Reduction Sample Problem

1. A Fun Reduction

Define 5-SAT as the following problem:
Input: An expression in CNF form, where every term has exactly 5 literals.
Output: `true` if there is a variable setting which makes the whole expression true, `false` otherwise.

And 3-SAT as in class:
Input: expression in CNF form, where every term has exactly 3 literals.
Output: `true` if there is a variable setting which makes the whole expression true, `false` otherwise.

Prove that 5-SAT is NP-complete using 3-SAT.

1.1. Read and Understand the Problem

Read the problem and answer these quick-check-questions.

Make sure you understand 5-SAT.

- What is the input type?
- What is the output type?
- Are any words in the problem technical terms? Do you know them all?

Solution:

For 5-SAT

- input: an expression in CNF form of n Boolean variables where each clause has 5 literals
- output: true or false (depending on if we have a variable setting that makes the whole expression true)
- CNF form is AND of ORs like $(z_a \vee z_b \vee z_i) \wedge (z_c \vee z_i \vee z_j) \wedge ...$, literals z_i are boolean variables or the negation of boolean variables x_i or $\neg x_i$

You're going to design a reduction – what will that reduction look like?

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going in the right direction"
- What is the output type for your reduction?

Solution:

For the reduction

- We want to show that 5-SAT is NP-complete, so we need to reduce 3-SAT, an NP-complete problem, to 5-SAT in polynomial time. We can assume we have an algorithm for 5-SAT. In other words, we want to show that 3-SAT $\leq$ 5-SAT
- output of the reduction: a boolean which is the answer to the 3-SAT (which we get by calling 5-SAT like a library function)

1.2. Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the "certificates" (the thing that makes it a YES instance) and transform from one type of certificate to the other.
Solution:

Let $x_1, \ldots x_n$ be the variables in the 3-SAT instance and $C_1, C_2, \ldots, C_m$ be the clauses.

Create two dummy variables d_1, d_2. For each clause C_i, create four clauses:
$C_1 \vee d_1 \vee d_2$
$C_1 \vee \neg d_1 \vee d_2$
$C_1 \vee d_1 \vee \neg d_2$
$C_1 \vee \neg d_1 \vee \neg d_2$

Our 5-SAT instance is: $x_n \ldots, x_n, d_1, d_2$
The $4m$ clauses are described above.

1.3. Write The Proof

(a) to be NP-Complete, 5-SAT needs to be in NP. Argue that it is (this argument is usually only 2-3 sentences).

Solution:

A verifier would take the variables' settings to true and false. Given a setting, a verifier would check that each clause (i.e., each constraint) is satisfied. This will take time linear in the length of the constraints, so it is polynomial time.

(b) Show your reduction is correct. Remember you need to prove two implications **and** that the running time is polynomial.

Solution:

Running Time: Our algorithm makes 4 copies of every clause and adds a constant length set of literals to each clause, so the running time to create the instance is polynomial (and we call the library only once, which is also at most polynomial).

Correctness
Let ϕ_3 be our 3-SAT instance and ϕ_5 be our 5-SAT instance.

Suppose ϕ_3 is satisfiable, we show that our reduction returns true. Since ϕ_3 is satisfiable, there is a setting of the variables which causes ϕ_3 to be true. Take that setting, and set d_1, d_2 arbitrarily. Every clause of ϕ_5 is a clause of ϕ_3 with extra literals ORed on, so since each clause of ϕ_3 is true, each clause of ϕ_5 is as well, and this is a satisfying assignment.

Conversely, suppose that our reduction returns true, and therefore ϕ_5 was satisfiable. Consider a satisfying assignment for ϕ_5. We claim that (ignoring d_1, d_2) the same assignment satisfies ϕ_3. Consider an arbitrary clause C_i of ϕ_3. In ϕ_5 there were four clauses built from C_i (each ORed with all combinations of literals of d_1, d_2. One of the created clauses in ϕ_5 had both inserted literals involving d_1, d_2 being false (since we included all possible combinations). Since ϕ_5 was satisfied, this clause evaluated to true, which means that C_i evaluated to true. Since C_i was arbitrary, we have that every clause is true, and therefore a satisfying assignment for ϕ_3, as required.