COL352 HW 1

Problem 1

Let $x, y, z \in \Sigma^*$. We say that z is a shuffle of x and y if the characters in x and y can be interleaved, while maintaining their relative order within x and y, to get z. Formally, if $|x| = m$ and $|y| = n$, then $|z|$ must be $m + n$, and it should be possible to partition the set $\{1, 2, \ldots, m + n\}$ into two increasing sequences, $i_1 < i_2 < \cdots < i_m$ and $j_1 < j_2 < \cdots < j_n$, such that $z[i_k] = x[k]$ and $z[j_k] = y[k]$ for all k.

Given two languages $L_1, L_2 \subseteq \Sigma^*$, define $\text{shuffle}(L_1, L_2) = \{z \in \Sigma^* \mid z \text{ is a shuffle of some } x \in L_1 \text{ and some } y \in L_2\}$. Prove that the class of regular languages is closed under the shuffle operation.

Solution.
We begin with some notation for the sake of convenience.

Notation 1

For a string $z[1 \ldots n]$ and a sequence $I = I_1 < I_2 \cdots < I_k$ satisfying $\{I_1, I_2, \ldots, I_k\} \subseteq \{1, 2, \ldots n\}$, we define $z[I] = z[I_1] \cdot z[I_2] \cdots z[I_k]$.

Since both L_1, L_2 are regular, they must be recognizable by a DFA. Let the DFAs be $D_1 = (Q_1, \Sigma, \delta_1, q_{10}, A_1)$ and $D_2 = (Q_2, \Sigma, \delta_2, q_{20}, A_2)$.

Construct the NFA $N = (Q, \Sigma, \Delta, Q_0, A)$ where

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \Delta \subseteq Q \times \Sigma \times Q &= \{((q_1, q_2), a, (\delta_1(q_1, a), q_2)) \; \forall q_1, q_2 \in Q_1 \times Q_2 \; \forall a \in \Sigma\} \\ &\quad \cup \{((q_1, q_2), a, (q_1, \delta_2(q_2, a))) \; \forall q_1, q_2 \in Q_1 \times Q_2 \; \forall a \in \Sigma\} \\ Q_0 &= \{(q_{10}, q_{20})\} \\ A &= A_1 \times A_2 \end{aligned}$$

We shall show that language recognized by the NFA N is precisely the shuffle of L_1 and L_2.

Claim 1.1

$$\mathcal{L}(N) = \text{shuffle}(L_1, L_2)$$

To prove this claim we will prove that $\mathcal{L}(N) \subseteq \text{shuffle}(L_1, L_2)$ as well as $\text{shuffle}(L_1, L_2) \subseteq \mathcal{L}(N)$.

Claim 1.2

$\mathcal{L}(N) \subseteq \text{shuffle}(L_1, L_2)$

Proof. Let $e = q_0 z_1 q_1 \ldots z_n q_n$ where $q_i \in Q$ and $z_i \in \Sigma$ be an accepting run of N. For any $1 \leq i \leq n$ we shall unpack $q_i \in Q_1 \times Q_2$ into (q_i^1, q_i^2) ("components" of a state). Note that since $q_{i-1} z_i q_i \in \Delta$ we have $q_i = (\delta_1(q_{i-1}^1, z_i), q_{i-1}^2)$ or $(q_{i-1}^1, \delta_2(q_{i-1}^2, z_i))$. Also let

$$\begin{aligned} I_x(k) &= \{i \mid 1 \leq i \leq k \wedge q_i = (\delta_1(q_{i-1}^1, z_i), q_{i-1}^2)\} \\ I_y(k) &= \{i \mid 1 \leq i \leq k \wedge q_i = (q_{i-1}^1, \delta_2(q_{i-1}^2, z_i))\} \setminus I_x(k) \end{aligned}$$

Note that since $A \cup (B \setminus A) = A \cup B$, we have $I_x(k) \cup I_y(k) = \{i \mid 1 \leq i \leq k \wedge q_i = (\delta_1(q_{i-1}^1, z_i), q_{i-1}^2)\} \cup \{i \mid 1 \leq i \leq k \wedge q_i = (q_{i-1}^1, \delta_2(q_{i-1}^2, z_i))\} = \{i \mid 1 \leq k \wedge \big(q_i = (\delta_1(q_{i-1}^1, z_i), q_{i-1}^2) \vee q_i = (q_{i-1}^1, \delta_2(q_{i-1}^2, z_i))\big)\} = \{1, \ldots, k\}$, where the last equality follows from

the fact that any transition is of one of those forms. Also, A and $B \setminus A$ are disjoint by definition. Hence $I_x(k)$ and $I_y(k)$ are disjoint and their union is $\{1, \ldots, k\}$, so $I_x(k)$ and $I_y(k)$ are disjoint partitions of $\{1, \ldots, k\}$. We shall prove the following induction hypothesis.

Induction Hypothesis $P(k)$: For all $0 \leq k \leq n$ $q_k = (\hat{\delta}_1(q_{10}, x(k)), \hat{\delta}_2(q_{20}, y(k)))$, where $x(k) := z[I_x(k)]$ and $y(k) := z[I_y(k)]$

Base Case When $i = 0$, $q_0 = (q_{10}, q_{20})$ as it is the only possible starting state in the NFA N.

Inductive Case Suppose we have shown that the inductive hypothesis holds for all $0 \leq k \leq i-1$. We shall now show that this implies that the hypothesis also holds for $k = i$.

