Assignment 3 solutions

1. For this problem you will be asked to prove that three languages are decidable. You may do this by giving high-level descriptions of Turing machines that decide these languages. Please try to keep your Turing machines as simple as possible—making use of Turing machines described in the lecture notes as subroutines may help in this regard.

(a) Prove that the following language is decidable:
   \[
   \{ \langle D, k \rangle : D \text{ is a DFA, } k \in \mathbb{N}, \text{ and } |L(D)| \geq k \}.
   \]

(b) Prove that the following language is decidable:
   \[
   \left\{ \langle D \rangle : D \text{ is a DFA with alphabet } \{0, 1\}, \text{ and every } w \text{ accepted by } D \text{ satisfies } |w|_0 \neq |w|_1 \right\}.
   \]

(c) Prove that the following language is decidable:
   \[
   \{ \langle G \rangle : G \text{ is a CFG and } L(G) \text{ is finite} \}.
   \]

Solution. There are, of course, multiple ways of answering each of the parts of this question. The answers that follow each represent just one way for each of the parts.

For part (a), let \( M \) be a DTM operating as follows:

On input \( \langle D, k \rangle \), where \( D \) is a DFA and \( k \) is a natural number:

1. Set \( n \leftarrow k \) and \( C \leftarrow D \).
2. If \( n = 0 \), then accept.
3. If \( \langle C \rangle \in E_{DFA} \), then reject.
4. Find the first string accepted by \( C \):
   a. Set \( x \leftarrow \varepsilon \).
   b. While \( \langle C, x \rangle \notin A_{DFA} \) do:
      Increment \( x \) (with respect to the lexicographic ordering of the alphabet of \( C \)).
5. Construct a DFA for the language \( L(C) \setminus \{x\} \), and set \( C \) to be this new DFA.
6. Set \( n \leftarrow n - 1 \), and goto 2.

One way to perform the construction of a DFA for the language \( L(C) \setminus \{x\} \) is as follows. First, assuming \( \Sigma \) is the alphabet of \( C \), construct a DFA \( E \) for the language \( \Sigma^* \setminus \{x\} \), which is straightforward. Then, using the Cartesian product construction described in Lecture 4, construct a DFA for the intersection of the languages recognized by \( C \) and \( E \).

It is evident from an inspection of \( M \) that this DTM decides the language described in part (a), and therefore that language is decidable.
For part (b), let $M$ be a DTM operating as follows:

On input $\langle D \rangle$, where $D$ is a DFA:

1. Let $G$ be the CFG for the language $\{w \in \{0, 1\}^* : |w|_0 = |w|_1\}$ from Lecture 7.
2. Construct a CFG $H$ for the language $L(G) \cap L(D)$ using the construction discussed in lecture (when we proved that the intersection of a context-free language and a regular language is context-free).
3. Decide whether or not $\langle H \rangle \in \mathbb{E}_{\text{CFG}}$ using the DTM for this language discussed in Lecture 14. Accept if $\langle H \rangle \in \mathbb{E}_{\text{CFG}}$, and reject otherwise.

It holds that $L(H)$ is empty if and only if every string accepted by $D$ lies outside of the language $L(G) = \{w \in \{0, 1\}^* : |w|_0 = |w|_1\}$, and therefore $M$ decides the given language.

For part (c), we will make use of the following fact:

If $G$ is a context-free grammar in Chomsky normal form having $m$ variables, then $L(G)$ is infinite if and only if there exists a string $x \in L(G)$ such that $|x| \geq 2^m$.

This fact follows from the proof of the pumping lemma for context-free languages from Lecture 10. With this fact in mind, the following DTM decides $B$:

On input $\langle G \rangle$ where $G$ is a CFG:

1. Convert $G$ to an equivalent CFG $H$ in Chomsky normal form, and let $m$ be the number of variables in $H$.
2. Construct a DFA $D$, having the same alphabet as $H$, that recognizes the language of all strings having length at least $2^m$ (which is a regular language).
3. Construct a new CFG $K$ for the language $L(H) \cap L(D)$. This is possible through the construction we discussed in Lecture 9 when proving that the intersection of a context-free language and a regular language is context-free.
4. Decide whether or not $\langle K \rangle \in \mathbb{E}_{\text{CFG}}$ using the DTM for this language discussed in Lecture 14. Accept if $\langle K \rangle \in \mathbb{E}_{\text{CFG}}$, and reject otherwise.

In more detail, if the CFG $K$ generates any strings, then $H$ must generate a string having length at least $2^m$, and therefore $H$ generates an infinite number of strings by the fact above (and so does $G$ because $H$ and $G$ are equivalent). If $K$ does not generate any strings, then $H$ must generate a finite number of strings, because there are only finitely many strings in total having length less than $2^m$.

