HW6CSE105F24: Homework assignment 6 Sample solution

CSE105F24

Due: December 3, 2024 at 5pm, via Gradescope

In this assignment,

You will practice analyzing, designing, and working with Turing machines. You will use general constructions and specific machines to explore the classes of recognizable and decidable languages. You will explore various ways to encode machines as strings so that computational problems can be recognized.

Resources: To review the topics for this assignment, see the class material from Weeks 8 and 9. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Sections 4.2, 5.3, 5.1. Chapter 4 exercises 4.9, 4.12. Chapter 5 exercises 5.4, 5.5, 5.6, 5.7. Chapter 5 problems 5.22, 5.23, 5.24, 5.28

Assigned questions

1. **What's wrong with these reductions? (if anything)** (15 points): Suppose your friends are practicing coming up with mapping reductions $A \leq_m B$ and their witnessing functions $f : \Sigma^* \to \Sigma^*$. For each of the following attempts, determine if it has error(s) or is correct. Do so by labelling each attempt with all and only the labels below that apply, and justifying this labelling.

- *Error Type 1:* The given function can't witness the claimed mapping reduction because there exists an $x \in A$ such that $f(x) \notin B$.
- *Error Type 2:* The given function can't witness the claimed mapping reduction because there exists an $x \notin A$ such that $f(x) \in B$.
- *Error Type 3:* The given function can't witness the claimed mapping reduction because the specified function is not computable.
- *Correct:* The claimed mapping reduction is true and is witnessed by the given function.

Clearly present your answer by providing a brief (3-4 sentences or so) justification for whether **each** of these labels applies to each example.

(a) (*Graded for completeness*) [1] $A_{\text{TM}} \leq_m HALT_{\text{TM}}$ and

$$f(x) = \begin{cases} \langle \text{start} \rightarrow (q_{\text{acc}}), \varepsilon \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ & \quad \text{and } w \in L(M) \\ \langle \text{start} \rightarrow (q_0 \xrightarrow{0,1,\sqcup \rightarrow R} q_0), (q_{acc}) \rangle & \text{otherwise} \end{cases}$$

Solution: The attempt has Error Type 3.

i. Error Type 1 is not present. If $x \in A_{TM}$, then we have $x = \langle M, w \rangle$ for a Turing machine M and string w and $w \in L(M)$, which falls into the first case of the function definition. In this case, the output of the function $f(x) = \langle \text{start} \rightarrow (q_{\text{acc}}), \varepsilon \rangle \in HALT_{TM}$, since the Turing machine encoded in the output immediately accepts and halts on ε.

ii. Error Type 2 is not present. Each element in $HALT_{TM}$ should be in the form $\langle M, w \rangle$ where M is a Turing machine and w is a string. If $x \notin A_{TM}$, it will fall into the second case of the function definition which has output $f(x) \notin HALT_{TM}$, since in this case $f(x)$ is a string encoding of a Turing machine that does not match the type of elements in $HALT_{TM}$.

iii. Error Type 3 is present because we cannot have an algorithm that decides if some TM M accepts w in the if statement. To put it another way, A_{TM} is undecidable, so we cannot know the answer of whether M accepts w reliably in finite time using a Turing machine.

(b) (*Graded for completeness*) $A_{\text{TM}} \leq_m EQ_{\text{TM}}$ with

$$f(x) = \begin{cases} \langle \text{start} \rightarrow (q_{\text{acc}}), M_w \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ \langle \text{start} \rightarrow (q_{\text{acc}}), \text{start} \rightarrow (q_{\text{rej}}), (q_{\text{acc}}) \rangle & \text{otherwise.} \end{cases}$$

[1] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we expect you to include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

Where for each Turing machine M, we define

$$M_w = \text{“On input } y$$
1. Simulate M on w.
2. If it accepts, accept.
3. If it rejects, reject.”

