Solutions to Homework 5

Problems marked with **E** are graded on effort, which means that they are graded subjectively on the perceived effort you put into them, rather than on correctness. For bonus questions, we will not provide any insight during office hours or Piazza, and we do not guarantee anything about the difficulty of these questions. We strongly encourage you to typeset your solutions in LaTeX.

If you collaborated with someone, you must state their names. You must write your own solution and may not look at any other student's write-up.

0. If applicable, state the name(s) and uniqname(s) of your collaborator(s).

> **Solution:**

E 1. Rice's Theorem is a powerful tool that allows us to classify many languages as undecidable. It states that if $\mathbb{P}$ is a semantic, nontrivial property, the language $L_{\mathbb{P}} := \{\langle M \rangle : L(M) \in \mathbb{P}\}$ is undecidable. Notice that $\mathbb{P}$ is a *set of languages*, so when we say *"a language L satisfies $\mathbb{P}$"*, we mean that $L \in \mathbb{P}$. Using this, we can state Rice's Theorem more informally as: if L is the set of Turing machines whose language satisfies $\mathbb{P}$, where $\mathbb{P}$ is a nontrivial semantic property, then L is undecidable.

(a) In your own words, define *a semantic property*. Give an example of a semantic property.

> **Solution:** A semantic property is any property relating to the output of a program, as opposed to its implementation details. Thus, a semantic property is anything concerning the language of a Turing Machine, so we express it set theoretically, as just some set of languages:
>
> $$\mathbb{P} = \{L_1, L_2, L_3 \dots\}$$
>
> However, this is slightly oversimplified, since there are an uncountably infinite number of languages. We may not be able to list them in this format with natural number indices. Recall that languages are sets of inputs that a Turing Machine accepts, and Σ^* is the set of inputs, so for every language L, $L \subseteq \Sigma^*$. Thus we can define a semantic property $\mathbb{P}$ as a set that satisfies:
>
> $$\mathbb{P} \subseteq \{L : L \subseteq \Sigma^*\}$$
>
> An example of a semantic property is the set
>
> $$\mathbb{P}_{EVEN} = \{L : |L| \text{ is even}\}$$

(b) In your own words, define *a nontrivial semantic property*. Give an example of a trivial semantic property and a nontrivial semantic property.

> **Solution:** A trivial semantic property is any semantic property that all languages of machines satisfy or no languages of machines satisfy. A nontrivial semantic property is any semantic property that isn't trivial.

Alternatively, a nontrivial semantic property is any semantic property $\mathbb{P}$ such that there are two Turing machines M_1 and M_2 such that $L(M_1) \in \mathbb{P}$ and $L(M_2) \notin \mathbb{P}$.

An example of a trivial semantic property is $\mathbb{P} = \emptyset$. Our example from (a) is a nontrivial semantic property:

$$\mathbb{P}_{EVEN} = \{L : |L| \text{ is even}\}$$

It is nontrivial because we can find recognizable languages: $L_1 = \{0, 1\} \in \mathbb{P}_{EVEN}$ and $L_2 = \{0\} \notin \mathbb{P}_{EVEN}$. Each of these languages is the language of some Turing machine since they are both finite.

(c) For your two examples in (*b*), state the languages associated with each property, then determine if each of these languages is undecidable or decidable by applying Rice's Theorem.

Solution: For the trivial semantic property $\mathbb{P} = \emptyset$, the associated language is:

$$L = \{\langle M \rangle : L(M) \in \emptyset\} = \emptyset$$

L is decidable by Rice's theorem. Additionally, this language is empty, so we can directly decide it with the program that immediately rejects all inputs.

$$L_{\mathbb{P}_{EVEN}} = \{\langle M \rangle : |L(M)| \text{ is even}\}$$

By Rice's Theorem, $L_{\mathbb{P}_{EVEN}}$ is undecidable.

2. For each of the following statements, determine if the statement is always, sometimes, or never true. Justify your statement with a proof.
Hint: To prove that something isn't always true, it is sufficient to provide a counterexample.

(a) If L is an unrecognizable language, then L is (always/sometimes/never) undecidable.

Solution: If L is an unrecognizable language, then L is (**always**/sometimes/never) undecidable.

Proof. We know that if L is decidable, then L is recognizable. Taking the contrapositive, we find that if L is not recognizable, then L is not decidable. □

(b) If L is a recognizable language then $\overline{L}$ is (always/sometimes/never) recognizable.

