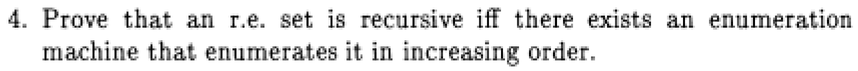
CSCI 3434, CU Boulder **Prof. Rafael Frongillo**
Fall 2020 **Problem Set 9**
Collaborators: None **Student: Alex Book**

- **Do not use external sources** beyond the materials linked to on the course website to solve these problems.
- You are encouraged to work with ($\leq$ 2) other students, but you must **write your solutions independently**.
- Be sure to **list your collaborators** by name clearly at top of your submission, or "no collaborators" if none.
- Recall: letters correspond to the section of the book: [H]omeworks starting page 301, [M]iscellaneous [E]xercises starting page 315.

1. Turing machine constructions (S20)

H9.4

Hint: take careful note of whether the set is finite vs infinite.


4. Prove that an r.e. set is recursive iff there exists an enumeration machine that enumerates it in increasing order.

If an r.e. set is recursive, there exists an enumeration machine that enumerates it in increasing order.
Given some recursive set S, we know there must be some total TM M that accepts on every input from S. Now consider the construction of another TM N. We give N input by feeding it every string in Σ^* in alphanumeric order (not necessarily just letters and numbers, but including any possible input characters; think of how a language like Python would order strings). Run M on each input. If M accepts, we will enumerate that input. If M rejects, we will not enumerate that input. Therefore, since M accepts all inputs in S and rejects all inputs outside of S, N will enumerate all inputs from the set S in increasing order.

If there exists an enumeration machine that enumerates an r.e. set in increasing order, that set is recursive.
Consider some enumerator N that enumerates an r.e. set S in increasing order. Now consider the construction of another TM M. On input q, M runs N until the string q is seen/printed, until something "greater" than q (after q in proper order) is seen/printed, or until it halts (whichever comes first). If the enumerator prints q, accept q. If the enumerator prints something larger than q before printing q, reject q. If the enumerator

halts before printing q, that must mean S doesn't contain q, so reject q.
Since M either accepts or rejects for every input (and accepts on every input from S), it is a total machine with S = L(M), so S is therefore recursive.

2. Classification: Computability (S22)

Classify the following languages as below, and justify your answer:

- (RE) recursive enumerable (r.e.),
- (CO-RE) co-r.e. (i.e. the complement is r.e.),
- (BOTH) both r.e. and co-r.e., or
- (NEITHER) neither r.e. nor co-r.e.

Note this means showing two things each: L is r.e. or not, and $\sim L$ is r.e. or not.

a. $L = \{M\#x\#y \mid M \text{ halts on } x \text{ and loops on } y\}$
(M halts on x) is r.e. and (M loops on y) is not r.e. Therefore, it is contradictory to try to create some TM that recognizes when both of these are true. Even if we were to construct a two-tape TM, the tape with the looping problem on it is not r.e., so L **itself must not be r.e.**

$\sim L = \{M\#x\#y \mid M \text{ loops on } x \text{ or halts on } y\}$
(M loops on x) is not r.e. and (M halts on y) is r.e. Say we construct a two-tape TM. If the tape with the halting problem on it doesn't accept, we know that the tape with the looping problem on it must (as it is an "or" construction). However, the looping problem is not r.e., so we can't say whether or not that tape will accept. Therefore, this constructed TM M won't always recognize an accept, so $\sim L$ is not r.e. Therefore, L **is not co-r.e.**

b. $L = \{M\#x\#y \mid M \text{ accepts } x \text{ and rejects } y\}$
(M accepts x) and (M rejects y) are individually r.e. If we "and" them, we can essentially create a two-tape TM and timeshare between the two tapes, and we get that L **is r.e.**

$\sim L = \{M\#x\#y \mid M \text{ rejects or loops on } x \text{ or accepts or loops on } y\}$
(M rejects or loops on x) and (M accepts or loops on y) are individually not r.e., because there is no way to know if M will halt or just keep on looping for the inputs x and y. Essentially, you don't know if M will eventually accept on input x or reject on input y, or if it will keep on looping. We need to know if it is going

to halt in a certain way (so the halting problem doesn't apply here) or if it will loop. You are basically dealing with some variation of the looping problem, which is not r.e. **Therefore, L is not co-r.e.**