Homework 6 – Due *Friday*, March 19, 2021 at 11:59 PM

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write "Collaborators: none" if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Note You may use various generalizations of the Turing machine model we have seen in class, such as TMs with two-way infinite tapes, stay-put, or multiple tapes. If you choose to use such a generalization, state clearly and precisely what model you are using.

Problems There are 5 required problems and 1 bonus problem.

1. (**Code as data**) Consider the following description of a three-tape TM H.

Algorithm: $H(\langle M, w\rangle)$

Input : Encoding of a basic TM M and a string $w \in \{0,1\}^*$

1. Copy the the string w to tape 2.
2. Repeat the following three steps forever:
3. Simulate M for one step on tape 2.
4. Erase the contents of tape 3. Copy the contents of tape 2 to tape 3, and check if the substring "1010" appears on tape 3. If it does, *accept*. Otherwise, continue.
5. If M halts (in either an accept or reject state), *reject*. Otherwise, continue.

(a) Let M_1 be the following (uninteresting) TM. Is $\langle M_1, \varepsilon\rangle \in L(H)$? Explain why or why not.

Algorithm: $M_1(x)$

Input : String $x \in \{0,1\}^*$

1. Write "010101" to the tape and *reject*.

(b) Let M_2 be the following TM. Is $\langle M_2, 101\rangle \in L(H)$? Explain why or why not.

Algorithm: $M_2(x)$

Input : String $x \in \{0,1\}^*$

1. Scan the input string x and (without writing anything) *accept* if the last symbol of x is 1. Otherwise, *reject*.

(c) What is the language $L(H)$ recognized by H?

(d) Is H a decider for the language $L(H)$? Explain why or why not.

2. (**Countable sets**)

   (a) `Whitespace` (`https://en.wikipedia.org/wiki/Whitespace_(programming_language)`) is a programming language in which valid programs consist only of spaces (S), tabs (T), and linebreaks (L). Show that the set $\mathcal{W}$ of all (finite) `Whitespace` programs is countable. (For the sake of readability, you can use the symbols S,T, L to stand for legal `Whitespace` characters in your proof.)

   (b) Show that if S and T are countable sets, then their union $S \cup T$ is also countable.

3. (**Uncountable sets**)

   (a) A game of Tetris is an infinite sequence of "tetrominos" which can have one of seven types $1, \ldots, 7$. For example, a Tetris game may look like $(7, 1, 4, 2, 2, \ldots)$. Use a proof by diagonalization to show that the set $\mathcal{T}$ of all Tetris games is uncountable.

   (b) An infinite sequence $(x_1, x_2, x_3, \ldots)$ is *collision-free* if no number appears more than once in that sequence. For example, $(1, 5, 2, 5, \ldots)$ is **not** collision-free because 5 appears at least twice. Use a proof by diagonalization to show that the set $\mathcal{C}$ of all collision-free infinite sequences of natural numbers is uncountable.

   Hint: When you construct a sequence contradicting the diagonal, make sure that it is indeed a member of $\mathcal{C}$.

4. (**Reduction Mad-Libs**) A language B is *excited* if every string in B takes the form ww for some $w \in \{0,1\}^*$. For example, $\{00, 1111, 1010\}$ is excited, but $\{00, 10\}$ is not excited. The language $EX_{\text{TM}} = \{\langle M \rangle \mid L(M) \text{ is excited}\}$ corresponds to the following computational problem: Given the encoding of a TM M, does M recognize an excited language? This exercise will walk you through a proof, by reduction, that EX_{TM} is undecidable.

   Assume, for the sake of contradiction, that EX_{TM} is decidable by a TM R. That is, there is a TM R that accepts $\langle M \rangle$ whenever $L(M)$ is excited, and rejects $\langle M \rangle$ whenever $L(M)$ is not excited. We will use R to construct a new TM S that decides the (undecidable) language A_{TM}.

Algorithm: $S(\langle M, w \rangle)$

Input : Encoding of a basic TM M over input alphabet $\{0,1\}$, string $w \in \{0,1\}^*$

1. Construct the following TM N:
   N = "On input a string $x \in \{0,1\}^*$:
   If $x = 00$, *accept*.
   Else, run M on input w. If it accepts, *accept*. Otherwise, *reject*."
2. Run R on input $\langle N \rangle$. If it accepts, __(i)__. Otherwise, __(ii)__.

   (a) Consider the machine N constructed inside algorithm S. If M accepts on input w, what is the language $L(N)$? Is $L(N)$ excited in this case?

   (b) If M does not accept on input w, what is the language $L(N)$? Is $L(N)$ excited in this case?

   (c) Fill in the blanks labeled (i) and (ii) with *accept* or *reject* decisions to guarantee the following conditions: If M accepts input w, then S accepts input $\langle M, w \rangle$, and if M does not accept input w, then S rejects input $\langle M, w \rangle$. Use parts (a) and (b) to explain why these conditions hold for your choices of how to fill in the blanks.

(Your job is done now, but you may want to keep reading to see how the proof concludes.) By part (c), the TM M exactly decides the language A_{TM}. But this language is undecidable, which is a contradiction. Hence our assumption that EX_{TM} was decidable is false, so we conclude that EX_{TM} is an undecidable language.

5. (**Unary acceptance**) A Turing machine M *accepts a unary string* if there exists a string $x \in \{1\}^*$ such that M accepts on input x. Consider the problem of determining whether (the encoding of) a TM M accepts a unary string.

   (a) Formulate this problem as a language UA. Caution: The only input to this computational problem is $\langle M \rangle$ for a TM M.

   (b) Consider the following "first attempt" at designing a TM N recognizing the language UA.

Algorithm: $N(\langle M \rangle)$
Input : Encoding of a basic TM M over input alphabet $\{0, 1\}$ 1. Let $x_n = 1^n$. For each $n = 0, 1, 2, 3, \ldots$: 2. Run M on input x_n. If it accepts, *accept*. Otherwise, continue.

   Explain why N **fails** to recognize UA.

   (c) Design and analyze a Turing machine that *does* recognize the language UA. Hint: Solved exercise 4.5 in Sipser illustrates the "dovetailing" trick that will be useful here.

   (d) Prove that the language UA is undecidable.
   Hint: Give a reduction from the undecidable language A_{TM}. That is, you should assume for the sake of contradiction that UA is decidable. Then under this assumption, construct a TM deciding A_{TM}, prove that this decider is correct, and as a result conclude that your assumption that UA is decidable must have been false.

6. (**Bonus problem**) Define the language $XOR_{\text{TM}} = \{\langle M, w, v \rangle \mid M \text{ is a TM that accepts exactly one of the strings } w, v\}$. Prove that both XOR_{TM} and its complement $\overline{XOR_{\text{TM}}}$ are unrecognizable.