Solution: Correct. The claimed mapping reduction is true and is witnessed by the given function.

i. Error Type 1 is not present. If $x \in A_{TM}$, then we have $x = \langle M, w\rangle$ for a Turing machine M and string w and $w \in L(M)$, which falls into the first case of the function definition. In this case, let's denote the output of the function $f(x) = \langle M_0, M_w\rangle$ where M_0 accepts all strings, and M_w also accepts all strings because M accepts w. Therefore we get $L(M_1) = L(M_w)$ and $f(x) \in EQ_{TM}$.

ii. Error Type 2 is not present. If $x \notin A_{TM}$, there are two cases:

- The first case is when x is not a string encoding $\langle M, w\rangle$ of a Turing machine M and string w, which falls into the second case of the function definition. In this case, let's denote the output $f(x) = \langle M_1, M_2\rangle$. Since M_1 accepts all strings and M_2 rejects all strings, $L(M_1) \neq L(M_2)$, so $f(x) \notin EQ_{TM}$.
- The second case is when $x = \langle M, w\rangle$ for a Turing machine M and string w and $w \notin L(M)$, which falls into the first case of the definition. In this case, let's denote the output of the function $f(x) = \langle M_0, M_w\rangle$ where M_0 accepts all strings. If M rejects w, then M_w rejects all strings; if M loops on w, then M_w loops on all strings. So M_w does not accept any string, i.e. $L(M_0) \neq L(M_w)$, thus $f(x) \notin EQ_{TM}$.

iii. Error Type 3 is not present because type checking and constructing M_w both take finite time, so we can build a Turing machine that computes this function.

(c) (*Graded for correctness*) [2] $HALT_{\text{TM}} \leq_m EQ_{\text{TM}}$ with

$$f(x) = \begin{cases} \langle \text{start} \to (q_{\text{acc}}), M_w\rangle & \text{if } x = \langle M, w\rangle \text{ for a Turing machine } M \text{ and string } w \\ \langle \text{start} \to (q_{\text{acc}}),\ \text{start} \to (q_{\text{rej}}) \quad (q_{\text{acc}})\rangle & \text{otherwise.} \end{cases}$$

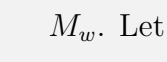
[2] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

Where for each Turing machine M, we define

$M_w =$ "On input y

1. If y is not the empty string, reject.
2. Else, simulate M on w.
3. If it accepts, accept.
4. If it rejects, reject."

Solution: This mapping reduction has an Error Type 1.

i. Here, we have an Error Type 1, as witnessed by $x = \langle M', w' \rangle$ where M' halts on w'. The input x has a correct format that falls into case 1, so we must construct M_w. Let's denote $A =$ 

so $f(x) = \langle A, M_w \rangle$.

The language of A is the set of all strings since a Turing machine does not have to process the entire string in order to accept that string. Upon starting computation, all strings are immediately accepted without reading the tape because the start state is the accept state.

That being said, irregardless of whether M' accepts or rejects w', $L(M_w)$ will never be equal to A. The new Turing machine we construct automatically rejects all strings except for the empty string, or strings that are in $L(A)$, thus $\langle A, M_w \rangle$ won't be in EQ_{TM} even if M' halts on w', violating membership properties.

ii. There is no Error Type 2. If $x \notin HALT_{TM}$, the first case is x doesn't type check, then we will output the encoding of a pair of Turing machines whose languages are not equivalent, i.e. the output is not in EQ_{TM}. If we have the correct input type $\langle M, w \rangle$ but M does not halt on w, then M_w will loop forever on ε and reject all other strings. In this case, $L(M_w) = \emptyset$, which is not equal to $L(A) = \Sigma^*$, so the output is also not in EQ_{TM}.

iii. Error Type 3 is not present because type checking and constructing M_w both take finite time, so we can build a Turing machine that computes this function.

(d) (*Graded for correctness*) $\{ww \mid w \in \{0,1\}^*\} \leq_m \emptyset$ and $f(x) = 1$ for each $x \in \{0,1\}^*$.

Solution: This mapping reduction has an Error Type 1.

i. We can notice that there is an Error Type 1 as witnessed by $x = 11$. We know that $x \in \{ww \mid w \in \{0,1\}^*\}$ and $x \in \{0,1\}^*$, so $f(x) = 1$. However, by definition, $1 \notin \emptyset$ because $\emptyset$ contains no strings.

ii. Error Type 2 does not exist, as for all strings $x \notin \{ww \mid w \in \{0,1\}^*\}$ we compute $f(x) = 1$ because $x \in \{0,1\}^*$. We know $1 \notin \emptyset$ as the empty set contains no strings.

iii. Error Type 3 is not present because computing $f(x)$ takes finite time, so we can build a Turing machine that computes this function.

