CSC236 Homework Assignment #3

Language Regularity, Regular Expressions, and DFAs/NFAs

Alexander He Meng

Prepared for November 25, 2024

Question #1

Let $\Sigma = \{0, 1\}$.

(a):

Claim: Σ^* is a regular language.

Proof.

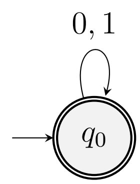
Let $L_1 = \{0\}$ and $L_2 = \{1\}$ be regular languages of Σ.
Define $L_3 = L_1 \cup L_2 = \{0, 1\}$ as the regular language obtained by the union of L_1 and L_2.

By definition, ${L_3}^*$ is a regular language. Since $L_3 = \Sigma$, Σ^* is also a regular language.

While this proof now is complete, the assignment encourages the use of determinstic finite automaton (or DFA) proofs to show that languages are regular.

So, as an alternative to *Part (a)* (this part's proof), and for all relevant subsequent parts, adopt a DFA proof.

Consider the **transition function** δ associated with the following DFA:



Old State	Symbol	New State
q_0	0	q_0
q_0	1	q_0

Table 1: State Transition Table

Using δ, define the DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$, where

$$Q = \{q_0\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : Q \times \Sigma \to Q \text{ is the transition function defined by } \textit{Table 1}$$

$$s = q_0 \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_0\} \subseteq Q \text{ is the set of accepting states of } \mathcal{D}.$$

For the sole state q_0 in $\mathcal{D}$, define its **invariant** for strings $x \in \Sigma^* = \{0, 1\}^*$:

$$P_{q_0}(x) : x \text{ is } \epsilon \text{ or consists of 0s and 1s.}$$

As the only state, q_0 is trivially **mutually exclusive**. As well, every string in $\Sigma^* = \{0, 1\}^*$ is either ϵ or consists of some number of 0s and 1s, which q_0 clearly satisfies; **exhaustivity** is satisfied.

Let $q \in Q = \{q_0\}$ and $x \in \Sigma^* = \{0, 1\}^*$ both be arbitrary (here, $q = q_0$ always).
Denote the predicate:

$$P_{\delta(q_0, x)}(x) \coloneqq P_q(x) \text{ is the state invariant.}$$

Perform structural induction on $P_{\delta(q_0,x)}(x)$ for all strings $x \in \Sigma^* = \{0, 1\}^*$, as follows:

<u>Base Case:</u>
Let $q = q_0$ and $w = \epsilon$.
$P_q(w) = P_{q_0}(\epsilon)$ is true as $w = \epsilon$.

<u>Induction Hypothesis:</u>
Assume that $P_q(w)$ is true for all $q \in \{q_0\}$ and some $w \in \{0, 1\}^*$.
This means w is either the empty string or consists of 0s and 1s.

<u>Induction Step:</u>
By the Induction Hypothesis, $P_q(w)$ is true for arbitrary $q \in \{q_0\}$ and some $w \in \{0, 1\}^*$.

Demonstrate that the invariant of q_0 holds when processing all possible strings from Σ^*.

This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0,1\}$.

Let $w \in \{0,1\}^*$ be arbitrary.

Let $q = q_0$, $z \in \{0,1\}$, and assume $P_{q_0}(w)$ is true.
Consider that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,z)}(wz) = P_{q_0}(wz)$. Since w is either the empty string or consists of 0s and 1s, while $z \in \{0,1\}$, wz remains to consist of 0s and 1s. This matches the definition of $P_{q_0}(wz)$. Thus, $P_{\delta(q,z)}(wz)$ holds.

There are no other state invariants in $\mathcal{D}$. By the principle of structural induction, $P_{\delta(q,x)}$ is true for all $q \in Q = \{q_0\}$ and $x \in \Sigma^* = \{0,1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language $L = \Sigma^*$ over $\Sigma = \{0,1\}$.
This is achievable by showing that if $x \in \Sigma^* = \{0,1\}^*$ is arbitrary, x is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(x)$ holds.

Recall the sole state invariant, $P_{q_0}(x) : x$ is ϵ or consists of 0s and 1s.
If $x \in L$, then x is either the empty string or consists of 0s and 1s. Clearly, $P_{q_0}(x)$ is true.
On the other hand, choose the accepting state $q_0 \in F$ and assume $P_{q_0}(x)$ is true. Clearly, $x \in L$, due to matching definitions.

Therefore, $P_{\delta(q_0,x)}(x)$ is true for all strings $x \in \Sigma^* = \{0,1\}^*$.

While this DFA proof is redundant in nature, it is clear that $\mathcal{D}$ accepts $L = \Sigma^*$ over $\Sigma = \{0,1\}$. Again, it has been demonstrated that L is a regular language.

□

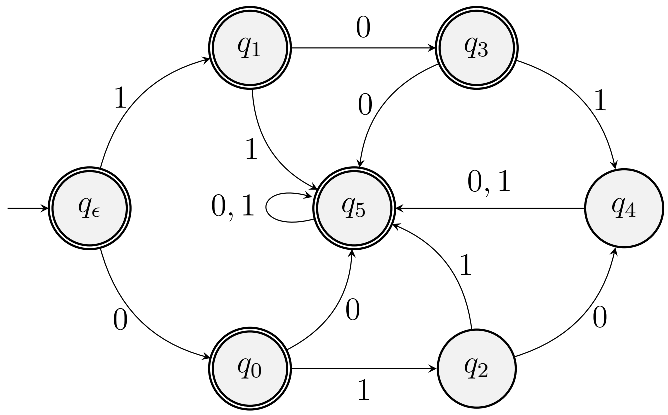
(b):

<u>Claim:</u> $\Sigma^* \setminus K, K = \{01, 101, 010\}$ is a regular language.

Proof.
This proof aims to show that the language $\Sigma^* \setminus K$ is a regular language, by constructing

a DFA (with proof) that accepts all strings over $\Sigma^* = \{0,1\}^*$ except the literal strings in $K = \{01, 101, 010\}$.

