

## Solutions for cs154 Homework #2

### Easy Problem 1

(a)  $L = \{0^i 1^j \mid i > j\}$

We will use Pumping Lemma.

$\forall p \geq 1 :$

$\exists s = 0^{p+1} 1^p$ , such that

$\forall x, y, z$  such that  $s = xyz$ ,  $|xy| \leq p$ ,  $|y| \geq 1$ , it is true that

$\exists k$  such that  $xy^k z \notin L$ . The value of  $k = 0$  makes  $xz \notin L$  because the new string is of the form  $0^x 1^p$  where  $x \leq p$ . Thus,  $L$  is not regular.

(b)  $L = \{0^m 10^n 10^{m+n} \mid m, n > 0\}$

We will use Pumping Lemma.

$\forall p \geq 1 :$

$\exists s = 0^p 10^p 10^{2p}$ , such that

$\forall x, y, z$  such that  $s = xyz$ ,  $|xy| \leq p$  and  $|y| \geq 1$ , it is true that

$\exists k$  such that  $xy^k z \notin L$ . The value of  $k = 0$  makes  $xz \notin L$  because the new string has the form  $0^x 10^p 10^{2p}$  where  $x \neq p$ . Thus,  $L$  is not regular.

Note that in both the problems, we *pumped down*, i.e., the choice  $k = 0$  worked.

### Easy Problem 2

#### Part (a)

*Union:* Let  $G_1 = (V_1, \Sigma_1, R_1, S_1)$  and  $G_2 = (V_2, \Sigma_2, R_2, S_2)$  be two CFGs where we have renamed the non-terminals in both grammars so that  $V_1 \cap V_2 = \emptyset$ . We will construct  $G = (V, \Sigma, R, S)$  such that  $V = V_1 \cup V_2 \cup \{S\}$ ,  $\Sigma = \Sigma_1 \cup \Sigma_2$  and  $R$  is the union of  $R_1$ ,  $R_2$  and  $\{S \rightarrow S_1 | S_2\}$ . The new grammar accepts exactly the union of the languages accepted by  $G_1$  and  $G_2$ .

*Concatenation:* Replace the production  $S \rightarrow S_1 | S_2$  by  $S \rightarrow S_1 S_2$  in the para above.

*Star:* Let  $G = (V, \Sigma, R, S)$  be a grammar. The grammar  $G_{new} = (V \cup \{S_{new}\}, \Sigma, R, S_{new})$  with all the rules of  $G$  alongwith the new rule  $S_{new} \rightarrow S_{new} S \mid \epsilon$  accepts  $L(G)^*$ .

#### Part (b)

Consider the parse tree for a regular expression. Leaves will consist of letters of the alphabet. Non-leaves will contain operators. There are three possible operators:  $*$  (closure),  $.$  (concatenation) or  $+$  (union). We construct a CFG with one non-terminal for each non-leaf node. If non-terminal  $A$

corresponds to operator  $*$  and has a child with terminal or non-terminal  $X$ , we create a production  $A \rightarrow AX \mid \epsilon$ . If a non-terminal  $A$  corresponds to operator  $.$  with children  $X_1$  and  $X_2$  (which could be terminals or non-terminals), we create a production  $A \rightarrow X_1X_2$ . If a non-terminal  $A$  corresponds to operator  $+$  with children  $X_1$  and  $X_2$  (which could be terminals or non-terminals), we create a production  $A \rightarrow X_1 \mid X_2$ . Finally, we create a new symbol  $S$ , the start symbol and add a rule  $S \rightarrow X$  where  $X$  is the terminal/non-terminal in the root of the parse tree. The grammar accepts exactly the regular expression we started out with.

### Easy Problem 3

(a)  $L = \{0^i \# 0^j \# 0^k \mid i + j + k \geq 2\} \cup \{0^i \# 0^j \mid j = 2i\}$ .

(b) Let  $L' = 0^* \# 0^*$ . Let  $L_{new} = L \cap L'$ . If both  $L$  and  $L'$  are regular then  $L_{new}$  must be regular as well. We know that  $L'$  is indeed regular. However,  $L_{new} = \{0^i \# 0^j \mid j = 2i\}$ , which is easy to prove non-regular using the Pumping Lemma. Therefore,  $L$  must have been non-regular.

### Easy Problem 4

(a) After removal of  $\epsilon$  productions:

$$S \rightarrow aa \mid aAa \mid bb \mid bBb \mid \epsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow C \mid b$$

$$C \rightarrow CDE$$

$$D \rightarrow A \mid B \mid ab$$

(b) After removal of unit productions:

$$S \rightarrow aa \mid aCa \mid aaa \mid bb \mid bCb \mid bbb \mid \epsilon$$

$$C \rightarrow CCE \mid CaE \mid CbE \mid CabE$$

(c) After removal of useless symbols:

$$S \rightarrow aa \mid aaa \mid bb \mid bbb \mid \epsilon$$

(d) Chomsky normal form:

$$S_0 \rightarrow AA \mid AX \mid BB \mid BY \mid \epsilon$$

$$X \rightarrow AA$$

$$Y \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

### Easy Problem 5

- (a)  $S \rightarrow 0 \mid 1 \mid 0S0 \mid 1S1 \mid \epsilon$   
 (b) Drawn later.

