

BU CS 332 – Theory of Computation

<https://forms.gle/bwihnC5PtYTVGA6bA>



Lecture 22:

- NP-completeness

Reading:

Sipser Ch 7.4-7.5

Mark Bun

April 23, 2025

Last time: Two equivalent definitions of NP

1) NP is the class of languages decidable in polynomial time on a nondeterministic TM

$$\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)$$

2) A **polynomial-time verifier** for a language L is a **deterministic** $\text{poly}(|w|)$ -time algorithm V such that

$w \in L \iff$ there **exists** a certificate c
such that $V(\langle w, c \rangle)$ accepts

Theorem: A language $L \in \text{NP}$ iff there is a polynomial-time verifier for L

Examples of NP languages

- Hamiltonian path

Given a graph G and vertices s, t , does G contain a Hamiltonian path from s to t ?

- Clique

Given a graph G and natural number k , does G contain a clique of size k ?

- Subset Sum

Given a list of natural numbers x_1, \dots, x_k , is there a subset of the numbers x_1, \dots, x_k that sum up to exactly t ?

- Boolean satisfiability (SAT)

Given a Boolean formula, is there a satisfying assignment?

- Vertex Cover

Given a graph G and natural number k , does G contain a vertex cover of size k ?

- Traveling Salesperson

Examples of NP languages: SAT

“Is there an assignment to the variables in a logical formula that make it evaluate to true?”

- **Boolean variable:** Variable that can take on the value true/false (encoded as 0/1) x_1 x_2
- **Boolean operations:** \wedge (AND), \vee (OR), \neg (NOT)
- **Boolean formula:** Expression made of Boolean variables and operations. **Ex:** $\varphi(x_1, x_2, x_3) = (x_1 \vee \overline{x_2}) \wedge x_3$
- An **assignment** of 0s and 1s to the variables **satisfies** a formula φ if it makes the formula evaluate to 1
$$\begin{array}{l} x_1 = 1 \\ x_2 = 1 \end{array} \quad x_3 = 1 \quad \varphi(1, 1, 1) = (1 \vee 0) \wedge 1 = 1 \wedge 1 = 1$$
- A formula φ is **satisfiable** if there exists an assignment that satisfies it
 \Rightarrow **satisfiable**

Examples of NP languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ YES, since $x_1=1, x_2=1, x_3=1$ is a satisfying assignment Satisfiable?

Ex: $(x_1 \vee x_2) \wedge \overline{x_1} \wedge \overline{x_2}$ Satisfiable?
Not Satisfiable

$SAT = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable formula} \}$

Claim: $SAT \in NP$

Proof 1 via NTM:

On input $\langle \varphi \rangle$ on variables x_1, \dots, x_m

1. Nondeterministically guess assignment $c \in \{0,1\}^m$

2. Evaluate $\varphi(c_1, \dots, c_m)$.

If $= 1$, accept
If $= 0$, reject.

Proof 2 via verifier:

Certificate $c \in \{0,1\}^m$

Verifier V :

On input $\langle \varphi; c \rangle$:

1. Evaluate $\varphi(c_1, \dots, c_m)$

If $= 1$, accept.

