Review: An enumerator is a modified multitape TM with a special tape (print tape) & designated q_{print} state.

If E is an enumerator, whenever E enters its print state (q_{print}), we say that E prints the string written on the print tape (its current nonblank contents) & that string is blanked out & the head on the print tape reset to the left end.

E takes no input (all tapes start blank) and E never halts (runs forever).

High level desc example:

E := "On no input:
1. —
2. —
3. If — then print x (x is some string)
⋮ "

don't accept or reject!

Recall: a lang L is T-rec if $L = L(M)$ for some TM, where
$L(M) := \{x \in \Sigma^* : M \text{ accepts } x\}$

Recall: If E is an enumerator, then
$L(E) := \{x \in \Sigma^* : E \text{ prints } x \text{ (at least once)}\}$

The language enumerated by E.

A lang L is enumerable (aka computably enumerable (c.e.) or recursively enumerable (r.e.)) if $L = L(E)$ for some enumerator E.

Thm: Let L be any language. L is enumerable if and only if L is T-rec.

Proof: ($\Rightarrow$) Suppose L is enumerable. Let let E be an enumerator such that $L = L(E)$. We define a TM recognizing L as follows:

M := "On input w:
1. Run E
2. If E ever prints w, then accept
[3. else loop]"

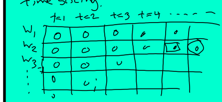
$\forall w$, E prints $w \Rightarrow M$ accepts w
E does not print $w \Rightarrow M$ loops on w

$\therefore$ M accepts $w \Leftrightarrow E$ prints w
$\therefore$ $L(M) = L(E)$
$\therefore$ L is T-rec (by M).

($\Leftarrow$) Suppose L is T-rec. Let M be a recognizer for L, i.e., M is a TM and $L = L(M)$. Define an enumerator E as follows:

E := "On no input:
1. Cycling through all strings $s_0, s_1, s_2, \ldots$
[2. For each s_i, if s_i does not encode a pair $\langle w, t\rangle$ where w is a string & t is a pos integer, then go on to the next string
3. Run M on input w for up to t steps.
4. If M accepts w is $\le t$ steps, then print w.
5. Go on to the next string."

E running multiple computations is called "dovetailing" or "time slicing".



$\forall w$, M does not accept w
$\Rightarrow M$ does not accept w within t steps, for any t
$\Rightarrow E$ never prints w.

M does accept w
$\Rightarrow M$ accepts w after t steps for some t.
$\Rightarrow E$ will print w when it examines the string $\langle w, t\rangle$
$\Rightarrow E$ prints w (because E eventually examines all pairs of the form $\langle \underset{\text{string}}{\_\_}, \underset{\text{pos int}}{\_\_} \rangle$)

$\therefore L(E) = L(M)$. ∎

Thm: Let L_1 & L_2 be T-rec langs. Then $L_1 \cup L_2$ & $L_1 \cap L_2$ are T-rec.

Proof: By the prev theorem, there exists enumerators E_1 and E_2 such that $L_1 = L(E_1)$ & $L_2 = L(E_2)$.

Define $E_\cup$ to enumerate $L_1 \cup L_2$ as follows:

$E_\cup$:= "On no input:
1. Simulate E_1 and E_2 simultaneously. ("time slicing")
2. [whenever] If E_1 prints a string x, then print x
3. [whenever] If E_2 prints a string y, then print y."

Clearly, $L(E_\cup) = L_1 \cup L_2$

$\therefore L_1 \cup L_2$ is T-rec (by the prev. thm).

Define a recognizer

$M_\cap$:= "On input w:
1. Run E_1.
2. If E_1 ever prints w, then
   a) Run E_2
   b) If E_2 ever prints w, then accept
   [else loop]
[3. else loop]"

Evidently, $L(M_\cap) = L_1 \cap L_2$ ▨

Thm: Let L be any language. If L & $\overline{L}$ are both T-rec, then L is decidable.

Proof: Let E & $\overline{E}$ be enumerators for L & $\overline{L}$ respectively. Let

D := "On input w:
1. Run E and $\overline{E}$ simultaneously (time-slicing)
2. If E ever prints w, then accept
3. If $\overline{E}$ ever prints w, then reject."

D is a decider, because one of E or $\overline{E}$ will print w eventually.

$\forall w$, $w \in L \Rightarrow E$ prints w and $\overline{E}$ does not print w
$\Rightarrow D$ accepts w

$w \notin L \Rightarrow w \in \overline{L}$
$\Rightarrow \overline{E}$ print w and E does not print w
$\Rightarrow D$ rejects w.

$\therefore D$ decides L. ▨

Cor: Let L be any T-rec, undecidable language (e.g. A_{TM}). Then $\overline{L}$ is not T-rec.

Thus $\overline{A_{TM}}$ is not T-rec.