Algorithms and Data Structures Analysis (ADSA)

P and NP

Class **P**

- A decision problem is polynomial solvable iff its characteristic function is polynomial-time computable.
- We use **P** to denote the class of polynomial-time-solvable decision problems.

Class NP

A decision problem L is in NP iff there is a predicate Q(x,y) and a polynomial p such that

1. for any $x \in \Sigma^*$, $x \in L$ iff there is a $y \in \Sigma^*$ with $|y| \leq p(|x|)$ and $Q(x, y)$, and

2. Q is computable in polynomial time

y is a witness that x belongs to L (guess such a witness y).
The predicate Q(x,y) is a function that returns true iff y is a witness that x belongs to L.
Verify y in polynomial time using Q.

Example: Class NP

The Hamiltonian Cycle Problem is in NP:

- We can guess a Hamiltonian cycle y in the input graph x.
- Given such a cycle y we can check in polynomial time whether it is a Hamiltonian cycle in x.

Reduction

A decision problem L' is polynomial-time reducible to a decision problem L if there is a polynomial time computable function g such that for all $x \in \Sigma^*$, we have

$$x \in L' \text{ iff } g(x) \in L.$$

Intuition: L is at least as hard as L'.

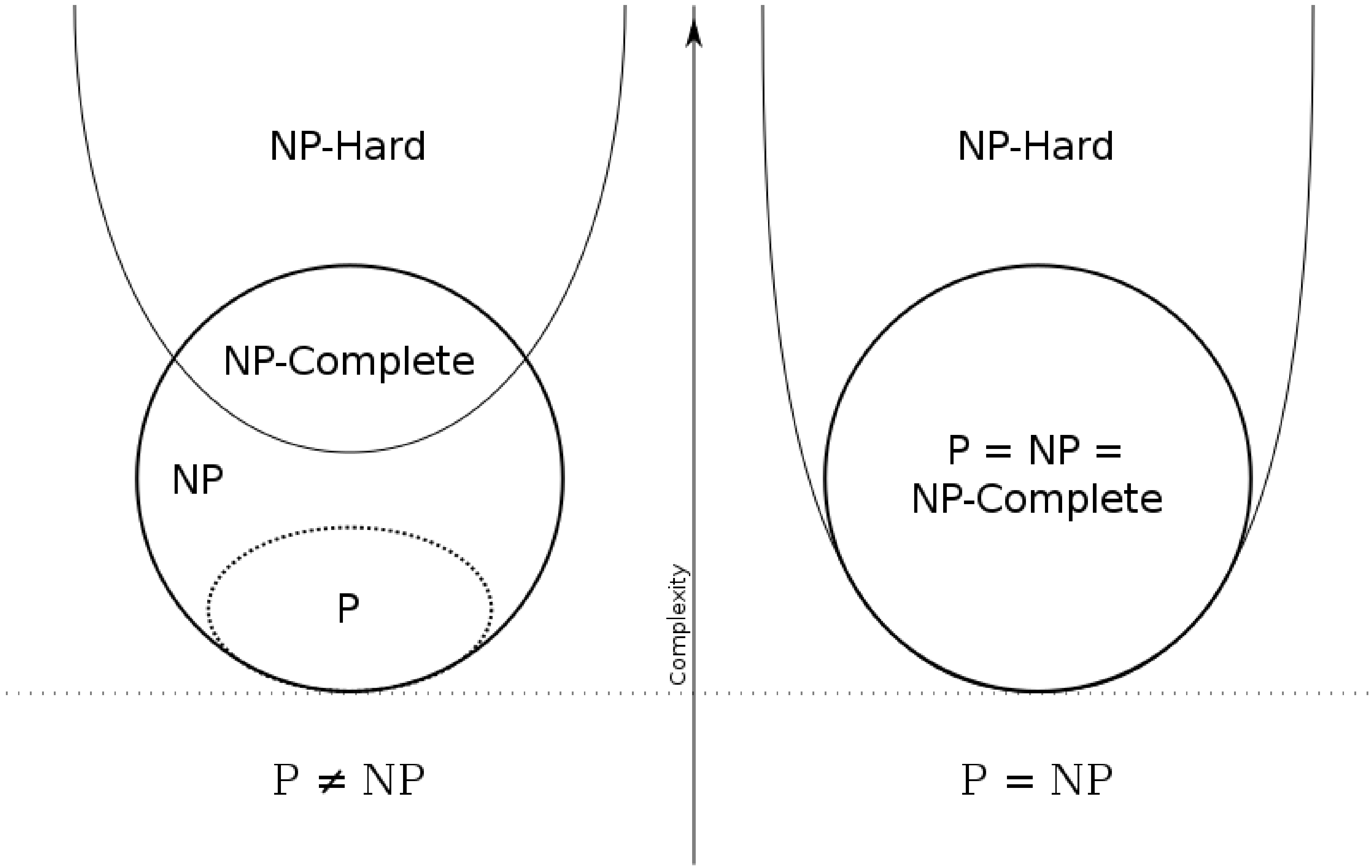
To solve L', we can use the function g and a solver for L.

NP-Hardness and NP-Completeness

- A decision problem L is NP-hard iff every problem in NP is polynomial-time reducible to it.
- A decision problem is NP-complete iff it is NP-hard and in NP.

Cook/Levin (1971): Boolean Satisfiability is NP-complete.

NP-Hard


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP =
NP-Complete
P ≠ NP
P = NP

NP-Hard

- How a problem A can be shown to be NP-Hard?
  - Find a known (another) NP-hard (or NP-complete) problem B
  - Show problem B can be solved by using A
  - In polynomial time!

How to show NP-completeness?

To show that a decision problem L is NP-complete, we need to show:

1. L in NP.
2. L is NP-hard, i.e., there is some *other* NP-complete problem L' that can be reduced to L in polynomial time.

Transitivity of reducibility relation implies that all problems in NP can be reduced to L.

Boolean Satisfiability problem

- Given: A Boolean expression in conjunctive normal form.
- Decide whether it has a satisfying assignment.

Conjunctive normal form is conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_k$

Clause is disjunction of literals $l_1 \vee l_2 \vee \ldots \vee l_h$.

Literal is variable or a negated variable.

Clique Problem

- Given: Undirected graph G=(V,E) and an integer k.
- Decide whether the graph contains a complete subgraph (clique) on k nodes.

Clique Problem

Theorem: The Clique problem is NP-complete.

Show that

1. The clique problem is in NP.
2. The clique problem is NP-hard.

Lemma 1: The Clique Problem is in NP.

- We can guess a witness y (clique of size k) and verify in polynomial time whether it is a clique of size k in the input graph given by x.

Lemma 2 (see Lemma 2.10 in Mehlhorn/Sanders):

The Boolean satisfiability problem is polynomial time reducible to the clique problem.