Since the inductive hypothesis holds for $k = i-1$ by assumption, then $q_{i-1} = (\hat{\delta}_1(q_{10}, x(i-1)), \hat{\delta}_2(q_{20}, y(i-1)))$. Now 2 cases arise:

1. $i \in I_x(i)$: This implies that $q_i = (\delta_1(q^1_{i-1}, z_i), q^2_{i-1})$, $I_x(i) = \{i\} \cup I_x(i-1)$ and $I_y(i) = I_y(i-1)$
2. $i \in I_y(i)$: This implies that $q_i = (q^1_{i-1}, \delta_2(q^2_{i-1}, z_i))$, $I_x(i) = I_x(i-1)$ and $I_y(i) = \{i\} \cup I_y(i-1)$

In either case we can now see that $q_i = (\hat{\delta}_1(q_{10}, x(i)), \hat{\delta}_2(q_{20}, y(i)))$. Hence the inductive step is complete and induction is established.

As a direct consequence of our inductive hypothesis we have that $q_n = (\hat{\delta}_1(q_{10}, x(n)), \hat{\delta}_2(q_{20}, y(n)))$. Since e is an accepting run q_n must be an accepting state. That is, $q_n \in A_1 \times A_2$ which implies $q^1_n \in A_1$ and $q^2_n \in A_2$. So finally we have that $\hat{\delta}_1(q_{10}, x(n)) \in A_1$ and $\hat{\delta}_2(q_{20}, y(n)) \in A_2$. Thus establishing that $x(n) \in L_1$ and $y(n) \in L_2$ (because they are recognized by the respective DFAs). So we have been able to partition $z_{1\ldots n} \equiv z$ into 2 sequences $x(n)$ and $y(n)$ such that $x(n) \in L_1$ and $y(n) \in L_2$ and z is a shuffle of $x(n)$ and $y(n)$. Since our choice of z was arbitrary, we have shown that for each $z \in \mathcal{L}(N)$, $z \in \text{shuffle}(L_1, L_2)$, which finishes the proof of this claim. □

Claim 1.3

$\text{shuffle}(L_1, L_2) \subseteq \mathcal{L}(N)$

Proof.
Consider any string $z[1 \ldots n] \in \text{shuffle}(L_1, L_2)$. By definition of a shuffle we have that there exists 2 increasing disjoint partitions of $\{1, \ldots, n\}$, namely I_x and I_y, such that $z[I_x] \in L_1$ and $z[I_y] \in L_2$. We shall construct an accepting run corresponding to this string on the NFA N.
Consider the run $q_0 z_1 q_1 \ldots z_n q_n$, where

$$q_0 = (q_{10}, q_{20})$$
$$q_i = \begin{cases} (\delta_1(q^1_{i-1}, z_i), q^2_{i-1}) & i \in I_x \\ (q^1_{i-1}, \delta_2(q^2_{i-1}, z_i)) & i \in I_y \end{cases} \quad 1 \leq i \leq n$$

Note that this is a valid run by construction as every transition belongs to Δ already, q_0 is a valid starting state, $z_i \in \Sigma$, and $q_i \in Q_1 \times Q_2$. To also prove that this run is an accepting run we shall prove the following induction hypothesis.

Induction Hypothesis $P(k)$: For all $0 \leq k \leq n$, $q_k = (\hat{\delta}_1(q_{10}, x(k)), \hat{\delta}_2(q_{20}, y(k)))$. Where $x(k) = z[\{i \in I_x : i \leq k\}]$ and $y(k) = z[\{i \in I_y : i \leq k\}]$.

Base Case When $k = 0$, we have $q_0 = (q_{10}, q_{20})$ as that is the only starting state of NFA N.

Inductive Case Suppose we have already shown that the hypothesis holds for all $0 \leq k \leq i-1$. We shall now show that this implies that hypothesis also holds for $k = i$.

Since hypothesis holds for $i-1$ we have $q_{i-1} = (\hat{\delta}_1(q_{10}, x(i-1)), \hat{\delta}_2(q_{20}, y(i-1)))$. Now, here we have 2 cases:

1. $i \in I_x$: Then we have $q_i = (\delta_1(q^1_{i-1}, z_i), q^2_{i-1})$ and $x(i) = x(i-1) \cdot z_i$ and $y(i) = y(i-1)$.
2. $i \in I_y$: Then we have $q_i = (q^1_{i-1}, \delta_2(q^2_{i-1}, z_i))$ and $x(i) = x(i-1)$ and $y(i) = y(i-1) \cdot z_i$.

In either case we can conclude that $q_i = (\hat{\delta}_1(q_{10}, x(i)), \hat{\delta}_2(q_{20}, y(i)))$ thereby establishing the inductive step. Hence the inductive hypothesis is proved.

Again as the direct consequence of the inductive hypothesis, we have $q_n = (\hat{\delta}_1(q_{10}, x(n)), \hat{\delta}_2(q_{20}, y(n)))$. But note that $x(n) = z[I_x]$ and $y(n) = z[I_y]$. And therefore $x(n) \in L_1$ and $y(n) \in L_2$ which implies that $\hat{\delta}_1(q_{10}, x(n)) \in A_1$ and $\hat{\delta}_2(q_{20}, y(n)) \in A_2$. Hence $q_n \in A_1 \times A_2$. Therefore, q_n is an accepting state and the constructed run is a valid accepting run on the NFA. This implies that for each string in L, there exists an accepting run of N on it, from where the conclusion of the claim follows. $\square$

Problem 2

Let L_1 be a regular language and L_2 be any language (not necessarily regular) over the same alphabet Σ. Prove that the language $L = \{x \in \Sigma^* \mid x \cdot y \in L_1 \text{ for some } y \in L_2\}$ is regular. Do this by mathematically defining a DFA for L starting from a DFA for L_1 and the language L_2.

