Assignment 4 solutions

1. Define a language
   \[ \text{ALL} = \{ \langle M \rangle : M \text{ is a DTM that accepts every string over its input alphabet} \} \].
   Prove that \( A \) is not Turing recognizable.
   
   **Solution.** Let us prove that \( \text{DIAG} \leq_m \text{ALL} \).
   Because DIAG is not Turing recognizable, this will imply that ALL is not Turing recognizable, as required.
   
   First, for an arbitrary DTM \( M \), let us define a new DTM \( K_M \) having input alphabet \( \{0\} \) as follows:
   
   On input \( 0^t \):
   1. Run \( M \) on input \( \langle M \rangle \) for \( t \) steps.
   2. If \( M \) accepts \( \langle M \rangle \) within \( t \) steps, then reject, otherwise accept.
   
   The function \( f(\langle M \rangle) = \langle K_M \rangle \) is computable—it would be a simple (but perhaps tedious) computational process to transform a description of \( M \) into a description of \( K_M \). (As usual when doing reductions such as this, if a string \( x \) does not encode a DTM at all, then we define \( f(x) \) to be some fixed but otherwise arbitrary string that is not contained in ALL.)
   
   It remains to observe that \( f \) is a reduction from DIAG to ALL. If \( \langle M \rangle \in \text{DIAG} \), then \( M \) does not accept \( \langle M \rangle \), and therefore \( K_M \) will accept every input \( 0^t \). It follows that \( \langle K_M \rangle = f(\langle M \rangle) \in \text{ALL} \). On the other hand, if \( \langle M \rangle \notin \text{DIAG} \), then \( M \) accepts \( \langle M \rangle \). This means that for any sufficiently large \( t \), the DTM \( K_M \) will reject any input \( 0^t \). (In particular, if \( t \) is at least as large as the number of steps \( M \) needs to accept \( \langle M \rangle \), then \( K_M \) will reject \( 0^t \).) This implies that \( \langle K_M \rangle = f(\langle M \rangle) \notin \text{ALL} \). We have proved that
   \[ \langle M \rangle \in \text{DIAG} \iff f(\langle M \rangle) \in \text{ALL}, \]
   and therefore \( \text{DIAG} \leq_m \text{ALL} \).

2. Define two languages \( A, B \subseteq \Sigma^* \) as follows:
   \[ A = \{ \langle M \rangle : M \text{ is a DTM that rejects } \langle M \rangle \}, \]
   \[ B = \{ \langle M \rangle : M \text{ is a DTM that accepts } \langle M \rangle \}. \]
   Prove that there does not exist a decidable language \( C \subseteq \Sigma^* \) such that \( A \subseteq C \) and \( B \subseteq \overline{C} \).
   
   **Solution.** Assume toward contradiction that there exists a decidable language \( C \) such that \( A \subseteq C \) and \( B \subseteq \overline{C} \). By the assumption that \( C \) is decidable, there exists a DTM \( M \) that decides \( C \). For this DTM we have
   \[ \langle M \rangle \in C \Rightarrow M \text{ accepts } \langle M \rangle \Rightarrow \langle M \rangle \in B \Rightarrow \langle M \rangle \in \overline{C} \Rightarrow \langle M \rangle \notin C \]
   and
   \[ \langle M \rangle \notin C \Rightarrow M \text{ rejects } \langle M \rangle \Rightarrow \langle M \rangle \in B \Rightarrow \langle M \rangle \in C. \]
   Therefore \( \langle M \rangle \in C \) if and only if \( \langle M \rangle \notin C \), which is a contradiction. It follows that there does not exist a decidable language \( C \) such that \( A \subseteq C \) and \( B \subseteq \overline{C} \).
3. Define

\[ A_{\text{DTM}} = \{ \langle M, w \rangle : M \text{ is a DTM and } w \in L(M) \} \]

as usual, and let \( \Sigma \) be the alphabet over which this language is defined (i.e., \( \Sigma \) is the alphabet that is used for the encoding scheme mentioned in the definition of \( A_{\text{DTM}} \)). Let \( B \subseteq \Sigma^* \) be any Turing-recognizable language over the same alphabet \( \Sigma \). Prove that \( B \leq_m A_{\text{DTM}} \).

**Solution.** In class we proved that if \( C \leq_m D \) for two languages \( C \) and \( D \), then \( D \) Turing recognizable \( \Rightarrow \) \( C \) Turing recognizable.

Because \( A_{\text{DTM}} \) is Turing recognizable, it follows from the assumption \( B \leq_m A_{\text{DTM}} \) that \( B \) is Turing-recognizable.

