<u>Theorem</u>: A_{TM} is undecidable.

Recall:

$A_{TM} := \{\langle M, w\rangle : M$ is a TM and M accepts input $w\}$.

<u>Proof</u>: Suppose A_{TM} is decidable. Let TM D be a decider for A_{TM}.

Consider the following TM

$B :=$ "On input $\langle M\rangle$ $\langle B\rangle$ where M is a TM:

1. Form the string $\langle M, \underbrace{\langle M\rangle}_{\text{string over } M\text{'s input alphabet}}\rangle$ $\langle B, \langle B\rangle\rangle$

2. Run D on input $\langle M, \langle M\rangle\rangle$ $\langle B, \langle B\rangle\rangle$

3. If D accepts $\langle M, \langle M\rangle\rangle$ $\langle B, \langle B\rangle\rangle$ then reject; else accept."
B rejects $\langle B\rangle$ B accepts $\langle B\rangle$

Consider what B does on input $\langle B\rangle$.

D accepts $\langle B, \langle B\rangle\rangle$
$\iff B$ accepts $\langle B\rangle$ (by assumption)
but
$\implies B$ rejects $\langle B\rangle$ (by B's design)

D rejects $\langle B\rangle$
$\implies B$ rejects $\langle B\rangle$ by assumption
but
$\implies B$ accepts $\langle B\rangle$ (by B's design)

$\therefore$ Our assumption that D exists leads to a contraction, so no such D can exist. //

<u>Use</u>: To show a lang. L undecidable, we "reduce" A_{TM} to L:

Any decider for L can be used to decide A_{TM} ↯

<u>Ex</u>:

$E_{TM} := \{\langle M\rangle : M$ is a TM & $L(M) = \emptyset\}$

<u>Prop</u>: E_{TM} is undecidable.

<u>Proof</u>: Assume Let D be a decider for E_{TM}. Let

$B :=$ "On input $\langle M, w\rangle$ where M is a TM & w is a string over M's input alphabet:

1. Let $R :=$ 'On input x:
 a) Run M on input w [& do what M does]'

// B hard-codes M & w into
// R's description.

2. Run D on input $\langle R\rangle$ [& do what D does]"

Claim: B decides A_{TM}

<u>Proof</u>: Assume Let D be a decider for E_{TM}. Let

B := "On input $\langle M, w \rangle$ where M is a TM & w is a string over M's input alphabet:

1. Let R := 'On input x:
   a) Run M on input w [& do what M does]'

// B hard-codes M & w into
// R's description.

2. Run D on input $\langle R \rangle$
3. If D accepts $\langle R \rangle$ then reject else accept."

<u>Claim</u>: B decides A_{TM}:

Let M & w be any TM & string.

$\langle M, w \rangle \in A_{TM} \iff M$ accepts w

$\Rightarrow R$ accepts all strings x

$\Rightarrow L(R) = \Sigma^*$

[Σ is R's input alphabet]

$\Rightarrow \langle R \rangle \notin E_{TM}$

$\Rightarrow D$ rejects $\langle R \rangle$ (by assumption)

$\Rightarrow B$ accepts $\langle M, w \rangle$.

$\langle M, w \rangle \notin A_{TM} \iff$ M does not accept w

$\Rightarrow R$ does not accept x

$\Rightarrow L(R) = \emptyset$

$\Rightarrow \langle R \rangle \in E_{TM}$

$\Rightarrow D$ accepts $\langle R \rangle$

$\Rightarrow B$ rejects $\langle M, w \rangle$.

<u>Shown</u>:
B is a decider & B accepts $\langle M, w \rangle$ iff $\langle M, w \rangle \in A_{TM}$.

$\therefore B$ decides A_{TM} ↯

$\therefore E_{TM}$ is undecidable.

<u>Enumerators</u>:

An enumerator E is a modified TM that behaves thus:

- E takes no input (blank tape initially)
- E runs forever (never halts)
- At any time E may <u>print</u> a string $w \in \Sigma^*$.

<u>Def</u>: Let E be enumerator. The language enumerated by E (written $L(E)$) is

$$L(E) := \{ w \in \Sigma^* : E \text{ prints } w \}$$

(E enumerates $L(E)$).

A language L is <u>enumerable</u> (aka <u>computably enumerable</u> (c.e.), or <u>recursively enumerable</u> (r.e.)) if $L = L(E)$ for some enumerator E.

<u>Thm</u>: A language is enumerable iff it is T-recognizable.

<u>Proof</u>: Fix a language $L \subseteq \Sigma^*$.

($\Longrightarrow$): Assume L is enumerated by some enumerator E $(L = L(E))$.
Let M be the following TM:

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

Claim that M recognizes L:
w arbitrary string.

$w \in L \iff E$ prints w (at some point)
$\iff M$ accepts w.

$\therefore M$ recognizes L.
[M either accepts or loops, so not a decider].
$\therefore L$ is T-recognizable (by M).

($\Longleftarrow$) Assume L is T-rec.
Let N be a TM recognizing L. Let E be the following enumerator:

$E :=$ "On no input:
1. Cycle through all strings of the form $\langle w, t \rangle$ where w is a string (over M's input alphabet) & t is a natural number:
2. For each such string $\langle w, t \rangle$,
   a) Run M on input w for t steps.
   b) If M accepts w within t steps, then <u>print</u> w "

finite time

<u>Claim</u>: $\forall w$, $w \in L \iff E$ prints w.

($\Longleftarrow$) obvious. E never prints a string w unless it sees that M accepts w, i.e., if $w \in L$

($\Longrightarrow$) Assume $w \in L$. Then M accepts w in, say, t_0 steps, for some t_0.
Then when E gets to the pair $\langle w, t_0 \rangle$, it notices that M accepts w, so E prints w.

$\therefore w \in L \iff E$ prints w
$\therefore L = L(E)$. ▨

<u>Thm</u>: Let L be any language. If L & $\overline{L}$ are both T-rec., the L is decidable.

<u>Proof</u>. Let E_1, E_2 be enumerators such that

$$L = L(E_1)$$
$$\overline{L} = L(E_2)$$

Decider for L:
$D :=$ "On input w:
1. Run E_1 & E_2 in tandem.
2. If E_1 prints w, then accept
3. If E_2 prints w, then reject"