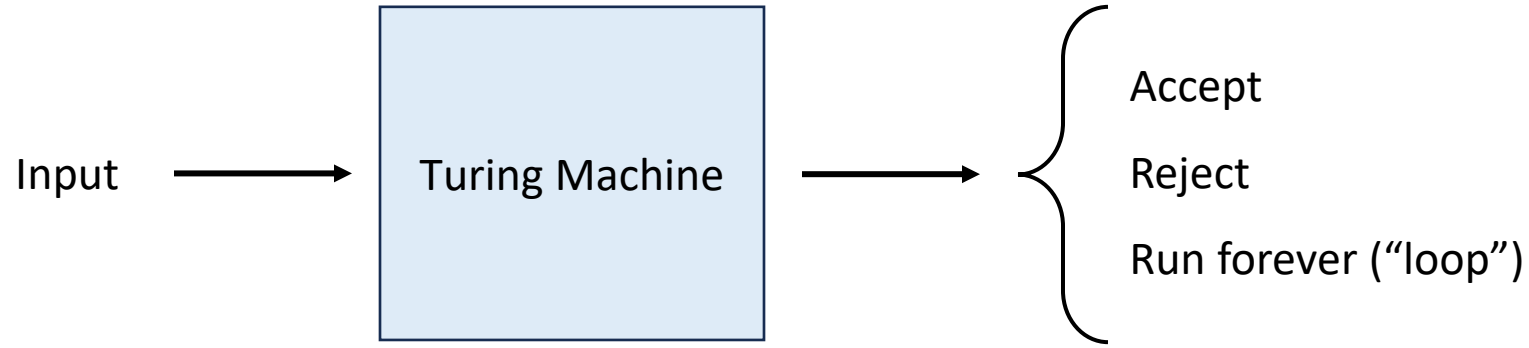
- There is an infinitely long tape
- The machine uses a head to read from and write to the tape
- The machine also has an internal state
- "Local evolution" of a Turing machine is described by the transition function $\delta: Q \times \Sigma \to Q \times \Sigma \times \{\mathrm{L}, \mathrm{R}\}$

Which problems

can be solved

through computation?

Deciding a language



- Let M be a Turing machine and let Y be a language
- We say that M decides Y if M accepts every $w \in Y$ and M rejects every $w \in \{0,1\}^* \setminus Y$

The Church-Turing Thesis

- Let $Y \subseteq \{0,1\}^*$

Church-Turing Thesis:

The problem of deciding whether a given string is in Y can be "solved through computation" if and only if there is a Turing machine that decides Y.

← Intuitive notion

← Mathematically precise notion

The Physical Church-Turing Thesis

- Let $Y \subseteq \{0,1\}^*$

Physical Church-Turing Thesis:

It is physically possible to build a device that decides Y

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

Code as data

- A Turing machine M represents an algorithm
- At the same time, 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 $w \in \{0,1\}^*$:

- 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

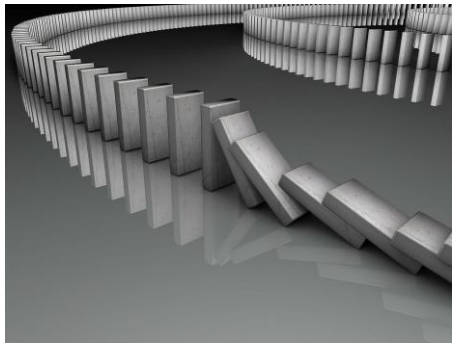
Undecidability

- $\text{SELF-REJECTORS} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$

Theorem: SELF-REJECTORS is undecidable

- We can rule out all possible algorithms!

Reductions

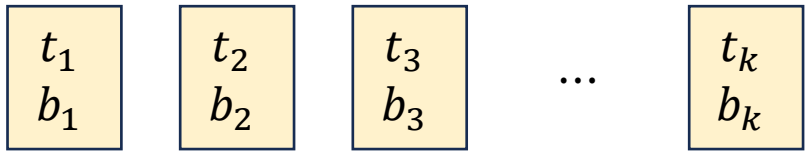


- To prove that Y_{NEW} is undecidable:
  - Identify some known undecidable language Y_{OLD}
  - Assume for the sake of contradiction that Y_{NEW} is decidable
  - Design an algorithm that decides Y_{OLD}, using hypothetical device deciding Y_{NEW}
- Example: $\mathrm{HALT} = \{\langle M, w\rangle : M \text{ halts on } w\}$

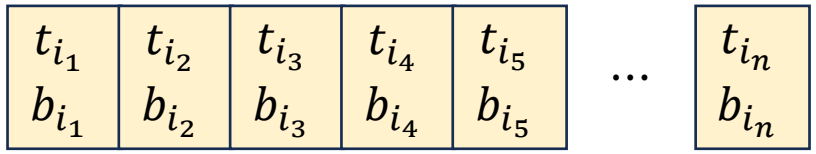
Theorem: HALT is undecidable

Post's Correspondence Problem

- Input: A set of "dominos"



