Harvard University
Computer Science 121

Problem Set 4

Due Tuesday, October 8, 2013 at 11:59 PM.
Submit your solutions electronically to cs121+ps4@seas.harvard.edu with "ps4 submission" in the subject line. The solutions to each part should be attached as separate PDF files, called lastname+ps4a.pdf, lastname+ps4b.pdf.
Late problem sets may be turned in until Friday, October 11, 2013 at 11:59 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)**

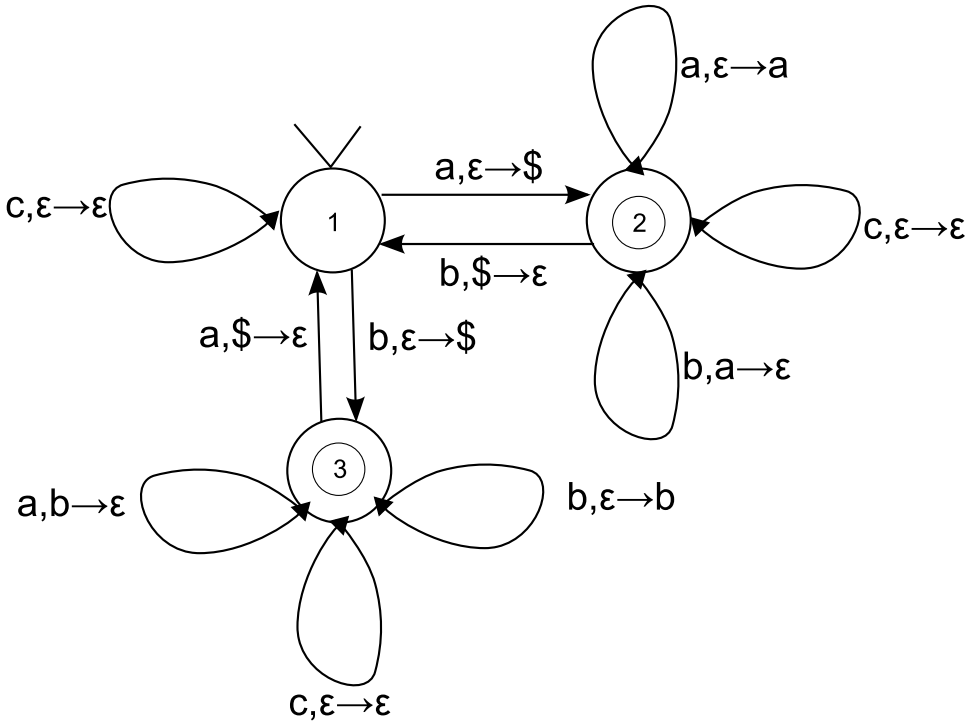
See syllabus for collaboration policy.

PART B (Graded by Paul and Anupa)

PROBLEM 1 (8 points)

Let L be the language $\{w : w \text{ has equal numbers of } a\text{'s}, b\text{'s and } c\text{'s}\}$. Prove that $\overline{L}$ is context free.

For a string w in $\overline{L}$, let j, k, and l be the number of a's, b's and c's in w respectively. Then observe that L consists of strings in which $j = k = l$, so $\overline{L}$ consists of strings in which $j \neq k$, $k \neq l$, $j \neq l$, but that any two of these suffice by transitivity, so in particular we take $j \neq k$ and $j \neq l$. We can create a PDA for the first case ($j \neq k$), as shown below:



This PDA accepts any string over the alphabet $\Sigma = \{a, b, c\}$ in which the number of a's does not equal the number of b's by only accepting if $j > k$ (state 2) or $k > j$ (state 3). Notice that $j > k$ or $k > j$ if and only if $j \neq k$. Also, this PDA ignores all c's.

A similar PDA can be constructed for the second case ($k \neq l$) by switching a's and b's with b's and c's respectively. CFLs are closed under union, so the union of the two languages recognized by these PDAs are context free. We've already argued that the union of these two languages is $\overline{L}$, so we conclude that $\overline{L}$ is context free.

PROBLEM 2 (8 + 4 points)

A context-free grammar G is **ambiguous** if there exists a string $w \in L(G)$ with two distinct leftmost derivations in G.

