Great Ideas in Theoretical CS

Lecture 17:
P vs. NP

Anil Ada
Ariel Procaccia (this time)

Millennium Prize Problems

- Seven famous problems in math stated in 2000 by the Clay Foundation

- $1,000,000 prize for solving any of them

- One of the problems: **P** vs. **NP**

Millennium Prize Problems

If one is solved in the next few years, it'll probably be **P** vs. **NP**

Keith Devlin

If, in the year 3000, one of them is unsolved, it will be **P** vs. **NP**

Laszlo Lovasz

Millennium Prize Problems

- The **P** vs. **NP** problem is the only Millennium Prize problem that has the potential to change the world
- So what is it?

SUDOKU

8			4		6			7
						4		
	1					6	5	
5		9		3		7	8	
				7				
	4	8		2		1		3
	5	2					9	
		1						
3			9		2			5

$$3 \times 3 \times 3 \times 3$$

Sudoku

- Sudoku: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be filled?

- Naive decision algorithm: Check all possibilities, in time $O\left(n^{2n^4}\right)$

- Verifying a solution: $O(n^4)$

- For $n = 100$

    o Verifying a solution: 100M steps

    o Deciding YES/NO: Number with 400M digits!

SUDOKU

- Question: Is there a polynomial-time algorithm that can solve SUDOKU?

- This is equivalent to the **P** vs. **NP** problem!

Is this famous problem really about Sudoku?

P vs. NP

- Informal formulation of **P** vs. **NP**:
  - Let *L* be an algorithmic task
  - Suppose there is an efficient algorithm for verifying solutions to *L* ($L \in NP$)
  - Is there an efficient algorithm for finding solutions to *L*? ($L \in P$)

Sudoku is not just one instance of this problem; if the answer is "yes" for Sudoku, it is "yes" in general!

Efficiency

- Efficient = polynomial time
- Given a decision problem L, $x \in L$ means that x is a YES instance of L; $|x|$ is its size
- $\mathbf{P}$ = Decision problems L such that there exists a constant c and an algorithm A such that A runs in time $|x|^c$ and $A(x) =$ YES if and only if $x \in L$
- We saw last time that 2-Coloring is in $\mathbf{P}$

Verifying solutions

- In problems like Sudoku, verifying the solution can be done efficiently

- **NP** = Decision problems whose solutions can be verified in polynomial time in their input size

The N in **NP** stands
for "nondeterministic"

Carnegie Mellon University

NP: SEMI-FORMAL DEFINITION

- $L \in \mathbf{NP}$ if and only if there are constants c, d and an algorithm V called the verifier such that:

  - V takes two inputs, x and y, where $|y| \leq |x|^d$; x is called the instance and y is called the certificate

  - $V(x, y)$ runs in time $O(|x|^c)$

  - If $x \in L$, $\exists y$ such that $V(x, y) = \text{YES}$

  - If $x \notin L$, $\forall y, V(x, y) = \text{NO}$

EXAMPLES

- SUDOKU: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be completed?

- Input size: n

- Certificate: board filled with numbers

- Verifier: Check that each square, row, and column contain all numbers

Instance

Certificate

EXAMPLES

- HAMILTONIAN-CYCLE: Given a graph $G = (V, E)$, does it contain a Hamiltonian cycle?

- Input size: $n = |V|$

- Certificate: A permutation of the n vertices

- Verifier: Check that the permutation contains each vertex exactly once, and there is an edge between adjacent vertices

Instance

Certificate

Carnegie Mellon University
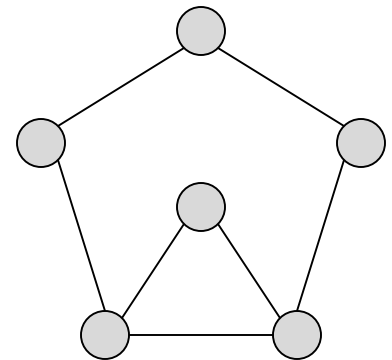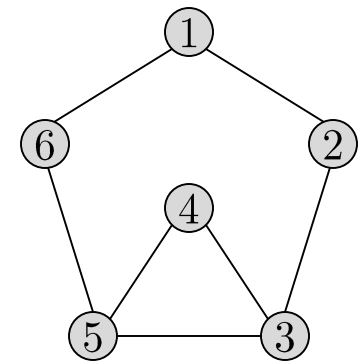
EXAMPLES