Solution: If L is a recognizable language then $\overline{L}$ is (always/**sometimes**/never) recognizable.
For the case where L and $\overline{L}$ are both recognizable, consider $L = \{1\}$.

L is recognizable; consider the program that accepts if the input is 1 and rejects otherwise. $\overline{L}$ is similarly decided (and hence recognized) by the program that rejects if the input is 1 and accepts otherwise.

Next, we need to find an example where L is recognizable but $\overline{L}$ is unrecognizable. We proved in discussion that any recognizable and undecidable language will have an unrecognizable complement. For one example, consider L_{ACC}.

3. For each of the following languages, state with proof whether it is **decidable** or **undecidable**. If Rice's Theorem is applicable, you **must** use it. Otherwise, prove your answer by constructing a decider or performing a Turing reduction.

   (a) $L_{\text{FINITE-ODD}} = \{\langle M\rangle : |L(M)| \text{ is finite and odd}\}$

Solution: We can apply Rice's theorem here. Our semantic property is

$$\mathbb{P}_{\text{FINITE-ODD}} = \{L \subseteq \Sigma^* : |L| \text{ is finite and odd}\}.$$

We determine

$$\begin{aligned}\langle M\rangle \in L_{\text{FINITE-ODD}} &\iff |L(M)| \text{ is finite and odd}\\ &\iff L(M) \in \mathbb{P}_{\text{FINITE-ODD}}\\ &\iff \langle M\rangle \in L_{\mathbb{P}_{\text{FINITE-ODD}}}.\end{aligned}$$

It follows that $L_{\text{FINITE-ODD}} = L_{\mathbb{P}_{\text{FINITE-ODD}}}$.
A recognizable language that does not satisfy $\mathbb{P}_{\text{FINITE-ODD}}$ is $\varnothing$ (recall that 0 is an even number); a recognizable language that satisfies the property is $\{0\}$ (this is recognizable since every finite language is decidable, and every decidable language is recognizable). By Rice's theorem, we have that $L_{\text{FINITE-ODD}} = L_{\mathbb{P}_{\text{FINITE-ODD}}}$ is undecidable.

   (b) $L_{\text{GCD}} = \{(\langle M\rangle, p, q) : M \text{ is a TM}, p, q \in \mathbb{N}, |L(M)| < \gcd(p, q)\}$

Solution: We can't apply Rice's Theorem, since Rice's Theorem applies only to languages of the form: $L_{\mathbb{P}} = \{\langle M\rangle : L(M) \in \mathbb{P}\}$, where $\mathbb{P}$ is a set of languages. Elements of L_{GCD} are triplets $(\langle M\rangle, p, q)$, instead of just Turing Machines, M.

Instead, we will do a Turing reduction $L_{\varnothing} \leq_{\text{T}} L_{\text{GCD}}$ to show that L_{GCD} is undecidable. Assume that there is a decider E for L_{GCD}.

D = "on input $(\langle M\rangle)$:
1: Run E on input $(\langle M\rangle, 1, 1)$
2: **if** E accepts $(\langle M\rangle, 1, 1)$ **then** accept
3: **else** reject"

We claim that D decides $L_{\varnothing}$. We have two cases:

- If $L(M) = \varnothing$, we see that $|L(M)| = 0 < 1 = \gcd(1, 1)$. It follows that E accepts on the third line, so D accepts the input $\langle M \rangle$.
- If $L(M) \neq \varnothing$, then $|L(M)| \geq 1 = \gcd(1, 1)$. It follows that E rejects, so D rejects the input $\langle M \rangle$.

Thus, D decides $L_\varnothing$, and so we have shown that $L_\varnothing \leq_\mathrm{T} L_\mathrm{GCD}$. Since $L_\varnothing$ is undecidable, we conclude that L_GCD is undecidable.

(c) $L_{RECOGNIZABLE} = \{\langle M \rangle : L(M) \text{ is recognizable}\}$

Solution: We can apply Rice's theorem here. The property $\mathbb{P}_\mathrm{RECOGNIZABLE} = \{L \subseteq \Sigma^* : L(M) \text{ is recognizable}\}$ is semantic because it is a characteristic of the language. $\mathbb{P}_\mathrm{RECOGNIZABLE}$ is also trivial because all recognizable languages satisfy this property. Because $\mathbb{P}_\mathrm{RECOGNIZABLE}$ is a semantic trivial property, by Rice's Theorem $L_{RECOGNIZABLE}$ must be decidable.

