Theory of Computation

PSet-4

05 November 2025

This is a compilation of solutions for PSet-4 of Theory of Computation class instructed by Prof. C Aishwariya. The solutions are of original design.

Table of contents

§0. Problems

1. Consider the following language: $\{\langle M\rangle \mid L(M) \text{ is regular}\}$

   (a) Is it Recursively Enumerable?

   (b) Is it Co-Recursively Enumerable?

   Justify.

2. A linear bounded automaton (LBA) is exactly like a one-tape Turing machine, except that the input string $x \in \Sigma^*$ is enclosed in left and right endmarkers $\vdash$ and $\dashv$ which may not be overwritten, and the machine is constrained never to move left of the $\vdash$ nor right of the $\dashv$. It may read and write all it wants between the endmarkers. (two-way, read/write)

   (a) Give a rigorous formal definition of deterministic linearly bounded automata, including a definition of configurations and acceptance. Your definition should begin as follows: "A deterministic linearly bounded automaton (LBA) is a 9-tuple

   $$M = (Q, \Sigma, \Gamma, \vdash, \dashv, \delta, s, t, r)$$

   where Q is a finite set of states..."

   (b) Let M be a linear bounded automaton with state set Q of size k and tape alphabet Γ of size m. How many possible configurations are there on input x, $|x| = n$?

   (c) Argue that the halting problem for deterministic linearly bounded automata is decidable. (Hint: You need to be able to detect after a finite time if the machine is in an infinite loop. Presumably the result of part (b) would be useful here.)

   (d) Prove by diagonalization that there exists a recursive set that is not accepted by any LBA.

§1. Problem 1

Exercise

Consider the following language: $\{\langle M\rangle \mid L(M) \text{ is regular}\}$

(a) Is it Recursively Enumerable?

(b) Is it Co-Recursively Enumerable?

Justify.

(a)

Proof. FTSOC, let there be a turing machine R that can recognize if another turing machine's identified language is regular. We will construct a turing machine S which can recognize pair (M, w) where M is a turing machine that accepts w.

Consider M' which accepts x if it is of the form $\{0^n1^n \mid n \in \mathbb{N}\}$ or if M accepts w. This exists as every CFL can be decided by a turing machine and $\{0^n1^n \mid n \in \mathbb{N}\}$ is a CFL; and as union of two Turing recognizable languages is a Turing recognizable.

S outputs the output obtained by running R on input $\langle M'\rangle$.

We claim we have arrived at a contradiction as if M accepts w then $L(M') = \Sigma^*$ as the second condition is always met and hence, M' accepts all x and similarly, if M doesn't accept w, then $L(M') = \{0^n1^n \mid n \in \mathbb{N}\}$ as the second condition is never met and hence, M' only accepts $x \in \{0^n1^n \mid n \in \mathbb{N}\}$.

Thus, R accepts M' if and only if M accepts w.

As we know that the acceptence problem is undecidable by turing machine, we have a contradiction.

Thus, $\{\langle M\rangle \mid L(M) \text{ is regular}\}$ is not recursively enumerable. □

(b)

Proof. FTSOC, let $\{\langle M\rangle \mid L(M) \text{ is not regular}\}$ be recursively enumerable. Let that be done by a Turing machine R. We will construct a turing machine S which can recognize pair (M, w) where M is a turing machine that accepts w.

Consider M' which accepts x if it is of the form if it is of the form $\{0^n1^n \mid n \in \mathbb{N}\}$ or if M accepts w.

S outputs the output obtained by running R on input $\langle M'\rangle$.

We claim we have arrived at a contradiction as if M accepts w then $L(M') = \Sigma^*$ as the second condition is always met and hence, M' accepts all x and similarly, if M doesn't accept w, then $L(M') = \{0^n1^n \mid n \in \mathbb{N}\}$ as the second condition is never met and hence, M' only accepts $x \in \{0^n1^n \mid n \in \mathbb{N}\}$.

