Fix an alphabet $\Sigma$.

(1) Show that the set of regular languages over $\Sigma$ is closed under union.

(2) Show that the set of regular languages over $\Sigma$ is closed under intersection.
(1) Suppose $L_1, L_2$ are regular languages over $\Sigma$. Therefore, by definition, there are DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L_1 = L(M_1)$ and $L_2 = L(M_2)$. We use the Cartesian product of the sets of states of the two DFAs to simulate running them in parallel and keeping track of the current state in each machine. That is, we define the (new) DFA $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F)$ where $\delta : (Q_1 \times Q_2) \times \Sigma \rightarrow (Q_1 \times Q_2)$ is

$$\delta((q, q'), x) = (\delta_1(q, x), \delta_2(q', x))$$

for each $(q, q') \in Q_1 \times Q_2$ and $x \in \Sigma$. Moreover, $F = \{(q, q') \in Q_1 \times Q_2 | q \in F_1 \text{ or } q' \in F_2\}$

**Proof of correctness:** we will show that $L_1 \cup L_2 = L(M)$ and hence there is a DFA which recognizes $L_1 \cup L_2$ (thus, $L_1 \cup L_2$ is a regular set). Suppose $w \in L_1 \cup L_2$; we want to show that ______. Without loss of generality, assume $w \in L_1$. Then, since $L_1 = L(M_1)$, the computation of $M_1$ on $w$ ends in an accepting state. Moreover, the computation of $M_1$ on $w$ is exactly the sequence of states in the first component of the states $M$ visits as it computes on $w$ (by definition of the transition function $\delta$). Thus, the first component of the last state in the computation of $M$ on $w$ is in $F_1$. By the definition of $F$, this means that the last state in the computation of $M$ on $w$ is in $F$. Thus, $M$ accepts $w$ and $w \in L(M)$. Conversely, suppose $w \notin L_1 \cup L_2$. By definition of union, this means $w \notin L_1$ and $w \notin L_2$. In particular, the computation of each machine $M_i$ on $w$ ______ in an accepting state. Therefore, the computation of $M$ on $w$ will end in some state $(q, q')$ where $q \in Q_1 \setminus F_1$ and $q' \in Q_2 \setminus F_2$. By definition of $F$, $(q, q') \notin F$ so $M$ ______ $w$, as required.

(2) Very similar to above; can you anticipate what will be different?

Suppose $L_1, L_2$ are regular languages over $\Sigma$. Therefore, by definition, there are DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L_1 = L(M_1)$ and $L_2 = L(M_2)$. We use the Cartesian product of the sets of states of the two DFAs to simulate running them in parallel and keeping track of the current state in each machine. That is, we define the (new) DFA $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F)$ where $\delta : (Q_1 \times Q_2) \times \Sigma \rightarrow (Q_1 \times Q_2)$ is

$$\delta((q, q'), x) = (\delta_1(q, x), \delta_2(q', x))$$

for each $(q, q') \in Q_1 \times Q_2$ and $x \in \Sigma$. Moreover, $F = \{(q, q') \in Q_1 \times Q_2 | q \in F_1 \text{ and } q' \in F_2\}$

**Proof of correctness:** we will show that $L_1 \cap L_2 = L(M)$ and hence there is a DFA which recognizes $L_1 \cap L_2$ so it is a regular set. Suppose $w \in L_1 \cap L_2$; we want to show that ______. By definition of intersection, $w \in L_1$ and $w \in L_2$. Since $L_1 = L(M_1)$, the computation of $M_1$ on $w$ ends in an accepting state. Moreover, the computation of $M_1$ on $w$ is exactly the sequence of states in the first component of the states $M$ visits as it computes on $w$ (by definition of the transition function $\delta$). Thus, the first component of the last state in the computation of $M$ on $w$ is in $F_1$. Similarly, since $L_2 = L(M_2)$, the ______ component in the last state in the computation of $M$ on $w$ is in $F_2$. By the definition of $F$, this means that the last state in the computation of $M$ on $w$ is in $F$. Thus, $M$ accepts $w$ and $w \in L(M)$. Conversely, suppose $w \notin L_1 \cap L_2$. By definition of ______, this means $w \notin L_1$ or $w \notin L_2$. Assume (without loss of generality) that $w \notin L_1$. Then computation of $M_1$ on $w$ does not end in an accepting state. Therefore, the computation of $M$ on $w$ will end in some state $(q, q')$ where $q \in Q_1 \setminus F_1$. By definition of $F$, $(q, q') \notin F$ so $M$ ______ $w$, as required.
(1) **Alternate solution, using NFAs:** Suppose \( L_1, L_2 \) are regular languages over \( \Sigma \). Therefore, by definition, there are DFAs \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) and \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) such that \( L_1 = L(M_1) \) and \( L_2 = L(M_2) \). We will build an NFA that nondeterministically chooses whether to check for membership in \( L_1 \) or in \( L_2 \), and thus will accept any string that is in at least one of these two sets. Then, since each NFA can be simulated by some DFA, we will conclude that \( L_1 \cup L_2 \) is regular. That is, we define the NFA

