Computational Models – Exercise 10

Due Saturday, 1 July 2023

Each student must solve the problems on their own. If you encounter difficulties, you may ask a classmate for a hint or the general idea. However, detailed discussion, note-taking, or sharing of written solutions is not allowed. Do not write down your answers while communicating with other people or show the answers for feedback.

Our grading app has severe limitations, such as no zoom tool. To make sure we can grade your work, please follow these technical guidelines:

Submit a **single PDF file** through Moodle.

The file size is limited to **10 MB**. If necessary, google *reduce PDF file size*.

Fill in your answers **on this form*** in the allocated spaces. The space provided gives you an indication of the expected length and level of detail of the answer. You may add a little more space if you need.

Include everything from this form in your submission. In particular, **include the problem statements**. Do not delete any text or omit pages, just add your answers.

Ensure your answers are **legible** (easy to read) at zoom 100% on a standard computer screen. Your text should be **large**, **sharp**, and in **high contrast** with the background.

Do not squeeze scanned solutions to fit in the space, as the text will become small.

Verify that pages are properly **ordered** and **oriented**.

The page size must be **A4**. Before submitting your file, check its page size using Acrobat Reader: go to File > Properties > Description and confirm that Page Size is around 21 × 29 cm. Note that scanning A4 pages does not guarantee the resulting page size will be A4, due to scaling. If necessary, google *resize PDF to A4*.

Do not add your answers as PDF comments. If you can drag them in Acrobat Reader, they are comments. If necessary, google *flatten PDF*.

A **5-point bonus** will be given to solutions typed in a word processor. Hand-sketched illustrations or diagrams will not deny you this bonus.

If there are technical issues with your submission, you may receive a fine. In extreme cases, your submission may not be graded at all.

If you need help or have questions, please use the course forum at Piazza.

*The only exception is in case you use LaTeX or a similar typesetting system. In that case, copy-paste everything from this file, except for illustrations or other hard-to-reproduce graphical elements. No need to fix corrupted formulas.

Worked with Junil Lee - 000805387

[1 pt] **Problem 1**

Give us a recommendation for a TV show we must watch, tell us what the TV show is about in just **one** sentence.

Game of Thrones. It's a gritty fantasy story about people of power trying to acquire the throne which is filled with character politics.

[2 pts] **Problem 2**

Write down the definition of $A \leq_m B$

This means that A is mapping reducible to mapping B, meaning that there exists a computable function $f : \Sigma^* \to \Sigma^*$ s.t $\forall w \in \Sigma^*$, $w \in A \Leftrightarrow f(w) \in B$.

Problem 3

Prove or disprove:

[2 pts] **1.** Let f be the function defined by the TM:

F = "On input $\langle M \rangle$:
1. Construct a TM $M_\emptyset$ = "On input x: reject".
2. Construct a TM M_{Σ^*} = "On input x: accept".
3. For i from 0 to ∞:
   3.1. For j from 0 to i:
      3.1.1. For every $x \in \Sigma^j$ in lexicographical order:
         3.1.1.1. Run M on x for i steps.
         3.1.1.2. If M accepted, return $\langle M_\emptyset, M_{\Sigma^*} \rangle$.
4. Return $\langle M_\emptyset, M_\emptyset \rangle$."

f is a mapping reduction from $\overline{E_{TM}}$ to $\overline{EQ_{TM}}$.

True. Proof:

Note that in step 3 of f, f loops over every possible word in the alphabet and checks if it is accepted by M, if so then f returns $\langle M_\emptyset, M_{\Sigma^*} \rangle$, otherwise it either does not halt or returns $\langle M_\emptyset, M_\emptyset \rangle$ in step 4. I will refer to this process as the "acceptance step".

1) Let M ∈ $\overline{E_{TM}}$:

Since M ∈ $\overline{E_{TM}}$, this implies that L(M) is not empty, meaning that there exists a word w s.t w ∈ L(M). Therefore f will return $\langle M_{\emptyset}, M_{\Sigma^*}\rangle$ as it passes the "acceptance step" as there does exists word that M accepts, that word being w. $\langle M_{\emptyset}, M_{\Sigma^*}\rangle$ are not equal since $M_{\emptyset}$ is the empty language and at the very least M_{Σ^*} includes the word w, as M_{Σ^*} contains all possible words constructed by the alphabet. Therefore $\langle M_{\emptyset}, M_{\Sigma^*}\rangle \in \overline{EQ_{TM}}$.

