CS46 practice problems 10

These practice problems are an opportunity for discussion and trying many different solutions. It is **not counted towards your grade**, and **you do not have to submit your solutions.** The purpose of these problems is to get more comfortable with reasoning and writing proofs about decidability, recognizability, and co-recognizability.

If you are stumped or looking for guidance, **ask**.

1. Consider the language $L = \{\langle M, w\rangle \mid M$ is a single-tape TM that never modifies the portion of the tape that contains the original input $w\}$.
   (a) Show that L is co-Turing-recognizable, by briefly describing the elements of $\overline{L}$ and then describing a recognizer for $\overline{L}$.
   (b) Is L decidable? Prove your answer.
   Note that if you can show that L is Turing-recognizable, then you can apply Theorem 4.22 and part (a) to show L is decidable.
2. For each of the following languages, review if the language is decidable, Turing-recognizable, co-Turing-recognizable, or none of these. A_{DFA}, A_{CFG}, A_{TM}, E_{DFA}, E_{CFG}, E_{TM}, ALL_{DFA}, ALL_{CFG}, ALL_{TM}, EQ_{DFA}, EQ_{CFG}, EQ_{TM}.
3. Consider the language of Turing machines which only accept strings consisting of as and bs:

$$\text{TargetGPA}_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a Turing machine and } L(M) \subseteq L((a \cup b)^*)\}$$

   Is this language regular? context-free? decidable? recognizable? co-recognizable?
   You may consider these parts in any order. (Some orders will be more helpful than others.) Support your answer for each part with a proof!
4. Consider the language of deciders:

$$\textsc{Decider}_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a decider}\}$$

   (a) Recall that $L_{\text{TM}} = \{\langle M\rangle \mid M$ is a Turing machine$\}$, so $\textsc{Decider}_{\text{TM}} \subseteq L_{\text{TM}}$.
   Does Rice's Theorem apply to $\textsc{Decider}_{\text{TM}}$?
   (b) Show that $\textsc{Decider}_{\text{TM}}$ is undecidable.
   (c) Show that $\textsc{Decider}_{\text{TM}}$ is both not recognizable and not co-recognizable.

ONLY IF you finish problems 1-4, look at problem 5. The busy beaver problem is interesting, and famous, and challenging. It is also *much, much* more complicated than any problem I would *ever* ask on a homework or exam. It's here for your intellectual enjoyment!

5. **Busy beaver.** (Sipser 5.16) This problem is a Theory of Computation classic. It explores the idea of "computable function" and asks: are there some *numerical* functions which are not computable?

   Let $\Gamma = \{a, b, \sqcup\}$ be the tape alphabet for all Turing machines in this problem.

   Define the set:

$$HALT_{\text{TM}}^{k,\varepsilon} = \{\langle M \rangle \mid \text{ deterministic Turing machine } M \text{ has } k \text{ states and halts on input } \varepsilon\}$$

   Let $BB : \mathbb{N} \to \mathbb{N}$ be a function defined as follows: $BB(k)$ is the maximum number of 'a's on the tape of any machine in $HALT_{\text{TM}}^{k,\varepsilon}$ after it halts on input ε. This is called the "busy beaver" function.

   Show that the function BB is not computable. This will be a proof by contradiction, in several steps:

   (a) Show that BB is a strictly increasing function: $BB(n) < BB(n+1)$.

   (b) Show that if $f : \mathbb{N} \to \mathbb{N}$ is a computable function, then there is some integer q such that $f(n) \leq BB(n+q)$.
   (Hint: design a machine with approximately q states that when started with input $w = a^n$ halts with output $a^{f(n)}$ on its tape.)

   (c) Use parts (a) and (b) to obtain a contradiction.
   (Hint: Assume for contradiction that $BB(n)$ is computable. Show that this implies that $g(n) = BB(2n)$ is computable. Use this to get a contradiction.)

Answers on next page, for when you feel ready. If you've gotten this far, give yourself and your discussion partners some time to try to figure out the parts of this problem before reading the solution.

Busy beaver. (Sipser 5.16) Let $\Gamma = \{a, b, \sqcup\}$ be the tape alphabet for all Turing machines in this problem.

Define the set:

$$HALT_{\mathrm{TM}}^{k,\varepsilon} = \{\langle M\rangle \mid \text{ Turing machine } M \text{ has } k \text{ states and halts on input } \varepsilon\}$$

Let $BB : \mathbb{N} \to \mathbb{N}$ be a function defined as follows: $BB(k)$ is the maximum number of 'a's on the tape of any machine in $HALT_{\mathrm{TM}}^{k,\varepsilon}$ after it halts on input ε. This is called the "busy beaver" function.

