TOC PROBLEM SET-13

SIDDHANT CHAUDHARY
BMC201953

Problem 1. In the following, giving a high level description of the Turing Machines suffices.
(a) Consider the language $L = \{a^k \mid k = 3^n \text{ for some } n \geq 2\}$. Show that L is REC by giving a Turing Machine for L.

Solution. The algorithmic description of the Turing Machine is as follows.

(1) On the input $\vdash a^k$, first go towards the end of the input and put a right end-marker $\dashv$, so the tape contents become $\vdash a^k \dashv$. The head will now keep sweeping between these two end-markers.
(2) Check if the input contains *exactly* 9 $a's$ (this can be done by using 9 states). If there are exactly 9 a's, then accept. If there are fewer a's, then reject. If there are more a's, then move the head to the extreme left again and go to step (3).
(3) In this step, we start scanning from left to right, replacing *every third* a we see by a blank symbol $\sqcup$ (if you can't see a third a, then reject, because then then number of a's is not even divisible by 3) until we reach the right endmarker $\dashv$. Then, we scan from right to left, again replacing *every third* a by a blank symbol $\sqcup$ until we reach the left endmarker $\vdash$. Now go to step (2).

In step (3), if we had k a's originally where $3|k$, then we have reduced the number of a's to $k/3$ by doing two scans; one from left to right, and the other from right to left. This Turing Machine is clearly total, because no matter what the input is, the machine will always halt, and that is easy to see. So, L is REC. ■

(b) Consider the language Perfect Power $= \{a^k \mid k = p^n \text{ for some } p, n \geq 2\}$. Show that L is REC.

Solution. (Note: I am assuming that p is *not* a prime). Let me first explain the working of the TM. The TM will have 3 tapes: the first will contain the input a^k, the second tape will contain a^p for the *current* value of p (p will initially be 2 and will be incremented after every step), and the last tape will be used to check whether the length of a^k is a power of p.

(1) On the first tape, write the input a^k. This tape will never be changed. On the second tape, write the word $aa = a^2$ (so that $p = 2$ initially).
(2) Make the third tape fully blank, and copy the input a^k from the first tape onto the third tape. Suppose the word a^p is written on the second tape. First, check if $k < p$, and if it is, then reject. Then just as in part **(a)**, put a right-endmarker $\dashv$ at the end of the word a^k, so the third tape becomes $\vdash a^k \dashv$. Then do the same procedure on this tape as in part **(a)**; replace every p^{th} a (where a^p is the current word on the second tape) and repeat

Date: November 2020.

the same procedure as in part **(a)** to check if the length is a power of p. Accept if it is a power of p. If it is not a power of p, go to step (3).

(3) On the second tape, add an a at the end of the current word, so the new word becomes a^{p+1} where a^p was the old word (this is incrementing p by one). Go to step (2).

This is clearly a Total Turing Machine because every word's run will be finite; there are no loops, which is easy to see. ■

Problem 2. Define EMPTY $= \{\langle TM\rangle \mid L(TM) = \phi,\ TM$ is a Turing Machine$\}$. Show that EMPTY is an undecidable language. Is it RE? co-RE? Explain.

Solution. First, we will show that EMPTY is undecidable by *reducing* it to HALT, the halting problem.

For the sake of contradiction, suppose EMPTY is decidable. So, there is some TTM K which accepts the language

$$\{\langle TM\rangle \mid L(TM) = \phi\}$$

Now we construct a TTM K' for HALT as follows.

(1) Suppose the given input is $M\#x$. The machine K' first checks whether M is a valid encoding of a TM, and whether x is a valid encoding of an input over M's input alphabet. If either of these is false, K' rejects.
(2) K' *constructs* a new machine N as follows (by *constructs*, I mean K' writes the encoding of a new machine N on its tape). On any given input y, N completely *ignores* y and writes x on its tape. Then, N just simulates M on x. Note that x and M will be somehow hard-wired in N's encoding. Also, N accepts if M halts on x.
(3) Finally, K' simulates K on the input N. If K accepts N, then K' will reject and if K rejects N then K' will acccept.

Note that in step (2) above, the machine N satisfies either $L(N) = \phi$ or $L(N) = \Sigma^*$, the latter case being true if M halts on x. In simple words, $L(N) = \phi$ if and only if M does *not* halt on x. So, it follows that K' is a TTM that decides HALT, which is a contradiction because HALT is undecidable. So, our hypothesis must be false, and hence EMPTY is undecidable.

Now, we show that EMPTY is co-RE. This will automatically show that EMPTY is *not* RE, because if a language is both RE and co-RE then it is REC, and in our case EMPTY is *not* REC. So, we will make a TM M that accepts the language

$$\{\langle TM\rangle \mid L(TM) \neq \phi\}$$

The basic idea is this: suppose the input $\langle TM\rangle$ is given; we will try to run TM on *all* words in Σ^*, and see whether it accepts any word. It may be possible that TM loops on some word, so to deal with that, we will run TM *parallelly* on multiple words. Intuitively, this is done as follows (suppose $\Sigma = \{a, b\}$): take the word ϵ, do one step of TM on this word. Then take a new word a, and do one step of TM on each of ϵ and a. Then take a new word b, and do one step of TM on each ϵ, a and b. This way, by doing one step at a time on many words, we can simulate TM parallelly on multiple words.

The TM M will have multiple tapes. The first tape will contain the encoding $\langle TM\rangle$. The second tape will contain the words on which TM is currently running. The third tape will the configuration of the runs of the words on which

TM is currently running. These configurations will be separated by a delimiter like #.

(1) On input $\langle TM \rangle$, M first checks whether $\langle TM \rangle$ is a valid encoding of a TM. If it is not, then M rejects. M then writes the word ϵ on its second tape (this is the smallest word in Σ^*).

(2) M checks all the words on its second tape; if a new word is found, M makes a computation slot (separated from the other slots by the delimiter #) for this new word. Then, M scans through all the slots in the third tape, simulating *one step* of TM on each slot. If any slot reaches the accept state of TM, then M accepts.

(3) M *adds* new words of Σ^* to its second tape (which will also be separated by some delimiter) by simply adding each letter of Σ at the end of each of the old words on the second tape. Having produced new words, M goes to step (2).

Clearly, we can see that $L(M) = \{\langle TM \rangle \mid L(TM) \neq \phi\}$. So, it follows that EMPTY is co-RE and undecidable. ■

Problem 3. (Difficult,more to discuss/think about) Show that decidable languages are not closed under homomorphisms.

Hint: Give a homomorphism h and a decidable language L such that $h(L) =$ HALT.

Solution. To be completed. ■