# TOC PROBLEM SET-9

SIDDHANT CHAUDHARY
BMC201953

**1.** Construct PDAs for the following languages.
**(a)** $L = \{a^n b^m c^m d^n \mid m, n \in \mathbb{N}\}$.

**Solution.** Let $\Sigma = \{a, b, c, d\}$ and let $\Gamma = \{\perp, A, B, C, D\}$. We will make a PDA with five states, namely $S_a, S_b, S_c, S_d$ and $S_{\text{final}}$. Let the starting state be $S_a$, and the only final state will be $S_{\text{final}}$. The transitions will be as follows.

$$(S_a, \perp) \xrightarrow{a} (S_a, A \perp)$$
$$(S_a, A) \xrightarrow{a} (S_a, AA)$$
$$(S_a, A) \xrightarrow{b} (S_b, BA)$$
$$(S_b, B) \xrightarrow{b} (S_b, BB)$$
$$(S_b, B) \xrightarrow{c} (S_c, \epsilon)$$
$$(S_c, B) \xrightarrow{c} (S_c, \epsilon)$$
$$(S_c, A) \xrightarrow{d} (S_d, \epsilon)$$
$$(S_d, A) \xrightarrow{d} (S_d, \epsilon)$$
$$(S_d, \perp) \xrightarrow{\epsilon} (S_{\text{final}}, \perp)$$

Let me explain the reasoning behind these transitions. We start in the state $S_a$ with the stack being empty. If we read the letter $a$, we keep pushing the symbol $A$ on the top of the stack. Then, if we encounter $b$, we keep pushing the symbol $B$ on the top of the stack. Next, when we encounter a $c$, we keep popping the symbol $B$ from the top of the stack, and when we encounter $d$, we keep popping $A$ from the top of the stack. The transitions are arranged in such a way that if we reach the state $(S_d, \perp)$, a word of the form $a^n b^m c^m d^n$ has been read, where $n, m \in \mathbb{N}$. So, non-deterministically, we can go from $(S_d, \perp)$ to $(S_{\text{final}}, \perp)$ to accept the word.

**(b)** Strings over the alphabet $\Sigma = \{1, +, =\}$ that denote valid equations of sums of unary numbers. Eg:

$$11111 + 111 = 11 + 11 + 1111$$

**Solution.** The idea here is simple: we just need to count the total number of $1's$ on either side of the equation, since we are dealing with unary numbers only. So, in the LHS as we read $1's$, we will keep pushing them on top of our stack, and the $+$ signs will essentially be ignored. When an $=$ sign is read, we must configure the PDA so that if a $1$ is read after, we pop it from the top of the stack

*Date*: October 2020.

(and continue to ignore the $+$ signs). So, let $\Sigma = \{1, +, =\}$ and let $\Gamma = \{\perp, O\}$ (the $O$ stands for one). Our PDA will contain five states, $s, t, t_+, e$ and $f$, where the starting state will be $s$ and the only final state will be $f$. The transitions are described below.

$$
\begin{aligned}
(s, \perp) &\xrightarrow{=} (t, \perp) \\
(s, \perp) &\xrightarrow{1} (s, O\perp) \\
(s, O) &\xrightarrow{1} (s, OO) \\
(s, O) &\xrightarrow{+} (s, +O) \\
(s, +) &\xrightarrow{1} (s, O) \\
(s, O) &\xrightarrow{=} (e, O) \\
(e, O) &\xrightarrow{1} (t, \epsilon) \\
(t, O) &\xrightarrow{1} (t, \epsilon) \\
(t, O) &\xrightarrow{+} (t_+, O) \\
(t_+, O) &\xrightarrow{1} (t, \epsilon) \\
(t, \perp) &\xrightarrow{\epsilon} (f, \perp)
\end{aligned}
$$

And let me explain the transitions. The first transition ensures that the equation $\epsilon = \epsilon$ is accepted. The next four transitions describe the action of the PDA on the LHS: there cannot be consecutive $+$ signs, and the equation must begin and end with a $1$. Finally, if we encounter an $=$ sign, we jump to the state $e$, to ensure that the next acceptable symbol is only $1$. The last three states describe the action of the PDA on the RHS. The point of $t_+$ is to ensure that no consecutive $+$ signs are read. Finally, any word that ends up in $(t, \perp)$ is a valid equation.

**(c)** $L = \{w \mid (w)_a = (w)_b$ and $(w)_a$ is even$\}$.

**Solution.** Let $\Sigma = \{a, b\}$ and let $\Gamma = \{\perp, A, B\}$. Our PDA will contain three states, namely $S_e, S_o$ and $f$ ($S_e$ stands for even number of $a'$s and $S_o$ stands for odd number of $a'$s). The initial state will be $S_e$ and the only final state will be $f$. The transitions are given below.

$$(S_e, \perp) \xrightarrow{a} (S_o, A\perp)$$
$$(S_e, \perp) \xrightarrow{b} (S_e, B\perp)$$
$$(S_e, A) \xrightarrow{a} (S_o, AA)$$
$$(S_e, A) \xrightarrow{b} (S_e, \epsilon)$$
$$(S_e, B) \xrightarrow{a} (S_o, \epsilon)$$
$$(S_e, B) \xrightarrow{b} (S_e, BB)$$
$$(S_o, \perp) \xrightarrow{a} (S_e, A\perp)$$
$$(S_o, \perp) \xrightarrow{b} (S_o, B\perp)$$
$$(S_o, A) \xrightarrow{a} (S_e, AA)$$
$$(S_o, A) \xrightarrow{b} (S_o, \epsilon)$$
$$(S_o, B) \xrightarrow{a} (S_e, \epsilon)$$
$$(S_o, B) \xrightarrow{b} (S_o, BB)$$
$$(S_e, \perp) \xrightarrow{\epsilon} (f, \perp)$$

and the reasoning behind the transitions is rather straightforward: each time we read an $a$ or $b$, we push an $A$ or $B$ or pop an $A$ or $B$ from the stack depending upon which letter is dominating. If we read an $a$, we switch from $S_e$ to $S_o$ and vice-versa. Finally, any word that ends up in $(S_e, \perp)$ is of the given form, and hence we can accept it.

