# CMSC 28100-1 / MATH 28100-1
# Introduction to Complexity Theory
# Fall 2017 – Homework 5

November 1, 2017

**Definition.** *A Turing machine* transducer *is a Turing machine that outputs strings, rather than accepting or rejecting. That is, if $M$ is a Turing machine, then $M$ computes the function which takes an input $x$ to the value that is left on the tape when $M$ halts. If $M$ does not halt on input $x$, then the function computed by $M$ is undefined on $x$. A function $f$ that is computed by a Turing machine transducer is called* partial computable *(partial, since $M$ may not halt on some inputs, and then $f$ is undefined on those inputs).*

**Exercise 1.** In class and in the book, you were given the following definition of a recursively enumerable language.

A language $L$ is recursively enumerable if there is a Turing machine $M$ such that $M$ accepts exactly those strings in $L$. (On strings not in $L$, the machine $M$ may either reject or not halt.)

Show that each of the following conditions is equivalent to $L$ being recursively enumerable (in the above sense).

(a) There is a partial computable function $f$ such that $L$ is the range of $f$, that is, we have $L = \{f(x) : x \in \Sigma^*\}$. Such an $f$ is called an *enumerator* of $L$ since when $f$ is run on all strings in parallel, it enumerates/lists the elements of $L$ (possibly with repetitions).

(b) There is a partial computable function $f$ such that $L$ is the domain of $f$, that is, we have $L = \{x : f(x) \text{ is defined}\}$.

(c) There is a Turing machine $M$ such that $L = \{x : M \text{ halts on input } x\}$.

**Exercise 2.** In this exercise, we will show the recursive analogue of item (a) of Exercise 1.

First, fix a computable enumeration $(x_n)_{n\in\mathbb{N}}$ of $\Sigma^*$ without repetitions, that is, the following hold.

- For every $n \neq m$, we have $x_n \neq x_m$.
- We have $\Sigma^* = \{x_n : n \in \mathbb{N}\}$.
- There is a Turing machine $E$ that halts on every input and if the input is a binary representation of $n \in \mathbb{N}$, the tape contains $x_n$ when $E$ halts, i.e., the machine $E$ computes the function $\langle n\rangle \mapsto x_n$.

This induces an order of the words in $\Sigma^*$ given by

$$x_n \leqslant x_m \iff n \leqslant m.$$

Show that a language $L$ is recursive if and only if there is an enumerator $f$ for $L$ such that $f$ is *monotone*, that is, for every $x, y \in \Sigma^*$ with $x \leqslant y$, we have $f(x) \leqslant f(y)$, where when $f(x)$ is undefined, we interpret as $f(x) = +\infty$.

Hint: it is useful to separate the case when $L$ is finite.

**Exercise 3.** Show that every infinite recursively enumerable language $L$ contains an infinite subset $L' \subseteq L$ such that $L'$ is recursive.

Hint: use Exercises 1 and 2.

**Exercise 4.** Consider the following two languages.

$$\begin{aligned} \text{Inf} &= \{\langle M \rangle : L(M) \text{ is infinite}\}; \\ \text{Tot} &= \{\langle M \rangle : M \text{ halts on all inputs}\}. \end{aligned}$$

(a) Give a reduction from Inf to Tot.

(b) Give a reduction from Tot to Inf.

(c) Using the generalized/second Rice's Theorem (a.k.a. Rice–Shapiro Theorem), show that neither Inf nor its complement $\overline{\text{Inf}}$ are recursively enumerable.

(d) Using items (a) and (c), show that neither Tot nor its complement $\overline{\text{Tot}}$ are recursively enumerable.

(e) Using the Rice–Shapiro Theorem directly (i.e., without using the previous items), show that neither Tot nor its complement $\overline{\text{Tot}}$ are recursively enumerable.