Problem Set 8

Due Date ..... November 1, 2022
Name ..... **Alex Ojemann**
Student ID **109722375**
Collaborators ..... **None**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. (See this short intro to LaTeX plus other resources on Canvas.)
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

(I agree to the above, Alex Ojemann). □

3 Standard 21: Computability Reductions

Problem 1. Complete the Canvas quiz. There is nothing you need to submit with the PDF.

4 Standard 20: Turing Machine Constructions

Problem 2. Consider the class of recursive (decidable) languages, which we denote REC. For each of the following closure properties, either (i) give a construction to show that REC is closed under said property, or (ii) give a counter-example to show that REC is *not* closed under said property.

In the case of (i), you do not have to prove that your construction works. In the case of (ii), you do not have to write a formal proof that your counter-example is correct, but you should write a couple of sentences carefully explaining such.

(a) Let $L_1, L_2 \in \mathsf{REC}$. Is $L_1 \cup L_2 \in \mathsf{REC}$?

Answer. Let M be the two tape Turing machine representing the union of L_1 and L_2 where tapes 1 and 2 are the Turing machines that accept on strings in L_1 and L_2 respectively and reject on strings not in L_1 and L_2 respectively.

M will accept when one of the tapes tapes accepts, and will reject when both tapes reject. Since each of the tapes represent recursive languages and M, in its worst case, will halt when both tapes are halted, the language accepted my M ($L_1 \cup L_2$) must also be recursive. □

(b) Let $L_1, L_2 \in \mathsf{REC}$. Is $L_1 \cap L_2 \in \mathsf{REC}$?

Answer. Let M be the two tape Turing machine representing the intersection of L_1 and L_2 where tapes 1 and 2 are the Turing machines that accept on strings in L_1 and L_2 respectively and reject on strings not in L_1 and L_2 respectively.

M will accept when both tapes accept, and will reject when one of the tapes rejects. Since each of the tapes represent recursive languages and M, in its worst case, will halt when both tapes are halted, the language accepted my M ($L_1 \cap L_2$) must also be recursive. □

(c) Let $L \in \mathsf{REC}$. Is the complement of L, $\sim L \in \mathsf{REC}$?

Answer. Let M be the Turing machine that accepts on strings in L and rejects on strings not in L. We can simply switch the accept and reject states of M, yielding a Total Turing Machine M' such that L(M') = $\sim L$.

Since the accept and reject states of M' are the same as those of M just reversed, L(M') = $\sim L$ must also be recursive. □

Problem 3. Consider the class of recursively enumerable (Turing recognizable) languages, which are denoted *r.e.* (Kozen) or RE (Complexity Theory). For each of the following closure properties, either (i) give a construction to show that RE is closed under said property, or (ii) give a counter-example to show that RE is *not* closed under said property.

In the case of (i), you do not have to prove that your construction works. In the case of (ii), you do not have to write a formal proof that your counter-example is correct, but you should write a couple of sentences carefully explaining such.

(a) Let $L_1, L_2 \in \mathsf{RE}$. Is $L_1 \cup L_2 \in \mathsf{RE}$?

Answer. Let M be the two tape Turing machine representing the union of L_1 and L_2 where tapes 1 and 2 are the Turing machines that accept on strings in L_1 and L_2 respectively.

M will accept when one of the tapes tapes accepts. Since each of the tapes represent r.e. languages and M will accept when one of the tapes is accepted, the language accepted my M ($L_1 \cup L_2$) must also be r.e. □

(b) Let $L_1, L_2 \in \mathsf{RE}$. Is $L_1 \cap L_2 \in \mathsf{RE}$?

Answer. Let M be the two tape Turing machine representing the intersection of L_1 and L_2 where tapes 1 and 2 are the Turing machines that accept on strings in L_1 and L_2 respectively.

M will accept when both tapes accept. Since each of the tapes represent r.e. languages and M will accept when both tapes are accepted, the language accepted my M ($L_1 \cap L_2$) must also be r.e. □

(c) Let $L \in \mathsf{RE}$. Is the complement of L, $\sim L \in \mathsf{RE}$?

Answer. A counterexample that shows this is not true the Halting Problem. The Halting Problem is r.e. because a Turing machine exists that accepts for any program that halts. However, its complement is not r.e. because there can't be a Turing machine that accepts for any program that doesn't halt because it will loop forever. □

5 Standards 2/3: Proofs

Problem 4. Prove that a recursively enumerable language L is recursive (decidable) if and only if there exists an enumeration machine that enumerates L in increasing order.

Proof. If L is recursive, the enumerator operates by generating the strings in increasing order and testing each in turn for membership in L using the decider. Those strings which are found to be in L are printed.
If L is enumerable in increasing order, there are two cases. If L is finite, it is recursive because all finite languages are recursive. If L is infinite, a decider for L operates as follows. On receiving input x, the decider enumerates all strings in L in order until some string after x appears, which must occur eventually because L is infinite. If x has appeared already then it accepts, but if it hasn't appeared yet, it never will because the machine has already passed the position where it would be, so it rejects. □