Solution.
Since the language L_1 is regular, there must be a DFA $D = (Q, \Sigma, \delta, q_0, A)$ that recognizes L_1.
Consider the following construction for the DFA of the language L:

$$D' = (Q, \Sigma, \delta, q_0, S)$$

Where

$$S = \{q \in Q \mid \exists y \in L_2 \text{ such that } \hat{\delta}(q, y) \in A\}$$

It is easy to see that this is indeed a DFA, as $S \subseteq Q$, and the rest of the properties follow from D being a DFA. We claim that this DFA accepts L.

We begin the proof with a lemma that shall be repeatedly used in the following claims.

Lemma 2.1

Let $D = (Q, \Sigma, \delta, q_0, A)$ be any DFA. For strings $x, y \in \Sigma^*$, and a state $q \in Q$, we have $\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(q, x \cdot y)$

Proof. The proof proceeds by induction on the length of y, say n.
The base cases are when $n = 0, 1$.

$n = 0$: in this case, we have $\hat{\delta}(q, x \cdot y) = \hat{\delta}(q, x \cdot \epsilon) = \hat{\delta}(q, x) = \hat{\delta}(\hat{\delta}(q, x), \epsilon) = \hat{\delta}(\hat{\delta}(q, x), y)$, as needed.
$n = 1$: in this case, we have $\hat{\delta}(q, x \cdot y) = \delta(\hat{\delta}(q, x), y) = \hat{\delta}(\hat{\delta}(q, x), y)$, since y is a single character.

$n > 1$: in this case, let $y = z \cdot a$ for some a of length 1. Then we have $\hat{\delta}(q, x \cdot y) = \hat{\delta}(q, (x \cdot z) \cdot a) = \hat{\delta}(\hat{\delta}(q, x \cdot z), a) = \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), z), a) = \hat{\delta}(\hat{\delta}(q, x), z \cdot a) = \hat{\delta}(\hat{\delta}(q, x), y)$, as required. The first equality follows by associativity of concatenation, the second by the base case applied on y replaced by a, the third by the inductive hypothesis applied on y replaced by z, the fourth by the base case applied on q replaced by $\hat{\delta}(q, x)$ and x, y replaced by z, a, and the fifth by the equality of y and $z \cdot a$.

Hence, we are done by induction and the result is established. □

We divide the rest of the proof into two claims, both when combined shall give $L = \mathcal{L}(D')$, after which we shall be done, since D' would be a DFA recognizing L, implying that L is regular.

Claim 2.1

$\mathcal{L}(D') \subseteq L$

Proof. Consider any string $x \in \mathcal{L}(D')$. Then, $x = x[1]x[2]\dots x[n]$ for some n, and $\hat{\delta}(q_0, x) \in S$. Suppose $\hat{\delta}(q_0, x) = q \in S$. Then by definition of S, we must have $\hat{\delta}(q, y) \in A$ for some $y \in L_2$. Hence, we have $\hat{\delta}(\hat{\delta}(q_0, x), y) \in A \implies \hat{\delta}(q_0, x \cdot y) \in A \implies x \cdot y \in L_1 \implies x \in L$. □

Claim 2.2

$L \subseteq \mathcal{L}(D')$

Proof. Consider any string $x \in L$. Then we have a string $y \in L_2$ such that $x \cdot y \in L_1$. We have $\hat{\delta}(\hat{\delta}(q_0, x), y) = \hat{\delta}(q_0, x \cdot y) \in A$, so by the definition of S, we must have $\hat{\delta}(q_0, x) \in S$. Hence this implies

that D' accepts x, so $x \in \mathcal{L}(D')$, whence we are done. $\square$

Problem 3

Prove that the class of regular languages is closed under inverse homomorphisms. That is, prove that if $L \subseteq \Gamma^*$ is a regular language and $f : \Sigma^* \to \Gamma^*$ is a homomorphism, then $f^{-1}(L) = \{x \in \Sigma^* \mid f(x) \in L\}$ is regular. Do this by mathematically defining a DFA for $f^{-1}(L)$ starting from a DFA for L and the function f.

Solution. Since the language L is regular then there must be DFA, $D = (Q, \Gamma, \delta, q_0, A)$ that recognizes L. Consider the following construction for the DFA of the language $f^{-1}(L)$:

$$D' = (Q, \Sigma, \delta', q_0, A)$$

Where

$$\delta'(q, a) = \hat{\delta}(q, f(a)) \text{ where } a \in \Sigma, q \in Q$$

Consider any $x \in f^{-1}(L)$. Then by definition of f^{-1}, we have $f(x) \in L$, and we claim that

Claim 3.1

$\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, f(x))$

Proof.
Let $x = x[1] \cdot x[2] \cdot x[3] \cdots x[n]$, where $x[i] \in \Sigma$, Now using the homomorphism property of f we get $f(x) = f(x[1]) \cdot f(x[2]) \cdots f(x[n])$ We shall now prove the claim by induction on n.

