COMP 3803 — Fall 2025 — Solutions Problem Set 6

Question 1: We have seen in class that the language

$$Halt = \{\langle M, w\rangle : \; M \text{ is a Turing machine that terminates on the input string } w\}$$

is undecidable. In the language *PrintB* that is defined below, Σ denotes the input alphabet of the Turing machine M, and Γ denotes its tape alphabet.

$$\begin{aligned} PrintB = \{\langle M, w, b\rangle \text{:} \quad & M \text{ is a Turing machine, } w \in \Sigma^*,\ b \in \Gamma, \text{ when running } M \\ & \text{on input } w,\ M \text{ writes } b \text{ on the tape at least once}\}. \end{aligned}$$

Prove that *PrintB* is undecidable.
Hint: Given an input $\langle M, w\rangle$ for *Halt*, modify M such that the resulting Turing machine prints a new symbol, say #, at the moment it terminates.

Solution: As always, the proof is by contradiction. We assume that *PrintB* is decidable. Let H be a Turing machine that decides *PrintB*: For any string $\langle M, w, b\rangle$,

- if the Turing machine M, on input w, writes b at least once, then H terminates and accepts $\langle M, w, b\rangle$.
- if the Turing machine M, on input w, never writes b, then H terminates and rejects $\langle M, w, b\rangle$.
- Note: It may happen that M does not terminate on input w. In this case, H is still able to find out whether or not M writes b at least once. In particular, H terminates.

We are going to show that *Halt* is decidable. Consider the following algorithm H', which takes as input $\langle M, w\rangle$:

Step 1: Modify the Turing machine M as follows, and denote the resulting Turing machine by M':

- Add a new symbol # to the tape alpahet.
- Replace each instruction
  $$ra \to q_{accept} *_1 *_2,$$
  where $*_1$ is in the old tape alphabet and $*_2$ is R, L, or N, by
  $$ra \to q_{accept} \# *_2 .$$
- Replace each instruction
  $$ra \to q_{reject} *_1 *_2,$$
  where $*_1$ is in the old tape alphabet and $*_2$ is R, L, or N, by
  $$ra \to q_{reject} \# *_2 .$$

- For each $r \notin \{q_{accept}, q_{reject}\}$, add instructions

$$r\# \to r\#N.$$

Note: M enters the accept or reject state (and, thus, terminates), if and only if M' prints $\#$.

Step 2: Run algorithm H on input $\langle M', w, \# \rangle$.

- If H terminates in the accept state, then H' terminates in the accept state.
- If H terminates in the reject state, then H' terminates in the reject state.

Since H terminates on every input, H' also terminates on every input. Since the following are equivalent

- $\langle M, w \rangle \in \mathit{Halt}$,
- M terminates on input w,
- on input w, M' prints $\#$ at least once,
- $\langle M', w, \# \rangle \in \mathit{PrintB}$,
- H accepts $\langle M', w, \# \rangle$,
- H' accepts $\langle M, w \rangle$,

algorithm H' decides *Halt*. Thus, *Halt* is decidable, which is a contradiction. We conclude that *PrintB* is undecidable.

Question 2: Let A be an arbitrary language that is enumerable, but not decidable. Recall what it means to enumerable: There exists a Turing machine M, such that for any input string w:

- If $w \in A$, then, on input w, M terminates in the accept state.
- If $w \notin A$, then, on input w, M either terminates in the reject state or does not terminate.

Consider the following function $f : \{0,1\}^* \to \mathbb{N}$:

$$f(w) = \begin{cases} \text{the number of steps made by } M \text{ on input } w, \text{ if } M \text{ terminates on } w, \\ 0, \text{ if } M \text{ does not terminate on } w. \end{cases}$$

In this question, you will prove that the function f is not computable, i.e., there does not exist an algorithm that, for any input string $w \in \{0,1\}^*$, terminates and returns the value of $f(w)$.

(2.1) Let $g : \{0,1\}^* \to \mathbb{N}$ be an arbitrary computable function. Prove that there exists a string w in $\{0,1\}^*$ such that $f(w) > g(w)$.
Hint: As you can expect, the proof is by contradiction. Thus, you assume that the claim is not true. Define a new Turing machine N that, for any input string w in $\{0,1\}^*$, runs the Turing machine M for $g(w)$ steps and then "does something".
(2.2) Prove that the function f is not computable.

