

## Problem Set 5 Solutions

**Problem 1.** *Decide if the following languages are regular or not. Prove your answers.*

**Part A.**  $L = \{w \in \{0, 1, 2\}^* : w \text{ has an equal number of } 01\text{'s and } 10\text{'s}\}$ .

Not regular. Suppose  $L$  were regular. Then so would be the language  $L' = L \cap (012 \cup 102)^*$ . Let  $N$  be the pumping length for  $L'$  and consider the string  $s = (012)^N(102)^N$ . By the semi-strong form of the pumping lemma we can decompose  $s$  into  $s = xyz$  where  $|xy| \leq N$ ,  $|y| \geq 1$ , and  $xz \in L'$ . But any decomposition of  $s = xyz$  where  $|xy| \leq N$ ,  $|y| \geq 1$ , just to have  $xz$  in  $(012 \cup 102)^*$ , would have  $xz$  look like  $(012)^n(102)^N$  for  $n < N$ . No such string is in  $L'$ .

**Part B.**  $L = \{w \in \{0, 1\}^* : w \text{ has an equal number of } 01\text{'s and } 10\text{'s}\}$ .

Regular. What do strings look like that have an equal number of 01's and 10's? If a string starts with a 0 and ends with a 0 it will have an equal number of 01's and 10's; if a string starts with a 1 and ends with a 1 it will have an equal number of 01's and 10's; if a string starts with a 0 and ends with a 1 it will have a different number of 01's and 10's; if a string starts with a 1 and ends with a 0 it will have a different number of 01's and 10's; if a string is empty it will have an equal number of 01's and 10's. Thus the language in question is  $D = 0^* \cup 1^* \cup 0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1$ .

**Problem 2.** *Give an algorithm to solve the following decision question: given a regular expression  $\alpha$ , is  $L(\alpha) = (L(\alpha))^R$ ?*

The NFA-acceptable languages are closed under reversal: the proof is to take an NFA  $M$  and convert it to an NFA  $M^R$ , where  $L(M^R) = (L(M))^R$ , by adding a new start state, connecting it to all the old final states, definalizing those final states, and finalizing the start state. Thus an algorithm to answer this question is as follows: convert  $\alpha$  into an NFA  $M$ ; construct the NFA  $M^R$  as above; and apply the procedure we did (convert to a DFA and use the product construction for symmetric difference, then DFS to decide emptiness)

**Problem 3.** *Are the following statements true or false? Either prove the statement or give a counter-example to it.*

**Part A.** *If  $L \cup L'$  is regular then  $L$  and  $L'$  are regular.*

False.  $L = \{a^n b^n : n \geq 0\}$  and  $L' = \{a, b\}^*$ .

**Part B.** *If  $L^*$  is regular then  $L$  is regular.*

False.  $L = \{1^{2^i} : i \geq 0\}$ .

**Part C.** *If  $LL'$  is regular then  $L$  and  $L'$  are regular.*

False.  $L = \{a^n b^n : n \geq 0\}$  and  $L' = \emptyset$ .

**Part D.** *If  $L$  and  $L'$  agree on all but a finite number of strings, then one is regular iff the other is regular.*

True.  $L \oplus R = L'$  and for some finite, and therefore regular,  $R$ . But the regular languages are closed under symmetric difference.

**Part E.** If  $R$  is regular,  $L$  is not regular, and  $L$  and  $R$  are disjoint, then  $L \cup R$  is not regular.

True. Suppose instead that  $L \cup R$  were regular. Then  $(L \cup R) \setminus R = L$  by disjointedness, and the regular languages are closed under symmetric difference, so  $L$  would be regular.

**Problem 4.** Define  $A = \{x \in \{a, b, \#\}^* : x \text{ contains an equal number of } a\text{'s and } b\text{'s or } x \text{ contains consecutive } \#\text{'s or letters}\}$ . Prove that  $A$  is not regular.