An alternative approach to this problem is to directly design an algorithm that inspects the rules of a CFG $G$ (or an equivalent CFG in Chomsky normal form) in various ways. For instance, your solution might have involved a search for a cycle among a collection of variables reachable from the start variable. By using the same algorithm used to decide $\mathbb{E}_{\text{CFG}}$, you could throw away all variables that are not capable of generating any strings, so as to potentially simplify the algorithm.
2. Define a language

\[ A = \{ \langle M \rangle : M \text{ is a DTM that halts on at least one input string} \} \, . \]

Prove that \( A \) is Turing-recognizable.

**Solution.** It suffices for us to define a DTM \( K \) for which \( L(K) = A \). Let us define \( K \) to operate as follows:

On input \( \langle M \rangle \), where \( M \) is a DTM:
1. Set \( t = 1 \).
2. Simulate \( M \) for \( t \) steps on every input string \( x \) (over the alphabet of \( M \)) with \( |x| \leq t \).
   If \( M \) halts during any of these simulations, then \textit{accept}.
3. Set \( t = t + 1 \) and goto step 2.

It is evident from an inspection of \( K \) that \( L(K) = A \), and therefore \( A \) is Turing-recognizable.

3. Let \( \Sigma \) be an alphabet, and suppose that \( A, B \subseteq \Sigma^* \) are Turing-recognizable languages for which both \( A \cap B \) and \( A \cup B \) are decidable. Prove that \( A \) is decidable.

**Solution.** Given that \( A \) and \( B \) are Turing-recognizable, there exist DTMs \( M_A \) and \( M_B \) such that \( L(M_A) = A \) and \( L(M_B) = B \). Define a DTM \( M \) as follows:

On input \( x \in \Sigma^* \):
1. If \( x \in A \cap B \), then \textit{accept}.
2. If \( x \not\in A \cup B \), then \textit{reject}.
3. Set \( t \leftarrow 1 \).
4. Run \( M_A \) for \( t \) steps on input \( x \): if \( M_A \) accepts \( x \) within \( t \) steps, then \textit{accept}.
5. Run \( M_B \) for \( t \) steps on input \( x \): if \( M_B \) accepts \( x \) within \( t \) steps, then \textit{reject}.
6. Set \( t \leftarrow t + 1 \) and goto 4.

Steps 1 and 2 can be performed by \( M \) under the assumption that \( A \cap B \) and \( A \cup B \) are decidable. If the DTM reaches step 3, then it must be the case that \( x \) is contained in the symmetric difference \( A \triangle B \), so we are guaranteed that exactly one of the DTMs \( M_A \) or \( M_B \) will eventually accept \( x \). When we reach a large enough value of \( t \) for this to happen, we are then able to correctly decide whether or not \( x \) is in \( A \). Because \( M \) decides \( A \), we have that \( A \) is decidable.
4. Prove that the following language is not Turing recognizable:

\[ \{ \langle M \rangle : M \text{ is a DTM such that } |L(M)| = 1 \} \]

Hint: try to come up with a DTM that recognizes DIAG using a DTM that recognizes the language above as a subroutine.

**Solution.** Let us assume toward contradiction that this language is Turing recognizable, so that there exists a DTM $K$ that recognizes it. We will use this DTM to come up with a new DTM $J$ satisfying $L(J) = \text{DIAG}$. Because we already know that DIAG is not Turing recognizable, this will give us a contradiction, therefore implying that the language in the statement of the question is not Turing recognizable.

Before describing the DTM $J$, let us consider the following simple way of modifying an arbitrary DTM. If $M$ is any DTM, then we may consider a new DTM $M_\varepsilon$ that operates as follows:

On input $w$, where $w$ is a string over the alphabet of $M$:

- If $w = \varepsilon$, then accept, otherwise run $M$ on input $\langle M \rangle$.

Notice that if $M$ accepts $\langle M \rangle$, then $M_\varepsilon$ accepts every input string, and if $M$ does not accept $\langle M \rangle$ then $M_\varepsilon$ accepts just the empty string. It therefore it holds that

\[ |L(M_\varepsilon)| = 1 \iff \langle M \rangle \in \text{DIAG}. \]

We may also observe that if you are given the description $\langle M \rangle$ of a DTM, then it is easy to compute a description $\langle M_\varepsilon \rangle$ of $M_\varepsilon$.

Now, here is a description of the DTM $J$ (which uses the DTM $K$ recognizing the language in the statement of the problem as a subroutine):

On input $\langle M \rangle$, where $M$ is a DTM:

1. Compute the description $\langle M_\varepsilon \rangle$ of the DTM $M_\varepsilon$ described above.
2. Run $K$ on input $\langle M_\varepsilon \rangle$.

Under the assumption that $K$ recognizes the language described in the problem, it holds that $J$ accepts $\langle M \rangle$ if and only if $|L(M_\varepsilon)| = 1$, and therefore $J$ accepts $\langle M \rangle$ if and only if $\langle M \rangle \in \text{DIAG}$. This contradicts the fact that DIAG is not Turing recognizable, and therefore the language in the statement of the problem is not Turing recognizable.