(d) $L_{EECS-CLASSES} = \{\langle M \rangle : M \text{ accepts } x \text{ for some } x \in \{183, 280, 281, 370, 376\}\}$.

Solution: We can apply Rice's theorem here. The semantic property is $\mathbb{P}_\mathrm{EECS} = \{L \subseteq \Sigma^* : \exists x \in \{183, 280, 281, 370, 376\} \text{ such that } x \in L\}$.

$$\begin{aligned}
\langle M \rangle \in L_\text{EECS-CLASSES} &\iff \exists x \in \{183, 280, 281, 370, 376\} \text{ such that } M(x) \text{ accepts} \\
&\iff \exists x \in \{183, 280, 281, 370, 376\} \text{ such that } x \in L(M) \\
&\iff L(M) \in \mathbb{P}_\mathrm{EECS} \\
&\iff \langle M \rangle \in L_{\mathbb{P}_\mathrm{EECS}}.
\end{aligned}$$

A recognizable language that satisfies the property is Σ^* (since $\Sigma^* \subseteq \Sigma^*$ and all inputs are in Σ^*). A recognizable language that does not satisfy this property is $\varnothing$. By Rice's theorem, we have that $L_\text{EECS-CLASSES}$ is undecidable.

(e) $L_{EMPTY-EMPTY} = \{\langle M \rangle : L(M) = L_\varnothing\}$

Solution: We can apply Rice's Theorem here. Let $\mathbb{P} = \{L_\varnothing\}$. Since $L_\varnothing$ is unrecognizable, no recognizable languages are equal to it, so $\mathbb{P}$ is trivial. Thus, $L_{EMPTY-EMPTY}$ is decidable by Rice's Theorem.

Alternatively, we could directly notice that since no languages of machines are equal to $L_\varnothing$ (since $L_\varnothing$ is unrecognizable), $L_{EMPTY-EMPTY}$ is itself empty, so we can decide it by simply rejecting all inputs.

In words, $L_{EMPTY-EMPTY}$ is the set of Turing Machines M, such that the language of M is $L_\varnothing$. This means that M accepts all Turing Machines whose language is empty, and rejects or loops on all Turing Machines whose language is not empty. Fortunately for us, no such M can exist, so this language is empty, as shown above.

4. For each of the following languages, state with proof whether it is **recognizable** or **unrecognizable**.

   (a) $L_{REJECT} = \{(\langle M \rangle, x) : M \text{ rejects } x\}$

Solution: L_{REJECT} is recognizable because we can construct a recognizer.

R = "on input $(\langle M \rangle, x)$:
1: Run M on input x
2: **if** M rejects **then** accept
3: **else** reject

We now show that R is indeed a recognizer.

- $(\langle M \rangle, x) \in L_{REJECT} \implies M$ rejects on input x, so R accepts.
- $(\langle M \rangle, x) \notin L_{REJECT} \implies M$ accepts or loops on input x. If M loops on x, then R will loop as well. If M accepts x, then R rejects M. Thus, when $(\langle M \rangle, x) \notin L_{REJECT}$, R will either loop or reject.

R recognizes L_{REJECT}, so L_{REJECT} is recognizable.

   (b) $L_{376-PRIMES} = \{\langle M \rangle : M \text{ accepts at least 376 prime numbers}\}$

Solution:
$L_{376-PRIMES}$ is recognizable. We can construct a recognizer R for $L_{376-PRIMES}$.

R = "on input $\langle M \rangle$:
1: Initialize Count $\leftarrow 0$
2: **for** $i = 1, 2, 3 \ldots \infty$ **do**
3: **for** $x = 2, 3, 5, 7, \ldots, p_i$, where p_i is the ith prime **do**
4: Run a step of M on x
5: **if** M accepts x **then** Count $\leftarrow$ Count+1
6: **if** Count≥ 376 **then** accept

Analysis:

- If $\langle M \rangle \in L_{376-PRIMES}$ then M will accept ≥ 376 primes. So Count will reach 376 and R will accept $\langle M \rangle$.
- If $\langle M \rangle \notin L_{376-PRIMES}$ then M will accept < 376 primes. So Count can never reach 376, because Count counts the distinct primes that M accepts. Thus, R will not accept $\langle M \rangle$ (and in fact loops on $\langle M \rangle$ – since the set of prime numbers is infinite, the dovetailing process will never run out of prime numbers to try).

