Computational Models – Exercise 9

Due Saturday, 24 June 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

Forward

From this exercise on, all descriptions of TMs should be high-level, unless indicated otherwise.

This exercise strongly relies on the concept of simulating one TM by another. This concept was explained in class. If M_1 and M_2 are two TMs, then the encoding of a TM M_1, denoted by $\langle M_1 \rangle$, is a string that can be an input or part of an input to another TM M_2. The high-level description of M_2 may then include steps such as "simulate running M_1 on input '1101'" (or simply "run M_1 on 1101"). The problems in this exercise are intended to help you understand the basics of this concept. We will discuss it in more depth over the next few weeks.

[1 pt] **Problem 1**

Read the forward above. On default, should you give low-level or high-level descriptions of TMs? High level descriptions

Problem 2

Some of the following statements do not make sense because of incorrect use of definitions or terminology. For each of them, determine if it makes sense or not. If not, explain why.

Let M_1 and M_2 be TMs.

Let A be a language.

Let x be a string.

[2 pts] **1.** Run x on M_1.

This does not make sense. You cannot run an input on a TM, but rather you run a TM on an input – so the correct statement would be "run M1 on x".

[2 pts] **2.** Run M_1 on $\langle M_2 \rangle$.

This makes sense.

[2 pts] **3.** $L(M_1) = L(M_2)$.

This makes sense.

[2 pts] **4.** $L(M_1) \leq L(M_2)$.

This does not make sense. A language cannot be less than another language, as there is no meaning of "less than" in this context. A possible meaning is that $L(M_1)$ has less words than $L(M_2)$, however this would be denoted as $|L(M_1)| \leq |L(M_2)|$.

[2 pts] **5.** $L(M_1) \in L(M_2)$.

This does not make sense. $L(M_2)$ is a set of strings, and therefore only a string can be an element of $L(M_2)$, therefore $L(M_1)$, which is a whole set/language cannot be an element of $L(M_2)$.

[2 pts] **6.** Run M_1 on A.

This does not make sense. You cannot give a language as an input to a Turing machine. Turing machines only accept singular strings as an input.

[2 pts] **7.** A is not recognizable.

This makes sense.

[2 pts] **8.** $L(M_2)$ is not recognizable.

This does not make sense. This is contradictory/paradoxical as $L(M_2)$ is, by definition, the language that is recognized by the TM M_2, and therefore it is recognizable.

[2 pts] **9.** $L(M_2) = x$.

This makes sense.

[2 pts] **10.** On input M_1….

This does not makes sense as a TM cannot take another TM as an input, but rather can take a string representation of a another TM as an input – so the correct statement would be "On input $< M_1 >$…."

Problem 3

[10 pts] **1.** Given two languages L_1 and L_2, the symmetric difference of L_1 and L_2 is defined by: $L_1 \triangle L_2 = \{w \in \Sigma^* | w \in L_1 \setminus L_2 \; or \; w \in L_2 \setminus L_1\}$.

Prove that if $L_1 \in R$ and $L_1 \triangle L_2 \in R$ then $L_2 \in R$.

Let M_{L_1} be a decider for L_1 and let $M_{L_1 \triangle L_2}$ be a decider for $L_1 \triangle L_2$. We define a decider for L_2, M_{L_2} as follows:

M_{L_2} = "On input w:

1. Run $M_{L_1 \triangle L_2}$ on w
2. Run M_{L_1} on w
3. If $M_{L_1 \triangle L_2}$ accepts and M_{L_1} accepts, then reject.

   If $M_{L_1 \triangle L_2}$ accepts and M_{L_1} rejects, then accept.

   If $M_{L_1 \triangle L_2}$ rejects and M_{L_1} accepts, then accept.

   If $M_{L_1 \triangle L_2}$ rejects and M_{L_1} rejects, then reject."

