Midterm 2

Problem 1 (33 points)

Let \( C = \{ \langle M_1, M_2 \rangle \mid L(M_1) = \overline{L(M_2)} \} \), that is \( \langle M_1, M_2 \rangle \in C \) iff for all \( x \in \{0,1\}^* \), exactly one of the two machines accepts \( x \). Show that \( C \) is not Turing-recognizable. You can use any statement proved in class or in the homeworks about any particular language being not Turing recognizable. Also, we will give partial credit for proving that \( C \) is just not decidable.

Solution. We give a reduction from the complement of the Acceptance problem, that we know to be not recognizable.

Given a Turing machine \( M \) and an input \( w \), we define the following two Turing machines \( M_{1,w} \) and \( M_2 \):

- On input \( x \), \( M_{1,w} \) compares \( x \) with \( w \). If \( x \neq w \) then \( M_{1,w} \) rejects, otherwise it simulates \( M \) on input \( w \).
- On input \( x \), \( M_2 \) always accepts.

Now, we have that \( \langle M_{1,w}, M_2 \rangle \in C \) if and only if \( M_{1,w} \) accepts no string, if and only if \( M \) does not accept \( w \).

This is a mapping reduction from the complement of the Acceptance problem to \( C \), and since the Acceptance problem is not Turing-recognizable it follows that \( C \) is not Turing-recognizable.

Remarks.

1. Several students first showed that \( C \) is undecidable by giving a reduction from \( A_{TM} \), and then tried to show that \( \overline{C} \) is Turing-recognizable. The latter is not true. Note that \( A_{TM} \leq_M C \) implies \( \overline{A_{TM}} \leq_M \overline{C} \), and since \( A_{TM} \) is not Turing-recognizable, \( \overline{C} \) is also not Turing-recognizable.

2. Several students gave correct reductions from \( E_{TM} \) to \( C \). We grant full credit if it is also noted that \( E_{TM} \) is not Turing-recognizable (this is because \( E_{TM} \) is not decidable, and \( \overline{E_{TM}} \) is Turing-recognizable).

3. Note that if we take a Turing machine \( M \) and switch its accept and reject states to obtain a new machine \( M' \), it is not necessarily the case that \( L(M') = \overline{L(M)} \). This is because if \( M \) loops on some input \( w \), \( M' \) would also loop on \( w \), and both \( M \) and \( M' \) would reject \( w \).

Problem 2 (33 points)

Let \( L \) be a decidable language such that for all positive integers \( n \), \( |L \cap \{0,1\}^n| \leq 2^{n/2} \). Prove that \( L \) contains finitely many incompressible strings, that is, prove that the set \( \{ x : x \in L \text{ and } K(x) \geq |x| \} \) is finite, where \( |x| \) denotes the length of \( x \).
Solution. Let $M$ be the Turing machine that on input the integers $(n, i)$ finds the $i$-th string of $L \cap \{0, 1\}^n$ in lexicographic order. Since $L$ is decidable, $M$ is well defined.

Now, every string $x \in L \cap \{0, 1\}^n$ is described by the triple $(M, n, i)$, where $1 \leq i \leq 2^n/2$, so that the Kolmogorov complexity of such an $x$ is at most $n/2 + 2\log n + c$, where $c$ is the constant length of the encoding of $M$.

If $L$ contained infinitely many incompressible strings, there would be infinitely many $n$ such that for some string $x$ of length $n$ we have $n \leq K(x) \leq n/2 + 2\log n + c$, which is clearly impossible because for sufficiently large $n$ we have $n \geq n/2 + 2\log n + c$.

Remarks. A slightly different variant of the solution is as follows: take the machine $M'$ that on input the integer $i$, finds the $i$-th string of $L$ in lexicographic order. Then, every string of $L \cap \{0, 1\}^n$ has a description of length $n/2 + O(1)$, because $\sum_{j=0}^{n} 2^{j/2} = \theta(2^{n/2})$.

Problem 3 (33 points)

Recall the SUBSET-SUM problem concerning integer arithmetic. In that problem, we have a collection of numbers $x_1, \ldots, x_k$ and a target number $t$, and we want to determine whether the collection contains a subcollection that adds up to $t$.

Now, consider the language:

$$1/3\text{-PARTITION} = \{(x_1, \ldots, x_n) \mid \text{for some } S \subseteq \{1, \ldots, n\}, \text{ we have } \sum_{i \in S} x_i = \frac{1}{3} \sum_{j=1}^{n} x_j\}$$

Prove that $1/3\text{-PARTITION}$ is NP-complete.

Solution. We give a reduction from PARTITION. Let $I = (a_1, \ldots, a_n)$ be an instance of PARTITION, and let $A$ be the sum of the elements. We define an instance of $1/3\text{-PARTITION}$ $I' = (a_1, \ldots, a_n, a_{n+1}, a_{n+2})$ where $a_{n+1} = 1.5 \cdot A$ and $a_{n+2} = 3.5 \cdot A$. Note that the sum of the integers in $I'$ is $6A$, and so $I'$ is a YES-instance of $1/3\text{-PARTITION}$ if and only if there is a subset of integers that sum to $2A$.

If $I$ is a YES-instance of PARTITION, then $\sum_{i \in S} a_i = A/2$ for some set $S \subseteq \{1, \ldots, n\}$. Define $S' = S \cup \{a_{n+1}\}$, then we have $\sum_{i \in S'} a_i = 2A$ and so $I'$ is a YES-instance of $1/3\text{-PARTITION}$.

If $I'$ is a YES-instance of $1/3\text{-PARTITION}$, then for some set $S' \subseteq \{1, \ldots, n + 2\}$ we have $\sum_{i \in S'} a_i = 2A$. It is clear $a_{n+2} \notin S'$, because otherwise the sum would be bigger than $2A$, and it is also clear that $a_{n+1} \in S'$, because otherwise the sum would be at most $A$. Define $S = S' - \{a_{n+1}\}$. Then we have $\sum_{i \in S} a_i = 2A - a_{n+1} = A/2$, which proves that $I$ is a YES-instance of PARTITION.

We have proved that PARTITION $\leq_p 1/3\text{-PARTITION}$, and so $1/3\text{-PARTITION}$ is NP-hard.

It follows from the definition that $1/3\text{-PARTITION}$ is in NP. We can define a verifier $V$ that on input $I = (a_1, \ldots, a_n)$ and $S \subseteq \{1, \ldots, n\}$ checks that $\sum_{i \in S} a_i = (1/3) \sum_{i=1}^{n} a_i$, and this computation can clearly be carried out in polynomial time.

$1/3\text{-PARTITION}$ is in NP and it is NP-hard, therefore it is NP-complete.

Remarks. Here is an alternative reduction from SUBSET-SUM: on input $\langle(x_1, \ldots, x_k), t\rangle$, output $(x_1, \ldots, x_k, A, A+3t)$ where $A = x_1 + \cdots + x_k$. On the other hand, the reduction from SUBSET-SUM that on input $\langle(x_1, \ldots, x_k), t\rangle$ outputs $(x_1, \ldots, x_k, 3t - A)$ fails because it can map a NO-instance of SUBSET-SUM to a YES-instance of $1/3\text{-PARTITION}$ (take for instance $\langle(2, 2, 2), 3\rangle$).