### Problem 1 (10 points)

**Claim 1:** If  $M$  accepts some string  $w$  with  $|Q| \leq |w| \leq 2|Q|$ , then  $M$  accepts an infinite language

**Proof:** Let  $w = w_1w_2 \dots w_\ell$  where  $\ell = |w|$ . Let  $q_0, q_1, \dots, q_\ell$  be a sequence of states such that  $q_0$  is the initial state and  $\delta(q_{i-1}, w_i) = q_i$  for  $1 \leq i \leq \ell$ . Since  $\ell = |w| \geq |Q|$ , there are at least  $|Q| + 1$  states in the sequence  $q_0, q_1, \dots, q_\ell$ . By Pigeonhole Principle, there must be at least one state that repeats. Let these states be  $q_g$  and  $q_h$  where  $0 \leq g < h \leq \ell$ . Let  $y = w_gw_{g+1} \dots w_{h-1}$ . Since  $g < h$ ,  $|y| \geq 1$ . This allows us to write  $w = xyz$  where  $|y| \geq 1$  such that  $xy^kz \in L(M)$  for any  $k \geq 0$ .

**Claim 2:** If  $M$  accepts no string  $w$  with  $|Q| \leq |w| \leq 2|Q|$ , then  $M$  accepts no string  $u$  with  $|u| > 2|Q|$ .

**Proof:** Proof by contradiction.

Let us assume that  $M$  accepts no string  $w$  with  $|Q| \leq |w| \leq 2|Q|$  but accepts some  $u$  with  $|u| > 2|Q|$ . We will arrive at a contradiction.

Let  $\hat{u}$  denote the string of shortest length satisfying  $u \in L(M)$  and  $|u| > 2|Q|$  (the shortest length is unique but there might be several strings with that length; if so, we select any one of them arbitrarily).

Let  $\hat{u} = u_1u_2 \dots u_\ell$  where  $\ell = |\hat{u}|$ . Let  $q_0, q_1, \dots, q_\ell$  be a sequence of states such that  $q_0$  is the initial state and  $\delta(q_{i-1}, u_i) = q_i$  for  $1 \leq i \leq \ell$ . Since  $\ell = |\hat{u}| > 2|Q|$ , there are at least  $2|Q| + 1$  states in the sequence  $q_0, q_1, \dots, q_\ell$ . Consider just the first  $|Q| + 1$  states:  $q_0, q_1, \dots, q_{|Q|+1}$ . By Pigeonhole Principle, there must be at least one state that repeats. Let these states be  $q_g$  and  $q_h$  where  $0 \leq g < h \leq |Q| + 1$ . Let  $y = u_gu_{g+1} \dots u_{h-1}$ . It follows that  $1 \leq |y| \leq |Q|$ . This allows us to write  $\hat{u} = xyz$  where  $1 \leq |y| \leq |Q|$  such that  $xy^kz \in L(M)$  for any  $k \geq 0$ . It follows that  $xz \in L(M)$ . Now  $|xz| \leq 2|Q|$  because we started with  $\hat{u} = xyz$  having the shortest possible length among those that satisfy both  $u \in L(M)$  and  $|u| > 2|Q|$ . Since  $|y| \leq |Q|$ , it follows that  $|xz| = |xyz| - |y| \geq |Q|$ . Thus we have discovered a string  $w = xy$  such that  $w \in L(M)$  and  $|Q| \leq |w| \leq 2|Q|$ , a contradiction.

### Problem 2(a) (6 points)

Grammar  $G = (V, \Sigma, R, S)$  with  $V = \{S, X, Y, C, A\}$ ,  $\Sigma = \{a, b, c\}$  and rules:

$$S \rightarrow XC \mid AY$$

$$\begin{array}{l}
X \rightarrow aXb \mid \epsilon \\
Y \rightarrow bYc \mid \epsilon \\
C \rightarrow cC \mid \epsilon \\
A \rightarrow aA \mid \epsilon
\end{array}$$

### Problem 2(b) (4 points)

Drawn later.

### Problem 3 (10 points)

We need to show that (a)  $G$  always generates a string that has an equal number of  $a$ 's and  $b$ 's and (b) Any string with an equal number of  $a$ 's and  $b$ 's can be generated by  $G$ . Part (a) is easy to show: Any derivation in  $G$  replaces  $S$  by either  $aSbS$ ,  $bSaS$  or  $\epsilon$ . All of them maintain the equality of the number of  $a$ 's and  $b$ 's. We now prove part (b):

Proof by induction on length of string. Strings that have an equal number of  $a$ 's and  $b$ 's will have even length. Let length be  $2k$  where  $k \geq 0$ . Only one string  $\epsilon$  corresponds to  $k = 0$  and  $\epsilon \in L(G)$ . We will proceed with induction for  $k \geq 1$ .

