HOMEWORK 3–CSC 320

Due Friday June 10

(1) Show that the grammar $S \to 1S \mid 0S1S \mid \epsilon$ is ambiguous by drawing two parse trees for some string.

(2) Use the technique in class to find the PDA which accepts strings generated by the grammar in question 1. You may NOT assume you are allowed to push more than one stack symbol on the stack at a time.

(3) Give a PDA to accept $L' = \{w \in \{0,1\} \mid w \text{ contains twice as many 1's as 0's}\}$.

(4) Under which of these operations are context free languages closed? Explain your answers.
Let G_1 and G_2 be grammars for context-free languages L_1 and L_2.
1) union

Create G for $L_1 \cup L_2$ by including all rules for G_1 and G_2. If they share variable names create new variable names for the shared variables so that the grammars have distinct variables. Then add a new start variable S_0 and the rules $S_0 \to S_1 \mid S_2$ where S_1 and S_2 are the start variables for G_1 and G_2.
2) concatenation
Similar to above but add rule $S_0 \to S_1S_2$ instead.

3) complement
It's not closed. $\{ww \mid w \in \{0,1\}^*\}$ is not a CFL but its complement is. We discussed this in class.

(5) Suppose you have designed a PDA. Now, I tell you we have changed our model; we require that the PDA's stack be empty for a string to be accepted. How would you modify your PDA?

Add a transition from the accept states to a new accept state which does nothing but pop the stack til it's empty, e.g., $\delta(q, \epsilon, A) = (q_{new}, \epsilon)$ for all $A \in \Gamma$.

(6) Give an algorithm which tells whether the language of a given grammar G is non-empty. Assume the variables of G are $V_0, V_1, ...V_k$ where V_0 is the start symbol. Hint: For each variable V_i in the grammar, create a variable x_i such that $x_i = 1$ if there is a rule $V_i \to a$ for some terminal a or $V_i \to w$ where w is a string of

terminals and variables and all variables in w evaluate to 1. Explain why $x_0 = 1$ iff G is non-empty. Give an efficient algorithm to determine if $x_0 = 1$. What is the running time of your algorithm as a function of the total number of symbols (including repetitions) in all rules?

Create a directed graph as follows. Store the variables in an array. For each variable, create a variable node. For each rule, create a rule node. For each rule node add an in-coming edge from each variable node where the variable is in the right-hand side of the rule. Add an outgoing edge from the rule node to the variable node for the variable on the left-hand side of the rule. For each rule node, keep a count of the distinct number of unevaluated variables in its right-hand side. For each variable node, keep a bit. Initially all variable nodes are 1.

Now, to run the algorithm, first scan the rule nodes. Form a stack of all rule nodes which have count 0
1) Pop the stack, follow the rule node's outgoing edges to variable nodes.
2) If a variable node x's bit is 1 set it to 0, and follow x's outgoing edges to rule nodes. For each rule node, if its count is greater than 0, decrement its count and push any newly decremented rule nodes which now have count 0 onto the stack.
3) Repeat (2) until no progress is made or the start symbol is 0.

The running time is analyzed as follows: To create the graph, each rule is scanned which costs linear in the size of the rules. The algorithm processes a rule node only to decrement its count, or time proportional to the size of the right-hand side and to add it to the stack and pop it once. Thus the overall time is proportional to the sum of the sizes of the right-hand side of the rules which is linear in the overall size of the rules.

(7) Use the pumping lemma to show that $\{0^n 1^n 0^n 1^n \mid n \geq 0\}$ is not context free.

Let p be the pumping length of the language. Let $s = 0^p 1^p 0^p 1^p$. Then $s = wvwxy$ where $|vwy| < p$. Think of this as $ABCD$ with $A = C =$ all 0's and $B = D =$ all 1's. If vwx consists of all 0's or all 1's then pumping up v and x would result in one of A, B, C or D being longer than the rest and so the resulting string would not be in the language. Alternatively, $vwx = 1^i j 0^j$ or $0^i 1^j$. If v and x were pumped up, then either AB or BC or CD would have too many 1's and 0's compared to the remaining segments and the string wouldn't be in the language.

(8) Explain how to simulate a TM by a PDA with 2 stacks.

The idea is for the PDA to keep a ”left” stack containing the symbols on the tape to the left of the current position which the TM has already read and a ”right” stack containing the symbols on the tape to the right side of the current position which the TM has already read, including blanks if the TM reads beyond the rightmost cell with input. Initially the PDA puts $ on the bottom of each stack to mark the bottoms. The PDA has the same states that the TM has except they are augmented so that the contents of the tape symbols are put on the appropriate stacks. The PDA pops the right stack to move right and the left stack to move left, unless the bottom symbol is reached. If this happens on the right stack, the PDA reads the next input symbol if there is one or blank if there isn't.

If the TM moves to a halting state, then the PDA reads all its remaining input if there is any and then halts in that state.