**Harvard University Extension School**
**Computer Science E-207**

**Problem Set 4**

Due Friday, October 12, 2012 at 11:59 PM Eastern Time.
Submit your solutions in a single PDF called lastname+ps4.pdf emailed to cscie207@seas.harvard.edu.
**LATE PROBLEM SETS WILL NOT BE ACCEPTED.**

Problem set by **ENTER YOUR NAME HERE**

Collaboration Statement: **FILL IN YOUR COLLABORATION STATEMENT HERE
(See the syllabus for information)**

See syllabus for collaboration policy.

PROBLEM 1 (5+5 points)

Show that the following languages are context-free:
(A) $L = \{a^p{+}a^q{>}a^r : p, q, r > 0 \text{ and } p + q > r\}$ over $\Sigma = \{a, +, >\}$ ($a^p$ is a representation of the number $p$ in "unary notation," so this language contains certain valid inequalities in unary arithmetic.)
(B) $L = \{R : R \text{ is a syntactically valid regular expression over } \{a, b\}\}$ over $\Sigma = \{(, ), a, b, \varepsilon, \emptyset, \cup, \circ, *\}$

PROBLEM 2 (6+5+5 points)

We define a strand of DNA as a string over the alphabet $\{A, G, C, T\}$. Two strands of DNA, $\sigma$ and $\tau$, are complementary when $\sigma = \sigma_1 \ldots \sigma_n$, $\tau = \tau_1 \ldots \tau_n$, and each pair $(\sigma_i, \tau_i)$ is equal to one of $(A, T)$, $(T, A)$, $(G, C)$, or $(C, G)$.

(A) Prove that the language $\{xy^R$: $x$ and $y$ are complementary strands of DNA$\}$ is not regular.
(B) Prove that the language $\{xy^R$: $x$ and $y$ are complementary strands of DNA$\}$ is context-free by finding a grammar that generates it. Justify the correctness of your grammar.
(C) Draw a parse tree for the string $AGCGCT$ for the CFG from (B).

PROBLEM 3 (6+ 6+(2) points)

(A) In lecture we mentioned that *right-regular* grammar is equivalent to regular languages, and proved that any regular language has a right-regular grammar. Show the other direction of the equivalence: If a language has a right-regular grammar, then it is regular.

(B) A *two-way regular grammar* is exactly like a CFG with the restriction that all of the productions are of the form $A \to xB$, $A \to Bx$, or $A \to x$ where $A$ and $B$ are variables and $x$ is a string in $\Sigma^*$. Show that there is a nonregular language that can be generated by a two-way regular grammar.

(C) (Challenge!) Using Part A, show that all context free languages over an alphabet with one symbol are regular.

PROBLEM 4 (3+4+6 points)

Let $G = (V, \Sigma, R, \langle \text{STMT} \rangle)$ be the following grammar, which could be used to construct a program in some generic programming language.

$$
\begin{aligned}
\langle \text{STMT} \rangle &\rightarrow \quad \langle \text{ASSIGN} \rangle \mid \langle \text{IF-THEN} \rangle \mid \langle \text{IF-THEN-ELSE} \rangle \\
\langle \text{IF-THEN} \rangle &\rightarrow \quad \texttt{if } \langle \text{CONDITION} \rangle \texttt{ then } \langle \text{STMT} \rangle \\
\langle \text{IF-THEN-ELSE} \rangle &\rightarrow \texttt{if } \langle \text{CONDITION} \rangle \texttt{ then } \langle \text{STMT} \rangle \texttt{ else } \langle \text{STMT} \rangle \\
\langle \text{CONDITION} \rangle &\rightarrow \quad \texttt{true} \mid \texttt{false} \\
\langle \text{ASSIGN} \rangle &\rightarrow \quad \texttt{a := 1} \mid \texttt{a := 0}
\end{aligned}
$$

(A) Show that $G$ is ambiguous.

(B) Find a 'program' $P$ generated by the above grammar that has two parse trees $T_1$ and $T_2$ such that if $P$ is interpreted according to $T_1$ then it will result in the variable $a$ being assigned value 0, and if $P$ is interpreted according to $T_2$ then it will result in the variable $a$ not being assigned any value.

(C) Give a new grammar that generates the same language as $G$ but is unambiguous. Justify briefly why your grammar generates the same language and why it is unambiguous.

PROBLEM 5 (10 points)

Given an arbitrary DFA $M$, provide a general procedure (i.e. an informally described algorithm, like those given in lecture) to compute $|L(M)|$. If $L(M)$ is finite, your procedure should determine this and calculate how many strings $M$ accepts. If $L(M)$ is infinite, your procedure should also determine this. Your procedure should always determine its answer in a finite number of steps (so you cannot say "Run $M$ on all strings in $\Sigma^*$"). You should justify the correctness of your procedure. (Hint: First determine if $L(M)$ is finite. If it is finite, then what is the maximum length of any string in $L(M)$?)