Consider the **transition function** δ associated with the following DFA:



Old State	**Symbol**	**New State**
q_ϵ	0	q_0
q_ϵ	1	q_1
q_0	0	q_5
q_0	1	q_2
q_1	0	q_3
q_1	1	q_5
q_2	0	q_4
q_2	1	q_5
q_3	0	q_5
q_3	1	q_4
q_4	0	q_5
q_4	1	q_5
q_5	0	q_5
q_5	1	q_5

Table 2: State Transition Table

Using δ, define the DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$, where

$$Q = \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : Q \times \Sigma \to Q \text{ is the transition function define by } \textit{Table 2}$$

$$s = q_\epsilon \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_\epsilon, q_0, q_1, q_3, q_5\} \subseteq Q \text{ is the set of accepting states of } \mathcal{D}.$$

For each state $q_i \mid_{i \in \{\epsilon,0,1,2,3,4,5\}}$ in $\mathcal{D}$, define a **state invariant** $P_q(x)$ for strings $x \in \Sigma^* = \{0,1\}^*$:

$P_{q_\epsilon}(x) : x$ is the empty string, ϵ

$P_{q_0}(x) : x$ is 0

$P_{q_1}(x) : x$ is 1

$P_{q_2}(x) : x$ is 01

$P_{q_3}(x) : x$ is 10

$P_{q_4}(x) : x$ is either 101 or 010

$P_{q_5}(x) : x$ consists of 0s and 1s but $x \notin \{0, 1, 10, 01, 010, 101\}$

It is clear that q_i for $i \in \{\epsilon, 0, 1, 2, 3, 4\}$ are mutually exclusive by their unique definitions. Moreover, the q_5 state is the direct complement of the **union of the previous** q_i **states** (as a non-initial state, q_5 shall never process ϵ), so q_5 must be mutually exclusive as well. Thus, all states in $\mathcal{D}$ are **mutually exclusive**.

For exhaustivity, notice that every string in $\Sigma^* = \{0,1\}^*$ is either ϵ or consists of some number of 0s and 1s. The q_ϵ state accounts for the empty string case, q_i for $i \in \{0,1,2,3,4\}$ account for specific strings of 0s and 1s, and q_5 accounts for all the remaining cases of 0s and 1s not covered by the q_i. Thus, the states in $\mathcal{D}$ are **exhaustive**.

Let $q \in Q = \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\}$ and $w \in \Sigma^* = \{0,1\}^*$ both be arbitrary.
Denote the predicate:

$$P_{\delta(q_0,w)}(w) \coloneqq P_q(w) \text{ is the state invariant.}$$

Perform structural induction as follows:

<u>Base Cases:</u>
Let $w = \epsilon$.

Let $q = q_\epsilon$. $P_q(w) = P_{q_\epsilon}(\epsilon)$ is true as $w = \epsilon$, matching the definition of $P_{q_\epsilon}(w)$.
Let $q \neq q_\epsilon$. $P_q(w) = P_{q_i}(\epsilon) \mid_{i \in \{0,1,2,3,4,5\}}$ are vacuously true as these corresponding states are non-initial and do not process ϵ.

<u>Induction Hypothesis:</u>
Assume that $P_q(w)$ is true for all $q \in \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\}$ and some $w \in \{0,1\}^*$.

<u>Induction Step:</u>
By the Induction Hypothesis, $P_q(w)$ is true for arbitrary $q \in \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\}$ and some $w \in \{0,1\}^*$.

Demonstrate that all state invariants hold when processing strings from $\Sigma = \{0,1\}$.
This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0,1\}$.

Let $w \in \{0,1\}^*$ be arbitrary. Then, consider the following cases.

<u>Case $(q = q_\epsilon, z = 0)$:</u>
Assume $P_{q_\epsilon}(w)$ is true. Then, $w = \epsilon$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_\epsilon,0)}(\epsilon 0) = P_{q_0}(0)$, which is clearly true by definition.

<u>Case $(q = q_\epsilon, z = 1)$:</u>
Likewise, assume $P_{q_\epsilon}(w)$ is true. Then, $w = \epsilon$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_\epsilon,1)}(\epsilon 1) = P_{q_1}(1)$, which is also clearly true by definition.

<u>Case $(q = q_0, z = 0)$:</u>
Assume $P_{q_0}(w)$ is true. Then, $w = 0$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,0)}(00) = P_{q_5}(00)$, which is true as $w = 00$ indeed consists of 0s and 1s and $w \notin \{0, 1, 10, 01, 010, 101\}$.

<u>Case $(q = q_0, z = 1)$:</u>
Likewise, assume $P_{q_0}(w)$ is true. Then, $w = 0$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,1)}(01) = P_{q_2}(01)$, which is clearly true by definition.

Case ($q = q_1, z = 0$):
Assume $P_{q_1}(w)$ is true. Then, $w = 1$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_1,0)}(10) = P_{q_3}(10)$, which is clearly true by definition.

Case ($q = q_1, z = 1$):
Likewise, assume $P_{q_1}(w)$ is true. Then, $w = 1$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_1,1)}(11) = P_{q_5}(11)$, which is true as $w = 11$ indeed consists of 0s and $1s$ and $w \notin \{0, 1, 10, 01, 010, 101\}$.

Case ($q = q_2, z = 0$):
Assume $P_{q_2}(w)$ is true. Then, $w = 01$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_2,0)}(010) = P_{q_4}(010)$, which is true as $w = 010$ is indeed either 101 or 010.

Case ($q = q_2, z = 1$):
Likewise, assume $P_{q_2}(w)$ is true. Then, $w = 01$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_2,1)}(011) = P_{q_5}(011)$, which is true as $w = 011$ indeed consists of 0s and $1s$ and $w \notin \{0, 1, 10, 01, 010, 101\}$.

Case ($q = q_3, z = 0$):
Assume $P_{q_3}(w)$ is true. Then, $w = 10$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_3,0)}(100) = P_{q_5}(100)$, which is true as $w = 100$ indeed consists of 0s and $1s$ and $w \notin \{0, 1, 10, 01, 010, 101\}$.