(A) Prove that the language of $G = (V, \Sigma, R, S)$, where $V = \{S\}, \Sigma = \{a, b\}$, and $R = \{S \to SSS, S \to bS, S \to Sb, S \to a\}$ is

$$L(G) = \{w\text{: the number of a's in } w \text{ is odd}\}$$

Hint: To show $\subseteq$, proceed by induction on the length of the derivation. To show $\supseteq$, proceed by induction on the number of b's.

(B) Show that the grammar of part (A) is ambiguous but that there is an unambiguous grammar for L(G).

(A) For $\subseteq$:
Base case: Derivation with 1 step. $S \to a$
Inductive Hypothesis: Assume that for every derivation $S \Rightarrow w, w \in L$
Inductive Step: Prove that every derivation with n+1 steps generates a string in L. Let $S \Rightarrow w$ be a derivation with n+1 steps. The derivation starts with either bS or Sb or SSS.

- In the bS case, w must be of the form bw_1, where w_1 is a string derived from the nonterminal S in the remaining n steps. By the inductive hypothesis, this w_1 has an odd number of a's, and adding a b does not change that.
- In the Sb case, w must be of the form w_2b, where w_2 is a string derived from the nonterminal S in the remaining n steps. By the inductive hypothesis, this w_2 has an odd number of a's, and adding a b does not change that.
- In the SSS case, w must be of the form $w_3w_4w_5$, where w_3, w_4, w_5 are strings derived from the nonterminal S's in the remaining n steps. By the inductive hypothesis, the w_3, w_4, w_5 all have an odd number of a's, and adding 3 odd numbers together will result in an odd number ($2m + 1 + 2p + 1 + 2q + 1 = 2(m + p + q + 1) + 1$).

Hence, we have proven that $L(G) \subseteq \{w$: the number of a's in w is odd$\}$

For $\supseteq$:
Base Case: We have 0 b's and $2n+1$ a's. Any string of this form can be generated using the $S \to a$ and $S \to SSS$ rules. We have proven above that we can generate a using the $S \to a$ rule. To

create more a's we can replace the $S \to a$ rule with $S \to SSS$ rule and then terminate each S using $S \to a$. Each time we will add 2 a's to each string, and in this way we can generate any string equal to a^{2n+1}.
Inductive Hypothesis: Assume that every string with k b's and an odd number of a's is in $L(G)$.
Inductive Step: Prove that every string with k+1 b's and an odd number of a's is in $L(G)$. This amounts to proving that our grammar allows us to insert a b anywhere in the string.

- We can insert a b adjacent to any a. Any terminal a must have been derived from the $S \to a$ rule so we just replace this rule with either $S \to bS$ or $S \to Sb$ and then terminate using the $S \to a$ rule. So we have now generated a string with the same odd number of a's as before and k+1 b's.
- We can insert a b adjacent to any b. We know that there must be at least one a in any string generated by this grammar. Therefore we simply go to the beginning or end of any continuous block of b's until we find an a and use the case above to insert a b adjacent to any a.

Hence, we have proven that $L(G) \supseteq \{w$: the number of a's in w is odd$\}$

(B) In slide 11 of the Oct. 1 lecture, Professor Lewis describes how to construct a regular grammar from a DFA. Since the grammar constructed is regular, no derivation will contain more than one non-terminal, and the non-terminal present will always be at the right-hand end of the intermediate string. Furthermore, each rule application will generate just one terminal symbol (except for the last, which will generate no terminal symbol at all), so the string generated will be derived one character at a time from left to right. Each step will take the form $wq \Rightarrow w\sigma p$ where $\delta(q, \sigma) = p$, and so there will be only one choice of rule to apply at each step, for the non-terminal at the end of the string and the next character to be generated determine q and σ, and δ is a function. Thus each string will have just one leftmost derivation (indeed, just one derivation of any kind) under a grammar constructed in this way, and such a grammar can be constructed for any regular language by following the directions in the lecture notes.

The total number of occurrences of a in any string derivable by S is odd. But this is just $L(b^*a(b^*ab^*a)^*b^*)$, which, being represented by a regular expression, is clearly regular, and thus not ambiguous.

The regular grammar generated from this expression is $G' = (\{S, R\}, \{a, b\}, \{S \to bS, S \to aR, R \to aS, R \to bR, R \to \epsilon\}, S)$.

PROBLEM 3 (Challenge! 3 points)

Show that every context-free language over a unary alphabet is regular.