CSCI 1010 - Fall'26 Theory of Computation Lorenzo De Stefani

HW6

Due: October 30, 2025

Reminder: Submit your assignment on Gradescope by the due date. Submissions must be typeset. Each page should include work for only one problem (i.e., make a new page/new pages for each problem). See the course syllabus for the late policy.

While collaboration is encouraged, please remember not to take away notes from any labs or collaboration sessions. Your work should be your own. Use of other third-party resources is strictly forbidden.

Please monitor Ed discussion, as we will post clarifications of questions there.

Problem 1

In class, we have seen how reductions from known undecidable problems, such as A_{TM}, can be used to prove that other languages are undecidable. Rice's Theorem provides a general result that establishes the undecidability of many problems concerning the languages recognized by Turing machines, without requiring a separate reduction for each one.

A *property* $\mathcal{P}$ of Turing-recognizable languages is a collection of Turing-recognizable languages. We say that a language L satisfies $\mathcal{P}$ if $L \in \mathcal{P}$. The property is *nontrivial* if at least one Turing-recognizable language satisfies it and at least one Turing-recognizable language does not.

Rice's Theorem. Let $\mathcal{P}$ be a property of Turing-recognizable languages, and let

$$L_{\mathcal{P}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \in \mathcal{P}\}.$$

Then $L_{\mathcal{P}}$ is undecidable if and only if $\mathcal{P}$ is nontrivial.

Rice's Theorem concerns properties of the *language recognized* by a Turing machine, not properties of its particular implementation. For example, whether $L(M)$ is infinite is a property of the language, whereas whether M has exactly 10 states is a property of the machine.

For each of the following languages, determine whether Rice's Theorem can be applied directly to prove undecidability. If it can, identify the relevant property, show that it is nontrivial by giving appropriate examples, and

conclude that the language is undecidable. If it cannot, explain why and prove undecidability using a reduction from A_{TM} without invoking Rice's Theorem. You may use Rice's Theorem as stated above without proving it.

(a)
$$L_{inf} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is infinite}\}.$$

(b)
$$L_{101} = \{\langle M \rangle \mid M \text{ is a TM and } 101 \in L(M)\}.$$

(c)
$$L_{seq} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) \subseteq L(M_2)\}.$$

Solution:

a. In this case, you can use Rice's Theorem to prove that L_{inf} is undecidable.

Rice's theorem states that if a language L consisting of descriptions of Turing Machines satisfies the following three conditions, it is undecidable:

(a) $\langle M \rangle \in L$ and $L(M) = L(N) \Rightarrow \langle N \rangle \in L$.
(b) $\exists \langle M \rangle \in L$.
(c) $\exists \langle M \rangle$ not in L.

L_{inf} satisfies (1): Suppose $\langle M \rangle \in L$ and $L(M) = L(N)$. Since $L(M)$ consists of the TM encodings that recognize infinite languages, we also have that $L(N)$ consists of TM encodings that recognize infinite languages, which implies that $\langle N \rangle \in L$.

L_{inf} satisfies (2): Let M be the TM that accepts immediately on all inputs. Then, $L(M)$ is not finite, and $\langle M \rangle$ is in L_{inf}.

L_{inf} satisfies (3): Let M be the TM that rejects immediately on all inputs. Then, $\mid L(M) \mid = 0$, and $\langle M \rangle$ is not in L_{inf}.

Therefore, by Rice's Theorem, L_{inf} must be undecidable.

b. In this case, you can use Rice's Theorem to prove that L_{101} is undecidable.

Rice's theorem states that if a language L consisting of descriptions of Turing Machines satisfies the following three conditions, it is undecidable:

(a) $\langle M \rangle \in L$ and $L(M) = L(N) \Rightarrow \langle N \rangle \in L$.

(b) $\exists \langle M \rangle \in L$.

(c) $\exists \langle M \rangle$ not in L.

L_{101} satisfies (1): Suppose $\langle M \rangle \in L$ and $L(M) = L(N)$. Since $L(M)$ contains 101, it must be that $L(N)$ contains 101.

L_{101} satisfies (2): Let M be the TM that accepts immediately on all inputs. Then $101 \in L(M)$ and $\langle M \rangle$ is in L_{101}.

L_{101} satisfies (3): Let M be the TM that rejects immediately on all inputs. Then $101 \notin L(M)$ and $\langle M \rangle$ is not in L_{101}.

Therefore, by Rice's Theorem, L_{101} must be undecidable.

c. We cannot use Rice's theorem to conclude that L_{SEQ} is undecidable because we are examining a language that is composed of tuples of TMs. The property is not solely about the language of a single TM; it is dependent on the language of another TM and their relation with one another.

To prove that L_{SEQ} is undecidable, we will do a proof by contradiction. Assume that L_{SEQ} is decidable, then there exists a TM R that decides L_{SEQ}. We construct a TM S that decides $EQ_{TM} = \{\langle M_1 \rangle, \langle M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ using R as follows:

$S =$ "On input $\langle M_1 \rangle, \langle M_2 \rangle$, where M_1 and M_2 are TMs:

(a) Ensure that M_1 and M_2 are TMs.

(b) Run R on input $\langle M_1 \rangle, \langle M_2 \rangle$. If R accepts, proceed; if R rejects, reject.

(c) Run R on input $\langle M_2 \rangle, \langle M_1 \rangle$. If R accepts, accept; if R rejects, reject."

Since R decides L_{SEQ}, S decides EQ_{TM}. But EQ_{TM} is undecidable as shown in class, so it must be that L_{SEQ} is undecidable, completing the proof.

