

CSE 322
Autumn Quarter 2003
Assignment 8
Due Wednesday, December 3, 2003

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

1. (10 points) In this problem we explore the “top down” and “bottom up” construction for PDAs from context-free grammars. Consider the grammar $G = (V, \Sigma, R, E)$ where

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

- (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 $(a + a) * a + a$. Beside it give the sequence of configurations from M_T that corresponds to the leftmost derivation. A configuration shows which symbol of the input is being processed, what state the machine is in, and the contents of the stack.
- (b) Design a (extended) PDA M_B by the “bottom up” construction that accepts $L(G)$. You may use a state diagram. Give a rightmost derivation of $(a + a) * a + a$ in reverse order, top-to-bottom. Beside it give the sequence of configurations from M_B that corresponds to the rightmost derivation.

The bottom up construction was given in class and is not found in the book. It works as follows. There is a 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 extended PDA in state q_ℓ can remove α^R from the stack and replace it with A . 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. If $S\$$ ever appears on the stack then the PDA can move from q_ℓ to its only accepting state q_f . In the start state q_0 , the symbol $\$$ is pushed on the stack, and the machine moves to state q_ℓ .

2. (10 points) In this problem you will get some practice with the pumping lemma for context-free languages. Consider the language $L = \{0^n 10^n 10^m : n \geq m \geq 0\}$. Use the pumping lemma for context-free languages to show that L is not context-free.

3. (10 points) In this problem you will execute an example of the CYK algorithm for deciding membership in a context-free language. Consider the grammar $G = (V, \Sigma, R, S)$ where

$$\begin{aligned} V &= \{S, A, B, C\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \rightarrow AB \mid BC \\ &= A \rightarrow BA \mid 0 \\ &= B \rightarrow CC \mid 1 \\ &= C \rightarrow AB \mid 0\} \end{aligned}$$

- (a) Show the result (the matrix of sets) of running the CYK algorithm on the input 10001.
- (b) If 10001 is generated by the grammar, then use the result of part (a) to construct a parse tree for it.
4. (10 points) In this problem you will design an algorithm to decide if the language generated by a context-free language is finite. First let $G = (V, \Sigma, R, S)$ be a Chomsky normal form context-free grammar with the property that every non-terminal is productive. Recall, that a non-terminal A is productive if $A \Rightarrow^* x$ for some terminal string x . We know that any context-free grammar can be converted to an equivalent grammar with this form. Define the relation $D_G \subseteq V \times V$ as follows. A pair $(A, B) \in D_G$ if there is a production of the form $A \rightarrow XB$ or $A \rightarrow BX$ in R for some X . Design an algorithm for deciding the finiteness of $L(G)$ that uses the reflexive, transitive closure of D_G , called D_G^* , which can be computed using Warshall's algorithm. Explain briefly why your algorithm works.