Solution: We start with the first part. We take an arbitrary computable function $g : \{0,1\}^* \to \mathbb{N}$ and assume that for every string w in $\{0,1\}^*$, $f(w) \leq g(w)$.

Consider the following algorithm N, which takes as input an arbitrary string w in $\{0,1\}^*$:

Step 1: Compute $g(w)$ and store the value in the variable k. (Note: Since g is computable, there exists an algorithm that computes $g(w)$.)

Step 2: Run the Turing machine M on input w and stop as soon as M terminates or M has made k steps.

Step 3:

- If M terminates in the accept state within k steps, then N terminates and accepts the string w.

- Otherwise, M did not terminate in the accept state after k steps. In this case, N terminates and rejects the string w.

Let us see what is going on here:

- Assume that $w \in A$. We know that, on input w, M terminates in the accept state. By the definition of f, the number of steps made by M is equal to $f(w)$. By our assumption, $f(w) \leq g(w) = k$. Thus, M terminates in the accept state within k steps. From Step 3, N terminates and accepts the string w.

- Now assume that $w \notin A$. Then M does not accept w. In particular, M does not accept w within k steps. From Step 3, N terminates and rejects the string w.

The two items above imply that algorithm N decides the language A. This is a contradiction, because A is undecidable. We conclude that there exists a string w in $\{0,1\}^*$ such that $f(w) > g(w)$.

Now the second part. We assume that the function f is computable. In the first part, we take $g = f$. Then we know that there exists a string w in $\{0,1\}^*$ such that $f(w) > f(w)$. This is a contradiction. Thus, f is not computable.

Question 3: We have seen in class that the language

$$\mathit{Halt} = \{\langle M, w\rangle : M \text{ is a Turing machine that terminates on the input string } w\}$$

is undecidable. Consider the language

$$\mathit{Halt}_\varepsilon = \{\langle M\rangle : M \text{ is a Turing machine that terminates on the input string } \varepsilon\}.$$

Professor Justin Bieber claims that the following reasoning proves that $\mathit{Halt}_\varepsilon$ is undecidable:

- We know that Halt is undecidable.
- Since $\mathit{Halt}_\varepsilon$ is a subproblem of Halt, $\mathit{Halt}_\varepsilon$ is also undecidable.

Is Professor Bieber's reasoning correct?

Solution: Sorry Justin, your reasoning does not make sense.

It is true that $\mathit{Halt}_\varepsilon$ is a subproblem of Halt: If $\langle M\rangle \in \mathit{Halt}_\varepsilon$, then $\langle M, \varepsilon\rangle \in \mathit{Halt}$.

Based on this, we cannot conclude that $\mathit{Halt}_\varepsilon$ is undecidable: It may be that $\mathit{Halt}_\varepsilon$ consists of "easy" inputs to Halt.

Question 4: Consider again the languages Halt and $\mathit{Halt}_\varepsilon$ from the previous question.

Prove that $\mathit{Halt}_\varepsilon$ is undecidable.

Hint: You are not allowed to say "Oh this follows directly from Rice's Theorem". Instead, you must give a complete proof.

Solution: As you can expect, the proof is by contradiction. Thus, we assume that $\mathit{Halt}_\varepsilon$ is decidable. Then there exists a Turing machine H, such that:

- The input to H is the encoding $\langle M\rangle$ of a Turing machine M.
- If $\langle M\rangle \in \mathit{Halt}_\varepsilon$, i.e., M terminates on the input string ε, then H terminates and outputs YES.
- If $\langle M\rangle \notin \mathit{Halt}_\varepsilon$, i.e., M does not terminate on the input string ε, then H terminates and outputs NO.
- H terminates on every input.

We are going to prove that Halt is decidable. This will be a contradiction.

For a fixed Turing machine M and a fixed string w, we define the Turing machine T_{Mw}:

- The input to T_{Mw} is the empty string ε.
- T_{Mw} writes the string w on the input tape.
- T_{Mw} runs the computation of M on input w.

