Problem Set I

- Construct a Turing machine that takes as input a binary number n as input and prints $n-1$ in binary. (if 0 is given as input, output may be 0).
- Show that every finite language is Turing decidable.
- Suppose L, L' are both Turing decidable, is it always true that $L \leq_m L'$.
- Suppose L is Turing decidable, show that L^* is Turing acceptable.
- Suppose L is Turing decidable, show that $L^2 = \{xy : x, y \in L\}$ is Turing decidable. Argue similarly that L^k is Turing decidable for each positive integer k.
- Show that given a machine M deciding L, you can design an algorithm A that produces on input positive integer k, an output machine M_k such that M_k decides L^k.
- Using the above observations (you can use the algorithm A as a subroutine) show that L^* is decidable if L is decidable. (Hint: You need to design an algorithm B that on input x decides whether x is accepted by M_k for some k. The favourable observation is that maximum value of k can be bounded once x is given).
- A non-deterministic Turing machine is for which $\delta : Q \times \Sigma \mapsto 2^{Q\times\Sigma\times\{L,R,-\}}$. That given a state and tape position, the machine may have multiple transitions possible and the machine is free to choose any possibility. The definition of acceptance is not changed. That is, if M is a non-deterministic Turing machine, M accepts $x \in \Sigma^*$ if and only if $(\underset{q_0}{\#}x\#) \vdash_M^* (\underset{q_A}{\#}\#)$. However, from a given configuration, the machine may move into different configurations and hence there are several possible runs for M on the same input x. Hence, the definition of acceptance above stipulates that M accepts x if at least **one of the possible runs** reach the accepting configuration. (This is similar in spirit to the definition of acceptance for finite automata.) The machine is said to be a **decider** if all possible runs on all possible inputs always end up either an accepting configuration or a rejecting configuration. That is, the whatever be the non-deterministic choices made by the machine, M never runs in an infinite loop on any input. (Note: A non deterministic machine may have one just run on some input x that is accepting and all other runs rejecting. In such case, the definition of acceptance says x is still accepted by M).
  1. Argue that any language accepted by a non deterministic Turing machine can also be accepted by a deterministic Turing machine. (Read any text book for details.) Thus just as in the case of finite state machines, non-determinism does not add more computing power.
  2. Given a machine M that decides a language L, show that it is easy to design a non-deterministic decider for L^*.
  3. Show that L is Turing decidable (in the standard sense) if and only if L has a non-deterministic decider.
- An enumerating Turing machine M with some particularities: 1. M starts with the blank tape (that is it is assumed that M does not take any inputs). 2. From the state q_A, the machine may have other outgoing transitions (so the machine doesn't halt upon reaching q_A). However, the machine does halt on reaching q_R. 3. The machine is assumed to be designed in such way that the head points to a blank symbol whenever it reaches q_A. 4. Suppose M reaches state q_A first time, we look at what is the string written to the right of the blank symbol (till the next blank on the right). We say this string is an output string generated by the machine. 5. The machine may continue to run from there (because of condition 2) and after some time reach state q_A again. In that case, the string found to the right of the blank this time too is a string generated by the machine. 6. The machine go on this way and may visit q_A infinitely many times and each string that is found to the right of the head at any time the machine reaches q_A is added to the set of strings generated by the machine. Formally stated, the language enumerated by the machine M (with the above programming conventions) is defined as

$L_E(M) = \{x : \#\underset{q_0}{\#}\# \vdash_M^* \underset{q_A}{\#}x\#\}$. Intuitively, an enumerating Turing machine is the mathematical abstraction for a program that takes no inputs, but prints a set of strings (possibly running in an infinite loop). The language enumerated by such a program is defined as the set of all output strings generated by the program.

1. Argue that a program L is Turing acceptable if and only if there is a Turing machine that enumerates all strings of the language. (Of course, no other strings must be enumerated).
2. Argue that a language L is Turing decidable if and only if there is a Turing machine that enumerates all strings of the language in the following order: a) String of length $i+1$ are generated only after all strings of length i are generated. b) Among strings of the same length, dictionary order is followed.

- Given two Turing machines M_1 and M_2, show that the problem of deciding whether $L(M_1) \cap L(M_2) = \emptyset$ is undecidable.
- Given two Turing machines M_1 and M_2, show that the problem of determining whether $L(M_1) \subseteq L(M_2)$ is undecidable.