CS21204 : Formal Language and Automata Theory (Spring 2026)

Class Test 2

26-Mar-2026 (Thursday) Maximum Marks: 30 06:30pm – 07:45pm

Roll: ____________________ **Name:** ____

[*Write your answers in question paper. Answer all questions. Be brief and precise.*]

Q1. [Turing Machines]

Consider the language $L = \{a^{n^2} \mid n \geq 1\}$ defined over the alphabet $\Sigma = \{a\}$. Describe a (total) Turing machine that decides the language L. **(7)**

Solution:

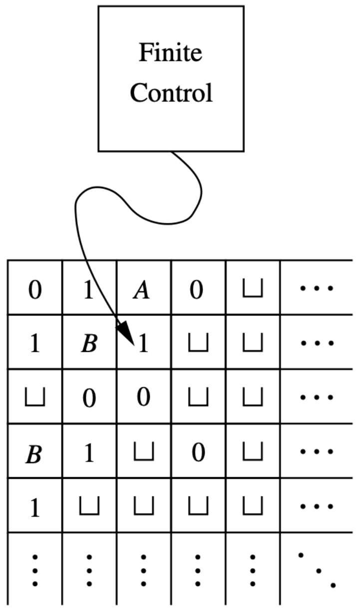
We now describe a (total) Turing machine M that decides L. M has two tapes. The first tape contains the input string (a string of the form a^k). M has to test whether k is a perfect square or not. Let X be a symbol other than a, $\vdash$ and $\sqcup$. The second tape initially contains 1 X followed by blanks. (Both tapes have a left endmarker in the left-most cell.)

M repeats the following.

- M advances tape-head 1 (reading one a) to the right and moves tape-head 2 one cell to the right reading an X.
- If both tape-heads have reached blank symbol, then accept and halt.
- If tape-head 1 reaches blank and tape-head 2 is pointing to an X, reject and halt.
- If tape-head 1 is reading a and tape-head 2 is reading blank, then write two more X's on tape-2 at the end of the existing string of X's.
- Move tape-head 2 to the left-end so that it points to the first occurrence of X.

Essentially the machine maintains an odd number of X's on tape 2: 1 initially, 3 in the second iteration, 5 in the third iteration, and so on. Tape-head 1 never moves backwards; instead it moves one step forward reading a for every occurrence of X on the second tape and this repeats for every odd-length string of X's genrated on tape-2. M checks if the number of a's in tape 1 is a sum of consecutive odd numbers (starting from 1). If so, the number of a's must be a perfect square, and vice-versa.

Q2. [Equivalent TM Models and R.E. Languages]

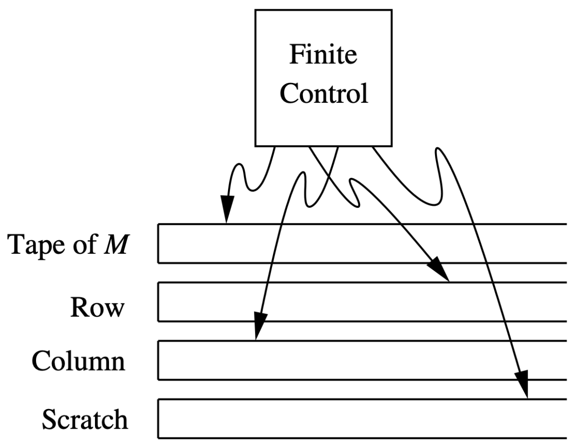


Consider a Turing machine M with a two-dimensional tape as shown in the adjacent figure. The initial input string is provided horizontally at the topmost row starting from the first column, the finite control of M is in the start state, and the head is positioned at the top-left cell of the tape. Each move of M is dependent on the state of the finite control and on the tape symbol scanned by the head. During each move the finite control switches to a new state (which may be the same as the old state), the tape symbol currently scanned by the head is overwritten by a new symbol (which may be the same as the old symbol), and the head moves by one cell in any one of the four directions north, east, west and south. M accepts by entering a final state.

Show that $\mathscr{L}(M)$ is R.E. (i.e., Turing-recognizable). **(10)**

Solution:

Let us design a TM N with four semi-infinite one-dimensional tapes such that $\mathscr{L}(N) = \mathscr{L}(M)$. Since $\mathscr{L}(N)$ is R.E., the result follows. The working of $\mathscr{L}(N)$ is described schematically as shown in the diagram below.



The two-dimensional tape of M is simulated in the first tape of N. Let us plan to use the bijection $\varphi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\varphi(i, j) = 1 + \frac{1}{2}\left[(i+j)^2 - i - 3j\right]$, for this purpose. If we number the cells of the tape of M by the pair $(i, j) \in \mathbb{N} \times \mathbb{N}$, where i denotes the row number and j denotes the column number, then the (i, j)-th cell of M is stored in the $\varphi(i, j)$-th cell of the first tape of N.

The second and third tapes of N hold the coordinates (i.e., the position (i, j)) of the head of M. A simple way to do this is to store the strings 0^i and 0^j.

A scratch tape is used for intermediate calculations. If necessary one may go for more than one (but only finitely many) scratch tapes. Let us assume that only one scratch tape is sufficient (in fact, it is!) and makes it easy enough to handle all the calculations that will be described below.

