Homework 5
Due Nov 10, 2004

Prof. Amit Chakrabarti
Computer Science Department
Dartmouth College

Please think carefully about how you are going to organise your answers *before* you begin writing. Make sure your answers are complete, clean, concise and rigorous.

1. Design a context-free grammar that generates the language

$$\{x \in \{0,1\}^* : x \text{ is not of the form } ww \text{ for any } w \in \{0,1\}^*\}.$$

[6 points]

2. For each of the following languages, say whether or not it is a CFL and prove your answer, either by designing an appropriate CFG or PDA or by using the pumping lemma. If designing a CFG/PDA, please explain your construction in brief so the grader can understand your design.

   2.1. $\{a^n b^n c^m : n \leq m \leq 2n\}$. [8 points]

   2.2. $\{a^n b^{n^2} : n \geq 0\}$. [8 points]

   2.3. $\{x_1 \# x_2 \# \cdots \# x_k : k \geq 1$, each $x_i \in \{a,b\}^*$, and for some i and j (possibly equal), $x_i = x_j^{\mathcal{R}}\}$. [8 points]

   2.4. $\{b_i \# b_{i+1} \in \{0,1,\#\}^* : i \geq 1\}$, where b_i is the binary representation of the integer i with no leading 0's (e.g. $b_5 = 101, b_{18} = 10010$). [10 points]

   2.5. $(a \cup b)^* - \{(a^n b^n)^n : n \geq 1\}$. [10 points]

3. Do problem 2.14 from your textbook (Sipser). [10 points]

4. Define the *length* of a rule in a CFG to be the number of characters required to write down the rule. Thus, if $G = (V, \Sigma, R, S)$ is a CFG and "$A \to w$" is a rule in R with $w \in (V \cup \Sigma)^*$, then the length of this rule is $2 + |w|$ if $w \neq \varepsilon$ and 3 if $w = \varepsilon$ (because we do have to write one character to represent ε, even though $|\varepsilon| = 0$). Define the *complexity* of a CFG to be the sum of the lengths of all the rules in the CFG.

   Note that things like "$A \to w \mid x \mid y$" are not rules; they are convenient notation for, in this case, three separate rules "$A \to w$", "$A \to x$", and "$A \to y$", so the lengths of these three rules must be figured separately and added up.

   4.1. Suppose the CFG $G = (V, \Sigma, R, S)$ is converted into an equivalent CFG G' in Chomsky normal form, using the procedure in Sipser's Theorem 2.6. Give the best possible asymptotic upper bound on the complexity of G', in terms of $|V|, |R|$ and the complexity of G. Assume $|\Sigma|$ is a constant; think $\Sigma = \{0,1\}$ if you like. Assume nothing about G: your upper bound must always hold.

   By "asymptotic" I mean that you don't need to give an exact bound, and should use big-O notation to simplify, where possible. You don't have to prove that your upper bound is best, but you should prove (at least informally) why your bound always holds. [5 points]

4.2. Suppose the PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is in normal form as discussed in class (i.e., each move is either a one-character push or a pop, but not both; there is only one accept state; and the stack is empty upon acceptance). Suppose we use the procedure of Sipser's Lemma 2.15 to convert it into a CFG G. Give the best possible asymptotic upper bound on the complexity of G, in terms of $|Q|$. Assume $|\Sigma|$ and $|\Gamma|$ are constants.

Again, assume nothing about M: your upper bound must hold even for the most outlandish of PDAs.

[5 points]

4.3. We proved that every regular language is context-free as follows: a regular language is recognized by a DFA, which is automatically a PDA (that ignores its stack), which is equivalent to some CFG. However, having solved the previous problem, you know that if we start with a DFA with n states, the complexity of the CFG that results by following this proof may be ***huge*** (in terms of n, assuming a constant-sized alphabet).

Come to the rescue by proving that any n-state DFA (over a constant-sized alphabet) can be converted into an equivalent CFG whose complexity is only $O(n)$.

Hint: Prove that any DFA can be converted into a CFG where every rule is either of the form "$A \to aB$", or of the form "$A \to \varepsilon$", where A, B are variables and a is a terminal. What would be a natural choice for the set of variables?

[15 points]

5. Do problem 2.21 from your textbook (Sipser).

Note: It's not very easy! Start thinking right away. [15 points]

Challenge Problems

CP5: A deterministic pushdown automaton (DPDA) is just like a PDA, except that its transition function must be deterministic. Thus, it is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, q_0 and F have exactly the same roles as in a PDA, but δ is a function of the form

$$\delta : Q \times \Sigma \times \Gamma_\varepsilon \to Q \times \Gamma_\varepsilon$$

instead of the PDA-like $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to 2^{Q \times \Gamma_\varepsilon}$. Unlike in the finite automaton case, where nondeterminism did not add extra power, nondeterminism provably adds power to pushdown automata. Your challenge is to prove this. The language that will help you do so is $\{0^n 1^n : n \geq 0\} \cup \{0^n 1^{2n} : n \geq 0\}$: prove that this language is context-free but cannot be recognized by a DPDA.