CSE 322 Autumn 2009
Assignment #8

Due: Friday, December 11, 2009 in class

Reading assignment: Read Sections 3.1, 3.3 and 4.2 of Sipser's book

Problems:

1. We saw that allowing nondeterminism did not allow finite automata to recognize any additional languages. In this problem you will show that this is not the case for PDAs.

   We define Deterministic Pushdown Automata (DPDA) below. DPDAs are similar to PDAs with a few important differences.

   - Most importantly, they are deterministic: any configuration has at most one next configuration, and any configuration that has not read the entire input has a unique next configuration.
   - Instead of beginning with an empty stack, DPDAs start with a special symbol \$ as the lone symbol on the stack. A DPDA can 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.
   - The input is presented with a special end marker $\#$.
   - DPDAs are allowed to push more than one symbol in a single step. (With PDAs we already have seen that we can simulate this by adding extra states. With DPDAs we must include this power in the base model so that we can require that a DPDA reads a symbol of the input at every step.)

   Formally, a DPDA is a 7-tuple $M = (Q, \Sigma, \Gamma, \$, \delta, q_0, F)$ where

   - Q, Σ, Γ, q_0, and F are defined as in an ordinary PDA except that we require that $\$ \in \Gamma$.
   - $\delta : Q \times (\Sigma \cup \#) \times \Gamma \to Q \times \Gamma^*$. Thus a DPDA reads precisely one symbol per step and always examines the top symbol on the stack. Given these, there is precisely one next state that can be reached and one option for how the stack changes. Moreover, for all $q \in Q$, and $a \in \Sigma$ we have $\delta(q, a, \$) \in Q \times \Gamma^*\$$; i.e., the only allowable stack operation when the bottom-of-stack symbol is read is to push some string on top of it.
   - The starting configuration on input x is $(q_0, x\#, \$)$. The DPDA accepts iff the computation ends in a final state (independent of what is on the stack).

   (a) Show that there is a DPDA that recognizes a non-regular language.

   (b) Show that if A is recognized by a DPDA then so is the complement of A.

(c) Use the fact shown in (b) to show that there is a CFL that cannot be recognized by a DPDA.

2. Sipser's text, 2nd edition Exercise 4.6 (1st edition Exercise 4.7). (Note that infinite binary sequences are not strings since any string has finite length.)

3. Sipser's text, 2nd edition Exercise 4.7 (1st edition Exercise 4.8).

4. Sipser's text, 2nd edition Problem 4.12 (1st edition Problem 4.11).

5. Define a 2-PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ to be just like a PDA except that it has two stacks which it can access at the same time. That is

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})^2 \rightarrow Q \times (\Gamma \cup \{\varepsilon\})^2.$$

A 2-PDA accepts a string x if when started with x as input and two empty stacks it can reach a state in F. Show that for any Turing machine there is a 2-PDA that accepts precisely the same set of input strings. (Hint: Figure out how the 2 stacks can simulate the Turing machine tape.)

6. (Extra credit) Show how a variant of a PDA that has a queue rather than a stack can simulate a Turing machine.

7. (Extra Credit) For a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ we say that a string $\alpha \in \Gamma^*$ is a *possible stack of* M if there is some input and some choice of moves of M such that α appears on the stack at some point during its computation. Prove that the language $L \subseteq \Gamma^*$ of possible stacks of M is regular.