Case ($q = q_3, z = 1$):
Assume $P_{q_3}(w)$ is true. Then, $w = 10$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_3,1)}(101) = P_{q_4}(101)$, which is true as $w = 101$ is indeed either 101 or 010.

Case ($q = q_4, z \in \{0, 1\}$):
Assume $P_{q_4}(w)$ is true. Then, $w \in \{101, 010\}$.

It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_4,z)}(wz) = P_{q_5}(wz)$. Notice that wz is a string with 4 symbols, because w and z are strings with 3 symbols and 1 symbol, respectively. As well, wz is constructed using only 0s and 1s. Thus, wz consists of 0s and 1s and $w \notin \{0, 1, 10, 01, 010, 101\}$; $P_{\delta(q,z)}(wz)$ holds.

Case ($q = q_5, z \in \{0, 1\}$):
Assume $P_{q_5}(w)$ is true.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_5,z)}(wz) = P_{q_5}(wz)$. Recall that $z \in \{0, 1\}$.

Since w consists of 0s and 1s and $w \notin \{0, 1, 10, 01, 010, 101\}$ by $P_{q_5}(w)$, concatenating z (which is either 0 or 1) to w does not result in $wz \in \{0, 1, 10, 01, 010, 101\}$. This is because $w \neq \epsilon$ (as q_5 does not process ϵ), so $wz \notin \{0, 1\}$, which implies $wz \notin \{10, 01\}$ (as q_5 will not process 10 and 01 if does not process 1 and 0, respectively), which also implies $wz \notin \{010, 101\}$ (as q_5 will not process 010 and 101 if it does not process 01 and 10, respectively).

End of Cases:
Hence, it is demonstrated that if $P_q(w)$ is true for all $q \in \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\}$ and some $w \in \{0, 1\}^*$, then $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0, 1\}$. By the principle of structural induction, $P_{\delta(q,x)}$ is true for all $q \in Q = \{q_\epsilon, q_0, q_1, q_2, q_3, q_4, q_5\}$ and $x \in \Sigma^* = \{0, 1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language $L = \Sigma^* \setminus \{01, 101, 010\}$ over $\Sigma = \{0, 1\}$.
This is achievable by showing that if $x \in \Sigma^* = \{0, 1\}^*$ is arbitrary, x is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(x)$ holds.

Recall the invariants of the accepting states, $q_i \in F \mid_{i\in\{\epsilon,0,1,3,5\}}$:

$$P_{q_\epsilon}(x) : x \text{ is the empty string, } \epsilon$$
$$P_{q_0}(x) : x \text{ is } 0$$
$$P_{q_1}(x) : x \text{ is } 1$$
$$P_{q_3}(x) : x \text{ is } 10$$
$$P_{q_5}(x) : x \text{ consists of 0s and 1s but } x \notin \{0, 1, 10, 01, 010, 101\}$$

Implication

If an arbitrary string $x \in L = \Sigma^* \setminus \{01, 101, 010\}$, then x is either an empty string or consists of 0s and 1s, but must not be either of 01, 101, nor 010. Choose q_ϵ to cover the empty string case, and q_5 to cover all cases **except** $x \in \{0, 1, 10, 01, 010, 101\}$. Since x must not be a member of $\{01, 010, 101\}$, it remains to account for $x \in \{0, 1, 10\}$. Conveniently, these cases are covered by q_0, q_1, and q_3. Thus, no matter which string $x \in L = \Sigma^* \setminus \{01, 101, 010\}$, there exists accepting states $q \in F$ such that $P_q(x)$ holds.

Implied-by

Conversely, $L = \Sigma^* \setminus \{01, 101, 010\}$ can be constructed by the same choices of accepting states. Choose from the accepting states $q_i \in F \mid_{i\in\{\epsilon,0,1,3,5\}}$ and assume $P_{q_i}(x)$ are true. Likewise, as demonstrated in the *implication* proof, reconstruct $L = \Sigma^* \setminus \{01, 101, 010\}$ using the accepting states $P_{q_i}(x)$, similarly. It is clear that the accepting state's invariants build a language consisting of the empty string (P_{q_ϵ}), and any combination of 0s and 1s, excluding the specific strings 01, 101, and 010 ($P_{q_0}, P_{q_1}, P_{q_3}, P_{q_5}$). Thus, if there are accepting states $q \in F$ such that $P_q(x) \mid_{x\in\Sigma^*=\{0,1\}^*}$ holds, then $x \in L = \Sigma^* \setminus \{01, 101, 010\}$.

Therefore, $\mathcal{D}$ accepts $L = \Sigma^* \setminus \{01, 101, 010\}$ over $\Sigma = \{0, 1\}$. This means L is a regular language.

$\square$

(c):

<u>**Claim:**</u> $\{w | w \text{ is a palindrome}\}$ is NOT a regular language.

Proof.
Seeking a contradiction, assume the language $L = \{w|w \text{ is a palindrome}\}$ is regular. Then, there exists a DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$ that recognizes L, where $|Q| = k, k \in \mathbb{N}$

Next, choose a string $w \in L$ such that $|w| \geq k + 1$.
Let $w = 0^k 1^1 0^k \in L$. Notice that $|W| = 2k + 1 \geq k + 1$, and that the DFA must accept w.

As $\mathcal{D}$ processes the first k symbols of $w = 0^k 1^1 0^k$, some state $q \in Q$ must repeat, because the DFA only has k states, but processes more than k symbols. This repetition implies that $\mathcal{D}$ transitions into a loop at state q while reading the initial segment 0^k.

Let this loop correspond to some repeated substring s of 0^k. Because the DFA cannot remember how many 0s it has processed within this loop, it treats strings with any number of repetitions of s the same way.

Now construct a string $w' = 0^{k+1} 1^1 0^k$, where 0^{k+1} has one more 0 than 0^k. Since DFA processes w' exactly as it processes w, then $\mathcal{D}$ must also accept w'.

However, w' is not a palindrome: the number of 0s on the left $(k + 1)$ does not match the number of 0s on the right (k). This contradicts the assumption that $\mathcal{D}$ recognizes L, as $\mathcal{D}$ accepts $w' \notin L$.

