CSE 531
Assignment 4

Due October 26, 2000

A new feature of this assignment is the "optional" problem. These problems will be evaluated by the TA or professor, but will not count toward the grade. These problems might be more challenging than the regular problems.

1. Using brief implementation descriptions show the following. You can use the fact that nondeterministic multi-tape Turing machines can be simulated by deterministic Turing machines. Assume L_1, L_2 and L are Turing-recognizable.

   (a) $L_1 \cup L_2$ is Turing-recognizable.

   (b) $L_1 \cap L_2$ is Turing-recognizable.

   (c) $L_1 \cdot L_2$ (the concatenation of L_1 and L_2) is Turing-recognizable.

   (d) $L^R = \{w^R : w \in L\}$ is Turing-recognizable.

2. We all know what a context-free grammar is, but many years ago people were interested in *general grammars* which are more powerful. A general grammar is of the form $G = (V, \Sigma, R, S)$ where V is a finite set of nonterminals, Σ is the set of terminals, R is a finite subset of $V^* \rightarrow (V \cup \Sigma)^*$, and $S \in V$ is the start symbol. A derivation step in a general grammar is essentially the same as in a context-free grammar. If uvw is a string and $v \rightarrow x$ is in R then $uvw \Rightarrow uxw$. The language $L(G)$ is the set of all terminal strings that can be derived from S in some number of steps.

   (a) Prove that for any general grammar G, $L(G)$ is Turing-recognizable.

   (b) Prove that for any Turing-recognizable language L there is a general grammar G such that $L(G) = L$. (Hint: If $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ is a Turing machine then we want to construct a general grammar G_M with the property that $L(M) = L(G_M)$. One idea is to use the result above that if L is Turing-recognizable then so is L^R. We can define a grammar G_M which which has terminal alphabet Σ and nonterminal alphabet $\Sigma' \cup (\Gamma - \Sigma) \cup Q \cup \{\#\}$ where $\Sigma' = \{a' : a \in \Sigma\}$. If $w \in \Sigma^*$ then define w' by priming all the symbols in w. For example, $(abab)' = a'b'a'b'$. From the start symbol of G_M a string of the form $w^R \# w' \sqcup^* \#$ can be generated. The grammar can then simulate M on the string between the two $\#$'s. Once an accepting state is reached the grammar can then erase everything to the right of and including the first $\#$. The challenge is to define the rules of the grammar to do all of this. Once this is done it will be the case that G_M generates L^R.)

3. (optional) Prove that for all languages B, $B <_T A^B_{TM}$. Recall $A^B_{TM} = \{\langle M, w\rangle : M$ is an oracle Turing machine that accepts w when given the oracle $B\}$.

4. (optional) Prove that $\text{Infinite}_{TM} \equiv_T \text{Decider}_{TM}$ where $\text{Infinite}_{TM} = \{\langle M\rangle : M$ accepts infinitely many inputs$\}$ and $\text{Decider}_{TM} = \{\langle M\rangle : M$ halts on all inputs$\}$.