(e) (*Graded for correctness*) $\emptyset \leq_m \{ww \mid w \in \{0,1\}^*\}$ and $f(x) = 1$ for each $x \in \{0,1\}^*$.

Solution: Correct. The claimed mapping reduction is true and is witnessed by the given function.

i. There is no Error Type 1 since this condition is vacuously true for the empty set, i.e. there are no strings in the empty set that need to be mapped to a string in $\{ww \mid w \in \{0,1\}^*\}$ and thus this condition is fulfilled.

ii. On the other hand, all strings not in the empty set, or equivalently, the set of all strings, must all be mapped to a string not in $\{ww \mid w \in \{0,1\}^*\}$. And we can see that this is true! For any string not in the empty set, or for any $x \in \{0,1\}^*$, $f(x) = 1$. Clearly, $1 \notin \{ww \mid w \in \{0,1\}^*\}$ so no Error Type 2 exists.

iii. Error Type 3 is not present because computing $f(x)$ takes finite time, so we can build a Turing machine that computes this function.

2. **Using mapping reductions** (14 points): Consider the following computational problems we've discussed

$$\begin{aligned} A_{TM} &= \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ accepts } w\} \\ HALT_{TM} &= \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ halts on } w\} \\ E_{TM} &= \{\langle M\rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\} \\ EQ_{TM} &= \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are both Turing machines and } L(M_1) = L(M_2)\} \end{aligned}$$

and the new computational problem

$$Two_{TM} = \{\langle M\rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly one string}\}$$

NOTE: the intended definition of Two_{TM} is that M accepts exactly two strings.

(a) (*Graded for correctness*) Give an example of a string that is an element of Two_{TM} and a string that is not an element of Two_{TM} and briefly justify your choices.

Solution: Let's consider alphabet $\Sigma = \{0,1\}$.

If using $Two_{TM} = \{\langle M\rangle \mid M$ is a Turing machine and M accepts exactly **one** string$\}$:

Define the two Turing machines below in high level description.

M: "On input x,

i. If $x = \varepsilon$, accept. Otherwise, reject."

M': "On input x,

i. Reject."

Notice that M only accepts if $x = \varepsilon$, otherwise it rejects. M accepts exactly one string. Additionally, notice that M' always rejects. M' does not accept exactly one string.

Consider the string encoding of M, $\langle M \rangle$. M is a Turing machine and M accepts exactly one string, as above, so $\langle M \rangle \in Two_{TM}$. On the other hand, consider the string encoding of M', $\langle M' \rangle$. M' is a Turing machine, and M' does not accept exactly one string, so $\langle M' \rangle \notin Two_{TM}$.

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{two} \text{ strings}\}$:

Define the two Turing machines below in high level description.

M: "On input x,

i. If $x = \varepsilon$ or $x = 0$, accept. Otherwise, reject."

M': "On input x,

i. Reject."

Similar to the above reasoning, since M only accepts ε and 0 and rejects all other strings, we have $\langle M \rangle \in Two_{TM}$. Since M' rejects all strings, we have $\langle M' \rangle \notin Two_{TM}$.

(b) (*Graded for completeness*) Prove that Two_{TM} is not decidable by showing that $A_{TM} \leq_m Two_{TM}$.

Solution:

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{one} \text{ strings}\}$:

To show that $A_{TM} \leq_m Two_{TM}$, we have to give a computable function that satisfies the "maintain membership property". Consider the computable function defined by the Turing machine below. Let $const_{out}$ be some constant not in Two_{TM}.

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{out}$.

1. $x = \langle M, w \rangle$. Construct M': "On input y,

   i. If $y \neq w$, reject.

   ii. Run M on w.

   iii. If M accepts, accept. If M rejects, reject."

2. Output $\langle M' \rangle$"

Let's verify that this Turing machine does compute the function that witnesses the mapping reduction $A_{TM} \leq_m Two_{TM}$. Consider the following cases:

Case 1: $x \neq \langle M, w \rangle$. Then, $x \notin A_{TM}$, since it is not of the correct type. It fails the type check, and $F(x) = const_{out} \notin Two_{TM}$.

