Problem 2.

(a) Basic idea: The machine nondeterministically guesses (when reading an input symbol 0) and raises a substring of \(0((0+1)^2)^*0\) that is forthcoming. \(0,1\)

\[
M: \quad \text{start} \rightarrow q_{\text{start}} \rightarrow o \rightarrow q_1 \rightarrow q_{\text{good}} \rightarrow q_{\text{even}} \rightarrow q_2 \rightarrow \varepsilon
\]

- \(q_{\text{start}}\): nondeterministically wait or guess on an input symbol 0
- \(q_1, q_{\text{good}}, q_{\text{even}}, q_2\): having encountered an input symbol 0, verify if a substring of the form \(0((0+1)^2)^*0\) appears.

Can verify that \(\forall x \in \{0,1\}^*\), \(M\) accepts \(x\) if \(x \in 0(0+1)*((0+1)^2)^+0(0+1)^*\).

(b) The given language is the disjoint union of the two languages:

\[
L_a = \{ x \in \{a,b,c\}^* \mid \#(a) \geq 3 \text{ and } 0 \leq \#(b), \#(c) \leq 2 \}
\]

\[
L_b = \{ x \in \{a,b,c\}^* \mid \#(a) \geq 3 \text{ and } 0 \leq \#(b), \#(c) \leq 2 \}
\]

Basic idea for constructing a DFA \(M\) accepting \(L_a\): each state has 3 components to record \(\#(a), \#(b), \#(c)\) in the input consumed so far.

\[
\Omega = \{ (i,j,k) \in \mathbb{N}^3 \mid i \geq 3, j \geq 2, \text{ and } k \leq 2 \}
\]

Start state: \((0,0,0)\)

Set of accepting states: \(\{(3,j_1,k) \mid 0 \leq j, k \leq 2\}\)
1-step transition function \( s : Q \times \{a, b, c\} \to Q \) is defined as:

\[
\begin{align*}
 s((i, j, k), a) &= \begin{cases} (i+1, j, k) & \text{if } i \leq 2 \\ (i, j, k) & \text{if } i = 3 \end{cases} \\
 s((i, j, k), b) &= \begin{cases} (i, j+1, k) & \text{if } j \leq 1 \\ \text{exceed} & \text{if } j = 2 \end{cases} \\
 s((i, j, k), c) &= \begin{cases} (i, j, k+1) & \text{if } k \leq 1 \\ \text{exceed} & \text{if } k = 2 \end{cases}
\end{align*}
\]

\( \forall \langle q_0, i, j, k \rangle, s(q_0, a, b, c) = q_{\text{exceed}} \)

A DFA \( M_b \) accepting \( L_b \) is similar.

A desired FA accepting \( L_a \cup L_b \) is:

\[
\begin{align*}
\text{start} &\rightarrow (q) \xrightarrow{\varepsilon} M_a \\
&\xrightarrow{\varepsilon} M_b
\end{align*}
\]

(c) Given that an FA \( M \) accepting \( L \) (without loss of generality, we may assume that \( M \) has one accepting state \( q_{\text{accept}} \)), we construct an FA \( M' \) accepting half \( (L) \).

The basic idea is that \( M' \) keeps track of two states in \( M \) (using two coordinates/tracks in a state of \( M' \)): 
"forward simulation" (For each input symbol read in \( M' \), \( M' \) uses first coordinate/track to simulate \( M \) on that symbol. At the same time, \( M' \) simulates the backward simulation starting at \( \text{accept} \) in \( M \).) Simultaneously, \( M' \) uses second coordinate/track to simulate \( M \) backwards on a guessed symbol.

\( M' \) accepts an input \( x \) if \( \text{forward simulation} \) (on \( x \)) and \( \text{the backward simulation} \) (on a guessed \( y \), \( |y| = |x| \)) are in a common state of \( M \).

Formally, assume that NFA \( M = (Q, \Sigma, \delta, q_0, \text{Accept}) \) accepts \( L \).