- Goal: Determine whether it is possible to generate a "match"



in which the sequence of top symbols equals the sequence of bottom symbols

Theorem: PCP is undecidable

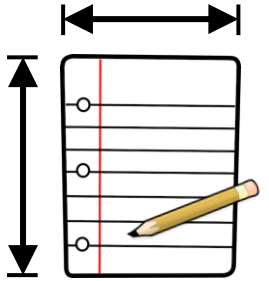
Asymptotic analysis

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

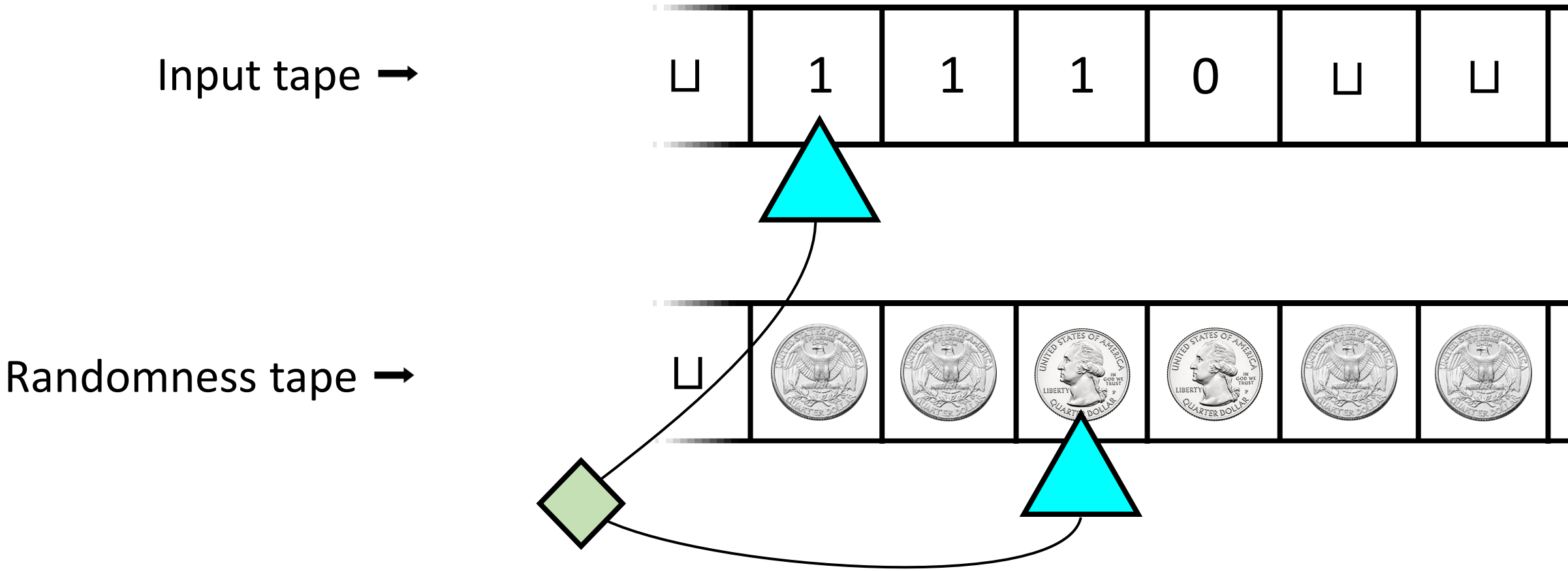
- We mainly focus on the distinction between polynomial-time algorithms and exponential-time algorithms
- We proved that $n^k = o(2^n)$ for every constant k
- Exponential-time algorithms are almost worthless
- Polynomial-time algorithms are usually usable

