A word on reductions

Recall:
Definition 1.
We say a language L_1 **polynomial-time mapping reduces** to a language L_2, written as $L_1 \leq_P L_2$, iff there is a polynomial-time computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that for all $x \in \Sigma_1^*$, $x \in L_1 \iff f(x) \in L_2$.

People get the direction wrong all the time, and that's fine (as long as they get it right in the end!). Here's one way to get it wrong less often: $A \leq_P B$ can be intuitively interpreted by replacing A and B with 'hardness of A' and 'hardness of B'; thus, the 'inequality' means that problem B is *at least* as hard to solve as problem A.

More generally, a mistake people often make when they're new to reductions and are attempting to prove a statement of the form "Show that a problem A is 'hard', for some meaning of 'hard'", is to start by "We reduce A *to* a problem B we know is 'hard' in the desired sense". However, this goes the wrong way; you need to reduce *from* a hard problem.

Reducing *to* a hard problem tells you that if you can solve the hard problem B, that allows you to solve A; yet, A may still be very easy. In terms of the above notation, this corresponds to $A \leq_P B$. Reducing *from* a hard problem means that your original problem allows you to solve the hard problem, and thus must capture its 'hardness' in a sense. In terms of notation, we have $A \geq_P B$.

NP-completeness

The motivation behind NP completeness is that we want to capture the 'essence' of the class NP. Recall the two basic definitions:
Definition 2.
A language L is NP**-hard** if it is 'harder than everything in NP'; formally,

$$L \text{ is NP-hard} \iff \forall L' \in \text{NP} : L' \leq_P L$$

Definition 3.
A language L is NP**-complete** if $L \in$ NP and it is NP-hard.

As it turns out, there are many problems that are NP-complete, and that can actually be thought of as 'the same' problem when we ignore polynomial-time differences. Thus, the property of being NP-complete is one way of describing what all these problems are the 'same as'.

Examples

Exercise. *(Longest Path) Prove that the* longest path *language from HW4, namely*

$$L = \{\langle G, k\rangle : G \textit{ has a simple path of length at least } k\}$$

is NP*-complete. (Hint: reduce from* Hamiltonian Circuit*, which you may assume is* NP*-complete.)*

Exercise. *(Bounded-Occurrence SAT) Let*

$$\mathrm{CNF}_k = \{\langle\phi\rangle : \phi \text{ is a satisfiable CNF-formula in which each variable appears at most } k \text{ times}\}.$$

Note that a variable x and its negation $\overline{x}$ count as two occurrences of the same variable, and that CNF_2 *is different from* k-SAT *(the language consisting of satisfiable CNF-formulas where each clause has k literals).*

1. *Prove that* $\mathrm{CNF}_2 \in \mathrm{P}$.
2. *Prove that* CNF_3 *is* NP*-complete.*

Exercise. *Let* DoubleSAT $= \{\phi \mid \phi$ *is a boolean formula with at least 2 satisfying assignments*$\}$*. Show that* DoubleSAT *is* NP*-complete.*

Exercise. *A subset of the nodes of a graph G is a* ***dominating set*** *if every other node in G is adjacent to some node in the subset. Show that*

$$\textsc{DominatingSet} = \{\langle G, k\rangle \mid G \text{ has a dominating set of size } \leq k\}$$

is NP*-complete.*