CSCI 3434, CU Boulder **Prof. Rafael Frongillo**
Fall 2020 **Problem Set 10**
Collaborators: Frank Stinar **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 (≤ 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.

1. 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. $L = \{M\#x \mid \text{on input } x,\ M \text{ never moves its head to the left}\}$

i. r.e.
On any input string, if M halts before reaching the end of the input and without moving left, we accept. Otherwise, we count the number of steps taken to reach the end of the input. If this number is greater than the length of the input itself, there must have been a movement to the left, so we reject. If the number is equal to the length of the input, we then proceed to then count the number of steps taken in reading the infinitely many blank tape squares that exist after the input. If we reach a number of steps equal to the number of states in our finite control without moving left, we accept (since we know that there are no longer any possible states that could move the read/write head to the left). **Therefore, L is r.e.**

ii. co-r.e.
$\sim L = \{M\#x \mid \text{on input } x,\ M \text{ moves its head to the left at least once}\}$
On any input string, we run M on x infinitely until M moves its head left. If it does, we accept. Otherwise we infinitely loop. So, $\sim L$ is r.e. **Therefore, L is co-r.e.**

b. $L = \{M \mid M \text{ accepts exactly 3 inputs}\}$

i. r.e.
This is essentially comparable to the looping problem, as once 3 inputs have been accepted, we would need to know that the machine would loop forever without ever accepting another input. **Therefore, L is not r.e.**

ii. co-r.e.
$\sim L = \{M \mid M \text{ accepts some number of inputs other than 3}\}$
Accepting if M accepts more than 3 inputs is okay, as we would simply loop infinitely until a 4th input is accepted, then accept. A problem arises, however, when one tries to accept if M accepts less than 3 inputs. This situation is comparable to the looping problem. If the number of inputs accepted is less than 3, we would need to know that the machine will loop forever and not accept any more than 2 inputs. Because of this, we wouldn't be able to guarantee for every case that less than or greater than 3 inputs would be accepted (since the less than 3 inputs option is analogous to the looping problem). So, $\sim L$ is not r.e. **Therefore, L is not co-r.e.**

2. Complexity Reductions (S23)

See Canvas quiz.

Canvas quiz completed.

3. Classification: Complexity (S24)

For the following problem, figure out whether it is in P or NP-complete, and prove it. For P this means giving a poly-time algorithm. For NP-complete, this means three things: (1) showing it's in NP, (2) giving a reduction from an NP-complete problem we cover in class, and (3) showing that reduction runs in polynomial time.

a. A city is having issues with its $n \times m$ grid of street lights: each light is of type A, B, or C, but due to electrical problems, the city must shut off all lights of type C. Additionally, for each column of the grid, the city must shut off all lights of type A in that column or all lights of type B in that column. Given the type of each light in the grid, the city would like to decide whether it is possible to resolve the electrical problems while still leaving one light on in every row.

This is in NP, as checking a candidate solution only requires $O(n * m)$ time.

I will show a reduction from SAT to this problem. This is done as follows:

i. For every clause, create a row.
ii. For every unique variable, create a column.
iii. In order to fill in this created table, you look at the row-variable. If this variable "helps" satisfy the clause, mark it as an A (i.e. variable = A, clause = (A or B)). If this variable directly goes against satisfying the clause, mark it as a B (i.e. variable = A, clause = (not$A$ or B)). If the variable doesn't appear in any literal in the clause, mark it as a C (i.e. variable = D clause = (A or B)).

This creates a problem/table identical to one represented by the given problem (a grid with lights of type A, B, and C, where the appropriate criteria must be met).

This reduction runs in polynomial time, as the number of rows is equal to number of clauses, and the number of columns is equal to number of unique variables. Filling in this table would take $O(\text{rows} * \text{columns})$ time, which is polynomial time.