Induction Hypothesis $P(n)$: For all $x \in \Sigma^*$ which can be written as $x = x[1] \cdot x[2] \cdot x[3] \cdots x[n]$, where $x[i] \in \Sigma$. We have, $\hat{\delta}(q_0, f(x)) = \hat{\delta}'(q_0, x)$

Base Case When $n = 0$, this means $x = \epsilon$ and therefore $f(x) = \epsilon$, so for both $\hat{\delta}(q_0, f(x)) = \delta'(q_0, x) = q_0$.
When $n = 1$, this is trivially true as $\hat{\delta}(q_0, f(x[1])) = \delta'(q_0, x[1])$ from definition of δ' and $\hat{\delta}'(q_0, x[1]) = \delta'(q_0, x[1])$ as $x[1] \in \Sigma$.

Inductive case Let us assume that $P(k)$ is true for all $1 \leq k \leq n-1$, Now we will show then that $P(n)$ holds true.

$$\begin{aligned}
\hat{\delta}(q_0, f(x)) &= \hat{\delta}(q_0, f(x[1]) \cdot f(x[2]) \cdots f(x[n])) \\
&= \hat{\delta}(\hat{\delta}(q_0, f(x[1]) \cdot f(x[2]) \cdots f(x[n-1])), f(x[n])) && \text{(Using lemma 2.1)} \\
&= \delta'(\hat{\delta}(q_0, f(x[1]) \cdot f(x[2]) \cdots f(x[n-1])), x[n]) && \text{(Using definition of } \delta') \\
&= \delta'(\hat{\delta}'(q_0, x[1] \cdot x[2] \cdots x[n-1]), x[n]) && \text{(Using } P(n-1) \text{ is true)} \\
&= \hat{\delta}'(q_0, x[1] \cdot x[2] \cdots x[n]) && \text{(Using definition of } \hat{\delta}') \\
&= \hat{\delta}'(q_0, x)
\end{aligned}$$

Hence the induction is established and claim is proved. □

Claim 3.2

$\mathcal{L}(D') = f^{-1}(L)$

We will show this in following two claims

Claim 3.3

$f^{-1}(L) \subseteq \mathcal{L}(D')$

Proof. Let $x \in f^{-1}(L)$, then by definition of f, $f(x) \in L$. Now as $f(x) \in L$, the DFA D recognizes it, that is,

$$\hat{\delta}(q_0, f(x)) \in A$$

From Claim 3.1 this translates to

$$\hat{\delta}'(q_0, x) \in A$$

which implies x is accepted in D', and hence $x \in \mathcal{L}(D')$, which further implies $f^{-1}(L) \subseteq \mathcal{L}(D')$ and hence our claim is proved □

Claim 3.4

$\mathcal{L}(D') \subseteq f^{-1}(L)$

Proof. Let $x \in \mathcal{L}(D')$, then it implies that on running x on DFA D' it ends in A, we have to show that if we run $f(x)$ on D then it should be accepted that it should also end in A,
Now as $x \in \mathcal{L}(D')$ therefore

$$\hat{\delta}'(q_0, x) \in A$$

From Claim 3.1 this translates to

$$\hat{\delta}(q_0, f(x)) \in A$$

which implies $f(x)$ is accepted by D, and hence $f(x) \in L$, which further implies $\mathcal{L}(D') \subseteq f^{-1}(L)$ and hence our claim is proved □

From combining Claim 3.3 and 3.4, we get $\mathcal{L}(D') = f^{-1}(L)$ which implies that D' recognises $f^{-1}(L)$, therefore $f^{-1}(L)$ is regular and one of the DFAs corresponding to it is D'.

Problem 4

Prove that the class of regular languages is closed under homomorphisms. That is, prove that if $L \subseteq \Sigma^*$ is a regular language, then so is $f(L) = \{f(x) \mid x \in L\}$. Here, it is advisable to informally describe how you will turn a DFA for L into an NFA for $f(L)$.

Note

To make the description easier to read, we shall resort to the following abuse of notation:

1. If a string x consists of a single character, we consider it to be equivalent to a single character. In other words, we do not distinguish between Σ^1 and Σ.
2. If a string $x = \epsilon$, then in the context of transitions of an NFA, we consider $(q, x, q'), (q, \epsilon, q')$ to refer to the same thing.

These will be helpful for Case 1 in the following solution.

Solution. Since the language L is regular, there must be a DFA $D = (Q, \Sigma, \delta, q_0, A)$ that recognizes L. Consider the following construction for a potential NFA of the language $f(L)$:

$$N = (Q \cup Q', \Gamma, \Delta, q_0, A)$$

We will define Q' and Δ below
Δ will contain tuples which we can divide in two cases depending upon the alphabets $a \in \Sigma$:

1. Case 1: $f(a) \in \Gamma \cup \epsilon$. In this case add the tuples $\{(q, f(a), \delta(q, a)) \mid q \in Q, a \in \Sigma\}$ to Δ
2. Case 2: $f(a) \notin \Gamma \cup \epsilon$. In this case first divide $f(a)$ into alphabets of Γ i.e. $f(a) = b[1] \cdot b[2] \cdots b[n]$ where $b[i] \in \Gamma$. Now for each transition $(q, a, \delta(q, a)) \in \delta$, create $n-1$ extra (intermediate) states, say $q'_{qa1}, q'_{qa2}, \ldots, q'_{qa(n-1)}$, and add the n tuples $(q, b[1], q'_{qa1})$, $(q'_{qai}, b[i{+}1], q'_{qa(i+1)})$ for $i \in \{1, 2, \ldots, n{-}2\}$ and $(q'_{qa(n-1)}, b[n], \delta(q, a))$ to Δ.

