Release date: March 6, 2021 Deadline: March 13, 2021: 23:00

This homework is primarily about proving that certain languages L are not regular. For this, we have the Pumping Lemma and the Myhill-Nerode Theorem at our disposal. Recall that the Pumping Lemma merely gives a sufficient condition for non-regularity. In some cases, using closure properties might give a much cleaner proof: assume that L is regular, then argue that some other language L' must also be regular, then apply Pumping Lemma to show that L' is, in fact, not regular. Remember that you can use any claim proven in class and in the previous quizzes and homeworks without reproducing its proof.

Problem 1

Prove that the language $\{x \mid x \text{ is the binary representation of } 3^{n^2} \text{ for some } n \in \mathbb{N}\}$ is not regular.

Solution. We use the pumping lemma to show that this language is not regular. Firstly, a note about an ambiguity in the problem statement is in order.

Note

Note that there are two interpretations of the problem, one where the binary representation has leading zeroes and the other one where leading zeroes are not allowed. Call those languages L_0 and L_1 respectively. We claim that if we show that L_0 is not regular, then it implies that L_1 is not regular too, and this would show that to cover both interpretations, it is enough to show that L_0 is not regular.

Claim 1.1

L_1 is regular $\implies$ L_0 is regular.

Proof. If L_1 is regular, then it has a regular expression R corresponding to it. The language of the regular expression 0^*R is precisely L_0, from where it follows that L_0 is regular. □

Claim 1.2

L_0 is not regular $\implies$ L_1 is not regular.

Proof. This follows directly from the contrapositive of the previous claim. □

Now we use the pumping lemma to show that L_0 is not regular.

Note that the length of the binary representation of 3^{n^2} (when there are no leading zeroes) is $1+\lfloor n^2 \log_2 3 \rfloor$, and such a representation is unique.

Choose any $p \in \mathbb{N}$. Consider the binary representation r of 3^{p^2} without leading zeros. Note that $|r| = 1 + \lfloor p^2 \log_2 3 \rfloor \geq p$. Choose any x, y, z such that $|xy| \leq p$ and $|y| > 0$ and $r = xyz$. Then consider the string $s = xy^2z$.

Firstly we show that this is a string whose first character is 1.

1. If x is non-empty, then the first character is the same as that of x, i.e., the same as that of $xyz = r$, which is 1 by definition of r.
2. If x is empty, then $r = yz$ and $s = y^2z$. Now since y is non-empty, the first character of r and s should be the same, and they equal 1.

To show that L_0 is not regular, it suffices to show that $xy^2z \notin L_0$. Suppose, for the sake of contradiction, that this string is in L_0.

Note that the length of xy^2z equals $|y| + |xyz| = |y| + 1 + \lfloor p^2 \log_2 3 \rfloor$. Since $xy^2z \in L$, xy^2z must be the binary representation of 3^{q^2} without leading zeroes, for some $q \in \mathbb{N}$.

Note that if $q \leq p$, then $|y| + 1 + \lfloor p^2 \log_2 3 \rfloor = 1 + \lfloor q^2 \log_2 3 \rfloor \leq 1 + \lfloor p^2 \log_2 3 \rfloor$, which contradicts $|y| > 0$. Hence we must have $q > p$, and since both are natural numbers, we have $q \geq p + 1$. This gives us the following chain of inequalities.

$$
\begin{aligned}
|xy^2z| &= 1 + \lfloor q^2 \log_2 3 \rfloor \\
&\geq 1 + q^2 \log_2 3 - 1 \\
&= q^2 \log_2 3 \\
&\geq (p+1)^2 \log_2 3 \\
&= p^2 \log_2 3 + 2p \log_2 3 + \log_2 3 \\
&> p^2 \log_2 3 + p + 1 \\
&\geq 1 + p + \lfloor p^2 \log_2 3 \rfloor \\
&= p + |xyz| \\
&\geq |xy| + |xyz| \\
&= |x| + |y| + |xyz| \\
&\geq |y| + |xyz| \\
&= |xy^2z|
\end{aligned}
$$

This implies that $|xy^2z| > |xy^2z|$, which is a contradiction.

Hence our assumption that L_0 is regular was wrong, which completes the proof.