By reaching a contradiction, the language $L = \{w|w \text{ is a palindrome}\}$ cannot be recognized by a DFA. Thus, L is not a regular language.

□

(d):
Claim: $\{ww|w \in \Sigma^*\}$ is NOT a regular language.

Proof.
Seeking a contradiction, assume the language $L = \{ww|w \in \Sigma^*\}$ is regular. Then, there exists a DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$ that recognizes L, where $|Q| = k, k \in \mathbb{N}$

Next, choose a string $x \in L$ such that $|x| \geq k+1$.
Let $x = 0^k 1^k 0^k 1^k \in L$, which is of length $2k+2 \geq k+1$. This string is in L because it has the form ww, where $w = 0^k 1^k$.

Since the DFA $\mathcal{D}$ has k states, it must repeat (loop) at some state $q \in Q$ while processing the first $k+1$ symbols of $x = 0^k 1^k 0^k 1^k$. That is

$$\delta(s, 0^i) = \delta(s, 0^{i+j}) \text{ for some } i \leq k \text{ and } j > 0.$$

Now consider modifying x by increasing the looped segment. Let $x' = 0^{k+j} 1^k 0^k 1^k$, where $j > 0$.
The DFA will process x' in the same way as x, because of the looping on state q.

It follows that the string $x' = 0^{k+j} 1^k 0^k 1^k \notin L$, because it does not have two identical halves. Specifically:

- x splits evenly as $x = 0^k 1^k 0^k 1^k = ww, w = 0^k 1^k$, but
- x' splits as $x' = 0^{k+j} 1^k 0^k 1^k$, where $0^{k+j} 1^k \neq 0^k 1^k$.

Yet, the DFA $\mathcal{D}$ cannot distinguish between x and x' (due to the loop), and will still accept w. This contradicts the assumption that $\mathcal{D}$ recognizes L, as $\mathcal{D}$ accepts $x' \notin L$.

By reaching a contradiction, it follows that no DFA can recognize $L = \{ww | w \in \Sigma^*\}$, because DFAs cannot ensure both halves of a string are identical. Thus, L is not regular.

$\square$

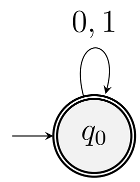
(e):
<u>Claim:</u> $\{w \mid ww \in \Sigma^*\}$ is a regular language.

Proof.
This proof aims to show that the language $\{w \mid ww \in \Sigma^*\}$ is a regular language, by constructing a DFA (with proof) that accepts all strings with its self-concatenation obtainable

over the alphabet $\Sigma = \{0, 1\}$.

Moreover, this proof recycles the DFA which was suggested in *Part (a)*, but with an adjustment to the state invariant. While it suffices to provide a double-subset inclusion proof to show an equivalence between $\{w \mid ww \in \Sigma^*\}$ and Σ^* (from *Part (a)*), proceed with the modified DFA proof nonetheless.

Consider the **transition function** δ with the following DFA:



Old State	**Symbol**	**New State**
q_0	0	q_0
q_0	1	q_0

Table 3: State Transition Table

Using δ, define the DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$, where

$$Q = \{q_0\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : Q \times \Sigma \to Q \text{ is the transition function defined by } \textit{Table 1}$$

$$s = q_0 \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_0\} \subseteq Q \text{ is the set of accepting states of } \mathcal{D}.$$

For the sole state q_0 in $\mathcal{D}$, define its **invariant** for strings $x \in \Sigma^* = \{0, 1\}^*$:

$$P_{q_0}(x) : xx \in \Sigma^* = \{0, 1\}^*$$

As the only state, q_0 is trivially **mutually exclusive**. As well, every string in $\Sigma^* = \{0, 1\}^*$ is either ϵ or consists of some number of 0s and 1s. Concatenate the empty string, ϵ, to itself and ϵ is the result. Concatenate a string consisting of 0s and 1s to itself, and the result is still a string consisting of 0s and 1s. There are no other strings in $\{0,1\}^*$ to test against this state invariant. Thus, q_0 satisfies **exhaustivity**.

Let $q \in Q = \{q_0\}$ and $x \in \Sigma^* = \{0,1\}^*$ both be arbitrary (here, $q = q_0$ always).
Denote the predicate:

$$P_{\delta(q_0,x)}(x) \coloneqq P_q(x) \text{ is the state invariant.}$$

Perform structural induction on $P_{\delta(q_0,x)}(x)$ for all strings $x \in \Sigma^* = \{0,1\}^*$, as follows:

Base Case:
Let $q = q_0$ and $w = \epsilon$.
$P_q(w) = P_{q_0}(\epsilon)$ is true as $\epsilon\epsilon = \epsilon \in \Sigma^* = \{0,1\}^*$.

Induction Hypothesis:
Assume that $P_q(w)$ is true for all $q \in \{q_0\}$ and some $w \in \{0,1\}^*$.
This means $ww \in \Sigma^* = \{0,1\}^*$.

Induction Step:
By the Induction Hypothesis, $P_q(w)$ is true for arbitrary $q \in \{q_0\}$ and some $w \in \{0,1\}^*$.

Demonstrate that the invariant of q_0 holds when processing all possible strings from Σ^*.
This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0,1\}$.

Let $w \in \{0,1\}^*$ be arbitrary.

Let $q = q_0$, $z \in \{0,1\}$, and assume $P_{q_0}(w)$ is true.
Consider that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,z)}(wz) = P_{q_0}(wz)$. Since $ww \in \Sigma^* = \{0,1\}^*$ and $z \in \{0,1\}$, it follows that $wzwz \in \Sigma^* = \{0,1\}^*$ as well. Namely, this is because strings consisting of 0s and 1s are members of $\Sigma^* = \{0,1\}^*$, and $wzwz$ matches this description as $w \in \{\epsilon, 0, 1\}$ while $z \in \{0,1\}$. By definition, $P_{q_0}(wz)$ holds. Thus, $P_{\delta(q,z)}(wz)$ holds.

