Homework 8 Solutions

1. Consider the problem of testing whether a DFA and a regular expression are equivalent. Express this problem as a language and show that it is decidable.

**Answer:** Define the language as

\[ C = \{ \langle M, R \rangle \mid M \text{ is a DFA and } R \text{ is a regular expression with } L(M) = L(R) \} \].

Recall that the proof of Theorem 4.5 defines a Turing machine \( F \) that decides the language \( EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \). Then the following Turing machine \( T \) decides \( C \):

\[ T = \text{“On input } \langle M, R \rangle, \text{ where } M \text{ is a DFA and } R \text{ is a regular expression:} \]

1. Convert \( R \) into a DFA \( D_R \) using the algorithm in the proof of Kleene’s Theorem.
2. Run TM \( F \) from Theorem 4.5 on input \( \langle M, D_R \rangle \).
3. If \( F \) accepts, accept. If \( F \) rejects, reject.”

2. Let \( A_{\varepsilon_{\text{CFG}}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates } \varepsilon \} \). Show that \( A_{\varepsilon_{\text{CFG}}} \) is decidable.

**Answer:** We need to ensure that we test all derivations, but we also need the derivations not to be infinite, or to loop forever. To do this, we first convert the CFG \( G \) into an equivalent CFG \( G' = (V, \Sigma, R, S) \) in Chomsky normal form. If \( S \to \varepsilon \) is a rule in \( G' \), where \( S \) is the start variable, then clearly \( G' \) generates \( \varepsilon \), so \( G \) also generates \( \varepsilon \) since \( L(G) = L(G') \). Since \( G' \) is in Chomsky normal form, the only possible \( \varepsilon \)-rule in \( G' \) is \( S \to \varepsilon \), so the only way we can have \( \varepsilon \in L(G') \) is if \( G' \) includes the rule \( S \to \varepsilon \) in \( R \). Hence, if \( G'' \) does not include the rule \( S \to \varepsilon \), then \( \varepsilon \notin L(G') \). Thus, a Turing machine that decides \( A_{\varepsilon_{\text{CFG}}} \) is as follows:

\[ M = \text{“On input } \langle G \rangle, \text{ where } G \text{ is a CFG:} \]

1. Convert \( G \) into an equivalent CFG \( G' = (V, \Sigma, R, S) \) in Chomsky normal form.
2. If \( G' \) includes the rule \( S \to \varepsilon \), accept. Otherwise, reject.”
3. Let $\Sigma = \{0, 1\}$, and define

$$A = \{ \langle R \rangle \mid R \text{ is a regular expression describing a language over } \Sigma \text{ containing at least one string } w \text{ that has } 111 \text{ as a substring (i.e., } w = x111y \text{ for some } x \text{ and } y) \}.$$  

Show that $A$ is decidable.

**Answer:** Define the language $C = \{ w \in \Sigma^* \mid w \text{ has } 111 \text{ as a substring} \}$. Note that $C$ is a regular language with regular expression $(0 \cup 1)^*111(0 \cup 1)^*$ and is recognized by the following DFA $D_C$:

![DFA Diagram]

Now consider any regular expression $R$ with alphabet $\Sigma$. If $L(R) \cap C \neq \emptyset$, then $R$ generates a string having $111$ as a substring, so $\langle R \rangle \in A$. Also, if $L(R) \cap C = \emptyset$, then $R$ does not generate any string having $111$ as a substring, so $\langle R \rangle \not\in A$. By Kleene’s Theorem, since $L(R)$ is described by regular expression $R$, $L(R)$ must be a regular language. Since $C$ and $L(R)$ are regular languages, $C \cap L(R)$ is regular since the class of regular languages is closed under intersection, as was shown in an earlier homework. Thus, $C \cap L(R)$ has some DFA $D_{C \cap L(R)}$. Theorem 4.4 shows that $E_{\text{DFA}} = \{ \langle B \rangle \mid B \text{ is a DFA with } L(B) = \emptyset \}$ is decidable, so there is a Turing machine $H$ that decides $E_{\text{DFA}}$. We apply TM $H$ to $\langle D_{C \cap L(R)} \rangle$ to determine if $C \cap L(R) = \emptyset$. Putting this all together gives us the following Turing machine $T$ to decide $A$:

$$T = \text{“On input } \langle R \rangle, \text{ where } R \text{ is a regular expression:}$$

1. Convert $R$ into a DFA $D_R$ using the algorithm in the proof of Kleene’s Theorem.
2. Construct a DFA $D_{C \cap L(R)}$ for language $C \cap L(R)$ from the DFAs $D_C$ and $D_R$.
3. Run TM $H$ that decides $E_{\text{DFA}}$ on input $\langle D_{C \cap L(R)} \rangle$.
4. If $H$ accepts, reject. If $H$ rejects, accept.”

