Problem Set 8

How powerful are Turing machines? What problems can they solve? What problems can they not solve? This problem set probes these questions in detail.

Start this problem set early. It contains seven problems (plus one survey question and one extra credit problem), several of which require a lot of thought. 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.

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 **9%** of your total grade (note that this is higher than usual). The earlier questions serve as a warm-up for the later problems, so do be aware that the difficulty of the problems does increase over the course of this problem set.

Good luck, and have fun!

Due Friday, December 2nd at 2:15 PM

Problem One: Designing Turing Machines (15 Points)

Consider the alphabet $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{≠}\}$ and the language

$$NE = \{ w\texttt{≠}x \mid w, x \in \{\texttt{0}, \texttt{1}\}^* \text{ and } w \neq x \}.$$

For example, $\texttt{0110≠011} \in NE$, $\texttt{≠01} \in NE$, but $\texttt{01≠01} \notin NE$, $\texttt{≠} \notin NE$, and $\texttt{0≠1≠} \notin NE$.

Design a Turing machine that recognizes *NE* by writing out a transition table or a state-transition diagram for *NE*. Provide an explanation of what each state in the machine does, along with a general description of how your machine operates. You must design a standard, deterministic TM and should not design a multitrack TM, multitape TM, doubly-infinite TM, etc.

Problem Two: Closure Properties (20 Points)

In lecture, we explored various closure properties of the recursive and recursively-enumerable languages. In this problem, you will explore several proposed closure properties of these languages.

i. Prove or disprove: the recursive languages are closed under set difference. That is, if L_1 and L_2 are recursive, then $L_1 - L_2$ is recursive.

ii. Prove or disprove: the recursively enumerable languages are closed under set difference. That is, if L_1 and L_2 are recursively enumerable, then $L_1 - L_2$ is recursively enumerable.

The **reversal** of a language is the language L^R defined as

$$L^R = \{ x \mid \exists w \in L.\ x \text{ is the reverse of } w \}$$

For example, if $L = \{ \texttt{0}, \texttt{10}, \texttt{100} \}$, then $L^R = \{ \texttt{0}, \texttt{01}, \texttt{001} \}$. Similarly, if $L = \texttt{0*1*}$, then $L^R = \texttt{1*0*}$.

iii. Prove or disprove: the recursive languages are closed under reversal.

iv. Prove or disprove: the recursively enumerable languages are closed under reversal.

Problem Three: The Unhalting Problem (10 Points)

In lecture, we saw that the question "does M halt on input *w*?" is extremely hard to solve; in fact, it is not recursive, but is RE. In this question, you will see how hard it is to answer the question "does M **not** halt on input *w*?"

Recall from lecture that the language *HALT* is defined as

$$HALT = \{ \langle M, w \rangle \mid M \text{ halts on } w \}$$

To be more explicit, we can write this as

$$HALT = \{ x \mid x \text{ has the form } \langle M, w \rangle \text{ for some TM M and string } w, \text{ and M halts on } w \}$$

As mentioned in lecture, since *HALT* is RE but not recursive, we know that $\overline{HALT}$ is not RE. Thus

$$\overline{HALT} = \{ x \mid \text{either } x \text{ has does not have the form } \langle M, w \rangle \text{ for any TM M and string } w, \text{ or M does not halt on } w \}$$

is not RE.

However, the language $\overline{HALT}$ does not naturally correspond to our intuition about what the opposite of the halting problem is because it contains extra garbage strings that don't correspond to encodings of TM/string pairs. To fix this, we might instead think about the language

$$NOHALT = \{ x \mid x \text{ has the form } \langle M, w \rangle \text{ and M does not halt on } w \}$$

Which is the language of TM/string pairs for which the TM runs forever on the input string. Clearly, no string can be in both *HALT* and *NOHALT*. However, since *NOHALT* is not necessarily the same language as $\overline{HALT}$ (make sure you understand why!) we cannot immediately tell how hard it is to solve.

Suppose that we know the language

$$BADENCODING = \{ x \mid x \text{ does not have the form } \langle M, w \rangle \text{ for any TM M and string } w \}$$

is recursive (it is, since the universal TM needs to be able to check whether a given input actually does encode a legal TM and string). Using closure properties of RE languages, prove that *NOHALT* is unrecognizable.

Problem Four: Why Decidability and Recognizability? (20 Points)

As we saw in lecture, there are two complexity classes associated with Turing machines – the recursively enumerable languages, corresponding to languages that can be recognized by a Turing machine, and the recursive languages, corresponding to languages that can be decided by a Turing machine. But why are there two different complexity classes? Why didn't we talk about a model of computation that accepted just the recursive languages and nothing else? After all, having such a model of computation would be useful – if we could reason about automata that just accept recursive languages, it would be much easier to see what problems are and are not decidable. It turns out, interestingly, that there is no class of automata with this property, and in fact the only way to build automata that can accept all recursive languages is to also have those automata also accept some languages that are RE but not recursive. This problem explores why.

Suppose, for the sake of contradiction, that there is an automaton called a **deciding machine** (or DM for short) that has the computational power to decide precisely the recursive languages. That is, L is recursive if and only if there is a DM that decides L.

We will make the following (reasonable) assumptions about deciding machines:

- Any recursive language is accepted by some DM, and each DM accepts a recursive language.
- Since DMs accept precisely the recursive languages, all DMs halt on all inputs. That is, all DMs are deciders.