There are no other state invariants in $\mathcal{D}$. By the principle of structural induction, $P_{\delta(q,x)}$ is true for all $q \in Q = \{q_0\}$ and $x \in \Sigma^* = \{0,1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language $L = \{w \mid ww \in \Sigma^*\}$ over $\Sigma = \{0,1\}$.
This is achievable by showing that if $x \in \Sigma^* = \{0,1\}^*$ is arbitrary, x is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(x)$ holds.

Recall the sole state invariant, $P_{q_0}(x) : xx \in \Sigma^* = \{0,1\}^*$.
If $x \in L$, then $xx \in \Sigma^*$. Clearly, $P_{q_0}(x)$ is true, due to matching definitions.
On the other hand, choose the accepting state $q_0 \in F$ and assume $P_{q_0}(x)$ is true. Then $xx \in \Sigma^*$, and $x \in L$ is, likewise, true due to matching definitions.

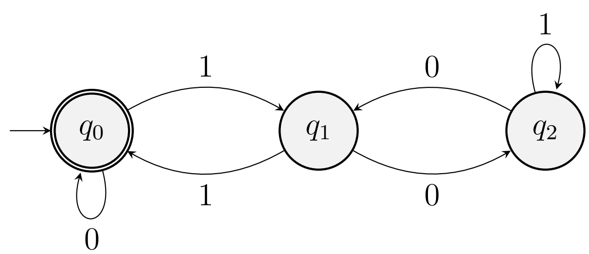
Therefore, $P_{\delta(q_0,x)}(x)$ is true for all strings $x \in \Sigma^* = \{0,1\}^*$. It has been demonstrated that $L = \{w \mid ww \in \Sigma^*\}$ is a regular language.

$\square$

(f):
<u>Claim:</u> $\{w \mid w$ is a binary representation of a multiple of $3\}$ is a regular language.

Proof.
This proof aims to show that the language $\{w \mid w$ is a binary representation of a multiple of $3\}$ is a regular language, by constructing a DFA (with proof) that accepts all strings over $\Sigma^* = \{0,1\}^*$ that are also a binary representation of multiples of 3.

Consider the **transition function** δ associated with the following DFA:



Old State	**Symbol**	**New State**
q_0	0	q_0
q_0	1	q_1
q_1	0	q_2
q_1	1	q_0
q_2	0	q_1
q_2	1	q_2

Table 4: State Transition Table

Using δ, define the DFA $\mathcal{D} = (Q, \Sigma, \delta, s, F)$, where

$$Q = \{q_0, q_1, q_2\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : Q \times \Sigma \to Q \text{ is the transition function define by } \textit{Table 4}$$

$$s = q_0 \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_0\} \subseteq Q \text{ is the set of accepting states of } \mathcal{D}.$$

For each state $q_i \mid_{i \in \{0,1,2\}}$ in $\mathcal{D}$, define a **state invariant** $P_q(x)$ for strings $x \in \Sigma^* = \{0, 1\}^*$:

$$P_{q_0}(x) : (0 \equiv (\text{integer representation of } x) \pmod 3) \text{ or } (x = \epsilon)$$
$$P_{q_1}(x) : 1 \equiv (\text{integer representation of } x) \pmod 3$$
$$P_{q_2}(x) : 2 \equiv (\text{integer representation of } x) \pmod 3$$

When performing division by 3 on integers, the result either yields a remainder of 0, 1, or 2. Clearly, these remainders are unique, so $P_{q_0}(x)$, $P_{q_1}(x)$, and $P_{q_2}(x)$ are **mutually exclusive** states. Note that ϵ is considered solely by the initial state, q_0.
Moreover, all strings in $\Sigma^* = \{0, 1\}^*$ are integer representations, if not ϵ. After a division by 3, the possible remainders for these integer representations are cases directly covered by the three state invariants. Therefore, some state invariant must hold for an arbitrary string's integer representation. This means the collection of states in $\mathcal{D}$ must be **exhaustive**.

Let $q \in Q = \{q_0, q_1, q_2\}$ and $w \in \Sigma^* = \{0, 1\}^*$ both be arbitrary.

Denote the predicate:

$$P_{\delta(q_0,w)}(w) \coloneqq P_q(w) \text{ is the state invariant.}$$

Perform structural induction as follows:

Base Cases:
Let $w = \epsilon$.
Let $q = q_0$. $P_q(w) = P_{q_0}(\epsilon)$ is true as $w = \epsilon$, satisfying the definition of the state invariant.
Let $q \neq q_0$. $P_q(w) = P_{q_i}(\epsilon) \mid_{i \in \{1,2\}}$ are vacuously true as these corresponding states are non-initial and do not process ϵ.

Induction Hypothesis:
Assume that $P_q(w)$ is true for all $q \in \{q_0, q_1, q_2\}$ and some $w \in \{0,1\}^*$.

Induction Step:
By the Induction Hypothesis, $P_q(w)$ is true for arbitrary $q \in \{q_0, q_1, q_2\}$ and some $w \in \{0,1\}^*$.

Demonstrate that all state invariants hold when processing strings from $\Sigma = \{0, 1\}$.
This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0, 1\}$.

Let $w \in \{0,1\}^*$ be arbitrary. Let W be the integer representation of the string w. Then, consider the following cases.

Case ($q = q_0, z = 0$):
Assume $P_{q_0}(w)$ is true. Then, $0 \equiv W \pmod 3$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,0)}(w0) = P_{q_0}(w0)$. Notice that, in treating the string $wz = w0$ as a binary integer representation, concatenating 0 to w performs a logical left shift (a multiplication by 2).

Recall that W is the integer representation of the string w; $2W$ shall be the integer representation of the string $wz = w0$ (due to the logical left shift). Since $0 \equiv W \pmod 3$, it follows that $3 \mid W$, meaning $\exists n \in \mathbb{N}^+$ such that $W = 3n$.

Perform the multiplication by 2 and notice, $2W = 2 \times 3n = 3(2n)$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W = 3(2n) = 3m$, meaning $3 \mid 2W$. It has been clearly demonstrated that $0 \equiv 2W \pmod 3$, where $2W$ is the integer representation of $wz = w0$. Thus, $P_{\delta(q,z)}(wz) = P_{q_0}(w0)$ is true.

