Theory of Computation

Mid-Term Examination on November 5, 2013
Fall Semester, 2013

Problem 1 (25 points) Show that if NP $\neq$ coNP, then P $\neq$ NP.

Ans: P is closed under complementation. If P = NP, then NP is also closed under complementation. In other words, NP = coNP. ■

Problem 2 (25 points) It is known that $H^* = \{M | M \text{ halts on all inputs}\}$ is undecidable. Show that L is undecidable, where

$$L = \{M_1; M_2 | M_1 \text{ and } M_2 \text{ are TMs and } M_1(x) = M_2(x) \text{ for all inputs } x\}.$$

Ans: We prove that L is undecidable by reducing H^* to L. Suppose L is decidable. Given a TM M, we construct M_1 and M_2 as follows. M_1 simulates M on any input and accepts if M halts. M_2 always accepts on its input. Obviously, M halts on all inputs if and only if $M_1(x) = M_2(x)$ for all inputs x. So $M \in H^*$ if and only if $M_1; M_2 \in L$. So if L were decidable, H^* would be decidable, a contradiction. Hence, L is undecidable. ■

Problem 3 (25 points) Prove that the language C_{NP} is NP-complete, where

$$C_{NP} = \{(N, x, 0^t) \mid N \text{ is an NTM that accepts } x \text{ within time } t\}.$$

Recall that 0^k denotes the string consisting of k 0s. Do not forget to show C_{NP} is in NP.

Ans: We first show that C_{NP} is in NP. With the input $(N, x, 0^t)$, we simulate N on x up to t steps of N and accept if N accepts x. The algorithm obviously runs in polynomial time. We next show that C_{NP} is NP-hard. Let $L \in$ NP be accepted by an NTM N that runs in polynomial time n^c for some constant c. To reduce L to C_{NP}, simply map the input x to the triple $(N, x, 0^{n^c})$. The reduction can evidently be performed in polynomial time. It is clear that $x \in L$ iff $(N, x, 0^{n^c}) \in C_{NP}$.

■

Problem 4 (25 points) We say that a function $f : \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+$ is a **proper complexity function** if

1. f is non-decreasing, i.e, $f(n+1) \geq f(n)$ for all positive integers n.
2. There is a k-string Turing Machine M_f with input and output that, given an input of length n,
   (a) outputs $\sqcap^{f(n)}$ on its output string in time $\mathcal{O}(n + f(n))$, and
   (b) uses $\mathcal{O}(f(n))$ space besides its input and output.

Show that the set of proper complexity functions is closed under sums (i.e., if f and g are proper complexity functions, then $f + g$ is also a proper complexity function.)

Ans: Let f and g be two proper complexity functions. Let's notice that

1. For all positive integers n,

$$(f+g)(n+1) = f(n+1) + g(n+1) \geq f(n) + g(n) = (f+g)(n)$$

because f and g are non-decreasing; hence $f+g$ is also non-decreasing.

2. Let M_f and M_g be the k_f-string and k_g-string Turing machines associated with f and g, respectively. Let's construct a $(k_f + k_g)$-string Turing machine called M_{f+g} as follows:

   (a) Given an input of length n, M_{f+g} first emulates M_f on it to write $\sqcap^{f(n)}$ in the k_fth string.

   (b) Then M_{f+g} emulates M_g on the original input to write $\sqcap^{g(n)}$ in the $(k_f + k_g - 1)$st string.

   (c) Finally, M_{f+g} concatenates the k_fth and $(k_f + k_g - 1)$st strings and outputs it in the $(k_f + k_g)$th tape.

   From the construction above, we notice that given an input of length n, the output of M_{f+g} is of length $(f+g)(n) = f(n) + g(n)$. Now, let's notice that

   (a) M_{f+g} runs in time

$$\mathcal{O}(n + f(n) + n + g(n) + f(n) + g(n)) = \mathcal{O}(n + f(n) + g(n)).$$

   (b) The maximum space M_{f+g} uses is

$$\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n)).$$

From the two items above, $f+g$ is a proper complexity function, hence the set of all proper complexity functions is closed under sums.

■