Harvard University
Computer Science 121

Problem Set 5

Due October 22, 2013 at 1:20 PM.
Submit your solutions electronically to cs121+ps5@seas.harvard.edu with "ps5 submission" in the subject line. The solutions to Parts A and B should be attached as separate PDF files, called lastname+ps5a.pdf and lastname+ps5b.pdf.
Late problem sets may be turned in until October 25, 2013 at 1:20 PM with a 20% penalty.

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.

PART B (Graded by Perry)

PROBLEM 1 (3+3+3+3+3 points)

True or false? Justify your answers.
(A) $\{a^{n^2} : n \geq 0\}$ is context free.
(B) $\{a^n b^m : n < m < 2n\}$ is context free.
(C) $\{a^n b^* a^n b^* a^n : n \in \mathbb{N}\}$ is context free.
(D) If L is context-free and R is regular, then $L - R$ is context-free.
(E) If L is context-free and R is regular, then $R - L$ is context-free.

(A) False. Suppose $\{a^{n^2} : n \geq 0\}$ were context free, and so had pumping length p. Consider a^{p^2}, which has partitioning $uvxyz$. $uv^2xy^2z = a^{p^2+|vy|}$. Since $|vxy| \leq p$, $1 \leq |vy| \leq p$, it follows that uv^2xy^2z has length between $p^2 + 1$ and $p^2 + p = p(p + 1)$. But, p^2 and $(p + 1)^2$ are consecutive squares, so any number strictly between them cannot be a square. Thus, uv^2xy^2z cannot have a length which is a perfect square, and so is not in $\{a^{n^2} : n \geq 0\}$, contradicting that it is context free.

(B) True. It is generated by the grammar:
$S \to aSb|aSbb|aabbb$

All strings generated by the grammar are in $\{a^n b^m : n < m < 2n\}$: Let s be the number of steps in the derivation of some string w. If $s = 1$, then $w = aabbb$, and $n = 2 < m = 3 < 2n = 4$, as desired. Next, assume that all strings derived in S or fewer steps are in the desired set. Then, consider a derivation of length $S + 1$. The first step of this derivation adds 1 a, and either 1 or 2 b's, followed by a derivation of length S. Say that this derivation of length S added n a's and m b's to the string. Then this derivation of length $S + 1$ has $n + 1$ a's, and either $m + 1$ or $m + 2$ b's. Observe that since $n < m < 2n$, it follows that $n+1 < m+1 < 2n+2$, and $n+1 < m+2 < 2n+2$. So, the derivation of length $S + 1$ generates a string in the set as desired.

Ever string in $\{a^n b^m : n < m < 2n\}$ is generated by the grammar: To generate $a^n b^m$ such that $n < m < 2n$, follow the first rule $2n - 1 - m$ times, the second rule $m - n - 1$ times, and

the last rule once. This will generate a string of $2n-1-m+m-n-1+2 = n$ a's, followed by $2n-1-m+2m-2n-2+3 = m$ b's, as desired.

(C) False. Let $L = \{a^n b^* a^n b^* a^n : n \in \mathbb{N}\}$. Since CFL are closed under intersection with regular languages, then if L is regular then so is $L' = L \cap a^* b a^* b a^*$. So, suppose L' were context free with pumping length p. Consider $w = a^p b a^p b a^p$, with partitioning $w = uvxyz$. Neither v nor y contains a b, since then uxz would have fewer than 2 b's and not be in the language. So, v and y each must be entirely within one of the groups of a's. So, uxz contains fewer a's in at least one of the groups of a's, but not in all 3. So, there must be one group of a's in uxz that has p a's, and one group that has $< p$ a's, and so $uxz \notin L'$, contradicting that L' is context free, and so that L is context free.

(D) True. $L - R = L \cap \overline{R}$, which must be CF since CFLs are closed under intersection with regular languages, and regular languages are closed under complement.

(E) False. Let $R = \Sigma^*$. The claim then becomes equivalent to the following: if L is CF, then so is $\overline{L}$. However, since CFLs are not closed under complement, this is false.

PROBLEM 2 (5 points)

Prove that if M is a PDA and there exists a number k such that for all $w \in L(M)$, the size of the stack is at most k in each step of every possible computation of w on M, then $L(M)$ is regular.

(A)

The main idea is extract the finite automata component of PDA M and create multiple copies of it, one for each possible stack configuration of M. This enables us to simulate the computation of M without a stack, hence giving us an NFA recognizing $L(M)$.

Formally, let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Since the stack length never exceeds k (on an accepting computation), consider the set $S = \{s \in \Gamma^* : |s| \leq k\}$. The size of S is bounded by $|\Gamma + 1|^k$ and hence is finite.

Our new NFA N will consist of $|S|$ copies of the states Q, one for each $s \in S$. The computation on N starts with the start state of Q_ε, denoted as (q_0, ε). The formal description of $N = (Q', \Sigma, \delta', q_0', F')$ follows:

$$\begin{aligned} Q' &= Q \times S, \\ \delta'((q, s\gamma), \sigma) &= \{(q', s\gamma') : (q', \gamma') \in \delta(q, \sigma, \gamma) \text{ and } s\gamma, s\gamma' \in S\}, \\ q_0' &= (q_0, \varepsilon) \\ F' &= F \times S. \end{aligned}$$

The equivalence of the PDA M and the NFA N can be shown as follows: Observe that there is a transition in N from (q, t) to (q', t') after reading σ, if and only if there is a transition in M upon reading σ which starts in state q with stack t, and ends in state q' with stack t', and $t, t' \in S$. It follows that there is computation in M which has only stack states in S and ends in a state in F, iff there is a computation in N which ends in a state in $F \times S = F'$. But these are exactly the accepting computations in the respective automata. So, since they accept all and only the same strings, they accept the same language, as desired.

PROBLEM 3 (Challenge! 3 points)

Write a context-sensitive grammar for the language $\{a^n b^n c^n : n \geq 0\}$.

$S \to D_A D_B D_C$
$D_A \to \varepsilon | D_A A$
$D_B \to \varepsilon$
$D_C \to \varepsilon$
$Aa \to aA$
$Bb \to bB$
$AD_B \to aD_B B$
$BD_C \to bD_C c$

The grammar works as follows: You start with $D_A D_B D_C$. These each represent dividers, separating the string into equal sections of As, Bs, and Cs. They can all transition to ε at any time to be removed from the string. D_A can also produce any number of A's. Each time an A is added to the string, it can be removed (and so create a valid string of all terminals) in only one way - by having that A switch position with each of the a's until it is adjacent to D_B, turning into $aD_B B$. This adds an a in the string in the correct place, but then creates a B which can also be removed in only one way - switching places with b's until adjacent with D_C, at which point it generates an additional b and c in the correct places. The result of this entire process is that for each A added, exactly 1 a, b, and c is added in the string in the right place.

So, let the number of A's generated by D_A (number of times $D_A \to D_A A$ is used in the derivation of a string) be n. By above, this adds n each of a's, b's, and c's. And, all a's are between D_A and D_B, all b's between D_B and D_C, and all c's after D_C. So, this derives the string $a^n b^n c^n$. Since the grammar can generate all and only strings of that form, it's language is as desired.