TOC PROBLEM SET-14

SIDDHANT CHAUDHARY
BMC201953

1. Consider

$$\text{FINITENESS} = \{\langle M\rangle \mid L(M) \text{ is finite}\}$$

Is FINITENESS RE? Is it co-RE?

Solution. We will show that FINITENESS is neither RE nor co-RE by reducing the language $\overline{\text{HALT}}$ to the languages FINITENESS and $\overline{\text{FINITENESS}}$. This will do our job, because we already know that $\overline{\text{HALT}}$ is *not* RE, since HALT is RE and undecidable.

First, let us reduce $\overline{\text{HALT}}$ to FINITENESS, i.e we show $\overline{\text{HALT}} \leq$ FINITENESS. Let M_0 be any Turing Machine such that $L(M_0) = \phi$, and hence $M_0 \in$ FINITENESS. We will now describe our computable function σ. Suppose a word $\langle M\rangle\#w$ is given. If either $\langle M\rangle$ is not a valid encoding of a Turing Machine, or if w is not a valid encoding of a word over M's input alphabet, then we put $\sigma(\langle M\rangle\#w) = M_0$ (notice that in this case it is clear that $\langle M\rangle\#w \in \overline{\text{HALT}}$, and that is why we mapped it to M_0). So, suppose the encoding $\langle M\rangle\#w$ is valid. We construct a TM $\sigma(\langle M\rangle\#w) = N_{M,w}$ that does the following (M and w are hard-wired in the encoding $N_{M,w}$)

(1) On an input y, $N_{M,w}$ *ignores* y completely.
(2) $N_{M,w}$ then writes the word w on its tape.
(3) Finally, $N_{M,w}$ simulates the machine M on the word w. $N_{M,w}$ accepts if M halts on w.

Observe that if M halts on w, then $L(N_{M,w}) = \Sigma^*$. Moreover, if M does not halt on w, then $L(N_{M,w}) = \phi$. Clearly, Σ^* is infinite and ϕ is finite. So from the above, it follows that

$$\langle M\rangle\#w \in \overline{\text{HALT}} \iff N_{M,w} \in \text{FINITE}$$

Thus, σ is a valid reduction, and this shows that FINITENESS is *not* RE.

Next, we reduce $\overline{\text{HALT}}$ to $\overline{\text{FINITENESS}}$, i.e we show $\overline{\text{HALT}} \leq \overline{\text{FINITNESS}}$. The idea here will be a little more involved. Let M_1 be any Turing Machine accepting the language Σ^*. We will now describe our computable function σ. So, suppose the word $\langle M\rangle\#w$ is given. As before, if either $\langle M\rangle$ is not a valid encoding of a Turing Machine, or if w is not a valid encoding of a word over M's input alphabet, then we put $\sigma(\langle M\rangle\#w) = M_1$ (notice that in this case it is true that $\langle M\rangle\#w \in \overline{\text{HALT}}$, and that is why we mapped it to M_1). So, suppose the encoding $\langle M\rangle\#w$ is valid. We construct a TM $\sigma(\langle M\rangle\#w) = N_{M,w}$ that does the following (and as before, M and w are hard-wired in the encoding $N_{M,w}$)

(1) On input y, $N_{M,w}$ writes y on a separate tape.
(2) On another tape, $N_{M,w}$ simulates the machine M on the word w for $|y|$ steps. The machine $N_{M,w}$ accepts y if M does not halt on w within $|y|$ steps, otherwise it rejects.

Date: November 2020.

Now observe the following. If M halts on w, then we see that

$$L(N_{M,w}) = \text{all words of length less than the halting time of } M \text{ on } w$$

i.e $L(N_{M,w})$ is finite. On the other hand, if M *does not* halt on w, then

$$L(N_{M,w}) = \Sigma^*$$

which is infinite. So we have shown that

$$\langle M\rangle\#w \in \overline{\text{HALT}} \iff N_{M,w} \in \overline{\text{FINITENESS}}$$

Thus, σ is a valid reduction, and this shows that $\overline{\text{FINITENESS}}$ is *not* RE either. This completes the solution. ■

2. Consider the following problem discussed in class, known as Intersection Non-Emptiness for CFGs.

$$\text{INE} = \{(G_1, G_2) \mid G_1, G_2 \text{ are CFGs}, L(G_1) \cap L(G_2) \neq \phi\}$$

Is INE RE? Is it co-RE?

