Problem Set 9

Due: **10:00pm, Monday, April 14**

This problem set focuses on the computability of functions, or equivalently, languages. The Halting Problem plays a central role. You are also expected to prove through reduction.

Write your answers in the ps9.tex LaTeX template. You will submit your solutions in GradeScope as a PDF file with your answers to the questions in this template. There are four "required" problems and one "bonus" practice, where the bonus gives extra points.

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 should 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 shall explicitly state the *content*, e.g., the main message in your collaborated discussion, the search keywords, the LLM/AI prompts, or the section in a book.

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/xgyzkjpzgcfj#d01097, and then copy this project.
2. Open your copy of the project and in the left side of the browser, you should see a file directory containing ps9.tex. Click on ps9.tex to see the LaTeX source for this file, and enter your solutions in the marked places.
3. The first thing you should do in ps9.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{Haolin Liu (srs8rh)}`.
4. Write insightful and clear answers to all of the questions in the marked spaces provided.
5. Before submitting your 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.)
   - Replace the second line in ps9.tex, `\usepackage{uvatoc}` with `\usepackage[response]{uvatoc}` so the directions do not appear in your final PDF. Starting from this Problem Set, we may deduct 3pt if you forgot this step.

Definition 1 (Computable functions) *A boolean function $f : \{0,1\}^* \to \{0,1\}$ is called computable if there exists a Turing machine M such that $M(x) = f(x)$ for all $x \in \{0,1\}^*$. The definition extends to functions $f : \{0,1\}^* \to \mathbb{N}$.*

Definition 2 (Universal Turing machine) *Consider an arbitrary binary representation of Turing machines (e.g., encoding the transition function in bits in a standard way). For any string $w \in \{0,1\}^*$, if w is the binary representation of a Turing machine, let TM_w be the Turing machine; otherwise, let TM_w be $\emptyset$. A Turing machine U is a* universal Turing machine *iff for any $w, x \in \{0,1\}^*$, it holds that*

$$U(w,x) = \begin{cases} \mathsf{TM}_w(x) & \text{if } \mathsf{TM}_w \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Problem 1 (6pt) *A reduction from* SELFREJECTS *to* ACCEPTS.

Recall that in Class 19, we defined ACCEPTS : $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ to be the following boolean function, and we proved that ACCEPTS is uncomputable.

$$\text{ACCEPTS}(w,x) = \begin{cases} 1 & \text{if } \mathsf{TM}_w \neq \emptyset \text{ and } \mathsf{TM}_w(x) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

We want to prove that ACCEPTS is uncomputable *in another way*. Hence, we define another boolean function SELFREJECTS below.

$$\text{SELFREJECTS}(w) = \begin{cases} 1 & \text{if } \mathsf{TM}_w \neq \emptyset \text{ and } \mathsf{TM}_w(w) = 0 \\ 0 & \text{if } \mathsf{TM}_w \neq \emptyset \text{ and } \mathsf{TM}_w(w) \neq 0 \\ 1 & \text{otherwise, } \mathsf{TM}_w = \emptyset \end{cases}$$

Intuitively, for any string $w \in \{0,1\}^*$, the function SELFREJECTS outputs 1 if and only if the Turing machine TM_w rejects the string w. Clearly, there are many self-rejecting Turing machines, e.g., some Turing machines reject all inputs. Suppose that we can prove that SELFREJECTS is uncomputable (it is a good exercise). Given that, we want a *reduction* from SELFREJECTS to ACCEPTS. That is, to show that if we can compute ACCEPTS, then we can compute SELFREJECTS. This contradicts that SELFREJECTS is uncomputable.

The following is a (potentially wrong) reduction. State whether it is a "correct" or "wrong" proof. If it is wrong, point to what is wrong clearly.

1. For contradiction, assume that ACCEPTS is computable.
2. By the definition of computable, there exists a Turing machine M_A that computes ACCEPTS.
3. By definition of binary representation, there exists a string $w_A \in \{0,1\}^*$ such that $M_A = \mathsf{TM}_{w_A}$.
4. Define a new Turing machine M_D to be
$$M_D(x) = NOT(U(w_A, (x,x))).$$
5. Because $U(w_A, (x,x)) = \mathsf{TM}_{w_A}(x,x) = M_A(x,x) = \text{ACCEPTS}(x,x)$, we have $M_D(x)$ computes $NOT(\text{ACCEPTS}(x,x))$.
6. The function $NOT(\text{ACCEPTS}(x,x))$ is identical to $\text{SELFREJECTS}(x)$.
7. $M_D(x)$ computes $\text{SELFREJECTS}(x)$.
8. We know $\text{SELFREJECTS}(x)$ is uncomputable, a contradiction.

Answer:

Problem 2 (7pt) *Prove that the Busy Beaver Problem (defined in class and below) is uncomputable.*

Definition 3 (Busy Beaver Problem) *For any $n \in \mathbb{N}$, define $BB_2(n)$ as the maximum number of steps for which a Turing Machine with n states and 2 symbols can execute and halt, starting from a blank tape.*

The Busy Beaver Problem is defined above. Prove the function $BB_2(n)$ is uncomputable. Your proof could show that assuming a Turing machine that can solve the Busy Beaver problem (that is, it can output the correct value of $BB_2(n)$ for any input $n \in \mathbb{N}$) leads to a contradiction (that it, it would allow for a TM that can compute some function we know is uncomputable).

Answer:

Problem 3 (6pt) *Halts in k steps*

Consider the language:

$$H_k = \{w \in \{0,1\}^* | \mathsf{TM}_w \text{ is a Turing machine which halts in } k \text{ or fewer steps when it receives no input}\}$$

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

You shall use the definition of *computable* languages, given in Definitions 5 and 6.

Answer:

Problem 4 (6pt) *Prove that H_* (defined below) is not computable.*

Define the language $H_* = \bigcup_{k \in \mathbb{N}} H_k$.

Answer:

Practice 1 (Bonus, 3pt) *Prove that the Busy Boa Problem (defined below) is uncomputable.*

Define the the Busy Boa Problem as:

Definition 4 (Busy Boa Problem) *For any $n \in \mathbb{N}$, define $BOA(n)$ as the largest integer that an idealized Python function implemented using at most n characters, and which takes no input, can return (and halt).*

The *idealized Python* language is the Python language you are familiar with, but without any arbitrary limits. So, for example, it provides a + operation that works on all natural numbers, unlike any actual Python implementation which can only implement + for a subset of $\mathbb{N}$. The Python function may use any "built-in" library provided by a standard Python interpreter. By *no input*, the Python function gets no input of any format, such as parameters, read from any I/O devices in the system, and "import" an arbitrary package.

Answer:

Practice 2 (Bonus, 3pt) *Computability is closed under union.*

Show that if the language L_1 is computable (i.e. there is some Turing Machine M_1 such that $L_1 = \mathcal{L}(M_1)$), and language L_2 is computable (i.e. there is some Turing Machine M_2 such that $L_2 = \mathcal{L}(M_2)$), then the language $L_1 \cup L_2$ is also computable.

You should use the following definitions of computable languages:

Definition 5 (Language of a Turing Machine) *We say that the language of a Turing Machine M, denoted $\mathcal{L}(M)$, is the set of all input strings which M accepts (i.e. halts and returns 1).*

Definition 6 (Computable Language) *We say that a language $\mathcal{L}_c$ is computable is there is some always-halting Turing Machine M such that $\mathcal{L}_c = \mathcal{L}(M)$.*

Answer:

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

This is the end of the problems for PS9. Remember to follow the last step in the directions on the first page to prepare your PDF for submission.

