Problem 4.2.6 by machine construction

a) Let $L$ be a regular language, and $M_L$ a DFA for $L$. We will construct a DFA $M'$ for $\text{min}(L)$.

Observe that if $w \in \text{min}(L)$, then $M_L$ on input $w$ ends on some final state. Furthermore, no previous state along the path taken by $w$ through $M_L$ is allowed to be a final state. So to construct $M'$, we simply take $M_L$, and for each final state, we remove all outgoing transitions (including self-loops), and replace them with a transition to a “trap” state for each possible input symbol. This gives us the desired $M'$.

b) Let $L$ be a regular language, and $M_L$ a DFA for $L$. We will construct a DFA $M'$ for $\text{max}(L)$.

Observe that if $w \in \text{max}(L)$, then $M_L$ on input $w$ ends on some final state. Furthermore, from this state, no final state is reachable via one or more transitions. So to construct $M'$, we apply the following algorithm to $M_L$: Let $M' = M_L$ initially. For each final state $q$ in $M_L$, we find the set of all states reachable from $q$ via one or more transitions, using depth-first search/breadth-first search/your favorite search algorithm. If there are any final states in this set (including $q$ itself), we make $q$ non-final in $M'$.

c) Similar to part (b), we can construct $M'$ as follows: Let $M' = M_L$ initially. For every non-final state $q$ in $M_L$, we find the set of all states reachable from $q$ via one or more transitions. If there are any final states in this set, we make $q$ final in $M'$.

Alternatively, we can also use a “reverse marking” algorithm: Starting with all of the final states in $M_L$, we look at the set of all states that can go to one of these final states on one transition. If any of them are non-final, we mark them as final and repeat the process. When no more new states are marked as final in an iteration, we can stop. The resulting DFA is the $M'$ we want.
1. **Problem 2.** Suppose the languages $L_1$ and $L_2$ belong to $\mathcal{L}$. We will show that $L_1 \cup L_2$ belongs to $\mathcal{L}$. First, because $\mathcal{L}$ is closed under homomorphisms, we can easily assume that $L_1$ and $L_2$ are under different alphabets, say $\Sigma$ and $\Delta$. Now, since $\mathcal{L}$ is closed under concatenation, if we define $L_3 = L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$, then $L_3 \in \mathcal{L}$. Also, if $h : (\Sigma \cup \Sigma) \cup (\Delta \cup \Delta) \to \Sigma \cup \Delta$, be defined as $h(a) = h(a) = a$, for all $a$, then $h^{-1}(L_3)$ is also in $\mathcal{L}$. What this does is for each string $w \in L_3$, in order to get $h^{-1}(w)$, we replace each symbol $a$ nondeterministically by either $a$ or $\bar{a}$. Now, in order to separate out the parts corresponding to $L_1$ and $L_2$, we define $L_4 = h^{-1}(L_3) \cap (\Sigma^* \Delta^* + \Sigma^* \Delta^*)$. Note that, what we have done is push the union operation down to the regular languages. Finally, define $L_5 = g(L_4)$ where $g(a) = \varepsilon$ and $g(a) = a$ for all $a \in \Sigma \cup \Delta$. Then, it is easy to check that $g(L_4) = L_1 \cup L_2$.

2. **Problem 3. A:** This statement is false. The counterexample is $\text{binaryA} = 10^*$. This language is obviously regular. But the $\text{unaryA}$ corresponding to this is $L = \{0^k \mid \exists m : k = 2^m\}$, which is not regular. The proof that this is not regular can either be done using the pumping lemma, or using a previous problem from the homework 3. Suppose, the DFA accepting the above language has $k$ states. Now, we can exhibit a set of $k+1$ prefixes $x_1 = 0^i$ such that for each pair $x_i$ and $x_j$ there is a string $z$ such that only one of $x_i$ or $x_j$ belongs to the language $L$. Take $x_i = 0^{2^k-i}$ for $i = 1, \ldots, k+1$. It is clear that for $z = 0^r$, the string $x_iz$ belongs to the language, but the string $x_jz$ does not, for any $j \neq i$.

**B:** This statement is true. Given a machine accepting the language $\text{unaryA}$, we construct a machine for the language $\text{binaryA}$.

