Note on E_{TM}

The purpose of this note is to look at all the different tools we have for figuring out decidability, recognizability, and co-recognizability. We'll specifically use all these tools for E_{TM}.

1 E_{TM} is undecidable

How shall we prove this? Let me count the ways...

1.1 The fast way

Skip ahead to section 2.2 or 2.3 and show that E_{TM} is not recognizable. Then by the theorem that "a language is decidable iff it is both recognizable and corecognizable", it must be that E_{TM} is not decidable. □

1.2 By contradiction

S'pose that E_{TM} were decidable by some Turing machine R. We will build a decider S for $\overline{HALT_{\text{TM}}}$ using R as a subroutine, as follows:

$S =$ "On input $\langle M, w\rangle$:

(0) If $\langle M, w\rangle$ is not a properly formatted pair (Turing machine, string) then accept.

(1) Build a new machine $N = \begin{cases} \text{"On input } x\text{:} \\ \text{(a) Run } M \text{ on input } w. \\ \text{(b) Accept."} \end{cases}$

(2) Run R on input $\langle N\rangle$ and do the same."

Now we need to verify that:

- S is a decider. Line 0 will stop in finite time. Line (1) is just building a Turing machine and will also finish in finite time. Line (2) is just running a decider, so it finishes in finite time.
- If $\langle M, w\rangle \in \overline{HALT_{\text{TM}}}$ then S accepts it. There are two ways that $\langle M, w\rangle$ can be in $\overline{HALT_{\text{TM}}}$: it can be badly formatted or it can be a properly (Turing machine, string) pair where the Turing machine does not halt on that input string. If it's badly formatted, then S will halt on line (0) and accept. If it's $\langle M, w\rangle$ where M is a Turing machine which does not halt on w, then N will be a Turing machine which always loops, no matter what input x it gets, so N will never accept any string x. This means that $L(N) = \emptyset$, so the decider R for E_{TM} will accept the input $\langle N\rangle$, so S will accept on line (2).
- If $\langle M, w\rangle \notin \overline{HALT_{\text{TM}}}$, then S rejects it. If $\langle M, w\rangle \notin \overline{HALT_{\text{TM}}}$ then it must be that $\langle M, w\rangle$ encodes a Turing machine M which halts on input w. In that case, the constructed machine N will always halt and accept on every input x, so $L(N) = \Sigma^*$, so decider R for E_{TM} will reject $\langle N\rangle$. Thus S will reject on line (2).

Ok, so we've built a decider for $\overline{HALT_{\text{TM}}}$. But $\overline{HALT_{\text{TM}}}$ is undecidable! $\Rightarrow\Leftarrow$ □

1.3 By mapping reduction

We need to reduce from a language which is not decidable to E_{TM}. The guts of this reduction will be very similar to the proof by contradiction method.
Let's choose $\overline{HALT_{\text{TM}}}$ as our undecidable language; we want to show that $\overline{HALT_{\text{TM}}} \leq_m E_{\text{TM}}$, so we will build a computable function f where

$$w \in \overline{HALT_{\text{TM}}} \Leftrightarrow f(w) \in E_{\text{TM}}$$

Define function f as follows:

$$N_{\text{rej}} = \text{“On input } x\text{: reject.”}$$

$$N = \text{“On input } x\text{, run } M \text{ on } w \text{ and then accept.”}$$

$$f(w) = \begin{cases} \langle N_{\text{rej}} \rangle, & \text{if } w \text{ is not an encoded pair } \langle M, w \rangle \\ \langle N \rangle, & \text{if } w = \langle M, w \rangle \end{cases}$$

We need to check:

- This function f is computable by the following Turing machine:

  $S =$ “On input $\langle M, w \rangle$:
  (0) If $\langle M, w \rangle$ is not an encoded pair (Turing machine, string) then erase the tape, write $\langle N_{\text{rej}} \rangle$, and accept.
  (1) Build a new machine $N = \begin{cases} \text{“On input } x\text{:} \\ \text{(a) Run } M \text{ on input } w. \\ \text{(b) Accept.”} \end{cases}$
  (2) Erase the tape, write $\langle N \rangle$, accept.”

