# Tutorial 5

1. Construct NPDA for the following languages over alphabet $\Sigma$.

   (a) $L = \{wcw^R \mid w \in \{a,b\}^*\}$. $\Sigma = \{a,b,c\}$.
   NPDA: $(\{q_0,q_1,q_2\},\{a,b,c\},\{a,b,Z_0\},\delta,q_0,Z_0,\{q_2\})$.
   Acceptance by final state.
   $\delta(q_0,a,X) = \{(q_0,aX)\}$, for $X \in \{a,b,Z_0\}$.
   $\delta(q_0,b,X) = \{(q_0,bX)\}$, for $X \in \{a,b,Z_0\}$.
   $\delta(q_0,c,X) = \{(q_1,X)\}$, for $X \in \{a,b,Z_0\}$.
   $\delta(q_1,a,a) = \{(q_1,\epsilon)\}$
   $\delta(q_1,b,b) = \{(q_1,\epsilon)\}$
   $\delta(q_1,\epsilon,Z_0) = \{(q_2,\epsilon)\}$
   Intuitively, on input $wcw^R$, in state $q_0$, the above PDA pushes $w$ on the stack. On input $c$, it changes state to $q_1$, and then matches the input $w^R$ with the $w$ (in reverse) on the stack.

   (b) $L = \{w \mid \#_a(w) > \#_b(w)\}$. Below is for $\Sigma = \{a,b\}$.
   NPDA: $(\{q_0,q_1\},\{a,b\},\{a,b,Z_0\},\delta,q_0,Z_0,\{q_1\})$.
   Acceptance by final state.

   $\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$
   $\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$
   $\delta(q_0,a,b) = \{(q_0,\epsilon)\}$
   $\delta(q_0,a,a) = \{(q_0,aa)\}$
   $\delta(q_0,b,a) = \{(q_0,\epsilon)\}$
   $\delta(q_0,b,b) = \{(q_0,bb)\}$
   $\delta(q_0,\epsilon,a) = \{(q_1,\epsilon)\}$
   Intuitively, the above PDA pushes excess $a$'s or $b$'s on the stack, and whenever it finds $a$ in the stack and $b$ in the input, or vice versa, it pops out the stack element. It accepts if the stack contains $a$ at the end.

   (c) $L = \{a^ib^jc^k \mid i = j \text{ or } j = k\}$. $\Sigma = \{a,b,c\}$.
   NPDA: $(\{q_0,q_1,\ldots,q_7\},\{a,b,c\},\{a,b,Z_0\},\delta,q_0,Z_0,\{q_3,q_7\})$
   Acceptance by final state.

   $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0),(q_4,Z_0)\}$.
   $\delta(q_1,\epsilon,Z_0) = \{(q_3,Z_0)\}$.
   $\delta(q_1,a,Z_0) = \{(q_1,aZ_0)\}$.
   $\delta(q_1,a,a) = \{(q_1,aa)\}$.
   $\delta(q_1,b,a) = \{(q_2,\epsilon)\}$.

$\delta(q_2, b, a) = \{(q_2, \epsilon)\}$.
$\delta(q_2, \epsilon, Z_0) = \{(q_3, Z_0)\}$.
$\delta(q_3, c, Z_0) = \{(q_3, Z_0)\}$.
$\delta(q_4, a, Z_0) = \{(q_4, Z_0)\}$.
$\delta(q_4, \epsilon, Z_0) = \{(q_7, Z_0)\}$.
$\delta(q_4, b, Z_0) = \{(q_5, bZ_0)\}$.
$\delta(q_5, b, b) = \{(q_5, bb)\}$.
$\delta(q_5, c, b) = \{(q_6, \epsilon)\}$.
$\delta(q_6, c, b) = \{(q_6, \epsilon)\}$.
$\delta(q_6, \epsilon, Z_0) = \{(q_7, \epsilon)\}$.
Intuitively, states $q_1$ to $q_3$ check if the input is of the form $a^i b^j c^k$, with $i = j$ (where this matching is done using standard way as done in class). Note that in state $q_3$, NPDA would just check if the remaining inputs are $c$. States $q_4$ to $q_7$ check if the input is of the form $a^i b^j c^k$, with $j = k$.

(d) $L = \{w_1 c w_2 \mid w_1, w_2 \in \{a, b\}^*$ and $w_1 \neq w_2^R\}$. $\Sigma = \{a, b, c\}$.
NPDA: $(\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_2, q_3\})$
Acceptance by final state.
$\delta(q_0, a, Z) = \{(q_0, aZ)\}$, for $Z \in \{a, b, Z_0\}$
$\delta(q_0, b, Z) = \{(q_0, bZ)\}$, for $Z \in \{a, b, Z_0\}$
$\delta(q_0, c, Z) = \{(q_1, Z)\}$, for $Z \in \{a, b, Z_0\}$
$\delta(q_1, a, a) = \{(q_1, \epsilon)\}$
$\delta(q_1, b, b) = \{(q_1, \epsilon)\}$
$\delta(q_1, a, b) = \{(q_2, \epsilon)\}$
$\delta(q_1, b, a) = \{(q_2, \epsilon)\}$
$\delta(q_1, a, Z_0) = \{(q_2, Z_0)\}$
$\delta(q_1, b, Z_0) = \{(q_2, Z_0)\}$
$\delta(q_1, \epsilon, a) = \{(q_3, \epsilon)\}$
$\delta(q_1, \epsilon, b) = \{(q_3, \epsilon)\}$
$\delta(q_2, a, X) = \{(q_2, X)\}$, for $X \in \{a, b, Z_0\}$
$\delta(q_2, b, X) = \{(q_2, X)\}$, for $X \in \{a, b, Z_0\}$
Intuitively, on input $w_1 c w_2$, the PDA above initially pushes $w_1$ on the stack, and on input $c$, transfers to state $q_1$. In state $q_1$ it matches the input character with stack letter, and if it matches it is popped out; if not then it goes to state $q_2$ (intending to accept, if there are only $a$'s and $b$'s in the remaining input). Similarly, if $|w_1| < |w_2|$, then the PDA transfers to $q_2$ (when there is $a$ or $b$ in the input, but stack has only $Z_0$). To check for the case that $|w_1| > |w_2|$, while in $q_1$, the PDA may nondeterminstically guess that the input is over, $|w_1| > |w_2|$, and transfer to state $q_3$ (in which case it does not read any further input).

