Homework 7

Response by: **TODO: replace this with your name (and computing id)**

Due: **10:00pm, Friday, April 3**

This problem set focuses on Turing Machines and (un)computability, Chapters 7–9 in TCS). Write your answers in the hw7.tex LaTeX template. You will submit your solutions in GradeScope as a PDF file with your answers to the questions in this template.

Collaboration Policy: You may discuss the problems with anyone you want. You are permitted to use any resources you find for this assignment **other than *solutions* from previous/concurrent CS3120 courses**. You must write up your own solutions and understand everything in them, and submit only your own work. You should note in the *Collaborators and Resources* box below the people you collaborated with and any external resources you used (you do not need to list resources you used for help with LaTeX).

Collaborators and Resources: TODO: replace this with your collaborators and resources (if you did not have any, replace this with *None*)

To do this assignment:

1. Open this read-only Overleaf project located at https://www.overleaf.com/read/stchqwqmnjyf#0d79fa, then select the "File" button at the top-left, and then select "Make a copy". You will have an opportunity to rename the project, and then Overleaf will create a new copy of the project which you can edit.
2. Open your copy of the project and in the left side of the browser, you should see a file directory containing hw7.tex. Click on hw7.tex to see the LaTeX source for this file, and enter your solutions in the marked places. (You will also see the uvatoc.sty file, a "style" file that defines some useful macros. You are welcome to look at this file but should not need to modify it.)
3. The first thing you should do in hw7.tex is set up your name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your name and UVA id, e.g., `\submitter{Wei-Kai Lin (twc7zv)}`.
4. Write insightful and clear answers to all of the questions. As typical people, we prefer short, precise, and comprehensive answers.
5. There are *optional* problems. There are no points, but if you wrote your solutions, we will tell you how you did.
6. Before submitting your hw7.pdf file, also remember to:
   - List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. (Remember to update this before submitting if you work with more people.)

A template proof. In Section 9.4.1 of the textbook [Bar23], it is claimed and proved that $HaltOnZero$ is uncomputable. We rewrite the proof below in a slightly shorter way. You may use it as a template for your proofs.

Let $HALT : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be the halting function, defined by: $HALT(w,x) = 1$ if Turing Machine $\mathsf{TM}_w(x)$ halts and $HALT(w,x) = 0$ otherwise. Theorem 9.6 states that $HALT$ is uncomputable.

Let $HaltOnZero : \{0,1\}^* \to \{0,1\}$ be the halt-on-zero function, which is defined as

$$HaltOnZero(w) = \begin{cases} 1 & \text{if } \mathsf{TM}_w(0) \text{ halts} \\ 0 & \text{otherwise,} \end{cases}$$

where TM_w denotes the Turing Machine represented by $w \in \{0,1\}^*$. We prove that $HaltOnZero$ is uncomputable.

Proof. Assume for contradiction, $HaltOnZero$ is computable by some Turing Machine A. We aim to construct another Turing Machine B such that it computes $HALT$, which will contradict that $HALT$ is uncomputable.

The description of B is the following.

> The algorithm $B(w,x)$ takes two inputs $w, x \in \{0,1\}^*$ and performs the following steps.
>
> 1. Prepare the following Turing Machine M'.
>
>> $M'(z)$ ignores its input z but performs the following.
>>
>> (a) Executes $U(w,x)$, where U is the universal Turing Machine.
>>
>> (b) Halt and output a fixed string, e.g., a smile `:)`.
>
> 2. Let w' be the string that represents M'. (Notice that both M' and w' depend on (w,x), and one may explicitly denote them by $M'_{w,x}$ and $w'_{w,x}$ if needed.)
>
> 3. Execute and output $A(w')$.

We claim that for all $w, x \in \{0,1\}^*$, it holds that $HALT(w,x) = HaltOnZero(w')$, where w' is constructed as per Step 2 of B. If so, we have $B(w,x) = A(w')$ by construction of B, and $A(w') = HaltOnZero(w')$ by the assumption of A, and $HALT(w,x) = HaltOnZero(w')$, and thus $B(w,x)$ computes $HALT(w,x)$, which is the desired contradiction.

It remains to prove the claim, $HALT(w,x) = HaltOnZero(w')$. For any $w, x \in \{0,1\}^*$, suppose that $HALT(w,x) = 1$. By definition of $HALT(w,x)$, $\mathsf{TM}_w(x)$ halts, and thus $U(w,x)$ halts at Step 1a by definition of universal Turing Machine U. That implies $M'(z)$ halts on $z = 0$, which implies $HaltOnZero(w') = 1$ by construction of w' and definition of $HaltOnZero(w')$. □

Problem 1 (8pt)**.** *Configurations*

A machine *configuration* contains all of the information needed to completely represent a snapshot (we're avoiding using "state" here because of the confusion with the internal machine state, which does not fully capture the execution state of a Turing machine) of an execution. Essentially, a configuration should have all of the necessary details to be able to pause and save an execution so that it can be resumed later.

As an example of this idea, operating systems must be able to occasionally pause computer programs temporarily in order to use the CPU for another process. When doing this, the operating system will store a configuration of that program (sometimes called an *image*) so it can perfectly pick up from where it left off. For a Python program, this configuration would most likely include the values of all variables, the contents of all data structures, the current call stack (which functions have invoked which other functions), and the program counter that identifies the next instruction to execute.

Answer the following sub-problems regarding machine configurations:

(a) List all of the things that should be included in the configuration of a nondeterministic finite state automaton.

(b) List all of the things that should be included in the configuration of a Turing Machine (select any definition of a TM you would like).

(c) One necessary reason that Turing Machines are more powerful than finite state automata is that a particular machine could have an infinite number of possible configurations. Identify what component(s) of your Turing Machine configuration would allow for an infinite number of configurations.

Answer:

(a)

(b)

(c)

Problem 2 (10pt)**.** *Halts in k steps*

Consider the function:

$$H_k(w) = \begin{cases} 1 & \mathsf{TM}_w \text{ is a Turing machine that halts in } k \text{ or fewer steps when it receives no input} \\ 0 & \text{otherwise.} \end{cases}$$

Show that H_k is computable for every choice of $k \in \mathbb{N}$.

Answer:

Problem 3 (12pt)**.** *Palindrome vs Turing Machines*

Let $F : \{0,1\}^* \to \{0,1\}$ be the Boolean function such that $F(x) = 1$ if x is a palindrome and $F(x) = 0$ otherwise.
Let $G : \{0,1\}^* \to \{0,1\}$ be the Boolean function such that $G(w) = 1$ if TM_w computes F and $G(w) = 0$ otherwise.

Is G computable? Prove that G is uncomputable. Hint: You may use Rice's Theorem, or show a reduction from $HALT$, similar to $HaltOnZero$.

Answer:

Problem 4 (Optional)**.** *Regular vs Turing Machine*

Let G be a function defined similarly to the definition of G in Problem 3, but the only difference is that F is a regular function (not the palindrome function).

Is G still uncomputable? Prove it. Try to use a reduction if you applied Rice's Theorem in the previous problem, and vise versa.

Answer:

Do not write anything on this page; leave this page empty.

This is the end of the problems for HW7. Remember to follow the last step in the directions on the first page to prepare your PDF for submission.

References

[Bar23] Boaz Barak. *Introduction to Theoretical Computer Science*. https://introtcs.org/public/. 2023.

