CS 172: Computability and Complexity, Spring 2010 **S. A. Seshia & O. Etesami**

HW 7: Undecidable/non-Turing Recognizable Languages

Assigned: March 28, 2010 *Due: April 5, 2010*

Note: Take time to write clear and concise solutions. Confused and long-winded answers may be penalized. Consult the course webpage for course policies on collaboration.

1. (8 points) Read about Rice's Theorem in Problem 5.28 in Sipser's book. A form of Rice's Theorem is stated as follows:

   Let P be a language of Turing machine descriptions (i.e., containing strings of the form $\langle M \rangle$ where M is a TM), where P satisfies two conditions:

   1) P is non-trivial: There exist TMs M_1 and M_2 such that $\langle M_1 \rangle \in P$ but $\langle M_2 \rangle \notin P$.
   2) P is a property of (only) a TMs language: For any TMs M_1 and M_2 such that $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in P$ if and only if $\langle M_2 \rangle \in P$.

   Then P is undecidable.

   The proof of Rice's Theorem (Prob. 5.28) is given at the back of Chapter 5 of Sipser's book. Review it, and then answer the following questions:

   (a) Show that both of the above conditions of Rice's theorem are necessary. In other words, show, for each condition, that leaving it out makes P decidable for some P.

   (b) Using Rice's Theorem, prove that the following language is undecidable:

   $$ALL_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$$

2. (5 points) Prove that there exists a subset of $\{1\}^*$ which is not Turing-recognizable.

3. (9 points)

   (a) (5 points) Let $L_1 = \{\langle M, w \rangle \mid$ Turing machine M on input w attempts to move its head left at some point during its computation when its head is on the left-most tape cell$\}$. Prove that L_1 is undecidable.

   (b) (4 points) Let $L_2 = \{\langle M, w \rangle \mid$ Turing machine M on input w attempts to move its head left at some point during its computation$\}$. Prove that L_2 is decidable.

4. (8 points) [Gödel's Incompleteness Theorem]
   One of the most important results of 20th century logic (and arguably of all 20th century

mathematics) was the discovery that there are some true mathematical statements that cannot be proven using the standard axioms of mathematics. This 1931 result by Kurt Gödel is known as Gödel's Incompleteness Theorem. In this problem, you will prove one version of Gödel's Theorem using what we know about Turing machines, 79 years too late to revolutionize modern mathematics.

(The actual proof given by Gödel is a little more complicated, since Turing Machines hadn't been invented yet, but it relies on the same basic notion of diagonalization, and the two results are intimately related.)

Some preliminaries: we say that a system of mathematics is *consistent* if there exists a special proof-checking Turing machine M^* that can verify or reject any proof that it is given in the language of that system. In other words, there exists a machine M^* that decides the language

$$PROOFS = \{\langle P, T\rangle | \text{ P is a valid proof of statement T}\}$$

Assume that our system of mathematics is *consistent*, so that we can check algorithmically whether a proof is correct. Assume for simplicity's sake that any statement T in our system of mathematics is either true or false - exactly one of T or $not(T)$ is true. For this question, you can express a mathematical statement in the usual way using English, rather than encoding it into any special logical language.

(a) Suppose for a contradiction that every true statement T_i has a proof P_i. Using this assumption, give a Turing machine that is a decider for the language $TSTMTS = \{\langle T\rangle | \text{ T is a true statement }\}$.

(b) Use the result of part (a) to construct a Turing machine that solves the halting problem.

(c) Conclude that there exists a true theorem that does not have a proof, proving Gödel's Theorem.