CSE 322: Formal Methods in Computer Science Spring 2003

PROBLEM SET 5

Due Friday, May 16, 2003, in class

---

1. Lewis and Papadimitriou, Problem 3.1.5.

2. Let $G = (\{S, X\}, \{a, b\}, R, S)$ be the grammar with rules:

$$\begin{aligned} S &\rightarrow aSb \mid bX \mid Xa \\ X &\rightarrow bX \mid aX \mid \epsilon \end{aligned}$$

   (a) Give a simple description of $L(G)$ in English.

   (b) Use the description from (a) above to give a CFG for $\overline{L(G)}$, the complement of $L(G)$.

3. Give context-free grammars that generate the following languages:

   (a) $L_1 = \{a^i b^j c^k d^\ell \mid i + j = k + \ell\}$

   (b) $L_2 = \{w\#x \mid w, x \in \{a, b\}^* \text{ and } w^R \text{ is a substring of } x\}$

4. Let $A = \{xy \mid x, y \in \{a, b\}^* \text{ and } |x| = |y| \text{ but } x \neq y\}$.

   (a) *(Tricky!)* Construct a context-free grammar that generates the language A.

   (b) Draw a parse tree for your grammar that derives the string $aabaabba \in A$.

5. Consider the following natural looking grammar $\text{PROG} = (V, \Sigma, R, \langle\text{STMT}\rangle)$ for a fragment of a programming language:

$$\begin{aligned} \Sigma &= \{\mathsf{if}, \mathsf{condition}, \mathsf{then}, \mathsf{else}, \mathsf{a} := 1\} , \\ V &= \{\langle\text{STMT}\rangle, \langle\text{IF} - \text{THEN}\rangle, \langle\text{IF} - \text{THEN} - \text{ELSE}\rangle, \langle\text{ASSIGN}\rangle\} , \end{aligned}$$

   and PROG has the following rules:

$$\begin{aligned} \langle\text{STMT}\rangle &\rightarrow \langle\text{ASSIGN}\rangle \mid \langle\text{IF} - \text{THEN}\rangle \mid \langle\text{IF} - \text{THEN} - \text{ELSE}\rangle \\ \langle\text{IF} - \text{THEN}\rangle &\rightarrow \mathsf{if\ condition\ then}\ \langle\text{STMT}\rangle \\ \langle\text{IF} - \text{THEN} - \text{ELSE}\rangle &\rightarrow \mathsf{if\ condition\ then}\ \langle\text{STMT}\rangle\ \mathsf{else}\ \langle\text{STMT}\rangle \\ \langle\text{ASSIGN}\rangle &\rightarrow \mathsf{a} := 1 \end{aligned}$$

   (a) Show that PROG is ambiguous. What "programming aspect" does this ambiguity capture?

   (b) Give a new unambiguous grammar that generates the same language as PROG. You do not have to *prove* unambiguity, but informally describe how you are resolving the ambiguity.