HW6CSE105W25: Sample solutions

CSE105W25 Team

Due: March 13, 2025 at 5pm, via Gradescope

In this assignment,

You will practice analyzing, designing, and working with reductions to compare the difficulty level of computational problems. 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)** (16 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

$f(x) =$

$$\begin{tikzcd} \langle \text{start} \arrow[r] & q_{\text{acc}} & , \varepsilon \rangle \end{tikzcd}$$

if $x = \langle M, w \rangle$ for a Turing machine M and string w and $w \in L(M)$

$$\begin{tikzcd} \langle \text{start} \arrow[r] & q_0 \arrow[loop above, "{0,1,\sqcup \to R}"] & & q_{\text{acc}} & \rangle \end{tikzcd}$$

otherwise

Solution: The attempt has Error Type 3.

i. Error Type 1 is not present: For arbitrary $x \in A_{TM}$, by definition we have $x = \langle M, w \rangle$ for a Turing machine M and string w and $w \in L(M)$, so by the first case of the function definition, 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: For arbitrary $x \notin A_{TM}$, the second case of the function definition means that the output $f(x)$ is a string encoding of a Turing machine that does not match the type of elements in $HALT_{TM}$, since 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.

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

$f(x) =$

$$\begin{tikzcd} \langle \text{start} \arrow[r] & q_{\text{acc}} & , M_w \rangle \end{tikzcd}$$

if $x = \langle M, w \rangle$ for a Turing machine M and string w

$$\begin{tikzcd} \langle \text{start} \arrow[r] & q_{\text{acc}} & , \text{start} \arrow[r] & q_{\text{rej}} & & q_{\text{acc}} & \rangle \end{tikzcd}$$

otherwise.

Where for each Turing machine M, we define

$M_w =$ "On input y
1. Simulate M on w.
2. If it accepts, accept.
3. If it rejects, reject."

[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.

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

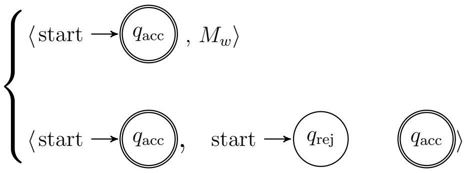
i. Error Type 1 is not present: for arbitrary $x \in A_{TM}$, we have $x = \langle M, w \rangle$ for a Turing machine M and string w and $w \in L(M)$, which falls into the first case of the function definition. In this case, let's denote the output of the function $f(x) = \langle M_0, M_w \rangle$ where M_0 accepts all strings, and M_w also accepts all strings because M accepts w (so for arbitrary string input y, if y is nonempty it is accepted immediately in step 1 of the defining algorithm for M_w and if y is the empty string, it is accepted after M_w simulates M on w). Therefore we get $L(M_1) = L(M_w)$ and $f(x) \in EQ_{TM}$.

ii. Error Type 2 is not present: for arbitrary $x \notin A_{TM}$, 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, we fall into the second case of the function definition so we can denote the output $f(x) = \langle M_1, M_2 \rangle$ where M_1 accepts all strings and M_2 rejects all strings. Since $L(M_1) \neq L(M_2)$, 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 piecewise definition of f. 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

$f(x) =$


if $x = \langle M, w \rangle$ for a Turing machine M and string w

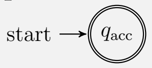
otherwise.

[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, accept.
    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. A witness for Error Type 1 would be $x = \langle M', w' \rangle$ where M' rejects w'. Specifically, we can consider $M' =$ "On input y : 1. Reject." and $w' = \varepsilon$. To calculate $f(x)$, we notice that the format for input x means it 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, since M' rejects w', M_w will reject the empty string (and accept all other strings). Since $\varepsilon \in L(A)$ but $\varepsilon \notin L(M_w)$, we have $L(A) \neq L(M_w)$, i.e. $f(x) = \langle A, M_w \rangle \notin EQ_{TM}$.

ii. There is no Error Type 2: for arbitrary $x \notin HALT_{TM}$, we consider two cases. Case 1: 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}. Case 2: $x = \langle M, w \rangle$ for some Turing machine M and string w but M does not halt on w. In this case, when we define $f(x)$ we use the second case and get $\langle A, M_w \rangle$, where M_w will loop forever on ε (because it simulates M on w as part of its computation) and accept all other strings. Since the language of M_w is not the set of all strings, $L(A) \neq L(M_w)$, so $f(x)$ 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*) $\{ww \mid w \in \{0,1\}^*\} \leq \Sigma^*$ and $f(x) = 11$ for each $x \in \{0,1\}^*$.

