Suppose \( V \) is a poly-time algorithm with two inputs, \( x, y \).

Say when \( |x| = n \) (length of \( x \))
then \( |y| = f(n) \) \( f(n) = \text{poly}(n) \).

If \( * \) is some assoc., comm. binary operation on \( \text{range}(V) \), we get a new problem

\[ \ast V(x) = \text{given } x, \text{ compute } \ast V(x, y). \]

Ex. If \( V(x, y) \) takes values in \( \{0, 1\} \) (0 = false, 1 = true)
then \( \text{OR-}V(x) = \text{given } x, \text{ compute } V \left( y \in \{0, 1\} \right) V(x, y) \)
= \text{given } x, \text{ does } \exists y \text{ st. } V(x, y) = 1? \)

Ex. Bipartite Perfect Matching
\( x = \) binary encoding of a bipartite adj. matrix.
\( y = \) binary encoding of adj. matrix of a matching
\( V(x, y) = \text{check that entries of } y \text{ are } \{0, 1\}. \)
row, column sums of \( y \) are exactly 1
\( y_{ij} \leq x_{ij} \quad \forall i,j \)

Ex. Min-Cost Bipartite perfect matching can be represented almost the same way, as

\[ \text{Min-}V(x) \]
where \( x = \) binary encoding of bipartite graph
with edge costs
\( y = \) same as before
\( V = \text{check if } y \text{ is a perfect matching contained in } X. \)

yes \( \rightarrow \) output \( \Sigma \text{ edge costs in } y \)

no \( \rightarrow \) output \( \infty \)

For the first \( V \) above, \( \Theta(V) \) counts if

\# perfect matchings is even or odd.

(can be solved in poly-time using determinants)

\[ +V(x) = \sum_{y \in \{0,1\}^F(x)} V(x,y) \]

\[ = \# \text{ of perfect matchings in } X \]

This is a complete problem for \#P.

**Def.** \#P is the class of problems \( \text{OR-}V(x) \),

where \( V \) is a poly-time algorithm, with

\#\{0,1\} output.

\( \text{coNP} \) is the class problems \( \text{AND-}V(x) \).

**Stereotypical \( \text{NP, coNP} \) problems:**

- Given a Boolean formula, is it satisfiable? \((\text{NP})\)
- Given a Boolean formula, is it a tautology? \((\text{coNP})\)
- Given a graph \( G \) and a parameter \( k \in \mathbb{N} \),
  does \( G \) contain a clique of size \( k \)? \((\text{NP})\)
- Given \( G \) and \( k \), is every clique
  smaller than \( k \) vertices? \((\text{coNP})\)
Reductions. A (poly-time Karp) reduction from problem $A$ to problem $B$ is a function $R$ st.

$$\forall x \ A(x) = B(R(x))$$

Informally, $R$ is poly-time alg that lets you solve $A$ by calling a subroutine to solve $B$ once, and outputting the result of that subroutine call.

If such a reduction exists we write $A \leq_P B$.

A problem $H$ is **NP-hard** if $A \leq_P H$ for every $A \in \text{NP}$ and $H$ is **NP-complete** if it is NP-hard and it belongs to NP.

Equivalently, NP-completeness of $H$ means it is a maximal element of $\text{NP}$ under $\leq_P$.

Then (Cook-Levin): NP-Complete problems exist. In fact, 3SAT is NP-complete.

**3SAT**: given (a binary string representing) a Boolean formula in the form $\bigwedge_{i=1}^{m} C_i$, where each clause $C_i$ is a disjunction of $\leq 3$ Boolean literals, is there a satisfying truth assignment?

Ex. $(x_1 \lor x_2) \land (x_3 \lor x_1 \lor \overline{x}_2) \land (\overline{x}_3 \lor x_2)$ satisfied by $x_1 = 0$, $x_2 = 1$, $x_3 = 1$. 
When someone hands you a decision problem \( D \), and you suspect it's \( \text{NP-complete} \), try:

1. Find a polytime verifier for \( D \), i.e.
   
   \[ V(x, y) \text{ s.t. } b(x) = \text{ok} \iff V(x, y) \]

   (usually easy)

2. Find a problem \( H \) already known to be \( \text{NP-hard} \), and show \( H \leq_p D \).

   This requires reducing FROM \( H \) TO \( D \).

   In other words the reduction transforms an instance of the known hard problem \( H \) to the new problem.

Ex. \( 4\text{SAT} \leq_p 3\text{SAT} \)

Given \( \Phi = \bigwedge_{i=1}^{m} \overline{C_i} \)

\[ C_i = x_1 \overline{x}_2 x_3 \overline{x}_4 \]

\[ C_i^1: x_1 \overline{x}_2 x_2 x_4 \]

\[ C_i^2: \overline{x}_1 x_2 x_3 x_4 \]

\[ C_i^1 \land C_i^2 \text{ is logically equivalent to } C_i \]