Since we have constructed a machine R which accepts all inputs $\langle M \rangle \in L_{376-PRIMES}$ and rejects or loops on all inputs $\langle M \rangle \notin L_{376-PRIMES}$, we have constructed a recognizer for $L_{376-PRIMES}$. Thus, we conclude that $L_{376-PRIMES}$ is recognizable.

(c) $L_{\leq 376} = \{\langle M \rangle : |L(M)| \leq 376\}$

Solution: $L_{\leq 376}$ is unrecognizable, because it is undecidable and its complement is recognizable. We can use Rice's Theorem to determine that $L_{\leq 376}$ is undecidable. The semantic property is:

$$\mathbb{P} = \{L \subseteq \Sigma^* : |L| \leq 376\}$$

Furthermore, $\mathbb{P}$ is nontrivial, since we can identify a recognizable language that satisfies this property and a recognizable language that does not. Σ^* does not satisfy this property (since $|\Sigma^*| > 376$). A recognizable language that satisfies this property is $\emptyset$, (since $|\emptyset| \leq 376$).

Now, we show that $\overline{L_{\leq 376}}$ is recognizable. We call this language:

$$L_{>376} = \{\langle M \rangle : |L(M)| > 376\} = \overline{L_{\leq 376}}$$

$L_{>376}$ is recognizable. We construct a recognizer R for $L_{>376}$ as follows. At a high level, we are basically trying to simulate running M on every possible input and keeping track of how many of them that M accepts. We proceed by numbering every input as $\Sigma^* = \{x_1, x_2, \dots\}$

R = "on input $\langle M \rangle$:

1: Initialize Count $\leftarrow 0$
2: **for** $i = 0, 1, 2, 3 \dots \infty$ **do**
3: **for** $x_j = x_1, x_2, x_3, \dots, x_i$ **do**
4: Run a step of $M(x_j)$
5: **if** M accepts x_j **then** Count $\leftarrow$ Count+1
6: **if** Count$>$ 376 **then** accept

Analysis:

- If $\langle M \rangle \in L_{>376}$ then M will accept more than 376 inputs, so it will accept at least 377 inputs. Label the indices of the first 377 inputs that M accepts $a_1, \dots, a_{377}$. Thus, M halts on all these inputs. Let $n_1, \dots, n_{377}$ be the number of iterations that each of these inputs requires to halt. Thus, when $i = \max(a_1 + n_1, \dots, a_{377} + n_{377})$, the for loop will accept the 377th input, and so R will accept $\langle M \rangle$. Note that the previous loops only run i machines for at most i steps each, so this will take finite time, and so R will not loop on a previous for loop iteration.
- If $\langle M \rangle \notin L_{>376}$ then M will accept less than or equal to 376 inputs. So Count can never be larger than 376, because Count counts the distinct inputs that M accepts. Thus, R will not accept $\langle M \rangle$ and in fact loops on $\langle M \rangle$.

Since we have constructed a machine R which accepts all inputs $\langle M \rangle \in L_{>376}$ and rejects or loops on all inputs $\langle M \rangle \notin L_{>376}$, we have constructed a recognizer for $L_{>376} = \overline{L_{\leq 376}}$. Since we already showed that $L_{\leq 376}$ is undecidable, we conclude that $L_{\leq 376}$ is unrecognizable.

(d) $L_{EECS-CLASSES} = \{\langle M \rangle : M \text{ accepts } x \text{ for some } x \in \{183, 280, 281, 370, 376\}\}$.

Solution: $L_{EECS-CLASSES}$ is recognizable. We construct a recognizer R for $L_{EECS-CLASSES}$ as follows:

R = "on input $\langle M \rangle$:
1: **for** $i = 1, 2, \cdots \infty$ **do**
2: **for** $x \in \{183, 280, 281, 370, 376\}$ **do**
3: Run the next step of the execution of M on x
4: **if** M accepts x **then** accept
5: reject

Analysis:

- If $\langle M \rangle \in L_{EECS-CLASSES}$, then M accepts x for some $x \in \{183, 280, 281, 370, 376\}$. Let k be the minimal number of steps M takes to accept an $x \in \{183, 280, 281, 370, 376\}$. Then, R will accept $\langle M \rangle$ in iteration $i = k$ of its for loop.
- If $\langle M \rangle \notin L_{EECS-CLASSES}$, then M does not accept x for any $x \in \{183, 280, 281, 370, 376\}$. If M loops on at least one input $x \in \{183, 280, 281, 370, 376\}$, then R will loop on $\langle M \rangle$. Otherwise, R rejects $\langle M \rangle$.