<u>Correctness</u>

For W $\in L_2$, there are two cases:

1. W $\in L_1 \triangle L_2$ => $M_{L_1 \triangle L_2}$ accepts w and M_{L_1} rejects w (we know w is not in L_1 as if it is in $L_1 \triangle L_2$ and in L_2, then it cannot be in L_1) => M_{L_2} accepts

2. W $\notin L_1 \triangle L_2$ => $M_{L_1 \triangle L_2}$ rejects w and M_{L_1} accepts w (we know w is in L_1 as if it is not in $L_1 \triangle L_2$ but it is in L_2, then w has to be in the intersection of L_1 *and* L_2) => M_{L_2} accepts

For W $\notin L_2$, there are two cases:

1. W $\in L_1 \triangle L_2$ => $M_{L_1 \triangle L_2}$ accepts w and M_{L_1} accepts w (we know w is in L_1 as if it is in $L_1 \triangle L_2$ and not in L_2, then the only remaining option as to why w is in $L_1 \triangle L_2$, is that it is in L_1) => M_{L_2} rejects

2. W $\notin L_1 \triangle L_2$ => $M_{L_1 \triangle L_2}$ rejects w and M_{L_1} rejects w (we know w is not in L_1 as it's not in $L_1 \triangle L_2$, and it can't be in the intersection of L_1 *and* L_2 either as it is not in L_2, meaning it is not in L_1 at all) => M_{L_2} rejects

[10 pts] **2.** Given two languages $L_1, L_2 \in RE$.

prove that $L = L_1L_2 \in RE$. Recall that $L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$.

Let M_{L_1} be a recognizer for L_1 and let M_{L_2} be a recognizer for L_2. We define a decider for L, M_L as follows:

M_L = "On input w:

1. non-deterministically split w into 2 parts, x and y, s.t w = xy and for each such split:

   a) Run M_{L_1} on x. If M_{L_1} rejects, then reject, otherwise go to step b.

   b) Run M_{L_2} on y and do as M_{L_2} does"

Correctness

1. W ∈ L => there exists a split of w into 2 parts x and y s.t w = xy => there will be 1 split of w that M_L processes that is the correct split => In this split, M_{L_1} on x accepts and M_{L_2} on y accepts => M_L *accepts*

2. W ∉ L => for any split of w into 2 parts x and y, M_{L_1} does not accept x or M_{L_2} does not accept y (or both) => M_{L_1} rejects or does not halt on x or M_{L_2} rejects or does not halt on y (or both) => M_L *rejects* or does not halt on w => M_L *does not accept w*

[6 pts] **3.** Fill in the following table with "True" or "False", indicating whether the specified class is closed under the specified operation or not. Some of these require non-trivial proofs, which you should work out, but do not need to include.

	Union	Intersection	Complement	Kleene star (*)
Class of finite languages	True	True	False	False
R	True	True	True	True
RE	True	True	False	True

Problem 4

You may assume Σ = {0,1}.

[19 pts] **1.** Let $A = \left\{ \langle M, s, c \rangle \middle| \begin{matrix} M \text{ is a TM}, s, c \in \mathbb{N} \text{ and } \exists x_1, \dots, x_s \in \Sigma^* \; s.t \; M \text{ accepts} \\ x_1, \dots, x_s \text{ using at most } c \text{ cells} \end{matrix} \right\}$.

Prove that $A \in R$.

M_c = "On input <M, w, c>, a TM, a word and a natural number:

1. Run M on w, but with every step check if the head has passed the c'th cell:

   a) if the head has passed the c'th cell, then reject

2. Accept (Note: only happens if 1a was never true)

M_A = "On input <M,s,c>:

1. Set a running counter on the second tape (will be referred to as j) initialized to 0.

2. For i from 0 to c:

   2.1 For every $y \in \Sigma^i$ in lexicographical order:

      2.1.1 Run M_c on <M, y, c>.

      2.1.2 If M accepted , then increment running counter j (on second tape)"

      2.1.3 If j = s, then accept

3. if j ≠ s, then reject

Explanation: The algorithm goes over every possible word that M can accept using at most c cells. That group of possible words only include words that have length of at most c, as words with length greater than c require more than c cells in order to store the word on the tape. Therefore the loop only loops over words of length 0 to c. Every time a word gets accepted by M without passing the c'th cell, we increment a counter, and once the counter reaches the number s, we know we have found s words that fulfill the condition, and so the TM accepts. If by the end of the loop, the counter does not equal s, then we know that we have checked every possible word that could fulfill the condition, but we weren't able to find enough words that did fulfill the condition, and so the TM rejects.

