Problem 1

Prove that for any two languages \( A, B \), if \( A \leq_m B \) then \( \overline{A} \leq_m \overline{B} \)

Proof idea

We suppose \( A \leq_m B \) and infer the existence of a computable function. The same function allows us to infer \( \overline{A} \leq_m \overline{B} \).

Proof. Suppose \( A \leq_m B \). Then there is some function \( f \) such that

\[
\begin{align*}
    w \in A &\rightarrow f(w) \in B \quad \text{and} \quad (1) \\
    w \notin A &\rightarrow f(w) \notin B \quad \text{and} \quad (2)
\end{align*}
\]

Further, for any set \( X \),

\[
    w \in X \iff w \notin \overline{X} \quad (3)
\]

So, by (1) and (3),

\[
    w \notin \overline{A} \rightarrow f(w) \notin \overline{B}
\]

and, by (2) and (3),

\[
    w \in \overline{A} \rightarrow f(w) \in \overline{B}
\]

Since \( f \) is computable by supposition, \( \overline{A} \leq_m \overline{B} \). \( \square \)
Problem 2

Consider the hometown problem from recitation where, a FLINT machine is a Turing machine that does not accept the string “ben”, and a BOSTON machine is a Turing machine that does accept the string “ben”.

Consider the language

\[ HOMETOWN = \{ \langle F, B \rangle \mid F \text{ is a FLINT machine and } B \text{ is a BOSTON machine} \} \]

Prove that \( HOMETOWN \) does not many one reduce to \( HALT \), \( HOMETOWN \not\leq_m HALT \).

Proof idea

We will show that \( HALT \) does many-one reduce to \( HOMETOWN \). Since \( HALT \) does not many-one reduce to \( HALT \), \( HOMETOWN \) does not many-one reduce to \( HALT \).

More formally, \( HOMETOWN \) is not computably enumerable, and \( HALT \) is. For any \( X \) and \( Y \), if \( X \leq_m Y \) and \( Y \) is computably enumerable, then \( X \) is computably enumerable. Therefore, \( HOMETOWN \not\leq_m HALT \).

Proof. \( HALT \leq_m HOMETOWN \) if and only if, there is a function \( f \) such that for any \( w \in \Sigma^* \)

\[ w \in \overline{HALT} \iff f(w) \in HOMETOWN \]

Let \( f((M, w)) = \langle N, N' \rangle \) with \( N \) and \( N' \) defined as follows.

Define \( N \): "on input \( x \)

i. Run \( M \) on input \( w \).
   (a) If \( M \) accepts, ACCEPT.
   (b) if \( M \) rejects, ACCEPT."

Define \( N' \): "on input \( x \)

i. ACCEPT."

Since \( N \) and \( N' \) are valid Turing machines (all operations are finite, no loops are detected, etc.), \( f \) is computable.

Suppose \( \langle M, w \rangle \in HALT \). Then, \( N \) loops on all inputs (including ‘ben’) and \( N' \) accepts all inputs (including ‘ben’). Therefore, \( N \) is a FLINT machine and \( N' \) is a BOSTON machine. So \( f((M', w)) \in HOMETOWN \).

Suppose \( \langle M, w \rangle \not\in HALT \). Then \( \langle M, w \rangle \in HALT \) and both \( N \) and \( N' \) accept any input (including ‘ben’). So \( N \) is not a FLINT machine. Therefore, \( f((M', w)) \not\in HOMETOWN \).

Suppose \( HOMETOWN \leq_m HALT \). Then, since \( HALT \) is computably enumerable, \( HOMETOWN \) is computably enumerable. Since \( HALT \leq_m HOMETOWN \), \( HOMETOWN \) is not computably enumerable. Therefore, \( HALT, HOMETOWN \not\leq_m HALT \). □
Problem 3

Prove that $HOMETOWN \leq_T HALT$

Definition

$A$ is Turing reducible to $B$ ($A \leq_T B$) if $A$ is decidable relative to $B$. In other words, if a decision procedure exists for $B$, then one also exists for $A$.

An oracle for a language $A$ is such that when ‘queried’ about a particular $w \in \Sigma^*$, always responds either ‘yes’ or ‘no’ depending on whether or not $w \in A$.

Proof idea

Suppose an oracle exists for $HALT$. Define machine $M^{HALT}_{HOME}$ as follows.

Define $M^{HALT}_{HOME}$: "on input $\langle M, N \rangle$

i. Query $\langle M, \text{‘ben’} \rangle \in HALT$.

ii. If ‘no’, CONTINUE.

iii. If ‘yes’, run $M$ on ‘ben’.

   (a) If $M$ accepts, REJECT.
   (b) If $M$ rejects, CONTINUE.

iv. Query $\langle N, \text{‘ben’} \rangle \in HALT$.

v. If ‘no’, REJECT.

vi. If ‘yes’, run $N$ on ‘ben’

   (a) If $N$ accepts, ACCEPT.
   (b) If $N$ rejects, REJECT.

