CS 341: Foundations of Computer Science II
Prof. Marvin Nakayama

Homework 9 Solutions

1. Let $\mathcal{B}$ be the set of all infinite sequences over $\{0, 1\}$. Show that $\mathcal{B}$ is uncountable, using a proof by diagonalization.

   Answer: Each element in $\mathcal{B}$ is an infinite sequence $(b_1, b_2, b_3, \ldots)$, where each $b_i \in \{0, 1\}$. Suppose $\mathcal{B}$ is countable. Then we can define a correspondence f between $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\mathcal{B}$. Specifically, for $n \in \mathcal{N}$, let $f(n) = (b_{n1}, b_{n2}, b_{n3}, \ldots)$, where b_{ni} is the ith bit in the nth sequence, i.e.,

n	$f(n)$
1	$(b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, \ldots)$
2	$(b_{21}, b_{22}, b_{23}, b_{24}, b_{25}, \ldots)$
3	$(b_{31}, b_{32}, b_{33}, b_{34}, b_{35}, \ldots)$
4	$(b_{41}, b_{42}, b_{43}, b_{44}, b_{45}, \ldots)$
$\vdots$	$\vdots$

   Now define the infinite sequence $c = (c_1, c_2, c_3, c_4, c_5, \ldots) \in \mathcal{B}$, where $c_i = 1 - b_{ii}$ for each $i \in \mathcal{N}$. In other words, the ith bit in c is the opposite of the ith bit in the ith sequence. For example, if

n	$f(n)$
1	$(0, 1, 1, 0, 0, \ldots)$
2	$(1, 0, 1, 0, 1, \ldots)$
3	$(1, 1, 1, 1, 1, \ldots)$
4	$(1, 0, 0, 1, 0, \ldots)$
$\vdots$	$\vdots$

   then we would define $c = (1, 1, 0, 0, \ldots)$. Thus, for each $n = 1, 2, 3, \ldots$, note that $c \in \mathcal{B}$ differs from the nth sequence in the nth bit, so c does not equal $f(n)$ for any n, which is a contradiction. Hence, $\mathcal{B}$ is uncountable.

2. Recall that $EQ_{\text{CFG}} = \{ \langle G_1, G_2 \rangle \mid G_1 \text{ and } G_2 \text{ are CFGs and } L(G_1) = L(G_2) \}$. Show that EQ_{CFG} is undecidable. For this problem, you may assume that ALL_{CFG} is undecidable, as established in Theorem 5.13.

   Answer: We will reduce ALL_{CFG} to EQ_{CFG}, where

   $$ALL_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \}.$$

Sipser (Theorem 5.13) shows that ALL_{CFG} is undecidable.

Define CFG $G_0 = (V, \Sigma, R, S)$, where $V = \{S\}$ and S is the starting variable. For each terminal $\ell \in \Sigma$, the CFG G_0 has a rule $S \to \ell S$ in R. Also, G_0 includes the rule $S \to \varepsilon$. For example, if $\Sigma = \{a, b\}$, then the rules in G_0 are $S \to aS \mid bS \mid \varepsilon$. It is easy to see that $L(G_0) = \Sigma^*$.

Let R be a TM that decides EQ_{CFG} and construct TM S to decide ALL_{CFG}. Then S works in the following manner.

S = "On input $\langle G \rangle$, where G is a CFG:

1. Run R on input $\langle G, G_0 \rangle$, where G_0 is the CFG defined above with $L(G_0) = \Sigma^*$.
2. If R accepts, *accept*. If R rejects, *reject*."

In stage 1, TM R determines if $L(G) = L(G_0)$, but because $L(G_0) = \Sigma^*$, this determines if $L(G) = \Sigma^*$. In other words, TM S decides ALL_{CFG}, but because ALL_{CFG} is undecidable, this is a contradiction. Hence, we must have that EQ_{CFG} is also undecidable.

3. Show that EQ_{CFG} is co-Turing-recognizable.

Answer: Recall that EQ_{CFG} is a co-Turing-recognizable language if and only if its complement $\overline{EQ_{\text{CFG}}}$ is a Turing-recognizable language. Now,

$$\overline{EQ_{\text{CFG}}} = C \cup D,$$

where

$$\begin{aligned} C &= \{\, w \mid w \text{ does not have the form } \langle G_1, G_2 \rangle \text{ for some CFGs } G_1 \text{ and } G_2 \}, \\ D &= \{\, \langle G_1, G_2 \rangle \mid G_1 \text{ and } G_2 \text{ are CFGs and } L(G_1) \neq L(G_2) \,\}. \end{aligned}$$

We claim that the set C (consisting of strings that violate the syntax for encoding $\langle G_1, G_2 \rangle$) is easy to recognize. We do not provide a formal proof though. The set D can be recognized as follows. We convert CFGs G_1 and G_2 into equivalent CFGs in Chomsky normal form. Then we start enumerating strings in Σ^* in string order $s_1, s_2, s_3, \ldots$, where Σ is the set of terminals for both G_1 and G_2. For each enumerated string s_i, we check whether it can be generated by G_1 and by G_2. If both CFGs or neither CFG can generate s_i, then TM moves on to consider the next string in string order. Otherwise, exactly one of the CFGs generates the string and the other CFG does not, so the CFGs are not equivalent, and the TM accepts. Thus, D is Turing-recognizable. We showed in a previous homework that the class of Turing-recognizable languages is closed under union, so $\overline{EQ_{\text{CFG}}}$ is Turing-recognizable.

