MLCS - Homework 3

Michele Conti
1599133

1 Computable, c.e. and not computable Problems

Exercise 1.2. Show by informal arguments the following points.

(1) If $L_1 \cap L_2$ is not computable and L_2 is computable then L_1 is not computable.

(2) If $L_1 \cup L_2$ is not computable and L_2 is computable then L_1 is not computable.

(3) If $L_1 \setminus L_2$ is not computable and L_2 is computable then L_1 is not computable.

(4) If $L_2 \setminus L_1$ is not computable and L_2 is computable then L_1 is not computable.

(5) If L_1 and L_2 are computably enumerable then $L_1 \cup L_2$ and $L_1 \cap L_2$ are computably enumerable.

Solution. The first four points can be proved proceeding by contradiction, slightly tweeking the rationale at each case.

Part 1:

Let's assume by contradiction that L_1 is computable. By hypothesis, we also know that L_2 is computable, therefore there exist two Turing Acceptors M_1 and M_2 such that, for $i = 1, 2$:

$$\text{if } x \in L_i \text{ then } M_i \text{ accepts } x; \text{ if } x \notin L_i \text{ then } M_i \text{ rejects } x.$$

Let's consider now a new Turing Acceptor M, defined as follows:

$$M(x) = \begin{cases} M_1(x) & \text{if } x \in L_1 \cap L_2 \\ M_1(x) & \text{if } x \in L_2 \setminus L_1 \\ M_2(x) & \text{if } x \in L_1 \setminus L_2 \\ M_2(x) & \text{if } x \notin L_1 \cup L_2 \end{cases}$$

where $M_1(x)$ and $M_2(x)$ are respectively the results of the computation of the Turing Acceptors M_1 and M_2 on the element x.

The existence of such machine is, by definition, equivalent to the fact that $L_1 \cap L_2$ is decidable, since the elements in $L_1 \cap L_2$ are going to be accepted by the machine M and all the elements outside $L_1 \cap L_2$ are going to be rejected. This is absurd by hypothesis. Therefore, we can conclude that L_1 is not computable.

Notice also that the for the first and last cases in the definition of M, we could have chosen either M_1 and M_2, getting the same results.

Part 2:

Similarly to the previous point, we can proceed by contradiction, assuming that L_1 is computable, and considering a new Turing Acceptor M:

$$M(x) = \begin{cases} M_1(x) & \text{if } x \in L_1 \\ M_2(x) & \text{if } x \in L_2 \\ M_2(x) & \text{if } x \notin L_1 \cup L_2. \end{cases}$$

The existence of this machine would be equivalent to the fact that $L_1 \cup L_2$ is computable, which again would be absurd. Therefore, we can conclude that L_1 is not computable.

Part 3:

Similarly to the previous points, we can proceed by contradiction, assuming that L_1 is computable. Again, we'll consider a new Turing Acceptor M, but we'll proceed in a slightly different way.

Since L_1 and L_2 are computable, given a point x they either accept or reject it. Therefore, we can define the new Turing Acceptor based on the behaviour of the two Acceptors:

$$\begin{cases} M \text{ accepts } x & \text{if } M_1 \text{ accepts } x \text{ and } M_2 \text{ rejects } x \\ M \text{ rejects } x & \text{otherwise.} \end{cases}$$

This machine would accept every element in $L_1 \setminus L_2$, and would reject everything else, therefore its existence would be equivalent to the fact that $L_1 \setminus L_2$ is computable. This is absurd, and thus we can conclude that L_1 is not computable.

Part 4:

We can proceed in a specular way to the previous point. Proceeding by contradiction, we assume that L_1 is computable. Let's consider the new Turing Acceptor M:

$$\begin{cases} M \text{ accepts } x & \text{if } M_1 \text{ rejects } x \text{ and } M_2 \text{ accepts } x \\ M \text{ rejects } x & \text{otherwise.} \end{cases}$$

This machine would accept every element in $L_2 \setminus L_1$, and would reject everything else, therefore its existence would be equivalent to the fact that $L_2 \setminus L_1$ is computable. This is absurd, and thus we can conclude that L_1 is not computable.

Part 5:

For this last point, we'll have to change our strategy. Both implications ($L_1 \cup L_2$ is computable and $L_1 \cap L_2$ is computable) can be proved using similar reasonings, but we'll analyze both of them separately.

Let's start by proving that $L_1 \cup L_2$ is computably enumerable. Consider the two Turing Machines M_1 and M_2 and a point x. Now, if we choose one of the two machines, we can express the computation of x as a collection of snapshots, and therefore we can analyze the whole computation step by step. Let's now consider this mechanism for both machines simultaneously: we first compute one computation step of the machine M_1, then we compute one computation step of the machine M_2, and we cyclically continue using this strategy. There are now two cases. If $x \notin L_1 \cup L_2$, then both machines would diverge, as both L_1 and L_2 are computably enumerable. On the other hand, if $x \in L_1 \cup L_2$, one of the two machines would eventually accept, and we can therefore stop the computation. This proves that $L_1 \cup L_2$ is computably enumerable.

Now to the second implication: $L_1 \cap L_2$ is computably enumerable. Following the same idea, we can again analyze the machines behaviour in a step-by-step fashion. Now, if $x \notin L_1 \cup L_2$, both machines would still diverge. On the other hand, we would stop the computation only if both the machines accepts at a certain time step. This would clearly never happen if $x \in L_1 \setminus L_2$ or if $x \in L_2 \setminus L_1$ (since one of the two machines would continue the computation forever), and would certainly happen on the intersection $L_1 \cap L_2$, since both machines are computably enumerable. This proves that $L_1 \cap L_2$ is computably enumerable.

$\square$

2 *NP* problems