CSE 322: Formal Models in Computer Science Winter 2005

Problem Set 6

Due Friday, February 25, 2005, in class

Reading assignment: Section 2.2 of Sipser's text.

There are **FIVE** questions. Each question is worth **12 points**.

---

1. Construct PDAs that recognize the following languages (you can give state diagrams or simply give informal but precise English description of the operation of the PDA):

   (a) The set of strings over the alphabet $\{a, b\}$ with twice as many a's as b's.

   (b) The complement of the language $A = \{w \in \{a, b, c\}^* \mid w \text{ has an equal number of } a\text{'s}, b\text{'s and } c\text{'s}\}$. We will later see that A itself is not context-free, and thus the class of context-free languages is not closed under complement.

2. Prove that the intersection of a context-free language and a regular language is another context-free language.

3. In class, we saw a "top-down" way to parse context-free languages, by giving a PDA which simulated the leftmost derivation of a string starting from start symbol on stack. This exercise illustrates another approach to construct a PDA to recognize a CFL.

   Let $G = (V, \Sigma, R, S)$ be a grammar. Assume without loss of generality S does not appear on the right hand side of any rule. Construct a PDA M with start state p, a single accept state q and transition function δ that satisfies the following properties:

   - $\delta(p, a, \epsilon) = \{(p, a)\}$ for every $a \in \Sigma$
   - $\delta(p, \epsilon, S) = \{(q, \epsilon)\}$
   - For each rule $A \to \alpha$, there are transitions which enable the PDA M in state p, to pop α^R (the reverse of α) from the top of the stack and push A in its place. M does so without reading any input symbol and ends up back in state p. (To do this formally, one would, of course, introduce some intermediate states just for simulating this rule $A \to \alpha$ that are not used for any other rule.)
   - No other transitions are permitted under δ.

   Now to your exercises:

   (a) Let G and M be as just presented. Give a *convincing* argument why $L(G) = L(M)$. (You are welcome to give a formal proof by induction if you prefer to do that.)

   (b) Given as input $w \in L(G)$, how does the computation of M on w relate to the derivation of w under G? Justify your answer.
   Suggestion: It might be instructive to pick a simple grammar G and a simple string in $L(G)$ and understand the stack contents of M when run on that input.

4. A string y is said to be a permutation of the string x if the symbols of y can be reordered to make x. For example, the permutations of $x = 011$ are $110, 101$, and 011. If L is a language, then $\text{perm}(L)$ is the set of strings that are permutations of strings in L. Thus, for $L = \{0^n 1^n \mid n \geq 1\}$, $\text{perm}(L)$ is the set of strings that have an equal number of 0's and 1's.

(a) Give an example of a regular language over alphabet $\{0,1\}$ such that $\text{perm}(L)$ is not regular. Justify your answer.
Suggestion: Try to find a regular language A such that $\text{perm}(A) = \{0^n1^n \mid n \geq 1\}$.

(b) *(Tricky!)* Prove that for every regular language $L \subseteq \{0,1\}^*$, $\text{perm}(L)$ is context-free.
Hint: Build a PDA for $\text{perm}(L)$ which guesses a permutation of the input string and runs the DFA for L on it, and uses the stack to check that it indeed guessed a valid permutation of the input string. The fact that the alphabet has only two symbols will be crucial for the latter check.

5. We saw that allowing nondeterminism did not add any power to finite automata in terms of the class of languages recognized. In this problem, you will show that this is not the case for PDAs.

We now define deterministic pushdown automata (DPDA). DPDAs are similar to PDAs with a few subtle differences:

- Most importantly, they are deterministic, any configuration has at most one next configuration, and in fact all configurations which haven't read the entire input have a unique next configuration.
- Instead of beginning with an empty stack, DPDAs start with a special symbol \$ as the lone symbol on the stack. The DPDA can thus always know when it has reached the bottom of the stack. The \$ symbol cannot be erased, i.e., every transition reading \$ must push it back on the stack.

For technical reasons that include the fact that \$ cannot be popped off the stack, we must allow the DPDA to push more than one symbol in a single step (for PDAs we insisted that at most one symbol is pushed in any single step). Incorporating this, we can formally define a DPDA as a 7-tuple $(Q, \Sigma, \Gamma, \$, \delta, q_0, F)$ where

- Q, Σ, Γ, q_0 and F are the same as for a PDA, with the requirement $\$ \in \Gamma$, and
- $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$ is the transition function, with the restriction that for every $q \in Q$ and $a \in \Sigma$, $\delta(q, a, \$) = (p, \gamma\$)$ for some $p \in Q$ and $\gamma \in \Gamma^*$.

(a) Show that DPDAs are more powerful than finite automata, i.e., find a non-regular language that is recognized by a DPDA.

(b) Show that DPDAs are less powerful than PDAs. You can use the statements we made in problem 1(b).