*Hint: let  $h : \Sigma \rightarrow \Gamma^*$  be function and extend  $h$  to strings and then to languages in the natural way. Show that if  $C$  is regular then so is  $h(C)$ . We say that “the regular languages are closed under homomorphisms.” Consider using this, as well as Problem 3E.*

As we explained in class, the pumping lemma won't work for this. It won't work because, whatever string  $s \in L$  you choose, the string will pump. In particular, the portion we usually denote  $y$  might be a single character, and repeating that character (or excising it) will give strings in  $A$ .

So to prove this will need some finesse. Define  $L = \{x \in \{a, b, \#\}^* : x \text{ contains an equal number of } a\text{'s and } b\text{'s and every other character is a } \#\}$ . Let  $R = (a \cup b \cup \#)^*((a \cup b)(a \cup b) \cup \#\#)(a \cup b \cup \#)^*$ . Then  $R$  is regular and  $L$  and  $R$  are disjoint, and  $L \cup R = A$ , so, to show that  $A$  is not regular it is enough to show that  $L$  is not regular.

Let  $h : \Sigma \rightarrow \Gamma^*$  and extend  $h$  character-wise to strings (ie  $h(a_1 \cdots a_n) = h(a_1) \cdots h(a_n)$ ) and string-wise to languages (ie  $h(C) = \{h(x) : x \in C\}$ ). We claim that if a language  $C$  is regular then so is  $h(C)$ . For if we are given a regular expression  $\alpha$  for  $C$  then (the properly parenthesized version of)  $h(\alpha)$  is a regular expression for  $h(L(\alpha))$ .

Now consider the specific map  $h$  where  $h(a) = a$ ,  $h(b) = b$ , and  $h(\#) = \varepsilon$ . Then  $h(L)$ , for the  $L$  we specified above, is the set  $L'$  of all strings over  $\{a, b\}$  with an equal number of  $a$ 's and  $b$ 's. We know this language to be not regular (we showed it in class, or you can show it with the pumping lemma or with closure properties). So  $L$  is not regular, and so  $A$  is not regular.

**Problem 5.** Give a context free grammar for  $L = \{a^n b^m : n \neq 2m\}$ . Make your grammar unambiguous—and explain why it is unambiguous.

A grammar for the language is

$$\begin{aligned} S &\rightarrow aaSb \mid A \mid B \\ A &\rightarrow aA \mid a, \\ B &\rightarrow bB \mid b. \end{aligned}$$

The idea is to generate twice as many  $a$ 's as  $b$ 's—and then generate some additional  $a$ 's (at least one) or some additional  $b$ 's (at least one). The grammar is not ambiguous: any string  $a^n b^m$  where  $n \neq 2m$  is either of the form  $a^m a^m a^i b^m$  where  $i \geq 1$  or else of the form  $a^m a^m b^i b^m$  where  $i \geq 1$ . Strings of the first kind have a unique parse tree, corresponding to applying the first rule  $m$  times, applying the second rule once, applying the fourth rule

$i - 1$  times, and then applying the fifth rule once. Strings of the second kind have a unique parse tree, corresponding to applying the first rule  $m$  times, applying the third rule once, applying the sixth rule  $i - 1$  times, and then applying the seventh rule once.

**Problem 6.** *Prove that the context-free languages are closed under reversal.*

Let  $G = (V, \Sigma, R, S)$  be a CFG for some CFL  $L$ . Construct a CFG  $G^R = (V, \Sigma, R', S)$  by replacing each rule  $A \rightarrow \alpha$  in  $R$  by the rule  $A \rightarrow \alpha^R$  for  $R'$ .

I claim that  $L(G) = (L(G^R))^R$ . One way to see this is to observe that every parse trees for  $G^R$  is just just the mirror image of a corresponding parse tree for  $G$ . If you don't like this "geometric" viewpoint, you can establish the correctness of this construction by induction. You argue, by induction on the length of the derivations, that for all  $A \in V$  the language  $L^R(A)$  of sentential forms  $\alpha$  derivable from  $A$  in  $G^R$  is exactly the reversal of the language  $L(A)$  of sentential forms derivable from  $A$  in  $G$ .