Note: the creation of $n-1$ states will be done for each transition $(q, a, \delta(q, a))$ where $f(a) \notin \Gamma \cup \epsilon$. Q' is the union of all the new states formed in the above Case 2.

Claim 4.1

$f(L) = \mathcal{L}(N)$

We will show that in the following two claims

Claim 4.2

$f(L) \subseteq \mathcal{L}(N)$

Proof. Let $x \in f(L)$ this means there exist a y such that $y \in L$, that is, a run of y on DFA D is accepted. We will use this to show that there exists an accepting run of x on NFA N.
Let the run of y on DFA D be $q_0 y[1] q_1 y[2] \ldots y[n] q_n$ where $y[i] \in \Sigma$ and $y = y[1] \cdot y[2] \cdots y[n]$ as the run is accepted therefore $q[n] \in A$.

$$\begin{aligned} x &= f(y) \\ x &= f(y[1] \cdot y[2] \cdots y[n]) \\ x &= f(y[1]) \cdot f(y[2]) \cdots f(y[n]) \end{aligned}$$

We claim that on running x on N, it will also end in q_n. we will prove this by induction.

Induction Hypothesis $P(k)$: For all $0 \leq k \leq n$, there exists a run of N on $f(y[1] \cdot y[2] \cdot ...y[k])$ on N which ends on q_k.

Base Case When $k = 0$, as the starting state of N is also q_0 so this is trivially true.

Inductive Case Let us assume that $P(k-1)$ is true. Now $f(y[1]\cdot y[2]\cdots y[k]) = f(y[1]\cdot y[2]\cdots y[k-1])\cdot f(y[n])$ using the homomorphism property. Using $P(k-1)$ we know that there exist a run of N on $f(y[1]\cdot y[2]\cdots y[k-1])$ ending on q_{k-1}. Now there are two cases possible for $f(y[n])$ (by a we denote $y[n]$).

1. Case 1: $f(a) \in \Gamma \cup \epsilon$.

   In this case $(q_{k-1}, f(y[k]), q_k) \in \Delta$ as defined above. So we can take this transition and reach q_k.

   More formally, appending $f(y[k]), q_k$ to a run of N on $f(y[1]\cdot y[2]\cdots y[k-1])$ that ends on q_{k-1} gives a run of N on $f(y[1]\cdot y[2]\cdots y[k])$ that ends on q_k.

2. Case 2: $f(a) \notin \Gamma \cup \epsilon$.

   In this case, let $f(a) = b[1]\cdot b[2]\cdots b[m]$ where $b[i] \in \Gamma$.

   As defined above, we have created new states*, say $q'_1, q'_2, \ldots, q'_{m-1}$, such that Δ has tuples $(q_{k-1}, b[1], q'_1)$, $(q'_i, b[i+1], q'_{i+1})$ for $i \in \{1, 2, \ldots, m-2\}$ and $(q'_{m-1}, b[m], \delta(q_{k-1}, a))$.

   Following them, we will reach $q_k = \delta(q_{k-1}, a)$ and hence the run we just constructed ends in q_k.

   More formally, appending $b[1], q'_1, b[2] \ldots, q'_{m-1}, b[m], q_k$ to a run of N on $f(y[1]\cdot y[2]\cdots y[k-1])$ that ends on q_{k-1} gives a run of N on $f(y[1]\cdot y[2]\cdots y[k])$ that ends on q_k.

   Note*: Here the state names are simplified for ease of reading.

As both the above cases are mutually exclusive and exhaustive, $P(k)$ holds when $P(k-1)$ is true. This establishes our induction.

Using the above claim for $k = n$, there exists a run of x on N ends on q_n, which belongs to A, therefore being accepted by N. Hence $x \in \mathcal{L}(N)$, and as x was an arbitrary string in $f(L)$, we have $f(L) \subseteq \mathcal{L}(N)$. □

Claim 4.3

$\mathcal{L}(N) \subseteq f(L)$

Proof. Consider any $x \in \mathcal{L}(N)$. We claim that $x \in f(L)$.

As $x \in \mathcal{L}(N)$ there exists a run $q_0 x[1] q_1 x[2] q_2 x[3] \ldots x[n] q_n$ where $q_n \in A$ and $x = x[1]\cdot x[2]\cdots x[n]$ and $x[i] \in \Gamma$.

As the state space of N is $Q \cup Q'$ and both Q and Q' are disjoint therefore each of the state $q_0, q_1, \ldots, q_n$ can either lie in Q or Q'. Let the indices of states which belong to Q be $r_0, r_1, \ldots, r_l$ and which belong to Q' be $p_1, p_2, \ldots, p_m$, where $l + m = n$, and $r_0 < r_1 < \ldots < r_l$, $p_1 < p_2 < \ldots < p_m$ and as the states q_0 and q_n belongs to Q, so $r_0 = 0$ and $r_l = n$.

We will now claim that $\exists y \in \Sigma^*$, such that $f(y) = x[1]\cdot x[2]\cdots x[n]$ and $\hat{\delta}(q_0, y) = q_n$. We will prove this by induction.

Induction Hypothesis $P(k)$: $\exists y \in \Sigma^*$ such that $f(y) = x[1]\cdot x[2]\cdots x[r_k]$, and $\hat{\delta}(q_0, y) = q_{r_k}$, where $0 \le k \le l$

Base Case When $k = 0$: This is trivially true as $r_0 = 0$, therefore the RHS would be ϵ, so for $y = \epsilon \in \Sigma^*$, since $f(y) = \epsilon$, we have $\hat{\delta}(q_0, f(y)) = \hat{\delta}(q_0, \epsilon) = q_0$.