Case 2: $x = \langle M, w \rangle$, and M rejects w. Then, $x \notin A_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M rejects w, M' also rejects w, so it rejects every string. So M' does not accept exactly one string. Thus, $F(x) = \langle M' \rangle \notin Two_{TM}$

Case 3: $x = \langle M, w \rangle$, and M loops on w. Then, $x \notin A_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M loops on w, M' also loops on w, so it rejects almost all strings, and loops on the remaining one. So M' does not accept exactly one string. Thus, $F(x) = \langle M' \rangle \notin Two_{TM}$

Case 4: $x = \langle M, w \rangle$, and M accepts w. Then, $x \in A_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M accepts w, M' also accepts w, so it rejects almost all strings, and accepts the remaining one. So M' accept exactly one string. Thus, $F(x) = \langle M' \rangle \in Two_{TM}$

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{two} \text{ strings}\}$:

Consider the computable function defined by the Turing machine below that witness the mapping reduction. Let $const_{out}$ be some constant not in Two_{TM}.

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{out}$.

1. $x = \langle M, w \rangle$. Construct M': "On input y,

i. If $y \neq w$ and $y \neq w0$, reject.

ii. Run M on w.

iii. If M accepts, accept. If M rejects, reject."

2. Output $\langle M' \rangle$"

The proof of correctness process is similar as above.

(c) (*Graded for correctness*) Give a different proof that Two_{TM} is not decidable by showing that $HALT_{TM} \leq_m Two_{TM}$.

Solution:

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{one} \text{ strings}\}$:

The case for $HALT_{TM}$ is very similar to A_{TM}. To show that $HALT_{TM} \leq_m Two_{TM}$, we have to give a computable function that satisfies the "maintain membership property". Consider the computable function defined by the Turing machine below. Let $const_{out}$ be some constant not in Two_{TM}.

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{out}$.

1. $x = \langle M, w \rangle$. Construct M': "On input y,

   i. If $y \neq w$, reject.

   ii. Run M on w.

   iii. If M accepts, accept. If M rejects, accept."

2. Output $\langle M' \rangle$"

Let's verify that this Turing machine does compute the function that witnesses the mapping reduction $HALT_{TM} \leq_m Two_{TM}$. Consider the following cases:

Case 1: $x \neq \langle M, w \rangle$. Then, $x \notin HALT_{TM}$, since it is not of the correct type. It fails the type check, and $F(x) = const_{out} \notin Two_{TM}$.

Case 2: $x = \langle M, w \rangle$, and M loops on w. Then, $x \notin HALT_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M loops on w, M' also loops on w, so it rejects almost all strings, and loops on the remaining one. So M' does not accept exactly one string. Thus, $F(x) = \langle M' \rangle \notin Two_{TM}$

Case 3: $x = \langle M, w \rangle$, and M rejects w. Then, $x \in HALT_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M rejects w, M' accepts w, so it rejects almost all strings, and accepts the remaining one. So M' accept exactly one string. Thus, $F(x) = \langle M' \rangle \in Two_{TM}$

Case 4: $x = \langle M, w \rangle$, and M accepts w. Then, $x \in HALT_{TM}$. Consider the behavior of M'. It rejects all strings not w. If its input is w, it runs M on w. Since M accepts w, M' also accepts w, so it rejects almost all strings, and accepts the remaining one. So M' accept exactly one string. Thus, $F(x) = \langle M' \rangle \in Two_{TM}$

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{two} \text{ strings}\}$:

Consider the computable function defined by the Turing machine below that witness the mapping reduction. Let $const_{out}$ be some constant not in Two_{TM}.

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{out}$.
1. $x = \langle M, w \rangle$. Construct M': "On input y,
   i. If $y \neq w$ and $y \neq w0$, reject.
   ii. Run M on w.
   iii. If M accepts, accept. If M rejects, accept."
2. Output $\langle M' \rangle$"

The proof of correctness process is similar as above.

(d) (*Graded for completeness*) Is Two_{TM} recognizable? Justify your answer.