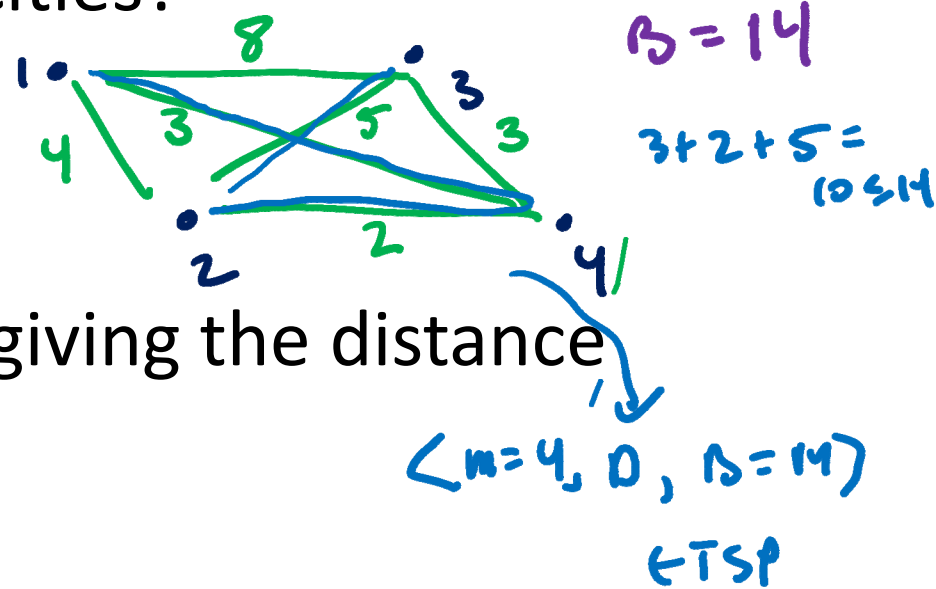
If $= 0$, reject.

Examples of NP languages: Traveling Salesperson

“Given a list of cities and distances between them, is there a ‘short’ tour of all of the cities?”

More precisely: Given

- A number of cities m
- A function $D: \{1, \dots, m\}^2 \rightarrow \mathbb{N}$ giving the distance between each pair of cities
- A distance bound B



$TSP = \{ \langle m, D, B \rangle \mid \exists \text{ a tour visiting every city with length } \leq B \}$

NTM:

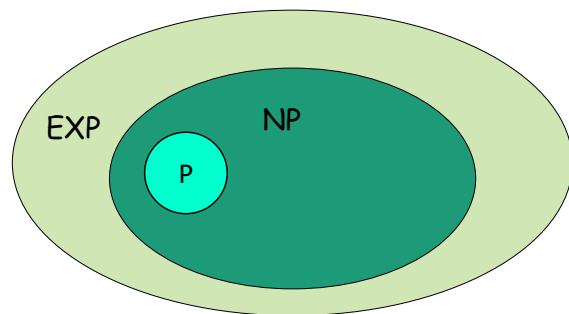
1. Nondeterministically guess sequence of cities
2. Check every city visited, and tour length $\leq B$.

P vs. NP

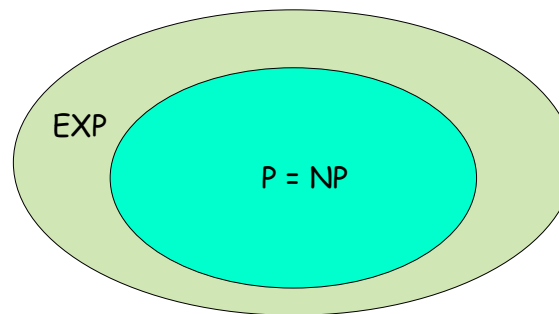
Question: Does $P = NP$?

Philosophically: Can every problem with an efficiently **verifiable** solution also be **solved** efficiently?

A central problem in mathematics and computer science



If $P \neq NP$



If $P = NP$

Millennium Problems

Yang-Mills and Mass Gap

Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

Riemann Hypothesis

The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part $1/2$.

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Navier-Stokes Equation

This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certainty, but also understanding.

Hodge Conjecture

The answer to this conjecture determines how much of the topology of the solution set of a system of algebraic equations can be defined in terms of further algebraic equations. The Hodge conjecture is known in certain special cases, e.g., when the solution set has dimension less than four. But in dimension four it is unknown.