- INDEPENDENT-SET: Given a graph $G = (V, E)$ and $k \in \mathbb{N}$, does G contain an independent set of size k?

- Input size: $n = |V|$

- Certificate: k vertices

- Verifier: Check that there are no edges between pairs of vertices

Instance

$+\ 3$

Certificate
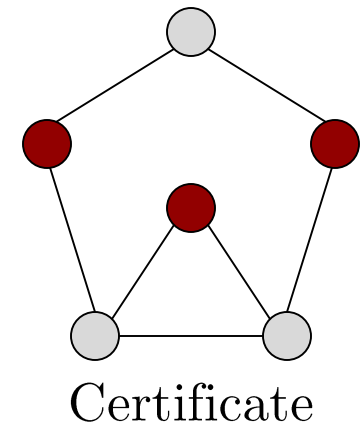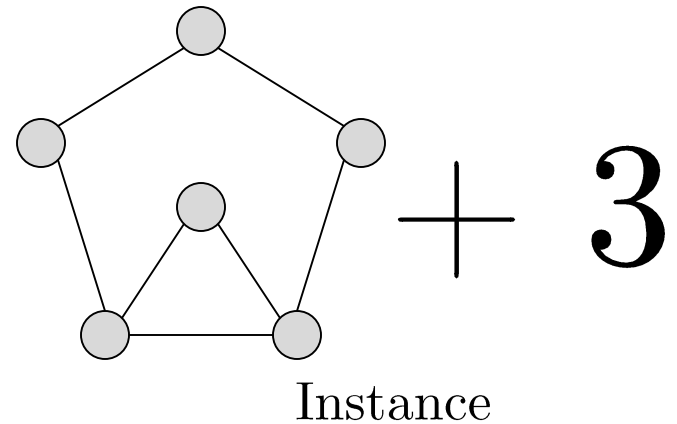
EXAMPLES

- Poll 1: Which of the following two problems is in **NP**?

  1. Given numbers $a_1, \ldots, a_n$ and $k \in \mathbb{N}$, is there a subset S such that $\sum_{i \in S} a_i = k$?

  2. Given a graph G and $k \in \mathbb{N}$, is the largest clique of size at most k?

  3. Both

  4. Neither

EXAMPLES

- Poll 2: Which of the following two problems is in **NP**?

  1. Given a graph G, does it not have a 2-coloring?

  2. Given a graph G, does it not have an Eulerian cycle?

  3. Both

  4. Neither

P vs. NP

- Theorem: $\mathbf{P} \subseteq \mathbf{NP}$

- Proof:
  - Suppose $L \in \mathbf{P}$
  - Let A be a poly-time algorithm that decides L
  - The verifier V takes as input the instance x and an empty certificate y
  - $V(x, y)$ outputs $A(x)$ ∎

P vs. NP

- We know that $\mathbf{P} \subseteq \mathbf{NP}$; does $\mathbf{P} = \mathbf{NP}$?

- If $\mathbf{P} = \mathbf{NP}$ then there would be an efficient algorithm for SUDOKU, 3-COLORING, CIRCUIT-SAT... Awesome!

- If $\mathbf{P} \neq \mathbf{NP}$ then there is some particular $L \in \mathbf{NP}$ such that $L \notin \mathbf{P}$; but maybe it is an obscure L?

