Problem Set 7 — Due May 20, 2004

Problem 1. Recall that an *unrestricted grammar* $G = (V, \Sigma, R, S)$ is just like a context-free grammar except that the rules are a finite subset of $(\Sigma \cup \Gamma)^*\Gamma(\Sigma \cup \Gamma)^* \times (\Sigma \cup \Gamma)^*$. Derivations in an unrestricted grammar are just like derivations in a CFG: if there is a rule $\alpha \to \beta$ and you see α in a sentential form, you can replace α by β (possibly resulting in the erasure or change of terminals). The language of G, $L(G)$ is the set of terminal strings derivable from the start symbol S.

Part A. Exhibit an unrestricted grammar for $L = \{ww : \ w \in \{a, b\}^*\}$

Part B. Prove that a language is r.e. if and only if it is generated by an unrestricted grammar.

Part C. Prove that there is no algorithm which takes an unrestricted grammar G and a word w and decides if $w \in L(G)$.

Problem 2 Classify each of the following problems as either **decidable**—I see how to decide this language; **r.e.**—I don't see how to decide this language, but I can see a procedure to accept this language; **co-r.e.**—I don't see how to decide this language, but I can see a procedure to accept the complement of the language; **neither**: I don't see how to accept this language nor its complement.

Part A. $\{\langle M\rangle : M$ is a TM that accepts some string of prime length$\}$.

Part B. $\{\langle M\rangle : M$ is a C-program that halts on $\langle M\rangle\}$.

Part C. $\{\langle G\rangle : G$ is a CFG and G accepts an odd-length string$\}$.

Part D. $\{\langle M\rangle : M$ is a TM and M has 150 states$\}$.

Part E. $\{\langle M\rangle : M$ is a TM and $L(M) = L(M)^*\}$.

Part F. $\{\langle M\rangle : M$ is a TM and $L(M) = \emptyset\}$.

Part G. $\{\langle M\rangle : M$ is a TM and $L(M)$ is r.e. $\}$.

Part H. $\{\langle G_1, G_2\rangle : G_1$ and G_2 are CFGs and $L(G_1) = L(G_2)\}$.

Part I. $\{\langle M\rangle : M$ is a TM and M will visit state q_{25} when run on some input $x\}$.

Problem 3 *(for fun, if you want—won't be graded)* A *three-pebble machine* is like an ordinary TM except that the input x is presented on a read-only tape, surrounded by delimiters, $[x]$, and there is a auxiliary tape, which the machine can not write to, but which the machine can place three pebbles on. The machine can move these pebbles around, picking up a pebble and moving to a neighboring square. Formalize the behavior of such a machine and show that a three-pebble machine can accept any r.e. language.