Computability and Complexity
Jacobs University Bremen
Dragi Kamov and Dushan Terzikj

Course: CO21-320352
March 12th, 2019

Homework 5

Exercise 1

Solution:
Denote the halting language $H = \{< M >; x | M(x) \neq \nearrow\}$.

For any word $< N >; x$ with $Code(< N >)$ and $x \in \Sigma^*$, where $\Sigma = \{0, 1, \#\}$, we can construct a TM $K_{<N>;x}$ which for any input $y \in \Sigma^*$ will first write y on its input tape then a separator (can be indicated with some word $w \in \Sigma^*$) and then x. $K_{<N>;x}$ then simulates $N(x)$, which should yield the same y, but on x part of the tape.

If the simulation reaches a halting configuration of N, then $K_{<N>;x}$ enters a subroutine where it checks whether the y and x part are the same, replacing every letter on both sides of the separator with ␣ if there is a match in the letters. If there is no match, then $K_{<N>;x}$ never halts. If $K_{<N>;x}$ reaches $\triangleright$ then it halts.

From this description it is clear that $K_{<N>;x}$ halts on all inputs iff $< N >; x \in H$. The reduction map here maps $< N >; x$ on $K_{<N>;x}$. $\square$

Exercise 2

Solution:
References: "Models Of Computation Exploring the Power of Computing by John E. Savage"

In order to show that L is not recursively enumerable, we need to prove that a machine accepting such language doesn't exist. Assuming that there is a machine that accepts L, we would create a contradiction by saying that that TM M_L that decides L and that suggesting the existence of another TM M_H that solves the halting problem.

If we were given a code function $< M >$ for a TM M and an input w, the TM M_H writes that w on the tape and when the tape is empty, causes a halt.

The language that is accepted by T(M, w) includes the $\emptyset$ iff M halts on the word w. Thus, that TM is deciding the halting problem, and that is not possible because the TM M_L doesn't exist.
$\implies$ L is not recursively enumerable.