



Complexity IBC028, Lecture 4

H. Geuvers

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: spring 2026





Outline

Decision Problems

Complexity classes **P** and **NP**

NP-hard and **NP**-complete





Many algorithmic problems are decision problems

DEFINITION

A **Decision Problem** is the question whether some input i satisfies a property $Q(i)$. Its solution is a **yes/no** answer.

Some examples:

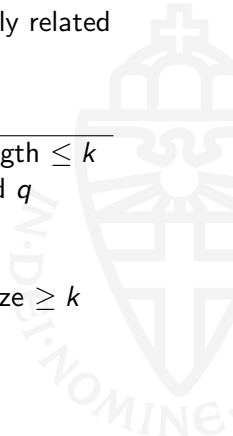
- Given a number n , is n prime?
- Given a graph G , does it have a Hamiltonian cycle?
(Recall: a Hamiltonian path visits every **node** exactly once.)
- Given a graph G , does it have an Euler cycle?
(Recall: an Euler path visits every **edge** exactly once.)
- Given a graph G and two points p and q in G , are p and q connected?
- Given a boolean formula φ , is φ satisfiable?



Optimization problems versus decision problems

For an optimization problem we (usually) have a closely related decision problem that is “equally difficult”.

Optimization problem	Decision Problem
What is the shortest path between points p and q in graph G ?	Is there a path of length $\leq k$ between points p and q in graph G ?
What is the largest clique in graph G ?	Is there a clique of size $\geq k$ in graph G ?





Encodings of decision problems

We can associate a decision problem Q with a **language**

$$L_Q \subseteq \{0, 1\}^*$$

$w \in L_Q \Leftrightarrow w$ is an encoding of a problem for which Q holds.

- The precise encoding is left implicit.
- We have the usual operations on languages: union, intersection, complement, concatenation, Kleene-star.

$$\text{Ham} \subseteq \{0, 1\}^*$$

:= collection of strings w that encode a graph G that has a Hamiltonian cycle

$$\text{Path} \subseteq \{0, 1\}^*$$

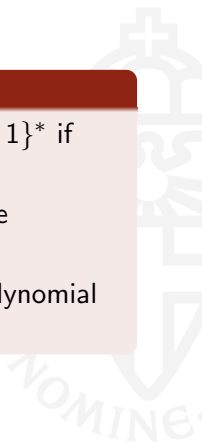
:= collection of strings w that encode $\langle G, p, q, k \rangle$, where G is a graph, $p, q \in G$, such that there is a path from p to q in G with at most k edges



Polynomial Time Decision Problems

DEFINITION

- The algorithm $f : \{0, 1\}^* \rightarrow \{0, 1\}$ **decides** $A \subseteq \{0, 1\}^*$ if $w \in A \iff f(w) = 1$.
- An **algorithm** f is **polynomial** if we have for its time complexity T that $T(n) = \mathcal{O}(n^k)$ for some k .
- A **decision problem** A is **polynomial** if **there is** a polynomial algorithm that decides A .





What encoding?

- Data types (graphs, formulas) need to be encoded as 01-strings.
- So: represent the set of graphs/formulas as subsets of $\{0, 1\}^*$.
- Precisely defining such an encoding is “high effort, little gain”.
- We assume an encoding to be “effective” ... (it is “easy” to determine that a string w is actually the code of an object we want to talk about: graph, formula, ...)
- We will leave the encodings implicit.

Encodings $\text{enc}_1, \text{enc}_2 : S \rightarrow \{0, 1\}^*$ are **polynomially related** if there are polynomial functions f and g such that $f(\text{enc}_1(s)) = \text{enc}_2(s)$ and $g(\text{enc}_2(s)) = \text{enc}_1(s)$ for all $s \in S$.

LEMMA For $\text{enc}_1, \text{enc}_2$ polynomially related, and $Q \subseteq S$:

$\text{enc}_1(Q)$ is polynomial if and only if $\text{enc}_2(Q)$ is polynomial.