Solution:

If using $Two_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly } \textbf{one} \text{ strings}\}$:

It is not recognizable. It's possible to show that $E_{TM} \leq_m Two_{TM}$. Consider the following Turing machine that computes the witnessing function. Let $const_{out}$ be some constant not in Two_{TM}

F : "On input x,

0. Type check. If $x \neq \langle M \rangle$, output $const_{out}$.
1. $x = \langle M \rangle$. Construct M': "On input y,
   i. If $y = \varepsilon$, accept.
   ii. If y does not start with 0, reject.
   iii. $y = 0z$. Run M on z.
   iv. If M accepts, accept. If M rejects, reject."
2. Output $\langle M' \rangle$"

Before diving into the case analysis, let's see what the idea of this reduction is. We make M' always accepts at least one string ε, and may accept more depending on x. We may naturally want to build M' such that $L(M') = \{\varepsilon\} \cup L(M)$ so it seems that if M accepts any string at all (when $x = \langle M \rangle \notin E_{TM}$), M' would accept more than one string (so $\langle M' \rangle \notin E_{TM}$). But there is one problem with this approach: we picked ε as the string for M' to always accept, and consider when $L(M) = \{\varepsilon\}$. Then in this case, $\langle M \rangle \notin E_{TM}$. However, $L(M') = \{\varepsilon\} \cup \{\varepsilon\} = \{\varepsilon\} \in Two_{TM}$. No matter what string we pick, we will always run into this problem. One rather clever way to avoid this problem is to establish a map between strings M accepts, and strings we want M' to accept. The map is $z \mapsto 0z$. Notice that ε is not in the image of this map, so we won't be overriding the acceptance of this map, and end up in the problematic case.

More formally, consider three cases for the proof of correctness:

i. When x doesn't type check, $x \notin E_{TM}$. The output $F(x) = const_{out} \notin Two_{TM}$.

ii. When $x = \langle M \rangle$ and M accepts some string, then $x \notin E_{TM}$. In this case, say M accepts string s, then the M' we construct will accept ε and $0s$, which means that M' accepts more than one string. Therefore, $F(x) = \langle M' \rangle \notin Two_{TM}$.

iii. When $x = \langle M \rangle$ and M accepts no string, then $x \in E_{TM}$. In this case, M' only accepts ε. Therefore, $F(x) = \langle M' \rangle \in Two_{TM}$.

If using $Two_{TM} = \{\langle M \rangle \mid M$ is a Turing machine and M accepts exactly **two** strings$\}$:

It is not recognizable. It's possible to show that $E_{TM} \leq_m Two_{TM}$. Consider the following Turing machine that computes the witnessing function. Let $const_{out}$ be some constant not in Two_{TM}

F : "On input x,

0. Type check. If $x \neq \langle M \rangle$, output $const_{out}$.

1. $x = \langle M \rangle$. Construct M': "On input y,

   i. If $y = \varepsilon$ or $y = 1$, accept.

   ii. If y does not start with 0, reject.

   iii. $y = 0z$. Run M on z.

   iv. If M accepts, accept. If M rejects, reject."

2. Output $\langle M' \rangle$"

The proof of correctness process is similar as above.

3. **Using mapping reductions** (14 points): Consider the following computational problems we've discussed

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ accepts } w\}$$
$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ halts on } w\}$$
$$E_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\}$$
$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are both Turing machines and } L(M_1) = L(M_2)\}$$

and the new computational problem

$Evens_{TM} = \{\langle M \rangle \mid M$ is a Turing machine and any string that M accepts must be even length (but there may even length strings that M doesn't accept)$\}$

(a) (*Graded for correctness*) Give an example of a string that is an element of $Evens_{TM}$ and a string that is not an element of $Evens_{TM}$ and briefly justify your choices.

Solution: Let's consider alphabet $\Sigma = \{0, 1\}$. Define the two Turing machines below in high level description.

M_1: "On input x,

i. If $x = \varepsilon$, accept. Otherwise, reject."

M_2: "On input x,

i. If $x = 1$, accept. Otherwise, reject."

Note that M_1 only accepts string ε which has even length, so $\langle M_1 \rangle \in Evens_{TM}$. M_2 accepts string 1 which has odd length, so $\langle M_2 \rangle \notin Evens_{TM}$.

(b) (*Graded for completeness*) Prove that $Evens_{TM}$ is not decidable by showing that $\overline{HALT_{TM}} \leq_m Evens_{TM}$. *This was changed Nov 26.*

Solution: To show that $\overline{HALT_{TM}} \leq_m Evens_{TM}$, we have to give a computable function that satisfies the "maintain membership property". Consider the computable function defined by the Turing machine below. Let $const_{in}$ be some constant in $Evens_{TM}$ (for example, the $\langle M_1 \rangle$ given in part (a)).