Problem 2

Prove that the language $\{0^m 1^n \mid m \neq n\} \subseteq \{0,1\}^*$ is not regular. As a challenge, construct a clean proof using the pumping lemma only. However, no extra credit will be given for this.

Solution.
To prove L=$\{0^m 1^n \mid m \neq n\}$
If L is regular then it must satisfy the pumping lemma. But, consider the string $s = 0^p 1^{p+p!} \in L$ for any $p \in \mathbb{N}$.

Claim 2.1

It is not possible to partition s into 3 strings x, y, z such that $s = xyz$ and

1. $|y| > 0$, and
2. $|xy| \leq p$, and
3. $xy^i z \in L$ for all $i \in \mathbb{N} \cup \{0\}$

Proof. Let's suppose that there exist x, y, z which satisfy the above property, then according to property 1 and 2, y must be of form $y = 0^a$ where $0 < a \leq p$.

Let $k = \frac{p!}{a} + 1$. As a lies between 1 and p therefore it will divide $p!$, hence $k \in \mathbb{N}$.

Now consider the *pumped* string $w = xy^k z$

$$w = 0^{p-a} \cdot 0^{a*(\frac{p!}{a}+1)} \cdot 1^{p+p!}$$
$$w = 0^{p-a} \cdot 0^{p!+a} \cdot 1^{p+p!}$$
$$w = 0^{p+p!} \cdot 1^{p+p!}$$

But this means $w \notin L$, but this is contradiction to the property 3, where $i = k$, So this means our assumption was wrong and L can not be regular.

$\square$

Problem 3

Construct the minimal DFA $D = (Q, \{0,1\}, \delta, q_0, A)$ that recognizes the language

$$\{x \in \{0,1\}^* \mid x \text{ is the binary representation of a number coprime with } 6\}.$$

Prove its minimality by giving a string $z_{q,q'}$ for each pair of distinct states $q, q' \in Q$ such that exactly one of $\widehat{\delta}(q, z_{q,q'})$ and $\widehat{\delta}(q', z_{q,q'})$ is in A. (Proof of correctness of your automaton is not required.)

Solution. DFA $D = (Q, \{0,1\}, \delta, q_0, A)$ is minimal and recognises the above given language, where

$Q = \{q_0, q_1, q_2, q_4, q_5\}$
$A = \{q_1, q_5\}$
δ described in the below table

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_4	q_5
q_4	q_2	q_0
q_5	q_4	q_5

Strings for each distinct pair q,q' is described in the below table

	q_0	q_1	q_2	q_4	q_5
q_0	-	ϵ	01	1	ϵ
q_1	ϵ	-	ϵ	ϵ	1
q_2	01	ϵ	-	1	ϵ
q_4	1	ϵ	1	-	ϵ
q_5	ϵ	1	ϵ	ϵ	-

Problem 4

Let $L_k \subseteq \{0,1\}^*$ be the language defined as $L_k = \{x \mid |x| \geq k$ and the EXOR of the last k bits of x is $1\}$. Prove that any DFA that recognizes L_k has at least 2^k states. (By the way, observe that L_k is recognized by an NFA with $O(k)$ states.)

Solution.
Let D_k be any DFA recognizing the language L_k, let n denote the number of states in this DFA. Then we know that

$$n \geq |\text{classes}(\sim_{D_k})| \geq |\text{classes}(=_{L_k})|$$

Where $\sim_{D_k}$ and $=_{L_k}$ are the equivalence relations that assume their usual meaning as defined in class, and classes$(\cdot)$ denotes the set of equivalence classes of an equivalence relation. Thus, to show that D_k must have at least 2^k states it is sufficient to show that $=_{L_k}$ has at least 2^k states.

Notation 1

Let $x \in \{0,1\}^*$ be any string such that $|x| \geq \alpha$, then we define $\text{last}_\alpha(x)$ as the string formed by last α characters of x.

Claim 4.1

For any 2 strings $x, y \in \{0,1\}^*$ where $|x|, |y| \geq k$ we have that $x =_{L_k} y$ if and only if $\text{last}_k(x) = \text{last}_k(y)$

Proof. To prove this claim we shall prove from both sides.

