CS 208 : Automata Theory and Logic Spring 2024

Homework 4

Due: 20th Apr, 2024 *Max Marks: 30*

Instructions:

- *Please start writing your* ***solution to each homework problem on a fresh page.***
- *For each homework problem, you must scan your solution and upload a separate PDF file on Moodle. Please check Moodle for detailed instructions on file naming and uploading instructions.*
- *Be brief, complete, and stick to what has been asked.*
- *Untidy presentation of answers, and random ramblings will be penalized by negative marks.*
- *Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.*
- ***If you need to make any assumptions, state them clearly.***
- ***Do not copy solutions from others. All detected cases of copying will be reported to DADAC with names and roll nos. of all involved. The stakes are high if you get reported to DADAC, so you are strongly advised not to risk this.***

---

1. Turing machines and natural numbers 10 points

We have studied in class that every natural number i can be thought of as encoding a Turing machine, say M_i. Similarly, every natural number j can be thought of as encoding a string, say w_j, over an alphabet Σ.

Choose ***any one*** of the languages L_i ($i \in \{1, 2\}$) defined below and determine whether L_i is recursive. If your answer is "Yes", you must describe how to construct a halting Turing machine (i.e. halts on all inputs) that decides L_i. Otherwise, you must give a proof why L_i is not recursive (i.e. undecidable). Answers without a Turing machine or proof will fetch no marks.

Choose any one of the languages below to answer this question. Indicate clearly in your answer which language you are choosing.

(a) $L_1 = \{n \in \mathbb{N} \mid \exists m \in \mathbb{N} \text{ s.t. } M_n \text{ halts on } w_m\}$.

(b) $L_2 = \{m \in \mathbb{N} \mid \exists \text{ infinitely many } n \in \mathbb{N} \text{ s.t. } M_n \text{ halts on } w_m\}$.

2. Post's Correspondence Problem (PCP) and grammars 10 points

We've seen that the halting problem for Turing machines is undecidable. Another very well-known undecidable problem is *Post's Correspondence Problem, also popularly call PCP*. An instance of PCP consists of two finite (ordered) lists, say A and B, of strings over an alphabet Σ, such that the lists are of equal length. Let the lists be $A = (w_1, w_2, \ldots w_k)$ and $B = (v_1, v_2, \ldots v_k)$, where each $w_i, v_i \in \Sigma^*$. For each $i \in \{1, \ldots k\}$, the strings w_i and v_i are called *corresponding strings*. A solution to the PCP instance is a finite sequence of integers $(i_1, i_2, \ldots i_m)$ such that the concatenated strings $w_{i_1} w_{i_2} \cdots w_{i_m}$ and $v_{i_1} v_{i_2} \cdots v_{i_m}$ are identical.

As an example, suppose $\Sigma = \{0, 1\}$ and let $A = (01, 10, 101, 101)$, $B = (0101, 101, 1, 01)$. In other words, $w_1 = 01, w_2 = 10, w_3 = w_4 = 101$, while $v_1 = 0101, v_2 = 101, v_3 = 1, v_4 = 01$. One solution to this instance of PCP is $(3, 1, 2, 4)$, since $w_3 w_1 w_2 w_4 = 1010110101 = v_3 v_1 v_2 v_4$.

On the other hand, if $A = (011, 1101, 110, 111)$ and $B = (001, 001, 00, 010)$ you can convince yourself that this instance of PCP doesn't have a solution, since every w_i has more 1s than the corresponding v_i.

The decision version of PCP can be stated as follows: *Given an instance of PCP, does it have a solution?*

It is known that ***PCP is undecidable***. In other words, there does not exist any halting Turing machine that takes as input an instance of PCP, and halts in a designated "Yes" state if the instance has a solution, and halts in a designated "No" state otherwise. For those interested, you can look up Section 9.4 of Hopcroft, Motwani and Ullman's book for a detailed proof of this result. In this problem, we won't use the details of the proof. Instead, we will simply appeal to the undecidability of PCP.

Show by using a reduction from PCP that the following problem is undecidable:

Given a context-free grammar G, does there exist a terminal string $w \in L(G)$ such that $w^R \in L(G)$ as well, where w^R denotes the string w reversed?

You must explain clearly how having a Turing machine that decides the above problem about CFGs would enable you to construct a Turing machine that decides PCP.

Answers without explanations will fetch no marks.

[Hint: Take an arbitrary instance of PCP and show how to construct a CFG G out of it such that both w and w^R are in $L(G)$ iff the PCP instance has a solution.]

3. Co-recursively enumerable languages 10 points

A language $L \subseteq \Sigma^*$ is said to be *co-recursively enumerable* if its complement (i.e. $\Sigma^* \setminus L$) is recursively enumerable. We have seen examples of co-recursively enumerable languages in class. For example, the diagonalization language L_d is co-recursively enumerable.

Let $\mathcal{F} = \{L_i \mid L_i \subseteq \Sigma^*,\ i \in \mathbb{N}\}$ be an (infinite) family of languages over an alphabet Σ such that

- For every $i \in \mathbb{N}$, L_i is not recursively enumerable
- For every $i \in \mathbb{N}$, $L_i \subseteq \Sigma^*$ is co-recursively enumerable.
- For every i, j s.t. $i \neq j$, $L_i \not\subseteq L_j$.

Answer ***any one*** of the following questions. Indicate clearly in your answer which question you are choosing to answer.

(a) Prove that the (infinite) intersection of all languages in $\mathcal{F}$, i.e. $\{w \mid \forall i \in \mathbb{N},\, w \in L_i\}$, is co-recursively enumerable.

(b) Give an example of $\mathcal{F}$ such that the (infinite) union of all languages in $\mathcal{F}$, i.e. $\{w \mid \exists i \in \mathbb{N},\, w \in L_i\}$, is not recursively enumerable but is co-recursively enumerable.
You must clearly state what the language L_i is for each $i \in \mathbb{N}$, show that these languages satisfy the three properties listed above, and prove that the infinite union is not recursively enumerable, although its complement is recursively enumerable.

Answers without proofs will not fetch any marks.