Poincaré Conjecture

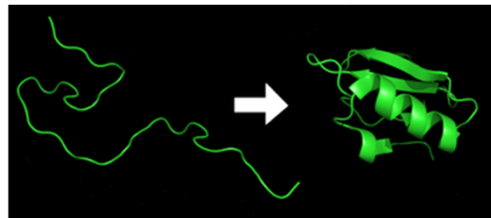
In 1904 the French mathematician Henri Poincaré asked if the three dimensional sphere is characterized as the unique simply connected three manifold. This question, the Poincaré conjecture, was a special case of Thurston's geometrization conjecture. Perelman's proof tells us that every three manifold is built from a set of standard pieces, each with one of eight well-understood geometries.

Birch and Swinnerton-Dyer Conjecture

Supported by much experimental evidence, this conjecture relates the number of points on an elliptic curve mod p to the rank of the group of rational points. Elliptic curves, defined by cubic equations in two variables, are fundamental mathematical objects that arise in many areas: Wiles' proof of the Fermat Conjecture, factorization of numbers into primes, and cryptography, to name three.

In a world where $P = NP$:

- Many important **decision** problems can be solved in polynomial time (*HAMPATH*, *SAT*, *TSP*, etc.)
- Many **search** problems can be solved in polynomial time (e.g., given a natural number, ***find*** a prime factorization)
- Many **optimization** problems can be solved in polynomial time (e.g., find the lowest energy conformation of a protein)



In a world where $P = NP$:

- Secure cryptography (as we know it) becomes impossible
An NP search problem: Given a ciphertext c , find a plaintext m and encryption key k that would encrypt to c
- AI / machine learning become easy: Identifying a consistent classification rule is an NP search problem
- Finding mathematical proofs becomes easy: NP search problem: Given a mathematical statement S and length bound k , is there a proof of S with length at most k ?

General consensus: $P \neq NP$

NP-Completeness

Understanding the P vs. NP question

Most believe $P \neq NP$, but we are very far from proving it

Question 1: How can studying specific computational problems help us get a handle on resolving P vs. NP?

Question 2: What would $P \neq NP$ allow us to conclude about specific problems we care about?

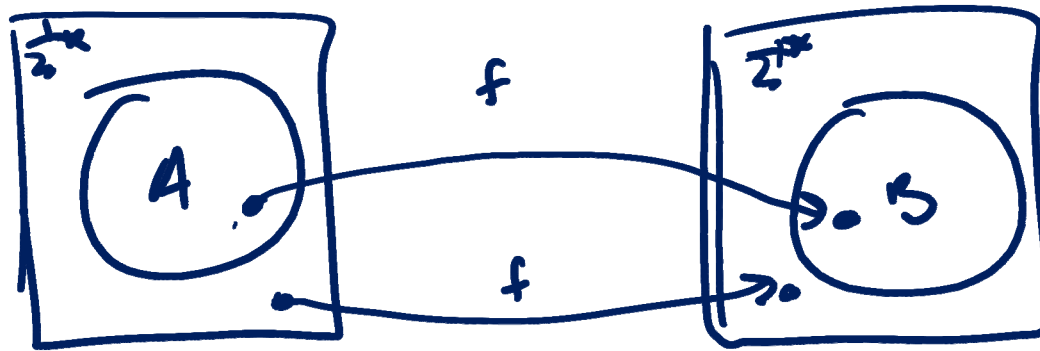
Idea: Identify the “hardest” problems in NP

Languages $L \in NP$ such that $L \in P$ iff $P = NP$

Recall: Mapping reducibility

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **computable** if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.



Definition:

Language A is **mapping reducible** to language B , written

$$A \leq_m B$$

if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$

Polynomial-time reducibility

c.f. $A \leq_m B$
for mapping reductions

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **polynomial-time computable** if there is a **polynomial-time** TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.