Note

We shall make use of some well-known identities involving the xor (xor of all bits in a bitstring) and the $\oplus$ (xor of 2 bits) operator.

$$\text{xor}(x \cdot y) = \text{xor}(x) \oplus \text{xor}(y) \tag{1}$$
$$a \oplus b = c \oplus b \implies a = c \tag{2}$$
$$a \oplus 0 = a \tag{3}$$

Suppose $x =_{L_k} y$, then we will first show that $\text{last}_k(x) = \text{last}_k(y)$. By definition of the equivalence relation $=_{L_k}$ we have that $x =_{L_k} y$ if and only if both $x \cdot a$ and $y \cdot a$ are in L_k or both are not in L_k for all strings $a \in \Sigma^*$. Specifically for our particular language L_k this translates to saying that $\text{xor}(\text{last}_k(x \cdot a)) = \text{xor}(\text{last}_k(y \cdot a))$. Now, in particular we choose strings of form $a = 0^m$ where $m < k$. Then we can write

$$\begin{aligned}
&\text{xor}(\text{last}_k(x \cdot a)) = \text{xor}(\text{last}_k(y \cdot a)) \\
\implies &\text{xor}(\text{last}_{k-m}(x) \cdot \text{last}_m(a)) = \text{xor}(\text{last}_{k-m}(y) \cdot \text{last}_m(a)) \\
\implies &\text{xor}(\text{last}_{k-m}(x)) \oplus \text{xor}(\text{last}_m(a)) = \text{xor}(\text{last}_{k-m}(y)) \oplus \text{xor}(\text{last}_m(a)) \quad \text{By property (1)} \\
\implies &\text{xor}(\text{last}_{k-m}(x)) = \text{xor}(\text{last}_{k-m}(y)) \quad \text{By property(2)}
\end{aligned}$$

Now observe that $\text{xor}(\text{last}_m(x))$ may be written as (m-th from last character in x) $\oplus\, \text{xor}(\text{last}_{m-1}(x))$. This leads to the following conclusions.

$$\begin{aligned}
\text{last}_1(x) &= \text{last}_1(y) && \text{directly from } \text{xor}(\text{last}_1(x)) = \text{xor}(\text{last}_1(y)) \\
\text{last}_2(x) &= \text{last}_2(y) && \text{from previous identity and } \text{xor}(\text{last}_2(x)) = \text{xor}(\text{last}_2(y)) \\
&\vdots \\
\text{last}_k(x) &= \text{last}_k(y) && \text{from previous identity and } \text{xor}(\text{last}_k(x)) = \text{xor}(\text{last}_k(y))
\end{aligned}$$

Hence first part of the claim is complete.

Now, suppose $\text{last}_k(x) = \text{last}_k(y)$. We shall now show that this implies that $x =_{L_k} y$.
To show this we need to show that for all strings $a \in \Sigma^*$, either both $x \cdot a$ and $y \cdot a$ are in L_k

or both are not in L_k. Specifically for our particular language L_k this translates to saying that $\mathrm{xor}(\mathrm{last}_k(x \cdot a)) = \mathrm{xor}(\mathrm{last}_k(y \cdot a))$.

To see this note that $\mathrm{xor}(\mathrm{last}_k(x \cdot a)) = \mathrm{xor}(\mathrm{last}_{\max(0,k-|a|)}(x) \cdot \mathrm{last}_{\min(|a|,k)}(a))$. And because xor is both commutative and associative we have that $\mathrm{xor}(\mathrm{last}_k(x \cdot a)) = \mathrm{xor}(\mathrm{last}_{\max(0,k-|a|)}(x)) \oplus \mathrm{xor}(\mathrm{last}_{\min(|a|,k)}(a))$.

Similarly, of course, we have that $\mathrm{xor}(\mathrm{last}_k(y \cdot a)) = \mathrm{xor}(\mathrm{last}_{\max(0,k-|a|)}(y)) \oplus \mathrm{xor}(\mathrm{last}_{\min(|a|,k)}(a))$. But, as $\mathrm{last}_k(x) = \mathrm{last}_k(y)$ by assumption we therefore have that $\mathrm{last}_k(x \cdot a) = \mathrm{last}_k(y \cdot a)$ and since our choice of a was arbitrary the result holds for all $a \in \Sigma^*$ which completes the proof.

$\square$

The above claim implies that $=_{L_k}$ has exactly 2^k equivalence classes, where each string x is classified into a equivalence class based on precisely the last k bits of the string x. Hence the solution is complete.