For the correctness of S, consider $\langle M \rangle, \langle M' \rangle \in EQ_{TM}$, which tells us that M and M' are TMs and $L(M) = L(M')$. Since $L(M) = L(M')$, we have that $L(M) \subseteq L(M')$ and $L(M') \subseteq L(M)$. Therefore, R will accept on inputs $\langle M \rangle, \langle M' \rangle$ and $\langle M' \rangle, \langle M \rangle$, meaning S accepts $\langle M \rangle, \langle M' \rangle$. Now, consider $\langle M \rangle, \langle M' \rangle \notin EQ_{TM}$. Either M is not a

TM, M' is not a TM, $L(M) \neq L(M')$, or a combination. If M or M' is not a TM, R will reject immediately. If $L(M) \neq L(M')$, S will reject on either step 2 or step 3. In all cases, S rejects $\langle M \rangle, \langle M' \rangle$.

For the decidability of S, we run R twice, which is decidable from our assumption. Therefore, we have that S is decidable.

Problem 2

Prove that for any language A, A is Turing recognizable if and only if $A \leq_m A_{TM}$.

Solution:

- A is Turing-recognizable $\implies$ $A \leq_m A_{TM}$: First, suppose that A is Turing-recognizable and let M be a Turing machine that recognizes A. The function f defined by $f(w) =< M, w >$ is a reduction from A to A_{TM} because it is obviously computable and we have that

  $$w \in A \iff M \text{accepts } w \iff < M, w > \in A_{TM} \iff f(w) \in A_{TM}$$

- $A \leq_m A_{TM}$ $\implies$ A is Turing-recognizable: We know that A_{TM} is Turing-recognizable, so by Theorem 5.28, A is Turing-recognizable.

Problem 3

Consider the following language:

$$L = \{\langle M, k \rangle \mid \ M \text{ is a TM that halts on at least } k \text{ inputs}\}$$

Is this language Turing-recognizable? If so, is it decidable? Prove your answers. (Hint: you may want to use nondeterminism.)

Solution: Yes, the language is Turing-recognizable. Consider an NTM that non-deterministically selects k inputs, then runs M on all of those inputs and accepts if they all halt. If there are at least k inputs that halt, then there is at least one accepting branch of the computation, and so we have constructed an NTM that will recognize the language.

Second, the deterministic way. The set of all string is enumerable, and therefore we can assign a number to every possible string in that set. So consider a Turing machine that does the following:

- Write the first input on the tape, followed by some kind of divider character. This divider character will separate the single tape into multiple tapes, of which we can add arbitrarily many, and which we can expand when necessary.
- Simulate the next step of M on all of the virtual tapes. If any one tape halts, add that to a counter that is initialized to 0. If that counter ever reaches k, halt and accept. After simulating one step of M on each virtual tape, append a new virtual tape, containing the next possible input string. Repeat.

Problem 4

A *very sparse language* is a language L over an alphabet Σ that contains exactly one string of each length. If L is a very sparse language, it contains the empty string, exactly one string of length one, exactly one string of length two, and so on. Prove that if L is a very sparse language and is Turing-recognizable, then L is decidable.

Solution:

a. To prove this, we can reduce from A_{TM}^{C}, which we know to be unrecognizable and thus also undecidable. Given an input $\langle M, w\rangle$, a TM that would recognize A_{TM}^{C} would need to recognize the cases when M rejects w or loops infinitely. Suppose M has n states. Then we already know that a TM for $A_{TM}^{(i)}$ when $i < n$ will not accept no matter what, because the TM has more than i states. If there exists some TM for $A_{TM}^{(i)}$ that rejects $\langle M, w\rangle$ where $i \geq n$, then it must be because M does not accept w. Now suppose for the sake of contradiction that for all natural numbers k, there exists some TM T_j that decides $A_{TM}^{(j)}$ for $j \geq k$.

Let S be a Turing machine that on input $\langle M, w\rangle$. It simulates M on w, accepts if M accepts, and rejects if M rejects. Let n be the number of states in S. By supposition, we can take some T_j that solves $A_{TM}^{(j)}$ for $j > n$. We can then build a decider for A_{TM}^{C} as follows:

On input of $\langle M, w\rangle$, our decider runs T_j on $\langle S, \langle M, w\rangle\rangle$. If T_j rejects, then our decider accepts; otherwise, it rejects.

This decider will accept iff T_j rejects $\langle S, \langle M, w\rangle\rangle$. That is, our decider will accept iff S does not accept $\langle M, w\rangle$; or, finally, our decider will

accept iff M does not accept w.

So we have clearly constructed a decider for A_{TM}^{C}, which we know to be unrecognizable; so our assumption must be false, and there must be some k such that for all $j \geq k$, $A_{TM}^{(j)}$ is undecidable.

b. Let L be a recognizable, very sparse language. Suppose R is a recognizer of L, and we would like to be able to decide L. We can construct such a decider D as follows:

On input x, measure the length n of x. Over a finite alphabet Σ, there are a finite number of strings with length n. So, D runs R on all of these strings simultaneously. Since we know that there is exactly one string in L of each length, eventually one of these strings must be recognized; and because we are running each simultaneously (by doing them one step at a time), even if some loop forever it will not block finding the string in L.

Thus, D's simulations of R will accept exactly once on one string of length n, which must be the corresponding string in L. If the accepted string is x, D accepts because x must be in L. Otherwise D rejects, because the only string in L with the same length as x must not be x. Hence, D is a decider for L.