On the other hand, if \( B \) is Turing-recognizable, then there is a DTM \( M \) with \( B = L(M) \). Define a computable function \( f \) as follows:

\[ f(x) = \langle M, x \rangle. \]

For any input \( x \), we have \( x \in B \iff f(x) \in A_{\text{DTM}} \). Therefore, \( B \leq_m A_{\text{DTM}} \).

4. Let \( \Sigma = \{0, 1\} \), let \( A \subseteq \Sigma^* \) be a language contained in \( \text{DTIME}(4^n) \), and define

\[ B = \{xx : x \in A\}. \]

(a) Give a high-level argument supporting the claim that \( B \in \text{DTIME}(2^n) \).

(b) Prove that \( A \leq^p_B \).

(c) Prove that \( \text{DTIME}(2^n) \neq \text{NP} \).

You may use the following fact (which is stated in the Lecture 19 notes without proof) without proving it: if \( C, D \subseteq \{0, 1\}^* \) are languages, \( D \in \text{NP} \), and \( C \leq^p_D \), then \( C \in \text{NP} \).

**Solution.** For part (a), suppose first that \( M_A \) is a DTM that decides the language \( A \) in time \( O(4^n) \).

Define a DTM \( M_B \) as follows:

On input \( w \in \Sigma^* \):

1. If \( w \) takes the form \( w = xx \) for some string \( x \in \Sigma^* \), then erase the second half of \( w \), leaving just \( x \) on the tape. Otherwise, reject.

2. Run \( M_A \) on \( x \).

The first step of this computation can be performed in \( O(n^2) \) steps, for \( n = |w| \). If the second step is performed, then it requires \( O(4^{n/2}) = O(2^n) \) steps, as it is a computation of \( M_A \) (which runs in time \( O(4^m) \) on inputs of length \( m \), run on a string of length \( n/2 \)). The running time of \( M_B \) is therefore \( O(2^n) \). It is immediate from an inspection of \( M_B \) that it decides \( B \), and therefore \( B \in \text{DTIME}(2^n) \).

For part (b), define a function \( f : \Sigma^* \rightarrow \Sigma^* \) as

\[ f(x) = xx \]

for all \( x \in \Sigma^* \). The function \( f \) can be computed in polynomial time. (More specifically, it can be computed in \( O(n^2) \) time.) It is immediate from the definition of \( B \) that

\[ x \in A \iff f(x) \in B \]
and therefore we have that $A \leq^p_m B$.

For part (c), choose a language $A \in \text{DTIME}(4^n)$ that is not contained in DTIME$(2^n)$. Such a language exists by the time hierarchy theorem, by virtue of the fact that

$$2^n \in o\left(\frac{4^n}{\log(4^n)}\right).$$

Also define $B$ as in the problem statement, for the language $A$ just selected. By parts (a) and (b) we conclude that $B \in \text{DTIME}(2^n)$ and $A \leq^p_m B$, but we already know that $A \not\in \text{DTIME}(2^n)$. The complexity class DTIME$(2^n)$ therefore does not have the property suggested in the statement of the problem that holds for NP. It follows that DTIME$(2^n)$ and NP are not the same classes, and so we are done.

5. Let $\Sigma = \{0, 1\}$ and let $A, B \subseteq \Sigma^*$ be languages. Prove that if $A$ is NP-hard, $B$ is in P, $A \cap B = \emptyset$, and $A \cup B \neq \Sigma^*$, then $A \cup B$ is NP-hard.

**Solution.** As $A$ is NP-hard, it is enough to show that $A \leq^p_m A \cup B$. Define a function $f$ as

$$f(x) = \begin{cases} 
  y & \text{if } x \in B \\
  x & \text{if } x \not\in B,
\end{cases}$$

where $y$ is any fixed string that is not in $A \cup B$. Such a string must exist because $A \cup B \neq \Sigma^*$, and it may be noted that the reduction $f$ does not need to search for $y$ or compute it in some other way—the string $y$ exists, so it may be considered as fixed and can be hard-coded into a DTM that computes $f$. Given that $B \in \text{P}$, we have that $f$ is polynomial-time computable.

If $x \in A$, then $f(x) = x$ (because $A \cap B = \emptyset$), and therefore $f(x) \in A \cup B$.

If $x \not\in A$, then there are two cases: $x \in B$ and $x \not\in B$. If $x \in B$, then $f(x) = y \not\in A \cup B$. If $x \not\in B$, then $f(x) = x$, and because $x \not\in A$ and $x \not\in B$ we have $f(x) \not\in A \cup B$.

Thus,

$$x \in A \iff f(x) \in A \cup B,$$

and therefore $A \leq^p_m A \cup B$. 