Problem 5

We all know that the set of strings over the alphabet $\{a, b\}$ containing an equal number of occurrences of ab and ba is regular. However, what if the alphabet is $\{a, b, c\}$? Prove that the language

$$\{x \in \{a, b, c\}^* \mid x \text{ contains an equal number of occurrences of } ab \text{ and } ba\}$$

is not regular. Here are some hints.

1. Take help of the regular expression $(abc \cup bac)^*$.
2. Use closure under inverse homomorphisms from Homework 1.

Solution. Let L_R be the language corresponding to the regular expression $R = (abc \cup bac)^*$. L_R is, of course, regular because regular expressions only generate regular languages. Now, denote by L the following language (which we need to show is not regular) -

$$L = \{x \in \{a, b, c\}^* \mid x \text{ contains an equal number of occurrences of } ab \text{ and } ba\}$$

Suppose L is regular. Then this implies that the language $L' = L \cap L_R$ must also be regular as regular languages are closed under intersection operation. Therefore, to show that that L is not regular it is sufficient to show that L' is not regular.

Claim 5.1

L' is not a regular language.

Proof. By definition $L' = L \cap L_R$ therefore L' can be written as

$$L' = \{x \in L_R | \quad x \text{ has equal number of occurrences of } ab \text{ and } ba\}$$

$$\implies L' = \{x \in L_R | \quad x \text{ has equal number of occurrences of } abc \text{ and } bac\}$$

If L' is regular then it must satisfy the pumping lemma. But, consider the string $s = (abc)^p(bac)^p \in L'$ for any $p \in \mathbb{N}$.

Claim 5.2

It is not possible to partition s into 3 strings x, y, z such that

1. $|y| > 0$, and
2. $|xy| \leq p$, and
3. $xy^iz \in L'$ for all $i \in \mathbb{N} \cup \{0\}$

Proof. Consider any partition of $s = xyz$ such that conditions (1) and (2) hold. Then xy will be a prefix of s strictly smaller than the prefix $(abc)^p$. Now we make cases on the string y.

1. y is of form $(abc)^k$ for some $k > 0$. This implies x must also be of form $(abc)^{k_1}$ for some choice of $k_1 \geq 0$, and z will be of form $(abc)^{k_2}(bac)^p$ for some $k_2 > 0$. Hence the *pumped down* string xy^0z will be of form $(abc)^{k_1}(abc)^{k_2}(bac)^p$ where $k_1 + k_2 < p$ as $k_1 + k_2 + k = p$ and $k > 0$. Note that this *pumped down* string is in L_R but not in the language L'.
2. y is not of form $(abc)^k$. Even in this case, the *pumped down* string xy^0z will either be not in L_R or it will not be in L' because of removal of atleast 1 character from the string as we have that xy is a prefix smaller than $(abc)^p$.

Hence, in either case we have shown that the (3) condition can never hold for all $i \in \mathbb{N}$. $\square$

Since L' does not satisfy the pumping lemma which is the necessary condition for a language to be regular, we have that L' can not be regular. Hence, the proof is complete. $\square$

Since L' is not regular, L can not be regular as well, by contradiction. Hence the solution is complete.

Problem 6

Prove that for any infinite regular language L, there exist two infinite regular languages L_1, L_2 such that $L = L_1 \cup L_2$ and $L_1 \cap L_2 = \emptyset$. Here are some hints.

1. Let D be any DFA and q be any one of its states. Prove informally that the language

$$L_q = \{x \mid x \in \mathcal{L}(D) \text{ and the run of } D \text{ on } x \text{ visits } q \text{ an odd number of times}\}$$

   is regular.

2. Recall the proof of the pumping lemma.

Solution. We proceed in two parts.

Claim 6.1

If L is language with a DFA D, and q is a state of D, then the language

$$L_q = \{x \mid x \in \mathcal{L}(D) \text{ and the run of } D \text{ on } x \text{ visits } q \text{ an odd number of times}\}$$

is regular.

Proof. Consider the DFA $D' = (Q', \Sigma, \delta', q_0', A')$ as follows.