THE COOK-LEVIN THEOREM

- Theorem (Cook 71, Levin 73):
  $\mathbf{P} = \mathbf{NP}$ if and only if
  CIRCUIT-SAT $\in \mathbf{P}$

- In particular, if $\mathbf{P} \neq \mathbf{NP}$ then
  CIRCUIT-SAT $\notin \mathbf{P}$

- In a sense, CIRCUIT-SAT is the
  hardest problem in $\mathbf{NP}$

15251 Fall 2017: Lecture 17

Carnegie Mellon University 19

Reductions, revisited

- L has a polynomial-time reduction to L', denoted $L \leq_T^P L'$, if and only if it is possible to solve L in polynomial time using a polynomial-time algorithm for L'

- If $L \leq_T^P L'$ then:
    1. $L' \in \mathbf{P} \Rightarrow L \in \mathbf{P}$
    2. $L \notin \mathbf{P} \Rightarrow L' \notin \mathbf{P}$

THE HARDEST PROBLEM(S)

- If CIRCUIT-SAT is in **P** then all of **NP** is in **P**

- Last lecture: there is a poly-time reduction from CIRCUIT-SAT to 3-COLORING

  ⇒ If 3-COLORING is in **P** then CIRCUIT-SAT is in **P**, and hence all of **NP** is in **P**

  ⇒ **P** = **NP** if and only if 3-COLORING ∈ **P**

THE HARDEST PROBLEM(S)

- Theorem (Yato-Seta 2002): There is a poly-time reduction from 3-COLORING to SUDOKU

  $\Rightarrow$ If SUDOKU is in $\mathbf{P}$ then 3-COLORING is in $\mathbf{P}$, and hence all of $\mathbf{NP}$ is in $\mathbf{P}$

  $\Rightarrow$ $\mathbf{P} = \mathbf{NP}$ if and only if SUDOKU $\in \mathbf{P}$

COOK-LEVIN, REVISITED

- Actual statement of Cook-Levin: Let $L \in$ **NP**, then there is a poly-time reduction from L to CIRCUIT-SAT

CIRCUIT-SAT $\in$ **P** $\Rightarrow$ **P** $=$ **NP**
P $=$ **NP** $\Rightarrow$ CIRCUIT-SAT $\in$ **P**

NP-COMPLETENESS

- *L* is **NP**-hard if every problem in **NP** has a polynomial time reduction to *L*

- *L* is **NP**-complete if $L \in$ **NP** and *L* is **NP**-hard

- To show that a problem is **NP**-complete:
  - Show that it is in **NP**
  - Show that a known **NP**-hard problem reduces to it