Solution: This mapping reduction has an Error Type 2.

i. Error Type 1 is not present: For any $x \in \{ww \mid w \in \{0,1\}^*\}$ (the source set), we get $f(x) = 11 \in \Sigma^*$ (the target set). Thus, Error Type 1 is not present.

ii. Error Type 2 is witnessed by $x = 0 \notin \{ww \mid w \in \{0,1\}^*\}$. Since $x \in \{0,1\}^*$, we still get $f(x) = 11$ which is in the target set (Σ^*) even though it shouldn't be, therefore Error Type 2 exists.

iii. Error Type 3 is not present because we can build a Turing machine that computes this constant function by first deleting the input and then writing 11 on the Turing machine tape and halting.

(e) (*Graded for correctness*) $\Sigma^* \leq_m \{ww \mid w \in \{0,1\}^*\}$ and $f(x) = 11$ for each $x \in \{0,1\}^*$.

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

i. For any string $x \in \Sigma^*$, $f(x) = 11 \in \{ww \mid w \in \{0,1\}^*\}$. Therefore there is no Error Type 1.

ii. There exists no $x \notin \Sigma^*$, so it is vacuously true that "for any $x \notin \Sigma^*, f(x) \notin \{ww \mid w \in \{0,1\}^*\}$", which also means that there is no Error Type 2.

iii. Error Type 3 is not present because because (as in the previous part) we can build a Turing machine that computes this constant function by first deleting the input and then writing 11 on the Turing machine tape and halting.

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

$$\begin{aligned}
IncludesEmptyString_{TM} = \{\langle M\rangle \mid\ & M \text{ is a Turing machine and} \\
& M \text{ accepts the empty string (and maybe other strings too)}\}
\end{aligned}$$

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

Solution: Let's consider the alphabet $\Sigma = \{0,1\}$. We will use the two Turing machines defined below using high-level descriptions:

M_1: "On input x,

i. If $x = \varepsilon$, accept.

```
    ii. Otherwise, reject."
```

M_2: "On input x,

```
    i. Reject."
```

Since M_1 accepts ε, we have $\langle M_1 \rangle \in \mathit{IncludesEmptyString}_{TM}$. Since M_2 rejects ε (i.e. does not accept ε), we have $\langle M_2 \rangle \notin \mathit{IncludesEmptyString}_{TM}$.

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

Solution: To show that $A_{TM} \leq_m \mathit{IncludesEmptyString}_{TM}$, we have to give a computable function that satisfies the "maintain membership property". Consider the computable function computed by the Turing machine below. Let $\mathit{const}_{out} = \langle M_2 \rangle$ for the Turing machine M_2 defined in part (a) (so const_{out} is not in $\mathit{IncludesEmptyString}_{TM}$).

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. $x = \langle M, w \rangle$ for some Turing machine $M$ and string $w$, construct $M'$: "On input $y$,
        i. Run $M$ on $w$.
        ii. If $M$ accepts, accept. If $M$ rejects, reject."
    2. Output $\langle M' \rangle$"
```

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

Case 1: $x \neq \langle M, w \rangle$ for any Turing machine M and string w. Then, $x \notin A_{TM}$, since it is not of the correct type. It fails the type check, and $F(x) = \mathit{const}_{out} \notin \mathit{IncludesEmptyString}_{TM}$.

Case 2: $x = \langle M, w \rangle$, for some Turing machine M and string w and M loops on w. In this case, $x \notin A_{TM}$ and we want to check that $F(x) \notin \mathit{IncludesEmptyString}_{TM}$. By definition of F, since x passes the type check, $F(x) = \langle M' \rangle$. Consider the behavior of M' on input ε. By definition, this computation first runs M on w. Since M loops on w, M' loops on each input string y, which means it also loops on ε and does not accept it. Thus, $F(x) = \langle M' \rangle \notin \mathit{IncludesEmptyString}_{TM}$.

Case 3: $x = \langle M, w\rangle$, and M rejects w. In this case also, $x \notin A_{TM}$. By definition of F, since x passes the type check, $F(x) = \langle M'\rangle$. Consider the behavior of M' on input ε. By definition, this computation first runs M on w. Since M rejects w, M' rejects every input string, which means it also rejects ε. Thus, $F(x) = \langle M'\rangle \notin IncludesEmptyString_{TM}$.

Case 4: $x = \langle M, w\rangle$, and M accepts w. Finally in this case, $x \in A_{TM}$. Again, because x passes the type check , $F(x) = \langle M'\rangle$. Consider the behavior of M' on ε. First, it runs M on w, and since M accepts w, M' accepts every input string, which means it also accepts ε. Thus, $F(x) = \langle M'\rangle \in IncludesEmptyString_{TM}$.

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

Solution: This solution is very similar to part (c), so we point out just the different pieces.
This time, we build the following Turing machine to compute a function: 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. Run $M$ on $w$.
    ii. If $M$ accepts, accept. If $M$ rejects, accept."
2. Output $\langle M'\rangle$"
```