1. $Q' = Q \times \{0, 1\}$
2. $\delta'((p,t),a) = \begin{cases} (\delta(p,a), t) & \text{if } \delta(p,a) \neq q \\ (\delta(p,a), 1-t) & \text{otherwise} \end{cases}$
3. $q_0' = \begin{cases} (q_0, 0) & \text{if } q_0 \neq q \\ (q_0, 1) & \text{otherwise} \end{cases}$
4. $A' = \{(q_a, 1) \mid q_a \in A\}$

Claim 6.2

If $\hat{\delta}'(q_0', x) = (p, b)$, then the following are true (and vice-versa):

1. $\hat{\delta}(q_0, x) = p$
2. b is the parity of the number of visits of q in the run of D on x.

Proof. We prove this by induction on the length of x. The base case is trivial: for $|x| = 0$, we must have $x = \epsilon$, so $\hat{\delta}'(q_0', x) = \hat{\delta}'(q_0', \epsilon) = q_0'$, and the property holds by definition.

For the inductive case, suppose $x = ya$ for some $y \in \Sigma^*$ and $a \in \Sigma$. Suppose $\hat{\delta}'(q_0', y) = (p', b')$ for some $p' \in Q, b' \in \{0, 1\}$. Then we have two cases:

1. $\delta(p', a) = q$.

   In this case, the parity of number of visits to q on the run of D on x is different from that for the run of D on y. Also, from the transition function, we note that $b = 1 - b'$, from which the second part of the claim follows. Again, from the transition function, we have $p = \delta(p', a) = \delta(\hat{\delta}(q_0, y), a) = \hat{\delta}(q_0, ya) = \hat{\delta}(q_0, x)$, so the first part of the claim holds true here too.

2. $\delta(p', a) \neq q$.

   In this case, the second part of the claim can be proved as in the previous case. For the first part of the claim, since $\delta(p', a) \neq q$, the parity of the number of visits to q on the run of D on x is the same as that for the run of D on y, and the parity doesn't change. Since the transition function gives $b = b'$, the claim is proved.

The reverse direction can be proved by simply reversing the argument. □

Claim 6.3

$\mathcal{L}(D') = L_q$.

Proof. Consider the following equivalences induced by the previous lemma:

$$\begin{aligned} x \in \mathcal{L}(D') &\iff \hat{\delta}'(q_0', x) \in A' \\ &\iff \hat{\delta}'(q_0', x) = (q_a, 1) \in A' \\ &\iff \exists q_a \in A\text{: the run of } D \text{ on } x \text{ has an odd number of visits to } q \wedge \hat{\delta}'(q_0, x) = q_a \\ &\iff x \in L(D) \text{ and the run of } D \text{ on } x \text{ visits } q \text{ an odd number of times} \\ &\iff x \in L_q \end{aligned}$$

This completes the proof of this claim. □

Since we have exhibited a DFA recognizing this language, this language must be regular. □

Claim 6.4

There exists a state q in Q such that there are infinitely many strings in L_q and $L \setminus L_q$.

Proof. Consider a string s with length at least $|Q|$ (Such a string s exists because L is infinite). Then in the run of D on x, there are $|Q|+1$ states, and hence two of them must be the same, say q. Suppose the substring of x corresponding to the simple cycle between the first two occurences is y, with $x = wyz$ for some $w, z \in \Sigma^*$. Note that $|y| > 0$. We consider the following two sets of strings:

1. $L_e = \{wy^{2i}z \mid i \in \mathbb{N}\}$
2. $L_o = \{wy^{2i+1}z \mid i \in \mathbb{N}\}$

Clearly, both of these languages have an infinite number of strings, and the following hold true (since pumping the cycle once increases the number of visits to q by exactly 1):

1. For any string s in L_e, the parity of the number of visits of the run of D on s is the same as that of the run of D on x.
2. For any string s in L_o, the parity of the number of visits of the run of D on s is different from that of the run of D on x.

Hence, one of these is a subset of L_q, and the other one of $L \setminus L_q$, which implies that both are infinite. □

By the definition of the complement, the languages L_q and $L \setminus L_q$ are disjoint and have union equal to L, which completes the proof in view of the previous claim, and the fact that $L \setminus L_q$ is a regular language due to being the set difference of two regular languages (closure under intersection and complementation), or just by noting that we can replace 1 by 0 in the set of accepting states in the definition of the DFA D'.