Since R accepts all $\langle M \rangle \in L_{EECS-CLASSES}$ and rejects or loops on all $\langle M \rangle \notin L_{EECS-CLASSES}$, R is a recognizer for $L_{EECS-CLASSES}$, so $L_{EECS-CLASSES}$ is recognizable.

5. Recall that the Kolmogorov complexity $K(s)$ of a string $s \in \{0,1\}^*$ (with respect to some fixed programming language) is defined as the bit length of a shortest program that, when run on the empty string ε as input, outputs s.

(a) Prove that, for all $n \geq 1$, $|\{s \in \{0,1\}^* : K(s) = n\}| \leq 2^n$, i.e. there are at most 2^n binary strings that have Kolmogorov complexity exactly n.

Solution: For each string with Kolmogorov complexity n, there needs to be a program of length n that outputs it. There are at most 2^n programs with length n (since a program of length n is a binary string with n digits). Therefore there are at most 2^n strings that have Kolmogorov complexity of n.

(b) Prove that, for all $n \geq 1$, $|\{s \in \{0,1\}^* : K(s) \leq n\}| \leq 2^{n+1} - 1$, i.e. there are at most $2^{n+1} - 1$ binary strings that have Kolmogorov complexity less than or equal to n.

Solution: Using our answer from part a, the number of strings with Kolmogorov complexity n is at most 2^n. Thus, the number of strings with Kolmogorov complexity

$\leq n$ satisfies:

$$|\{s \in \{0,1\}^* : K(s) \leq n\}| = \sum_{i=0}^{n} |\{s \in \{0,1\}^* : K(s) = i\}|$$
$$\leq \sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$$

(c) Conclude that for all $n \geq 1$, there exists a binary string of length $n+1$ whose Kolmogorov complexity is strictly greater than n.

Solution: As shown above, the number of strings with Kolmogorov complexity less than or equal to n is at most $2^{n+1}-1$. There are 2^{n+1} binary strings of length $n+1$, so at least one of these strings must have Kolmogorov complexity more than n.

Such a string is called an *incompressible* string. These strings cannot be output by any program shorter than them. The existence of these strings can be used to give proofs of various results in mathematics, such as "there are an infinite number of primes," and certain impossibility results similar to those shown by Gödel, Cantor, and Turing.

E (d) A company has released a new compression algorithm COMPRESS that will compress *every* file to at most $\frac{281}{376}$ of its original size. Determine, with justification, if an accompanying *lossless* decompression algorithm can exist for this compression algorithm.
(A decompression algorithm is *lossless* if it always outputs the original file, given the compression algorithm's output when run on the original file.)

Solution: We model files as binary strings.
Consider files of length 376, of which there are 2^{376}. COMPRESS applied on any file of length 376 will output some file of length ≤ 281. There are $\sum_{i=0}^{281} 2^i$ different files of length ≤ 281. We know that $\sum_{i=0}^{281} 2^i = 2^{281+1} - 1 < 2^{282}$, so there are fewer than 2^{282} different files of length ≤ 281.
Now for any arbitrary decompression algorithm, we can run it on each of the compressed files of length ≤ 281. This must yield fewer than 2^{282} different files of size 376 because we had fewer than 2^{282} different compressed files. That leaves $2^{376} - 2^{282} > 0$ files $F \in \{0,1\}^{376}$ that are not among these outputs of the decompression algorithm. So, compressing any such F and then decompressing the result does not yield the original F. Therefore, there is no lossless decompression algorithm for this compression scheme.

E 6. Come up with an unrecognizable language and prove that it is unrecognizable. The language you choose should not be one that has been presented in lecture, discussion, or homework, nor should it be the complement of a language that was presented.

For your reference, the languages L_{ACC}, L_{HALT}, $L_{\varepsilon-\text{HALT}}$, $L_{\varnothing}$, L_{EQ}, $L_{\text{HALT}<376}$, all languages defined in this assignment, and these languages' complements may not be used as answers to this.

Solution: Answers will vary. Any valid answer should make use of the following theorem from class:

Let $L \subseteq \Sigma^*$ be undecidable yet recognizable. Then $\overline{L}$ is unrecognizable.

or its equivalent:

Let $L \subseteq \Sigma^*$ be undecidable, with $\overline{L}$ recognizable. Then L is unrecognizable.