Complexity classes



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

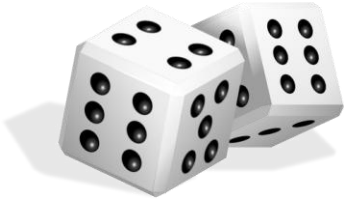
Randomized Turing machines



The complexity class $\mathbf{NP}$

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

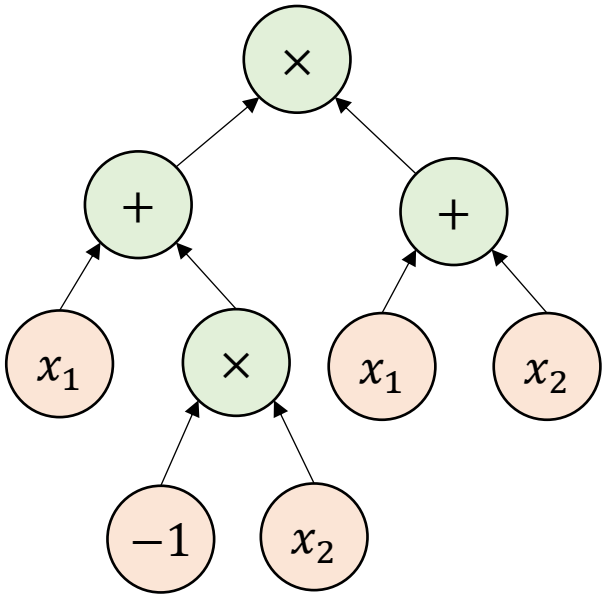
The complexity class BPP



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

Example: Polynomial identity testing

- **Given:** An arithmetic formula F
- **Goal:** Figure out whether $F \equiv 0$

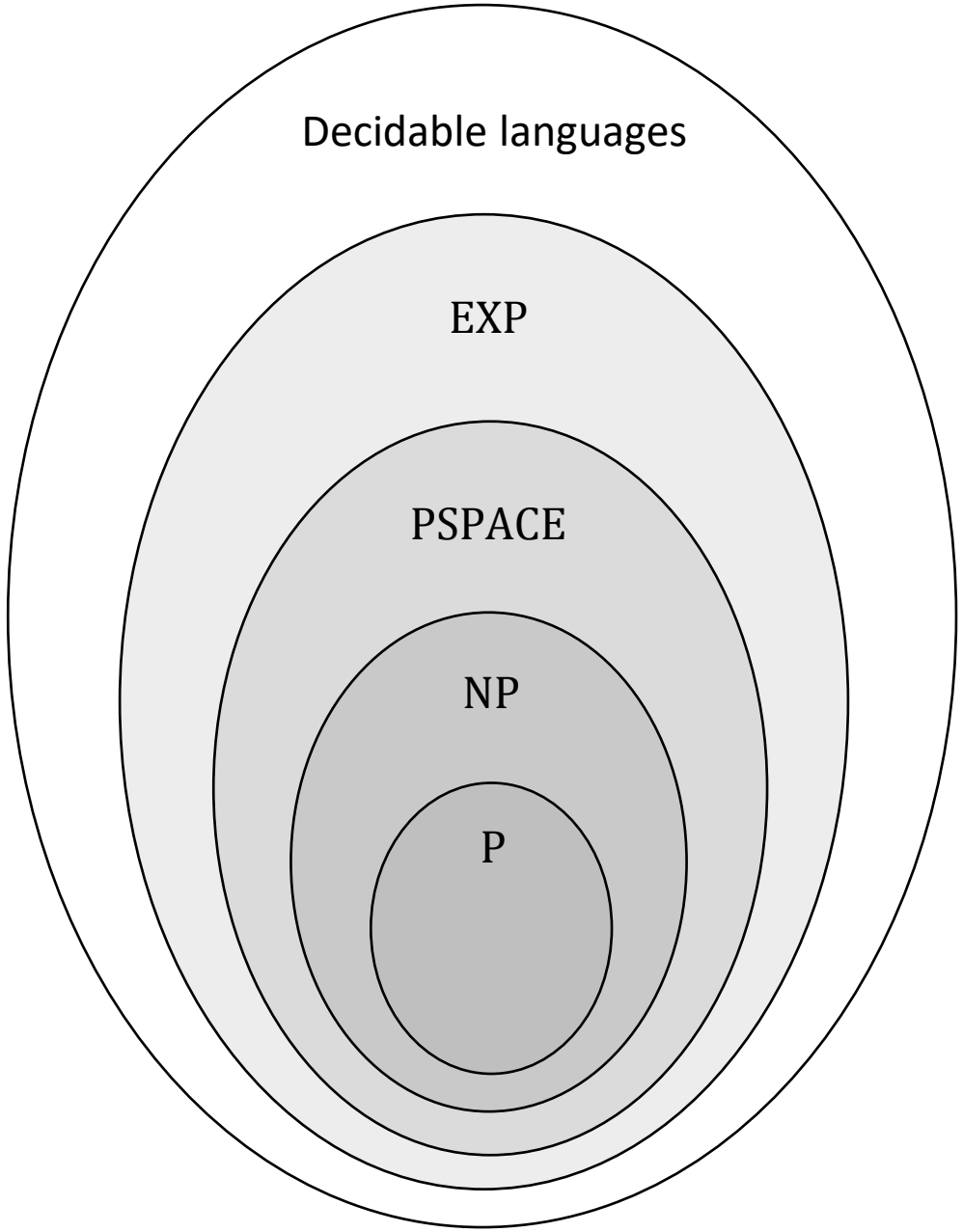


Theorem: $\mathrm{PIT} \in \mathrm{BPP}$

- **Algorithm idea:** Pick $\vec{x}$ at random and check whether $F(\vec{x}) = 0$