  Every step of S will terminate, so S will halt on all inputs; we can observe that S computes f in both the cases exactly as f is defined.

- If $w \in \overline{HALT_{\text{TM}}}$ then $f(w) \in E_{\text{TM}}$. We can argue this like in the “by contradiction” proof above, but let's for fun argue by contrapositive instead: if $f(w) \notin E_{\text{TM}}$ then $w \notin \overline{HALT_{\text{TM}}}$, that is, $w \in HALT_{\text{TM}}$.

  If $f(w) \notin E_{\text{TM}}$ then $f(w)$ must encode a Turing machine T which recognizes a language that is *not* empty, that is, $f(w) = \langle T \rangle$ where T is a Turing machine that accepts some string. Thus T is obviously not N_{rej}. So it must be that T is N from line (1), and if T accepts some string, then we can see that it must be that “run M on w” halted. So M halts on w, so $\langle M, w \rangle \in HALT_{\text{TM}}$.

- If $w \notin \overline{HALT_{\text{TM}}}$ then $f(w) \notin E_{\text{TM}}$. If $w \notin \overline{HALT_{\text{TM}}}$ then $w \in HALT_{\text{TM}}$ so $w = \langle M, y \rangle$ where M is a Turing machine that halts on input y. Thus by construction $f(w)$ will be $\langle N \rangle$ where machine N runs M on y and then accepts, no matter what input N got. So $L(N) \neq \emptyset$ so $\langle N \rangle = f(w) \notin E_{\text{TM}}$.

□

1.4 Using Rice's Theorem

$E_{\text{TM}} \subset L_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a Turing machine}\}$ so E_{TM} is a candidate for using Rice's theorem. We need to check that:

- E_{TM} is not empty. It definitely contains $\langle N_{\text{rej}} \rangle$ where N_{rej} = "On input x: reject." So $E_{\text{TM}} \neq \emptyset$.
- $E_{\text{TM}} \neq L_{\text{TM}}$. It definitely does not contain $\langle N_{\text{acc}} \rangle$ where N_{acc} = "On input x: accept." So $E_{\text{TM}} \neq L_{\text{TM}}$.
- E_{TM} is a property. If we have two Turing machines M_1 and M_2 where $L(M_1) = L(M_2)$ then either $L(M_1) = L(M_2) = \emptyset$ (and then $\langle M_1 \rangle$ and $\langle M_2 \rangle \in E_{\text{TM}}$) or $L(M_1) = L(M_2) \neq \emptyset$ (and then $\langle M_1 \rangle$ and $\langle M_2 \rangle \notin E_{\text{TM}}$). So E_{TM} is a property.

Thus Rice's theorem applies, so E_{TM} is not decidable. □

2 E_{TM} is unrecognizable

2.1 The fast way

Skip back to section 1.2 or 1.3 and show that E_{TM} is not decidable. Then skip ahead to section 3 and show that E_{TM} is co-recognizable. Then by the theorem that "a language is decidable iff it is both recognizable and corecognizable", it must be that E_{TM} is not recognizable. □

2.2 By contradiction[1]

S'pose that E_{TM} were recognizable by some Turing machine R. We will build a recognizer S for $\overline{HALT_{\text{TM}}}$ using R as a subroutine, as follows:

$S =$ "On input $\langle M, w \rangle$:
(0) If $\langle M, w \rangle$ is not a properly formatted pair (Turing machine, string) then accept.
(1) Build a new machine $N = \begin{cases} \text{"On input } x\text{:} \\ \text{(a) Run } M \text{ on input } w. \\ \text{(b) Accept."} \end{cases}$
(2) Run R on input $\langle N \rangle$ and do the same."

