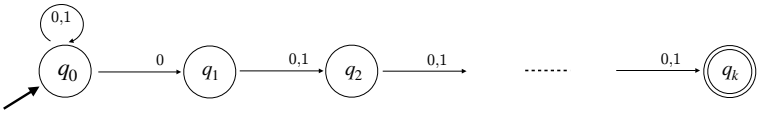
CS–172 Computability & Complexity, Spring 2021

Homework 3 Solutions

Note: These solutions are not necessarily model answers. Rather, they are designed to be tutorial in nature, and sometimes contain more explanation than is required for full points. Also, bear in mind that there may be more than one correct solution. The maximum total number of points available is 32.

1. (a) An NFA for L_k is as shown. At each step, if it sees a 0 it "guesses" if this is in fact the kth last symbol by going into state q_1; from there, it verifies its guess by moving in exactly $k-1$ more steps to the final state. This NFA has $k+1$ states. *2pts*



(b) We construct a DFA such that each of the states represents one of the 2^k possible length-k suffixes. Exactly half of these states, those representing suffixes that start with 0, are accepting states. (If a string ends up in one of the other states, the string must have had a 1 in the kth position from the end, and therefore must be rejected.) We transition from state to state by simply lopping off the first digit of the suffix and appending the next character to be read. (For instance, if $k = 4$ then one transition is $(s_{1001}, 0) \to s_{0010}$.) *2pts*

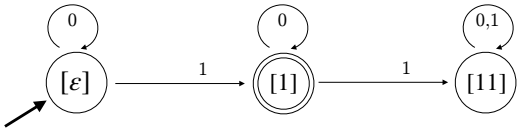
$$\begin{aligned} Q &= \{s_z : z \in \{0,1\}^k\} \\ \Sigma &= \{0,1\} \\ \delta &= \{(s_{aw}, b) \to s_{wb} : a, b \in \{0,1\}, w \in \{0,1\}^{k-1}\} \\ q_0 &= s_{1^k} \\ F &= \{s_z : z \text{ starts with } 0\} \end{aligned}$$

Note the choice of initial state $q_0 = s_{1^k}$. This is in line with the fact that no zero has been seen when the computation begins.

(c) It suffices to identify a set of 2^k distinguishable strings w.r.t. L_k. The set we choose will be the set of all length-k strings over $\{0,1\}$, which obviously has the right cardinality. To see that these are all pairwise distinguishable, consider any two length-k strings x, y with $x \neq y$. Since $x \neq y$, they differ in at least one position: so suppose without loss of generality that $x_i = 0$, $y_i = 1$. Let $w = 1^{i-1}$. Then $xw \in L_k$ and $yw \notin L_k$. (This is because in xw the kth letter from the end is x_i, and similarly for yw.) Thus x and y are distinguishable and we are done. *3pts*

2. (a) Three distinguishable strings are ε, 1 and 11. (Other choices are possible, but this is the simplest and most natural.) To see that they are distinguishable, note that only 1 is in L, so it is distinguishable from both of the others (by taking $z = \varepsilon$). And ε is distinguishable from 11 via the distinguishing string $z = 1$ (because $\varepsilon 1 = 1 \in L$ while $111 \notin L$). *2pts*

(b) Following the proof of the Myhill-Nerode theorem, and assuming for the moment that the maximal size of a set of distinguishable strings in L is 3, we construct a DFA with three states, one for each of the above equivalence classes in the relation $\sim_L$. The transitions between equivalence classes are determined by the transitions between their representative elements, namely ε, 1 and 11. This gives us the machine below. We can immediately see that this machine does indeed recognize the language L, and hence must indeed be the minimal DFA for L (since any DFA for L must have at least three states). *2pts*

