MATH 4LT/6LT3 Assignment #4
Due: Monday, 10 November by 11:59pm.

1. Recall that the language

$$SAP = \{\ulcorner M \urcorner \mid M \text{ is a DTM that accepts } \ulcorner M \urcorner\},$$

is the Self Acceptance Problem. We saw that this language is CE but not coCE. Let $L = 0 \cdot SAP \cup 1 \cdot \overline{SAP}$. So a string from $\{0,1\}^*$ is in L if it is of the form 0σ, where $\sigma \in SAP$ or of the form 1σ where σ is in the complement of SAP. Prove that L is not CE and is also not coCE.

Solution: One way to show this is to many-one reduce languages that are not CE or coCE to L. This would imply that L is not CE and not coCE. We can use the languages SAP and $\overline{SAP}$ for this. Define ρ_0 and ρ_1 to be the functions that on a string x outputs the strings $0 \cdot x$ and $1 \cdot x$ respectively. Clearly both of these functions are computable.

We claim that ρ_0 is a many-one reduction from SAP to L. This is almost immediate, since if $x \in SAP$, then by definition $0 \cdot x$ is in L. Also, if $x \notin SAP$, then $0 \cdot x \notin L$, since the only strings in L whose first symbol is 0 are of the form $0 \cdot y$ for some string $y \in SAP$. Similarly, ρ_1 is a many-one reduction from $\overline{SAP}$ to L.

Since SAP is not coCE and $\overline{SAP}$ is not CE then by Theorem 2.8.7, L is neither CE nor coCE.

2. Exercise 2.10.15 from the textbook.

Solution: Suppose that B is a computable language, and let M_B be a halting DTM with $L(M_B) = B$. Let M'_B be the DTM that does the following on input $x \in \Sigma^*$:

- Run M_B on x,
- Once M_B has halted, if it accepts x, then erase the tape and write a 1 on the tape and halt, with the head over cell #1. If M_B rejects x, then erase the tape and write a 0 on the tape and halt with the head over cell #1.

We claim that M'_B computes the function χ_B, since by design, if $x \in B$, then on input x, M'_B will output 1 and if $x \notin B$, then M'_B will output 0. So χ_B is a computable function.

Conversely, suppose that χ_B is computable and that the DTM M computes it (so it is a halting DTM as well). Let M_B be the DTM that on input x does the following:

- Run M and wait for the computation to end.
- If the first cell of the tape contains a 1, M_B halts in its accept state. Otherwise, M_B halts in its reject state.

Then M_B is a halting DTM with $L(M_B) = \{x \in \Sigma^* \mid \chi_B(x) = 1\} = B$. So, B is computable.

3. Exercise 2.10.16 from the textbook.

Solution: Let B be a CE language and M_B a DTM with $L(M_B) = B$. Let M be the DTM that on input x does the following:

- Runs M_B on input x.
- If M_B halts in the reject state, then M will loop.
- If M_B halts in the accept state, then M accepts x.

The DTM M has the property that if $x \in B$ then M_B will halt in its accept state on input x and hence that M will do the same. So $x \in L(M)$. On the other hand, if $x \notin B$ then either M_B will loop on x, in which case M will as well, or M_B halts and enters its reject state. In this case, M loops on input x. So, if $x \notin B$, then M fails to halt on input x. Thus the set of input strings for which M halts is exactly B.

Conversely, if M has the stated property for B, let M_B be the DTM that on input a string x does the following:

- Run M on x.
- If M halts, in either the accept or reject state, then M_B accepts x.

It follows that $L(M_B)$ is equal to the set of strings on which M halts and so $L(M_B) = B$, showing that B is CE.

4. Exercise 2.10.18, part 1 from the textbook.

   Solution: Let B and C be computable languages with M_B and M_C halting DTMs with $L(M_B) = B$ and $L(M_C) = C$. Let M be the DTM that does the following on input $x = a_1a_2 \dots a_n \in \Sigma^*$:

   - Initialize $i = 0$.
   - While $i \leq n$,
     - Run M_B on input $a_1a_2 \dots a_i$ and run M_C on input $a_{i+1}a_{i+2} \dots a_n$. If both DTMs accept, M accepts x.
     - If not, set $i = i + 1$.
   - M rejects x.

   We claim that M is a halting DTM with $L(M) = B \cdot C$. It is halting, since, no matter the length of the input string $x = a_1a_2 \dots a_n$, M will halt after at most $|x|$ iterations of the while loop, and both M_B and M_C are halting DTMs. If M accepts a string x, then the only way for this to happen is that for some $i \leq n$, M_B accepts $a_1a_2 \dots a_i$ and M_C accepts $a_{i+1}a_{i+2} \dots a_n$. This implies that $a_1a_2 \dots a_i \in B$ and $a_{i+1}a_{i+2} \dots a_n \in C$, which means that $x \in B \cdot C$.

   Conversely, if $x = u \cdot v$ for some $u \in B$ and $v \in C$, then in the ith iteration of the while loop, with $i = |u|$, M_B will accept $a_1a_2 \dots a_i$ and M_C will accept $a_{i+1}a_{i+2} \dots a_n$ and so M will accept x. Thus $L(M) = B \cdot C$, establishing that this language is computable.