**2.** Let $G$ be a grammar in CNF. Let $a \in \Sigma, X \in N$. Is the language

$$\{\alpha \in N^* \mid X \xrightarrow{*} a\alpha\}$$

a regular language? Why/Why not?

**Solution.** The answer is that it may or may not be regular. I will give an example supporting each case. First, consider the following grammar, which is evidently in CNF:

$$S \to AB \mid AX$$
$$X \to SB$$
$$A \to a$$

I claim that

(†) $\{\alpha \in N^* \mid S \xrightarrow{*} a\alpha\} = \{A^{n-1}B^n | n \in \mathbb{N}\} \cup \{A^{n-1}SB^n | n \in \mathbb{N}\} \cup \{A^{n-1}XB^{n-1}\}$

(I am not proving this, but it is not hard to see that any sentencial form that is generated by $S$ is of the form $A^nB^n$ , $A^nSB^n$ or $A^nXB^{n-1}$ for $n \geq 1$). Now the language appearing in the RHS of equation (†) is *not* regular because its homomorphic image obtained by mapping $A \to A$, $B \to B$, $S \to \epsilon$ and $X \to B$, which is simply

$$\{A^{n-1}B^n \mid n \in \mathbb{N}\}$$

is *not* regular.

Next, consider the following simple grammar.

$$S \to XY$$
$$X \to a$$
$$Y \to a$$

and clearly

$$\{\alpha \in N^* \mid S \xrightarrow{*} a\alpha\} = \{Y\}$$

which is clearly regular. So this shows that the given language may or may not be regular.

**3.** Ogden's lemma is a generalisation of the pumping lemma for context free languages that gives you more control over which portion of the word gets pumped. It states that if $L$ is a context free language, then there is a constant $n$ such that if $z$ is any string of length at least $n$ in $L$, for any choice of at least $n$ positions of $z$ marked as distinguished, we can write $z = uvwxy$ such that:

(1) $vwx$ has atmost $n$ distinguished positions.
(2) $vx$ has atleast one distinguished position.
(3) For all $i$, $uv^iwx^iy$ is in $L$.

Prove Ogden's Lemma.
**Hint:** Can you modify the proof of pumping lemma for CFL's to prove this?

**Solution.** I still have to prove this in my own words. However, I found a link containing a proof.

**4.** Prove the following language is not context free using Ogden's Lemma. Can this be shown using the usual pumping lemma for context free languages? Where does the argument fail there?

$$L = \{a^ib^jc^kd^l \mid i = 0, \text{ or } j = k = l\}$$

**Hint:** For the proof via Ogden's lemma, given pumping length $n$, choose a long enough word in the language which contains atleast one $a$, and none of the distinguished positions are an $a$.

**Solution.** For the sake of contradiction, suppose the given language is context free. Let $n$ be the pumping length as guaranteed by Ogden's Lemma. Consider the following word:

$$ab^{n+1}c^{n+1}d^{n+1}$$

which clearly belongs to our language. Let all the $b'$s, $c'$s and $d'$s be marked as the distinguished positions (so clearly we have marked atleast $n$ positions). So we can write this word as $uvwxy$ where the conditions provided by Ogden's Lemma are satisfied, namely: $vx$ contains atleast one distinguised position, $vwx$ contains atmost $n$ distinguised positions and $uv^iwx^iy$ is in $L$ for all $i \geq 0$. Now, consider the subword $vwx$. Observe that this subword does not contain atleast one of $b, c$ or $d$, for if it contained all these three symbols, it would mean that (since the subword $vwx$ is continguous) $vwx$ contains all of the $n + 1$ $c'$s, which contradicts the fact that $vwx$ contains atmost $n$ distinguised positions. Without loss of generality, suppose the word $vwx$ does not contains the symbol $d$. Also by one of the conditions, we see that $vx$ contains atleast one of the symbols $b$ or $c$. So, if $i$ is large enough, it would mean that the word $uv^iwx^iy$ contains more number of $b$'s or $c'$s than $d'$s, which contradicts the fact that $uv^iwx^iy$ is in $L$ (since this word contains atleast one $a$). So this shows that the given language is *not* context free, completing the proof.

Now if we tried to apply the normal pumping lemma, we could break down the word as

$$ab^{n+1}c^{n+1}d^{n+1} = uvwxy$$

where $u = \epsilon$, $v = a$, $w = \epsilon$, $x = \epsilon$ and $y = b^{n+1}c^{n+1}d^{n+1}$. In that case, all words $uv^iwx^iy$ will be accepted. So by using the idea of distinguished positions, we are able to control the breaking down of the word more.

**5.** Prove that any context-free language $L$ over the alphabet $\Sigma = \{a\}$ is regular. **Hint:** By pumping lemma we know there exists a pumping length $p$. Can words of size greater than $p$ be collected into a finite union of regular languages?

**Solution.** To be completed