2) Let $f(M) = \langle M_1, M_2\rangle \in \overline{EQ_{TM}}$:

Since $\langle M_1, M_2\rangle \in \overline{EQ_{TM}}$, that means that M_1 *and* M_2 are not equal. This means that $\langle M_1, M_2\rangle$ cannot be $\langle M_{\emptyset}, M_{\emptyset}\rangle$, as $M_{\emptyset}$ is equal to itself. Therefore, the only other return step in f that does not return $\langle M_{\emptyset}, M_{\emptyset}\rangle$ is step 3.1.1.2, in the "acceptance step". This step only occurs if f passes the "acceptance step" i.e only if there exists a word w which is accepted by M. Therefore, we know that there is a word w that is accepted by M. Therefore, M is not the empty language, Therefore M ∈ $\overline{E_{TM}}$.

[2 pts] **2.** $\forall L \in RE$ it holds that $L \leq_m \overline{E_{TM}}$.

F = "On input x:
1. Run M, the TM that recognizes L, on x
   1.1. If M accepts x, return the TM $M_{\emptyset}$ = "On input x: reject"

2. Return the TM M_{Σ^*} = "On input x: accept".

Correctness

1) If x ∈ L, then that means M, the TM that recognizes L, accepts x. Therefore when f is run on x, it enters step 1.1, and therefore returns $M_{\emptyset}$. L($M_{\emptyset}$) is the empty language and therefore $M_{\emptyset} \in \overline{E_{TM}}$

2) If x ∉ L then that means M, the TM that recognizes L, rejects x or does not halt. If M rejects x, then when f is run on x, it enters step 2, and therefore returns M_{Σ^*} which is not an empty language. Therefore $M_{\Sigma^*} \notin \overline{E_{TM}}$. If M does not halt on x, then F does not halt and therefore returns the empty language. Therefore F(M) ∉ $\overline{E_{TM}}$.

3. Given language $L \subseteq \Sigma^*$ and a function (not necessarily computable) $f: \Sigma^* \to \Sigma^*$.

Define $f(L) = \{f(x) | x \in L\}$.

Assume that $f(x)$ is defined for all $x \in \Sigma^*$.

[2 pts] **a**. If $L \in R$, then $f(L) \in RE$.

I don't know

4. Let $C = \{L | A_{TM} \leq_m L\}$.

[2 pts] **a**. Let A and B be a pair of languages, if $A \leq_m B$ and $A \in C$, then $B \in C$.

Since $A \leq_m B$, then there exists a computable mapping function $f: \Sigma^* \to \Sigma^*$ that maps A to B. Since $A \in C$, then there exists a computable mapping function $g: \Sigma^* \to \Sigma^*$ that maps A_{TM} to A. To prove $B \in C$, we need to show a mapping function $h: \Sigma^* \to \Sigma^*$, that maps A_{TM} to B. Let $h(x) = f(g(x))$, which is a computable function as it is a composition of 2 computable functions. We will show that $A_{TM} \leq_m B$ with the mapping function h(x)

Proof:

1) Let <M,w> $\in A_{TM}$:

$h(< \mathrm{M}, \mathrm{w} >) = f\big(g(< \mathrm{M}, \mathrm{w} >)\big) = f(w_A) = w_B$ where w_A is some word in A that is mapped to from $g(< \mathrm{M}, \mathrm{w} >)$ and w_B is a word in B that is mapped to from $f(w_A)$. Recall that g maps from A_{TM} to A, and f maps A to B. Therefore $h(< \mathrm{M}, \mathrm{w} >) = w_B \in \mathrm{B}$.

2) Let <M,w> $\notin A_{TM}$:

$h(< \mathrm{M}, \mathrm{w} >) = f\big(g(< \mathrm{M}, \mathrm{w} >)\big) = f(w \notin A) = \mathrm{w} \notin B$. This is because since <M,w> $\notin A_{TM}$, then $g(< \mathrm{M}, \mathrm{w} >)$ maps to a word not in A (by definition of what it means to be mapping reducible), and same reasoning for why $f(w \notin A) = \mathrm{w} \notin B$ with the mapping function in this case being the mapping function for $A \leq_m B$. Therefore $h(< \mathrm{M}, \mathrm{w} >) = \mathrm{w} \notin B$

Therefore $A_{TM} \leq_m B$, and therefore $B \in C$ by definition of C.

[2 pts] **b**. If $L \notin C$ then $L \in R$.

False. Counterexample:

Let $\mathrm{L} = \overline{A_{TM}}$. $\overline{A_{TM}} \notin C$ as A_{TM} is not map reducible to $\overline{A_{TM}}$. This is because if they were, then since $A_{TM} \leq_m \overline{A_{TM}}$ and A_{TM} is RE, that would imply $\overline{A_{TM}}$ is

also RE. However, this would imply that A_{TM} is decidable – this is a contradiction. Therefore A_{TM} is not map reducible to $\overline{A_{TM}}$ and therefore $\overline{A_{TM}} \notin C$. However, $\overline{A_{TM}} \notin$ R.

