Tutorial 6
Turing Machines, Recursive and R.E. Sets

1. Design a total Turing machine that accepts the set $\{0^{2^i} \mid i \geq 1\}$.

   Solution: Here is an informal description for the Turing machine accepting strings of 0's of length a positive power of 2.

   (i) Make a pass through the input string crossing off every other uncrossed 0.

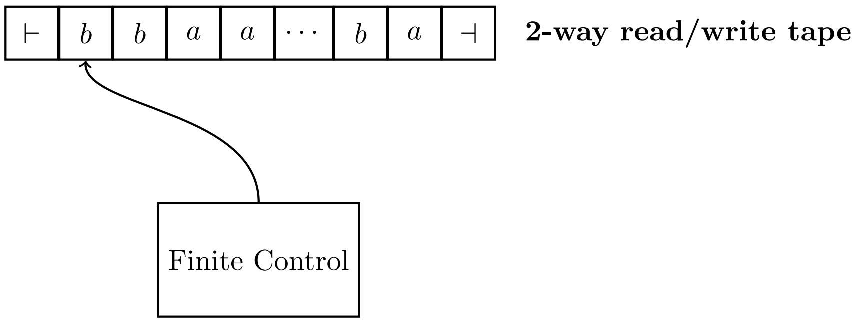
   (ii) If after the pass there are uncrossed 0's to the right of the last crossed 0 on the tape, then reject.

   (iii) If at the beginning of the pass, only 1 0 (the first one) is uncrossed, then accept.

   (iv) Go to step (i).

   Write down the formal description yourself.

2. A linear bounded automaton (LBA) is exactly like a 1-tape TM, except that the input string $x \in \Sigma^*$ is enclosed in left and right endmarkers $\vdash$ and $\dashv$ which may not be overwritten. The machine is constrained never to move left of $\vdash$ or right of $\dashv$. It is allowed to read/write between these markers.



   (a) Give a rigorous formal definition of deterministic linearly bounded automata, including a definition of configurations and acceptance.

   Solution: An LBA is given by a 9-tuple, $(Q, \Sigma, \Gamma, \delta, \vdash, \dashv, s, t, r)$ where

   - Q is the set of states
   - Σ is the input alphabet
   - Γ is the tape alphabet
   - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function

- $\vdash, \dashv \in \Gamma \setminus \Sigma$ are the left and right endmarkers respectively
- s, t, r are respectively the start, accept and reject states

Restrictions that ensure that the Turing machine never overwrites the endmarkers and never move left or right of the left and right endmarkers are given below.

- $\forall p \in Q$, $\exists q \in Q$ such that $\delta(p, \vdash) = (q, \vdash, R)$.
- $\forall p \in Q$, $\exists q \in Q$ such that $\delta(p, \dashv) = (q, \dashv, L)$.

Define a configuration of $\mathcal{M}$ to be a string in $Q \times \Gamma^* \times \mathbb{N}$ containing the current state, contents of the tape and position of the tape head. The start configuration on input x is $(s, \vdash x \dashv, 0)$. Also define the next configuration relation $\xrightarrow[\mathcal{M}]{1}$ as follows.

$$(p, z, n) \xrightarrow[\mathcal{M}]{1} \begin{cases} (q, s_b^n(z), n+1) \text{ if } \delta(p, z_n) = (q, b, R) \\ (q, s_b^n(z), n-1) \text{ if } \delta(p, z_n) = (q, b, L) \end{cases}$$

Let $\xrightarrow[\mathcal{M}]{*}$ be the reflexive transitive closure of $\xrightarrow[\mathcal{M}]{1}$. The machine $\mathcal{M}$ is said to accept a string x if $(s, \vdash x \dashv, 0) \xrightarrow[\mathcal{M}]{*} (t, \vdash y \dashv, n)$ for some $n \in \mathbb{N}$ with $n \leq |x| + 1$ and some $y \in \Gamma^{|x|}$.

(b) Let $\mathcal{M}$ be an LBA with state set Q of size k and tape alphabet Γ of size m. How many possible configurations are there on input x with $|x| = n$?

