22C:131 Midterm Exam

Date: Friday, 3/3

Notes: Solve all 3 problems listed below. The exam will be graded out of 200 points (worth 20% of your grade). The first two problems are worth 75 points each and the last is worth 50 points. This is an open notes/book exam.

1. Let L be a Turing-recognizable language and let $\overline{L}$ (the complement of L) be such that it is not Turing-recognizable. Consider the language:

   $$L' = \{0w \mid w \in L\} \cup \{1w \mid w \notin L\}.$$

   Is L' Turing-decidable, Turing-recognizable, or not even Turing-recognizable? Justify your answer.

   Solution: L' is not even Turing-recognizable. Here is a proof by contradiction. Suppose L' were Turing-recognizable. Then there would exist a Turing machine M' that would, given an input $w \in \Sigma^*$, halt in an accepting state if $w \in L'$ and either halt in a rejecting state or keep looping forever if $w \notin L'$. Using M', we can build a Turing machine $\overline{M}$ that recognizes $\overline{L}$. $\overline{M}$ takes an input $w \in \Sigma^*$ and sends $1w$ as input to M' and accepts if M' accepts and rejects if M' rejects. Therefore, $\overline{M}$ accepts strings $w \in \Sigma^*$ for which $1w \in L'$. Now note that by the definition of L' (in terms of L and $\overline{L}$), $1w \in L'$ iff $w \in \overline{L}$. This means that $\overline{M}$ accepts precisely the strings in $\overline{L}$. Since we are given that $\overline{L}$ is not a Turing-recognizable language, we have a contradiction.

2. Let L be the language of all Turing machine descriptions $\langle M \rangle$ such that there exists some input on which M makes at least 5 moves. Show that L is decidable.

   Solution: To show that L is decidable we will construct a Turing machine R that takes as input a Turing machine description $\langle M \rangle$ and determines if there exists some input on which M makes at least 5 moves. The general idea for R is that it generates *all possible* inputs for M, simulates M on each of these, and accepts if on at least one of these inputs M runs for 5 or more moves; otherwise it rejects. In general, the set of all possible inputs is infinite in size, but in this case, since we are only interested is the first 5 moves of M, it suffices to generate all length 5 prefixes of inputs to M. The number of such prefixes is $|\Sigma|^5$.

   So R starts by generating a length 5 prefix, say w, and simulates M on w. The string w could be the lexicographically smallest length 5 string in Σ^*. R counts the number of moves M makes (maybe on a separate tape, for convenience) and if this count reaches 5, then R accepts. Otherwise, if M halts in 4 or fewer moves, R erases w, and generates the lexicographically next length 5 string and repeats the above steps. If R has simulated M on the lexicographically last length 5 string and even on this string M has halted in 4 or fewer moves, then R rejects.

3. Your friend claims that if $L \in NP$, then $\overline{L} \in NP$. To bolster her claim she says "Look, it is obvious that $COMPOSITES \in NP$ and with a little knowledge of number theory, one can show that $PRIMES \in NP$." Can you point your friend to a language L that is in NP and for which it seems quite difficult to claim $\overline{L} \in NP$. Briefly explain your answer.

Solution: There are a number of languages L that are in NP for which it is not known whether $\overline{L}$ is in NP or not. Consider for example, the language CLIQUE. This is the set of all (G, k) pairs, where G is a graph and k is a nonnegative integer such that G has a clique of size at least k. CLIQUE is in NP because it can be verified in polynomial time using as certificate a clique in G of size k or more. Now $\overline{CLIQUE}$ is the language of all (G, k) pairs such that G has no clique of size at least k. How would we verify $\overline{CLIQUE}$ in polynomial time? One reasonable candidate for a certificate would be a partition of the vertices of G into independent sets. If the size of this partition is less than k, then clearly G has no clique of size k or more. However, if the size of this partition is k or more, it does not necessarily mean that G has a clique of size k or more. There are graphs for which the size of a smallest partition into independent sets is much larger than the size of the largest clique. This is why, this attempt at certifying the absence of a large clique, fails. Of course, there may be more sophisticated ways of constructing polynomial time verifiable certificates for the absence of a large clique. However, whether this is possible or not, is not yet known.