Definition:

Language A is **polynomial-time reducible** to language B , written

$$A \leq_p B$$

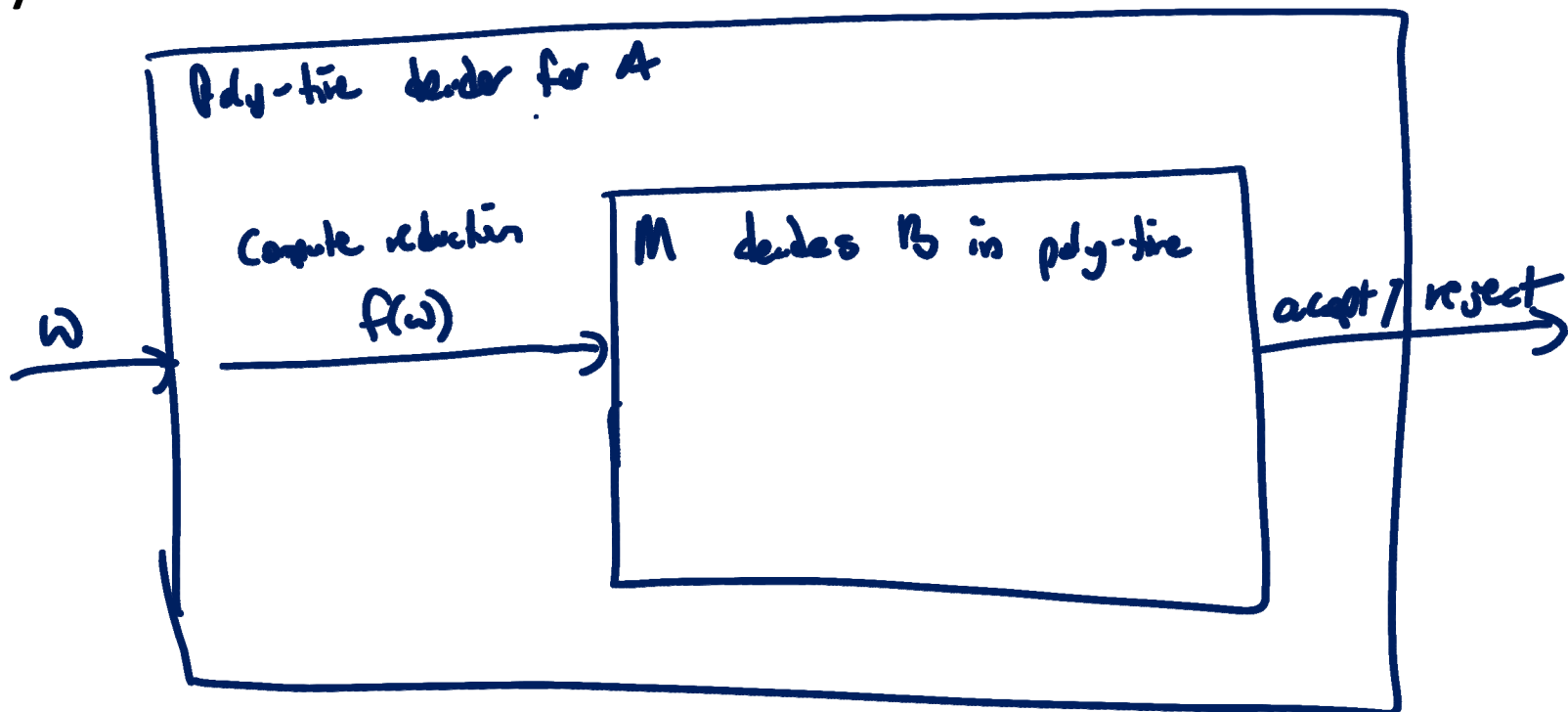
if there is a **polynomial-time** computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$

Implications of poly-time reducibility

cf. If $A \leq_m B$ and B decidable, then A is decidable

Theorem: If $A \leq_p B$ and $B \in P$, then $A \in P$

Proof: Let M decide B in poly time, and let f be a poly-time reduction from A to B . The following TM decides A in poly time:



Suppose B is decided by poly-time TM M

Mapping reduction f from A to B is poly-time computable

The following TM decides A in poly-time:

TM V :

On input w :

- 1) compute $f(w)$.
- 2) Run M on input $f(w)$
- 3) If accept, accept.
if rejects, reject.