Thus, R accepts M' if and only if M doesn't accept w.

s we know that the acceptence problem is undecidable by turing machine, we have a contradiction.

Thus, $\{\langle M\rangle \mid L(M) \text{ is not regular}\}$ is not recursively enumerable. □

§2. Problem 2

Exercise

A linear bounded automaton (LBA) is exactly like a one-tape Turing machine, except that the input string $x \in \Sigma^*$ is enclosed in left and right endmarkers $\vdash$ and $\dashv$ which may not be overwritten, and the machine is constrained never to move left of the $\vdash$ nor right of the $\dashv$. It may read and write all it wants between the endmarkers. (two-way, read/write)

(a) Give a rigorous formal definition of deterministic linearly bounded automata, including a definition of configurations and acceptance. Your definition should begin as follows: "A deterministic linearly bounded automaton (LBA) is a 9-tuple

$$M = (Q, \Sigma, \Gamma, \vdash, \dashv, \delta, s, t, r)$$

where Q is a finite set of states..."

(b) Let M be a linear bounded automaton with state set Q of size k and tape alphabet Γ of size m. How many possible configurations are there on input x, $|x| = n$?

(c) Argue that the halting problem for deterministic linearly bounded automata is decidable. (Hint: You need to be able to detect after a finite time if the machine is in an infinite loop. Presumably the result of part (b) would be useful here.)

(d) Prove by diagonalization that there exists a recursive set that is not accepted by any LBA.

(a)

Solution. A deterministic linearly bounded automaton (LBA) is a 9-tuple

$$M = (Q, \Sigma, \Gamma, \vdash, \dashv, \delta, s, t, r)$$

where

- Q is a finite set of states.
- Σ is a finite set of input alphabets.
- Γ is a finite set of tape alphabets.
- $\vdash \in \Gamma - \Sigma$ is the left endmarker.
- $\dashv \in \Gamma - \Sigma$ is the right endmarker.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
- $s \in Q$ is the start state
- $t \in Q$ is the accept state
- $r \in Q$ is the reject state

□

(b)

Solution. A configuration needs us to define which state we are on, what is the string on the tape and where is the machine read/write-ing.

As we have k states, m tape alphabets and a string of length n; we could have $k * n^m * (n + 2)$ configurations as the head could be on either of the end markers as well. □

(c)

Proof. As the LBA is deterministic, Given input x with length n, the total number of possible configurations is $l = |Q|\ n^{|\Gamma|}(n+2)$. So if the machine runs for more than l steps, by pigeonhole principle, we must have repeated a configuration and hence are stuck in an infinite loop.

Thus, to decide whether M halts on x: We can simulate M on x for l steps. If it halts within the steps, we output "yes" and if it doesn't, we output "no".

Thus, the halting problem for LBA is decidable. □

(d)

Proof. Using the scheme similar to the one for Turing Machines, we can encode LBA with alphabet $\{0, 1\}$.

Using the lexicographic ordering on all the LBA and input word; and as the Halting Problem for deterministic LBA is decidable; thus, the function

$$f(i,j) = \begin{cases} 1 \text{ if } M_i \text{ accepts } w_j \\ 0 \text{ otherwise} \end{cases}$$

is computable.

Consider the language $L = \{w_i \mid M_i \text{ doesn't accept } w_i\}$. Notice, L is recursive as given w, we can compute i; thus, compute M_i and thus, check if w_i is identified by M_i.

But, L is not accepted by any LBA. FTSOC, let there exist some M_j which accepts L.

If M_j accepts $w_j \Rightarrow w_j \in L$, but by the definition of L, then $w_j \notin L$.

Similarly, if M_j doesn't accept $w_j \Rightarrow w_j \notin L$, but by the definition of L, then $w_j \in L$.

Thus, we have a contradiction. Thus, no such M_j exist.

Thus, there exists a recursive set that is not accepted by any LBA. □