Case ($q = q_0, z = 1$):
Likewise, assume $P_{q_0}(w)$ is true. Then, $0 \equiv W \pmod 3$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_0,1)}(w1) = P_{q_1}(w1)$.

Declare $2W+1$ as the integer representation of the string $wz = w1$, due to a logical left shift and addition by 1 (the result of concatenating 1 to w instead of 0). Since $0 \equiv \pmod 3$, there exists $n \in \mathbb{N}^+$ such that $W = 3n$.

Perform the multiplication by 2, followed by an addition of 1, and notice that $2W + 1 = 2 \times 3n + 1 = 3(2n) + 1$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W = 3(2n) + 1 = 3m + 1$, meaning $2W$ yields a remainder of 1 after division by 3.

It follows that $1 \equiv 2W + 1 \pmod 3$. This means $P_{\delta(q,z)}(wz) = P_{q_1}(w1)$ holds, as $2W + 1$ is the integer representation of $wz = w1$.

Case ($q = q_1, z = 0$):
Assume $P_{q_1}(w)$ is true. Then, $1 \equiv W \pmod 3$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_1,0)}(w0) = P_{q_2}(w0)$.

Declare $2W$ to be the integer representation of the string $wz = w0$, due to a logical left shift in concatenating 0 to w. Since $1 \equiv W \pmod 3$, it follows that $\exists n \in \mathbb{N}^+$ such that $W = 3n + 1$.

Perform the multiplication by 2 and notice, $2W = 2 \times (3n + 1) = 3(2n) + 2$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W = 3(2n) + 2 = 3m + 2$, meaning $2W$ yields a remainder of 2 after division by 3. It has been clearly demonstrated that $2 \equiv 2W \pmod 3$, where $2W$ is the integer representation of $wz = w0$. Thus, $P_{\delta(q,z)}(wz) = P_{q_2}(w0)$ is true.

<u>Case $(q = q_1, z = 1)$:</u>
Likewise, assume $P_{q_1}(w)$ is true. Then, $1 \equiv W \pmod 3$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_1,1)}(w1) = P_{q_0}(w1)$. Note that, in treating $wz = w1$ as a binary integer representation, concatenating 1 to w performs a logical left shift (a multiplication by 2) followed by an addition by 1.

Declare $2W + 1$ to be the integer representation of the string $wz = w1$. Since $1 \equiv W \pmod 3$, it follows that $\exists n \in \mathbb{N}^+$ such that $W = 3n + 1$.

Perform the multiplication by 2, followed by an addition of 1, and notice that $2W + 1 = 2 \times (3n + 1) + 1 = 3(2n) + 2 + 1 = 3(2n) + 3 = 3(2n + 1)$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W + 1 = 3(2n + 1) = 3m$, meaning $3 \mid 2W + 1$.

It follows that $0 \equiv 2W + 1 \pmod 3$. This means, $P_{\delta(q,z)}(wz) = P_{q_0}(w1)$ holds, as $2W + 1$ is the integer representation of $wz = w1$.

<u>Case $(q = q_2, z = 0)$:</u>
Assume $P_{q_2}(w)$ is true. Then, $2 \equiv W \pmod 3$.
It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_2,0)}(w0) = P_{q_1}(w0)$.

Declare $2W$ as the integer representation of the string $wz = w0$, due to the logical left shift. Since $2 \equiv W \pmod 3$, it follows that $\exists n \in \mathbb{N}^+$ such that $W = 3n + 2$.

Perform the multiplication by 2 and notice, $2W = 2 \times (3n+2) = 3(2n)+4 = 3(2n)+3+1 = 3(2n + 1) + 1$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W = 3(2n + 1) + 1 = 3m + 1$, meaning $2W$ yields a remainder of 1 after division by 3. It has been demonstrated that $1 \equiv 2W \pmod 3$, where $2W$ is the integer represesntation of $wz = w0$. Thus, $P_{\delta(q,z)}(wz) = P_{q_1}(w0)$ is true.

<u>Case $(q = q_2, z = 1)$:</u>
Likewise, assume $P_{q_2}(w)$ is true. Then, $2 \equiv W \pmod 3$.

It follows that $P_{\delta(q,z)}(wz) = P_{\delta(q_2,1)}(w1) = P_{q_2}(w1)$.

Declare $2W + 1$ to be the integer representation of the string $wz = w1$ (one shift logically left yields multiplication by 2, add 1 as $z = 1$). Since $2 \equiv W \pmod 3$, it follows that $\exists n \in \mathbb{N}^*$ such that $W = 3n + 2$.

Perform the multiplication by 2, followed by an addition of 1, and notice that $2W + 1 = 2 \times (3n+2)+1 = 3(2n)+4+1 = 3(2n)+3+2 = 3(2n+1)+2$. Declare that $\exists m \in \mathbb{N}^+$ such that $2W+1 = 3(2n+1)+2 = 3m+2$, meaning $2W+1$ yields a remainder of 2 after division by 3.

It follows that $1 \equiv 2W + 1 \pmod 3$. This means $P_{\delta(q,z)}(wz) = P_{q_2}(w1)$ holds, as $2W + 1$ is the integer representation of $wz = w1$.

<u>End of Cases:</u>
Hence, it is demonstrated that if $P_q(w)$ is true for all $q \in \{q_0, q_1, q_2\}$ and some $w \in \{0,1\}^*$, then $P_{\delta(q,z)}(wz)$ holds for all $z \in \{0,1\}$. By the principle of structural induction, $P_{\delta(q,x)}$ is true for all $q \in Q = \{q_0, q_1, q_2\}$ and $x \in \Sigma^* = \{0,1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language

$$L = \{w \mid w \text{ is a binary representation of a multiple of } 3\}$$

over $\Sigma = \{0, 1\}$.
This is achievable by showing that if $x \in \Sigma^* = \{0,1\}^*$ is arbitrary, x is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(x)$ holds.