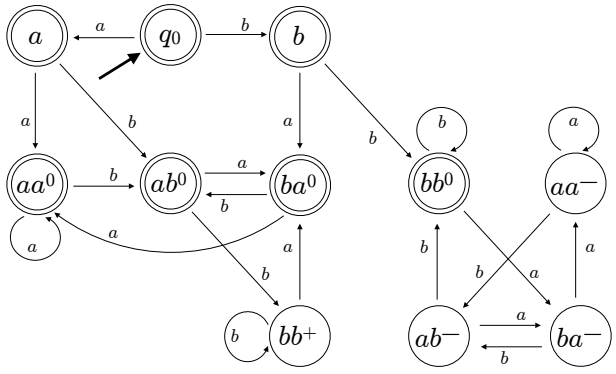


3. (a) Not regular. To prove this, it suffices to find an infinite set of pairwise distinguishable strings w.r.t. this language L. We claim that the set $S = \{(^i : i \geq 1\}$ is such a set. To see that any two strings, $x = (^i$ and $y = (^j$ for $i \neq j$, are distinguishable, we can use the distinguishing string $z =)^i$, for then $xz \in L$ while $yz \notin L$. *2pts*

(b) Not regular. Again it suffices to find an infinite set of distinguishable strings. We can take $S = \{0^i : i \geq 1\}$. To see that strings $x = 0^i$ and $y = 0^j$ (with $i \neq j$) are distinguishable, we can use the distinguishing string $z = 1^i$, for then $xz \notin L$ and $yz \in L$. *2pts*

(c) Regular. The key observation here is that successive occurrences of abb and of bba in any string over $\{a, b\}$ must *alternate* along the string. To see this, we can show that in any string w, between any two occurrences of abb there is an occurrence of bba and vice versa. Consider an arbitrary substring of w delimited by two occurrences of abb. This string has the form $abbuabb$, where u is a possibly empty string. If u contains no a symbols, then the string $bbua$ ends in bba. Otherwise, suppose that the first a in u occurs at position i; then the string $bbu_1 \ldots u_i$ ends in bba. For the other direction, again consider an arbitrary substring of w delimited by two occurrences of bba. Then the *reversal* w^R of w has the form $abbuabb$ for some string u. By the above argument, w^R must contain bba as a substring, so w itself contains an occurrence of abb. *3pts*

For a string w, let $D(w)$ denote the difference between the number of occurrences of abb and of bba in w. By the above argument, for any w, $|D(w)| \leq 1$. At this point it is not too difficult to see what a DFA for our language should look like. The states should keep track of the last two symbols seen, as well as the sign of the quantity $D(w)$ (indicated by the superscripts in the figure below). The accepting states are those whose superscript is neither $+$ nor $-$.



[*Note: Many students gave complicated or messy constructions that were not easy to understand or check, and did not provide a clear explanation for their construction. We graded this part leniently: if you got full points, this does not necessarily mean that your construction is completely correct!* **In future, always give a high-level description of your constructions even if it is not explicitly asked for.**]

(d) Not regular. Again, we need to find an infinite set of distinguishable strings w.r.t. this language. We take the set $S = \{a^i : i \geq 3\}$. Then to distinguish strings $x = a^i$ and $y = a^j$, with $i \neq j$, we can take the string $z = b^i$, since xz has $i - 2$ occurrences of both aaa and bbb, so is in L, while yz has $j - 2$ occurrences of aaa and $i - 2$ occurrences of bbb, so is not in L. *2pts*

4. To show that PRIMES is not regular, as usual it suffices to find an infinite set of pairwise distinguishable strings w.r.t. PRIMES. This is a little trickier than our other examples. Following the hint, we will actually take the set PRIMES itself, and show that the strings $1^p, 1^q$, for any pair of distinct primes $p > q$, are distinguishable. Since there are infinitely many primes, we can immediately deduce that PRIMES is not regular. *4pts*

To prove that $1^p, 1^q$ are distinguishable, we use a proof by contradiction. Suppose not. Then it must be the case that, for *every* $\ell \geq 0$, the strings $1^p 1^\ell$ and $1^q 1^\ell$ are either both in PRIMES or both not in PRIMES. I.e., the integers $p + \ell$ and $q + \ell$ are either both prime or both composite. We apply this fact for the sequence of values $\ell = k(p - q)$, where $1 \leq k \leq p$.

In the case $k = 1$, we get that $2p - q$ and p are both prime or both composite; since p is prime, this means that $2p - q$ is also prime.

In the case $k = 2$, we get that $3p - 2q$ and $2p - q$ are both prime or both composite, which together with the previous paragraph implies that $3p - 2q$ is prime.

