CSCI 1010 - Fall'25 Theory of Computation Lorenzo De Stefani

HW5

Due: n/a

Reminder: Submit your assignment on Gradescope by the due date. Submissions must be typeset. Each page should include work for only one problem (i.e., make a new page/new pages for each problem). See the course syllabus for the late policy.

While collaboration is encouraged, please remember not to take away notes from any labs or collaboration sessions. Your work should be your own. Use of other third-party resources is strictly forbidden.

Please monitor Ed discussion, as we will post clarifications of questions there.

Problem 1

Explain why the following is not a description of a legitimate Turing Machine.

$M_{bad} =$ "The input is a polynomial p over variables $x_1, ..., x_k$.

1. Try all possible settings of $x_1, ..., x_k$ to integer values
2. Evaluate p on all of these settings.
3. If any of these settings evaluates to 0, accept; otherwise reject."

Solution: Here is a brief justification. Consider a polynomial p such that p has no integer roots. Note that there are infinite possible settings of $x_1, \ldots, x_k$. On any given evaluation, we will reject, and we cannot stop early if we reject since we may have to evaluate p for another setting. Thus, evaluating p on all of these settings will take an infinite amount of time. Thus, we are unable to reject after seeing all of these settings, because seeing all of these settings takes an infinite amount of time.

Problem 2

1. Consider a variant Turing machine in which the head of the reader can either stay put or move to the right.
   (a) Describe why this is not equivalent to a normal Turing machine.

(b) What class of languages does this machine recognize? Briefly describe why.

2. Consider a variant Turing machine that has three independent heads. At each step, the Turing machine can choose which of the three heads to move. Only one head can move in each step. Each head is still not allowed to move left. Prove that such a machine can simulate a normal Turing machine.

Solution Part One The machine can never actually store information since it cannot go back once something is written. Writing to the tape only provides constant additional working room beyond it's own encoding.

Solution Part Two This machine operates as a DFA. It can read the input plus use it's own encoding as a state machine. It can write its end state to the tape when finished.

Solution Part Three We will name the three heads the read-head, the print-head and the working-head. The working-head on the tape will simulate a normal Turing machine. If it moves right or stays put there is no problem. If it needs to move left, it will perform the following routine:

The read-head will scan the tape. The print-head will print out a copy a new copy of the entire tape to the end. Then the working-head will move until it is to the left of where the working-head was on the newly printed copy of the tape.

We will prove this always works with the following lemma:

Assume the read-head is at the start of the tape. The read-head is therefore capable of reading all characters written to the tape. The print-head can be anywhere, as it just has to move to the end of the tape to print. After the procedure, the read-head will be at the start of the newly printed contents of the tape after having scanned the the last sections contents (and then stopping). Since all characters to its left have been passed by all three heads, without loss of generality we can consider it to be "deleted". Therefore after the procedure, the read-head is again at the start of the tape.

At the start of machines operations the lemma's conditions trivially hold. After running the routine, the tape is identical to the previous contents of the tape, except its head is one space to the left. (Remember we've reset the start of the tape effectively deleting all old characters.) After running the subroutine the machine is still in a state where the conditions for the lemma hold. Therefore we have not only simulated moving the working-head to the left once, but we can repeatedly do so.

Problem 3

Given an arbitrary Turing decidable language L', consider the following language

$$L = \{w | \exists w' \in L' \text{ such that } w \text{ is a substring of } w'\}$$

1. Prove that L is Turing recognizable.
2. Without providing rigorous proof, give an intuitive argument for whether you believe L to be decidable.
3. Suggest a property that, if L' were to have it, would guarantee L to be decidable. Motivate your answer.

Solution

1. Turing-recognizability

   By definition of Turing decidability, there exists some TM M' that decides over language L'. Given M', we can construct a TM M that recognizes (but not necessarily decides) over L as follows: M simulates M' it by treating what's the original input string as the input string to M'. If the simulation reveals that M' accepts the string, M also accepts the string. If the simulation reveals that M' rejects the string, M appends a character from the alphabet onto the string, and simulates M' running on this new string. If the simulation reveals that M' accepts the string M also accepts the string. If the simulation reveals that M' rejects the string, M replaces the previously appended character with a different character and re-simulates M' on this new string. This process is repeated until all characters in the alphabet have been tried.

   If all characters have been used, M reverts to the original string, and tries prepending each character this time. If nothing is accepted, it goes back to appending, this time appending two characters. Then, it prepends one character and appends two characters, and lastly prepends two characters and appends two characters. M repeats this process until a string is eventually accepted.

2. Turing-decidability

   It is not decidable because the language L' may be infinite. If it is infinite, for a string that is not a substring, M is bound to run forever since its termination condition is when a superstring is found.

3. Restricting L'

   There are multiple possible answers.

   - Limit the length of strings in L'
   - Make L' regular