Problem Set 8

What lies beyond what can be solved by a computer? What intuitions can we build about those sorts of problems? In this problem set, you will learn how to reason about the unsolvable.

Start this problem set early. It contains five problems (plus one survey question and one extra credit problems), some of which require a fair amount of thought. I would suggest reading through this problem set at least once as soon as you get it to get a sense of what it covers.

As much as you possibly can, please try to work on this problem set individually. That said, if you do work with others, please be sure to cite who you are working with and on what problems. For more details, see the section on the honor code in the course information handout.

In any question that asks for a proof, you **must** provide a rigorous mathematical proof. You cannot draw a picture or argue by intuition. You should, at the very least, state what type of proof you are using, and (if proceeding by contradiction, contrapositive, or induction) state exactly what it is that you are trying to show. If we specify that a proof must be done a certain way, you must use that particular proof technique; otherwise you may prove the result however you wish.

As always, please feel free to drop by office hours or send us emails if you have any questions. We'd be happy to help out.

This problem set has 125 possible points. It is weighted at 7% of your total grade. The earlier questions serve as a warm-up for the later problems, so be aware that the difficulty increases over the course of this problem set.

Good luck, and have fun!

Due Friday, June 1st at 2:15 PM

Problem One: Understanding Mapping Reductions (25 Points)

We often use mapping reductions to determine how hard one problem is by relating it to some other problem we already know about. However, we have to be careful when doing so.

i. Below is an **incorrect** proof that $REGULAR_{\text{TM}}$ (the language of TMs whose language is regular) is **RE**, even though in lecture we proved that it was neither **RE** or co-**RE**. Identify what is wrong with this proof. You should be sure that you are 100% positive what is wrong with this proof before you attempt any other problems on this problem set, because the mistake made here is extremely common!

Theorem: $REGULAR_{\text{TM}} \in$ **RE**.

Proof: We exhibit a mapping reduction from A_{TM} to $REGULAR_{\text{TM}}$; since A_{TM} is **RE**, this proves that $REGULAR_{\text{TM}}$ is **RE** as well. Given a TM/string pair $\langle M, w\rangle$, let the function $f(\langle M, w\rangle) = \langle M'\rangle$, where M' is the TM defined in terms of M and w as follows:

```
M' = “On input x:
        If x has the form 0ⁿ1ⁿ for some n ∈ ℕ, accept.
        Run M on w.
        If M accepts w, accept.
        If M rejects w, reject.”
```

By the parameterization theorem, f is computable. We claim, moreover, that $\langle M, w\rangle \in A_{\text{TM}}$ iff $f(\langle M, w\rangle) \in REGULAR_{\text{TM}}$. To see this, note that $f(\langle M, w\rangle) = \langle M'\rangle \in REGULAR_{\text{TM}}$ iff $\mathscr{L}(M')$ is regular. We claim that $\mathscr{L}(M') = \Sigma^*$ if M accepts w and otherwise is $\{ \texttt{0}^n\texttt{1}^n \mid n \in \mathbb{N} \}$; from this, we have that $\mathscr{L}(M')$ is regular iff M accepts w. To see that this is true, note that if M accepts w, then M' accepts all strings, either because they are immediately accepted because they have the form $\texttt{0}^n\texttt{1}^n$, or because they are accepted when M accepts w. Thus $\mathscr{L}(M') = \Sigma^*$. Otherwise, if M does not accept w, then M' accepts x iff x has the form $\texttt{0}^n\texttt{1}^n$. Consequently, M' accepts x iff $x \in \{ \texttt{0}^n\texttt{1}^n \mid n \in \mathbb{N} \}$, so $\mathscr{L}(M') = \{ \texttt{0}^n\texttt{1}^n \mid n \in \mathbb{N} \}$, as required.

We thus have that $\mathscr{L}(M')$ is regular iff M accepts w iff $\langle M, w\rangle \in A_{\text{TM}}$. Thus $\langle M, w\rangle \in A_{\text{TM}}$ iff $f(\langle M, w\rangle) \in REGULAR_{\text{TM}}$. Therefore, f is a mapping reduction from A_{TM} to $REGULAR_{\text{TM}}$, so $A_{\text{TM}} \leq_{\text{M}} REGULAR_{\text{TM}}$. Consequently, $REGULAR_{\text{TM}} \in$ **RE**. ■

We've used mapping reductions to relate the difficulty of **R**, **RE**, and co-**RE** languages to one another. Could we use them to relate problems from classes of languages as well? Usually, the answer is no.

ii. Find an pair of languages L_1 and L_2 where $L_1 \leq_{\text{M}} L_2$, L_2 is regular, but L_1 is not regular, then prove that this is the case by exhibiting a mapping reduction from L_1 to L_2. This means that we cannot establish that a language is regular by finding a mapping reduction from it to a known regular language.