Recall the sole accepting state invariant,

$$P_{q_0}(x) : (0 \equiv (\text{integer representation of } x) \pmod 3) \text{ or } (x = \epsilon).$$

<u>*Implication*</u>

If $x \in L$ is arbitrary, then x can be interpreted as a binary representation of a multiple of 3. Notice that the definition of $P_{q_0}(x)$ directly fits the definition of L, so $P_{q_0}(x)$ is indeed true for $q_0 \in F$.

Implied-by

Consider $q_0 \in F$ and assume $P_{q_0}(x)$ is true for arbitrary $x \in \Sigma^* = \{0,1\}^*$. Clearly, x is also a member of L as $P_{q_0}(x)$ suggests that the integer representation of x is indeed a multiple of 3.

Therefore, $\mathcal{D}$ accepts $L = \{w \mid w \text{ is a binary representation of a multiple of } 3\}$ over $\Sigma = \{0,1\}$. This means L is a regular language.

$\square$

Question #2

Claim: Regular expressions that also have access to complement can still only express the same class of languages (i.e. the class of regular langauges) as regular expressions without the complement operation.

Proof.
Suppose r is an arbitrary regular expression with alphabet Σ so that $L = \mathcal{L}(r)$ is a regular language.
Assume r has access to the complement operation.

Let $\overline{L} = \{x \in \Sigma^* \mid x \notin L\}$ be the regular language representing the *complement* of L.
Assume $\overline{r}$ is a regular expression, with $\mathcal{L}(\overline{r}) = \overline{\mathcal{L}(r)}$.

By definition, there exists a DFA $\mathcal{M}$ that accepts L.

Construct $\mathcal{M}$:
$\mathcal{M} = (Q, \Sigma, \delta, s, F)$, where

- Q is a set of finite states;
- Σ is the alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function;
- s is the start state;
- $F \subseteq Q$ is the set of accepting states.

Next, define $\overline{L} = \Sigma^* \setminus L$ as the regular language containing everything obtainable from Σ except members of L.

Construct $\overline{M}$ to be identical to M, except for its accepting states:

- Q is a set of finite states;

- Σ is the alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function;
- s is the start state;
- $Q \setminus F \subseteq Q$ is the set of accepting states.

Notice that $\overline{M}$ share the same states, alphabet, transition function, and start state as those of M. The difference is the set of accepting states in $\overline{M}$, which is designed to be mutually exclusive from that of M.

Self-Note: now add the proof that $\overline{M}$ accepts $\overline{L}$, and connect it back to regexes. Next, let $\overline{r}$ be some regular expression without the complement operation.

$\square$

Question #3

Counter-free languages are a subset of languages that satisfy the condition:

$$(\exists n \in \mathbb{N})(\forall x, y, z \in \Sigma^*)(\forall m \geq n)(xy^m z \in L \iff xy^n z \in L).$$

Star-free regular expressions are regular expressions without the Kleene star, but with complementation.

It is known in formal language theory that counter-free languages are equivalent to the languages that can be expressed as **star-free regular expressions**.

(a):
Claim: $(ab)^*$ can be matched with a star-free regular expression, where $\Sigma = \{a, b\}$.

Proof.
The expression $(ab)^*$ represents strings in the set $\{\epsilon, ab, abab, ababab, \dots\}$. In other words, strings in $(ab)^*$ consist of zero or more repetitions of the substring ab. This means matching strings are strings where every occurrence of a is immediately followed by a b and the string must not have any extraneous characters or mismatches.

Strings in $(ab)^*$ **must not**:

- Start with b,
- End with a,
- Contain adjacent as (aa),
- Contain adjacent bs (bb).

Note that if L is a language, then its *complement* is the language $\overline{L} = \{x \in \Sigma^* \mid x \notin L\}$. For an equivalent notation in regular expressions, if r is a regex matching $\mathcal{L}(r)$, then $\overline{r}$ is a regular expression such that $\mathcal{L}(\overline{r}) = \overline{\mathcal{L}(r)}$. This makes $\overline{\varnothing}$ is the star-free regex of Σ^*.

Let the complement regex of $(ab)^*$ be $\overline{(ab)^*} = b\overline{\varnothing} + \overline{\varnothing}a + \overline{\varnothing}aa\overline{\varnothing} + \overline{\varnothing}bb\overline{\varnothing}$.

Then, $(ab)^* = \overline{b\overline{\varnothing} + \overline{\varnothing}a + \overline{\varnothing}aa\overline{\varnothing} + \overline{\varnothing}bb\overline{\varnothing}}$.

Clearly, $r = \overline{b\overline{\varnothing} + \overline{\varnothing}a + \overline{\varnothing}aa\overline{\varnothing} + \overline{\varnothing}bb\overline{\varnothing}}$ is a star-free regex. Thus, $(ab)^*$ can be expressed as a star-free regular expression by using complements to describe its constraints.

$\square$

(b):

<u>Claim:</u> $(ab)^*$ is a counter-free language, where $\Sigma = \{a, b\}$.

Proof.

By definition, if $(ab)^*$ is a counter-free language, there exists natural n for all $x, y, z \in \{a, b\}^*$ and for all $m \geq n$ such that $xy^m z \in (ab)^* \iff xy^n z \in (ab)^*$.

Let $n \in \mathbb{N}$. Let $x, y, z \in \{a, b\}^*$ and $m \geq n$ both be arbitrary.

<u>Show that $xy^m z \in (ab)^* \implies xy^n z \in (ab)^*$:</u>

Suppose $xy^m z \in (ab)^*$.

The following cases may arise.

<u>Case $(x = a)$</u>

Assume $x = a$. Then, $xy^m z \in (ab)^*$ implies $z = b$, as strings that start with a must end with b. This makes $y = ba$ so that $y^m = (ba)^m$ is a sequence of ba. It follows that y^n is also a sequence of ba.

Therefore, the string $xy^n z$ starts with a, follows with a sequence of ba, and ends with b. Collectively, this string is indeed some number of concatenations of ab, so $xy^n z \in (ab)^*$.