Solution: Let $[n_1, n_2]$ for $n_2 > n_1$ denote the set of integers $n_1, n_1 + 1, \ldots, n_2$. There are a total of m possible symbols that can be written on tape cells $1, \ldots, n$. At positions 0 and $n+1$, the symbols are fixed. The LBA can be in k possible states. The tape head can be at one of $n+2$ positions. Hence, the set of all possible configurations is given by $[0, n+2] \times Q \times (\Gamma \setminus \{\vdash, \dashv\})^n$. The number of configurations possible is therefore $(n+2) \cdot k \cdot (m-2)^n$.

(c) Argue that it is possible to detect in finite time whether an LBA loops on a given input.
Hint: Use part (b).

Solution: Run the given LBA for $k(n+2)(m-2)^n + 1$ steps. The automaton either halts or not. If it does not, atleast one configuration repeats; then output that the LBA loops. Correctness follows from the fact that the LBA is deterministic.

3. Show that the class of recursively enumerable sets is closed under union and intersection.

Solution: Let A, B be *r.e.* sets and let $\mathcal{M}_A, \mathcal{M}_B$ be the respective Turing machines recognising them. Consider the TM $\mathcal{N}$ that does the following on input x.

- Run $\mathcal{M}_A$ on x and $\mathcal{M}_B$ on x in a round-robin fashion.
- Accept if either $\mathcal{M}_A$ or $\mathcal{M}_B$ accepts.

Clearly if $x \in A \cup B$, either $\mathcal{M}_A$ or $\mathcal{M}_B$ accepts it causing $\mathcal{N}$ to accept. $\mathcal{N}$ is a valid TM and hence $A \cup B$ is *r.e.*

Next, let $\mathcal{K}$ be a TM that, on input x, does the following.

- Run $\mathcal{M}_A$ on x.
- If $\mathcal{M}_A$ accepts, then run $\mathcal{M}_B$ on x.
- Accept if $\mathcal{M}_B$ accepts.

$\mathcal{K}$ accepts x only if both $\mathcal{M}_A$ and $\mathcal{M}_B$ accept x and hence $L(\mathcal{K}) = A \cap B$. Note that $\mathcal{M}_A$ could loop on x in which case x is clearly not in A or $A \cap B$. So, it does not matter if we do not check whether or not $\mathcal{M}_B$ accepts x.

4. Prove that a language L is recursive if and only if there is an enumeration machine enumerating the strings of L in a non-decreasing order of length (string of the same length may be arranged in lexicographic order). For example, strings of $0, 1^*$ would be arranged as $0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots$.

   Solution: Suppose that $L \subseteq \Sigma^*$ is a recursive language. Then there exists a total TM $\mathcal{K}$ deciding L. Construct an enumeration machine $\mathcal{E}$ that does the following for each string y in Σ^* according to non-decreasing order of length.

   - Simulate $\mathcal{K}$ on y.
   - Enumerate y if $\mathcal{K}$ accepts y
   - If $\mathcal{K}$ rejects y, then do not enumerate.

   Since $\mathcal{K}$ is total, the enumeration of every string in L occurs after finitely many steps.

   Conversely, suppose that there is an enumeration machine $\mathcal{E}$ enumerating strings of a language L in non-decreasing order length. Define a TM $\mathcal{K}$ that does the following on input $y \in \Sigma^*$,

   - Run $\mathcal{E}$ until it enumerates y or a string of length greater than $|y|$.
   - If y is enumerated, then accept and halt.
   - Otherwise, reject and halt.

   $\mathcal{E}$ should eventually enumerate y if $y \in L$ or a string longer than y. The latter happens when $y \notin L$. As a result, in finite time $\mathcal{K}$ determines if $y \in L$ or not. Therefore, $\mathcal{K}$ is total and L is recursive.

5. Is the set $\{\mathcal{M} \mid \mathcal{L}(\mathcal{M})$ contains atleast 100 elements$\}$ recursively enumerable?

   Solution: This set is *r.e.* It is possible to construct a TM that runs a given $\mathcal{M}$ all strings in a round-robin fashion. If atleast 100 strings are accepted by $\mathcal{M}$, then halt and accept. Otherwise keep testing for other stings.