Notice that step 1.ii is the different piece compared to part (c).
The only case that's materially different from part (c) is when $x = \langle M, w\rangle$ and M rejects w:

Case 3: $x = \langle M, w\rangle$ for some Turing machine M and string w, and M rejects w. Then, $x \in HALT_{TM}$ and we want to show that $F(X) \in IncludesEmptyString_{TM}$. Consider the behavior of M' on ε. First the computation, runs M on w. Since M rejects w, M' accepts every input string, which means it also accepts ε. Thus, $F(x) = \langle M'\rangle \in IncludesEmptyString_{TM}$.

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

Solution: Yes, it is recognizable. We can build a Turing machine that recognizes it:

M_R: "On input $\langle M\rangle$:

i. Run M on ε

ii. If it accepts, accept. If it rejects, reject."

Let's verify M_R recognizes $IncludesEmptyString_{TM}$. If the input doesn't type check, M_R rejects the input. If the input $x = \langle M \rangle$ and M loops on ε, then $x \notin IncludesEmptyString_{TM}$, and M_R loops on x and does not accept. If the input $x = \langle M \rangle$ and M rejects ε, then $x \notin IncludesEmptyString_{TM}$, and M_R rejects x. Finally, if the input $x = \langle M \rangle$ and M accepts ε, then $x \in IncludesEmptyString_{TM}$, and M_R accepts x. Therefore, M_R recognizes $IncludesEmptyString_{TM}$ and $IncludesEmptyString_{TM}$ is recognizable.

3. **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

$NotIncludesEmptyString_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ does not accept the empty string}\}$

(a) (*Graded for correctness*) Prove that $NotIncludesEmptyString_{TM}$ is not the complement of E_{TM} and is also not the complement of $IncludesEmptyString_{TM}$.

Solution: Let x be a string such that $x \neq \langle M \rangle$ for any Turing machine M. Then $x \notin E_{TM}$ and $x \notin IncludesEmptyString_{TM}$ since it doesn't type check, i.e. $x \in \overline{E_{TM}}$ and $x \in \overline{IncludesEmptyString_{TM}}$. However, $x \notin NotIncludesEmptyString_{TM}$ since it also does not type check. Since there exists a string that is in the complement of E_{TM} and in the complement of $IncludesEmptyString_{TM}$ but is not in $NotIncludesEmptyString_{TM}$, we proved that $NotIncludesEmptyString_{TM}$ is not the complement of E_{TM} and is also not the complement of $IncludesEmptyString_{TM}$.

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

Solution: To show that $\overline{HALT_{TM}} \leq_m NotIncludesEmptyString_{TM}$, we have to give a computable function that satisfies the "maintain membership property". Consider the computable function computed by the Turing machine below. Let $const_{out} = \langle M_2 \rangle \in NotIncludesEmptyString_{TM}$ where M_2 is the TM defined in 2(a) which rejects every input string.

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. Run M on w.
   ii. If M accepts, accept. If M rejects, accept."
2. Output $\langle M' \rangle$"

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

Case 1: $x \neq \langle M, w \rangle$ for any Turing machine M and string w. In this case, $x \in \overline{HALT_{TM}}$, since it is not formatted to represent the correct type of object to be in $HALT_{TM}$. Computing $F(x)$, in step 0 we see $F(x) = const_{out} \in NotIncludesEmptyString_{TM}$ (by definition of $const_{out}$.

Case 2: $x = \langle M, w \rangle$ for some Turing machine M and string w, and M loops on w. Then, $x \in \overline{HALT_{TM}}$. Consider the computation of M' on ε: after the type check, in step 1 the computation runs M on w. Since M loops on w, M' loops on every input string, which means it also loops on ε and does not accept it. Thus, $F(x) = \langle M' \rangle \in NotIncludesEmptyString_{TM}$.