Q2 Solution (continued):

The input string $\alpha := a_1 \ldots a_n$ is provided to N on Tape 1 starting from the leftmost cell of that tape. In order to simulate the initial configuration of M, N first relocates a_i to the $\varphi(1,i)$-th cell for all $i = 1, \ldots, n$. The scratch tape (or tapes) can be used for these relocations. All other cells of the first tape of N are made empty. Then M's first head is positioned at the first cell of the first tape. N's finite control mimics the states of M, i.e., N initializes its finite control to the start state of M.

In order to simulate a move of M, N scans its first tape and also consults its current state (which is the same as M's current state). If M does not specify a transition in this situation, N halts without accepting. Otherwise N simulates M's (unique) move as follows:

- N changes its state as M would do.
- N replaces the symbol of the cell scanned by its first head as M would do.
- Depending on the move of M from the (i, j)-th cell to the (i', j')-th cell, N adjusts its first head and the contents of its second and third tapes as follows. The new coordinates $(i', j')$of M's head is related to the old coordinates (i, j) as follows:

Move of M	(i', j')	$\varphi(i', j') - \varphi(i, j)$
North	$(i-1, j)$	$-(i+j-1)$
East	$(i, j+1)$	$+(i+j-1)$
West	$(i, j-1)$	$-(i+j-2)$
South	$(i+1, j)$	$+(i+j)$

  The first head of N should be moved right by $\varphi(i', j') - \varphi(i, j)$ cells. (A negative value of $\varphi(i', j') - \varphi(i, j)$ indicates a left movement.) N reads i and j from its second and third tapes and computes in its scratch tape $\varphi(i', j') - \varphi(i, j)$ as shown in the third column of the above table. It then moves its first head by an appropriate amount and in the appropriate direction. It then replaces the contents of its second and third tapes from 0^i and 0^j to $0^{i'}$ and $0^{j'}$. For all the moves of M's head one of i and j remains unaltered, the other either increases or decreases by 1. Thus it is only necessary to add or delete one 0 from one of the second and third tapes.
- Finally N checks if the updated state is an accepting state for M. If so, N accepts and halts. If not, N simulates the next move of M.

It is clear from the above description that N accepts a string α if and only if M accepts α and therefore $\mathscr{L}(N) = \mathscr{L}(M)$. One can convert N to a standard one-tape one-head TM with a one-dimensional tape (infinite in both directions) using the procedures described in the lectures. Thus $\mathscr{L}(N) = \mathscr{L}(M)$ is R.E. or a Tuing-recognizable language.

Q3. [Properties of Recursive Languages]

(a) Prove that Turing-recognizable (R.E.) languages are closed under union. **(4)**

Solution:

Let $L_1 = \mathscr{L}(M_1)$ and $L_2 = \mathscr{L}(M_2)$ be two Turing-recognizable (R.E.) languages, where M_1 and M_2 are TMs. Let us design a TM M with $\mathscr{L}(M) = L_1 \cup L_2$. M runs M_1 and M_2 in "parallel". M may be designed as a two-tape machine. To start with M copies the input α to the second tape and positions both the heads at the beginning of the respective copies of the input. M simulates the behavior of M_1 on the first tape and that of L_2 on the second tape *simultaneously*. M accepts if and only if either the simulation of M_1 or that of M_2 accepts. If both M_1 and M_2 reject α after halting, then M also rejects. If one of M_1 and M_2 rejects after halting and the other by not halting, or if both M_1 and M_2 reject by not halting, then M rejects by not halting.

(b) Let $L_1, L_2, \ldots, L_n$ be pairwise disjoint Turing-recognizable (R.E.) languages over the same alphabet Σ. Suppose that $\bigcup_{i=1}^{n} L_i = \Sigma^*$. Prove that each L_i is Turing-decidable (Recursive). **(4)**

Solution:

From **Q3(a)**, Turing-recognizable (R.E.) languages are closed under finite union. It follows that $\bigcup_{\substack{1 \le j \le n \\ j \ne i}} L_j = \Sigma^* \setminus L_i$ is Turing-recognizable (R.E.), that is, $\overline{L_i}$ is Turing-recognizable (R.E.). Since L_i is also Turing-recognizable (R.E.), it follows that L_i is Turing-decidable (Recursive).

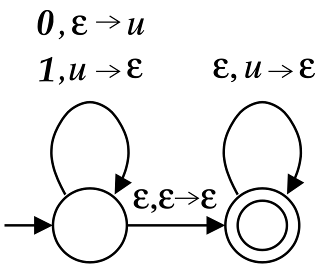
Q4. [Push-Down Automaton]

Let $\alpha = a_1 a_2 \dots a_n$ be a string of length n. A string β is called a prefix of α if $\beta = a_1 a_2 \dots a_i$ for some $i \in \{0, 1, 2, \dots, n\}$ (the case $i = 0$ corresponds to $\beta = \varepsilon$). Consider the following language,

$$L_0 = \left\{ \alpha \in \{0,1\}^* \;\middle|\; \text{no prefix of } \alpha \text{ contains less 0's than 1's} \right\}$$

Prove that L_0 is context-free by formally presenting a push-down automaton (PDA) that accepts L_0. **(5)**

Solution:



— End of Question Paper —