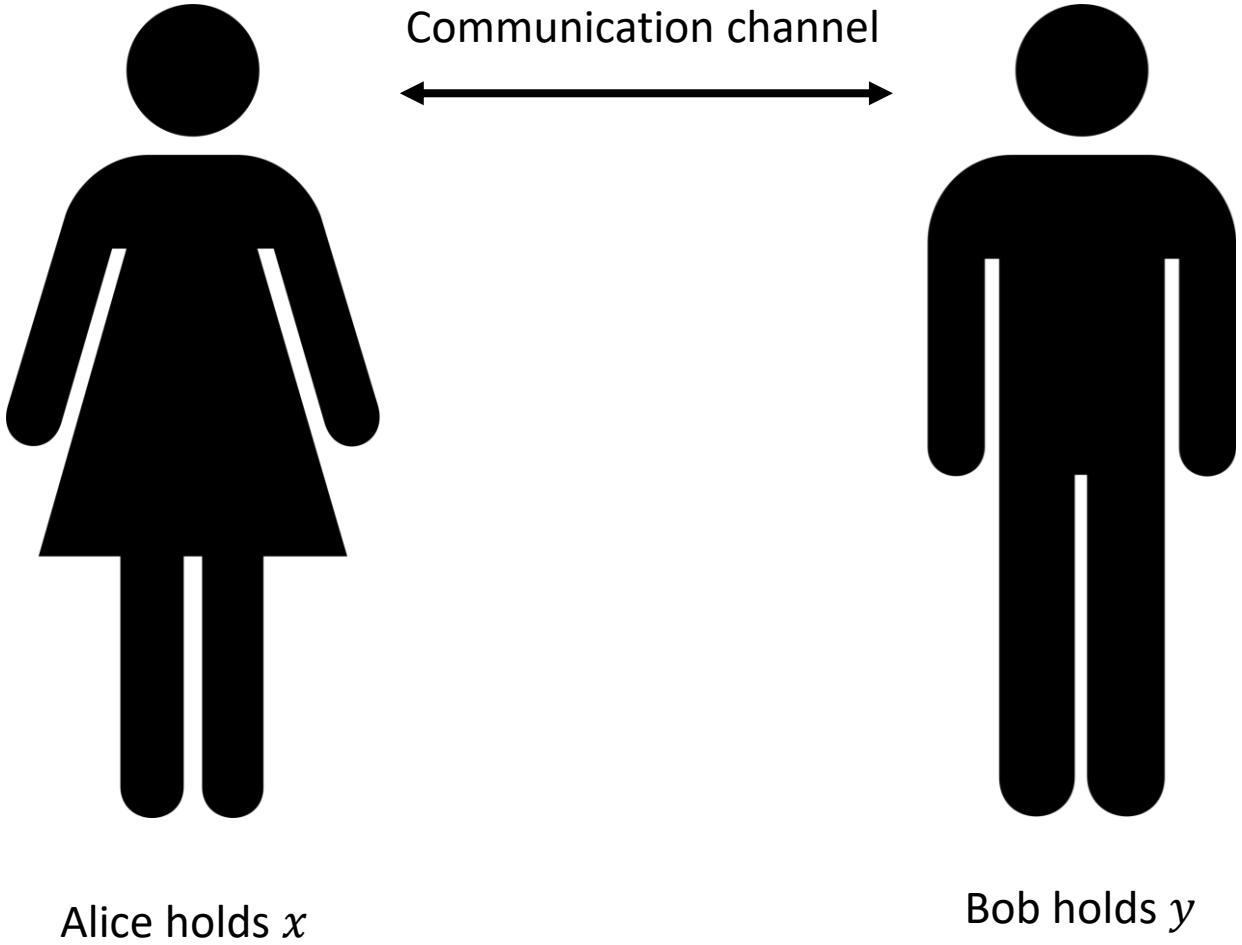
Relationships between complexity classes

- $\mathrm{P} \subseteq \mathrm{BPP}$ and $\mathrm{P} \subseteq \mathrm{NP}$ because we can elect to not use our random bits
- $\mathrm{BPP} \subseteq \mathrm{PSPACE}$ and $\mathrm{NP} \subseteq \mathrm{PSPACE}$ because we can try all possible settings of the random tape (“brute-force derandomization/search”)
- $\mathrm{PSPACE} \subseteq \mathrm{EXP}$ because a polynomial-space algorithm that uses more than exponential time would repeat a configuration (Exercise 5), hence it would get stuck in an infinite loop

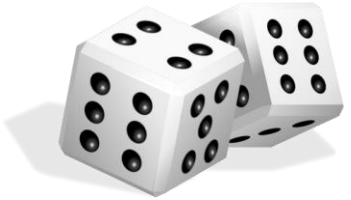

Decidable languages
EXP
PSPACE
NP
P

Communication complexity

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



Communication complexity of EQ_n

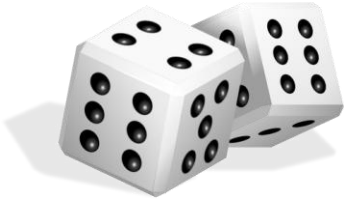


- $\mathrm{EQ}_n(x, y) = 1 \Leftrightarrow x = y$

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



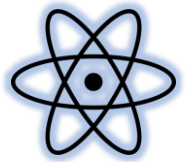
- To prove that certain problems are intractable, we need a mathematical model of tractability
- P and BPP are both reasonable models of tractability
- Which should we use?
- **Conjecture:** $\mathrm{P} = \mathrm{BPP}$, so it's a moot point

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 $Y \in \mathrm{P}$.

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



The Time Hierarchy Theorem

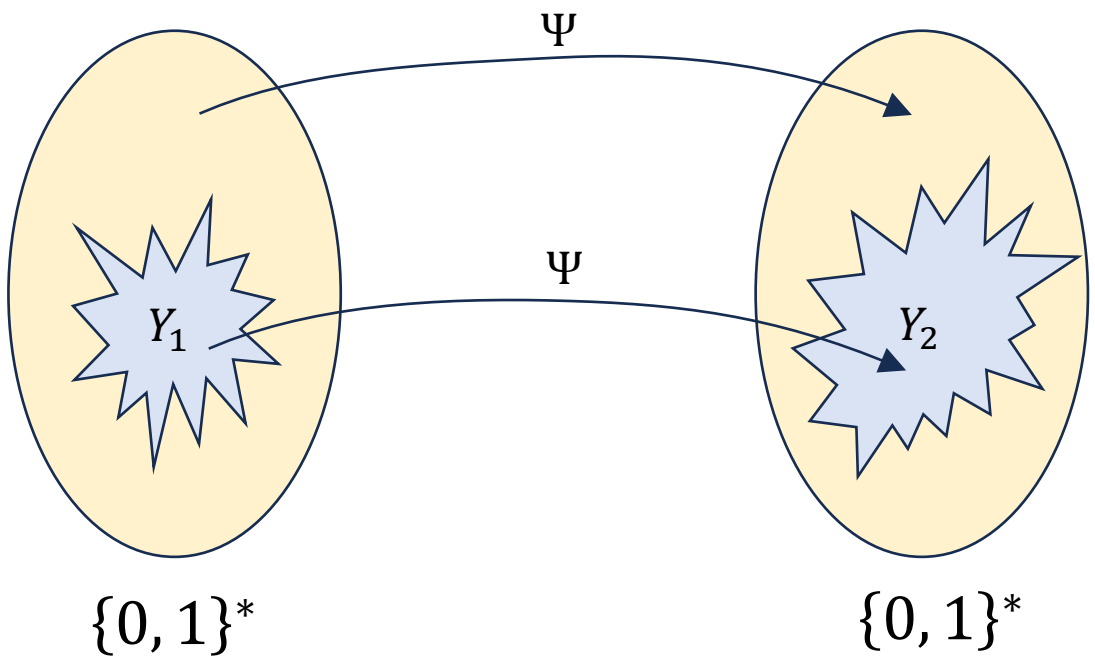
- Let $T: \mathbb{N} \to \mathbb{N}$ be any "reasonable" (time-constructible) function