Inductive Case Let us assume that the induction holds for $k-1$, that is $P(k-1)$ is true. Now there are two cases possible for the number of states between $q_{r_{k-1}}$ and q_{r_k}.

1. Case 1: $r_k - r_{k-1} = 1$. In this case as both the states are in Q, and from the way we have defined Δ, we have $x[r_k] \in \Gamma \cup \epsilon$ and $\exists a \in \Sigma$ such that $f(a) = x[r_k]$. From $P(k-1)$ we have

$\exists y_1 \in \Sigma^*$ such that $f(y_1) = x[1] \cdot x[2] \cdots x[r_{k-1}]$. So we have

$$\begin{aligned} & f(y_1) = x[1] \cdot x[2] \cdots x[r_{k-1}] \wedge f(a) = x[r_k] \\ \implies & f(y_1 \cdot a) = f(y_1) \cdot f(a) \\ & = x[1] \cdot x[2] \cdots x[r_k] \end{aligned}$$

Now as $y_1 \in \Sigma^*$ and $a \in \Sigma$ therefore $y_1 \cdot a \in \Sigma^*$. Also, as $f(a) = x[r_k] \in \Gamma \cup \epsilon$, there exists a transition $(q_{r_{k-1}}, a, q_{r_k}) \in \Delta$, and from the definition of Δ, we have $\delta(q_{r_{k-1}}, a) = q_{r_k}$. So we have

$$\begin{aligned} \hat{\delta}(q_0, y_1 \cdot a) &= \hat{\delta}(\hat{\delta}(q_0, y_1), a) \\ &= \hat{\delta}(q_{r_{k-1}}, a) \\ &= q_{r_k} \end{aligned}$$

Hence in this case the induction hypothesis holds.

2. Case 2: $r_k - r_{k-1} > 1$. In this case all the states between $q_{r_{k-1}}$ and q_{r_k} in the run are in Q', i.e., they are the extra states we created in the definition of Δ.
Now as we have created new states for every transition $(q, a, \delta(q, a))$ where $f(a) \notin \Gamma \cup \epsilon$ and the indegree and outdegree of every new state is 1, there is only one way to go to these states and come out to $\delta(q, a)$, which is when the value of string entered at q is $f(a)$.
Hence, for x to traverse these extra states, it must be the case that $\exists a \in \Sigma$ such that $f(a) = x[r_{k-1} + 1] \cdots x[r_k]$. Also, since $P(k-1)$ is true, there must exist $y \in \Sigma^*$ such that $f(y_1) = x[1] \cdot x[2] \cdots x[r_{k-1}]$. Consider the string $y \cdot a$.

$$\begin{aligned} & f(y_1) = x[1] \cdot x[2] \cdots x[r_{k-1}] \wedge f(a) = x[r_{k-1} + 1] \cdots x[r_k] \\ \implies & f(y_1 \cdot a) = f(y_1) \cdot f(a) \\ & = x[1] \cdot x[2] \cdots x[r_k] \end{aligned}$$

Now as $y_1 \in \Sigma^*$ and $a \in \Sigma$ therefore $y_1 \cdot a \in \Sigma^*$. Now since $f(a) = x[r_{k-1} + 1] \cdots x[r_k]$, the paragraph above the above set of equations implies that there is a transition $\delta(q_{r_{k-1}}, a) = q_{r_k}$.

$$\begin{aligned} \hat{\delta}(q_0, y_1 \cdot a) &= \hat{\delta}(\hat{\delta}(q_0, y_1), a) \\ &= \hat{\delta}(q_{r_{k-1}}, a) \\ &= q_{r_k} \end{aligned}$$

Hence in this case, the induction hypothesis holds true as well.

As both the cases are exhaustive, the inductive step is complete.
Using the induction hypothesis for $k = n$, we get that $\exists y \in \Sigma^*$, such that $f(y) = x[1] \cdot x[2] \cdots x[n] = x$ and $\hat{\delta}(q_0, y) = q_n \in A \implies y \in L$. This implies that $x \in f(L)$. Since x was arbitrary, we have $\mathcal{L}(N) \subseteq f(L)$, as needed. □

By claims 4.2, 4.3, we have that $\mathcal{L}(N) = f(L)$, so N is an NFA that recognizes $f(L)$, implying $f(L)$ is a regular language.

Problem 5

Prove that if $L \subseteq \Sigma^*$ is a regular language then the language $L_0 = \{x \subseteq \Sigma^* \mid x \cdot rev(x) \in L\}$ is also regular, where $rev(x)$ is the reverse of string x. Here, instead of constructing an NFA for L_0 directly, it could be more convenient to use the already proven closure properties. For example, it might be better to write L_0 as a union of a finite collection of languages, and then construct an NFA for each language in that collection.

Note

The motivation for the construction in the following proof comes from the fact that we want to run the DFA both forwards and backwards until both "runs" end on the same state. This can be done if we reverse the second DFA and combine the states in a suitable manner using a cross product.

Also note that we use "runs" in the context of DFAs to make the proof simpler (since there is a natural NFA for each DFA, as proved in class, which has precisely a single run, due to δ being a function).

