Page 285

3) Let L be a finite language. Show that L^+ is recursively enumerable.

$L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$

That is L^+ is all strings in L unioned w/ the set of all strings generated by concatenating any 2 strings from L unioned w/ the set of all strings generated by concatenating any 3 strings from L, etc.

For example, if $L = \{a, ba\}$ then

$$L^+ = \{\underbrace{a, ba}_{L^1}, \underbrace{aa, aba, baba, baa}_{L^2}, \underbrace{aaa, aaba, \ldots}_{L^3}\}$$

L^+ is recursively enumerable if there is a T.M. that can write every string in L^+.

To write every string in L^+ (i.e. to enumerate all strings in L^+) just do the above procedure:

write L, then L^2, then L^3, etc. each of these is finite so you will eventally write any given string.

So, for example, if you pick a string $w \in L^N$ (w is the concatenation of N strings from L) it will eventually be written.

5) Let L be a language that is <u>not</u> R.E.

Assume $\overline{L}$ is recursive. As discussed in class if $\overline{L}$ is recursive then L must be recursive as well. But if L is not R.E. its also <u>not</u> recursive - a contradiction, thus the assumption that $\overline{L}$ is recursive must be false.

6) Done in the text

8) First show that the family of recursively enumerable (R.E.) languages is closed under <u>union</u>.

Let L_1 and L_2 be two R.E. languages, then there exists Turing Machines M_1 and M_2 that can enumerate (i.e. list) the strings in L_1 and L_2 respectively. (Because L_1 and L_2 might not be recursive, it might not be possible to list the strings <u>not</u> in L_1 or L_2, but a T.M. can list the strings <u>in</u> L_1 and L_2.)

Now construct a new T.M. M_3 (based on a universal T.M.) first it emulates M_1 until a string is written, then it emulates M_2 until a string is written, then it emulates M_1 again and so forth. In this way M_3 will list all of the strings in $L_1 \cup L_2$ proving that the union is also R.E.

Next prove that <u>intersection</u> is closed. Use L_1, L_2 and $T_1 T_2$ as above

Again it is important to alternate between the languages.

First write a string from L_1 on the "working memory" then write a string from L_2, if they are the same write the string to the "answer tape" and erase them other wise leave them in "working memory". Write another string from L_1 and see if it matches any of the current strings from L_2, etc. Basically you use a new TM to alternatively add from L_1 and L_2 to a "pooled set" and check for matches.