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 = "On input $\langle M, R\rangle$, where M is a DFA and R 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 $S \to \varepsilon$, then clearly G can generate ε. But G could still generate ε even if it doesn't include the rule $S \to \varepsilon$. For example, if G has rules $S \to XY$, $X \to aY \mid \varepsilon$, and $Y \to baX \mid \varepsilon$, 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 $S \to \varepsilon$. So it's not sufficient to simply check if G includes the rule $S \to \varepsilon$ 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 ε if and only if $S' \to \varepsilon$ 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 $S' \to \varepsilon$ is a rule in G', then clearly G' generates ε, so G also generates ε since $L(G) = L(G')$. Since G' is in Chomsky normal form, the only possible ε-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 = "On input $\langle G \rangle$, where G is a CFG:

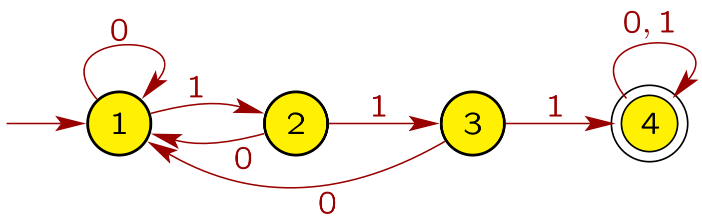
1. Convert G into an equivalent CFG $G' = (V', \Sigma, R', S')$ in Chomsky normal form.
2. If G' includes the rule $S' \to \varepsilon$, *accept*. Otherwise, *reject*."

3. Let $\Sigma = \{0, 1\}$, and consider the decision problem of testing whether a regular expression with alphabet Σ 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

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

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:



Now consider any regular expression R with alphabet Σ. 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 $\langle R \rangle \notin A$. 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_{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 = "On input $\langle R \rangle$, where R 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_{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:

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

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

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

(Actually, $\overline{E_{\rm TM}}$ 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 $\overline{E_{\rm TM}}$, the complement of $E_{\rm TM}$. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^*. 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 $\langle M \rangle \in \overline{E_{\rm TM}}$; if M accepts none of the strings, then $L(M) = \emptyset$, so $\langle M \rangle \notin \overline{E_{\rm TM}}$. 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 $\langle M \rangle \in \overline{E_{\rm TM}}$. 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 $\overline{E_{\rm TM}}$:

R = "On input $\langle M \rangle$, where M 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*.

5. Recall that the equivalence problem for DFAs has language

$$\begin{aligned} EQ_{\rm DFA} &= \{\, \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \,\} \\ &\subseteq \{\, \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs} \,\} \equiv \Omega. \end{aligned}$$

In other words, for any $\langle A, B \rangle \in \Omega$, the pair $\langle A, B \rangle \in EQ_{\rm DFA}$ if and only if A and B *agree* on every possible input string $s \in \Sigma^*$, where Σ is the input alphabet for both A and B. Specifically, we say that A and B *agree* on a string s if A and B both accept s or both reject s. Similarly, A and B *disagree* on s if one of the DFAs accepts and the other rejects. Theorem 4.5 establishes that $EQ_{\rm DFA}$ is decidable. The proof builds a DFA C for the symmetric difference of $L(A)$ and $L(B)$, which is possible because the class of regular languages is closed under complementation, intersection, and union. Then it checks if the symmetric difference is empty using the decider for $E_{\rm DFA}$.

Rather than using the proof that was given for Theorem 4.5, suppose we instead try to prove that $EQ_{\rm DFA}$ is decidable using the following TM M' that checks if the two inputted DFAs A and B always agree on every possible input string. Let $s_1, s_2, \ldots$ be an enumeration of Σ^*, and define TM M' as follows:

M' = "On input $\langle A, B\rangle \in \Omega$, where A and B are DFAs:

0. Check if $\langle A, B\rangle$ is a proper encoding of 2 DFAs. If not, *reject*.

1. Repeat the following for $i = 1, 2, 3, \ldots$

2. Run A and B on s_i.

3. If A and B disagree on s_i, then *reject*.

4. If A and B always agree on each s_i, then *accept*. "

What is the problem with this approach?

Answer: The problem is that M' loops on every $\langle A, B\rangle \in EQ_{\text{DFA}}$, so M' does not even *recognize* EQ_{DFA}, so it cannot possibly *decide* EQ_{DFA}. To see why, suppose that $\langle A, B\rangle \in EQ_{\text{DFA}}$, so A and B agree on every $s_i \in \Sigma^*$. Thus, M' will be stuck in a loop in Stages 1–3, trying the infinitely many strings in Σ^* but never finding one on which A and B disagree. Hence, M' never gets to Stage 4 to accept, so M' loops on every YES instance $\langle A, B\rangle \in EQ_{\text{DFA}}$; i.e., M' does not recognize EQ_{DFA}.