Time Hierarchy Theorem: There is a language $Y \in \mathrm{TIME}(T^4)$ such that $Y \notin \mathrm{TIME}\big(o(T)\big)$

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

Mapping reductions

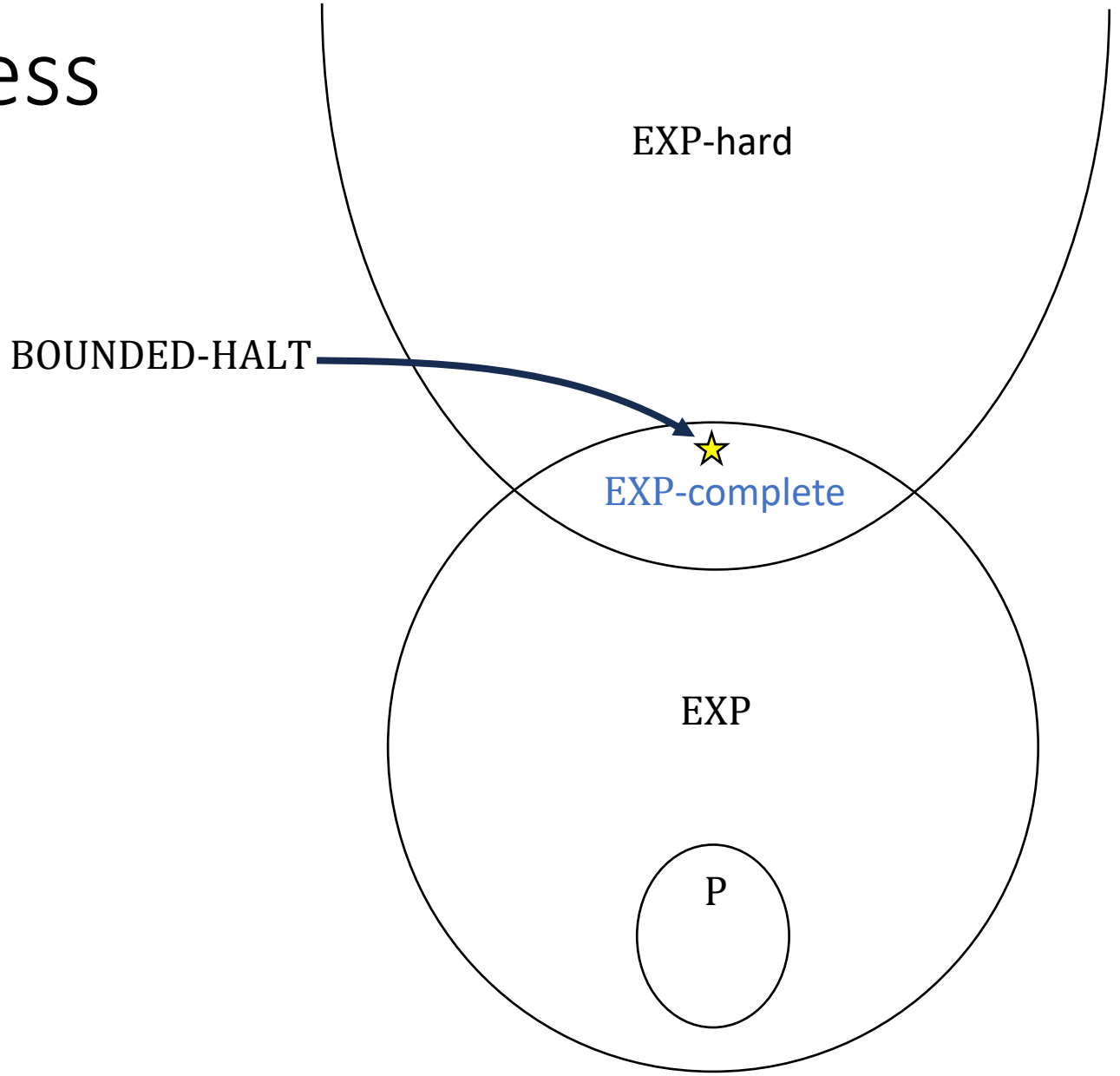
- Let $Y_1, Y_2 \subseteq \{0,1\}^*$
- **Definition:** $Y_1 \leq_P Y_2$ if there is a poly-time computable function $\Psi: \{0,1\}^* \to \{0,1\}^*$ such that
  - $w \in Y_1 \Rightarrow \Psi(w) \in Y_2$
  - $w \notin Y_1 \Rightarrow \Psi(w) \notin Y_2$