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{in}$.
1. $x = \langle M, w \rangle$. Construct M': "On input y,
   i. If $|y|$ is even, reject.
   ii. Run M on w.
   iii. If M accepts, accept. If M rejects, accept."
2. Output $\langle M' \rangle$"

Let's verify that this Turing machine does compute the function that witnesses the mapping reduction $\overline{HALT_{TM}} \leq_m Evens_{TM}$. Consider the following cases:

Case 1: $x \neq \langle M, w \rangle$. Then, $x \in \overline{HALT_{TM}}$, since it is not of the correct type. It fails the type check, and $F(x) = const_{in} \in Evens_{TM}$.

Case 2: $x = \langle M, w \rangle$, and M loops on w. Then, $x \in \overline{HALT_{TM}}$. Consider the behavior of M'. It rejects all strings not of even length. Then, it runs M on w. Since M loops on w, M' also loops on w, so it rejects all even strings, and loops on odd ones. So M' does not accept any string. Thus, $F(x) = \langle M' \rangle \in Evens_{TM}$ since no odd strings are accepted by M'.

Case 3: $x = \langle M, w \rangle$, and M rejects w. Then, $x \notin \overline{HALT_{TM}}$. Consider the behavior of M'. It rejects all strings with even length. If its input is of odd length, it runs M on w. Since M rejects w, M' accepts y, so it rejects even strings, and accepts odd ones. So M' accepts exactly only odd strings. Thus, $F(x) = \langle M' \rangle \notin Evens_{TM}$

Case 4: $x = \langle M, w \rangle$, and M accepts w. Then, $x \notin \overline{HALT_{TM}}$. Consider the behavior of M'. It rejects all even strings. If its input is not even, it runs M on w. Since M accepts w, M' also accepts y, so it rejects all even strings, and accepts the only odd strings. Thus, $F(x) = \langle M' \rangle \notin Evens_{TM}$

(c) (*Graded for correctness*) Give a different proof that $Evens_{TM}$ is not decidable by showing that $\overline{A_{TM}} \leq_m Evens_{TM}$. *This was changed Nov 26.*

Solution: This is a very similar proof to the previous section, with one small tweak. Let $const_{in}$ be some constant in $Evens_{TM}$ (for example, the $\langle M_1 \rangle$ given in part (a)).

F: "On input x,

0. Type check. If $x \neq \langle M, w \rangle$ for some Turing machine M and string w, output $const_{in}$.
1. $x = \langle M, w \rangle$. Construct M': "On input y,
   i. If $|y|$ is even, reject.
   ii. Run M on w.
   iii. If M accepts, accept. If M rejects, reject."
2. Output $\langle M' \rangle$"

Let's verify that this Turing machine does compute the function that witnesses the mapping reduction $\overline{A_{TM}} \leq_m Evens_{TM}$. Consider the following cases:

Case 1: $x \neq \langle M, w \rangle$. Then, $x \in \overline{A_{TM}}$, since it is not of the correct type. It fails the type check, and $F(x) = const_{in} \in Evens_{TM}$.

Case 2: $x = \langle M, w \rangle$, and M loops on w. Then, $x \in \overline{A_{TM}}$. Consider the behavior of M'. It rejects all strings not of even length. Then, it runs M on w. Since M loops on w, M' also loops on w, so it rejects all even strings, and loops on odd ones. So M' does not accept any string. Thus, $F(x) = \langle M' \rangle \in Evens_{TM}$ since no odd strings are accepted by M'.

Case 3: $x = \langle M, w \rangle$, and M rejects w. Then, $x \in \overline{A_{TM}}$. Consider the behavior of M'. It rejects all strings with even length. If its input is of odd length, it runs M on w. Since M rejects w, M' rejects y, so it rejects all strings. So M' does not accept any string. Thus, $F(x) = \langle M' \rangle \in Evens_{TM}$ since no odd strings are accepted by M'.

Case 4: $x = \langle M, w \rangle$, and M accepts w. Then, $x \notin \overline{A_{TM}}$. Consider the behavior of M'. It rejects all even strings. If its input is not even, it runs M on w. Since M accepts w, M' also accepts y, so it rejects all even strings, and accepts the only odd strings. So M' accepts odd strings. Thus, $F(x) = \langle M' \rangle \notin Two_{TM}$

(d) (*Graded for completeness*) Is $Evens_{TM}$ recognizable? Justify your answer.