NP-COMPLETENESS

All problems in **NP**

↓

CIRCUIT-SAT

3-COLORING   SUDOKU   INDEP.-SET

↓

CLIQUE

Carnegie Mellon University
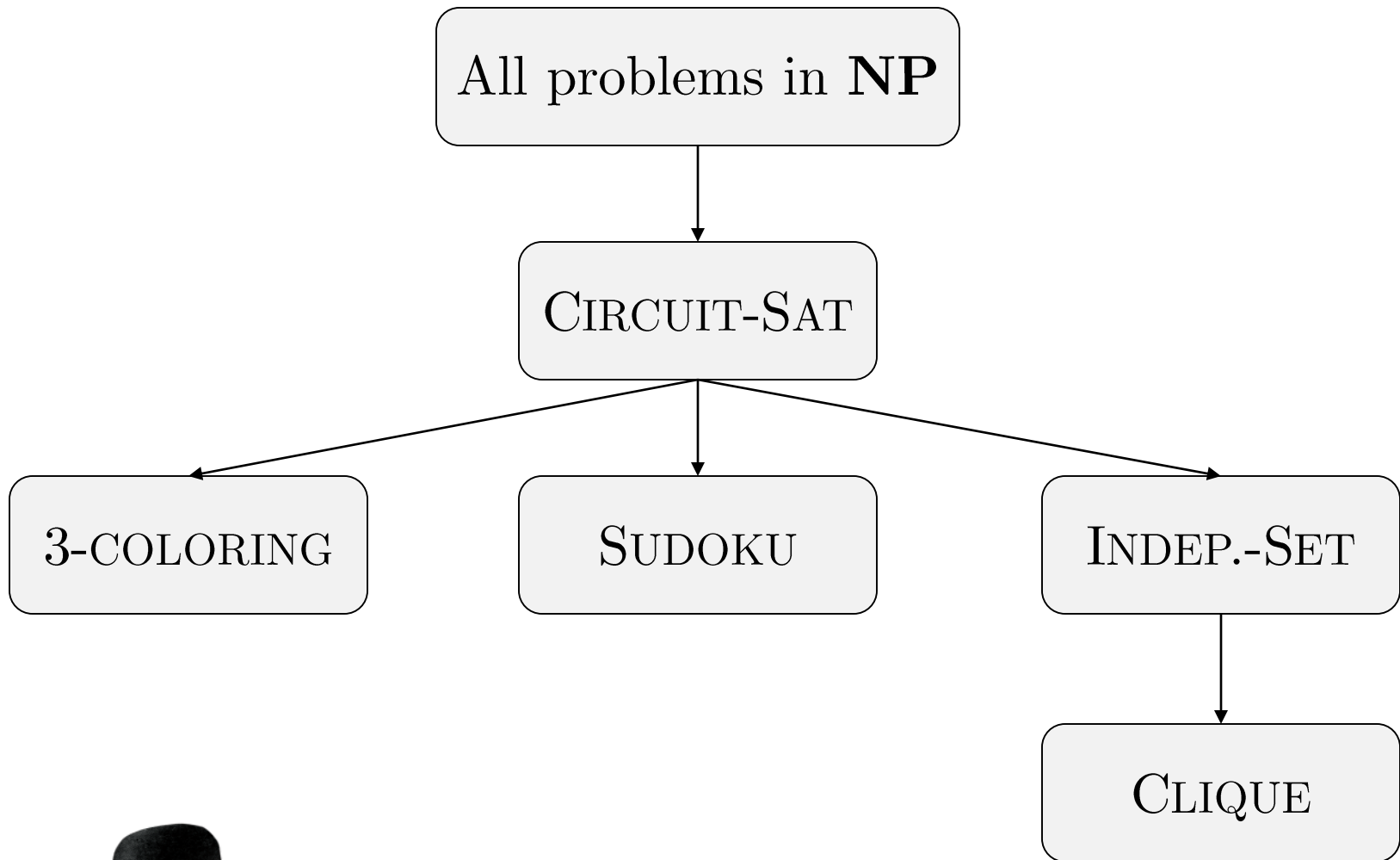
NP-COMPLETE PROBLEMS

- Tens of thousands of problems are known to be **NP**-complete

- If even one of them has a poly-time algorithm then all of them are in **P**

NP-COMPLETE PROBLEMS

- CYCLE-COVER: Given a directed graph and $L \in \mathbb{N}$, is there a collection of disjoint cycles of length $\leq L$ that covers $\geq k$ vertices?

- Theorem: CYCLE-COVER is **NP**-complete

- Relevant to kidney exchange

Carnegie Mellon University

NP-COMPLETE PROBLEMS

NP-COMPLETE PROBLEMS

Classic Nintendo Games are (NP-)Hard

Greg Aloupis[*]   Erik D. Demaine[†]   Alan Guo[†‡]

March 9, 2012

Abstract

We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon. Our results apply to Super Mario Bros. 1, 3, Lost Levels, and Super Mario World; Donkey Kong Country 1–3; all Legend of Zelda games except Zelda II: The Adventure of Link; all Metroid games; and all Pokémon role-playing games. For Mario and Donkey Kong, we show NP-completeness. In addition, we observe that several games in the Zelda series are PSPACE-complete.