- Since deciding machines are a type of automaton, each DM is finite and can be encoded as a string. For any DM D, we will let the encoding of D be represented by $\langle D \rangle$.

- Since DMs are a feasible model of computation, the Church-Turing thesis says that the Turing machine is at least as powerful as a DM. Thus there is some Turing machine U_D that takes as input a description of a DM D and some string *w*, then accepts if D accepts *w* and rejects if D rejects *w*. Note that U_D can never loop infinitely, because D is a deciding machine and always eventually accepts or rejects. More specifically, U_D is the decider "On input $\langle D, w \rangle$, simulate the execution of D on *w*. If D accepts *w*, accept. If D rejects *w*, reject."

Unfortunately, these four properties are impossible to satisfy simultaneously.

i. Consider the language $REJECT_{DM} = \{ \langle D \rangle \mid D \text{ is a DM that rejects } \langle D \rangle \}$. Prove that $REJECT_{DM}$ is decidable.

ii. Prove that there is no DM that decides $REJECT_{DM}$.

Your result from (ii) allows us to prove that there is no class of automaton like the DM that decides precisely the recursive languages. If one were to exist, then it should be able to decide all decidable problems, including $REJECT_{DM}$. However, there is no DM that accepts the decidable language $REJECT_{DM.}$ This means that one of our assumptions must have been violated, so at least one of the following must be true:

- DMs do not accept precisely the recursive languages, or
- DMs are not deciders, or
- DMs cannot be encoded as strings (meaning they lack finite descriptions), or
- DMs cannot be simulated by a TM (they are not feasible models of computation)

Thus there is no feasible model of computation that accepts just the recursive languages.

Problem Five: Accepting Short Strings (20 Points)

(Depending on the pace of Friday's lecture, we may not have covered reductions until after Thanksgiving break. If you are interested in working on this problem, you may want to read chapter 5, section 1 of Sipser before proceeding)

Consider the language $A_{<4} = \{ \langle M \rangle \mid M \text{ accepts some string of length less than 4} \}$ over $\Sigma = \{0, 1\}$.

i. Prove that $A_{<4}$ is recognizable.

ii. Prove that $A_{<4}$ is undecidable by reducing *HALT* to it. That is, show that if there were a decider for $A_{<4}$, there would be a decider for *HALT*.

Problem Six: Accept all the Strings! (20 Points)[*]

(We will cover the material necessary to answer this question after returning from Thanksgiving break. If you are interested in working on this problem, you may want to read chapter 5, section 3 of Sipser before proceeding)

Consider the language

$$A_{ALL} = \{ \langle M \rangle \mid M \text{ accepts all strings} \}$$

This language is not Turing-recognizable, and in this problem you will prove why.

Suppose that you are given a Turing machine M and a string *w*. We can construct a new TM M' that works as follows:

M' = "On input *x*:
Run M on *w* for |*x*| steps.
If M halted in |*x*| steps, loop infinitely.
Otherwise, accept."

For example, on input 000, M' would run M on *w* for 3 steps, looping infinitely if M halted within that time and accepting otherwise. Similarly, if M' were run on 010101, M' would accept if M did not halt on *w* within 6 steps and would loop infinitely otherwise.

i. Prove that M loops infinitely on *w* iff $L(M') = \Sigma^*$.

ii. Prove that A_{ALL} is not recognizable by reducing *NOHALT* to it (see Problem 3 for a description of *NOHALT*). That is, prove that if there is a TM that recognizes A_{ALL}, there would have to be a TM that recognizes *NOHALT*.

---

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

Problem Seven: Rice's Theorem (15 Points)

(We will cover the material necessary to answer this question after returning from Thanksgiving break. If you are interested in working on this problem, you may want to read exercises 5.28 and 5.30 of Sipser before proceeding)

For each of the following languages, explain whether or not Rice's theorem applies to those languages. For those languages for which Rice's theorem does not apply, prove whether the language is decidable or undecidable.

i. $L = \{ \langle M \rangle \mid M \text{ accepts at least one even-length string.} \}$

ii. $L = \{ \langle M \rangle \mid M \text{ never prints the tape symbol under the tape head on a transition.} \}$

iii. $L = \{ \langle M \rangle \mid M \text{ rejects all descriptions of Turing machines} \}$

Problem Eight: 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. Did you attend Monday's problem session? If so, did you find it useful?

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

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

Submission Instructions

There are four ways to submit this assignment:

1. Hand in a physical copy of your answers at the start of class.

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 Venture Fund Laboratories.” There will be a clearly-labeled filing cabinet into which you can submit your homework.

3. Submit a physical copy of your answers in the drop-off box in the Gates basement. We'll send out an email announcement about the location of this box on the night of Friday, December 2.

4. Send an email with an electronic copy of your answers to **cs103@cs.stanford.edu**

Extra Credit Problem: Non-Closure under Homomorphism (10 Points Extra Credit)

Unlike the regular languages and context-free languages, recursive languages are *not* closed under homomorphism. In particular, there exists a recursive language L over some alphabet Σ_1 and a homomorphism $h : \Sigma_1^* \to \Sigma_2^*$ such that h(L) is undecidable. Find a language L and homomorphism h with this property, then prove that L is recursive, h is a homomorphism, and h(L) is undecidable.

As a hint, the recursive languages are closed under a special type of homomorphism called an **ε-free homomorphism**, which is a homomorphism that does not map any of the input symbols to ε.