Proceeding in this way, we deduce that all the numbers $p + k(p - q)$ are prime, and hence in particular, setting $k = p$, we get that $p + p(p - q) = p(1 + p - q)$ is prime, which is clearly a contradiction. Hence $1^p, 1^q$ are distinguishable, as claimed.

5. [*Note: In parts (a), (c), (d) of this problem, you must give a* **concrete** *counterexample, for explicit languages L, L' etc. It is not enough to talk about abstract languages here.*]

(a) False. E.g., $L' = \{11\}$ is regular over $\Sigma = \{1\}$ but $L = \{1^p : p \text{ is a prime}\}$ is not. *2pts*
Many other counterexamples are possible. Note that it is not *a valid counterexample to exhibit a regular language which contains a non-regular language!*

(b) For any such L (in fact, it doesn't even have to be regular), the language L_{100} is finite (since it contains exactly 100 strings). Therefore, it is certainly regular. (Any finite language can be written as a regular expression that is just the union over its strings.) *2pts*
[*Note: Some students said that the original DFA or NFA for L could just recognize the first 100 strings in it. But there is no way to modify the original FA to do this. Note that in fact this proof is* non-constructive*, in the sense that if we are given a DFA (or regular expression) for L, we don't have any easy way to convert it into an FA (or regular expression) for L_{100}, even though we know that these objects exist!*]

(c) False. E.g., let $\Sigma = \{0, 1\}$, $L = \{0^n 1^n : n \geq 0\}$, and $L' = \{0^m 1^n : m \neq n\}$. Then L and L' are both non-regular. However $L_1 \cap L_2 = \varnothing$, which is regular. *2pts*
[*Again, many other counterexamples are possible.*]

(d) False. For a counterexample, let L be any non-regular language (e.g., $L = \{0^i 1^i : i \geq 0\}$). Then we can write $L = \bigcup_{i=1}^{\infty} L_i$, where each L_i consists just of the ith string in L in lexicographic order. Clearly each L_i is finite and hence regular. However, the union of all of the L_i is L, which is not regular. *2pts*

[*Note that we have seen in class that the union $L_1 \cup L_2$ of two regular languages is regular. By induction, this implies that any finite union $\bigcup_{i=1}^{n} L_i$ is also regular. However, it says nothing about countably infinite unions as in this problem.*]

6. Call the given DFA M. Following the algorithm in Note 1, we start with the base relation $\equiv_M^0$, whose equivalence classes correspond to accepting/non-accepting states, i.e., they are $\{E\}$ and $\{A, B, C, D, F, G\}$. *4pts*

Next we construct the relation $\equiv_M^1$. Recall that

$$p \equiv_M^1 q \quad \Leftrightarrow \quad (p \equiv_M^0 q) \wedge (\forall a)(\delta(p, a) \equiv_M^0 \delta(q, a)).$$

I.e., we need to find those pairs of states p, q that are in the same equivalence class of $\equiv_M^0$ and for which $\delta(p, a)$, $\delta(q, a)$ are also in the same equivalence class for both $a = 0$ and $a = 1$.

Checking each pair in the non-trivial class $\{A, B, C, D, F, G\}$, we see that the equivalence classes of $\equiv_M^1$ are $\{A, C, G\}$ and $\{B, D, F\}$ as well as the previous class $\{E\}$. (To see this, note that all transitions out of states A, B, C, D, F, G go to this same set, with the exception of the 1-transitions out of B, D, F, all of which go to E.)

Next we construct the relation $\equiv_M^2$ by applying exactly the same rule as above to the three equivalence classes of $\equiv_M^1$. We find that all 0-transitions out of A, C, G go to the class $\{B, D, F\}$, and all 1-transitions out of A, C, G go to the class $\{A, C, G\}$. Similarly, all 0-transitions out of B, D, F go to $\{B, D, F\}$ and all 1-transitions go to $\{E\}$. Hence the equivalence classes don't change (i.e., $\equiv_M^2$ is the same relation as $\equiv_M^1$), so our algorithm terminates. The resulting minimal DFA for this language has three states (one for each of the above equivalence classes), and is shown below.