# Reduction Algorithm

Given an SAT problem (NP-complete!) with variables and a formula, turn all in the formula into a new variable in a new formula, which is not negated. Also, add new clauses for all , where . Run the imaginary Monotone SAT code on this new formula with as . The answer from this is the answer for SAT. For example:

Will be reduced to

This reduction algorithm is polynomial because you’re just going through the entire formula and creating a new, non-negated variable for each negated variable (linear, actually). Transforming the output is constant time.

# Proof of Correctness

This problem is NP because given a yes answer an a solution, it is easy to verify the correctness of the solution in polynomial time (just plug it in and get the formula’s truth value).

The problem is NP-hard because SAT, an NP-complete problem, can be solved by reducing to Monotone SAT. We prove the correctness of our reduction as follows.

We need to prove both of the two following claims:

1. if the answer to is yes, then the answer to is also yes.
2. if the answer to is yes, then the answer to is also yes.
3. Suppose returns yes ( is satisfiable with at most variables set to true). We added new clauses. So, returning yes means the new clauses are all true, meaning at least one variable in each of those disjunctions are true; and since we have the constraint of setting at most variables to true, exactly one variable from each of these new clauses is true. This satisfies that for all , and cannot both be true in the original problem. The remaining clauses are essentially the original problem. Thus, also returns yes.
4. Since there is a one-to-one correspondence from all variables to all variables ’ (injection from variables to variables), we copy over ’s truth value to ’s truth value for all variables: set all , and . The new clauses are all obviously true, and the remaining clauses are essentially the original problem, hence also true. Thus, also returns yes.