CSE 322: Formal Models in Computer Science Winter 2005

PROBLEM SET 5

Due Friday, February 18, 2005, in class

Reading assignment: Section 2.1 of Sipser's text, Handout on Chomsky Normal Form conversion. There are **SIX** questions. Each question is worth **10 points**.

---

1. Prove that context-free languages are closed under union, concatenation, and Kleene star operation.

2. 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) $L_2 = \{R \mid R \text{ is a regular expression over } \{a, b\}\}$.

3. (a) Let G be an arbitrary grammar in Chomsky Normal Form. How many steps does it take to derive a string of length $n \geq 1$ in $L(G)$ using the rules of G?

   (b) Convert the following grammar (where S is the start symbol) into Chomsky Normal Form. Show all intermediate steps clearly.

$$\begin{aligned} S &\rightarrow aAa \mid bBb \mid \epsilon \\ A &\rightarrow C \mid a \\ B &\rightarrow C \mid b \\ C &\rightarrow CE \mid BCD \mid \epsilon \\ D &\rightarrow A \mid B \mid ab \end{aligned}$$

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

$$\begin{aligned} S &\rightarrow aAB \mid aBA \mid bAA \mid \epsilon \\ A &\rightarrow aS \mid bAAA \\ B &\rightarrow aABB \mid aBAB \mid aBBA \mid bS \ . \end{aligned}$$

   Prove that $L(G)$ is the language consisting of all words that have exactly twice as many a's as b's.

5. 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 $aabaabba \in A$.

6. Consider the following natural looking grammar 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:

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

(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.