1 Introduction

A series of recent papers have analyzed the computational complexity of playing many different video games [1, 4, 5], yet the most well-known Nintendo games of our youth have yet to be included among these results. In this paper, we consider some of the best-known Nintendo games of all time—Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon—and prove that it is NP-hard to play generalized versions of many games in these series. In particular, our results for Mario apply to the NES games Super Mario Bros., Super Mario Bros.: The Lost Levels, Super Mario Bros. 3, and Super Mario World (developed by Nintendo); our results for Donkey Kong apply to the SNES games Donkey Kong Country 1–3 (developed by Rare Ltd.); our results for Legend of Zelda apply to all Legend of Zelda games (developed by Nintendo) except the side-scrolling Zelda II: The Adventure of Link; our results for Metroid apply to all Metroid games (developed by Nintendo); and our results for Pokémon apply to all Pokémon role-playing games (developed by Game Freak and Creatures Inc.).[1]

Our results are motivated in particular by tool-assisted speed runs for video games. A *speed run* of a game is a play through that aims to achieve a fast completion time, usually performed by a human. In a *tool-assisted* speed run, the player uses special tools, such as emulators, to allow them to slow the game down to a frame-by-frame rate in order to achieve superhuman reflexes and timing. In some sense, tool assistance is not cheating because the rules of the game are still obeyed. The resulting speed runs are impressive to watch, as the paths taken by a tool-assisted player are

1

---

gadget includes an item block containing a Super Mushroom which makes Mario into Super Mario (see Figure 2). The Super Mushroom serves two purposes: first, Super Mario is 2 tiles tall, which prevents him from fitting into narrow horizontal corridors, a property essential to our other gadgets; second, Super Mario is able to destroy bricks whereas normal Mario cannot. In order to force the player to take the Super Mushroom in the beginning, we block off the Finish gadget with bricks (see Figure 3).
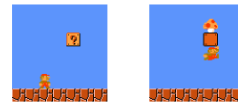
Figure 2: Left: Start gadget for Mario. Right: The item block contains a Super Mushroom
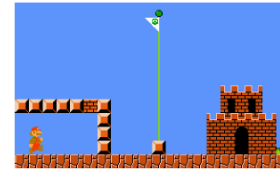
Figure 3: Finish gadget for Mario

Next, we implement the Variable gadget, illustrated in Figure 4. There are two entrances, one from each literal of the previous variable (if the variable is x_i, the two entrances come from x_{i-1} and $\neg x_{i-1}$). Once Mario falls down, he cannot jump back onto the ledges at the top, so Mario cannot go back to a previous variable. In particular, Mario cannot go back to the negation of the literal he chose. To choose which value to assign to the variable, Mario may fall down either the left passage or the right.

Now we present the Clause gadget, illustrated in Figure 5. The three entrances at the top come from the three literals that appear in the clause. To unlock the clause, Mario jumps onto a Red Koopa, kicks its shell down, which bounces and breaks all the bricks in the corridor at the bottom, opening the path for later checking. Note that falling down is no use because Super Mario cannot fit into the narrow corridor unless he gets hurt by the Koopa, in which case he will not be able to reach the goal. There is not enough space for Mario to run and crouch-slide into the corridor. The gap at the bottom of the wide corridor is so the Koopa Shell does not unlock other clauses.