Examples of Polynomial Time Decision Problems

- Given $n \in \mathbb{N}$, is n even?
- Given a formula φ , does φ contain a negation?
- Given a graph G and nodes p, q , is there a path from p to q ?
(Think of what you learned in Algorithms and Data Structures)
- Given a graph G , does it have an Euler path?
- Given a formula φ , is φ in conjunctive normal form?

A formula is in conjunctive normal form if it is a conjunction of disjunctions of atoms / negated atoms

Examples: $(x \vee \neg y) \wedge (x \vee y)$, $\neg x \vee \neg y$

Non-examples: $(x \wedge y) \rightarrow z$, $(x \wedge y) \vee (x \wedge \neg y)$



Closure operations for Polynomial Decision Problems

A problem is a subset of $\{0, 1\}^*$. Recall

- $x \in A \cup B$ if and only if $x \in A$ or $x \in B$
- $\overline{A} = \{w \in \{0, 1\}^* \mid w \notin A\}$
- $x \in AB$ if there are v, w with $v \in A$ and $w \in B$ and $x = vw$

LEMMA

Polynomial decision problems are closed under **complement**, **intersection**, **union**, **concatenation**.

Proof (two cases)

- If f decides $A \subseteq \{0, 1\}^*$ in polynomial time, then $g(w) := 1 - f(w)$ decides \overline{A} in polynomial time.
- If f_i decides A_i in polynomial time, then $g(w) := \text{sg}(f_1(w) + f_2(w))$ decides $A_1 \cup A_2$ in polynomial time.



The class **P**

DEFINITION

$$\mathbf{P} := \{A \subseteq \{0,1\}^* \mid \exists f, f \text{ polynomial, } f \text{ decides } A\}$$

- Path $\in \mathbf{P}$, EulerTour $\in \mathbf{P}$,
- Ham $\notin \mathbf{P}$ (...everyone thinks) (...everyone thinks)

For Ham, no polynomial algorithm is known (and it is believed that no polynomial algorithm exists).

But there is a notion of **certificate** that can be checked in polynomial time.

$$w \in \text{Ham} \iff w \text{ encodes a graph } G \quad \wedge \\ \exists y (y \text{ encodes a Hamiltonian cycle in } G).$$



Non-deterministic Polynomial Decision Problems

DEFINITION

- The algorithm f **verifies** $A \subseteq \{0, 1\}^*$ if $f : \{0, 1\}^* \rightarrow \{0, 1\}$ and

$$w \in A \iff \exists y \in \{0, 1\}^* (f(w, y) = 1).$$

- $A \subseteq \{0, 1\}^*$ is **non-deterministic polynomial** (NP) if there is a polynomial algorithm f that **verifies** A with **polynomial certificates**, that is

$$w \in A \iff \exists y \in \{0, 1\}^* (|y| \text{ polynomial in } |w| \wedge f(w, y) = 1).$$

- Ham is non-deterministic polynomial.
- NonPrime (determining whether a number is not prime) is non-deterministic polynomial.



P and NP

P :=

$$\{A \subseteq \{0, 1\}^* \mid \exists f, f \text{ polynomial}, w \in A \iff f(w) = 1\}$$

NP :=

$$\{A \subseteq \{0, 1\}^* \mid \exists f, f \text{ polynomial}, \\ w \in A \iff \exists y \in \{0, 1\}^* (|y| \text{ polynomial in } |w| \wedge f(w, y) = 1)\}$$

- **P** = the class of polynomial time decision problems.
- **NP** = the class of non-deterministic polynomial time decision problems.
- First property: **P** \subseteq **NP**.



Examples of NP Decision Problems

For each example: (1) What is the “certificate”? (2) Is it polynomial in the size of the input? (3) Can its correctness be decided in polynomial time?

- Given $n \in \mathbb{N}$, is n a composite number?
- Given a formula φ , is φ satisfiable?
- Given a graph G , does G have a Hamiltonian path?
- Given n items with weight w_i and value v_i . Can we pick items in such a way that the sum of values is at least V and the sum of the weights is at most W ? (*Knapsack problem*)
- Given an $n^2 \times n^2$ Sudoku, does it have a solution?



Closure operations for **NP** Decision Problems

LEMMA

NP decision problems are closed under **intersection**, **union**, **concatenation**.

