CSE 322
Winter Quarter 2001
Assignment 8
Due Friday, March 2, 2001

All solutions should be neatly written or type set. All major steps in proofs and algorithms must be justified.

1. (20 points) Consider the grammar $G = (V, \Sigma, R, E)$ where

$$\begin{array}{rcl} V & = & \{T, F, E\} \\ \Sigma & = & \{+, *, (,), id\} \\ R & = & \{E \rightarrow E + T, \\ & = & E \rightarrow T, \\ & = & T \rightarrow T * F, \\ & = & T \rightarrow F, \\ & = & F \rightarrow (E), \\ & = & F \rightarrow id\} \end{array}$$

   (a) Design a PDA M_T by the "top down" construction that accepts $L(G)$. You may use a state diagram. Give a leftmost derivation of $(id + id) * id + id$. Beside it give the sequence of configurations from M_T that corresponds to the leftmost derivation.

   (b) Design a PDA M_B by the "bottom up" construction that accepts $L(G)$. You may use a state diagram. Give a rightmost derivation of $(id + id) * id + id$. Beside it give the sequence of configurations from M_B that corresponds to the rightmost derivation.

   Note: The top down construction is given in the book (pages 107-109). The bottom up construction was given in class and is not found in the book. It works as follows. There is a loop state q_ℓ which has two roles. The first role is to manage *reduce* steps. In a reduce step, if $A \rightarrow \alpha$ is a production, then the PDA in state q_ℓ can remove α^R from the stack and replace it with A. Naturally, helper states may be needed to accomplish this one symbol at at time. The second role is to manage *shift* steps. In a shift step, the PDA in state q_ℓ can take an input symbol and push it on to the stack. No helper states are needed for this. The start state q_0 pushes \$ on to the stack and moves to state q_ℓ. If $S\$$ ever appears on the stack then the PDA can move from q_ℓ to its only accepting state q_f.

2. (10 points) Let $G = (V, \Sigma, R, S)$.

   (a) A nonterminal A is *productive* if $A \Rightarrow_G^* w$ for some $w \in \Sigma^*$. That is, some terminal string can be generated from A. Give an algorithm for finding all the productive nonterminals in a grammar G.

   (b) Use the algorithm in part (a) as part of an algorithm for deciding if the language generated by a context-free grammar is empty.

(c) Use the algorithm in part (a) to construct a context-free grammar G' such that $L(G) = L(G')$ and for all α, if $S \Rightarrow^*_{G'} \alpha$ then $\alpha \Rightarrow^*_{G'} w$ for some $w \in \Sigma^*$. That is, G' and G generate the same language and in G' every partial derivation can be eventually completed into the derivation of some terminal string

3. (10 points) Let $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_1, F_1)$ be a PDA and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be a DFA.

   (a) Use a cross product construction to build a PDA M such that $L(M) = L(M_1) \cap L(M_2)$. This shows that the context-free languages are closed under intersection with regular languages.

   (b) State a behavioral lemma for your construction that can be used to show $L(M) = L(M_1) \cap L(M_2)$.