**Base case:**  $k = 1$ . There are two strings  $ab$  and  $ba$ . Both belong to  $L(G)$ .

**Induction step:** Assuming that all strings with length  $2, 4, \dots, 2k$  are derivable from rules in  $G$ , we will show that all strings of length  $2k + 2$  are derivable from rules in  $G$ .

Let  $w$  be a string with length  $2k + 2 \geq 4$  with an equal number of  $a$ 's and  $b$ 's. Consider the case when the first character in  $w$  happens to be  $a$  (proof for  $b$  follows the same lines as the problem is symmetric in  $a$  and  $b$ ). Let us analyze the tail of this string consisting of  $2k + 1$  characters. Start scanning the tail from left to right and identify the first  $b$  where the number of  $b$ 's (in the tail) exceeds the number of  $a$ 's (in the tail) by one. Such a  $b$  exists because the tail consists of  $k$   $a$ 's and  $k + 1$   $b$ 's. This  $b$  splits the tail such that we can write  $w = av_1bv_2$  where  $v_1$  and  $v_2$  have equal number of  $a$ 's and  $b$ 's. Note that it is possible for  $v_1$  or  $v_2$  to be empty. It follows that  $|v_1| + |v_2| = 2k$ . Since both  $v_1$  and  $v_2$  have length at most  $2k$ , by the induction hypothesis, there exist sequences of derivations such that  $S \xrightarrow{*} v_1$  and  $S \xrightarrow{*} v_2$ . From this, we infer that the following derivation exists:  $S \rightarrow aSbS \xrightarrow{*} av_1bS \xrightarrow{*} av_1bv_2 = w$ .

### Problem 4(a) and 4(b) (6 points + 4 points)

Drawn later. It is an interesting exercise to formally prove that the machine drawn for Problem 4(b) indeed accepts the language it is designed for.

## Problem 5 (10 points)

This problem is tricky. A PDA that accepts  $L$  is drawn. The top part of the PDA accepts strings  $x\#y$  where  $|x| \neq |y|$ . The bottom part of the PDA accepts strings  $x\#y$  where  $x$  and  $y$  differ in at least one bit position. Thus, some strings can be accepted by either the top part or the bottom part, which is okay.

The top part works as follows. Having pushed  $\$$  onto the stack, we push  $|x|$   $\pounds$  symbols onto the stack. When we encounter  $\#$ , we start popping one  $\pounds$  for each 0 or 1 we encounter. Then, we transit to another state by either (a) popping  $\$$  when we see 0 or 1 (which means that  $|y| > |x|$ ), or (b) popping  $\pounds$  and then emptying the stack of all  $\pounds$  and  $\$$  symbols (which means that  $|x| > |y|$ ). The top part is reminiscent of the PDA for Problem 4(a).

The bottom part accepts a string of the form  $x\#y$  where  $x$  and  $y$  differ in at least one bit position. We first push zero or more  $\pounds$  symbols onto the stack. Then, if we encounter 1, we push 0; otherwise we push 1. We ignore the rest of  $x$ . Effectively, we have *guessed* the position and *remembered* the value of the bit which we will make sure is different in  $y$ . The position is captured by the size of the stack. The value is captured by the top value in the stack. Upon encountering  $\#$ , we inspect the top of the stack. We jump to either of two states, depending on the bit value lying on top of the stack. We then ignore as many bits as the number of  $\pounds$  symbols we had remembered. The next bit is the crucial one. We jump to the next state only if this bit is the same as the bit we had popped from the stack a short while ago. Thereafter, we pop  $\$$ , read the remainder of  $y$  and accept.

Here is a CFG that accepts the language:

$$\begin{aligned} S &\rightarrow S_0 \mid S_1 \\ S_0 &\rightarrow AS_0A \mid AB\# \mid \#AB \\ S_1 &\rightarrow X1B \mid Y0B \\ X &\rightarrow AXA \mid 0B\# \\ Y &\rightarrow AYA \mid 1B\# \\ B &\rightarrow AB \mid \epsilon \\ A &\rightarrow 0 \mid 1 \end{aligned}$$

Intuition:  $A = 0 + 1$  and  $B = (0 + 1)^*$ .  $S_0$  derives a string  $x\#y$  where  $|x| \neq |y|$ .  $X$  derives a string of the form  $v0(0 + 1)^*\#w$  such that  $|v| = |w|$ . Now,  $S_1 \rightarrow X1B$ . Therefore  $S_1$  derives strings of the form  $v0(0 + 1)^*\#w1(0 + 1)^*$  where  $|v| = |w|$ . Similarly,  $S_1 \rightarrow Y0B$  derives strings of the form  $v1(0 + 1)^*\#w0(0 + 1)^*$  with  $|v| = |w|$ .