CMSC 28100-1 / MATH 28100-1
Introduction to Complexity Theory
Fall 2017 – Homework 4
Solution

October 19, 2017

Exercise 1 (HMU 9.3.1)**.** Show that the set of Turing machine codes for TMs that accept all inputs that are palindromes (possibly along with other inputs) is undecidable, that is, show that the language

$$L = \{\langle M\rangle : \forall w \in \Sigma^*, (w = w^R \implies w \in L(M))\}$$

is undecidable, where w^R denotes the reverse of the string w.

Solution. Let P be the language of palindromes and consider the class $\mathcal{C}$ of all recursively enumerable languages R such that $P \subseteq R$. Note that $\varnothing \notin \mathcal{C}$ and $\Sigma^* \in \mathcal{C}$, hence $\mathcal{C}$ is a non-trivial class of recursively enumerable languages.

Note also that the language of all TMs that accept all inputs that are palindromes is precisely

$$L = \{\langle M\rangle : P \subseteq L(M)\} = \{\langle M\rangle : L(M) \in \mathcal{C}\},$$

hence by Rice's Theorem, it follows that L is undecidable. ◁

Exercise 2 (HMU 9.3.2)**.** The Big Computer Corp. has decided to bolster its sagging market share by manufacturing a high-tech version of the Turing machine, called BWTM, that is equipped with *bells* and *whistles*. The BWTM is basically the same as your ordinary Turing machine, except that each state of the machine is labeled either a "bell-state" or a "whistle-state". Whenever the BWTM enters a new state, it either rings the bell or blows the whistle, depending on which type of state it has just entered. Prove that it is undecidable whether a given BWTM M, on given input w, ever blows the whistle.

Solution. Suppose not, that is, suppose that there exists a Turing machine D that decides the language

$$W = \{\langle M, w\rangle : M \text{ is a BWTM that blows the whistle at some point on input } w\}.$$

Recall that the language

$$U = \{\langle M, w\rangle : M \text{ is a TM and } w \in L(M)\}$$

is undecidable.

Consider now the Turing machine A given by the following algorithm.

Algorithm 2.1: Algorithm for the TM A.

On input $\langle M, w\rangle$, let $\widetilde{M}$ be the BWTM obtained from M by letting all states be bell states, adding an extra state q_f, which is the unique final state of $\widetilde{M}$ and is a whistle state, and adding transitions $(q, \sigma) \mapsto (q_f, \sigma, R)$ for every $\sigma \in \Sigma$ and every q that is a final state of M that does not have a transition for (q, σ). The initial state of $\widetilde{M}$ is the same as the initial state of M and all other transitions of M are preserved.

Run D on input $\langle \widetilde{M}, w\rangle$.

if D *accepts* **then** Accept.

else Reject.

Note first that A always halts since D always halts.

Note now that if $w \in L(M)$, then the machine $\widetilde{M}$ built by A above gets to a final state of M when given input w. It then transitions to the newly added state q_f, which blows the whistle, hence $\langle \widetilde{M}, w\rangle \in W$, which implies that D accepts $\langle \widetilde{M}, w\rangle$, hence A accepts $\langle M, w\rangle$.

On the other hand, if $w \notin L(M)$, then the machine $\widetilde{M}$ never transitions to q_f when given input w, since for this to happen, the machine M would have to halt on a final state. Since q_f is the only whistle state, it follows that $\langle \widetilde{M}, w\rangle \notin W$, hence A rejects $\langle M, w\rangle$.

Therefore A decides U, which is a contradiction. ◁

Exercise 3 (HMU 9.3.3)**.** Show that the language of codes for TMs that, when started with blank tape, eventually write a 1 somewhere on the tape is undecidable.

Solution. Suppose not, that is, suppose that there exists a Turing machine D that decides the language

$$W = \{\langle M\rangle : M \text{ writes a 1 somewhere in the tape on empty input}\}.$$

Recall that the language

$$U = \{\langle M, w\rangle : M \text{ is a TM and } w \in L(M)\}$$

is undecidable.

Consider now the Turing machine B given by the following algorithm.

Algorithm 3.1: Algorithm for the TM B.

On input $\langle M, w\rangle$, let $\widetilde{M}$ be the TM that runs M on input w using a symbol $\widetilde{1}$ in place of 1 in both the execution of M and in w, and if M accepts w, the machine $\widetilde{M}$ then writes a 1 and halts.