To motivate why it is that we've defined reductions as we have, suppose that we change the definition of a mapping reduction as follows: For languages A and B, we say that $A \leq_{\text{M}} B$ iff there exists a computable function f such that for any $w \in A$, $f(w) \in B$.

iii. Prove that under this modified definition, *any* language A is mapping reducible to *any* language $B \neq \emptyset$. This (hopefully!) explains why we didn't define reductions this way.

Problem Two: Accept all the Strings! (25 Points)[*]

Consider the language

$$A_{ALL} = \{ \langle M \rangle \mid \mathscr{L}(M) = \Sigma^* \}$$

This language is neither **RE** nor co-**RE**, and in this problem you will see why.

i. Prove that $A_{TM} \le_M A_{ALL}$. Since $A_{TM} \notin$ co-**RE**, this proves that $A_{ALL} \notin$ co-**RE** either.

The trickier part of the proof is proving that A_{ALL} is not **RE**. To do this, we will reduce $\overline{A_{TM}} \le_M A_{ALL}$. Since $\overline{A_{TM}} \notin$ **RE**, this proves that $A_{ALL} \notin$ **RE** either.

Suppose that you are given a Turing machine *M* and a string *w*. We can construct a new TM *M'* (using the parameterization theorem) as follows:

```
M' = "On input x:
        Run M on w for |x| steps.
        If M accepted within |x| steps, reject.
        Otherwise, accept."
```

For example, on input `000`, *M'* would run *M* on *w* for 3 steps, rejecting if *M* accepted *w* within that time and accepting otherwise. Similarly, if *M'* were run on `010101`, *M'* would accept if *M* did not accept *w* within 6 steps and would reject otherwise.

ii. Prove that *M* does not accept *w* iff $\mathscr{L}(M') = \Sigma^*$.

iii. Using your answer from (ii), prove that $\overline{A_{TM}} \le_M A_{ALL}$. Since $\overline{A_{TM}}$ is not **RE**, this proves that A_{ALL} is not **RE** either.

Problem Three: The Nature of RE and co-RE (10 Points)

When we first saw the language L_D, we described it as a problem that was "harder" than A_{TM} because A_{TM} is **RE** while L_D is not. However, this might not have been a fair characterization. Both L_D and A_{TM} are hard problems, but neither one is "harder" than the other in the sense that neither one is mapping reducible to the other.

Prove that there is no mapping reduction from A_{TM} to L_D or vice-versa.

[*] Original image by Allie Brosh. This image courtesy of quickmeme.com.

Problem Four: Disjoint Unions (30 Points)

In what follows, assume that all languages are over the alphabet $\Sigma = \{ \texttt{0}, \texttt{1} \}$.

In the normal union of two languages, we simply combine all strings contained in both languages: $L_1 \cup L_2$ is the set of all strings in at least one of L_1 and L_2. When taking the union of two languages that are in known to be "hard," we can often end up constructing a language that is substantially "easier" than either of the input languages. For example, the language L_D is co-**RE** but not **RE**, and similarly the language $\overline{L_D}$ is **RE** but not co-**RE**. However, their union $L_D \cup \overline{L_D} = \Sigma^*$, which is extremely easy to decide (it's regular, and there is a one-state DFA for it!)

The reason this problem is so much easier to solve than either L_D or $\overline{L_D}$ is that to decide whether a given string w is contained in $L_D \cup \overline{L_D}$, we don't have to determine which of the two languages contains w. Any string w has to be in at least one of the languages, and therefore we can immediately say that w is contained in the union without determining whether it was in L_D or $\overline{L_D}$.

A more interesting construction is the *disjoint union* of two languages, a way of combining together strings from two different languages that tags each string with information about which language it came from. Formally, the disjoint union of two languages L_1 and L_2 is the language

$$L_1 \uplus L_2 = \{ \texttt{0}w \mid w \in L_1 \} \cup \{ \texttt{1}w \mid w \in L_2 \}$$

For example, if $L_1 = \{ \texttt{1}, \texttt{10}, \texttt{100}, \texttt{1000} \}$ and $L_2 = \{ \varepsilon, \texttt{0}, \texttt{1}, \texttt{00}, \texttt{01}, \texttt{10}, \texttt{11} \}$, then $L_1 \uplus L_2$ is the set

$$L_1 \uplus L_2 = \{ \texttt{01}, \texttt{010}, \texttt{0100}, \texttt{01000}, \texttt{1}, \texttt{10}, \texttt{11}, \texttt{100}, \texttt{101}, \texttt{110}, \texttt{111} \}$$