Proof of $A, B \in \mathbf{NP}$ implies $A \cap B \in \mathbf{NP}$

Suppose f verifies A and g verifies B . Define

$$h(x, y) := \text{if } y = \langle y_1, y_2 \rangle \text{ then } f(x, y_1) \cdot g(x, y_2) \text{ else } 0.$$

We have

- h is polynomial.
- $\exists y$ (y polynomial in $|x| \wedge h(x, y) = 1$) if and only if $\exists y_1, y_2$ (y_1, y_2 polynomial in $|x| \wedge f(x, y_1) = g(x, y_2) = 1$) if and only if $x \in A \cap B$.

Open problem: $A \in \mathbf{NP} \stackrel{??}{\implies} \bar{A} \in \mathbf{NP}$ (...everyone thinks not)



What is the non-determinism in NP?

Polynomial algorithm for A = a deterministic Turing Machine M that halts on every input w in a number of steps polynomial in $|w|$ such that $w \in A$ iff $M(w)$ halts in q_f .

Non-deterministic polynomial algorithm for A = a **non-deterministic** Turing Machine M that halts on every input w in a number of steps polynomial in $|w|$ such that $w \in A$ iff $M(w)$ **has a computation** that halts in q_f .

A non-deterministic TM can be turned into a deterministic TM by making choices. The “certificate” is the successful choice from the list of possible choices.



Polynomial Reducibility

DEFINITION

A_1 (polynomially) **reduces to** A_2 , notation $A_1 \leq_P A_2$ if there is a polynomial function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that

$$x \in A_1 \iff f(x) \in A_2$$

LEMMA

- \leq_P is transitive: if $A \leq_P B$ and $B \leq_P C$ then $A \leq_P C$.
- If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
- If $A \leq_P B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.



NP-hard and NP-complete

DEFINITION

- A is called **NP-hard** if

$$\forall A' \in \mathbf{NP} (A' \leq_P A).$$

That is: all **NP**-problems can be reduced to A.

- **NPH** := {A | A is **NP-hard**}.
- A is called **NP-complete** if $A \in \mathbf{NP}$ and A is **NP-hard**.
- **NPC** := $\mathbf{NP} \cap \mathbf{NPH}$.

THEOREM

If $A \in \mathbf{NPH}$ and $A \leq_P B$, then $B \in \mathbf{NPH}$.

Proof: Let $X \in \mathbf{NP}$ (TP: $X \leq B$.) Then $X \leq_P A$, and by $A \leq_P B$ we conclude $X \leq_P B$. □



NP-hard and NP-complete problems

How to prove that A is **NP**-complete?

- First prove that $A \in \mathbf{NP}$: give a polynomial algorithm and a polynomial certificate for each input.
- Pick a well-known $A' \in \mathbf{NPH}$ and show that $A' \leq_P A$.

There are very many known **NP**-hard problems.

- $\text{SAT} \in \mathbf{NPH}$ (Cook-Levin, 1970), to be discussed further. In the final lecture we will prove that $\text{SAT} \in \mathbf{NPH}$.
- $\text{Ham} \in \mathbf{NPH}$ and so is “traveling salesman problem” (TSP)
- “Clique” and “vertex cover” are graph-problems in **NPH**.

$$\mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{EXPSPACE}$$

All these inclusions are known; for none of them it is known if they are strict inclusions.



Satisfiability

DEFINITION

The boolean formulas are built from

- Atoms, p, q, r, \dots
- Boolean connectives \wedge, \vee, \neg (plus possibly $\rightarrow, \leftrightarrow, \perp, \top$).

A formula is **satisfiable** if we can assign values (from $\{0, 1\}$) to the atoms such that the formula is true.

SAT is the problem of deciding if a boolean formula is satisfiable.

- SAT is clearly in **NP**: The witness is an **assignment** $a : \text{Atoms} \rightarrow \{0, 1\}$; it is a simple polynomial (even linear) check whether a makes formula φ true.
- SAT was the first problem shown to be **NPH** (and thus **NP**-complete).