EXP-completeness

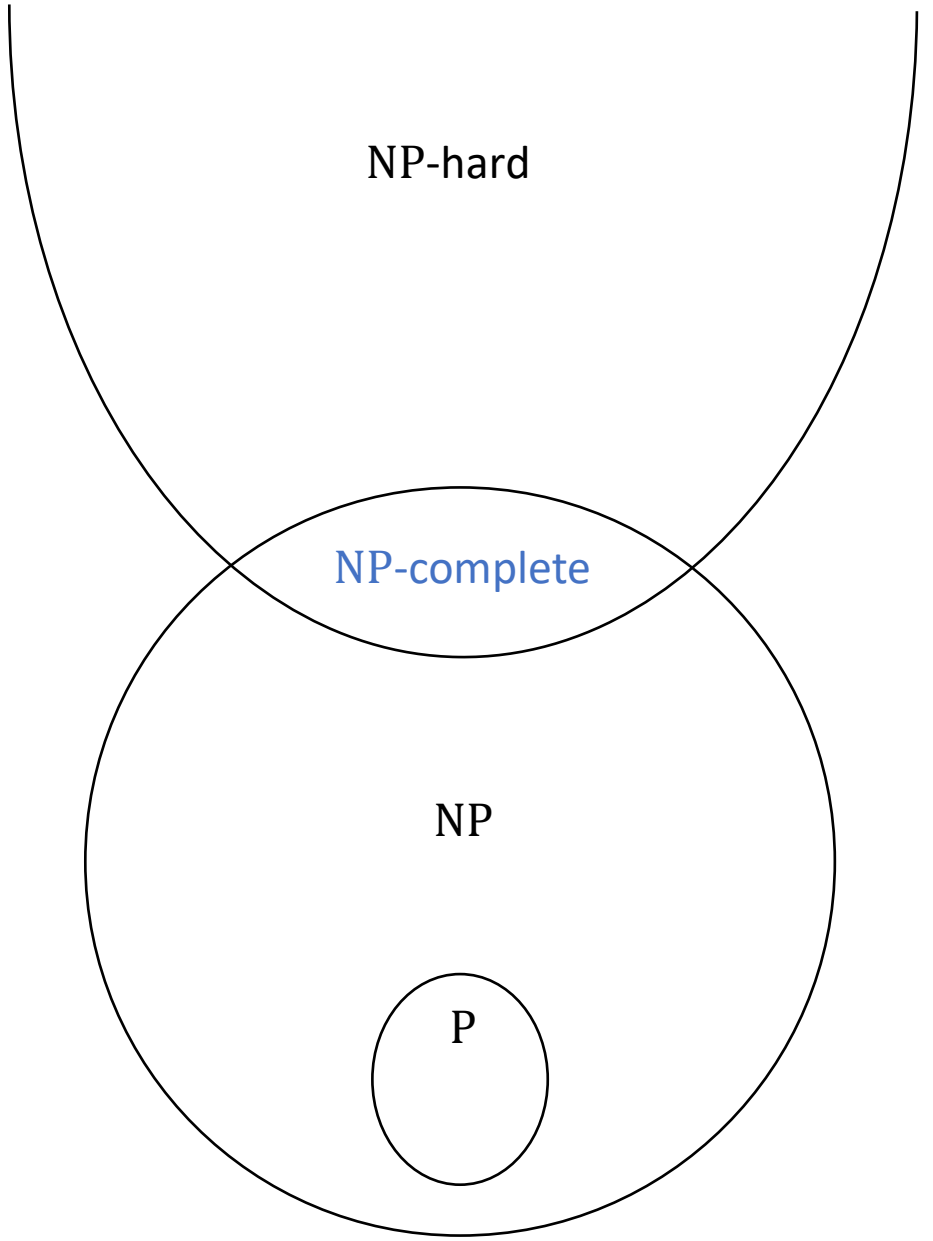
- We say that Y is EXP-hard if $Z \leq_{\mathrm{P}} Y$ for every $Z \in \mathrm{EXP}$
- We say that Y is EXP-complete if Y is EXP-hard and $Y \in \mathrm{EXP}$
- EXP-complete languages are not in P
- Example: $\mathrm{BOUNDED\text{-}HALT}$ is EXP-complete

EXP-completeness



NP-completeness

- A language Y is NP-complete if $Y \in \mathrm{NP}$ and $Z \leq_{\mathrm{P}} Y$ for every $Z \in \mathrm{NP}$



Disjunctive/conjunctive normal form

- A literal is a Boolean variable or its negation (x_i or $\bar{x}_i$)
- DNF formula: OR of ANDs of literals
  - "Disjunction of terms"
- CNF formula: AND of ORs of literals
  - "Conjunction of clauses"

