CS172 Computability & Complexity, Spring 2021

Homework 6

Out: 5 Mar. Due: 12 Mar.

Instructions: *Submit your solutions in pdf format on Gradescope by* **5pm on Friday, March 12**. *Solutions may be written either in LaTeX (with either machine-drawn or hand-drawn diagrams) or* **legibly** *by hand. (The LaTeX source for this homework is provided in case you want to use it as a template.) Please be sure to begin the solution for each problem on a new page, and to tag each of your solutions to the correct problem! Per course policy, no late solutions will be accepted. Take time to write* **clear** *and* **concise** *answers; confused and long-winded solutions may be penalized. You are encouraged to form small groups (two to four people) to work through the homework, but you* **must** *write up all your solutions on your own. Depending on grading resources, we reserve the right to grade a random subset of the problems and check off the rest; so you are advised to attempt all the problems.*

1. A *Turing enumerator* for a language $L \subseteq \{0,1\}^*$ is a multi-tape Turing machine, one of whose tapes is designated as a special "output" tape. Initially, all tapes are blank and all tape heads at the left-hand ends of their tapes. The Turing machine is required to write on its output tape an infinite sequence of the form $w_0\#w_1\#w_2\#\ldots$, where each w_i is a binary string, such that the following properties hold:

(i) Each w_i belongs to L.

(ii) For every $w \in L$, there is some i for which $w_i = w$ (i.e., every string in L eventually appears on the output tape).

(Note that the strings w_i do *not* have to be distinct.) Prove the following:

(a) L is recursively enumerable (Turing-recognizable) if and only if there exists a Turing enumerator for L. [NOTE: You need to prove two directions here. For the "only if" direction, you need to think how to run a TM on all possible inputs in a "fair" way. Think about our proof in class that the set of *pairs* of natural numbers is countable.]

(b) L is recursive (decidable) if and only if there exists a Turing enumerator for L with the additional property that the output strings $w_0, w_1, w_2, \ldots$ are in lexicographic order. [NOTE: This is a bit easier than part (a).]

[NOTE: Part (a) above explains the terminology "enumerable" in the phrase "recursively enumerable." The term "recursive" comes from the theory of recursive functions, which is a mathematical model of computation equivalent to the Turing machine. So "recursively enumerable" means that there is a TM that enumerates the strings in the language.]

2. Let L be a language over the binary alphabet. Show that L is Turing-recognizable (r.e.) if and only if there exists a decidable language L' such that

$$L = \{x : \exists y \in \{0,1\}^* \text{ s.t. } (x,y) \in L'\}.$$

[NOTE: We can think of the string y in the above definition as a *witness* to the fact that $x \in L$: this is because, given y, we can *decide* whether $x \in L$, even if we can't decide this without y because L is only r.e. Note that this is an "if and only if" statement, so you need to prove both directions separately!]

[continued on next page]

3. Consider the problem of determining whether, given two Turing machines, there is a string which is accepted by both of them. This problem corresponds to the following language:

$$L_{\text{int}} = \{\langle M_1, M_2\rangle : L(M_1) \cap L(M_2) \neq \emptyset\}.$$

(a) Show that L_{int} is Turing-recognizable (r.e.).

(b) By performing a mapping reduction from A_{TM}, show that L_{int} is undecidable.

(c) Now consider the language

$$L_{\text{nint}} = \{\langle M_1, M_2\rangle : L(M_1) \cap L(M_2) = \emptyset\}.$$

corresponding to the question of whether two given Turing machines accept *no* common strings. Prove, using parts (a) and (b) and *without* using an additional reduction, that L_{nint} is not Turing-recognizable (r.e.).

4. A software company markets a new product which they call the POC, or "Perfectly Optimizing Compiler." This product claims to have the following specification: Given as input a piece of Java code, it returns another piece of Java code that has exactly the same input-output behavior as the original code, and has the minimum possible number of lines among all pieces of code with this behavior. (The small print also stipulates that, in case there is more than one minimum-length piece of code, the POC outputs the lexicographically first one.)

Your boss asks you to purchase this product and put it to use in your company's development department as soon as possible. Write a short memo to your boss explaining why you are skeptical about the product and would not buy it.

[NOTES: You should refer to the undecidability of one of the problems we have seen in class, and also to the Church-Turing Thesis. Your memo should be rigorous and precise, but not technical or formal.]