CS 4510: Automata and Complexity

Spring 2015

Home work 4 // Due: Monday, March 9, 2015

Sagar Laud, slaud3, 902792910

1. (15 points) If A and B are languages, define

$$A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}.$$

Show that if A and B are regular languages, then $A \diamond B$ is a context-free language.

In order to prove this, we simply need to construct a PDA which will recognize $A \diamond B$. Let us start by defining two DFAs which recognize A and B (which must exist because A and B are regular languages). So let us denote these machines as M_A and M_B. All that we need in this case is a stack alphabet consisting of two symbols, namely 0 and \$. So let us now define our PDA. In the start state of the PDA, begin by pushing a \$ onto the stack and transition to the start state of M_A. From here, every single time we read a letter from our input and are in M_A, push a 0 onto the stack. Now, we nondeterministically guess the split between x and y and ϵ transition to the start state of M_B. From here, every time we read a letter, we pop off a 0 from the stack instead of pushing one on. Finally, from the accept states of M_B, we transition to the accept state of our overall PDA if we read a \$ from the stack. This PDA essentially uses the same principles as showing that regular languages are closed under concatenation. It just adds the stipulation that both strings must be of equal length by using a stack in this case.

Sagar Laud, slaud3, 902792910

2. Assume the following fact as given:

   FACT: Every CFL without ϵ can be generated by a CFG in which each rule is in one of the three forms below:

$$\begin{aligned} A &\rightarrow a \\ A &\rightarrow aB \\ A &\rightarrow aBC \end{aligned}$$

   where A, B are variables and a is an alphabet symbol.

   (15 points) Show that every CFL without ϵ can be generated by a CFG such that no consecutive symbols on the right-hand side of rules are variables.

   Looking at the rules, we see that only one of the rules is truly a violation of the new conditions. The rule that violates this is the rule that is of the form $A \rightarrow aBC$. This rule has two consecutive variables. There are two cases that we should look at here.

   Case 1: We can replace B and C (or just C) with a rule of the form $A \rightarrow a$ or $A \rightarrow aB$. If this is possible, we only need to replace C (or we can replace both, it doesn't really matter). Replacing C or both of the variables will lead to a rule that meets our conditions. We will need to add one rule, for each and every combination of substitutions that we have (or we can put them all as one rule separated by vertical lines if this is allowed by the form that we are in).

   Case 2: We cannot replace B and C (or just C) with a rule which causes the rule to directly meet our conditions. In this case, we will need to make a series of substitutions. We simply look at the rule which is a problem (it may be both B and C). Each rule now becomes an instance of case 1 or case 2. If it is case 1, we just go ahead and treat it as such. If it is case 2, the recursive relationship continues. Eventually, we traverse back up our recursive tree making the substitutions until every rule for B and C are in correspondance with the conditions of our grammar. We now finish it off. Because we have eliminated case 2, we now treat case 2 as if it is case 1 and substitute all possible combinations as in case 1 to either make a series of rules or one huge rule separated by vertical lines if allowed by the grammar.

   By handling both cases, we have shown that as long as the CFL does not have ϵ, it can be generated by a CFG in which each rule is in one of the forms specified above.

Sagar Laud, slaud3, 902792910

3. (15 points) Let G be a CFG in Chomsky normal form that contains b variables. Show that if G generates some string using a derivation with at least 2^b steps, $L(G)$ is infinite.

   You can assume as given the following fact:

   FACT: If G is a context-free grammar in Chomsky normal form, then for any string $w \in L(G)$ of length $n \geq 1$, exactly $2n - 1$ steps are required for any derivation of w.

   Let us start by defining a grammar for our purposes. Let us call it G. Since G generates a string using a derivation which has atleast 2^b steps, then the parse tree for this string will have atleast 2^b internal nodes. What is more important is that based on this logic, the parse tree has a height of atleast b+1. This is important as we have shown that one variable is repeated atleast twice along the parse tree by virtue of the pigeonhole principle. Because of this repeated variable, we can now generate an infinite number of strings which are all in L(G).

Sagar Laud, slaud3, 902792910

4. (15 points) Show that the following language is not context-free using the pumping lemma:

$$\{t_1\#t_2\#\cdots\#t_k \mid k \geq 2, \text{ each } t_i \in \{a,b\}^*, \text{ and } t_i = t_j \text{ for some } i \neq j\}$$

We will analyze 2 possible cases to show that this language is not context free. Let us begin by assuming that the language is indeed context free. This means that we have a pumping constant p. Let us choose our string now. Let w = $0^p1^p\#0^p1^p$ and |vxy| $\leq$ p, |vy| $\geq$1.

Case 1: vxy is entirely in t1 or entirely in t2 We see that this will cause problems. If we pump down and set i=0, so our string is uv^0xy^0z, the string cannot possibly be in our language as t1 and t2 must be the same.