4. Consider the emptiness problem for Turing machines:

$$E_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a Turing machine with } L(M) = \emptyset \}.$$  

Show that $E_{\text{TM}}$ is co-Turing-recognizable. (A language $L$ is co-Turing-recognizable if its complement $\overline{L}$ is Turing-recognizable.) Note that the complement of $E_{\text{TM}}$ is

$$\overline{E_{\text{TM}}} = \{ \langle M \rangle \mid M \text{ is a Turing machine with } L(M) \neq \emptyset \}.$$
(Actually, $\overline{E_{\text{TM}}}$ also contains all $\langle M \rangle$ such that $\langle M \rangle$ is not a valid Turing-machine encoding, but we will ignore this technicality.)

**Answer:** We need to show there is a Turing machine that recognizes $\overline{E_{\text{TM}}}$, the complement of $E_{\text{TM}}$. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in $\Sigma^*$. For a given Turing machine $M$, we want to determine if any of the strings $s_1, s_2, s_3, \ldots$ is accepted by $M$. If $M$ accepts at least one string $s_i$, then $L(M) \neq \emptyset$, so $\langle M \rangle \in \overline{E_{\text{TM}}}$. If $M$ accepts none of the strings, then $L(M) = \emptyset$, so $\langle M \rangle \not\in E_{\text{TM}}$. However, we cannot just run $M$ sequentially on the strings $s_1, s_2, s_3, \ldots$. For example, suppose $M$ accepts $s_2$ but loops on $s_1$. Since $M$ accepts $s_2$, we have that $\langle M \rangle \in \overline{E_{\text{TM}}}$. But if we run $M$ sequentially on $s_1, s_2, s_3, \ldots$, we never get past the first string. The following Turing machine avoids this problem and recognizes $\overline{E_{\text{TM}}}$:

$$R = \text{"On input $\langle M \rangle$, where $M$ is a Turing machine:}
\begin{enumerate}
\item \text{Repeat the following for } i = 1, 2, 3, \ldots.
\item \text{Run $M$ for } i \text{ steps on each input } s_1, s_2, \ldots, s_i.
\item \text{If any computation accepts, accept.}
\end{enumerate}\text{"}$$

5. Let $A$ and $B$ be two disjoint languages over a common alphabet $\Sigma$. Say that language $C$ *separates* $A$ and $B$ if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that if $A$ and $B$ are any two disjoint co-Turing-recognizable languages, then there exists a decidable language $C$ that separates $A$ and $B$. (A language $L$ is co-Turing-recognizable if its complement $\overline{L}$ is Turing-recognizable.)

**Answer:** Suppose that $A$ and $B$ are disjoint co-Turing-recognizable languages. We now prove that there exists a decidable language $C$ that separates $A$ and $B$. Since $A$ is co-Turing-recognizable, its complement $\overline{A}$ must have an enumerator $E_{\overline{A}}$. Similarly, the fact that $B$ is co-Turing-recognizable implies $\overline{B}$ has an enumerator $E_{\overline{B}}$. Since $A$ and $B$ are disjoint, i.e., $A \cap B = \emptyset$, we have that $\overline{A} \cup \overline{B} = \Sigma^*$ by DeMorgan’s law. Thus, every string in $\Sigma^*$ is in the union of $\overline{A}$ and $\overline{B}$. Furthermore, since $A$ and $B$ are disjoint, every string in $B$ is in $\overline{A}$, and every string in $A$ is in $\overline{B}$.

Using these facts, we construct a Turing machine $M$ as follows:

$$M = \text{"On input $w$, where } w \in \Sigma^*:}
\begin{enumerate}
\item \text{Run $E_{\overline{B}}$ and $E_{\overline{A}}$ in parallel.}
\item \text{Alternating between the enumerators, and starting with $E_{\overline{B}}$, compare the outputs of each of the enumerators, one string at a time, to the input $w$.}
\item \text{If some output of $E_{\overline{B}}$ matches } w, \text{ accept.}
\item \text{If some output of $E_{\overline{A}}$ matches } w, \text{ reject.}"
\end{enumerate}$$
Let $C$ be the language recognized by TM $M$. Since $\overline{A} \cup \overline{B} = \Sigma^*$, every string is enumerated by $E_A$ or $E_B$ (or both). Hence, $M$ will halt on all inputs, so $M$ is a decider for language $C$.

We now need to show that $C$ separates $A$ and $B$. Since every string in $A$ is in $\overline{B}$, the output of $E_B$ contains all strings of $A$. Thus, $M$ accepts all strings that are output by only $E_B$, so $M$ accepts all strings of $A$ since $E_A$ never outputs any strings in $A$. Likewise, since every string in $B$ is in $\overline{A}$, the output of $E_A$ contains all strings of $B$. But $M$ rejects all strings that are output by only $E_A$, so $M$ rejects all strings in $B$ since $E_B$ never outputs strings from $B$. Thus, $M$ accepts all strings in $A$ and rejects all strings in $B$, so its language $C$ separates $A$ and $B$.

Note that we did not prove which set $C$ of strings $M$ accepted. The particular language of $C$ depends on the order of the outputs of the enumerators. However, the only strings in question are the strings that are in $\overline{A} \cap \overline{B}$. Whether these strings are in $C$ or in $\overline{C}$ is not relevant to the question of separating $A$ and $B$. 