Solution.
Since L is regular, there must be a DFA $D = (Q, \Sigma, \delta, q_0, A)$ that recognizes it. Consider the NFA $N = (Q, \Sigma, \Delta, A, \{q_0\})$, where $\Delta = \{(q', a, q) \mid \delta(q, a) = q', q \in Q, a \in A\}$, which recognizes $rev(L)$ as done in class.
Now consider the NFA $N' = (Q \times Q, \Sigma, \Delta', \{q_0\} \times A, \{(q, q) \mid q \in Q\})$, where

$$\Delta' = \{((q, q'), a, (q'', q''')) \mid \delta(q, a) = q'' \wedge (q', a, q''') \in \Delta \wedge q, q', q'', q''' \in Q \wedge a \in \Sigma\}$$

We shall show the following two claims, which shall complete the proof:

Claim 5.1
$L_0 \subseteq \mathcal{L}(N')$

Proof. Consider any string x such that $x \cdot rev(x) \in L$. Then we need to construct an accepting run of x on N'.

Suppose $x = x[1]x[2]\dots x[n]$. Then $x \cdot rev(x) = x[1]\dots x[n]x[n]\dots x[1]$. Since L accepts $x \cdot rev(x)$, the run of D on $x \cdot rev(x)$ must be accepting. Suppose the run is $q_0x[1]q_1 \dots x[n]q_nx[n]q_{n+1}x[n-1]\dots q_{2n-1}x[1]q_{2n}$.

Now we claim that $(q_0, q_{2n})x[1](q_1, q_{2n-1})\dots x[n](q_n, q_n)$ is an accepting run of N' on x.

Proof. We shall break the proof into two parts.

1. If it is a run, it is an accepting run: This holds because the last state is (q_n, q_n), which is an accepting state by definition of the set of accepting states of N'.
2. It is a run: This holds because of the following:
   (a) $x = x[1]x[2]\dots x[n]$: This is true by the definition of $x[i]$'s.
   (b) $(q_i, q_{2n-i}) \in Q \times Q$: This is true because $q_i \in Q$ for all i, by definition of D and the run of D on $x \cdot rev(x)$.
   (c) $x[i] \in \Sigma$ for all i: This is true as $x \in \Sigma^*$.
   (d) $((q_i, q_{2n-i}), x[i+1], (q_{i+1}, q_{2n-i-1}))$ is in Δ', which is true because $\delta(q_i, x[i]) = q_{i+1}$ (following from definition of q_i, q_{i+1} in the run of D on x), and $\delta(q_{2n-i-1}, x[i+1]) = q_{2n-i} \implies (q_{2n-i}, x[i+1], q_{2n-i-1}) \in \Delta$.
   (e) (q_0, q_{2n}) is a starting state: This is true because q_{2n} is an accepting state in D, and hence $q_{2n} \in A$, so $(q_0, q_{2n}) \in \{q_0\} \times A)$.

Using these two states, we have shown that this is an accepting run of N' on x, whence we are done. □

We have thus constructed an accepting run of N' on x whenever $x \in L_0$, so we are done. □

Claim 5.2

$\mathcal{L}(N') \subseteq L_0$

Proof. Consider any string $x \in \mathcal{L}(N')$. Then we need to show that $x \cdot rev(x) \in L$. By the definition of x, there exists an accepting run of x on N'.

Suppose $x = x[1]x[2]\dots x[n]$. Note that there is no ϵ transition in N', and thus there are exactly n transitions in an accepting run of x. Also, since an accepting run ends at an accepting state, the accepting state must be of the form (q_n, q_n) for some q_n. Suppose the run is $(q_0, q_{2n})x[1](q_1, q_{2n-1})\dots x[n](q_n, q_n)$ for some q_i's.

Now we claim that $q_0x[1]q_1\dots x[n]q_nx[n]q_{n+1}x[n-1]\dots x[1]q_{2n}$ is the run of the DFA D on $x \cdot rev(x)$, and that D accepts $x \cdot rev(x)$.

Proof. We break the proof into two parts.

1. $q_0x[1]q_1\dots x[n]q_nx[n]q_{n+1}x[n-1]\dots x[1]q_{2n}$ is the run of the DFA D on $x \cdot rev(x)$: Note the following:
   (a) $x[1]x[2]\dots x[n]x[n]\dots x[1] = x \cdot rev(x)$
   (b) $q_i \in Q$ for each i.
   (c) $x[i] \in \Sigma$ for each i.
   (d) q_0 is the starting state of this run, so the starting state of the run works.
   (e) Since $((q_i, q_{2n-i}), x[i], (q_{i+1}, q_{2n-i-1})$ is a transition of N', we must have $\delta(q_i, x[i]) = q_{i+1}$ and $(q_{2n-i}, x[i], q_{2n-i-1}) \in \Delta \implies q_{2n-i} = \delta(q_{2n-i-1}, x[i])$.

   This implies that it is indeed the run of D on $x \cdot rev(x)$.
2. D accepts $x \cdot rev(x)$: Since the final state of the run is q_{2n}, and the starting state of the run of N' on x is (q_0, q_{2n}), by definition of the starting states of N', we have $q_{2n} \in A$, which implies that q_{2n} is an accepting state, from which the conclusion follows.

□

From here, it follows that what we constructed is indeed the run of D on x, and thus for each $x \in \mathcal{L}(N')$, we have $x \in L_0$. □

From these two claims, it follows that $L(N') = L_0$, so the NFA N' accepts L_0, from where it follows that L_0 is a regular language.

Problem 6

Design an algorithm that takes as input the descriptions of two DFAs, D_1 and D_2, and determines whether they recognize the same language.