Variants of satisfiability I

CNF-SAT:= satisfiability of **conjunctive normal forms**

Definition: Conjunctive Normal Form (CNF)

- A CNF is a **conjunction of clauses**
- A clause is a **disjunction of literals**
- a literal is an atom or a negated atom.

Examples of CNF:

- $(p \vee \neg q \vee r \vee \neg s) \wedge (p \vee \neg r) \wedge (q \vee s)$
- $(q \vee p \vee \neg q) \wedge (q \vee \neg p) \wedge (\neg p \vee q)$

Not in CNF:

- $(p \wedge \neg q) \vee (r \wedge \neg s)$
- $((q \rightarrow p) \vee \neg q) \leftrightarrow (q \vee \neg p)$.
- The (seemingly simpler) problem CNF-SAT is also **NP**-complete.

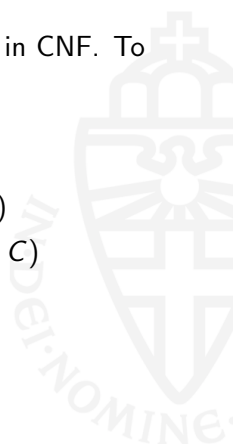


Putting formulas in CNF

LEMMA Every formula φ is equivalent to a formula ψ in CNF. To compute ψ :

- Remove (bi)implications (use $A \rightarrow B \equiv \neg A \vee B$)
- Push negations inside, next to atoms (use $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$)
- Put in CNF using $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$

NB. This can blow up a formula exponentially!





Variants of satisfiability II

DNF-SAT:= satisfiability of **disjunctive normal forms**

Definition: Disjunctive Normal Form (DNF)

A DNF is a **disjunction of conjunctions of literals**

Examples of DNF:

- $(p \wedge \neg q \wedge r \wedge \neg s) \vee (p \wedge \neg r) \vee (q \wedge s)$
- $(q \wedge p \wedge \neg q) \vee (q \wedge \neg p) \vee (\neg p \wedge q)$
- The problem DNF-SAT is in **P**.

NB. Transforming a formula $\varphi \in \text{CNF}$ into DNF may lead to an exponential blow up of φ .



NP and co-NP

DEFINITION

co-NP := $\{A \mid \bar{A} \in \mathbf{NP}\}$. (\bar{A} is the complement of A .)

- Prime (is n is a prime number?), is clearly in **co-NP**.
- It was already known for some time that Prime \in **NP**, and in 2002 it has been proven that Prime \in **P**.
- The (arguably) most well-known example of a **co-NP** problem is TAUT, deciding if a boolean formula is a **tautology**.

$$\varphi \in \text{TAUT} \quad := \quad a(\varphi) = 1 \text{ for all assignments } a.$$

- We have:

$$\varphi \in \text{TAUT} \quad \iff \quad \neg\varphi \notin \text{SAT},$$

so indeed TAUT \in **co-NP**.

- TAUT is also **co-NP hard** (and therefore **co-NP complete**): for all $A \in$ **co-NP** we have $A \leq_P$ TAUT.



P and NP and co-NP

The precise relations between **P**, **NP** and **co-NP** are a major open question in Computer Science.

Most notably there is the open question:

$$P \stackrel{??}{=} NP.$$

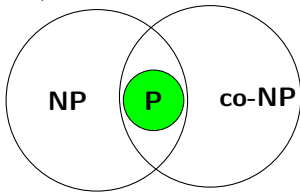
One of the 7 [Millennium Prize Problems](#), \$1 million prize for a solution, established in 2000 by the Clay Mathematics Institute (Cambridge, Massachusetts).



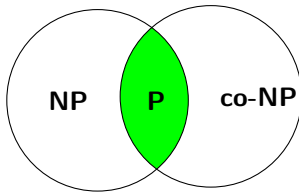


The possible relation between P and NP and co-NP

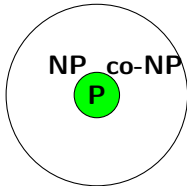
$$P \subsetneq NP \cap \text{co-NP}$$



$$P = NP \cap \text{co-NP}$$



$$P \subsetneq NP = \text{co-NP}$$



$$P = NP = \text{co-NP}$$

