MLCS - Homework 3

Michele Conti
1599133

1 Computable, c.e. and not computable Problems

<u>Exercise</u> 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.

<u>Solution</u>. 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:

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

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. But this is absurd by hypothesis. Therefore, we can conclude that L_1 is not computable.

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:

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

The existence of this machine would be equivalent to the fact that $L_1 \cup L_2$ is computable, since the elements in $L_1 \cup L_2$ are going to be accepted by the machine M and all the elements outside $L_1 \cup L_2$ are going to be rejected, which, again, would be an 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:

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

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.

□

2 *NP* problems

Exercise 2.3. Let's call a formula a formula in Universal Second Order Logic if it has the form $\forall R_1 \dots \forall R_n F$, where F is a first-order formula. Show that the property of being a connected graph is expressible by a Universal Second Order formula using only quantification on relations $R_1, \dots, R_n$ of arity 1.

(Hint: start by defining the negation of connectivity).

Solution. As the author suggests, we can start by defining the formula expressing the negation of connectivity for a graph.

A graph is not connected if it exists a proper subset of the set of vertices of the graph (i.e., at least one node of the graph belongs to this subset, and at least one doesn't) such that, however we choose two nodes, if it exists an edge between them, then they either both belong to this subset, or they both don't.

We can express this using an Existential Second Order formula, as follows:

$$\exists R(\exists x(R(x)) \land \exists y(\neg R(y)) \land \forall x \forall y(E(x,y) \to (R(x) \leftrightarrow R(y))))$$

where $R(x)$ informally expresses the fact that the point x is in a certain subset of the set of nodes of the graph, while E is the usual edge relation between two nodes.

Now, we can negate the above formula, obtaining a Universal Second Order formula:

$$\forall R(\forall x(\neg R(x)) \lor \forall y(R(y)) \lor \exists x \exists y(E(x,y) \land ((R(x) \land \neg R(y)) \lor (\neg R(x) \land R(y))))).$$

$\square$