Case 3: $x = \langle M, w \rangle$ for some Turing machine M and string w, and M rejects w. In this case, $x \notin \overline{HALT_{TM}}$ and we need to show that $F(x) \in NotIncludesEmptyString_{TM}$. Consider the computation of M' on ε: after the type-check, the computation runs M on w. Since M rejects w, M' accepts every input string, which means it also accepts ε. Thus, $F(x) = \langle M' \rangle \notin NotIncludesEmptyString_{TM}$ (because of the double negative).

Case 4: $x = \langle M, w \rangle$ for a Turing machine M and string w, and M accepts w. Then, $x \notin \overline{HALT_{TM}}$ and we need to show that $F(x) \in NotIncludesEmptyString_{TM}$. Consider the computation of M' on ε: after the type check, the computation runs M on w. Since M accepts w, M' accepts every input string, which means it also accepts ε. Thus, $F(x) = \langle M' \rangle \notin NotIncludesEmptyString_{TM}$ (because of the double negative).

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

Solution: This solution is very similar to part (b), so we point out just the different pieces.
This time, we build the following Turing machine to compute a function:
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. Run M on w.
   ii. If M accepts, accept. If M rejects, reject."
2. Output $\langle M' \rangle$"

Notice that step 1.ii is the different piece compared to part (b). The only case that's materially different from part (c) is when $x = \langle M, w \rangle$ and M rejects w:

Case 3: $x = \langle M, w \rangle$ for some Turing machine M and string w, and M rejects w. In this case, $x \in \overline{A_{TM}}$. Consider the computation of M' on ε. After the type check, M' simulates running M on w. Since M rejects w, M' rejects every input string, which means it also rejects ε. Thus, $F(x) = \langle M' \rangle \in NotIncludesEmptyString_{TM}$, as required.

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

Solution: Since we've proved that $\overline{A_{TM}} \leq_m NotIncludesEmptyString_{TM}$ and $\overline{A_{TM}}$ is unrecognizable, we know that $NotIncludesEmptyString_{TM}$ is also unrecognizable (by corollary in week 9 notes page 4).

4. **Examples of languages** (6 points):

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

For each language L over an alphabet Σ, we have the associated sets of strings (also over Σ)

$$L^* = \{w_1 \cdots w_k \mid k \geq 0 \text{ and each } w_i \in L\}$$

and

$$SUBSTRING(L) = \{w \in \Sigma^* \mid \text{there exist } x, y \in \Sigma^* \text{ such that } xwy \in L\}$$

and

$$EXTEND(L) = \{w \in \Sigma^* \mid w = uv \text{ for some strings } u \in L \text{ and } v \in \Sigma^*\}$$

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

Solution: Consider (over the alphabet Σ) languages $L_1 = HALT_{TM}$ and $L_2 = \overline{HALT_{TM}}$, which are proved in class as undecidable languages (week 9 notes page 6). We get $L_1 \cup L_2 = HALT_{TM} \cup \overline{HALT_{TM}} = \Sigma^*$. To see if $L_1 \cup L_2$ is co-recognizable, we calculate and analyze its complement: $\overline{L_1 \cup L_2} = \overline{\Sigma^*} = \emptyset$. The empty set is regular (since it can be described by the regular expression $\emptyset$) and thus also recognizable. Therefore, $L_1 \cup L_2$ is co-recognizable.

(b) (*Graded for correctness*) An unrecognizable language L_3 for which $EXTEND(L_3)$ is regular or write **NONE** if there is no such example (and explain why).

Solution: Because we have not specified the exact encoding syntax, we have two situations to consider.
Case 1: $\varepsilon \neq \langle M, w\rangle$ for any Turing machine M and string w, or if it does, at least not for $\langle M, w\rangle$ where the Turing machine M accepts the string w. In this case, let $L_3 = \overline{A_{TM}}$. We've proved in class that $\overline{A_{TM}}$ is unrecognizable (week 8 notes page 6). Moreover, $\varepsilon \in \overline{A_{TM}}$. According to Lemma 1 proved in HW1 Q3, we know that since $\varepsilon \in L_3$, we have $EXTEND(L_3) = \Sigma^*$ which is regular since it can be described by the regular expression Σ^*.