Construct an NFA \( M' = (Q', \Sigma, \delta', q'_0, F') \) as follows: \( Q' = Q \times Q \), \( q'_0 = (q_0, \text{Accept}) \), \( F' = \{(q, q') \mid q \in Q \} \), and \( \delta' : Q' \times \Sigma \rightarrow 2^Q \) is defined as:

\[
\forall (p, q) \in Q' \quad \forall a \in \Sigma \\
\delta'(p, q), a = \begin{cases} \delta(p, a) & \text{forward simulation} \\
\delta(p, a) \cap \{ b \in \Sigma \mid b \in Q \} & \text{guessed symbol} \\
\delta(p, a) \cap \{ b \in \Sigma \mid b \in \text{accept} \} & \text{backward simulation} 
\end{cases}
\]
Problem 3. (Similar to Homework 1, problem 9)

\[ L = \{ x \in \{0,1\}^* \mid x^r = x^3 \} \]

We show that there does not exist any DFA accepting \( L \).

Suppose the contrary that \( L = L(M) \) for some DFA \( M = (Q, \Sigma, \delta, q_0, F) \), where \( Q = \{ q_1, q_2, \ldots, q_n \} \) for some positive integer \( n \).

Consider the sequence of strings:

\[ \gamma_1 = 0^n \]
\[ \gamma_2 = 0^{n+1} \]
\[ \vdots \]
\[ \gamma_n = 0^n \]
\[ \gamma_{n+1} = 0^{n+1} \]

By Pigeonhole Principle, there exist \( i \neq j \) such that \( \gamma_i = \gamma_j \).

The two inputs \( \gamma_i \) and \( \gamma_j \) cause two identical versions of \( M \), starting from \( q_1 \), to

The same state, say \( p \in Q \).

![Diagram](attachment:image.png)

Now, consider suffixing \( 10^n \) to augment the two input strings \( \gamma_i \) and \( \gamma_j \) to \( \gamma_i 10^n \) and \( \gamma_j 10^n \), respectively.

Notice that:

The augmentation \( 10^n \) causes the two versions of \( M \)

To a common state, say \( p'' \), respectively. (Why?)

But...
The input string $0^a10^b$ is a palindrome ($= L$), so $M$ should accept $0^a10^b$, i.e., $p' \in F$.

But, the input string $0^a10^b$ ($a \neq b$) is not a palindrome ($\neq L$), so $M$ should reject $0^a10^b$, i.e., $p' \notin F$, a contradiction!
Problem 6

Basic idea of constructing a DFA $N$ is that it essentially mimics the behavior of $M$, but in addition, $N$ keeps track of a bit that indicates if the state $r$ has been visited.

The bit starts out as 0, and is flipped to 1 in the event that $r$ is reached. The bit is never flipped back once it turns to 1.

The accepting states of $N$ are of the form $(1,q)$ where $q \in F$ as they indicate that $M$ is in an accepting state $(q,F)$ and the state $r$ has been visited.

$N = \left( \{0,1\} \times Q, \Sigma, \delta, (0,q_0), \{(1,q) | q \in F\} \right)$

where

$\delta : \left( \{0,1\} \times Q \right) \times \Sigma \rightarrow \left( \{0,1\} \times Q \right) \cup$

defined as:

$\delta'(0,q,a) = \begin{cases} (0, \delta(q,a)) & \text{if } \delta(q,a) \neq r \\ (1, \delta(q,a)) & \text{if } \delta(q,a) = r \end{cases}$

and

$\delta'((1,q),a) = (1, \delta(q,a))$.
Problem 6

Use the same construction given in the proof of Theorem 1.39, which shows the equivalence of NFAs and DFAs. We need only change $F'$, the set of accept states of the new DFA. Here we let $F' = \mathcal{P}(F)$. The change means that the new DFA accepts only when all of the possible states of the all-NFA are accepting.