Correctness

1. $\langle M, s, c\rangle \in A \Rightarrow \exists x_1, \ldots, x_s \in \Sigma^*\ s.t\ M$ accepts $x_1, \ldots, x_s$ using at most c cells => there will be s times that step 2.1.2 is true and the counter j is incremented => 2.1.3 is executed and M_A accepts

2. $\langle M, s, c\rangle \notin A \Rightarrow \forall (x_1, \ldots, x_s) \in \Sigma^*$, $M\ does\ not$ accept $(x_1, \ldots, x_s)$ using at most c cells => The amount of words that were accepted by M using at most c cells was a number less than s (possibly zero) => 2.1.2 is true less than s times => j is less than s when the loop stops => step 3 is executed => M_A rejects

[19 pts] **2.** Let $A = \left\{ \langle M_1, M_2\rangle \;\middle|\; \begin{array}{c} M_1\ and\ M_2\ are\ TMs, \forall x \in \Sigma^*\ and\ \forall k \in \mathbb{N} \\ if\ M_1\ halts\ on\ x\ in\ exactly\ k\ steps\ then \\ M_2\ doesn't\ accept\ x\ in\ less\ than\ 2k\ steps \end{array} \right\}$.

Prove that $A \in coRE$.

$\bar{A} = \left\{ \langle M_1, M_2\rangle \;\middle|\; \begin{array}{c} M_1\ and\ M_2\ are\ TMs, \exists x \in \Sigma^*\ and\ \exists k \in \mathbb{N}\ s.t \\ if\ M_1\ halts\ on\ x\ in\ exactly\ k\ steps\ then \\ M_2\ accepts\ x\ in\ less\ than\ 2k\ steps \end{array} \right\}$

To prove $A \in coRE$, we must prove $\bar{A} \in RE$. We will build a recognizer for $\bar{A}$:

M_A = "On input $<M_1, M_2>$:

1. For $i = 0$ to ∞:
   1.1. For $j = 0$ to i:
      1.1.1. For every $y \in \Sigma^j$ in lexicographical order:
         1.1.1.1. For k = 0 to ∞:
            1.1.1.1.1. Run M_1 on y for k steps.
            1.1.1.1.2. If M accepted or rejected, go to step 1.1.1.1.2.1."
               1.1.1.1.2.1. Run M_2 on y for at most 2k-1 steps
               1.1.1.1.2.2. If M_2 accepted, accept

2. Reject"

Explanation:

Run over every word and for every word run over every natural number. For the current word y and natural number k, check if M_1 halts on y in k steps, if it does then check if M_2 accepts y in at most 2k-1 steps, and if so then accept. If no such k and y exist, then M_A will be stuck in an infinite loop and never halt. The "reject" at the end of the algorithm is technically unnecessary but is there for clarity.

Correctness

1. $\langle M_1, M_2 \rangle \in \bar{A} => \exists x \in \Sigma^* \; and \; \exists k \in \mathbb{N} \; s.t \; M_1 \; halts \; on \; x \; in \; exactly \; k \; steps \; and$
$M_2 \; accepts \; x \; in \; less \; than \; 2k \; steps =>$ for at least one word x and number k ,
$M_1 \; halts \; on \; x \; in \; exactly \; k \; steps$ => for a specific word x and number k, 1.1.1.1.2 will be true => for this specific x and k, 1.1.1.1.2.2 will be true => M_A accepts

2. $\langle M_1, M_2 \rangle \notin \bar{A} \; => there \; doesn't \; exist \; a \; word \; x \; and \; natural \; number \; k \; s.t \; M_1 \; halts \; on \; x \; in$
$exactly \; k \; steps \; and \; M_2 \; accepts \; x \; in \; less \; than \; 2k \; steps =>$ for every word x and number k ,
$M_1 \; does \; not \; halt \; on \; x \; in \; exactly \; k \; steps$ => for any word x and number k, 1.1.1.1.2 will be false => the loop will infinitely run => M_A will never halt => M_A does not accept

Problem 5

Given a TM M and string w, we define the following pair of TM's:

$N_{M,w}$ = "On input x:

1. If $x = w \circ w^R$, accept.
2. For $i = 0$ to ∞:
   2.1. For $j = 0$ to i:
      2.1.1. For every $y \in \Sigma^j$ in lexicographical order:
         2.1.1.1. Run M on y for i steps.
         2.1.1.2. If M accepted or rejected, accept."