\[
N = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F)
\]

where we assume (without loss of generality, because states can be relabelled) that \( Q_1 \cap Q_2 = \emptyset \) and that \( q_0 \notin Q_1 \cup Q_2 \). Define \( \delta : Q_1 \cup Q_2 \times \Sigma \rightarrow Q_1 \cup Q_2 \) as

\[
\delta((q,x)) = \begin{cases}
  (q,x) & \text{if } q = q_0, x = \varepsilon \\
  (q,\delta(q,x)) & \text{if } q = q_0, x \in \Sigma \\
  (\delta(q_i,x),i) & \text{if } q \in Q_i, i \in \{1,2\}, x \in \Sigma \\
  (\delta(q_i,x),i) & \text{if } q \in Q_i, i \in \{1,2\}, x = \varepsilon
\end{cases}
\]

for each \( q \in Q_1 \cup Q_2 \cup \{q_0\} \) and \( x \in \Sigma \).

**Proof of correctness:** We will show that \( L_1 \cup L_2 = L(N) \) and hence there is an NFA which recognizes \( L_1 \cup L_2 \) so, since each NFA can be simulated by some DFA, \( L_1 \cup L_2 \) is a regular set. Suppose \( w \in L_1 \cup L_2 \); we want to show that \( w \in L(N) \). Without loss of generality, assume \( w \in L_1 \). Then, since \( L_1 = L(M_1) \), the computation of \( M_1 \) on \( w \) ends in an accepting state. Consider the computation of \( N \) in which the first “move” from the initial state is a spontaneous transition to \( q_1 \). Since \( \delta \) follows the allowed transitions of \( \delta_1 \) for states in \( Q_1 \), the only possible computation path of \( N \) after this initial transition matches the computation of \( M_1 \) on \( w \). Since \( L_1 = L(M_1) \) and \( w \in L_1 \), the computation of \( M_1 \) on \( w \) ends in an accepting state of \( M_1 \). By definition of \( F = F_1 \cup F_2 \), accepting states of \( M_1 \) are also accepting states of \( M \). Thus, there is at least one successful computation of \( N \) on \( w \) and so \( w \in L(N) \). Conversely, suppose \( w \notin L_1 \cup L_2 \). By definition of union, this means \( w \notin L_1 \) and \( w \notin L_2 \). In particular, the computation of each machine \( M_i \) on \( w \) does not end in an accepting state. The only nondeterminism in \( N \) is the presence of the spontaneous transitions from \( q_0 \) to \( q_1 \) and \( q_2 \). Therefore, there are exactly \( \bigcup \) computation paths of \( N \) on \( w \): one that first moves to \( q_1 \) and then follows the computation of \( M_1 \) on \( w \); and one that first moves to \( q_2 \) and then follows the computation of \( M_2 \) on \( w \). Since \( w \notin L(M_1) \) and \( w \notin L(M_2) \), each of these computation paths ends in a state that is not in \( F_1 \) and not in \( F_2 \). Thus, these last states are not in \( F \) and so \( M \) \( \bigcap \) \( w \), as required.

(2) **Alternate solution, using set operations:**

Suppose \( L_1, L_2 \) are regular languages over \( \Sigma \). By closure of the class of regular languages under complementation, \( L_1, L_2 \) are each regular languages too. Moreover, by (1) above (i.e. the closure of the class of regular languages under union), the set

\[
\overline{L_1 \cup L_2}
\]

is regular too. Applying closure under complementation once more, we observe that

\[
\overline{\overline{L_1 \cup L_2}} = \overline{\overline{L_1 \cap L_2}} = \overline{L_1 \cap L_2}.
\]