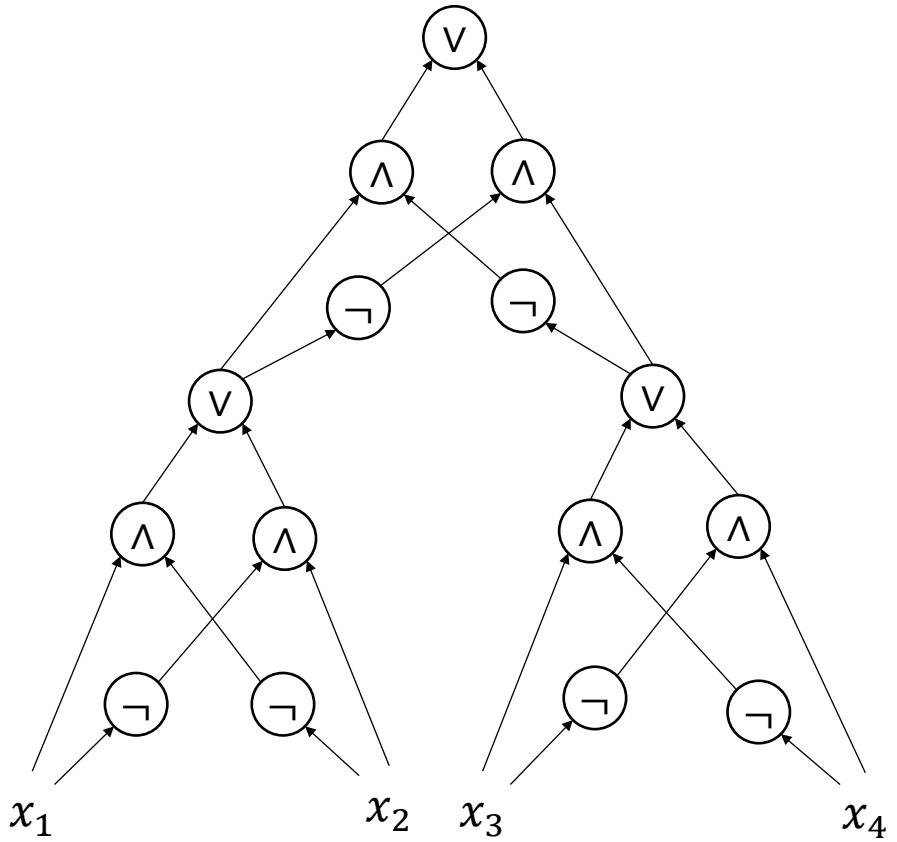
Disjunctive/conjunctive normal form

Theorem: Every function $f: \{0,1\}^n \to \{0,1\}$ can be represented by a DNF formula with at most 2^n terms and at most n literals per term

Theorem: Every function $f: \{0,1\}^n \to \{0,1\}$ can be represented by a CNF formula with at most 2^n clauses and at most n literals per clause

Boolean circuits

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



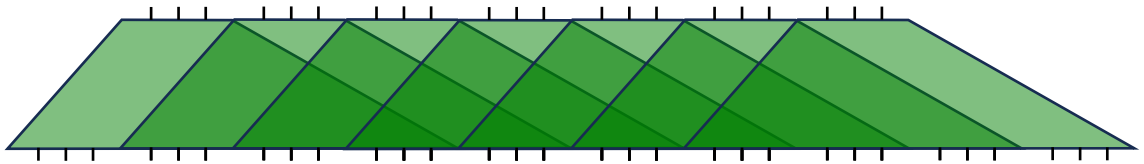
Circuit complexity

- CNF representation ⇒ Every function $f: \{0,1\}^n \to \{0,1\}^m$ can be computed by a circuit of size $O(2^n \cdot n \cdot m)$
- Exercise 22: There exists a function $f: \{0,1\}^n \to \{0,1\}$ with circuit complexity $\Omega(2^n/n)$

Polynomial-size circuits

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

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



- Polynomial-Time Algorithm ⇒ Polynomial-Size Circuits
- Exercise 23: $\mathrm{P} \neq \mathrm{PSIZE}$

Adleman's theorem

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

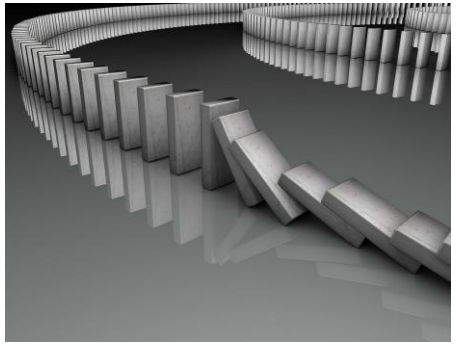
- Tantalizingly similar to the statement "$\mathrm{P} = \mathrm{BPP}$"