Notice how each string in $L_1 \uplus L_2$ is tagged with which language it originated in. Any string that starts with **0** came from L_1, and any string that starts with **1** came from L_2. Because of this tagging, the disjoint union of two languages produces a new language that is at least as hard as either of the input languages. In this problem, you will prove various important properties about the disjoint union.

i. Prove that if L_1 and L_2 are regular, then $L_1 \uplus L_2$ is regular. This shows that the regular languages are closed under disjoint union.

ii. Prove that if L_1 and L_2 are any languages, then $L_1 \leq_M L_1 \uplus L_2$. A similar proof can be used to show that $L_2 \leq_M L_1 \uplus L_2$, but you don't need to do that here.

iii. Prove that if L is recognizable but undecidable, then $L \uplus \overline{L}$ is neither **RE** nor co-**RE**.

Your result from (iii) shows how to construct extraordinarily hard problems out of problems that are already known to be hard. For example, $A_{TM} \uplus \overline{A_{TM}}$ is neither **RE** nor co-**RE**, nor is $\mathit{HALT} \uplus \overline{\mathit{HALT}}$.

Problem Five: Finding Universal Turing Machines (25 Points)

In lecture we discussed the existence of a universal Turing machine U_{TM} with the following description:

```
U_TM = "On input ⟨M, w⟩:
            Run M on w.
            If M accepts w, accept.
            If M rejects w, reject."
```

There is not just one universal Turing machine; in fact, there are infinitely many machines that are universal Turing machines. Given this fact, could we determine whether or not a particular machine was universal? Consider the language *UNIVERSAL* defined as follows:

$$UNIVERSAL = \{ \langle M \rangle \mid \mathscr{L}(M) = A_{TM} \}$$

i. Prove that $\overline{L_D} \le_M UNIVERSAL$. This proves that *UNIVERSAL* is not co-**RE**. *(Hint: Construct a deterministic TM M' such that $\mathscr{L}(M') = \emptyset$ if $\langle M \rangle \notin \mathscr{L}(M)$ and $\mathscr{L}(M') = A_{TM}$ otherwise. You may want to use the machine* U_{TM} *as a subroutine inside the machine M' that you construct.)*

ii. Prove that $L_D \le_M UNIVERSAL$. This proves that *UNIVERSAL* is not **RE**. *(Hint: Construct a* <u>*nondeterministic*</u> *TM M' such that $\mathscr{L}(M') = \Sigma^*$ if $\langle M \rangle \in \mathscr{L}(M)$ and $\mathscr{L}(M') =$* A_{TM} *otherwise. As before, you may want to use* U_{TM} *as a subroutine in M'.)*

Problem Six: Course Feedback (5 Points)

We want this course to be as good as it can be, and we'd really appreciate your feedback on how we're doing. For a free five points, please answer the following questions. We'll give you full credit no matter what you write (as long as you write something!), but we'd appreciate it if you're honest about how we're doing.

i. How hard did you find this problem set? How long did it take you to finish?

ii. Does that seem unreasonably difficult or time-consuming for a five-unit class?

iii. How is the pace of this course so far? Too slow? Too fast? Just right?

iv. Is there anything in particular we could do better? Is there anything in particular that you think we're doing well?

v. We will be holding a final exam review session and want it to be as useful as possible. Are there any topics we should specifically focus on? Any topics that you think we can skip?

Submission instructions

There are three ways to submit this assignment:

1. Hand in a physical copy of your answers at the start of class. This is probably the easiest way to submit if you are on campus.
2. Submit a physical copy of your answers in the filing cabinet in the open space near the handout hangout in the Gates building. If you haven't been there before, it's right inside the entrance labeled "Stanford Engineering Venture Fund Laboratories." There will be a clearly-labeled filing cabinet where you can submit your solutions.
3. Send an email with an electronic copy of your answers to the submission mailing list (cs103-spr1112-submissions@lists.stanford.edu) with the string "[PS8]" somewhere in the subject line. If you do submit electronically, please submit your assignment as a single PDF if at all possible. Sending multiple image files makes it much harder to print out and grade your submission.

If you are an SCPD student, we would strongly prefer that you submit solutions via email. Please contact us if this will be a problem.

Extra Credit Problem: Three-State Turing Machines (5 Points Extra Credit)

A three-state Turing machine is (as the name suggests) a Turing machine with three states, two of which are the accept and reject states. This leaves just one "work state" for the TM.

Prove that the language

$$A_{3TM} = \{ \langle M, w \rangle \mid M \text{ is a three-state TM that accepts } w \}$$

is decidable.