Runtime analysis:

1) Computing $f(w)$ takes
 $\text{poly}(|w|)$ time
and moreover,
 $|f(w)| \leq \text{poly}(|w|)$

2) Running M on poly-time
input $f(w)$ takes
 $\text{poly}(|f(w)|)$
 $= \text{poly}(\text{poly}(|w|))$
 $= \text{poly}(|w|)$ time.

\Rightarrow total time is $\text{poly}(|w|)$

Correctness:

If $w \in A$, then $f(w) \in B$ [correctness of reduction]
 $\Rightarrow M$ accepts [correctness of M]
 $\Rightarrow V$ accepts \checkmark

If $w \notin A$, then $f(w) \notin B \Rightarrow M$ rejects $\Rightarrow V$ rejects \checkmark

Is NP closed under poly-time reductions?

If $A \leq_p B$ and B is in NP, does that mean A is also in NP?

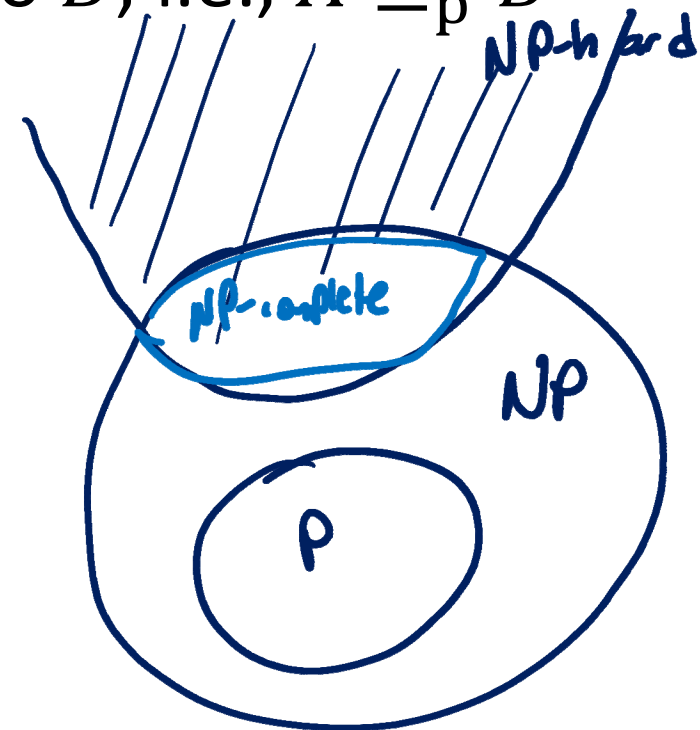
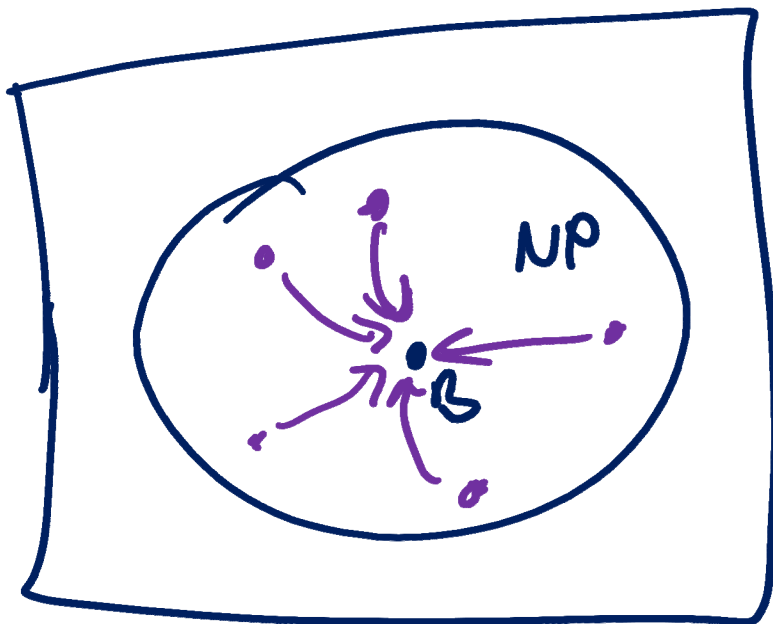