Case 2: Perhaps we have a strange encoding scheme where $\varepsilon = \langle M, w\rangle$ for some specific Turing machine M and string w and M accepts w. In this case, $\varepsilon \notin \overline{A_{TM}}$. However, in this case, let $L_3 = \overline{A_{TM}} \cup \{\varepsilon\}$. Since $\varepsilon \in L_3$, by Lemma 1, once we prove that L_3 is unrecognizable, we will have that L_3 is our example because $EXTEND(L_3) = \Sigma^*$, a regular set. Lemma 2 below will finish off our proof: once we show that, for any unrecognizable A and finite B, $A \cup B$ is unrecognizable, we will have that L_3 is unrecognizable (using $A = \overline{A_{TM}}$ is unrecognizable and $B = \{\varepsilon\}$ which is finite).

Recall Lemma 1: For any alphabet Σ, if a language L over Σ contains ε, then $EXTEND(L)$ is the set of all strings over Σ.

Proof of lemma 1: Let Σ be an alphabet and assume L is a language over Σ that has ε as an element. We need to prove that $EXTEND(L) = \Sigma^*$. Since $EXTEND(L)$ is a set of strings over Σ, it's enough to prove that $\Sigma^* \subseteq EXTEND(L)$. Let w be an arbitrary string over Σ. We need to show that $w \in EXTEND(L)$. To do that we need to write w as uv for some $u \in L$ and $v \in \Sigma^*$. Because of our assumption that the empty string is in L, we can let $u = \varepsilon$ and $v = w$. Then $w = \varepsilon w = uv$, as required to prove that $w \in EXTEND(L)$.

Lemma 2: For A be an arbitrary unrecognizable language and B be an arbitrary finite language, $A \cup B$ is unrecognizable.

Proof of lemma 2: Suppose towards a contradiction that there is an unrecognizable language A and a finite language B where $A \cup B$ is recognizable. Let $N = A \cap B$. Since the intersection of any set with a finite set is finite, N is finite, and hence decidable. Therefore, there are Turing machines M_U, M_B, and M_N where $L(M_U) = A \cup B$ and $L(M_B) = B$ and $L(M_N) = N$, and M_B and M_N are deciders. We define the following Turing machine:

```
M = "On input $w$ :
      1.Run $M_N$ on $w$
      2.If accepts, accept.
      3.If rejects, run $M_U$ on $w$
      4.  If rejects, reject.
      5.  If accepts, run $M_B$ on $w$
      6.    If accepts, reject.
      7.    If rejects, accept."
```

For arbitrary string $w \in A$ we want to show that M accepts it. Case 1: $w \in B$. In this case, $w \in A \cap B = N$, so when we run M on w and in step 1, simulate M_n on w, it accepts, and M accepts in step 2. Case 2: $w \notin B$. In this case, $w \notin A \cap B = N$ and $w \in A \cup B$ and $w \notin B$. When we run M on w, in step 1 M_N on w halts and rejects (because M_N is a decider), so in step 3, we run M_U on w which halts and accepts, so in step 5 we run M_B on w, which halts and rejects (because M_B is a decider), so in step 7, we accept. Thus, if $w \in A$, M accepts it.
For arbitrary string $w \notin A$ we want to show that M does not accept it. Case 1: $w \in B$. In this case, $w \notin A \cap B = N$, $w \in A \cup B$. When we run M on w, in step 1 M_N on w halts and rejects (because M_N is a decider), so in step 3, we run M_U on w which halts and accepts, so in step 5 we run M_B on w, which halts and accepts, so in step 6, we reject. Case 2: $w \notin B$ so (since $w \notin A$) $w \notin A \cup B$. When we run M on w, in step 1 M_N on w halts and rejects (because M_N is a decider), so in step 3, we run M_U on w which either halts and rejects (in which case we reject in step 4) or loops (in which we case M also loops). Thus, if $w \notin A$ M does not accept it.
We proved that, if B is finite and $A \cup B$ is recognizable, then A is recognizable, a contradiction with our assumption that A is unrecognizable.

(c) (*Graded for completeness*) A co-recognizable language L_4 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.
This part of the question uses definitions from Week 10 of the course.

Solution: We let $L_4 = 3SAT$. The Cook-Levin Theorem we discussed in class proves that this set is NP complete (it is in NP and every NP problem polynomial-time reduces to it). Each NP complete problem is decidable (because Turing machines and nondeterministic Turing machines are equally expressive). 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.