<u>Case $(x = ab)$</u>

Assume $x = ab$. Then, $xy^m z \in (ab)^*$ implies $z = ab$ as well, where $y = ab$ so that $y^m = (ab)^m$ is a sequence of ab. It follows that y^n is also a sequence of ab.

Therefore, the string xy^nz starts with ab, follows with a sequence of ab, and ends with ab. Collectively, this string is indeed some number of concatenations of ab, so $xy^nz \in (ab)^*$.

Altogether:
Thus, $xy^nz \in (ab)^*$.

Show that $xy^mz \in (ab)^* \impliedby xy^nz \in (ab)^*$:
Suppose $xy^nzin(ab)^*$.

Consider the same cases as previously noted.

Case $(x = a)$
Assume $x = a$. Then, $xy^nz \in (ab)^*$ implies $z = b$, as strings that start with a must end with b. This makes $y = ba$ so that $y^n = (ba)^n$ is a sequence of ba. It follows that y^m is also a sequence of ba (having only a greater number of concatenations of ba).

Therefore, the string xy^mz starts with a, follows with a sequence of ba, and ends with b. Collectively, this string is indeed some number of concatenations of ab, so $xy^mz \in (ab)^*$.

Case $(x = ab)$
Assume $x = ab$. Then, $xy^nz \in (ab)^*$ implies $z = ab$ as well, where $y = ab$ so that $y^n = (ab)^m$ is a sequence of ab. It follows that y^m is also a sequence of ab (having only a greater number of concatenations of ab).

Therefore, the string xy^mz starts with ab, follows with a sequence of ab, and ends with ab. Collectively, this string is indeed some number of concatenations of ab, so $xy^mz \in (ab)^*$.

Altogether:
Thus, $xy^mz \in (ab)^*$.

Conclusion:
It has been demonstrated that $xy^mz \in (ab)^* \implies xy^nz \in (ab)^*$ and $xy^mz \in (ab)^* \impliedby$

$xy^nz \in (ab)^*$. Therefore, $xy^mz \in (ab)^* \iff xy^nz \in (ab)^*$. This makes $(ab)^*$ a counter-free language over $\Sigma = \{a, b\}$.

□

(c):

<u>Claim:</u> $(aa)^*$ is NOT a counter-free language, where $\Sigma = \{a\}$. *Counterexample.*

Proof.
proofgoeshere

□

Question #4

Let $k \in \mathbb{N}$ be arbitrary. Let $w \in \Sigma^*$, where $|\Sigma| \geq 2$ and has 1 as one of its symbols.

Consider the language $L = \{w | \text{the } k^{\text{th}} \text{ to last character of } w \text{ is } 1\}$.

(a):

Claim: A DFA that accepts L has to have at least 2^k number of states.

Proof.
ACTUALLY, SHOW THIS BY CONTRADICTION: Suppose whatever... less than 2^k states.

A DFA is determinstic and requires states to remember the "history" of the input. For the language $L = \{w | \text{the } k^{\text{th}} \text{ to last character of } w \text{ is } 1\}$, the DFA must track the last k characters of the input string.

Notice that there are 2^k possible combinations of k-length binary substrings, and each of these combinations must map to a unique state in the DFA for accurate processing.

Any DFA with fewer than 2^k states cannot differentiate between all possible k-length suffixes, causing the automaton to classify strings incorrectly.

Any DFA with more than 2^k states either accepts L with a larger alphabet with more than 2 symbols, or is introducing repetitive and redundant states, but still works.

□

(b):

Claim: The smallest NFA that accepts L has to have exactly k number of states.

Proof.
In an NFA, non-determinism allows the automaton to "guess" when it is k-steps away from the end of the string.

The NFA for L needs only k states because: - The start state (initial state) represents the starting position; - The NFA transitions through $k-1$ intermediate states to track progress.

□

(c):
Claim: The smallest DFA that accepts L has to have exactly $2^{k+1}-1$ number of states.

Proof.
proofgoeshere

□

Question #5

<u>Claim:</u> Every finite language can be represented by a regular expression (meaning all finite languages are regular).

Proof.
Let Σ be an arbitrary alphabet. Let L be an arbitrary finite language over Σ.

Let n be an arbitrary natural number.
Denote the predicate:

$$P(n) := |L_n| = n \implies L_n \text{ can be represented as a regular expression.}$$

This proof uses the principle of simple induction to show $P(n)$ for all $n \in \mathbb{N}$.

<u>Base Cases:</u>
Let $n = 0$.
This means $|L_n| = 0$, so $L_n = \emptyset$. By definition, the empty set is a regular expression.
Thus, $P(0)$.

Let $n = 1$.
Then $|L_n| = 1$, so $L_n = \{w\}$ for some string $w \in \Sigma^*$. By definition, any single string over an alphabet is a regular expression.
Thus, $P(1)$.

<u>Induction Hypothesis</u>
Assume that $P(k)$ holds for some natural k.

This means if L_k has k strings, then L_k can be represented as a regular expression.

<u>Induction Step:</u>
Let $L_{k+1} = \{w_1, w_2, \ldots, w_k, w_{k+1}\}$, where $w_i \in \Sigma^*$ for $i \in [1, k+1] \cap \mathbb{N}$.

By the Induction Hypothesis, $L_k = L_{k+1} \setminus \{w_{k+1}\} = \{w_1, w_2, \ldots, w_k, w_{k+1}\} \setminus \{w_{k+1}\} = \{w_1, w_2, \ldots, w_k\}$ has language has regular expression r_k such that $L_k = \mathcal{L}(r_k)$.

Notice that:

- The regex r_k represents the language L_k;
- The regex w_{k+1} represents the language $\{w_{k+1}\}$.

Then, L_{k+1} can be constructed as a regex as follows:

$$L_{k+1} = L_k \cup \{w_{k+1}\}$$

By definition, the union of two regexes is a regex. Construct r_{k+1}:

$$r_{k+1} = r_k + w_{k+1}$$

As desired, the regex r_{k+1} represents the language L_{k+1}.

Conclusion:
By the principle of simple induction, $P(n)$ holds for all $n \in \mathbb{N}$. It follows that all finite languages must be regular.

$\square$