**Claim 1** Let $M = (Q, \Sigma, \delta, s, F)$ be the DFA for accepting $\text{unaryA}$. Then we can write $Q$ and $\delta$ as follows:

- $Q = \{q_1, \ldots, q_n\} \cup \{r_0, \ldots, r_m\}$ with
- $\delta(q_i, 0) = q_{i+1}$ for $i \in \{0, \ldots, n-1\}$ and $\delta(q_n, 0) = r_0$ and $\delta(r_j, 0) = r_{(j+1) \mod m}$

**Proof.** the proof of this follows from the fact that $M$ is a DFA and that the language is over an unary alphabet and has to be ultimately periodic. Hence, if we choose $n$ to be the constant after which the length of strings are periodic, then we have our proof.

After this claim, we define our new machine for accepting $\text{binaryA}$ based on $M$, as follows. Define $M' = (Q, \Sigma, \delta', s', F')$, where $\Sigma = \{0, 1\}$, $s' = s$, $F' = F$, and $\delta'$ is defined as follows.
Similarly, \( \delta'(q_i, 1) = q_{2i+1} \) if \( 2i + 1 \leq n \), else \( \delta(q_i, 1) = r_{(2i+1-1) \mod m} \).

- Similarly we can show the other case when \( \delta'(r_j, 0) = r_{(2j) \mod m} \)
- And, \( \delta'(r_j, 1) = r_{(2j+1) \mod m} \)

As usual, the following claim proves the correctness of our solution.

**Claim 2** If \( w \) is the binary representation of number \( i \), then \( \hat{\delta}'(s, w) = \hat{\delta}(s, 0^i) \).

**Proof.** By induction on the length of \( w \). The base case occurs when \( i = 0 \) and \( w = \epsilon \).

Assume then the claim is proved for \( |w| \leq k \). Let \( w' = w0 \). Then the unary string corresponding to \( w' \) is \( 0^{2i} \). Hence, \( \delta'(s, w') = \delta'(\hat{\delta}'(s, w), 0) \). Also, by inductive hypothesis, \( \delta'(s, w') = \delta(s, 0^i) \).

Now we consider two cases.

(a) \( \hat{\delta}'(s, w) = q_i \). If \( 2i \leq n \), we have \( \hat{\delta}'(s, w') = q_{2i} = \hat{\delta}(s, 0^{2i}) \). Else, \( \hat{\delta}'(s, w') = r_{(2i-1) \mod m} \).

Again, \( \delta(s, 0^{2i}) = r_{(2i-1) \mod m} \). Hence, the two are equal in this case.

(b) Similarly we can show the other case when \( \hat{\delta}'(s, w) = r_j \) for some appropriate \( j \). Hence the whole proof.

So, the machine \( M' \) does accept the language \textbf{binaryA}, since the start and accepting states remain the same.

3. **Problem 4.** We transform each of the given languages into \( L_1 \) or \( L_2 \) using transformations that preserve regularity. Then, since neither \( L_1 \) nor \( L_2 \) are regular, we will have shown that the given languages are not regular either.

- \( L = \{0^{2i}1^{3i} \mid i \geq 0\} \). Then \( L_1 = h^{-1}(L) \), where \( h \) is defined as \( h(0) = 00 \) and \( h(1) = 111 \). Note that it is not enough to say that \( h(L_1) = L \). We cannot then claim that \( L \) is non-regular, because, there are homomorphisms that transform non-regular languages to regular ones, this is a trivial one \( h(0) = h(1) = \epsilon \).
- \( L = \{ww \mid w \in \{0, 1\}^*\} \). Define \( h \) such that \( h(0) = h(\bar{0}) = 0 \) and \( h(1) = h(\bar{1}) = 1 \). Then, \( L_4 = h^{-1}(L_3) \cap 0^i1^i = \{0^i1^i \mid i \geq 0\} \). Finally, if \( g \) is defined such that \( g(0) = 0 \), \( g(1) = g(\bar{1}) = \epsilon \) and \( g(0) = 1 \), then \( g(L_4) = L_1 \).
- \( L = \{0^i1^j \mid i = j + 50\} \). Here, let \( h(0) = h(\bar{0}) = 0 \), and \( h(1) = 1 \). So, \( L_3 = h^{-1}(L) \cap 0^i\bar{0}501^i = \{0^i\bar{0}501^i \mid i \geq 0\} \). Next, define \( g(0) = 0 \), \( g(1) = 1 \) and \( g(\bar{0}) = \epsilon \). Then, \( L_1 = g(L_3) \).
- \( L = \{0^i1^j2^k \mid i = j \lor j = k\} \). Then, \( (L \cap 0^+1^+) \cup \{\epsilon\} = L_1 \).