COMP 330 Winter 2021
Assignment 3 Solutions

Prakash Panangaden

March 14, 2021

Question 1[20 points] Are the following statements true or false? Prove your answer in each case. We have some fixed alphabet Σ with at least two letters. In the following A and B stand for languages, *i.e.* subsets of Σ^*.

- If A is regular and $A \subseteq B$ then B must be regular. [3]
- If A and AB are both regular then B must be regular. [7]
- If $\{A_i | i \in \mathbb{N}\}$ is an infinite family of regular sets then $\bigcup_{i=1}^{\infty} A_i$ is regular. [5]
- If A is not regular it cannot have a regular subset. [5]

Solutions

- Certainly not! Any finite set is regular so we can take B to be any non-regular language and A to be a finite subset of B.
- This is also false. If A is Σ^* and B is anything at all, other than $\emptyset$ then $AB = \Sigma^*$ which is regular but I can choose B to be nastily non-regular.
- This is also false. I can take $A_i = \{a^i b^i\}$ which is regular because it is finite but the union over all i is our basic example of a non-regular set $\{a^n b^n | n \geq 0\}$.
- Certainly not, see part 1.

Question 2[20 points]
Show that the following language is not regular using the pumping lemma.

$$\{a^n b^{2n} | n > 0\}$$

Solution I will format the answer in the form of the game against the demon.

1. Demon chooses some pumping length $p > 0$.

2. I choose $a^p b^{2p}$.

3. The demon's choice of x, y is constrained by the requirement that $|xy| \leq p$ and $|y| > 0$. Thus, y has to consist exclusively of a's and cannot be the empty string: $y = a^k$ for some $p \geq k > 0$.

4. I choose $i = 2$, so I get my pumped string to be $a^{p+k}b^{2p}$ and since $2p + 2k \neq 2p$ we have a pumped string that is not in the language.

This shows that we have a winning strategy against the demon and the language is therefore not regular.

Question 3[20 points] Show that the language

$$F = \{a^i b^j c^k | i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}$$

is not regular. Show, however, that it satisfies the statement of the pumping lemma as I proved it in class, i.e. there is a p such that all three conditions for the pumping lemma are met. Explain why this does not contradict the pumping lemma.

Solution: First, I will show that F is not regular. Obviously we cannot use the pumping lemma *directly*, but we can use the closure properties of regular languages to massage the language first so that we get something to which we can apply the pumping lemma. If F were regular, than the intersection of F and any regular language would also be regular. We know that $L = \{ab^i c^j | i, j \geq 0\}$ is a regular language, and clearly $L \cap F = \{ab^n c^n\}$, since any word in L must begin with exactly one a, and so if it is also in F then $i = j$. But we know that $\{ab^n c^n\}$ is not regular by an easy variation of the example done in class, so F must not be regular.

Next I will show that F satisfies the pumping lemma. This is also done with games but we are using the pumping lemma not its negation so the quantifiers, and hence the moves, are reversed. *You* choose p, the demon chooses a word, you choose x, y, z subject to the constraints and the demon chooses i.

Let $p = 2$. Then the demon gives you a word w from F, of the form $a^l b^j c^k$. First, consider the case where $l = 0$, so $w = b^j c^k$. Then choose $x = \epsilon$ and $y = b$ if the first letter is a b, or $y = c$ otherwise. Let z be the rest of the word. If $y = b$, then $z = b^{j-1}c^k$, and pumping results in a string of the form $b^n b^{j-1} c^k$, which is clearly in F for any n. If $y = c$, then $z = c^{k-1}$. So pumping results in a string of the form $c^n c^{k-1}$ which is also clearly in F.

Next consider the case where $l = 1$, so $w = ab^j c^j$. Again let $x = \epsilon$, $y = a$, and $z = b^j c^j$. Then pumping results in a string of the form $a^n b^j c^j$, which is clearly in F.

Next consider the case where $l = 2$, so $w = aab^j c^k$. Now we can't choose $x = \epsilon$ and $y = a$, because if we do, then $xy^0 z = ab^j c^k$ is not necessarily in our language. So choose $y = aa$.

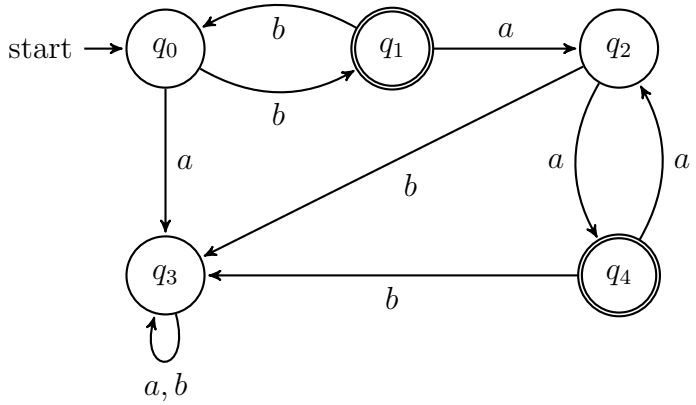
Then pumping cannot result in a string of the form ab^jc^k, because we either pump down and have no a's, or we pump up and have more than one a. So pumping results in strings which are still in our language.

Finally, consider the case where $l > 2$. Again choose $x = \epsilon$ and $y = a$. If we pump down, we remove only one a, so our string still has at least two a's, and j and k don't have to match. If we pump up, the string is clearly still in our language. This proves that the language satisfies the pumping lemma.

This does not contradict the pumping lemma because the pumping lemma claims that all regular languages are "pumpable," which is not the same as claiming that all languages that are "pumpable" must be regular. Logically, $p \Rightarrow q$ is not the same as $q \rightarrow p$.

Question 4[20 points] Let D be the language of words w, over the alphabet $\{a, b\}$ such that w has an even number of a's and an odd number of b's and does not contain the substring ab. By this I mean that once you see an a you can never have a b later on.

1. Give a DFA with *only five* states, including any dead states, that recognizes D.
   Solution Once you see an a you can never have a b later on. If your first letter is an a it has to be rejected because there is now no way to get a b and the word has to have at least one b. The picture is here.



2. Give a regular expression for this language.
   Solution $b(bb)^*(aa)^*$.

Question 5[20 points] Consider the language $L = \{a^n b^m | n \neq m\}$; as we have seen this is not regular. Recall the definition of the equivalence $\equiv_L$ which we used in the proof of the Myhill-Nerode theorem. Since this language is not regular $\equiv_L$ cannot have finitely many equivalence classes. Exhibit explicitly, infinitely many distinct equivalence classes of $\equiv_L$.

Solution Let me clear up some misconceptions. First, the relation $\equiv_L$ is defined between *all* pairs of strings, not just strings in L. Second, we require either that both xz and yz are

in L *or* that *neither of them* are in L; some of you seem to think that the defining condition states that both xz and yz *must be in* L. Third this condition has to hold for *all* $z \in \Sigma^*$ not just one z[1].

I claim that $a^i \not\equiv_L a^j$ for $i \neq j$. Recall what the relation $\equiv_L$ means. It has a *universal* quantifier in the definition, so the **negated version** has an existential quantifier in it. In order to check that $x \not\equiv_L y$ we need to find *one* string z such that $xz \in L$ and $yz \notin L$ or vice versa. In our case, we have $x = a^i$ and $y = a^j$ so consider $z = b^i$. Clearly $xz = a^i b^i \notin L$ and $yz = a^j b^i \in L$. Thus the equivalence classes $[a^i]$ are different for every i and I have now shown you infinitely many distinct equivalence classes.

[1]Did I ever mention the importance of reading the quantifer?