Solution. It is easy to see that INE is RE. We can give an easy description of a TM K that accepts the language INE. We can have a Turing Machine K that enumerates words of Σ^* one-by-one in length-lexicographic order, and for each enumerated word w, K checks whether $w \in L(G_1)$ and $w \in L(G_2)$ using the CYK algorithm. It is then clear that K accepts the language INE, however K is not a total Turing Machine.

We will now show that the language INE is actually undecidable by reducing PCP to it (PCP:Post's Correspondence Problem), i.e we will show that

$$\text{PCP} \leq \text{INE}$$

Because PCP is undecidable, this will show that INE is undecidable, and therefore this will show that INE is *not* co-RE because above we have shown that it is RE.

Here is the reduction (recall that in PCP, the input is pairs $(u_1, v_1), (u_2, v_2), ..., (u_k, v_k)$ of words). Suppose we are given the input $(u_1, v_1), (u_2, v_2), ..., (u_k, v_k)$. Our computable function σ will be as follows. Let $\sigma((u_1, v_1), ..., (u_k, v_k)) = (G_1, G_2)$, where G_1 is the CFG

$$S \to 1Su_1 \mid 2Su_2 \mid ... \mid kSu_k \mid 1u_1 \mid 2u_2 \mid ... \mid ku_k$$

and G_2 is the CFG

$$T \to 1Tv_1 \mid 2Tv_2 \mid ... \mid kTv_k \mid 1v_1 \mid 2v_2 \mid ... \mid kv_k$$

It is easy to see that

$$L(G_1) = \{a_1a_2...a_nu_{a_n}u_{a_{n-1}}...u_{a_1} \mid a_1a_2...a_n \in \{1, ..., k\}^* \setminus \{\epsilon\}\}$$
$$L(G_2) = \{a_1a_2...a_nv_{a_n}v_{a_{n-1}}...v_{a_1} \mid a_1a_2...a_n \in \{1, ..., k\}^* \setminus \{\epsilon\}\}$$

and this can be easily seen by the nature of productions of the given grammars.

Now, I will show that

$$(u_1, v_1), (u_2, v_2), ..., (u_n, v_n) \in \text{PCP} \iff L(G_1) \cap L(G_2) \neq \phi$$

First, suppose there is a PCP solution to this input, i.e there is some word $a_1a_2...a_n \in \{1, ..., k\}^* \setminus \{\epsilon\}$ such that

$$u_{a_1}u_{a_2}...u_{a_n} = v_{a_1}v_{a_2}...v_{a_n}$$

Then, we have

$$a_n a_{n-1} \ldots a_1 u_{a_1} u_{a_2} \ldots u_{a_n} = a_n a_{n-1} \ldots a_1 v_{a_1} v_{a_2} \ldots v_{a_n} \in L(G_1) \cap L(G_2)$$

so that $L(G_1) \cap L(G_2) \neq \phi$. Conversely, suppose $L(G_1) \cap L(G_2) \neq \phi$. So, there is some word $a_1 a_2 \ldots a_n \in \{1, \ldots, k\}^* \setminus \{\epsilon\}$ such that

$$a_1 a_2 \ldots a_n u_{a_n} u_{a_{n-1}} \ldots u_{a_1} = a_1 a_2 \ldots a_n v_{a_n} v_{a_{n-1}} \ldots v_{a_1}$$

So, we see that

$$u_{a_n} u_{a_{n-1}} \ldots u_{a_1} = v_{a_n} v_{a_{n-1}} \ldots v_{a_1}$$

and hence $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ has a PCP solution. So, σ is a valid reduction, and hence INE is undecidable, because PCP is undecidable. So, INE is RE and not co-RE. ■

3. Consider

$$\text{AMBIGUOUS} = \{G \mid G \text{ is an ambiguous CFG}\}$$

Is AMBIGUOUS RE? Is it co-RE?
Hint: You may use the fact that PCP is RE but not co-RE.