$P_{M,w}$ = "On input x:

2. If $x = w \circ w^R$, reject.
3. For $i = 0$ to ∞:
   3.1. For $j = 0$ to i:
      3.1.1. For every $y \in \Sigma^j$ in lexicographical order:
         3.1.1.1. Run M on y for i steps.
         3.1.1.2. If M accepted, accept."

[5 pts] **1.** Prove or disprove: $L(N_{M,w})$ is decidable for every M and w.

False, $L(N_{M,w})$ is not decidable for every M and w. Proof by counterexample:

Let M = "On input x:

1. Scan a symbol and move the head to the right

2. Once the head has reached the second blank space after the word, accept"

Let w = "100"

Let the input to $L(N_{M,w})$ be x = "101":

x $\neq w \circ w^R$, therefore we move to step 2 of the algorithm. Step 2 checks for every i, if there exists a word of length <= i s.t M can decide this word in i or less steps. But since, by design, M only accepts a word after going over each symbol in the word, and then also an extra step, it will always decide the word in i + 1 steps, for every i, and therefore $N_{M,w}$ will infinitely run to check for every value of i, but will never accept. Therefore $N_{M,w}$ will never halt. Therefore $N_{M,w}$ is not decidable.

[5 pts] **2.** Given M and w, what is the relation between $L(N_{M,w})$ and $L(P_{M,w})$? That is, is one contained in the other? Could they be equal or none of the above? Does it depend on M and w? Explain shortly.

If M is a TM s.t there exists a word of length <= i that M can decide in i or less steps, then $L(N_{M,w})$ will contain every word as step 2 of the algorithm will always be true and return accept. This means that no matter what, $L(P_{M,w})$ is contained in $L(N_{M,w})$. It is also noteworthy to mention that in step 2 of both algorithms are the same, except for that $P_{M,w}$ has a further restriction/condition that M has to accept y, rather than just decide y like in $N_{M,w}$. If there does not exist such a word (a word of length <= i that M can decide in i or less steps) then $L(N_{M,w})$ will contain the word $w \circ w^R$, and $L(P_{M,w})$. Will be an empty language. Also in this case $L(P_{M,w})$ is contained in $L(N_{M,w})$. In both above cases, the choice of w is irrelevant in terms of the relationship between $L(N_{M,w})$ and $L(P_{M,w})$.

[5 pts] **3.** Prove or disprove: For every M and w, $\langle N_{M,w}, w\rangle \in A_{TM}$ iff $\langle P_{M,w}\rangle \in \overline{E_{TM}}$.

Let M be a TM and w be a word:

If $\langle P_{M,w}\rangle \in \overline{E_{TM}}$, then that means that $L(P_{M,w}) \neq \emptyset$, which means that $P_{M,w}$ accepts at least one word. The only step in the algorithm of $P_{M,w}$ where a word can be accepted is step 3.1.1.2. This means that M accepted in this step, and that there exists a word in the alphabet which M accepts in at most i steps, where i is the length of the word – denote this word as x. In the algorithm of $N_{M,w}$, step 2 is identical to step 3 of $P_{M,w}$, except for a weaker restriction on step 2.1.1.2. This means that when $N_{M,w}$ is run on input w, and in step 2 checks if there exists a word in the alphabet which M accepts **or rejects** in at most i steps (where i is the length of the word), we already know that there exists such a word – word x. Therefore $N_{M,w}$ accepts in this case. Therefore $\langle N_{M,w}, w\rangle \in A_{TM}$

If $\langle N_{M,w}, w\rangle \in A_{TM}$, that means that $N_{M,w}$ accepted word w. We know that w has to have been accepted in step 2.1.1.2, as it couldn't have been accepted in step 1 (w $\neq w \circ w^R$). This means

that there exists a word in the alphabet which M accepts or rejects in at most i steps, where i is the length of the word – denote this word as x. I don't know where to go from here.