Name :- Sumit Kumar Yadav   Roll No. 18CS30042

Solution 2 :-

Given a set A, $A^R = \{ \omega^R \mid \omega \in A \}$ where ω^R denotes the reverse off the string ω.

⇒ Let's consider a P to be a property on r.e. sets which is defined as

$$P(A) = \begin{cases} T & \text{if } A = A^R \\ F & \text{otherwise} \end{cases}$$

So, we can say by using Rice theorem, we consider languages over Σ of size > 1 for otherwise the problem is trivally decidable.

Now, Trivially $P(\phi) = T$.

also $P(\{a_1 a_2\}) = F$ for some set $a_1, a_2 \in \Sigma$ with $a_1 \neq a_2$ since $(a_1 a_2)^R = a_2 a_1 \notin L(N)$.

Now, we have exhibited two sets one for which the P holds and the other for which it does not. It follows that P is a non-trivial property and hence undecidable by rice theorem.

Solution 5 :-

Let's consider that A of minimal TM's over a fixed alphabet ie $\{0,1\}$ is an infinite r.e. set.

Assumption:-

I only consider the TMs over input alphabet $\{0,1\}$ and stack alphabet $\Gamma = \Sigma \cup \{\vdash, \sqcup\}$

Now, then there exists a machine N that enumerates A.

For a Turing machine M denote by $s(M)$ the number of states of M. Let K be a K be a machine that computes a map $\sigma : N \to N$ defined as : $\sigma(x)$ is the first machine J enumerated by N such that $s(J) > s(M_x)$ (where M_x denotes the TM with description i) on input x, K does the following

Also we can say that

→ construct M_x from x

→ use N to enumerate TMs in A

→ stop when a TM J with $s(J) > s(M_x)$ is enumerated. This event will occur eventually as there are only finitely many TMs with fewer states than M_x and A is infinite.

→ Output the description/ encoding of J.

K is a total TM and hence σ is a total recursive fn. Now, by recursion theorem there exists a fixed point x_0 such that $L(M_{x_0}) = L(M_{\sigma(x_0)})$: but $M_{\sigma(x_0)} \in A$ is a minimal TM for $L(M_{x_0})$ and yet $s(M_{x_0}) < s(M_{\sigma(x_0)})$. This contradicts the minimality of $M_{\sigma(x_0)}$ and consequently our assumption that A is an indefinite r.e. set.

Solution 3:-

(a) $L(G) = L(G)L(G)$.

$\overline{HP} \leq_m \{G \mid L(G) = L(G)L(G)\}$.

Now, make a reduction consider the input is $M \# w$ (an instance of $\overline{HP}$) and the output is a CFG G such that $L(G) = L(G)L(G)$ if and only if M does not halt on w. We take G to be a grammar for

$L = \overline{VALCOMP(M, w)}$.

Now, consider the 2 situations,

i) If M does not halt on w, then $VALCOMP(M, w) = \phi$ and so $L = L(G) = \Delta^*$ and we have $\Delta^* = \Delta^* \Delta^*$.

ii) If M halts on w, then $VALCOMP(M, w) \neq \phi$. Let $\gamma = \# C_0 \# C_1 \# C_2 \cdots \# C_N \#$ be a valid computation history M on w.

Take $\alpha = \# C_0$ and $\beta = \# C_1 \# C_2 \# \cdots \# C_N \#$. Then α is not a valid computation history of M on w because the string does not end with $\#$. Moreover β too is not a valid ~~conf~~ computation history of M on w, because the head is not at the leftmost cell in the first configuration. Therefore $\alpha, \beta \in L$ whereas $\gamma = \alpha\beta \notin L$. That is in this case $L(G)$ does not satisfy $L(G) = L(G) \cdot L(G)$.

(b) ∵ We have language $\{\langle G \rangle \mid G$ a CFG and $L(G)$ is ambiguous$\}$.

So, for that consider $\alpha_1, \ldots, \alpha_m$, $\beta_1, \ldots, \beta_m$.

construct the following CFG $G = (V, \Sigma, R, S)$

& $V = \{S, S_1, S_2\}$

Now, $R = \{S \rightarrow S_1 \mid S_2,\ S_1 \rightarrow \alpha_1 S_1 \sigma_1 \mid \cdots \mid \alpha_m S_1 \sigma_m \mid \alpha_1 \sigma_1 \mid \cdots \mid \alpha_m \sigma_m,$

$S_2 \rightarrow \beta_1 S_2 \sigma_1 \mid \cdots \mid \beta_m S_2 \sigma_m \mid \beta_1 \sigma_1 \mid \cdots \mid \beta_m \sigma_m\}$

where σ_i are new characters added to the alphabet eg, $\sigma_i = i$

Now if the language is ambiguous, then there is a derivation of some string w in 2 different ways. Supposing that the derivations both start with the rule $S \to S_1$ reading the new characters backwards until they end makes sure there can only be one derivation so that's not possible. Hence, we see that the only ambiguity can come from one S_1 and one S_2 start but then taking the substring of w up to the beginning of the new characters, we have a solution to the PCP.

Similarly if there is no ambiguity then the PCP cannot be solved, since a solution would imply an ambiguity that just follows $S \Rightarrow S_1 \Rightarrow^* \alpha \tilde{\sigma}$ and $S \Rightarrow S_2 \Rightarrow^* \beta \tilde{\sigma}$ where $\alpha = \beta$ are strings of matching α's and β's.

Hence we have reduced to PCP and since that's undecidable.

Solution 1 :-

$\because$ L(R) = {(M, x, p) | M on input x visits state p during the computation}.

Consider this problem is a state entry ~~problem~~ problem. state entry problem can be reduced to halting problem.

Now, construct a turing machine M with final state 'q'.

we run a turing machine R (for each state entry problem)

now consider input : M, q, w.

we give w as input to M.

Now, if M halts in the final state q then R accepts the input. so the given problem is partially decidable.

if M ~~halt~~ goes in an infinite loop then M can not output anything so R rejects the input. So, we can say that given problem becomes undecidable.

Solution 4 :-

(a) PCP over unary alphabet 1.

→ If the alphabet is unary the dominoes only differ in the number of 1s that each has ~~to~~ on the top and bottom. This specific case of PCP is solved easily by the following algo :

consider M · to be a collection of dominoes, now

→ if some dominoe has the same no. of 1s on the top and bottom, there is a trivial match so accept.

→ if all the dominoes have more than 1's on top than on bottom there is no possibility of a match so reject, likewise if all the dominoes have less 1's on top then on bottom reject.

→ find one dominoes with more 1's on top than on bottom (consider a difference of a 1s) and one dominoe with more ~~than 1s~~ 1s on bottom than on top (say a difference of b 1s) Now choosing b of the first dominoe and a of the second should make an equal number of 1s on both top and bottom and hence a match.

(b)