CS181: Homework 5 solutions

March 9, 2019

1. (20 points). Prove that the language 
\[ \text{COMPL}_{\text{TM}} = \{ \langle M_1, M_2 \rangle \mid L(M_1) = \overline{L(M_2)} \}, \text{where } M_1 \text{ and } M_2 \text{ are Turing Machines} \]
is not Turing-recognizable.

**Solution.** Suppose the language is recognizable. Let \( R \) be the recognizer for the language. Then we construct a paradoxical machine \( M \) as follows.

\( M(y) \):
- Let \( z = \langle M \rangle \), as obtained from the recursion theorem.
- Construct a TM \( N \) such that it accepts everything.
- Run \( R(z, \langle N \rangle) \).
- If \( R \) accepts then output \text{accept}.
- If \( R \) rejects then output \text{reject}.

Consider the following cases:

**Case.** \( M(y) \) accepts for all \( y \in \Sigma^* \): This happens only if \( R(z, \langle N \rangle) \) accepts. Further, \( R \) accepts \( \langle z, \langle N \rangle \rangle \) only if \( L(M) = \emptyset \) since we already know that \( L(N) = \Sigma^* \). But if \( L(M) = \emptyset \) then \( M(y) \) should reject, a contradiction.

**Case.** \( M(y) \) rejects or loops for all \( y \in \Sigma^* \): This happens only if \( R(z, \langle N \rangle) \) rejects or loops. This is only possible if \( L(M) \neq \emptyset \) since we already know that \( L(N) = \Sigma^* \). But if \( L(M) \neq \emptyset \) then \( M(y) \) should accept, a contradiction.

Since we arrive at a contradiction in both the cases, we have that is unrecognizable.

2. (30 points). Define \( \text{SUBSET}_{\text{TM}} \) to be the problem of testing whether the set of strings accepted by a turing machine, say \( M_1 \), is also accepted by another turing machine, say \( M_2 \). More formally,

\[ \text{SUBSET}_{\text{TM}} = \{ (M_1, M_2) \mid L(M_1) \subseteq L(M_2) \} \]
Show that \( \text{SUBSET}_{TM} \) is undecidable.

**Solution:** Suppose that \( \text{SUBSET}_{TM} \) is a decidable language. Denote \( D \) to be the Decider for \( \text{SUBSET}_{TM} \). We use the recursion theorem to construct a paradoxical machine \( N \) as follows.

\[
N(y):
\]

Let \( x = \langle N \rangle \) \(/\!\!/ \text{by recursion theorem}\)

Construct a machine \( M \) that accepts every string in \( \Sigma^* \).

Let \( z = \langle M \rangle \)

Run \( D(z,x) \)

If \( D \) accepts, then reject \( y \).

If \( D \) rejects, then accept \( y \).

Consider the following cases:

**Case.** \( N(y) \) accepts for all \( y \in \Sigma^* \): That is, \( L(N) = \Sigma^* \). This happens only if \( D(z,x) \) rejects for all \( y \in \Sigma^* \). This means that \( (M) \not\subseteq L(N) \) (Since \( z = \langle M \rangle \)). But we know that \( L(M) = \Sigma^* \) from the definition of \( M \) which leads to a contradiction because \( \Sigma^* \not\subseteq \Sigma^* \).

**Case.** \( N(y) \) rejects for all \( y \in \Sigma^* \): That is, \( L(N) = \phi \). This happens only if \( D(z,x) \) accepts for all \( y \in \Sigma^* \). This means that \( (M) \subseteq L(N) \) (Since \( z = \langle M \rangle \)). But we know that \( L(M) = \Sigma^* \) from the definition of \( M \) which leads to a contradiction because \( \Sigma^* \not\subseteq \phi \).

From both the cases, we arrive at the conclusion that \( N \) cannot exist which contradicts the existence of \( D \). Thus, \( \text{SUBSET}_{TM} \) is not a decidable language.