Finally, we implement the Crossover gadget, illustrated in Figure 6. There are four entrances/exits: top left, top right, bottom left, and bottom right. The Crossover gadget is designed so that, if Mario
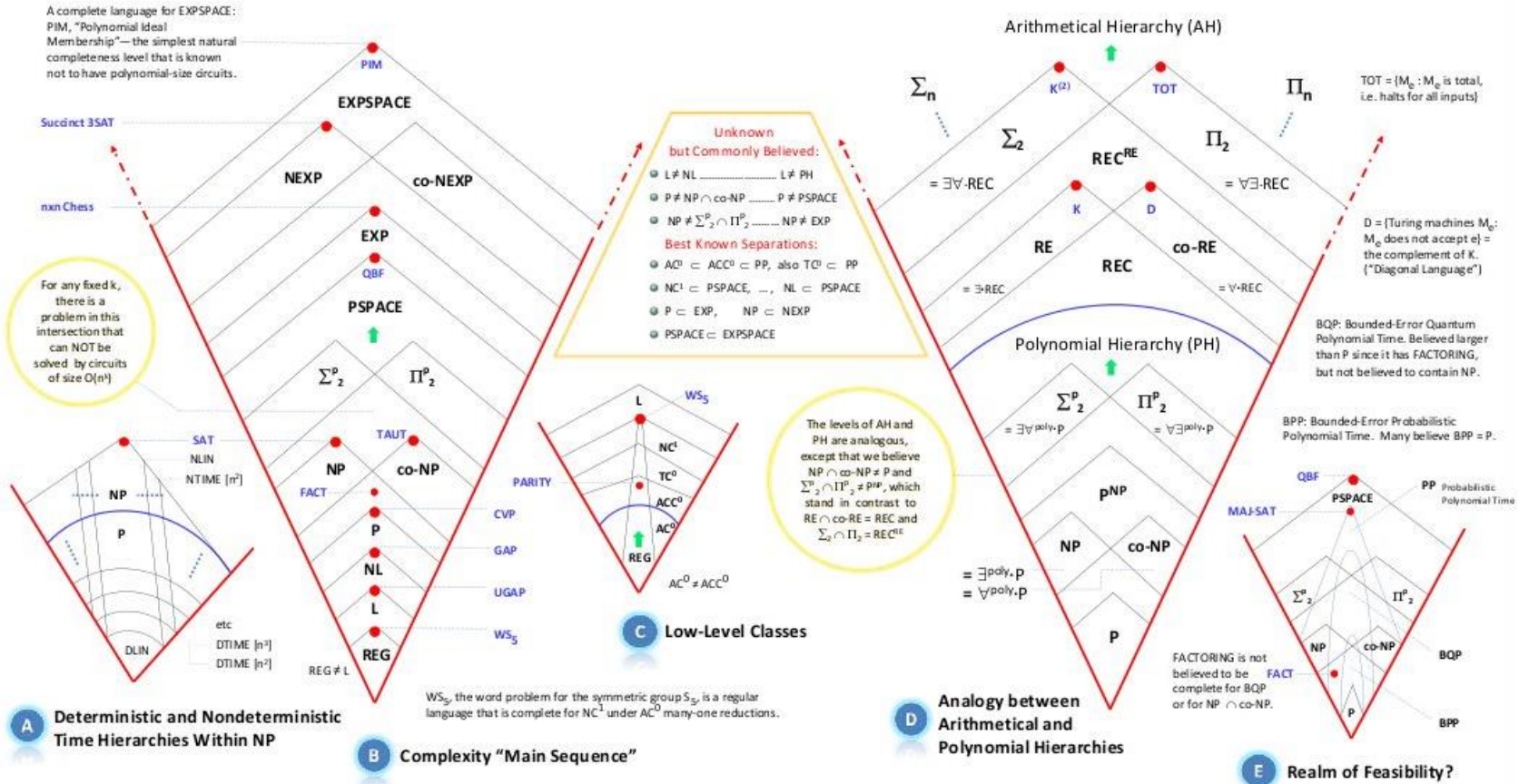
5

P vs. NP

- So what do the experts think about the P vs. NP problem?

- Two polls from 2002 and 2012
  - 100 respondents in 2002
  - 152 respondents in 2012