Acceptable answers for this are the complement of any known undecidable-but-recognizable language. An example would be $\overline{L_{\text{SELFIE}}}$ where

$$L_{\text{SELFIE}} = \{\langle M \rangle \mid M \text{ accepts its own source code}\}.$$

We first show that L_{SELFIE} is undecidable by showing that $L_{\text{HALT}} \leq_T L_{\text{SELFIE}}$. We assume we have a blackbox decider S for L_{SELFIE} and construct a decider H for L_{HALT} as follows:

H = "on input $(\langle M \rangle, x)$:
1: Construct a machine M' as follows:

> M' = "on input w:
> 1: Run M on x
> 2: Accept"

2: Run $S(\langle M' \rangle)$ and return as S does"

Analysis:

- $(\langle M \rangle, x) \in L_{\text{HALT}} \implies M$ halts on $x \implies M'$ accepts all $w \implies M'$ accepts $\langle M' \rangle$ (because $\langle M' \rangle \in \Sigma^*$) $\implies S$ accepts $M' \implies H$ accepts $(\langle M \rangle, x)$
- $(\langle M \rangle, x) \notin L_{\text{HALT}} \implies M$ loops on $x \implies M'$ loops on all $w \implies M'$ loops on $\langle M' \rangle$ (because $\langle M' \rangle \in \Sigma^*$) $\implies S$ rejects $M' \implies H$ rejects $(\langle M \rangle, x)$

Thus $L_{\text{HALT}} \leq_T L_{\text{SELFIE}}$.

Now we show that L_{SELFIE} is recognizable by constructing a recognizer R.

R = "on input $\langle M \rangle$:
1: Run M on $\langle M \rangle$
2: Return as M does"

Analysis:

- If $\langle M \rangle \in L_{\text{SELFIE}}$, then M accepts $\langle M \rangle$, so R accepts $\langle M \rangle$
- If $\langle M \rangle \notin L_{\text{SELFIE}}$, then either M loops on $\langle M \rangle$ (in which case R loops on $\langle M \rangle$) or M rejects $\langle M \rangle$ (in which case R rejects $\langle M \rangle$)

Thus R recognizes L_{SELFIE}.

Since L_{SELFIE} is undecidable but recognizable, $\overline{L_{\text{SELFIE}}}$ is unrecognizable.

Optional bonus questions

For bonus questions, we will not provide any insight during office hours or Piazza, and we do not guarantee anything about the difficulty of these questions. Please submit these normally via Gradescope unless the problem states otherwise. *Only attempt these questions **after** you have finished the rest of the homework.*

1. Show that both L_{EQ} and $\overline{L_{\text{EQ}}}$ are unrecognizable.

Solution: We cannot show that a language is unrecognizable through a Turing reduction; in fact, we have previously shown that $L \leq_{\text{T}} \overline{L}$ for any language L, so we have $\overline{L_{\text{ACC}}} \leq_{\text{T}} L_{\text{ACC}}$ even though $\overline{L_{\text{ACC}}}$ is unrecognizable but L_{ACC} is recognizable. (There is another form of reduction called a *mapping reduction* that can be used to show unrecognizability, but it is beyond the scope of this course. You can read about it in Sipser if you are interested.)

Recall though that a Turing reduction is just syntactic sugar over a proof by contradiction – our original proof that L_{HALT} is undecidable was through a full proof by contradiction that L_{HALT} being decidable means that L_{ACC} is decidable, and we then introduced the concept of a Turing reduction as a simplification of that proof structure. We can use the same kind of proof to show that a language is unrecognizable – assume that it is recognizable, then show that that assumption leads to a known unrecognizable language being recognizable.