Suppose $\langle M, N \rangle \in HOMETOWN$. Then, on input ‘ben’, $M$ either rejects or loops, and $N$ accepts. If $M$ loops, then when queried on $\langle M, \text{‘ben’} \rangle$, the oracle responds ‘no’ and $M^{HALT}_{HOME}$ continues to instruction iii. If $M$ rejects, then when queried on $\langle M, \text{‘ben’} \rangle$, the oracle responds ‘yes’ and $M^{HALT}_{HOME}$ continues to instruction iii. Since $N$ accepts, when queried on $\langle N, \text{‘ben’} \rangle$, the oracle responds ‘yes’ and $M^{HALT}_{HOME}$ accepts.

Suppose $\langle M, N \rangle \not\in HOMETOWN$. Then, on input ‘ben’, either $M$ accepts, or $N$ rejects or loops. If $M$ accepts, then when queried on $\langle M, \text{‘ben’} \rangle$, the oracle responds ‘yes’, and $M^{HALT}_{HOME}$ rejects. If $N$ loops, then when queried on $\langle N, \text{‘ben’} \rangle$, the oracle response ‘no’ and $M^{HALT}_{HOME}$ rejects. If $N$ rejects, then when queried on $\langle N, \text{‘ben’} \rangle$, the oracle responds ‘yes’ and $M^{HALT}_{HOME}$ rejects.

So, $M^{HALT}_{HOME}$ accepts if $\langle M, N \rangle \in HOMETOWN$, and $M^{HALT}_{HOME}$ rejects if $\langle M, N \rangle \not\in HOMETOWN$.

Therefore, $HOMETOWN$ is decidable.

Therefore, $HOMETOWN \leq_T HALT$. 

3
Problem 4

Show that for any two languages $A$ and $B$, a language $J$ exists, where $A \leq_T J$ and $B \leq_T J$.

Proof idea

We designate $J$ to be the set of pairs $\langle x, A \rangle$ and $\langle y, B \rangle$ such that $x \in A$ and $y \in B$. Intuitively, we might think of $J$ as containing all the members of both $A$ and $B$, and ‘tagging’ each element with the set that it belongs to. Given such a definition, it is trivial to show that a decision procedure for $J$ yields a decision procedure for both $A$ and $B$.

Proof. Let

$$J = \{ \langle x, A \rangle \mid x \in A \} \cup \{ \langle y, B \rangle \mid y \in B \}$$

Suppose there is an oracle for $J$. We define a machine $M_A$ which decides $A$ below, and prove by cases that $M_A$ decides $A$. A machine to decide $B$, and cases for $B$, are similar.

Define $M_A$: "on input $w$

i. Query $\langle w, A \rangle \in J$.

ii. If ‘yes’, ACCEPT.

iii. If ‘no’, REJECT.

Suppose $x \in A$. Then, by definition, $\langle x, A \rangle \in J$. So, when queried on $\langle w, A \rangle$, the oracle responds ‘yes’. So, $M_A$ accepts.

Suppose $x \not\in A$. Then, by definition, $\langle x, A \rangle \not\in J$. So, when queried on $\langle w, A \rangle$, the oracle responds ‘no’. So, $M_A$ rejects.

Since the case for $B$ is similar, $A$ and $B$ are both decidable relative to $J$. So, $A \leq_T J$ and $B \leq_T J$. $\square$
Problem 5

Consider the following language

\[ L = \{ w | w = 0x \text{ for some } x \in K \text{ or } w = 1x \text{ for some } x \in \overline{K} \} \]

Show that neither \( L \) nor \( \overline{L} \) are recognizable.

Proof idea

We many-one reduce \( K \) and \( \overline{K} \) to \( L \). In reducing \( K \) to \( L \) we show, by the result in problem 1, that \( \overline{K} \) reduces to \( \overline{L} \), which shows \( \overline{L} \) to be unrecognizable. In reducing \( \overline{K} \) to \( L \), we show \( L \) to be unrecognizable.

Proof. \( K \leq_m L \).

Define \( f \):

\[ f(M) = 0\langle M \rangle \]

If \( \langle M \rangle \in K \), then \( M \) accepts on \( \langle M \rangle \), so \( 0\langle M \rangle \in L \). So, \( f(M) \in L \).

If \( \langle M \rangle \notin K \), then \( M \) does not accept on \( \langle M \rangle \), so \( 0\langle M \rangle \notin L \). So, \( f(M) \notin L \). \( \square \)

Proof. \( \overline{K} \leq_m L \).

Define \( f \):

\[ f(M) = 1\langle M \rangle \]

If \( \langle M \rangle \in \overline{K} \), then \( M \) does not accept on \( \langle M \rangle \), so \( 1\langle M \rangle \in L \). So, \( f(M) \in L \).

If \( \langle M \rangle \notin \overline{K} \), then \( M \) accepts on \( \langle M \rangle \), so \( 0\langle M \rangle \notin L \). So, \( f(M) \notin L \). \( \square \)

Proof. \( K \leq_m L \), so, by the result in problem 1, \( \overline{K} \leq_m \overline{L} \). Since \( K \) is undecidable, \( \overline{L} \) is undecidable. Also, since \( \overline{K} \leq_m L \), and \( \overline{K} \) is undecidable, \( L \) is undecidable. \( \square \)