Note that T_{Mw} terminates on the input string ε if and only if M terminates on the input string w.

We define the following Turing machine Q:

- The input to Q is the encoding $\langle M, w \rangle$ of a Turing machine M and a string w.
- Q constructs the Turing machine T_{Mw}.
- Q runs H on the input $\langle T_{Mw} \rangle$.
- If H terminates and returns YES, then Q terminates and returns YES.
- If H terminates and returns NO, then Q terminates and returns NO.

We claim that Q decides *Halt*.

- Since H terminates on every input, Q terminates on every input.
- Assume that $\langle M, w \rangle \in \mathit{Halt}$, i.e., M terminates on input w. We have seen above that T_{Mw} terminates on input ε. Thus, H returns YES on input $\langle T_{Mw} \rangle$. Thus, Q returns YES on input $\langle M, w \rangle$.
- Assume that $\langle M, w \rangle \notin \mathit{Halt}$, i.e., M does not terminate on input w. We have seen above that T_{Mw} does not terminate on input ε. Thus, H returns NO on input $\langle T_{Mw} \rangle$. Thus, Q returns NO on input $\langle M, w \rangle$.

Question 5: In class, we have seen that the language

$$\mathit{Halt} = \{\langle P, w \rangle : \ P \text{ is a Java program that terminates on the binary input string } w\}$$

is undecidable.

A Java program P is called a *Hello-World-program*, if the following is true: When given the empty string ϵ as input, P can do whatever it wants, as long as it outputs the string `Hello World` and terminates. (We do not care what P does when the input string is non-empty.)

Consider the language

$$HW = \{\langle P \rangle : \ P \text{ is a Hello-World-program}\}.$$

The questions below will lead you through a proof of the claim that the language HW is undecidable.

(5.1) Consider a fixed Java program P and a fixed binary string w.

We write a new Java program J_{Pw} which takes as input an arbitrary binary string x. On such an input x, the Java program J_{Pw} does the following:

Algorithm $J_{Pw}(x)$:
run P on the input w;
print `Hello World`

- Argue that P terminates on input w if and only if $\langle J_{Pw} \rangle \in HW$.

Solution:

- Assume that P terminates on input w.

  Let x be an arbitrary input string for J_{Pw}. We go through the pseudocode for J_{Pw} and see what happens: First, we run P on the input w. Because of our assumption, this part of the pseudocode terminates. Then, in the next line, `Hello World` is printed and J_{Pw} terminates.

  Thus, for any x, the computation of $J_{Pw}(x)$ prints `Hello World` and terminates. In particular, this is true for $x = \epsilon$. It follows that J_{Pw} is a Hello-World-program and, thus, $\langle J_{Pw} \rangle \in HW$.

- Assume that P does not terminate on input w.

  Let x be an arbitrary input string for J_{Pw}. We go through the pseudocode for J_{Pw} and see what happens: First, we run P on the input w. Because of our assumption, this part of the pseudocode does not terminate. As a result, J_{Pw} does not terminate.

  Thus, for any x, the computation of $J_{Pw}(x)$ does not terminate. In particular, this is true for $x = \epsilon$. It follows that J_{Pw} is not a Hello-World-program and, thus, $\langle J_{Pw} \rangle \notin HW$.

(5.2) The goal is to prove that the language HW is undecidable. We will prove this by contradiction. Thus, we assume that H is a Java program that decides HW. Recall what this means:

- If P is a Hello-World-program, then H, when given $\langle P \rangle$ as input, will terminate in the accept state.

- If P is not a Hello-World-program, then H, when given $\langle P \rangle$ as input, will terminate in the reject state.

We write a new Java program H' which takes as input the binary encoding $\langle P, w \rangle$ of an arbitrary Java program P and an arbitrary binary string w. On such an input $\langle P, w \rangle$, the Java program H' does the following:

> **Algorithm** $H'(\langle P, w \rangle)$:
> construct the Java program J_{Pw} described above;
> run H on the input $\langle J_{Pw} \rangle$;
> **if** H terminates in the accept state
> **then** terminate in the accept state
> **else** terminate in the reject state
> **endif**

Argue that the following are true:

- For any input $\langle P, w\rangle$, H' terminates.