   Note that another approach, via non determinism can also be used to show that $B \cdot C$ is computable. Rather than searching through all possible ways to express an input string x as the concatenation of two other strings u and v and checking to see if these strings are in B and C respectively, a nondeterministic DTM would first "guess" the value of i and then check to see if $a_1a_2 \dots a_i \in B$ and $a_{i+1}a_{i+2} \dots a_n \in C$.

5. Exercise 2.10.24 from the textbook.

   Solution: To slightly simplify this presentation, assume that Σ contains distinct symbols 0 and 1 (and possibly others) so that we can make use of our standard encoding of ordered pairs of strings. Let B be a nonempty CE language and M_B a DTM with $L(M_B) = B$. Let $\sigma \in B$

be some fixed string that we will use to define a computable function $f : \Sigma^* \to \Sigma^*$ with $B = \{f(x) \mid x \in \Sigma^*\}$. The idea behind defining f is via a related function $g : \Sigma^* \times \mathbb{N} \to \Sigma^*$, where $g(x, n)$ is equal to x if on input x, the DTM M_B has halted within n steps and accepts x, and is equal to the fixed string σ otherwise.

It can be seen that the range of g is B, since if $x \in B$, M_B will accept x, and if the computation of M_B on input x has length n, then $g(x, n) = x$. On the other hand, the only strings that are in the range of g are, by design, strings that M_B accepts. The definition of g provides an algorithm for computing it: on input x and n, run M_B on input x for up to n steps and see what happens. If within this number of steps, M_B halts and accepts x, then output x. Otherwise, output σ.

Strictly speaking, we need to present a one-variable computable function on Σ^* whose range is B. We can use our encoding method for ordered pairs to do this. Let M be a DTM that does the following on input $z \in \Sigma^*$:

- If z is not of the form $\langle x, y\rangle$ for any strings x, $y \in \Sigma^*$, then M halts and outputs the string σ. So, in this case, M erases its tape, writes the string σ on it and halts in the accept state.
- Otherwise, $z = \langle x, y\rangle$ for some strings x, $y \in \Sigma^*$. In this case, run M_B on input x for up to $|y|$ steps. If M_B has halted in the accept state before $|y|$ steps, M halts in the accept state with the string x on its tape. It not, then M halts in the accept state with σ on its tape.

It can be seen that M is a halting DTM that computes a function (closely related to g) whose range is the set B.

Conversely, suppose that f is a computable function whose range is the language B. Let M_f be a DTM that computes f. We will rely on the fact that the elements of Σ^* can be effectively enumerated in the following sense. There is a listing of the members of Σ^*, $x_1, x_2, \ldots, x_n, \ldots$ for which there is an algorithm that one by one, outputs the strings in this sequence. Such a listing can be produced by first listing all strings of legth 0, then all strings of length 1, then all strings of length 2, and so on. For a given length n, the strings of length n can be listed in

some definite order, such as lexicographically. Let M be a DTM that outputs this list.

Let M_B be the DTM that on input a string x does the following:

- Initialize $i = 0$.
- Iterate:
  - Run M until it has produced the string x_i
  - Run the DTM M_f on input x_i.
  - If M_f halts with x on its tape (so if $f(x_i) = x$), then M halts in its accept state.
  - If not (so $f(x_i) \neq x$), set $i = i + 1$.

It can be seen that on input x the DTM M will halt (in its accept state) if and only if x is in the range of f. So $L(M)$ is equal to the range of the function f, showing that B is a CE language.

The following question is for students enrolled in MATH 6LT3. Students in MATH 4LT3 can treat it as a bonus question.

B1 Exercise 2.10.39.

Solution: We make use of the fact that the following language is not computable:

$$HALT_\epsilon = \{\ulcorner M \urcorner \mid M \text{ is a DTM with } \epsilon \in L(M)\}.$$

This can be (easily) shown using Rice's Theorem, or can be shown by many-one reducing some known noncomputable language, such as SAP, to it.

Now, suppose that there is a computable function $g : \Sigma^* \to \Sigma^*$ with $|b(1^n)| \leq |g(1^n)|$ for each $n \geq 0$, and suppose M_g is a DTM that computes g. Let M' be the DTM that does the following on input x:

- If x is not of the form $\ulcorner M \urcorner$ for any DTM M, then M' halts in the reject state.
- If $x = \ulcorner M \urcorner$ for some DTM M, set i to be the number of states of M.

- Run M_g on input 1^n and let 1^m be its output (so $g(1^n) = 1^m$).
- Run M on input ϵ for up to $m + 1$ steps.
- If before m steps, M has halted, then M' accepts x if M has halted in the accept state, and rejects x if M has halted in the reject state.
- If M hasn't halted by $m + 1$ steps, then M' rejects x.

By design, M' is a halting DTM, since for any DTM M, M' will halt on input $\ulcorner M \urcorner$ after at most $m + 1$ steps, where M has n states and $g(1^n) = 1^m$. Furthermore, M' will only accept strings of the form $\ulcorner M \urcorner$ where M is a DTM that accepts the empty string. To see this, suppose that M accepts the empty string. Then by definition, it will halt on input ϵ after at most $b(1^n)$ steps. So after at most $g(1^n)$ steps, M will have halted. If it has halted in the accept state, then M' will accept the string, and it not it will reject. If by $g(1^n)$ steps M has not halted, then by the definition of b, we know that M will never halt on input ϵ and so $\ulcorner M \urcorner \notin HALT_\epsilon$. Thus M is a halting DTM with $L(M) = HALT_\epsilon$, contradicting that this language is not computable.