[2 pts] **c**. $\forall L \in C$ it holds that $L \notin coRE$.

True. Proof:

Let $L \in C$. This means that $A_{TM} \leq_m L$, and since $A_{TM} \notin coRE$, then $L \notin coRE$

[no pts] **5**. If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$ (mapping reducibility is transitive).

[no pts] **6**. Let $L \in R$, then L is mapping-reducible to any non-trivial language (the trivial languages are ∅ and Σ^*).

Problem 4

[11 pts] **1**. $L = \{\langle M, w\rangle \in A_{TM} : |\langle M\rangle| \leq 2^{42}\}$. Prove that language <u>L is not decidable</u> **using a Turing reduction**. You can assume that the encoding of a universal TM is less than length 2^{42}.

I don't know

*[11 pts]***2**. Let $L = \left\{\langle M\rangle \,\middle|\, \begin{array}{l} M \text{ is a TM and } \exists \text{ polynomial } P \text{ s.t. } M \text{ halts} \\ \text{on every input } w \text{ in at most } P(|w|) \text{ steps} \end{array}\right\}$. Prove that language <u>L is not decidable</u> **using a Turing reduction**.

ATC that L is decidable. Therefore L has a decider D that decides it. We will use D to build a decider for $HALT_{TM}$:

M = "On input <M, w>:

1. Construct a TM M_w = "On input x: Run M on w"

2. Run D on $\langle M_w \rangle$ and do as D does"

Since M is a valid decider for $HALT_{TM}$, then $HALT_{TM}$ is decidable, which is a contradiction to $HALT_{TM}$ being undecidable. Therefore, by contradiction, L is undecidable.

<u>Correctness</u>

1) Let <M,w> ∈ $HALT_{TM}$:

This means that M halts on input w in a certain amount of steps – denote the number of steps by j. This means that M_w halts in j steps (for any input). Define the polynomial p(x) = x + j. For any input x of M_w, p(|x|) = |x| + j >= j. Therefore for any input x, M_w halts in at most p(|x|) steps. Therefore $M_w \in$ L, since there exists such a polynomial (the polynomial specified in the condition for L). Therefore D accepts $\langle M_w \rangle$. Therefore M accepts $\langle M_w \rangle$.

1) Let <M,w> $\notin$ $HALT_{TM}$:

This means that M does not halt on input w. This means that M_w does not halt. Since M_w does not halt for any input, then there cannot be a polynomial P for which M_w halts on every input x in p(|x|) steps (since it does not halt at all). Therefore $M_w \notin$ L, Therefore D rejects $\langle M_w \rangle$. Therefore M rejects $\langle M_w \rangle$.

[11 pts] **3**. For a TM M, we define the function $\text{dist}_M: \Sigma^* \to \mathbb{N}$ as follows: $\text{dist}_M(w)$ is the index of the right-most tape cell to which M reaches during its run on w (the index of the left-most tape cell is 1). If there is no right-most cell, then $\text{dist}_M(w) = 0$.

Let $L = \{\langle M \rangle \mid M \text{ is a TM and } \exists w \in \Sigma^* \text{ s.t. } \text{dist}_M(w) = 0\}$. Prove that language L is not recognizable **using a Turing reduction**.

ATC that L is recognizable. Then there exists a recognizer R for L. Using R we can build a recognizer for $\overline{HALT_{TM}}$ as follows:

M = "On input <M, w>:

1. Construct a TM M_w = "On input x: Run M on w"

2. Run R on $\langle M_w \rangle$ and do as R does"

Since M is a valid recognizer for $\overline{HALT_{TM}}$, then $\overline{HALT_{TM}}$ is recognizable, which is a contradiction to $\overline{HALT_{TM}}$ being unrecognizable. Therefore, by contradiction, L is not recognizable.

Correctness:

1) Let <M, w> $\in$ $\overline{HALT_{TM}}$:

Therefore M does not halt on w. Therefore M_w from step 1, does not halt. Therefore $M_w \in$ L, since $\text{dist}_{M_w}(w) = 0$ (when a TM does not halt, its distance is 0). Therefore R will accept $\langle M_w \rangle$, and therefore M will accept.

2) Let <M, w> $\notin$ $\overline{HALT_{TM}}$:

Therefore M does halts on w. Therefore M_w from step 1, does halt. Therefore $M_w \notin$ L, since $\text{dist}_{M_w}(w) \neq 0$ (when a TM does halt, its distance is at least 1). Therefore R can reject $\langle M_w \rangle$, and therefore M will reject. Or R does not halt.