  Solution: This follows from the fact that H terminates on any input.

- If P terminates on input w, then H' (when given $\langle P, w\rangle$ as input), terminates in the accept state.

  Solution: We assume that P terminates on input w. We know from the first part of the question that $\langle J_{Pw}\rangle \in HW$. Since H decides the language HW, it follows that, on input $\langle J_{Pw}\rangle$, H terminates in the accept state. It then follows from the pseudocode for H' that this program, on input $\langle P, w\rangle$, terminates in the accept state.

- If P does not terminate on input w, then H' (when given $\langle P, w\rangle$ as input), terminates in the reject state.

  Solution: We assume that P does not terminate on input w. We know from the first part of the question that $\langle J_{Pw}\rangle \notin HW$. Since H decides the language HW, it follows that, on input $\langle J_{Pw}\rangle$, H terminates in the reject state. It then follows from the pseudocode for H', together with the fact that H terminates on any input, that H', on input $\langle P, w\rangle$, terminates in the reject state.

(5.3) Now finish the proof by arguing that the language HW is undecidable.

Solution: Above, we have assumed that HW is decidable. Based on this assumption, we have constructed a Java program H' that has the following property:

- H' terminates on any input string $\langle P, w\rangle$.
- H' accepts the input string $\langle P, w\rangle$ if and only if P terminates on the input string w.
- This means: H' accepts $\langle P, w\rangle$ if and only if $\langle P, w\rangle \in \mathit{Halt}$.
- But, by definition, this means that the language *Halt* is decidable.
- However, *Halt* is undecidable. Therefore, HW is undecidable.

Question 6: Consider the two languages

$$\mathit{Empty} = \{\langle M\rangle : \ M \text{ is a Turing machine for which } L(M) = \emptyset\}$$

and

$$\begin{aligned}\mathit{UselessState} = \{\langle M, q\rangle: \quad & M \text{ is a Turing machine, } q \text{ is a state of } M, \\ & \text{for every input string } w, \text{ the computation of } M \text{ on} \\ & \text{input } w \text{ never visits state } q\}.\end{aligned}$$

(6.1) Use Rice's Theorem to show that *Empty* is undecidable.

(6.2) Use **(6.1)** to show that *UselessState* is undecidable.

Solution: For **(6.1)**, we verify the three conditions in Rice's theorem:

- Let M be the Turing machine that does the following: In the start state, and no matter which symbol is read, M switches to the reject state. This Turing machine rejects every input string and, therefore, $L(M) = \emptyset$. This implies that $\langle M \rangle \in \mathit{Empty}$. Thus, there exists a Turing machine M such that $\langle M \rangle \in \mathit{Empty}$.
- In class, we have seen several Turing machines N for which $L(N) \neq \emptyset$. Thus, there exists a Turing machine N such that $\langle N \rangle \notin \mathit{Empty}$.
- It is obvious that for any two Turing machines M_1 and M_2 with $L(M_1) = L(M_2)$, either both $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in *Empty* or none of them is in *Empty*. (In other words, whether or not $\langle M \rangle$ is in *Empty* only depends on the language of M.)

Since all three conditions in Rice's theorem are satisfied, it follows that *Empty* is undecidable.

Next we do **(6.2)**. That is, we will use **(6.1)** to show that *UselessState* is undecidable.

Let M be a Turing machine and let q be a state of M. We say that q is a *useless state* if for every input string w, the computation of M on input w never visits state q.

Here is the main observation: Let q_{accept} be the accept state of the Turing machine M. Then

$$L(M) = \emptyset \text{ if and only } q_{accept} \text{ is a useless state.}$$

We assume that the language *UselessState* is decidable. Let H be a Turing machine that decides *UselessState*. Consider the following algorithm H', which takes as input the binary encoding $\langle M \rangle$ of a Turing machine M:

Algorithm $H'(\langle M \rangle)$:
let q_{accept} be the accept state of M;
run H on the input $\langle M, q_{accept} \rangle$;
if H terminates in the accept state
then accept and terminate
else reject and terminate

This new algorithm H' decides the language *Empty*. This is a contradiction, because we saw in **(6.1)** that *Empty* is undecidable.