

Computability and Logic Homework 8

Due: Thursday, November 10, 2005

Problems to Hand In

1. Let \prec denote the standard lexicographical ordering on strings. A language L is said to be *lexicographically enumerable* if there exists a Turing machine that enumerates the elements of L **in order**; that is, in such a way that if w_1 is displayed before w_2 , then $w_1 \prec w_2$. Prove that a language is lexicographically enumerable if and only if it is decidable. (Be careful: a machine that lexicographically enumerates a language need not ever halt, even if the language is finite!)

2.

a. Prove that the following language is undecidable:

$$\{\langle M \rangle \# v \# w \mid M \text{ outputs } w \text{ on input } v\}$$

b. Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a (total) function from natural numbers to natural numbers. Prove that f is recursive if and only if the language $\{1^n 0 1^{f(n)} \mid n \in \mathbb{N}\}$ is decidable.

c. A *partial function* is like a function, except that there may be some inputs on which it is undefined. In particular, f is a *partial recursive function* if there is a Turing machine which, given input x , outputs $f(x)$ if it is defined and does not halt otherwise.

Prove that f is a partial recursive function if and only if the language $\{1^n 0 1^{f(n)} \mid n \in \mathbb{N}, f(n) \text{ defined}\}$ is recursively enumerable.

3. Consider the language $\mathcal{E} = \{\langle M_1 \rangle \# \langle M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$ (that is, pairs of Turing machines that halt on exactly the same input strings).

a. Prove that \mathcal{E} is undecidable by showing how to use a machine that decides this language to solve the Halting Problem.

b. Prove that \mathcal{E} is not recursively enumerable by showing how to use an acceptor for \mathcal{E} to accept the language of Turing machines that always halt.

c. Prove that $\overline{\mathcal{E}}$ is recursively enumerable.

4. For any Turing machine M and any string w_0 , define the M -iterates of w_0 to be the strings $w_0, w_1, w_2 \dots$ defined as follows:

w_1 is the output of M on input w_0 ,
 w_2 is the output of M on input w_1 ,
 w_3 is the output of M on input w_2 ,
 \vdots

Define a k -iterator to be a Turing machine that takes as input a Turing machine encoding $\langle M \rangle$ and a string w and outputs the k th M -iterate of w .

- a. Describe a Turing machine N that takes a Turing machine encoding $\langle M \rangle$ as input and, if M is a k -iterator, outputs the string 1^k . (It doesn't matter what N does if M is not an iterator.)

Note: For parts (b) through (d), solutions that do not rely on part (a) are preferable.

- b. Describe a Turing machine S that takes a Turing machine encoding $\langle M \rangle$ as input and outputs the encoding of a Turing machine M' such that if M is a k -iterator, then M' is a $(k + 1)$ -iterator.
- c. Describe a Turing machine P that takes as input two Turing machine encodings M_1 and M_2 and outputs the encoding of a Turing machine M' such that if M_1 is a k_1 -iterator and M_2 is a k_2 -iterator, then M' is a $(k_1 + k_2)$ -iterator.
- d. Describe a Turing machine T that takes as input two Turing machine encodings M_1 and M_2 and outputs the encoding of a Turing machine M' such that if M_1 is a k_1 -iterator and M_2 is a k_2 -iterator, then M' is a $k_1 k_2$ -iterator.