Theory of Computation

Solutions for Homework 3

Problem 1. Suppose L_1 is NP-complete, L_2 is in NP and L_1 is reducible to L_2, prove that L_2 is NP-complete.

Proof. Given L_2 is in NP, it remains to show that every language L in NP is reducible to L_2.

Because L_1 is NP-complete, every language in NP is reducible to L_1. As L_1 is reducible to L_2, by the transitive property of reducibility (see Proposition 8.2 in textbook), L is reducible to L_2. Hence every language in NP is reducible to L_2.

Problem 2. Define the language

$C_{NP} = \{\langle M, x, 1^s \rangle | M \text{ is a nondeterministic TM that accepts } x \text{ within } s \text{ steps}\}$

Prove that C_{NP} is NP-complete. (Recall that 1^k denotes the string consisting of k bits of 1's.)

Proof. First, we show that C_{NP} is in NP. With the input $\langle M, x, 1^s \rangle$, we simulate M on x up to s steps and *accept* if M accepts x. The algorithm clearly runs in polynomial time. We proceed to show that C_{NP} is NP-hard. Let $L \in$ NP is accepted by an NTM M that runs in polynomial time $O(n^c)$ for some constant c. To reduce L to C_{NP}, we simply map input x (is $x \in L$?) to the triple $\langle M, x, 1^{n^c} \rangle$. Our reduction can obviously be performed in polynomial time (to be sure as the time may be n^{c+1} or more). It is clear that $x \in L$ iff $\langle M, x, 1^{n^c} \rangle \in C_{NP}$. C_{NP} is therefore NP-complete.