Theory of Computation

Midterm Examination on October 25, 2016

Fall Semester, 2016

Problem 1 (25 points) Prove that the following language L is undecidable:

$$L = \{M; x; n | \text{ a TM } M \text{ with input } x \text{ executes the } n\text{th instruction of } M.\}$$

Proof: Suppose that L is decidable. We now reduce the halting problem to L. Consider an instance $M; x$. Then replace all instructions $\delta(q, s) = (r, t, D)$, where r is a "yes," a "no," or an h, with $\delta(q, s) = (Q, t, D)$, where Q is a new state. Then add instructions which make the head move to the beginning of the tape (with symbol $\triangleright$) while remaining at state Q. Let k be the number of instructions of the aforesaid machine. Finally, add the instruction $\delta(Q, \triangleright) = (h, \triangleright, -)$ numbered $n = k + 1$. Call this modified machine M'. Now we construct a TM M'' such that

$$M''(M'; x; n) = \begin{cases} \text{"yes"}, & \text{if } M'(x) \text{ executes the } n\text{th instruction;} \\ \text{"no"}, & \text{otherwise.} \end{cases}$$

Clearly $M'; x; n \in L$ if and only if $M; x \in H$, a contradiction. Hence L is undecidable. ■

Problem 2 (25 points) Prove that if L is recursively enumerable but not recursive, then $\overline{L}$ is not recursively enumerable.

Proof: Assume that $\overline{L}$ is recursively enumerable. Then both L and $\overline{L}$ are recursively enumerable (see p. 150 of the slides). This implies that L is recursive, a contradiction. ■

Problem 3 (25 points)

1. Show that the satisfiability of DNF is polynomial-time solvable.
2. By transforming any boolean expression into a DNF (as explained in the slides), we can solve all satisfiability problem in polynomial time. What is wrong with this argument.

Proof:

1. A DNF is satisfiable if and only if there is at least one implicant which does not contain some variable and its negation. Testing this condition is easy. So, the satisfiability of DNF is polynomial-time solvable.
2. The reduction may take exponential time.

■

Problem 4 (25 points) Let L be an NP-complete language. Prove that P = NP if and only if $L \in$ P.

Proof: First, suppose $L \in$ P. Every language $L' \in$ NP can reduce to L because L is an NP-complete language. Since P is closed under reductions, $L' \in$ P. Thus NP $\subseteq$ P. As P $\subseteq$ NP, we conclude that P = NP. On the other hand, suppose P = NP. As $L \in$ NP, it is obvious that $L \in$ P. ■