Homework 8 Solutions

1. Consider the decision 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 decider \( F \) from Theorem 4.5 on input \( \langle M, D_R \rangle \).
3. If \( F \) accepts, accept. If \( F \) rejects, reject.”

2. Consider the decision problem of testing whether a CFG generates the empty string. Express this problem as a language and show that it is decidable.

Answer: The language of the decision problem is

\[ A_{\varepsilon_{\text{CFG}}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates } \varepsilon \}. \]

If a CFG \( G = (V, \Sigma, R, S) \) includes the rule \( S \rightarrow \varepsilon \), then clearly \( G \) can generate \( \varepsilon \). But \( G \) could still generate \( \varepsilon \) even if it doesn’t include the rule \( S \rightarrow \varepsilon \). For example, if \( G \) has rules \( S \rightarrow XY, X \rightarrow aY | \varepsilon, \) and \( Y \rightarrow baX | \varepsilon \), then the derivation \( S \Rightarrow XY \Rightarrow \varepsilon Y \Rightarrow \varepsilon \varepsilon = \varepsilon \) shows that \( \varepsilon \in L(G) \), even though \( G \) doesn’t include the rule \( S \rightarrow \varepsilon \). So it’s not sufficient to simply check if \( G \) includes the rule \( S \rightarrow \varepsilon \) to determine if \( \varepsilon \in L(G) \).

But if we have a CFG \( G' = (V', \Sigma, R', S') \) that is in Chomsky normal form, then \( G' \) generates \( \varepsilon \) if and only if \( S' \rightarrow \varepsilon \) is a rule in \( G' \). Thus, we first convert the CFG \( G \) into an equivalent CFG \( G' = (V', \Sigma, R', S') \) in Chomsky normal form. If \( S' \rightarrow \varepsilon \) is a rule in \( G' \), 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 \( S' \rightarrow \varepsilon \), so the only way we can have \( \varepsilon \in L(G') \) is if \( G' \) includes the rule \( S' \rightarrow \varepsilon \) in \( R \). Hence, if
$G'$ does not include the rule $S' \rightarrow \varepsilon$, then $\varepsilon \not\in L(G')$. Thus, a Turing machine that decides $A_{\varepsilon_{\text{CFG}}}$ is as follows:

$$M = \text{"On input } \langle G \rangle, \text{ where } G \text{ is a CFG:}\n$$

1. Convert $G$ into an equivalent CFG $G' = (V', \Sigma, R', S')$ in Chomsky normal form.
2. If $G'$ includes the rule $S' \rightarrow \varepsilon$, accept. Otherwise, reject.”

3. Let $\Sigma = \{0, 1\}$, and consider the decision problem of testing whether a regular expression with alphabet $\Sigma$ generates at least one string $w$ that has 111 as a substring. Express this problem as a language and show that it is decidable.

**Answer:** The language of the decision problem is

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

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

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 $E_{\text{DFA}} = \{ \langle B \rangle | B \text{ is a DFA with } L(B) = \emptyset \}$ 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 $A$:

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

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:

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

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

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

(Actually, \( \overline{E_{TM}} \) 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 \( \overline{E_{TM}} \), the complement of \( E_{TM} \). 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 \( \langle M \rangle \in \overline{E_{TM}} \); if \( M \) accepts none of the strings, then \( L(M) = \emptyset \), so \( \langle M \rangle \notin \overline{E_{TM}} \). 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 \( \langle M \rangle \in \overline{E_{TM}} \). 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 \( \overline{E_{TM}} \):

\[ R = \text{ "On input } \langle M \rangle, \text{ where } M \text{ is a Turing machine:} \]

1. Repeat the following for \( i = 1, 2, 3, \ldots \).
2. Run \( M \) for \( i \) steps on each input \( s_1, s_2, \ldots, s_i \).
3. If any computation accepts, accept.