Problem Set 4

CS373 - Spring 2011

Due: Thursday April 7 at 2:00 PM in class (151 Everitt Lab)
Please follow the homework format guidelines posted on the class web page:
`http://www.cs.uiuc.edu/class/sp11/cs373/`

1. Pumping Lemma / Ogden's Lemma
[**Category**: Proof, **Points**: 15] Ogden's Lemma is a more general version of the pumping lemma for context-free languages. It states that for every context-free language L, there exists an integer m such that for every $w \in L$ where $|w| \geq m$, and every choice of m or more "distinguished" positions in w, there exist strings u, v, x, y, z such that $w = uvxyz$ and the following three conditions hold.

   A) vy contains at least one distinguished position.

   B) vxy contains m or fewer distinguished positions.

   C) For every $i \geq 0$, $uv^ixy^iz \in L$.

   If every position in w is distinguished, then the result is the pumping lemma. Ogden's Lemma allows one to restrict the set of decompositions that need to be considered when proving that a language is not context-free.

   i) Let $n_0(w)$ be the number of 0's in a string w. Let $n_1(w)$ be the number of 1's in w. Let $n_2(w)$ be the number of 2's in w. Use the pumping lemma for CFLs to prove $L_1 = \{w \in \{0,1,2\}^* \mid n_0(w) = min(n_1(w), n_2(w))\}$ is not context-free. (5 Points)

   ii) Prove that $L_2 = \{0^i1^{i+j}0^j \mid j \neq i\}$ satisfies the pumping lemma for CFLs.(5 Points)

   iii) Use Ogden's Lemma to prove that L_2 is not context-free. (5 Points)

2. Unpoppable PDA
[**Category**: Proof, **Points**: 10] An unpoppable PDA (UPDA) is a pushdown automaton that cannot remove anything from its stack. The top symbol may be read, and more symbols may be added, but symbols cannot be removed from the stack once placed on it. Prove that the set of languages recognized by UPDAs is exactly the regular languages.

3. Priority PDA
[**Category**: Construction, **Points**: 10] A priority PDA (PPDA) is a pushdown automaton that uses a priority stack instead of a stack. Let f be an injective function from the stack alphabet to the integers (you are allowed to choose this function). The symbol on the stack that can be read and/or popped by the machine is the symbol a that has a minimal value of $f(a)$ from among all the symbols on the stack. For example, if $\Gamma = \{a, b, c\}$, and $f(a) = 3, f(b) = 1, f(c) = 8$, then b would be at the top of the stack if the stack contained any b's. An a would be at the top of the stack if

the stack contained an a but no b's. A c would be at the top of the stack only if the stack contained no a's or b's. The stack sorts itself immediately when new symbols are added. Find a non-context-free language that can be recognized by a PPDA, and construct a PPDA that recognizes it.

4. CYK Algorithm
   [**Category**: Algorithm, **Points**: 15] Let G be a context-free grammar with terminal set $\{0, 1, /, +, -\}$ and start symbol S defined by the following production rules:

   $S \rightarrow N/N$

   $N \rightarrow N + N \mid N - N \mid B$

   $B \rightarrow 0B \mid 1B \mid 0 \mid 1$

   Let $L(G)$ be the language generated by G.

   i) Convert G to Chomsky Normal Form (5 Points)

   ii) Use the CYK algorithm to parse $w_1 = 1 + 11/10$ and determine whether or not w_1 is in $L(G)$. (5 Points)

   iii) Use the CYK algorithm to parse $w_2 = 1/10 - 01$ and determine whether or not w_2 is in $L(G)$. (5 Points)

5. Context-sensitive grammar
   [**Category**: Construction, **Points**: 20] Let L be the language $\{a^{i^2} \mid i \geq 0\}$.

   i) Prove that L is not context-free. (5 Points)

   ii) Create a context-sensitive grammar for L. Explain the purpose of each non-terminal and the basic algorithm your grammar uses to generate the strings. (15 Points)

6. Square root Turing machine
   [**Category**: Construction, **Points**: 15] Construct a Turing Machine that recognizes $L = \{a^i\#a^j \mid \lfloor\sqrt{i}\rfloor = j\}$. Give an implementation-level description of how your machine works, and briefly argue its correctness.

7. Prime number Turing machine
   [**Category**: Construction, **Points**: 15] Let b_i be the representation of a non-negative integer i in binary with no leading zeros. Construct a Turing Machine that recognizes $L = \{b_i \mid i \text{ is prime }\}$. Give an implementation-level description of how your machine works, and briefly argue its correctness.