CS 4510: Automata and Complexity

Spring 2015

Home work 5 // Due: Friday, April 3, 2015

Sagar Laud, slaud3, 902792910

1. (a) (8 points) Show that $L(G)$, where G is a context-free grammar, is infinite iff G generates a string whose length is at least p, where p is the pumping length identified in the proof of the pumping lemma.

   This is not a difficult proof. We know that G is a CFG. So, it can be pumped. So, if we take an arbitrary string from the language of length atleast p, regardless of what p is, we can pump the string up infinitely using the first condition of the pumping lemma. For this reason, the language itself is infinite.

   (b) (7 points) Let $INFINITE_{\text{PDA}} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language}\}$. Show that $INFINITE_{\text{PDA}}$ is decidable.

   This is done by defining by defining a Turing Machine which decides this. We will design a Turing Machine will traverse the PDA and convert it to it's corresponding CFG. Convert the CFG into Chomsky Normal Form and perform BFS on it (this can be written formally if needed). Look for any repetition in the derivation. That is, at any point, does a given variable V eventually derive a rule with V in it? If so, accept, otherwise halt and reject.

2. (15 points) A *Turing machine with stay put instead of left* is similar to an ordinary Turing machine, but the transition function has the form

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{R, S\}.$$

At each point the machine can move its head right or stay in the same position. Show that this Turing machine variation is *not* equivalent to the usual version. What class of languages do these machines recognize? Justify your answer.

These machines recognize the class of regular languages. We will show this by showing a bidirectional relationship between this form of a Turing Machine and a DFA. Naturally we can simulate every DFA on a Turing Machine. The only modification that we need to make is to add transitions from any accept states in the DFA to q_{accept} and from all other states to q_{reject} upon reading a blank space. We have shown one direction so now we must show the other. We must transform this Turing Machine into a DFA. We will make two key modifications here to make this transformation simpler. We will add a symbol to our tape alphabet and will use this to denote a blank space (specifically when we write) and when we read, we will treat it as if we have just read a blank space. Secondly, we will force the reading head to move to the right and specifically never stay put when we enter our accept or reject state.

After these modifications, we can formally define our DFA.

Let us define M = $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.
Our DFA D = $(Q', \Sigma', \delta', q_0', F)$.

- Q' = Q
- $\Sigma' = \Sigma$
- $\delta(q, a)$ is defined as follows
  - $\delta(q, a) = q$ if q is an accept or reject state
  - $\delta(q, a) = q_{reject}$ if there is looping upon reading a
  - $\delta(q, a) = p$ where p is state which M enters upon reading a and moving right
- $q_0' = q_0$
- F = all states in (Q - q_{reject}) such that if M is in q and reads blanks, it will eventually enter q_{accept}

3. (15 points) Show that the complement $\bar{E_{\text{TM}}}$ of E_{TM} is Turing-recognizable. Explain why from this we can conclude that E_{TM} is not Turing-recognizable.

   We can show this rather easily. We will do this by showing an algorithm that a Turing Machine can use to recognize the complement of our language. The complement of our language is the situation that M is a Turing Machine and L(M) is not the empty set. So, if any is an element of our language, M will accept. We will use a form of iterative deepening rather than pure sequential testing because this may result in us looping forever on a string which could throw a wrench in our plans. So, let us take a list of strings in our language. Let a, b, c, d, ... etc all be elements of Σ^*. Now for each string that is an element of the language, run an iterative deepening algorithm. For each i that we have where i $\geq$ 1, run each string on our machine M for i steps. This removes the issue of looping. If any string accepts, then the machine accepts. This allows us to conclude that E_{TM} is not recognizable. If it was recognizable, then $\bar{E}_{TM}$ would be decidable, therefore making E_{TM} decidable. We know that E_{TM} is not decidable, therefore this is not possible.

4. (15 points) On input M, w (where M is the code of a Turing machine and w is an input to M), a reduction machine constructs the Turing machine M' described below:

   On input x:

   (a) Simulate M on w for $|x|$ steps (that is, it erases one symbol of x for each step of M on w that it simulates).

   (b) Accept if M has *not* halted within that time, reject otherwise.

   What is $L(M')$? Why?

   $L(M')$ is not well defined. We do not know when the machine will actually halt, specifically, a solution to this relies on a solution to the halting problem. As a result, a simple answer to this is not quite possible. If M never halts on w, then L(M') = Σ^*. However, if the machine does halt at any point, then the language is the set of all strings in Σ^* whose length is less than that of our input m. This is quite a discrepency however as if M never halts, then L(M') is infinite. However if it does halt, then L(M') is finite! Since we do not know whether or not the machine will halt, we cannot define a language for this machine. All we can say is that if the machine does not halt, then L(M') = Σ^*. If it does halt, then L(M') is the set of all strings in Σ^* such that the length of any given string is less than the length of w.

Sagar Laud, slaud3, 902792910

5. Let $FINITE3_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a } 1-\text{tape DTM and } L(M) \text{ has } \textit{exactly} \text{ three strings}\}$.

   (15 points) Show that $A_{\text{TM}} \leq_m FINITE3_{\text{TM}}$.

   We will use reduction for this. Begin by assuming that $FINITE3_{\text{TM}}$ is decidable. Then, by definition, there must be a decider, D, for our language. We will construct a decider for A_{TM} using this decider. We will first process the input to our decider for A_{TM}. The input to A_{TM} is the machine M and the string w. We will construct another machine to pass into our decider for $FINITE3_{\text{TM}}$. Let us call this machine M'. M', which is constructed using the inputs for the decider for A_{TM} will generate a new machine which takes in an input x. On this input x, M' will first run M on w and save the result in a variable. If the value saved in the variable is 'reject', then M' will reject. Otherwise, if our input x is equal to one of 3 strings that we decide, let us call them 'a', 'b', and 'c', then we accept. Otherwise, we reject. This is the machine that we will feed into D. This is significant. The language of the machine that we have just created has only two possible values. If M rejects the input w or loops, then the language of M' is empty. Otherwise, if M accepts w, then the language of M' contains only three strings, namely 'a', 'b', and 'c'. We take this machine, M', and pass it into D. Finally, we define our decider for A_{TM}. This decider will simply return the value that D returns. Therefore, our decider for A_{TM} only accepts when our machine M accepts w. However, this is a contradiction. If this were true, then we would have defined a decider for A_{TM}. This is not possible as this problem is undecidable. As a result, we have shown that $FINITE3_{\text{TM}}$ is undecidable as well.