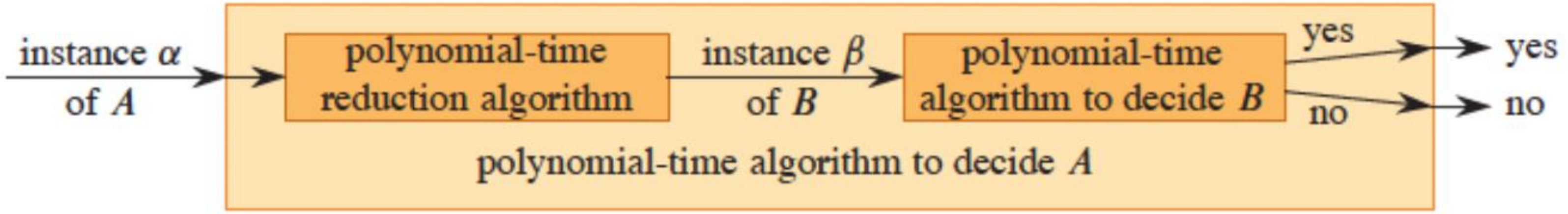
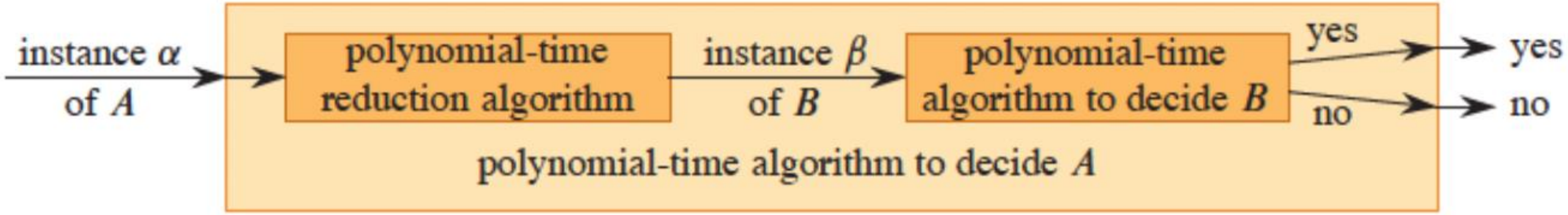
Reduction

Solve Problem A by solving Problem B



Solve Problem A by solving Problem B



Which problem is harder?

- Solving an instance of A is "harder" as it encompasses solving an instance of B as a subprocedure
- But from the point of view of algorithm design, designing an algorithm for B is harder!
- An algorithm for B entails an algorithm for A
- A polynomial-time algorithm for B entails a polynomial-time algorithm for A

Polynomial-time reduction

Definition: A *polynomial time reduction* from A to B is a polynomial time computable function f such that for every input w, $w \in A$ if and only if $f(w) \in B$. We say that A is polynomial time reducible to B, denoted by $A \leq_p B$.

Propositions:

- If $A \leq_p B$, then $\bar{A} \leq_p \bar{B}$
- 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 $A \in \mathbf{P}$

Example: k-colorability problem

k-COLOR = $\{\langle G, k\rangle \mid G$ is an undirected graph that can be colored using k colors$\}$

k-COLOR $\leq_p$ SAT

Proof: Construct a set of clauses $\Gamma_{G,k}$ such that G is k-colorable if and only if $\Gamma_{G,k}$ is satisfiable.

- c_n^i is true if and only if the n-th node has the i-th color

$$\bigwedge_n (\bigvee_i c_n^i)$$

$$\bigwedge_n \left(\bigwedge_{i,j} \neg(c_n^i \wedge c_n^j)\right)$$

$$\bigwedge_{(n,m)\in E} \left(\bigwedge_i \neg(c_n^i \wedge c_m^i)\right)$$

Hardness and Completeness

Definition: A language B is said to be $\mathbf{NP}$-*hard* iff for every $A \in \mathbf{NP}$, $A \leq_p B$. B is said to be $\mathbf{NP}$-*complete* iff $B \in \mathbf{NP}$ and B is $\mathbf{NP}$-hard.

Propositions:

- If B is $\mathbf{NP}$-hard and $B \in \mathbf{P}$, then $\mathbf{P} = \mathbf{NP}$
- If B is $\mathbf{NP}$-complete, then $B \in \mathbf{P}$ *if and only if* $\mathbf{P} = \mathbf{NP}$
- If $A \leq_p B$ for some $\mathbf{NP}$-complete A, then B is $\mathbf{NP}$-hard. If in addition, $B \in \mathbf{NP}$, then B is $\mathbf{NP}$-complete

An endless list of NPC problems:

- SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, HAM-CYCLE, TSP, SUBSET-SUM, …
- https://en.wikipedia.org/wiki/List_of_NP-complete_problems

The SAT problem

- The historically *first* problem known to be NP-complete!
- **Cook-Levin Theorem (1971):** The satisfiability problem for propositional logic is NP-complete.
- **What is satisfiability?**