CS 5311 **HW 3: Turing Machines** **Due: Thurs., 3/7/13**

(75 points)

1. (20) Prove that if the languages L and $\overline{L}$ are Turing-recognizable then L is decidable.

   To prove this statement, you may assume that M and $\overline{M}$ are TMs that recognize L and $\overline{L}$, respectively. Show how to construct a TM D that decides L. D will use M and $\overline{M}$ to do its job. Express your proof in the algorithmic style that the author develops in Chapter 3.

   M, $\overline{M}$, and D must all be deterministic TMs.

   Note that since M and $\overline{M}$ are "only" recognizers, they might not halt on some inputs. Note also that if M is given a string w that is in L, M will halt in its accept state. However, for that same string w, $\overline{M}$ might halt in its reject state or it might never halt. Similar comments apply if $w \notin L$ with M and $\overline{M}$ interchanged.

2. (5) Prove or disprove: There is a Turing-recognizable language L that is also decidable. (This is easy.)

3. (20) Problem 3.18. Express your proof in the algorithmic style that the author develops in Chapter 3. Your proof will have two parts:

   (a) Prove that if the language L is decidable, then an enumerator E can be constructed that enumerates all and only the strings of L in lexicographic order.

   (b) Prove that if the strings of a language L can be enumerated in lexicographic order by some enumerator E, then L is decidable.

4. (10) This question is about *encodings* of formal objects, particularly, encodings of Turing machine *descriptions*. (See the last subsection of Chapter 3.)

   Let L be a Turing-recognizable language and let M be a Turing machine that recognizes L. (In the notation of encodings, $\langle M \rangle$ is the string that is the encoding of M.)

   Prove or disprove: The language $L_d = \{ \langle M_d \rangle \mid M_d \text{ is a Turing machine that decides } L \}$ is decidable. (*Hint:* See problem 3.22.)

5. (10) (See Section 3.3.)

   There are TMs which, on certain inputs, do not halt. Other TMs always halt no matter what input they are given. An algorithm must always halt, no matter what its input is. How can we (and the C-T thesis) claim that TMs exactly characterize the idea of "algorithm" when there is this significant difference between the possible behavior of a TM (some do not always halt) and the behavior of an algorithm (which requires halting on all inputs)?

6. (10) Suppose you have invented a new formal model of computation called a Hilbert machine (HM). (In this problem a "formal model of computation" is a mathematically defined type of object such as a DFA, an NFA, a PDA, or a TM.)

   (a) Suppose that you publish a paper that contains your definition of HMs and all readers agree that HMs are a formal model of computation. Also, suppose you believe that HMs are more powerful than TMs. What would have have to do to show that HMs are strictly more powerful than TMs? (*Hint:* To solve this problem, assume that there is a problem called "*HP*" that no TM can decide.)

   (b) How would your result from part (a) affect the standing of the Church-Turing thesis? Explain why your result would have the effect you claim it would have.