PART A (Graded by Francisco and Anupa)

PROBLEM 1 (4+8+8 points)

You may find it useful to review Rice’s theorem and its proof before doing this problem (Lecture 17 Slides 10-11). Let \( L_0 \) be any arbitrary but fixed Turing-recognizable language other than \( \emptyset \) or \( \Sigma^* \). We will prove that the language \( X = \{ \langle M \rangle : L(M) = L_0 \} \) is neither Turing-recognizable nor co-recognizable.

(A) Prove that \( A_{\epsilon} = \{ \langle M \rangle : \epsilon \not\in L(M) \} \) is not Turing-recognizable. \((Hint: \ this \ can \ be \ done \ without \ a \ reduction.)\)

(B) Prove that \( X \) is not Turing-recognizable by giving a mapping reduction from \( A_{\epsilon} \) to \( X \).

(C) Prove that \( \overline{X} \) is not Turing-recognizable, again by reduction from \( A_{\epsilon} \).

(A) Note that \( \overline{A_{\epsilon}} = \{ \langle M \rangle : \epsilon \in L(M) \} \). \( \overline{A_{\epsilon}} \) is recognizable since we can construct a recognizer which on input \( \langle M \rangle \) simulates \( M \) on \( \epsilon \). However, \( \overline{A_{\epsilon}} \) is not decidable by Rice’s theorem, since if we let \( P \) be the subset of r.e. languages \( L \) where \( P = \{ L : \epsilon \in L \} \), then \( \overline{A_{\epsilon}} = \{ \langle M \rangle : L(M) \in P \} \) \((P \) is clearly defined to be a property of the language rather than the machine).

If \( A_{\epsilon} \) were recognizable, then \( \overline{A_{\epsilon}} \) would be both recognizable and co-recognizable, so \( \overline{A_{\epsilon}} \) would be decidable. This a contradiction, so \( A_{\epsilon} \) is not recognizable.

(B) Suppose we have a mapping reduction from \( L_1 \) to \( L_2 \), \( L_1 \leq_m L_2 \) and \( L_2 \) is recognizable. Then we can apply the map \( f \) to input \( w \) and run \( f(w) \) on \( L_2 \) to recognize if \( w \in L_1 \). Thus if \( L_2 \) is recognizable then \( L_1 \) is recognizable, and if \( L_1 \) is unrecognizable then \( L_2 \) must be unrecognizable.
Let $M_0$ be a TM which recognizes $L_0$. Let $f$ be computed as follows: On input $(M)$, construct a TM $M_f$ which, on input $w$, simulates $M$ on $\varepsilon$ and $M_0$ on $w$ simultaneously by interleaving their steps. $M_f$ accepts when either machine accepts, and rejects when both machines reject, and otherwise continues simulating. $(M_f)$ is the output, so $f((M)) = (M_f)$.

- If $(M) \in A_\Sigma$ then $\varepsilon \notin L(M)$, so $M_f$ will accept iff $M_0$ accepts $w$, so $L(M_f) = L(M_0) = L_0$. Thus $(M_f) \in X$.
- If $(M) \notin A_\Sigma$, then $\varepsilon \in L(M)$, so $M_f$ will accept any $w \in \Sigma^*$, so $L(M_f) = \Sigma^*$. Since we assumed that $L_0 \neq \Sigma^*$, this means that $L(M_f) \neq L_0$, so $(M_f) \notin X$.

Thus $f$ is a computable function such that $\forall w \in \Sigma^*, w \in A_\Sigma$ iff $f(w) \in X$ and we are done.

(C) As in the previous problem, to show that $\overline{X}$ is unrecognizable, we can give a mapping reduction by providing a computable function $f$ such that $\forall w \in \Sigma^*, w \in A_\Sigma$ iff $f(w) \in \overline{X}$. This $f$ is very similar to the one in the previous problem, except instead of producing a machine accepting when either submachine accepts, it produces one which accepts when both submachines accept.

Let $M_0$ be a TM recognizing $L_0$. Let $f$ be computed as follows: On input $(M)$, construct a TM $M_f$ which, on input $w$, simulates $M$ on $\varepsilon$ and $M_0$ on $w$ simultaneously by interleaving their steps. $M_f$ accepts when BOTH accept, and rejects when either machine rejects, and otherwise continues simulating. $(M_f)$ is the output, so $f((M)) = (M_f)$.