2. Find an NPDA which accepts $aa^*ba$. The presence of stack (or any memory device) can sometimes reduce the number of states required 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?

Ans: For acceptance by empty stack, one can accept any regular language using one state as discussed in class.

For acceptance by final state, consider the following NPDA:

$(\{q_0, q_1\}, \{a, b\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_1\})$

$\delta(q_0, a, Z_0) = \{(q_0, A)\}$

$\delta(q_0, a, A) = \{(q_0, A)\}$

$\delta(q_0, b, A) = \{(q_0, B)\}$

$\delta(q_0, a, B) = \{(q_1, \epsilon)\}$

Informally, an NFA needs at least four states to accept the above language, as it has at least one accepting state, and the non-accepting states need to be able to distinguish between whether none or at least one $a$ has been received and whether $b$ has been received.

More formally, consider any NFA accepting the language. Consider the accepting run of NFA on input $aba$. Let $q_0$ be the starting state, $q_1$ be the state after reading $a$, $q_2$ be the state after reading $ab$ and $q_3$ be the state after reading $aba$. Then, if $q_0, q_1, q_2, q_3$ are all distinct, then we are done. Otherwise, if at least two of $q_0, q_1, q_2, q_3$ are same, say $q_i = q_j$, with $i < j$, then deleteing the part of the string $aba$, which took the NFA from $q_i$ to $q_j$, we have that the shorter string is still accepted by the NFA. However, the shorter string is not in $L$.

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)?

   Ans: A two stack NPDA is denoted by $(Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, Z_0, Y_0, F)$, where $Z_0 \in \Gamma_1, Y_0 \in \Gamma_2$ are the initial symbols on the two stacks. Rest of the parameters have usual meaning.

   Transition function is a mapping from $Q \times \Sigma \cup \{\epsilon\} \times \Gamma_1 \times \Gamma_2$ to a subset of $Q \times \Gamma_1^* \times \Gamma_2^*$.

   Intuitively, $\delta(q, a, X, Y)$ containing $(p, \alpha, \beta)$ means that when the two stack NPDA reads (consumes) $a$ from the input, has $X$ and $Y$ on the top of the first and second stack, then it goes to state $p$ while pushing $\alpha$ and $\beta$ on the two stacks respectively (after popping $X$ and $Y$ from the stacks).

   Instantaneous description of the NPDA looks like: $(q, w, \alpha, \beta)$, where $q$ is the current state, $w$ is the input left to be read and $\alpha, \beta$ are the contents of the two stacks.

   $(q, aw, X\alpha, Y\beta) \vdash (p, w, \alpha'\alpha, \beta'\beta)$, if $\delta(q, a, X, Y)$ contains $(p, \alpha', \beta')$.

   $ID \vdash^* ID'$ can then be defined by considering 0 or more steps of $\vdash$.

   For acceptance by final state,

   $L(NPDA) = \{w : (q_0, w, Z_0, Y_0) \vdash^* (q_f, \epsilon, \alpha, \beta)$, for some $q_f \in F$ and $\alpha \in \Gamma_1^*, \beta \in \Gamma_2^*\}$.

Two stack NPDA can accept $\{a^n b^n c^n : n \geq 0\}$, by first pushing the $a$'s in both the stacks, and then comparing $b$'s with $a$'s in the first stack and then comparing $c$'s with the $a$'s in the second stack. However, $\{a^n b^n c^n : n \geq 0\}$ is not context free (to be done in class), and thus not accepted by any NPDA with 1 stack.

In fact, we will later show that two stacks are enough to simulate a Turing Machine, and thus that is as powerful as any computing device.

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 $\epsilon$ 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 $\Sigma$.

   (a) Formally define DPDA.

   Ans: Definition is similar to NPDA, except that for any $\delta(q, a, X)$, we have at most one possibility, and if $\delta(q, \epsilon, X)$ is non-empty, then $\delta(q, a, X)$ is empty for all $a \in \Sigma$.

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

   Ans: For acceptance by final state, answer is yes, as the DPDA can ignore the stack and just simulate the DFA as it is.

   For acceptance by empty stack, DPDA cannot accept $\{a, aa\}$, as if it accepts $a$, then it cannot accept any extension of it. Thus, it cannot accept all regular languages.