Proving that a Problem is NP-Complete
Example: Set Intersection

Dorothea Blostein, CISC365

Problem Statement

Prove that the Set Intersection problem (defined below) is NP-complete. Two things are required:

- Show that Set Intersection is in NP.
- Show that CNF-satisfiability is polynomially reducible to Set Intersection. (Note: It is known from Cook's proof that CNF-satisfiabillity is NP-complete.)

Definition of the Set Intersection problem. The input consists of the finite sets $A_1, A_2, ..., A_m$ and $B_1, B_2, ..., B_n$. The problem is to decide whether there is a set S that meets the following two conditions.

(1) The intersection of S with each of the A_i sets has at least one element in it.
(2) The intersection of S with each of the B_j sets has at most one element in it.

Hint: You need to find a way to translate the SAT input (a Boolean expression) into an input for set intersection. Your goal is to construct set S so that it represents a truth assignment: there should be an obvious correspondence between the truth assignment and the items in set S. You can make this happen by choosing an appropriate definition for how the clauses in SAT's input are translated into sets A_i and B_j.

For example, suppose the CNF expression is $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_2)$. This is satisfied by the truth assignment x_1 = true, x_2 = true, and x_3 = false. To help figure out the translation, choose mnemonic names for the elements in sets A_i and B_j. I suggest using set element T_i to mean "x_i is True" and set element F_i to mean "x_i is false". So in this example, your goal is to define the translation from SAT's clauses to sets A_i and B_j in such a way that set S is forced to be $\{T_1, T_2, F_3\}$. In summary, you need to find a way to define sets A_i and B_j to make S behave this way. (Things to watch out for: If the CNF expression cannot be satisfied, then the set intersection problem should come up with a "no" answer, and there should be no set S meeting the two conditions. Also, make sure that set S can never contain both T_i and F_i; otherwise the elements in S cannot be interpreted as a truth assignment.)

Hint: Start this problem by going over a few examples of the set intersection problem. Write down some sets A and B, and figure out if there is a solution set S. Then go over some examples of the desired translation from SAT to set intersection: "Here is a Boolean formula. What do we want set S to look like? What rules can we use for translating from the Boolean formula to sets A and B, in order to force set S to look like this?" Figure out how the size of the SAT input will determine the size of the set-intersection input that you construct. Suppose the SAT input contains U literals, V clauses, and W Boolean variables. How will you use the values U, V, and W to choose values for m and n in the set intersection problem?

Solution

Part 0: describe how the input size is measured. Call the input size "I_{size}", since "n" is already used for the number of B sets. The input size I_{size} is the sum of the set sizes: $I_{size} = |A_1| + |A_2| + ... + |A_m| + |B_1| + |B_2| + ... + |B_n|$. Let K denote the number of distinct elements in the input sets: $K = |A_1 \cup A_2 \cup ... \cup A_m \cup B_1 \cup B_2 \cup ... \cup B_n|$. By definition, $K \leq I_{size}$.

Part 1: *set intersection* is in NP. A guessed solution consists of a subset of the K elements in the input sets. The number of guessed solutions is 2^K, since each of the K elements can be included or excluded from the guessed solution set. Check a solution S by computing $A_i \cap S$ for $1 \leq i \leq m$ and computing $B_j \cap S$ for $1 \leq j \leq n$. This takes time polynomial in the input size I_{size}. [Here is a quick upper bound to prove this. Checking a solution requires at most I_{size} set intersections to be computed. Each set intersection involves two sets that have at most I_{size} elements each. Thus, the time for one set intersection is $O(I_{size}^2)$ and the overall checking time is $O(I_{size}^3)$.]

Part 2: CNF Satisfiability can be polynomially-reduced to set intersection. The input to satisfiability is a CNF expression, consisting of U literals arranged into V clauses. The number of Boolean variables is W. (For example, $(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\overline{x}_2 \vee x_4 \vee x_3)$ has 7 literals arranged into 3 clauses. There are 4 Boolean variables: x_1, x_2, x_3, x_4.) Define transform T as follows. (Clearly this transform can be computed in polynomial time.)

- Translate clause i in the CNF expression into set A_i as follows. Throw away the logical "or" operators. Turn each unnegated literal x_i into set element T_i. Turn each negated literal $\overline{x}_i$ into set element F_i. Do this for all clauses ($1 \le i \le V$). The value m in the set intersection problem is equal to V (the number of clauses) in the CNF satisfiability problem.
- For $1 \le j \le W$, define $B_j = \{T_j, F_j\}$. The value n in the set intersection problem is equal to W (number of Boolean variables) in the CNF satisfiability problem. We see that $I_{size} = U + 2*W$ and $K=2*W$.

Here is an example of this transform from CNF satisfiability to set intersection (using a b c instead of x_1 x_2 x_3):

The input to CNF satisfiability is $(a \vee \overline{b} \vee c) \wedge (\overline{a} \vee \overline{c}) \wedge (b \vee c) \wedge (a \vee b)$. In this case, U = 9, V = 4, and W = 3.

The transform produces this input for the set intersection problem (m = 4, n = 3, I_{size} = 15, and K = 6):

$A_1 = \{T_a, F_b, T_c\}$ $A_2 = \{F_a, F_c\}$ $A_3 = \{T_b, T_c\}$ $A_4 = \{T_a, T_b\}$ $B_1 = \{T_a, F_a\}$ $B_2 = \{T_b, F_b\}$ $B_3 = \{T_c, F_c\}$

The solutions to these two problems correspond. For example, the truth assignment "a=True, b=True, c=False" satisfies the CNF expression; similarly "$S=\{T_a, T_b, F_c\}$" solves the set intersection problem.

Justification that this transform is correct.

The sets B_j ensure that set S cannot contain both T_j and F_j. Therefore, the elements in set S can be interpreted as a truth assignment: each variable is either True or False (but not both at once).

The sets A_i force set S to define a truth assignment that satisfies the CNF expression. Each set A_i corresponds to one clause of the CNF expression. Since set S has a non-empty intersection with A_i, the truth assignment in S must satisfy clause number i in the CNF expression. [Note: if S does not contain either T_j or F_j, this means that x_j can be either True or False; either way, the CNF expression is satisfied by the other elements in S.]