Show that the function BB is not computable. Proof by contradiction, in several steps:

(a) Show that BB is a strictly increasing function: $BB(n) < BB(n + 1)$.

Let M be a Turing machine with k states, in $HALT_{\mathrm{TM}}^{k,\varepsilon}$, which halts with $BB(k)$ 'a's on its tape. (We know there is at least one Turing machine that hits this maximum. There might be more, but we just need M to be one of them.) We modify it by adding one new state, q_{new}. Anytime M had a transition to its accept (or reject) state like:

$$\delta(q, \sigma) = (q_{\text{accept}}, a, D)$$

where $D \in \{L, R\}$ and $\sigma \in \{a, b, \sqcup\}$, we modify it to be this instead:

$$\delta(q, \sigma) = (q_{\text{new}}, a, R)$$

We also add the transitions

$$\delta(q_{\text{new}}, a) = (q_{\text{new}}, a, R)$$
$$\delta(q_{\text{new}}, \sqcup) = (q_{\text{accept}}, a, R)$$

Thus, whenever M was about to halt, our new machine will instead go into state q_{new}, go to the rightmost end of the 'a's on the tape, and add one more 'a', then halt. Thus the new machine has $k + 1$ states, and when started on ε, halts with $BB(k) + 1$ consecutive 'a's on the tape.

Since $BB(k+1)$ is the maximum over all such $k+1$-state machines, it must be that $BB(k+1) \geq BB(k) + 1$.

(b) Show that if $f : \mathbb{N} \to \mathbb{N}$ is a computable function, then there is some integer q such that $f(n) \leq BB(n + q)$.

(Hint: design a machine with approximately q states that when started with input $w = a^n$ halts with output $a^{f(n)}$ on its tape.)

We'll assume that we are using unary notation. By definition, if f is computable, then there is a Turing machine M_f which, when started with a^n on the tape, finishes with only $a^{f(n)}$ on the tape. This machine M has q states.

We define a new machine M'_f which, when started with ε on the tape, uses some new, additional states q_0, q_1, $\ldots$, q_{n-1}, to first write a^n on the tape, then go back to the leftmost end of the tape, then run M_f (so that it will halt with $a^{f(n)}$ on the tape). The new state q_0 is the start state of M_f'; assume that the start state of M_f is called q_{M_f}. The transitions for these new states are:

$$\begin{aligned}
\delta(q_0, \sqcup) &= (q_1, a, R) \\
\delta(q_1, \sqcup) &= (q_2, b, R) \\
\delta(q_2, \sqcup) &= (q_3, a, R) \\
&\vdots \\
\delta(q_{n-2}, \sqcup) &= (q_{n-1}, a, L) \\
\delta(q_{n-1}, a) &= (q_{n-1}, a, L) \\
\delta(q_{n-1}, b) &= (q_{M_f}, a, L)
\end{aligned}$$

The new machine M_f' is in $HALT_{\text{TM}}^{k,\varepsilon}$ for $k = n + q$ and halts with $a^{f(n)}$ on the tape, so because $BB(n+q)$ is the maximum over all such Turing machines, $BB(n+q) \geq f(n)$.

(c) Use parts (a) and (b) to obtain a contradiction.

(Hint: Assume for contradiction that $BB(n)$ is computable. Show that this implies that $g(n) = BB(2n)$ is computable. Use this to get a contradiction.)

Assume $BB(n)$ is computable, and let M_{BB} be the Turing machine that computes it. (So, started on an input tape of a^n, it halts with just $a^{BB(n)}$ on the tape.)

We define a new machine M' which, when started with an empty tape,

(a) first writes n 'a's on the tape,

(b) then doubles this to be $2n$ 'a's,

(c) then simulates M_{BB}.

The first step requires at most n states, and steps 2 and 3 require some constant c more states (some for the doubling, and then all the states of M_{BB}). So machine M' has some $n + c = \ell$ states, and when started on an empty string, halts with $BB(2n)$ 'a's on the tape. Thus $BB(n+c) \geq BB(2n)$.

But notice that c is some constant, while n is a variable. Consider setting $n = c + 1$; then we have shown that $BB(c+1+c) \geq BB(2c+2)$, which contradicts the fact (from part (a) that BB is a strictly increasing function, since it must be that $BB(c+1+c) < BB(2c+2)$.