- If $(M) \in A_\Sigma$ then $\varepsilon \notin L(M)$, so $M_f$ won’t accept any string and $L(M_f) = \emptyset$. Since we assumed that $L_0 \neq \emptyset$, this means that $L(M_f) \neq L_0$, so $(M_f) \in \overline{X}$.
- If $(M) \notin A_\Sigma$, then $\varepsilon \in L(M)$, so $M_f$ will accept a string $w$ iff $M_0$ accepts it. Thus $L(M_f) = L(M_0) = L_0$ and $(M_f) \notin \overline{X}$

Thus $f$ is a computable function such that $\forall w \in \Sigma^*, w \in A_\Sigma$ iff $f(w) \in \overline{X}$ and we are done.

PROBLEM 2 (3+3+3 points)

Let $f(n)$ and $g(n)$ be functions over: positive natural numbers $\rightarrow$ positive real numbers. Prove or give a counterexample for the following conjectures (and explain briefly why any given counterexamples are valid).

(A) $f(n) = \Theta(g(n))$ implies $g(n) = \Theta(f(n))$.
(B) $f(n) = O((f(n))^2)$.
(C) $f(n) = \Theta(g(n))$ implies $2^{f(n)} = \Theta(2^{g(n)})$. 


(A) True. \(f(n) = \Theta(g(n))\) implies that \(f(n) = O(g(n))\) and \(g(n) = O(f(n))\), which means \(g(n) = \Theta(f(n))\).

(B) False. Let \(f(n) = \frac{1}{n}\), then \(1/n \neq O(1/n^2)\). To see that this is a counterexample, assume there exists \(C,N\), such that \(\forall n > N, \frac{1}{n} \leq C \cdot \frac{1}{n^2}\). Then, \(\forall n > N, n \leq C\), which is impossible.

(C) False. Let \(f(n) = \log_2(n)\) and \(g(n) = 2 \log_2(n)\). Clearly \(f(n) = \Theta(g(n))\) using \(N = 1\) and \(C = 1, 2\). Note however that \(2f(n) = n\) and \(2g(n) = n \log_2(n^2) = n^2\), but \(n \neq \Theta(n^2)\) since \(n^2 \neq O(n)\). To show that \(n^2 \neq O(n)\), assume that \(n^2 = O(n)\), so there exist \(C, N\) such that \(\forall n > N, n^2 \leq C \cdot n\). Then, \(\forall n > N, n \leq C\), which is impossible. Thus \(n^2 \neq O(n)\).

PROBLEM 3 (Challenge! 2 points)

Show that every infinite regular language has a subset that it recognizable but not decidable.

Solution:
Consider the set of all Turing Machines. We can order them in some way (let’s say lexicographically according to their string representations, but it doesn’t really matter). Consider the language \(L_\varepsilon = \{\langle M \rangle : M\) halts on \(\varepsilon\} \) (or any other known undecidable language with \(\langle M \rangle\) as input). Now let’s name our infinite regular language \(R\), and we’ll number its elements \(r_0, r_1, \ldots\) in some ordering, probably lexicographically. Then we can define \(R' \subseteq R = \{r_i \in R : \langle M_i \rangle \in L_\varepsilon\}\).

\(R'\) is Turing-recognizable because we can build a Turing Machine \(M_{R'}\) that recognizes it as follows:

- On input \(r_i\), simulate \(M_i\) on \(\varepsilon\).
- Halt if \(M_i\) halts.

\(M_{R'}\) will halt on precisely the strings that correspond to Turing Machines in \(L_\varepsilon\).

\(R'\) is not, however, Turing-decidable. If it were, and were decided by \(M_{R'}\), then we could build a Turing Machine \(M_{L_\varepsilon}\) to decide \(L_\varepsilon\) as follows:

- On input \(\langle M_i \rangle\), write \(r_i\).
- Feed \(r_i\) to \(M_{R'}\), returning \(M_{R'}\)’s answer.

\(M_{R'}\) accepts if and only if \(\langle M_i \rangle\) halts on \(\varepsilon\) and rejects if and only if \(\langle M_i \rangle\) does not halt on \(\varepsilon\), so this gives us a decision procedure for \(L_\varepsilon\), a known undecidable language. Thus \(M_{R'}\) must not exist, and \(R'\) must not be decidable. As you can see we only used that we can enumerate the elements of the regular language in some order, hence the same argument holds for the case when instead of regular we have recursive languages.