Solution. To start with, we can assume WLOG that both D_1 and D_2 are DFAs over the same alphabet Σ (for if not, let $\Sigma = \Sigma_1 \bigcup \Sigma_2$ and add appropriate error state transitions in both the DFA).
Now, let the regular language recognized by the DFA D_1 and D_2 be L_1 and L_2 respectively. To design an algorithm which can recognize whether L_1 and L_2 are same we will use following bi-implication from set theory (note that $L_1, L_2 \subseteq \Sigma^*$)

Claim 6.1

$$L_1 = L_2 \iff L_1 \cap L_2^C = \emptyset \text{ and } L_1^C \cap L_2 = \emptyset$$

Where $L^C = \Sigma^* \setminus L$.

Proof. The forward direction follows directly from the definition of set complement, so we shall focus our proof on showing the backwards implication. Specifically we shall prove

$$L_1 \cap L_2^C = \emptyset \text{ and } L_1^C \cap L_2 = \emptyset \implies L_1 = L_2$$

To prove this we prove the following 2 statements.

1. $L_1^C \cap L_2 = \emptyset \implies L_2 \subseteq L_1$. Proof by contradiction. Suppose there exists some $x \in L_2$ such that $x \notin L_1$. But $x \notin L_1 \implies x \in L_1^C$ which therefore implies that $x \in L_1^C \cap L_2$, which is a contradiction.
2. $L_1 \cap L_2^C = \emptyset \implies L_1 \subseteq L_2$. Proof by contradiction. Suppose there exists some $x \in L_1$ such that $x \notin L_2$. But $x \notin L_2 \implies x \in L_2^C$ which therefore implies that $x \in L_1 \cap L_2^C$, which is a contradiction.

Thus, from both the above statements we have $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$ which implies $L_1 = L_2$ and hence the proof is complete. □

This bi-implication lays down the foundation of our algorithm. By closure properties of regular languages we know that they are closed under both complementation and intersection. Further, we already know how to derive DFA for these cases starting from the DFA(s) of the initial languages. Thus, we can obtain the DFAs for the languages $L' = L_1 \cap L_2^C$ and $L'' = L_1^C \cap L_2$. Let's call the DFAs D' and D''.

$$\begin{aligned} L_1 = L_2 &\iff L', L'' = \emptyset \\ &\iff \mathcal{L}(D'), \mathcal{L}(D'') = \emptyset \\ &\iff \text{none of } D', D'' \text{ accepts any string over } \Sigma \end{aligned}$$

For such DFAs, no accepting state can be reachable from the initial state through any sequence of transitions. Thus, to determine if $L_1 = L_2$ we construct the DFAs D' and D'' and verify the set of reachable states from the initial state is disjoint with set of accepting states.

In the algorithm below we define the routine CheckIfSame as the required algorithm. It uses helper routines ComplementDFA, IntersectionDFA and DFS. The briefs of each routine is given alonside the code. The formal proof of correctness of these sub-routines is skipped as they have already been covered in the class.

```
1:                                          ▷ Helper functions for the main function below
2: function ComplementDFA(D)               ▷ Returns the DFA D' such that L(D') = L(D)^C
3:     unpack (Q, Σ, δ, q0, A) ← D
4:     return (Q, Σ, δ, q0, Q \ A)
5: end function
6:
7: function IntersectionDFA(D1, D2)        ▷ Returns the DFA D' such that L(D') = L(D1) ∩ L(D2)
8:     unpack (Q1, Σ, δ1, q10, A1) ← D1
```

```
    unpack (Q_2, Σ, δ_2, q_20, A_2) ← D_2
    function δ((q_1, q_2), a)
        return (δ_1(q_1, a), δ_2(q_2, a))
    end function
    return (Q_1 × Q_2, Σ, δ, (q_10, q_20), A_1 × A_2)
end function

function CheckIfSame(D_1, D_2)
    let D_1^C ← ComplementDFA(D_1)
    let D_2^C ← ComplementDFA(D_2)
    let D' ← IntersectionDFA(D_1, D_2^C)
    let D'' ← IntersectionDFA(D_1^C, D_2)
    unpack (Q', Σ, δ', q'_0, A') ← D'
    unpack (Q'', Σ, δ'', q'_0, A'') ← D''
                                                          ▷ Reduce DFAs to graphs
    let graph G' ← (Q', {(q, q') | ∃a ∈ Σ ∋ δ'(q, a) = q'})
    let graph G'' ← (Q'', {(q, q') | ∃a ∈ Σ ∋ δ''(q, a) = q'})
                          ▷ Assume DFS returns boolean array denoting reachability for each vertex
    let reachable'[1 ... |Q'|] ← DFS(G', q'_0)
    let reachable''[1 ... |Q''|] ← DFS(G'', q''_0)
    for all q ∈ A' do
        if reachable'[q] then
            return False          ▷ An accepting state is reachable in D' implies L_1 ∩ L_2^C ≠ ∅
        end if
    end for
    for all q ∈ A'' do
        if reachable''[q] then
            return False          ▷ An accepting state is reachable in D'' implies L_1^C ∩ L_2 ≠ ∅
        end if
    end for
    return True                   ▷ No accepting state is reachable in either D' or D''
end function

function DFS(G(V, E), v_0)                    ▷ Standard iterative version of DFS algorithm
    let reachable ← boolean array of size |V| initialized to all False.
    let stack ← empty stack
    push v_0 into stack
    reachable[v_0] ← True
    while stack is not empty do
        let v ← pop from stack
        for all u : (v, u) ∈ E do
            if not reachable[u] then
                reachable[u] ← True
                push u into stack
            end if
        end for
    end while
    return reachable
end function
```