- a) Yes, the same proof works using NTMs instead of TMs
- b) No, because the new machine is an NTM instead of a deterministic TM
- c) No, because the new NTM may not run in polynomial time
- d) No, because the new NTM may accept some inputs it should reject
- e) No, because the new NTM may reject some inputs it should accept

NP-completeness

Definition: A language B is NP-complete if

- 1) $B \in \text{NP}$, and
- 2) B is NP-hard: **Every** language $A \in \text{NP}$ is poly-time reducible to B , i.e., $A \leq_p B$



Implications of NP-completeness

Theorem: Suppose B is NP-complete.

Then $B \in P$ iff $P = NP$

(consequence of B NP-complete)

Proof:

\Leftarrow

$P = NP \Rightarrow B \in NP = P$

\Rightarrow

Suppose B NP-complete and $B \in P$

Let $A \in NP$ be arbitrary

$A \leq_P B$ {since B NP-hard}

Then $B \in P \Rightarrow A \in P$

$\Rightarrow NP \subseteq P \Rightarrow NP = P$

Implications of NP-completeness

Theorem: Suppose B is NP-complete.

Then $B \in P$ iff $P = NP$

Consequences of B being NP-complete:

- 1) If you want to prove $P = NP$, you just have to prove $B \in P$
- 2) If you want to prove $P \neq NP$, a good candidate is to try to show that $B \notin P$
- 3) If you believe $P \neq NP$, then you also believe $B \notin P$

Cook-Levin Theorem and NP-Complete Problems

Do NP-complete problems exist?

Theorem: $TMSAT = \{\langle N, w, 1^t \rangle \mid$
NTM N accepts input w within t steps $\}$ is NP-complete

Proof sketch: 1) $TMSAT \in NP$: *hard et. guess sequence of t transitions in NTM N , check that they lead N to accept w .*

2) $TMSAT$ is NP-hard: Let $L \in NP$ be decided by NTM N running in time $\underline{T(n)}$. The following poly-time TM shows $L \leq_p TMSAT$: *same polynomial*

“On input w (an instance of L):

Output $\langle N, w, 1^{T(|w|)} \rangle$.”

Correctness:

$w \in L \Rightarrow N$ accepts w w/in $T(|w|)$

$\Rightarrow \langle N, w, 1^{\overset{\text{steps}}{T(|w|)}} \rangle \in TMSAT$

$w \notin L \Rightarrow N$ does not accept w

$\Rightarrow \langle N, w, 1^{T(|w|)} \rangle \notin TMSAT.$

Cook-Levin Theorem (Sipser Ch. 7.4)

Theorem: *SAT* (Boolean satisfiability) is NP-complete

“Proof”: Already know *SAT* \in NP. (Much) harder direction:
Need to show every problem in NP reduces to *SAT*



Stephen A. Cook (1971)



