First, we will talk about the decision version of the SAT problem and prove it is NP-complete. In the decision version, there are two inputs: an SAT formula and an integer . The question is whether you can satisfy the formula by setting at most variables to true (and the rest of the variables to false). This problem is NP-complete because a given solution can be verified in polynomial time (just set the given variables to true and see if the whole formula is true) – this shows it is NP; and the regular SAT problem can be reduced to this decision version by setting to the number of total variables in the formula – thus it is NP-hard.

Now that we’ve shown the decision version of the SAT is NP-complete, we will use it to show Monotone SAT is also NP-complete. Monotone SAT is NP because a given solution is easily verified in polynomial time (same reason as given above); and Monotone SAT is NP-hard because we can reduce decision SAT to Monotone SAT in polynomial time:

# Reduction Algorithm

Given a decision SAT problem with its two inputs of a formula and a , turn all in the formula into a new variable in the new formula, which is not negated. Run the imaginary Monotone SAT code on this new formula with the same . The answer from this is the answer for the decision SAT problem. 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