Tutorial 5

1. Construct NPDAs for the following languages. Let $\#_a(w)$ denote the number of a's in string w, where $a \in \Sigma$.

   (a) $L = \{wcw^R \mid w \in \{a,b\}^*\}$. $\Sigma = \{a,b,c\}$.

   (b) $L = \{w \mid \#_a(w) > \#_b(w)\}$. $\Sigma = \{a,b\}$.

   (c) $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$. $\Sigma = \{a,b,c\}$.

   (d) $L = \{w_1 c w_2 \mid w_1, w_2 \in \{a,b\}^* \text{ and } w_1 \neq w_2^R\}$. $\Sigma = \{a,b,c\}$.

2. Find an NPDA (accepting by final state) which accepts aa^*ba. The presence of stack (or any memory device) can sometimes reduce the number of states requried to accept a regular language. The above language is regular, but any NFA accepting the above language needs to have at least four states. How many states does your NPDA accepting the above language has? Could you give a NPDA accepting the above language which has only two states?

3. Suppose one has two stacks instead of one stack in NPDA. Intuitively, the NPDA can now push (possibly different) strings on the two stacks, and base its actions on the top symbol of each of the stacks as well as on the input symbol.

   Formally define a two stack NPDA. Is it more powerful than one stack NPDA (that is can it accept something which cannot be accepted by one stack NPDA)?

4. A DPDA (or deterministic push down automata) is just like an NPDA, but all its moves are deterministic, that is, in each state, for each top of stack symbol and input symbol, there is at most one possible next move. Additionally, if there is an ϵ move for some state q and top of stack symbol A, then there is no move involving state$=q$, top of stack $= A$, and any input symbol in Σ.

   (a) Formally define DPDA.

   (b) Can all regular languages be accepted by a DPDA?