Problem 5

Determine whether $L \in R, L \in RE \backslash R, L \in coRE \setminus R$, or none of these, and prove your answer. **When proving non-belonging use mapping reductions, don't use Turing Reductions.**

[10 pts] **1.** $L = \left\{ \langle M_1, M_2, M_3 \rangle \middle| \begin{matrix} M_1, M_2 \text{ and } M_3 \text{ are TMs and} \\ |L(M_1) \triangle L(M_2)| \leq |L(M_2) \triangle L(M_3)| \end{matrix} \right\}$.

$\triangle$ means symmetric difference (ex9, problem 3.1).

L is not in any of the groups. Proof:

$f =$ "On input $\langle M_1, M_2 \rangle$:

1. Return $\langle M_1, M_2, M_2 \rangle$"

f is a mapping from EQ_{TM} to L. Since $EQ_{TM} \leq_m L$, and EQ_{TM} does not belong to any of the groups, then L does not belong to any of the groups.

Correctness

1) Let $\langle M_1, M_2 \rangle \in EQ_{TM}$:

This means that $\text{L}(M_1) = \text{L}(M_2)$. Therefore $|L(M_1) \triangle L(M_2)| = 0$, as every word in $L(M_1)$ is in $L(M_2)$. $|L(M_2) \triangle L(M_2)| = 0$ as it's the symmetric difference between a language and itself. Therefore $|L(M_1) \triangle L(M_2)| = 0 = |L(M_2) \triangle L(M_2)|$. Therefore $f(\langle M_1, M_2 \rangle) = \langle M_1, M_2, M_2 \rangle \in$ L

2) Let $\langle M_1, M_2 \rangle \notin EQ_{TM}$:

This means that $\text{L}(M_1) \neq \text{L}(M_2)$. Therefore $|L(M_1) \triangle L(M_2)| > 0$, as there exists a word in $L(M_1)$ that is not $L(M_2)$ or vice versa. $|L(M_2) \triangle L(M_2)| = 0$ as it's the symmetric difference between a language and itself.

Therefore $|L(M_1) \triangle L(M_2)| > |L(M_2) \triangle L(M_2)|$. Therefore $f(\langle M_1, M_2 \rangle) = \langle M_1, M_2, M_2 \rangle \notin$ L

[10 pts] **2.** $L_{pc} = \left\{0^p 1^c \middle| \begin{matrix} p \text{ is a prime number and} \\ c \text{ is a composite number} \end{matrix}\right\}$.

I don't know

[10 pts] **3.** $L = \left\{\langle M\rangle \middle|_{L(M) = \{\langle N\rangle | N \text{ is a TM and } L(N) = L_{pc}\}} \right\}$.

f = "On input <M>:
1. For every TM encoding $\langle N\rangle \in \Sigma^*$:
    1.1 Run M on $\langle N\rangle$:
        1.1.1 If M accepts, then for j from 0 to $infinity$:
            1.1.1.1 For every $x \in \Sigma^j$ (alphabet of N) in lexicographical order:
                1.1.1.1.1 Run N on x and Run R, the recognizer/decider for L_{pc}, on x:
                    1.1.1.1.1.1 If R rejects or N rejects (but NOT both), then return
                        $M_\emptyset$ = "On input x: reject".

2. Return M_{Σ^*} = "On input x: accept"

f is a mapping reduction from L to $\overline{E_{TM}}$. Therefore since $\overline{E_{TM}}$ is only in $coRE \setminus R$, then so is L.

Explanation of f:

The general idea of f is to iterate over every $\langle N\rangle$ in L(M) and check that if a word is in L(N), then it must be in L_{pc} and vice versa. (i.e inclusion both ways).

$f(\langle M\rangle)$ will iterate over every TM in the alphabet of L and for every TM <N> that is recognized by M, the algorithm will then loop over every possible word x (in the alphabet of N) and check if N recognizes it and if R, the recognizer/decider for L_{pc}, recognizes it. If one recognizes x and the other doesn't, then $M_\emptyset$ will be returned. If it loops over every TM of alphabet L and every word in that TM's language is accepted by both N and R or rejected by both of them, then step 2 will occur and M_{Σ^*} will be returned.

Correctness

1) Let $\langle M\rangle \in L$:
Therefore for every <N> in L(M), L(N) = L_{pc}. Therefore when f iterates over every word in each L(N), both N and R accept the word. Therefore f completes the loop in step 1 without returning and goes to step 2, which returns M_{Σ^*}. Therefore $f(\langle M\rangle) = M_{\Sigma^*} \in \overline{E_{TM}}$.