Circuit satisfiability

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

Theorem: CIRCUIT-SAT is NP-complete

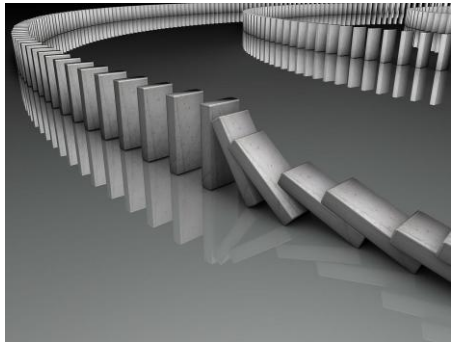
The Cook-Levin Theorem



- **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

More NP-complete problems



- CLIQUE
- UNDIRECTED-HAM-CYCLE
- 3-COLORABLE (Exercise 26)
- SUBSET-SUM
- KNAPSACK

The P vs. NP problem

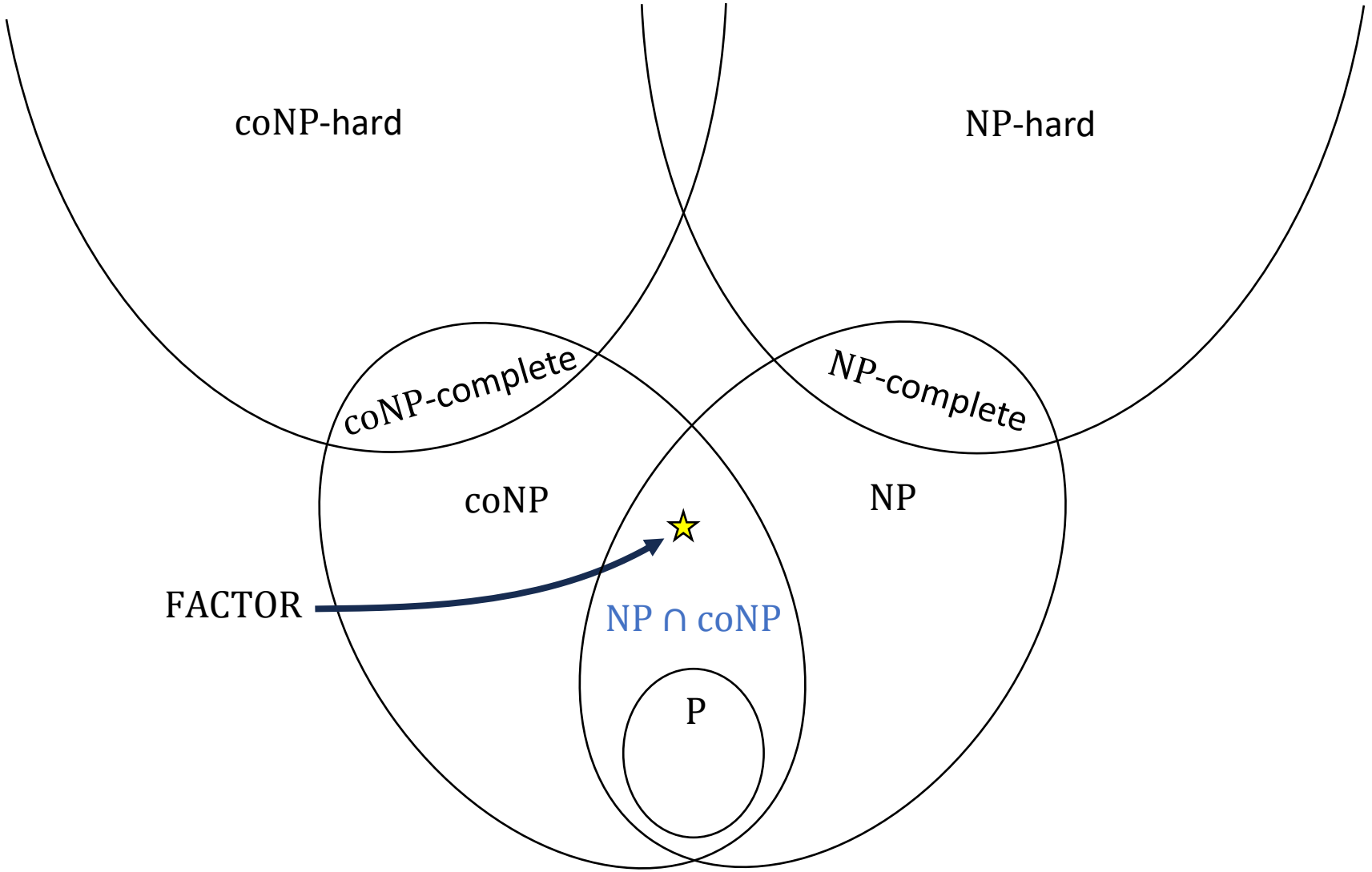
- We conjecture that $\mathrm{P} \neq \mathrm{NP}$: Solving and verifying are different
- A proof that $\mathrm{P} = \mathrm{NP}$ would change the world
  - *Assuming the proof gives us truly practical algorithms
- We could solve countless important problems in polynomial time 🥲
- Hackers could break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

The complexity class $\mathbf{NP \cap coNP}$

- $\mathrm{FACTOR} = \{\langle K, R\rangle : K \text{ has a prime factor } p \leq R\}$

Theorem: $\mathrm{FACTOR} \in \mathrm{NP} \cap \mathrm{coNP}$

- The prime factorization of K can be used to certify that K does have a small factor or to certify that K does not have a small factor
- Consequence: FACTOR is probably not NP-complete


coNP-hard
NP-hard
coNP-complete
NP-complete
coNP
NP
FACTOR
NP ∩ coNP
P

Approximation algorithms

- We can find a “99% optimal” solution to a given instance of the knapsack problem in polynomial time 🙂
- If $\mathrm{P} \neq \mathrm{NP}$, then we cannot even find a “1% optimal” solution to a given instance of the clique problem in polynomial time 😟

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 June 1)

Three big lessons

1. We can study all possible algorithms simultaneously by developing mathematical models of computation!
2. Computation has severe unavoidable limitations!
3. Computer science is deep / profound / sublime, not merely useful!
   - What is the nature of the physical universe?
   - What is the nature of the human condition?

- **See you at office hours and the final exam!**