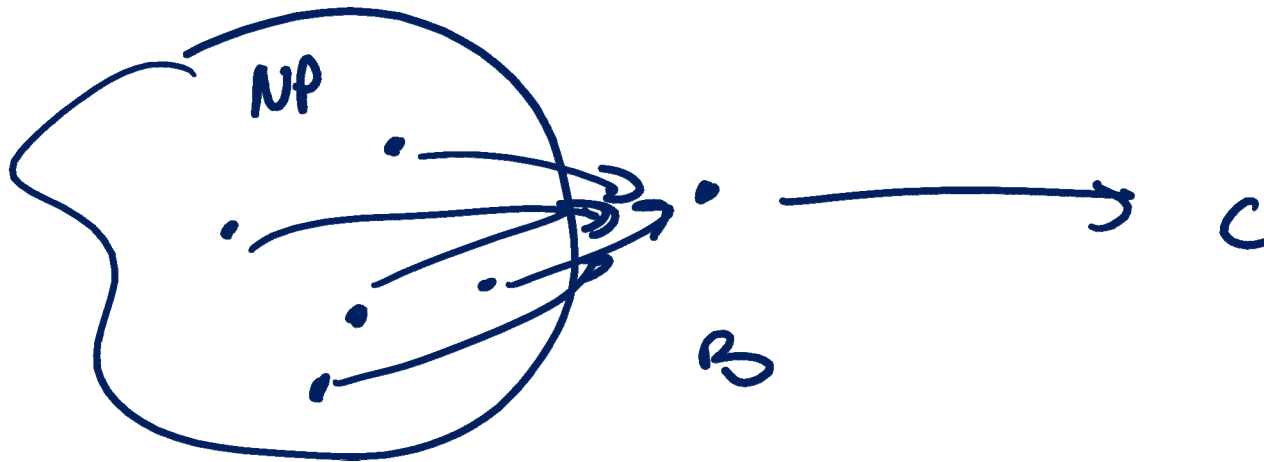
Leonid Levin (1973)

New NP-complete problems from old

Lemma: If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$
(poly-time reducibility is transitive)

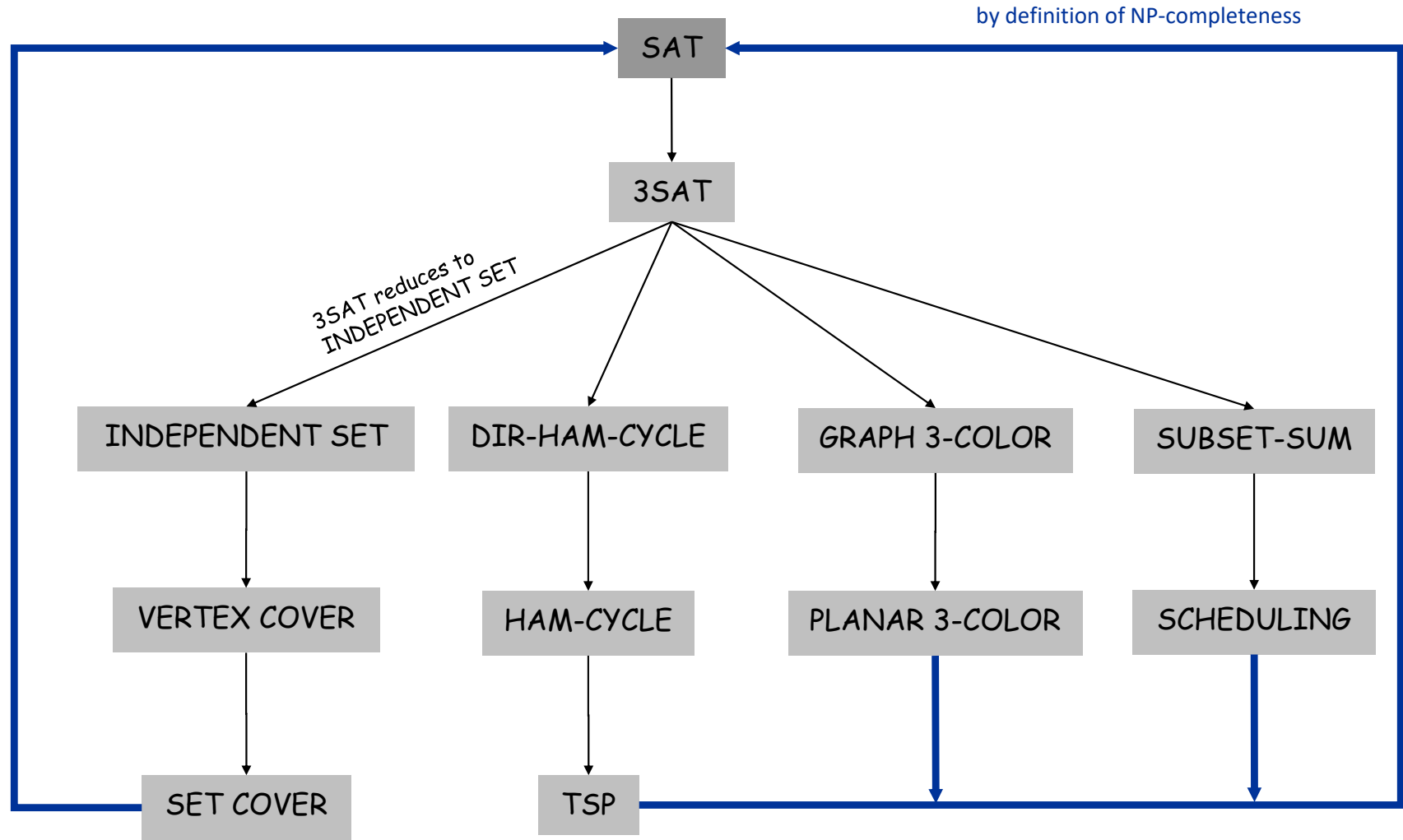
Theorem: If $B \leq_p C$ for some NP-hard language B , then C is also NP-hard