Run D on input $\langle \widetilde{M}\rangle$.

if D *accepts* **then** Accept.

else Reject.

Since D always halts, we know that B always halts.

Note that from our substitution of 1 by $\widetilde{1}$, we know that during the simulation of M by $\widetilde{M}$, no symbol 1 is ever written on the tape. This means that $\widetilde{M}$ writes a 1 on the tape if and only if $w \in L(M)$. But this implies that B accepts $\langle M, w\rangle$ if and only if $w \in L(M)$.

Therefore B decides U, a contradiction. ◁

Exercise 4 (HMU 9.3.5)**.** Let L be the language consisting of pairs of TM codes plus an integer (M_1, M_2, k) such that $L(M_1) \cap L(M_2)$ contains at least k strings. Show that L is recursively enumerable, but not recursive.

Solution. Let us first show that the language

$$P = \{\langle M_1, M_2, k\rangle : |L(M_1) \cap L(M_2)| \geqslant k\}$$

is recursively enumerable.

Consider the Turing machine R given by the following algorithm.

Algorithm 4.1: Algorithm for the TM R.

Given input $\langle M_1, M_2, k\rangle$, let $T_1 \leftarrow \varnothing$ and $T_2 \leftarrow \varnothing$.
for $n \leftarrow 0, 1, \ldots$ **do**
    Let W_n be the set of all input strings of length at most n.
    for $w \in W_n$ **do**
        Run M_1 on input w for n steps.
        if M_1 *accepts* **then** $T_1 \leftarrow T_1 \cup \{w\}$.
        Run M_2 on input w for n steps.
        if M_2 *accepts* **then** $T_2 \leftarrow T_2 \cup \{w\}$.
    if $|T_1 \cap T_2| \geqslant k$ **then** Accept.

Note that R never gets stuck inside the inner "for" loop, that is, if R loops forever, then it must continuously increment n in the outer "for" loop.

Note that if $|L(M_1)\cap L(M_2)| \geqslant k$, then for $w_1, \ldots, w_k \in L(M_1)\cap L(M_2)$ distinct, if we let n_0 be the maximum length of $w_1, \ldots, w_k$, then $w_1, \ldots, w_k \in W_{n_0}$. Furthermore, we know that for every $i \in \{1, \ldots, k\}$, there exists n_i^1 and n_i^2 such that M_1 accepts w_1 within n_i^1 steps and M_2 accepts w_2 within n_i^2 steps.

This implies that if M is given input $\langle M_1, M_2, k\rangle$, then when it gets to

$$n = \max\{n_0, n_1^1, n_2^1, \ldots, n_k^1, n_1^2, n_2^2, \ldots, n_k^2\},$$

all $w_1, \ldots, w_k$ are added both to T_1 and T_2, which implies that M accepts $\langle M_1, M_2, k\rangle$.

On the other hand, if $|L(M_1) \cap L(M_2)| < k$, then M never accepts $\langle M_1, M_2, k\rangle$, since for this to happen we would need that $|T_1 \cap T_2| \geqslant k$ at some point, but T_1 only contains words that are accepted by M_1 and T_2 only contains words that are accepted by M_2.

Therefore $L(R) = P$, hence P is recursively enumerable.

Let us now prove that R is not recursive. Suppose not, that is, suppose that there exists a Turing machine D that decides D.

Recall that the language

$$U = \{\langle M, w\rangle : M \text{ is a TM and } w \in L(M)\}$$

is undecidable.

Consider the Turing machine C given by the following algorithm.

Algorithm 4.2: Algorithm for the TM C.

On input $\langle M, w\rangle$, let M_1 be the Turing machine that accepts all inputs and let M_2 be the Turing machine that runs M on w, and if M accepts, then M_2 accepts (regardless of its input).
Run D on input $\langle M_1, M_2, 1\rangle$.
if D *accepts* **then** Accept.
else Reject.

Note that C always halts.

If $w \in L(M)$, then we have $L(M_2) = \Sigma^* = L(M_1)$, hence $\langle M_1, M_2, 1\rangle \in P$, so C accepts $\langle M, w\rangle$.

On the other hand, if $w \notin L(M)$, then we have $L(M_2) = \varnothing$, hence $\langle M_1, M_2, 1\rangle \notin P$, so C rejects $\langle M, w\rangle$.

Therefore C decides U, a contradiction. $\triangleleft$