We first show that L_{EQ} is unrecognizable. Assume for the purposes of contradiction that it is recognizable. Then there exists a recognizer R such that:

- if $L(M_1) = L(M_2)$, then R accepts $(\langle M_1 \rangle, \langle M_2 \rangle)$
- if $L(M_1) \neq L(M_2)$, then R rejects or loops on $(\langle M_1 \rangle, \langle M_2 \rangle)$

We will construct a recognizer S for $\overline{L_{\text{ACC}}}$ such that:

- if M rejects or loops on x, then S accepts $(\langle M \rangle, x)$
- if M accepts x, then S rejects or loops on on $(\langle M \rangle, x)$

We construct S as follows:

S = "on input $(\langle M \rangle, x)$:
1: Construct a machine M_1 as follows:

> M_1 = "on input w:
> 1: Run M on x and answer as M"

2: Construct a machine M_2 as follows:

> M_2 = "on input w:
> 1: Reject"

3: Run R on $(\langle M_1 \rangle, \langle M_2 \rangle)$ and answer as R"

Analysis:

- $(\langle M \rangle, x) \in \overline{L_{ACC}} \implies M$ rejects or loops on $x \implies M_1$ rejects or loops on all inputs $\implies L(M_1) = \emptyset = L(M_2) \implies R$ accepts $(\langle M_1 \rangle, \langle M_2 \rangle) \implies S$ accepts $(\langle M \rangle, x)$
- $(\langle M \rangle, x) \notin \overline{L_{ACC}} \implies M$ accepts $x \implies M_1$ accepts all inputs $\implies L(M_1) = \Sigma^* \neq L(M_2) = \emptyset \implies R$ rejects or loops on $(\langle M_1 \rangle, \langle M_2 \rangle) \implies S$ rejects or loops on $(\langle M \rangle, x)$

Thus, S is a recognizer for $\overline{L_{\text{ACC}}}$. This is a contradiction, since we know that $\overline{L_{\text{ACC}}}$ is unrecognizable, so our original assumption that L_{EQ} is recognizable must be false.

We now show that $\overline{L_{\text{EQ}}}$ is unrecognizable. Assume for the purposes of contradiction that it is recognizable. Then there exists a recognizer T such that:

- if $L(M_1) \neq L(M_2)$, then T accepts $(\langle M_1 \rangle, \langle M_2 \rangle)$
- if $L(M_1) = L(M_2)$, then T rejects or loops on $(\langle M_1 \rangle, \langle M_2 \rangle)$

We will construct a recognizer S for $\overline{L_{\text{ACC}}}$ such that:

- if M rejects or loops on x, then S accepts $(\langle M \rangle, x)$
- if M accepts x, then S rejects or loops on on $(\langle M \rangle, x)$

We construct S as follows:

> S = "on input $(\langle M \rangle, x)$:
> 1: Construct a machine M_1 as follows:
>
> > M_1 = "on input w:
> > 1: Run M on x and answer as M"
>
> 2: Construct a machine M_2 as follows:
>
> > M_2 = "on input w:
> > 1: Accept"
>
> 3: Run T on $(\langle M_1 \rangle, \langle M_2 \rangle)$ and answer as T"

Analysis:

- $(\langle M \rangle, x) \in \overline{L_{ACC}} \implies M$ rejects or loops on $x \implies M_1$ rejects or loops on all inputs $\implies L(M_1) = \emptyset \neq L(M_2) = \Sigma^* \implies T$ accepts $(\langle M_1 \rangle, \langle M_2 \rangle) \implies S$ accepts $(\langle M \rangle, x)$
- $(\langle M \rangle, x) \notin \overline{L_{ACC}} \implies M$ accepts $x \implies M_1$ accepts all inputs $\implies L(M_1) = \Sigma^* = L(M_2) \implies T$ rejects or loops on $(\langle M_1 \rangle, \langle M_2 \rangle) \implies S$ rejects or loops on $(\langle M \rangle, x)$

Thus, S is a recognizer for $\overline{L_{\text{ACC}}}$. This is a contradiction, since we know that $\overline{L_{\text{ACC}}}$ is unrecognizable, so our original assumption that $\overline{L_{\text{EQ}}}$ is recognizable must be false.

2. (From HW1) Compute a tight upper bound on the number of moves in the flipping game described in lecture. That is, find a positive integer t_n such that:

   - For all possible starting configurations, and for all possible sequences of legal moves, the flipping game on the $n \times n$ board ends in at most t_n moves.
   - There is some starting configuration, and some sequence of legal moves, such that the flipping game on the $n \times n$ board ends in exactly t_n moves.

   In order to be awarded bonus points, your answer **must** be supplemented with a formal proof.

   Note that we only defined the flipping game for boards with odd dimensions, so n should be odd (though we could attempt to extend to n being even by specifying what are the rules when the number of OSU and UMich chips in a column or row are the same).

   Submit your solution to this problem via e-mail directly to Professor Volkovich at ilyavol@umich.edu.