3. **(60 points)** Here we define a new notion of a “\( \mathbb{a} \)” language. A language \( L \) over the alphabet \( \{0, 1\} \) is called “\( \mathbb{a} \)” if there exists a turing machine \( M \) satisfying the following conditions:

- For all \( x \in L \), there exists a \( y \in \{0, 1\}^* \) such that \( M(x, y) \) accepts.
- For all \( x \not\in L \), for all \( y \), \( M(x, y) \) rejects.
  (We think of \( y \) as being the “certificate” that allows \( x \) to be in the language.)

(a) **(15 points)** Let \( Halt_\epsilon \) denote the language of all the turing machines which halt on input \( \epsilon \). More formally,

\[
Halt_\epsilon = \{ \langle N \rangle \mid N \text{ halts on } \epsilon \},
\]

where \( \langle N \rangle \) denotes the code of the machine \( N \).

Show that \( Halt_\epsilon \) is a language.

**Solution.** A certifier for the language \( Halt_\epsilon \) is described as follows.

\[
M(\langle N \rangle, t): \quad \text{Execute } N \text{ on } \epsilon \text{ for at most } t \text{ number of steps. If it halts within } t \text{ number of steps then output } \text{Accept. If } N \text{ has not halted, then output } \text{Reject.}
\]
If \( \langle N \rangle \in \text{Halt} \), then there exists \( t^* \geq 0 \) such that \( N \) on input \( \epsilon \) halts within \( t^* \) number of steps. Hence, \( M(\langle N \rangle, t^*) \) outputs accept. If \( \langle N \rangle \notin \text{Halt} \) then there does not exist any \( t^* \geq 0 \) such that \( N \) on input \( \epsilon \) halts within \( t^* \) number of steps. Thus, \( M(\langle N \rangle, t^*) \) outputs reject for all \( t^* \geq 0 \).

(b) (15 points). Explain why your method does not work if you try to prove that the language \( \text{Halt}_{all} = \{ \langle N \rangle \mid N \text{ halts on all inputs} \} \) is a language.

Solution. If we directly use the certifier above for \( \text{Halt}_{all} \) then we run into problems because our certificate should now carry enough information (which in the above case is the time taken by the machine) that helps the certifier check whether the machine halts on all inputs, and not just one input (as in the above case). And since the number of such possible inputs are infinite, a finite certificate can no longer contain all the information. Hence, the above idea does not immediately work for \( \text{Halt}_{all} \).

(c) (30 points). Use the recursion theorem to prove that the language \( \text{Halt}_{all} \), is not a language.

Solution. Suppose that \( \text{Halt}_{all} \) is a certified language. Denote \( M \) to be the certifier of \( \text{Halt}_{all} \). We use the recursion theorem to construct the paradoxical machine \( N \) as follows.

\[
N(x):
\]
Let \( z = \langle N \rangle \) // by recursion theorem
Run \( M(\langle N \rangle, x) \)
If \( M \) accepts then loop.
If \( M \) rejects then accept.

Consider the following cases:

Case. \( N(x) \) accepts for all \( x \in \Sigma^* \): This happens only if \( M(\langle N \rangle, x) \) rejects for all \( x \in \Sigma^* \). This means that \( \langle N \rangle \notin \text{Halt}_{all} \). This further implies that there must exist some input \( x^* \) such that \( N \) does not halt, and so does not accept, on input \( x^* \). This leads to a contradiction.

Case. \( N(x) \) loops for some \( x \in \Sigma^* \): This happens only if \( M(\langle N \rangle, x) \) accepts. This is again possible only if \( \langle N \rangle \in \text{Halt}_{all} \). But this means that \( \langle N \rangle \) halts on every input in \( \Sigma^* \). This leads to a contradiction.

From both the cases, we arrive at the conclusion that \( N \) cannot exist which contradicts the existence of \( M \). Thus, \( \text{Halt}_{all} \) is not a language.