Case 2: vxy contains a # This will also cause issues if we pump down and let our string be uv^0xy^0z. This is simply because if we do this, we either remove the # from our string which removes it from the language, or we remove some portion of t1 or t2 because of which they are no longer equal, thereby removing the string for the language.

5. Define a *Constrained Two-Headed* (CTH) machine as follows.

   An CTH machine is a DTM with one tape that is unbounded on one side and that has two heads FH and BH.

   - The head FH is a read-only head and the head BH is a write-only head.
   - Both heads can only move right.
   - If a symbol on the tape under the read head FH is read, the head FH moves right by one position.
   - If a symbol is written on the tape under the write head BH, the head BH moves right by one position.
   - In each step, the machine M being in a state and either reading the symbol on the tape under the head FH or not reading the symbol can: (a) change state, and (b) either write a symbol on the tape or not write a symbol.
   - The string on the tape in positions i through j is said to be *current* if FH is on position i and BH is on position $j + 1$.

   The following is an example of how a CTH machine computes.

   Example: Let the current region on the tape be 100°0110\$. (That is the head FH is pointing to 1 and the head BH is pointing to the blank symbol ⊔ after the \$ symbol on the tape. The alphabet consists of $0, 1$ and marked 0.)

   The CTH machine writes the string 1010°110\$ after this string on the tape and make the new string the current region as follows:

```
WHILE the symbol under the FH head is not $ DO
     IF this symbol is not marked 0 THEN write it on the tape
     ELSE write the symbol 1 on the tape
          Read the next symbol (which is a 0) and write a marked 0 on the tape

Write $ on the tape
```

   That is, the tape is modified so that is has the string 100°0110\$1010°110\$ with the string 1010°110\$ as its current string.

   (a) (10 points) Let the current region on the tape be 100°0110\$. (That is the head FH is pointing to 1 and the head BH is pointing to the blank symbol ⊔ after the \$ symbol on the tape.)
   Show how to write the string 10°10110\$ after this string on the tape and make the new string the current region. (That is, modify the tape so that is has the string 100°0110\$10°10110\$ with the string 10°10110\$ as its current string.)

```
WHILE the symbol under the FH head is not 0 DO
     WRITE the symbol on the tape
Read the next symbol (which is a 0) and write a marked 0 on the tape
Read the next symbol (which is a marked 0) and write a 1 on the tape
WHILE the symbol under the FH head is not $ DO
     WRITE the symbol on the tape

Write $ on the tape
```

   (b) (5 points) Write a formal definition of a CTH machine.
   Let us define a CTH Machine as a 7 Tuple.
   $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
   - Q represents the set of states
   - Σ represents the input alphabet excluding the blank symbol
   - Γ represents the tape alphabet with the stipulation that Σ is a subset of Γ
   - $Q \times \{\Gamma \cup \epsilon\} \to Q \times \{\Gamma \cup \epsilon\}$, ϵ represents that there isn't any reading or writing
   - q_0 represents the start state and is an element of Q
   - q_{accept} and q_{reject} represent the accept and reject states respectively, they are distinct states and are both elements of Q.

6. Given any TM M, construct a 2-PDA P that accepts the same language as M using the following steps. Let $S1$ and $S2$ denote the two stacks. Let $w_1 w_2 \cdots w_m$ be the input.

   Maintain the invariant that at any time, the contents of $S1$ are the contents of the tape of M to the left of the input head and the contents of $S2$ are the contents of the tape of M from the head location to the right end of the tape (the right-most non-blank cell).

   In a tape we can go to a arbitrary cell and perform some operations on it. But a normal PDA can not read, say the fifth character from the bottom of the stack and modify it without altering the contents of the stack above it.

   (a) (3 points) Show how to store the input onto $S2$ such that the first character from the input is at the top.

   This is a rather simple task. Begin by first pushing a \$ onto $S1$. Then for each character read, push it onto $S1$ until the end of input is reached. Finally, reverse the string in $S1$, pop an element off of $S1$ and push it onto $S2$. Do this until you pop a \$ from $S1$.

   (b) Let the current state of M be q. Let M on seeing an a at state q write a b, move left and enter state q'.

   (5 points) How do you simulate this move ?

   When in state q and reading an a from the input:

   Start by writing a b onto $S2$. Then, pop the element on top of $S1$ and push it onto $S2$ (there are quite a few ways to do this, perhaps some dummy states for each symbol in the language would work). Transition to state q' (and move head left if the transition to this state does not already denote that).

   (c) Let M on seeing an a at state q write a b, move right and enter state q'.

   (5 points) How do you simulate this move ?

   This is far more simple than the previous move. When in state q and upon reading an a from the input:

   Write a b to $S1$ and then transition to state q' (and move head right prior to transitioning to this state if the transition itself does not already denote that).

   (d) (2 point) When does P accept?

   P will accept when all elements of the input have been pushed onto $S1$ (with the first symbol in the input being at the bottom of the stack) and when it reads a \$ from $S2$ and enters an accept state.