2) Let $\langle M \rangle \notin L$:
Therefore there exists a TM <N> in L(M), s.t L(N) $\neq L_{pc}$. Therefore there exists a word w that is either not in L(N) and is in L_{pc} or vice versa. When f iterates over every word in L(N), it will eventually iterate over w. When it iterates over w either N will accept the word and R will reject it or vice versa. Or N or R will not halt on w. In the 1st case f enters step 1.1.1.1.1.1 and returns $M_{\emptyset}$, which is not in $\overline{E_{TM}}$. Therefore $f(\langle M \rangle) = M_{\emptyset} \notin \overline{E_{TM}}$. In the second case f does not halt, and therefore produces the empty language which is also not in $\overline{E_{TM}}$.

[10 pts] **4.** The question has been cancelled.

[10 pts] **5.** $L = \{\langle M \rangle | M \text{ is a TM and } \overline{E_{TM}} \subseteq L(M)\}$.

Define the mapping function f by the following TM:

f = "On input <M>:
1. For every TM encoding $\langle N \rangle \in \Sigma^*$:
    1.1.1 Run R, the recognizer for $\overline{E_{TM}}$, on $\langle N \rangle$ and Run M on ⟨N⟩:
        1.1.1.1 If R accepts and M rejects, then return $M_{\emptyset}$ = "On input x: reject".

2. Return M_{Σ^*} = "On input x: accept"

f is a mapping reduction from L to $\overline{E_{TM}}$. Therefore since $\overline{E_{TM}}$ is only in $coRE \setminus R$, then so is L.

<u>Correctness</u>

1) Let $\langle M \rangle \in L$:
Therefore $\overline{E_{TM}} \subseteq L(M)$. Therefore $f(\langle M \rangle)$ will loop over every TM in the alphabet and for every TM that is recognized by R (the recognizer for $\overline{E_{TM}}$) , M will accept as well since $\overline{E_{TM}} \subseteq L(M)$. Therefore $f(\langle M \rangle)$ will not return anything during the loop and will get to the end of the algorithm after the loop, namely step 2. Therefore $f(\langle M \rangle)$ will return M_{Σ^*} in step 2, which is in $\overline{E_{TM}}$. Therefore $f(\langle M \rangle) = M_{\Sigma^*} \in \overline{E_{TM}}$.

2) Let $\langle M \rangle \notin L$:
Therefore $\overline{E_{TM}} \not\subset L(M)$ (and $\overline{E_{TM}} \neq L(M)$). Therefore there exists a word/TM $\langle N \rangle$ s.t $\langle N \rangle$ is in $\overline{E_{TM}}$ but is not in L(M). Therefore when $\langle N \rangle$ is iterated over in the loop (step 1) in f, it will be accepted by R and, if M halts, rejected by M – therefore it will enter step 1.1.1.1 and return $M_{\emptyset}$ which is an empty language. If M does not halt then the empty language will still be returned by f. In both cases the empty language is returned and the empty language is not in $\overline{E_{TM}}$. Therefore $f(\langle M \rangle) = \emptyset \notin \overline{E_{TM}}$.

[no pts] **6**. $L = \{< M > \mid M\ is\ a\ TM\ and\ there\ exist\ n, m \in \mathbb{N}\ s.t.\ 1^n \in L(M), 1^m \notin L(M)\}$.

[no pts] **7**. $L = \{< M > \mid M\ is\ a\ TM\ and\ L(M) \cup HALT_{TM} \in RE\}$

[no pts] **8**. $L = \{< M > \mid M$ is a TM and $\exists x \in \Sigma^*$ s.t. M halts on $x\}$

[no pts] **Problem 6**

For each of the following languages L, determine whether Rice's theorem applies. If yes, state the set of languages C such that $L = L_C$. If not, explain why.

1. $L = \left\{\langle M\rangle \middle| \begin{matrix} M\ is\ a\ TM\ and\ its\ number\ of\ states\ is\ minimal \\ among\ all\ TM's\ that\ recognize\ L(M) \end{matrix}\right\}$.

2. $L = \{\langle M\rangle \mid M\ is\ a\ TM\ and\ M\ accepts\ all\ inputs\ of\ length\ 1\}$.

3. $L = \{\langle M\rangle \mid M\ is\ a\ TM\ and\ M\ never\ enters\ q_{rej}\ on\ it's\ run\ on\ \varepsilon\}$.

4. Given TM M_0 that halts on all inputs. $L = \{\langle M\rangle | M\ is\ a\ TM\ and\ \langle M_0\rangle \in L(M)\}$.