Here are the details of a TM T that recognizes $\overline{EQ_{\text{CFG}}}$, where $s_1, s_2, s_3, \ldots$ is an enumeration of strings in Σ^* in string order:

T = "On input $\langle G_1, G_2 \rangle$, where G_1 and G_2 are CFGs:

0. Check if G_1 and G_2 are valid CFGs. If at least one isn't, *accept*.

1. Convert G_1 and G_2 each into equivalent CFGs G_1' and G_2', both in Chomsky normal form.

2. Repeat the following for $i = 1, 2, 3, \ldots$

3. Test if both G_1' and G_2' generate s_i.
If exactly one of them does and the other doesn't, *accept*."

Why did we convert the CFGs into Chomsky normal form? The reason is that there is a procedure that always halts for checking whether a CFG in Chomsky normal form can generate a particular string w or not; e.g., see the proof of Theorem 4.7, which shows A_{CFG} is decidable.

4. Let $S_{\mathrm{TM}} = \{ \langle M \rangle \mid M \text{ is a TM that accepts } w^{\mathcal{R}} \text{ whenever it accepts } w \}$. Show that S_{TM} is undecidable.

Answer: The basic idea is to reduce A_{TM} to S_{TM}, where

$$A_{\mathrm{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \},$$

which we know is undecidable by Theorem 4.11. To get a contradiction, let us assume that S_{TM} is decidable, and let S be a decider for S_{TM}. To show that A_{TM} reduces to S_{TM}, we will now use the decider S as a subroutine to build a TM A that decides A_{TM}, as follows:

A = "On input $\langle M, w \rangle$, where M is a TM and w is a string:

0. Check if $\langle M, w \rangle$ is a valid encoding of a TM M and string w.
If not, *reject*.

1. Construct the following TM M_2 from M and w:

M_2 = "On input x:

1. If $x \in L(\mathtt{00}^*\mathtt{11}^*)$, *accept*.

2. If $x \notin L(\mathtt{00}^*\mathtt{11}^*)$, then run M on input w.
If M accepts w, *accept*; else, *reject*."

2. Run S on input $\langle M_2 \rangle$.

3. If S accepts, *accept*; if S rejects, *reject*."

Before showing that TM A decides A_{TM}, first consider the language $L(\mathtt{00}^*\mathtt{11}^*)$ that appears in the constructed TM M_2. The language $L(\mathtt{00}^*\mathtt{11}^*)$ does **not** have the property that if $y \in L(\mathtt{00}^*\mathtt{11}^*)$, then $y^{\mathcal{R}} \in L(\mathtt{00}^*\mathtt{11}^*)$; e.g., $\mathtt{001} \in L(\mathtt{00}^*\mathtt{11}^*)$, but its reverse $(\mathtt{001})^{\mathcal{R}} = \mathtt{100} \notin L(\mathtt{00}^*\mathtt{11}^*)$. So if we have a TM T with language $L(\mathtt{00}^*\mathtt{11}^*)$, then $\langle T \rangle \notin S_{\mathrm{TM}}$.

On the other hand, consider the language $L((\mathtt{0} \cup \mathtt{1})^*)$, which consists of all strings of $\mathtt{0}$s and $\mathtt{1}$s. This language does the property that if $y \in L((\mathtt{0} \cup \mathtt{1})^*)$, then $y^{\mathcal{R}} \in L((\mathtt{0} \cup \mathtt{1})^*)$ because $L((\mathtt{0} \cup \mathtt{1})^*)$ contains all strings over $\{\mathtt{0}, \mathtt{1}\}$. So if we have a TM T' with language $L((\mathtt{0} \cup \mathtt{1})^*)$, then $\langle T' \rangle \in S_{\mathrm{TM}}$.

Now let's figure out the language of the TM M_2. In stage 1 of TM M_2, it automatically accepts any string $x \in L(\mathtt{0}\mathtt{0}^*\mathtt{1}\mathtt{1}^*)$. For any string $x \notin L(\mathtt{0}\mathtt{0}^*\mathtt{1}\mathtt{1}^*)$, TM M_2 accepts x if and only if M accepts w. Thus, the language $L(M_2)$ of M_2 has two possibilities:

- If M accepts w, then $L(M_2)$ is $L((\mathtt{0} \cup \mathtt{1})^*)$, so $\langle M_2 \rangle \in S_{\mathrm{TM}}$.
- If M does not accept w, then $L(M_2)$ is $L(\mathtt{0}\mathtt{0}^*\mathtt{1}\mathtt{1}^*)$, so $\langle M_2 \rangle \notin S_{\mathrm{TM}}$.

Hence, $\langle M_2 \rangle$ belongs to S_{TM} if and only if M accepts w, so a solution for S_{TM} can be used to solve A_{TM}; i.e., A_{TM} reduces to S_{TM}. Because S is assumed to decide S_{TM}, the TM A decides A_{TM} because stage 3 of the TM A accepts $\langle M, w \rangle$ if and only if S accepts $\langle M_2 \rangle$. But we know that A_{TM} is undecidable, so S_{TM} must also be undecidable.