Solution. I claim that AMBIGUOUS is RE but not co-RE. To prove that AMBIGUOUS is RE, we can give an easy description of a TM accepting AMBIGUOUS. So let K be a TM that does the following on input G:

(1) K enumerates each natural number $n \in \mathbb{N}$ one by one on a separate tape.
(2) For each natural number n enumerated, K then finds all *left-most* derivations of length n in G and writes the derivations on a separate tape, where each derivation is separated by a delimiter like #. For each of these derivations, K then checks whether the derivation derives a word of Σ^*, and if it does then K writes this word on a separate tape. Each of these words written will be separated by a delimiter like #.
(3) K then goes through each of the written words, and checks whether two words are the same. If they are, then K accepts. Otherwise, K erases everything and goes back to step (1).

It is clear that this TM K accepts the language AMBIGUOUS. So, AMBIGUOUS is RE.

Next, we will show that AMBIGUOUS is undecidable by reducing PCP to it. This will automatically show that AMBIGUOUS is *not* co-RE, since we have already shown that it is RE. The reduction is as follows. Let the input $(u_1, v_1), \ldots, (u_k, v_k)$ to PCP be given. Then, consider the following grammar.

$$\begin{aligned} S &\to A \mid B \\ A &\to u_1 A 1 \mid u_2 A 2 \mid \ldots \mid u_k A k \mid u_1 1 \mid u_2 2 \mid \ldots \mid u_k k \\ B &\to v_1 B 1 \mid v_2 B 2 \mid \ldots \mid v_k B k \mid v_1 1 \mid v_2 2 \mid \ldots \mid v_k k \end{aligned}$$

By the nature of the productions, it is clear that

$$L(S) = \{u_{a_1} u_{a_2} \ldots u_{a_n} a_n a_{n-1} \ldots a_1 \mid a_1 \ldots a_n \in \{1, \ldots, k\}^* \setminus \{\epsilon\}\}$$
$$\cup$$
$$\{v_{a_1} v_{a_2} \ldots v_{a_n} a_n a_{n-1} \ldots a_1 \mid a_1 \ldots a_n \in \{1, \ldots, k\}^* \setminus \{\epsilon\}\}$$

Moreover, observe that the only ambiguity in S comes from the production $S \to A \mid B$. Now, suppose there is a PCP solution to the input $(u_1, v_1), ..., (u_k, v_k)$. So, there is some $a_1...a_n \in \{1, ..., k\}^* \setminus \{\epsilon\}$ such that

$$u_{a_1}...u_{a_n} = v_{a_1}...v_{a_n}$$

which implies that

$$u_{a_1}...u_{a_n}a_n...a_1 = v_{a_1}...v_{a_n}a_n...a_1$$

and hence the word $u_{a_1}...u_{a_n}a_n...a_1$ has two distinct left-most derivations in the grammar S, i.e S is ambiguous. Conversely, if S is ambiguous, then there is some $a_1...a_n \in \{1, ..., k\}^* \setminus \{\epsilon\}$ such that

$$u_{a_1}...u_{a_n}a_n...a_1 = v_{a_1}...v_{a_n}a_n...a_1$$

because this word will have two left-most derivations, and the only difference will be the choice between $S \to A$ and $S \to B$. Hence, it follows that there is a PCP solution to the input $(u_1, v_1), ..., (u_k, v_k)$. What we have shown is that

$$(u_1, v_1), ..., (u_n, v_n) \in \text{PCP} \iff S \in \text{AMBIGUOUS}$$

so this is a valid reduction. Because PCP is undecidable, it follows that AMBIGUOUS is undecidable as well. So, we conclude that AMBIGUOUS is RE but not co-RE. ■