Now we need to verify that:

- S is a recognizer. Every Turing machine is a recognizer. This doesn't need to be justified.
- If $\langle M, w \rangle \in \overline{HALT_{\text{TM}}}$ then S accepts it. There are two ways that $\langle M, w \rangle$ can be in $\overline{HALT_{\text{TM}}}$: it can be badly formatted or it can be a properly (Turing machine, string) pair where the Turing machine does not halt on that input string. If it's badly formatted, then S will halt on line (0) and accept. If it's $\langle M, w \rangle$ where M is a Turing machine which does not halt on w, then N will be a Turing machine which always loops, no matter what input x it gets, so N will never accept any string x. This means that $L(N) = \emptyset$, so the recognizer R for E_{TM} will eventually accept the input $\langle N \rangle$, so S will accept on line (2).

[1] Note that I literally copy-pasted section 1.2 to this section, then updated "decidable" → "recognizable" and "reject" → "reject or loop", then reread it to check it makes sense. It does. The point is that you only need to think through the guts of this proof once, the same guts might prove more than one thing we're interested in proving.

- If $\langle M, w\rangle \notin \overline{HALT_{\text{TM}}}$, then S does not accept it. If $\langle M, w\rangle \notin \overline{HALT_{\text{TM}}}$ then it must be that $\langle M, w\rangle$ encodes a Turing machine M which halts on input w. In that case, the constructed machine N will always halt and accept on every input x, so $L(N) = \Sigma^*$, so the recognizer R for E_{TM} will either reject or loop on $\langle N\rangle$. Thus S will reject or loop on line (2).

Ok, so we've built a recognizer for $\overline{HALT_{\text{TM}}}$. But $\overline{HALT_{\text{TM}}}$ is undecidable! $\Rightarrow\Leftarrow$ $\square$

2.3 By mapping reduction

See section 1.3 for a reduction $\overline{HALT_{\text{TM}}} \leq_m E_{\text{TM}}$. Since $\overline{HALT_{\text{TM}}}$ is not recognizable, then E_{TM} is not recognizable.

3 E_{TM} is co-recognizable

3.1 By construction

We will build a co-recognizer C for E_{TM}.

$C =$ " On input $\langle M\rangle$:
(0) If $\langle M\rangle$ is not an encoding of a Turing machine, then reject.
(1) For $i = 1, 2, 3, \ldots$:
(a) Run the enumerator for Σ^* until it has printed the first i strings s_1, s_2, s_3, ..., s_i.
(b) Run M on each of these strings for i steps.
(c) If M accepted any of them, reject."

We need to check that C is a co-recognizer for E_{TM}:

- If $\langle M\rangle \notin E_{\text{TM}}$ then C rejects $\langle M\rangle$. If $\langle M\rangle \notin E_{\text{TM}}$ then either it is not an encoding of a Turing machine (so C rejects on line (0)) or it is an encoding of a Turing machine M where $L(M) \neq \emptyset$, so M accepts some string x. Since $x \in \Sigma^*$ the loop on line (1) will eventually get to a value of i large enough that $s_i = x$. Since M accepts string x it will accept after some number of steps k. On loop iteration number $\max(i, k)$ the machine C will run M on x for enough steps that M accepts, so C will reject.

- If $\langle M\rangle \in E_{\text{TM}}$ then C either accepts or loops on $\langle M\rangle$. First off, we wrote C and it *never* accepts — just look at it above! So we just need to argue that it loops in this case. If $\langle M\rangle \in E_{\text{TM}}$ then $L(M) = \emptyset$ so there are no strings which M accepts. (M is a machine that either loops or rejects on every string.) Thus C will never trigger line (1c) since M never accepts. Similarly, C will not trigger line (0) since $\langle M\rangle$ is an encoding of a Turing machine. So it must be that C just continues to run the loop on line (1) forever. Hooray!

Thus we've built a co-recognizer for E_{TM}. $\square$

3.2 Any other way

Don't. Just use construction. Skip back to section 3.1.