Corollary: If $C \in \text{NP}$ and $B \leq_p C$ for some NP-complete language B , then C is also NP-complete



New NP-complete problems from old

All problems below are NP-complete and hence poly-time reduce to one another!



3SAT (3-CNF Satisfiability)



Definitions:

- A **literal** either a variable or its negation $x_5, \overline{x_7}$
- A **clause** is a disjunction (OR) of literals **Ex.** $x_5 \vee \overline{x_7} \vee x_2$
- A **3-CNF** is a conjunction (AND) of clauses where each clause contains exactly 3 literals

Ex. $C_1 \wedge C_2 \wedge \dots \wedge C_m =$

$$(x_5 \vee \overline{x_7} \vee x_2) \wedge (\overline{x_3} \vee x_4 \vee x_1) \wedge \dots \wedge (x_1 \vee x_1 \vee x_1)$$

$$3SAT = \{\langle \varphi \rangle \mid \varphi \text{ is a satisfiable 3 - CNF}\}$$

3SAT is NP-complete

Theorem: 3SAT is NP-complete

Proof idea: 1) 3SAT is in NP (why?)

2) Show that $SAT \leq_p 3SAT$



Your classmate suggests the following reduction from SAT to 3SAT: “On input φ , a 3-CNF formula (an instance of 3SAT), output φ , which is already an instance of SAT .” Is this reduction correct?

- a) Yes, this is a poly-time reduction from SAT to 3SAT
- b) No, because φ is not an instance of the SAT problem
- c) No, the reduction does not run in poly time
- d) No, this is a reduction from 3SAT to SAT ; it goes in the wrong direction

3SAT is NP-complete

Theorem: 3SAT is NP-complete

Proof idea: 1) 3SAT is in NP (why?)

2) Show that $SAT \leq_p 3SAT$

Idea of reduction: Give a poly-time algorithm converting an arbitrary formula φ into a 3CNF ψ such that φ is satisfiable iff ψ is satisfiable

Illustration of conversion from φ to ψ

Some general reduction strategies

- Reduction by simple equivalence

Ex. $IND - SET \leq_p VERTEX - COVER$

$VERTEX - COVER \leq_p IND - SET$

- Reduction from special case to general case

Ex. $VERTEX - COVER \leq_p SET - COVER$

$3SAT \leq_p SAT$

- “Gadget” reductions

Ex. $SAT \leq_p 3SAT$

$3SAT \leq_p IND - SET$