Solution: Note that in part (c) we proved that $\overline{A_{TM}} \leq_m Evens_{TM}$. Since $\overline{A_{TM}}$ is unrecognizable, we know that $Evens_{TM}$ is also unrecognizable (by the corollary in Week 9, pg. 3).

Alternative approach: Let's try a similar strategy to finding whether Two_{TM} was unrecognizable by showing the mapping reduction $E_{TM} \leq_m Evens_{TM}$. Consider the following Turing machine that computes the witnessing function. Let $const_{out}$ be some constant not in $Evens_{TM}$

F : "On input x,

0. Type check. If $x \neq \langle M \rangle$, output $const_{out}$.
1. $x = \langle M \rangle$. Construct M': "On input y,
   i. If y has an even length, accept.
   ii. $y = uv$ where $|u| = |v| + 1$. Run M on v.
   iii. If M accepts, accept. If M rejects, reject."
2. Output $\langle M' \rangle$"

Consider three cases for the proof of correctness:

i. When x doesn't type check, $x \notin E_{TM}$. The output $F(x) = const_{out} \notin Evens_{TM}$.

ii. When $x = \langle M \rangle$ and M accepts some string, then $x \notin E_{TM}$. In this case, M' will accept some odd-length string when reaching step iii. Thus we have $F(x) = \langle M' \rangle \notin Evens_{TM}$.

iii. When $x = \langle M \rangle$ and M accepts no string, then $x \in E_{TM}$. In this case, M' will not accept any odd-length strings, since M' can only accept odd-length strings in step iii when M accepts some string. Therefore, $F(x) = \langle M' \rangle \in Evens_{TM}$.

4. **Examples of languages** (7 points):

For each part of the question, use precise mathematical notation or English to define your examples and then briefly justify why they work.

(a) (*Graded for correctness*) Two undecidable languages L_1 and L_2 over the same alphabet whose intersection $L_1 \cap L_2$ is decidable, or write **NONE** if there is no such example (and explain why).

Solution: Let $L_1 = EQ_{TM}$ and $L_2 = E_{TM}$. We've shown that EQ_{TM} is not recognizable (and thus also undecidable) in Week 9, pg. 5, and shown that E_{TM} is not recognizable (and thus also undecidable) in Week 9, pg. 4. The elements in EQ_{TM} and the elements in E_{TM} are in the form of $\langle M_1, M_2 \rangle$ and $\langle M \rangle$ respectively, where M, M_1, M_2 are Turing machines. Notice no strings in one will be in the other, since they are not of the correct type to be in the other. Thus, it is the case that

$L_1 \cap L_2 = \varnothing$, which is a decidable language that can be decided by a Turing machine that rejects all strings.

(b) (*Graded for correctness*) A regular language L_3 and an unrecognizable language L_4 over the same alphabet whose set-wise concatenation $L_3 \circ L_4$ is unrecognizable, or write **NONE** if there is no such example (and explain why).

Solution: Let $L_3 = \{\varepsilon\}$, and $L_4 = E_{TM}$. L_3 is regular, described by the regular expression ε. And $L_4 = E_{TM}$ is unrecognizable as shown in Week 9, pg. 4. The set-wise concatenation is

$$L_3 \circ L_4 = \{uv \mid u \in L_3, v \in L_4\} = L_4$$

Since L_4 is unrecognizable, so the set-wise concatenation $L_3 \circ L_4$ is unrecognizable.

(c) (*Graded for completeness*) A co-recognizable language L_5 that is NP-complete, or write **NONE** if there is no such example (and explain why). Recall the definition: A language L over an alphabet Σ is called **co-recognizable** if its complement, defined as $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$, is Turing-recognizable.

Solution: Consider $3SAT$. Notice that there is a brute-force algorithm to decide whether or not a string is an encoding of a satisfiable 3cnf-formula. This language is decidable. Recall the fact that a language L is decidable if and only if L and its complement $\overline{L}$ is recognizable. We now have the complement of $3SAT$ is recognizable, so $3SAT$ is co-recognizable. Also, $3SAT$ is NP-complete by Cook-Levin Theorem.