Year	P≠NP	P=NP	Ind.	DC	BM
2002	61%	9%	4%	1%	22%
2012	83%	9%	3%	3%	1%

THE TWO POSSIBLE WORLDS

NP hard

NP-c

NP

P

NP hard

P=**NP**=**NP**-c
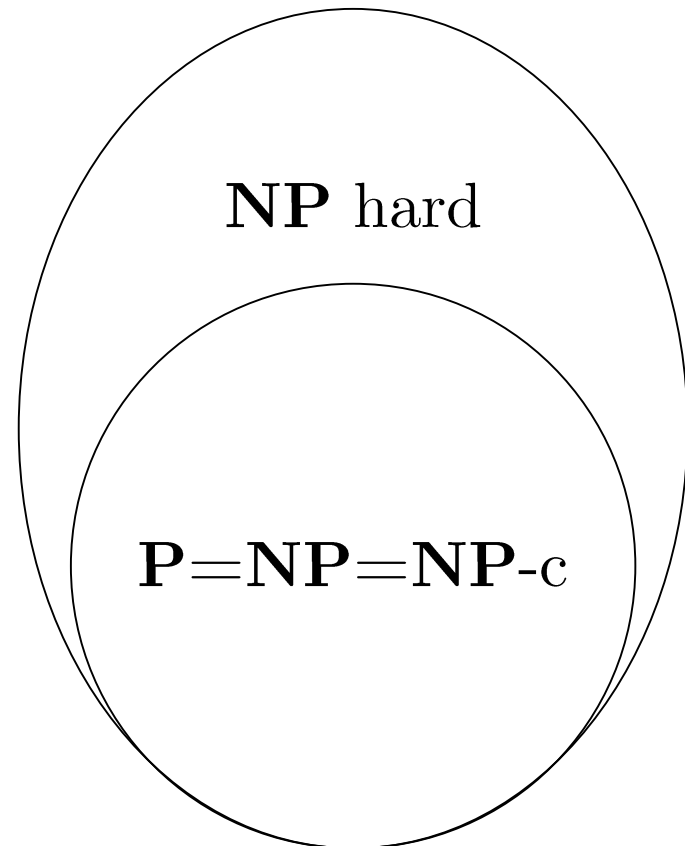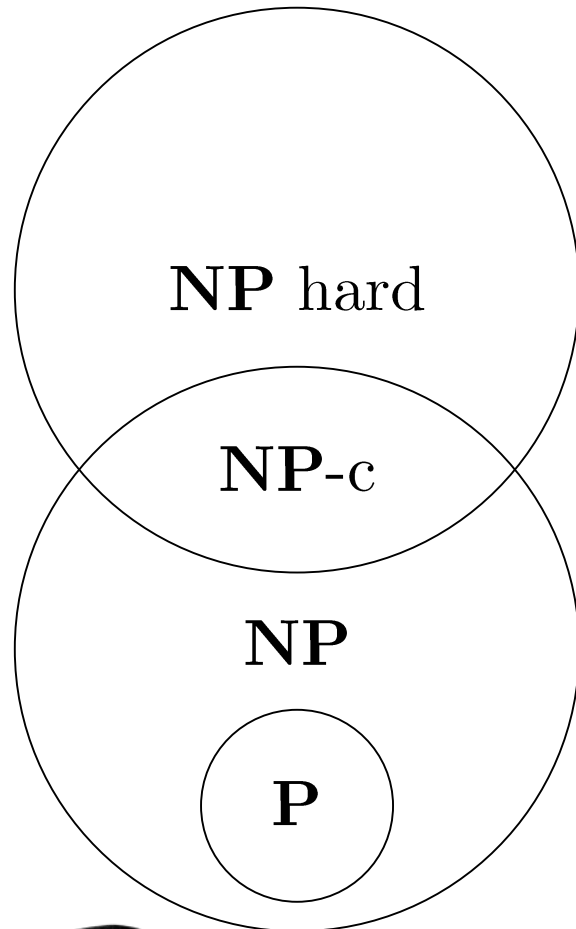
COMPLEXITY UNIVERSE

SUMMARY

- Terminology / facts
    - $\mathbf{P}$ and $\mathbf{NP}$
    - Cook-Levin Theorem
    - $\mathbf{NP}$-complete

- Principles:
    - Proving that problems are in $\mathbf{P}$, $\mathbf{NP}$, or $\mathbf{NP}$-complete

Carnegie Mellon University 33