Enumerators

Note: "Shortlex" (or just "standard") order for strings is shorter strings before longer ones, and after that by 'dictionary' order, e.g. for $\{0,1\}^*$ it's $[\varepsilon, 0, 1, 00, 01, 10, 11, 000, \cdots]$

a. Let Turing Machine M recognize language L.

   i (Sipser 3.6) Why might the following enumerator not enumerate L?

   E = "Ignore the input.

   - For each string s of Σ^* (in standard order):
     - Run M on s. If it accepts, *print s*."

   Solution:

   Because if there is some string s for which M does not halt, then all strings later than s (in standard order) will never be tested or printed regardless of whether they are in L or not.

   ii Design an enumerator that enumerates L.

   Solution:

   E = "Ignore the input.

   - For each i in 1,2,3,...:
     - Run M on the first i strings (in standard order) for i steps each. Print each string it accepts."

   Notice that every string in L will get printed eventually: if $w \in L$ is the kth string of Σ^* in standard order and M accepts w in t steps, then w will get printed once i reaches $max(k, t)$. *(And printed again in each iteration after that, but that's allowed.)*

b. Prove that a language is (Turing-)recognizable iff some enumerator enumerates it. (And as a result, we will sometimes use the name "RE", i.e. "recursively *enumerable*", to refer to the class of recognizable languages.)

   Solution:

   One direction is already done in the previous question. For the other direction, assume some language L is enumerated by some enumerator E. Then the following machine recognizes L:

   M = "On input w: Run E until it prints w, then *accept*."

   (so strings in L will be accepted eventually because they are eventually printed, and strings not in L will never be accepted because "until it prints w" will just run forever.)

c. (Sipser 3.18) Show that a language is decidable iff some enumerator enumerates the language in the standard string order.

   Solution:

   Forward direction: Assume that language L is decidable. Then let M be a decider for L, and note that the enumerator from a.i enumerates L in standard order (when we originally presented it in a.i we did not know M halts, but now we do since by assumption it is a decider).

Reverse direction: Assume that E enumerates a language L in standard order. Then there are two cases:

Case I: L is finite. In this case, we have shown elsewhere that L is decidable (in fact, it's *regular*).

Case II: L is infinite. Then consider the machine $M =$ "On input w: Run E until one of the following occurs:

- E prints w. In this case, *accept*.
- E prints a string later than w in standard order. In this case, *reject*.

Since the language is infinite and there are only finitely many strings before w in standard order, one of the two cases will always happen, and thus M decides L. (Note that this strategy does *not* work for finite languages, because if w comes later in order than the final string of the language, then the enumerator may run forever after printing the entire language without ever reaching either end case.)

d. (Sipser 3.19) Show that every infinite Turing-recognizable language has an infinite decidable subset. *(Hint: use the result from the previous problem.)*

Solution:

Let L be an infinite recognizable language. Then let E enumerate L. We construct a new enumerator:

$E' =$ "Ignore the input.

- Run E. For each string it prints, *print* that string if we have not yet printed any later string (in standard order)."

By design E' only prints strings in standard order. And since L is infinite, there can never be a "last" string printed by E': E keeps printing infinitely many new strings, and at any point there are only finitely many strings that come before whatever was printed most recently. So $L(E')$ is an infinite subset of $L(E)$, and since it is enumerated in standard order, the previous problem tells us it's decidable.

e. (Sipser 4.30) Let A be a Turing-recognizable language consisting of descriptions of Turing machines, $\{\langle M_1\rangle, \langle M_2\rangle, \cdots\}$, where every M_i is a decider. Prove that some decidable language D is not decided by any decider M_i whose description appears in A. (Hint: You may find it helpful to consider an enumerator for A.)

Solution:

Case I: A is finite. Then since there are infinitely many decidable languages (including e.g. $\emptyset$, $\{0\}$, $\{00\}$, $\{000\}$, $\cdots$), some of them must not be decided by any machine described in A.

Case II: A is infinite. Let E enumerate A. Then define TM M as follows:

$M =$ "On input $\langle n\rangle$, where n is a positive integer:

- Run E until it outputs n distinct machines $M_1 \cdots M_n$.
- Run M_n on $\langle n \rangle$. *Accept* if it rejects, and *reject* if it accepts.

M is a decider since E will definitely output n distinct machines in finite time, and M_n is a decider. However for any i, we see that $L(M) \neq L(M_i)$: by construction, $\langle i \rangle$ is in *exactly one of* $L(M)$ and $L(M_i)$. Thus $L(M)$ is decidable but not decided by any machine described in A.

f. (Sipser 4.20) Let A and B be two disjoint languages. Say that language C separates A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-Turing-recognizable languages are separable by some decidable language. *(A language is co-RE if its complement is RE. Hint: First show that* every *string is in at least one of $\overline{A}$ and $\overline{B}$.)*

Solution:

Since A and B are co-RE, $\overline{A}$ and $\overline{B}$ are RE. So let $E_{\overline{A}}$ and $E_{\overline{B}}$ enumerate $\overline{A}$ and $\overline{B}$ respectively. Then we construct a TM M as follows:

$M =$ "On input w:

- Run $E_{\overline{A}}$ and $E_{\overline{B}}$ in parallel. If $E_{\overline{A}}$ prints w, *reject*; if $E_{\overline{B}}$ prints w, *accept.*

Let $L(M) = C$. I claim that C separates A and B, and that M is a decider for C. There are three cases for a string w:

Case I: $w \in A$. Then since A and B are disjoint, $w \notin B$, so $w \in \overline{B}$ and $w \notin \overline{A}$. Thus w will eventually get printed by only $E_{\overline{B}}$, so the machine will accept, and thus $w \in C$.

Case II: $w \in B$. Then since A and B are disjoint, $w \notin A$, so $w \in \overline{A}$ and $w \notin \overline{B}$. Thus w will eventually get printed by only $E_{\overline{A}}$, so the machine will reject, and thus $w \notin C$.

Case III: w is in neither A nor B. In this case both enumerators will print it eventually, so the machine will halt (and will either accept or reject based on whichever happened to print w first).

Thus in all cases the machine halts, and from case I and II we see that C separates A and B.