Assignment #1 Solutions

1. This problem can be solved using the subset construction technique. We will solve the general case (part (b)) first and then use the same idea to solve part (a).

   (b) Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
   Define a DFA $D = (2^Q, \Sigma, \delta_D, \{q_0\}, F_D)$, where
   $F_D = 2^F - \phi$ i.e., all subsets of F except ϕ
   $\delta_D : 2^Q \times \Sigma \to 2^Q$ defined as $\delta_D(S, a) = \bigcup_{r \in S} \delta(r, a)$ for all $S \subseteq Q$ and $a \in \Sigma$.

   Points to note here:

   - The definition of δ_D is the same as in the subset construction technique we studied in class. So we can use the following result covered in class (and in the texts): For all $S \subseteq Q$ and $w \in \Sigma^*$, $\hat{\delta_D}(S, w) = \bigcup_{q \in S} \hat{\delta}(q, w)$.
   - The accepting states of the DFA are all $S \subseteq F$, such that $S \neq \phi$. This is in contrast to the usual subset construction where all S such that $S \cap F \neq \phi$ are chosen as accepting states.

   We now claim that $L(D) = L^{\#}(N)$. To prove this, we show set containment in both directions.

   $\mathbf{L(D) \subseteq L^{\#}(N)}$: Let $w \in L(D)$. By definition of D, this implies $\hat{\delta_D}(\{q_0\}, w) \in F_D$. In other words, $\hat{\delta_D}(\{q_0\}, w)$ is a non-empty subset of F. From the definiton of δ_D, it follows that $\hat{\delta}(q_0, w) \subseteq F$ and $\hat{\delta}(q_0, w) \neq \phi$. Therefore, $w \in L^{\#}(N)$.

   $\mathbf{L^{\#}(N) \subseteq L(D)}$: Let $w \in L^{\#}(N)$. By definition, $\hat{\delta}(q_0, w) \subseteq F$ and $\hat{\delta}(q_0, w) \neq \phi$. Since $\hat{\delta_D}(\{q_0\}, w) = \hat{\delta}(q_0, w)$, it follows that $\hat{\delta_D}(\{q_0\}, w) \in F_D$. Thus, $w \in L(D)$.

   Therefore, $L^{\#}(N)$ is regular.

(a) Let the starting state, accepting state and the remaining state of the NFA be labeled q_0, q_2 and q_1, respectively. We now apply the idea of part (a) to get the following DFA:

	DFA state	0	1
Start state	$\{q_0\}$	$\{q_0, q_2\}$	$\{q_2\}$
	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$
Accepting state	$\{q_2\}$	$\{q_1\}$	$\{q_0, q_2\}$
	$\{q_1\}$	$\{q_0\}$	$\{q_0, q_1\}$
	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

Observe that we do not need to consider all $2^3 = 8$ subsets of $\{q_0, q_1, q_2\}$. The remaining 2 subsets (namely, ϕ and $\{q_1, q_2\}$) cannot be reached from $\{q_0\}$, and hence they need not be considered when constructing the DFA of $L^{\#}(N)$.

2. (a) Let $\tilde{s}^{-1}(L) = L_1$ and let $w \in \tilde{s}(L_1)$. By definition of $\tilde{s}$, there exists an $x \in L_1$ such that $w \in \hat{s}(x)$.

   However, by definition of $\tilde{s}^{-1}$, if $x \in L_1$ (which is $\tilde{s}^{-1}(L)$), then $\hat{s}(x) \subseteq L$. Since $w \in \hat{s}(x)$, it follows that $w \in L$.

   Therefore, $\tilde{s}(L_1) = \tilde{s}(\tilde{s}^{-1}(L)) \subseteq L$.

   (b) Consider $\Sigma_1 = \{0, 1\}$ and $\Sigma_2 = \{a, b, c\}$. Let $s : \Sigma \to 2^{\Sigma_2^*}$ be defined as $s(0) = \{a\}$ and $s(1) = \{b\}$. Let $L = \{a, b, c\}$. Then, $\tilde{s}^{-1}(L) = \{0, 1\}$. However, $\tilde{s}(\tilde{s}^{-1}(L)) = \{a, b\} \neq L$.

   (c) This problem can be solved (and 10 points earned!) simply by inspection. First, note that L is the set of strings in Σ_2^* that end in c. Now consider an arbitrary string $w \in (\mathbf{0} + \mathbf{1})^*$. There are three possibilities:

   - $w = \epsilon$: In this case, $\tilde{s}(w) = \{\epsilon\}$ which is not a subset of L. Therefore, ϵ cannot be in $\tilde{s}^{-1}(L)$.
   - $w \in (\mathbf{0} + \mathbf{1})^*\mathbf{0}$: In other words, w ends in a 0. In this case, $\tilde{s}(w)$ contains strings that end in a or b; the last 0 in w can be substituted by any string from $s(0) = \mathbf{c}^+(\mathbf{a} + \mathbf{b})^*$. Therefore, $\tilde{s}(w)$ is not a subset of L, and hence no such string w can be in $\tilde{s}^{-1}(L)$.

- $w \in (\mathbf{0}+\mathbf{1})^*\mathbf{1}$: In this case, w ends in a 1. Since $s(1) = (\mathbf{a}^*\mathbf{b}^*)^*\mathbf{c}^+ = (\mathbf{a}+\mathbf{b})^*\mathbf{c}^+$, it follows that $\tilde{s}(w)$ is a string in Σ_2^* that ends in a c. Therefore, $\tilde{s}(w) \subseteq L$ for all such strings w.

Since we have considered all possibilities of strings $w \in \Sigma_1^*$, it follows that $\tilde{s}^{-1}(L) = (\mathbf{0}+\mathbf{1})^*\mathbf{1}$. Clearly, $\tilde{s}^{-1}(L)$ is regular.

(d) We will first introduce some notation to help us sail (or paddle, if that's your choice!) through the proof. While the basic idea is simple, we need a rather detailed notation to be mathematically precise.

This proof has been deliberately written out in length to help you understand on the eve of mid-sem exams. Your actual answers could skip the more obvious steps.

Let $M_0 = (Q_0, \Sigma_2, \delta_0, q_{0,1}, F_0)$ be a DFA accepting L (recall that L is a regular language in Σ_2^*). Without loss of generality, let $Q_0 = \{q_{0,1}, \ldots q_{0,m_0}\}$, where $|Q_0| = m_0$. Here, $q_{0,1}$ is the starting state, while the others are non-starting states of M_0.

Let $\Sigma_1 = \{a_1, \ldots, a_n\}$. For each $i \in \{1, 2, \ldots, n\}$, we now define the following automata:

- $M_i = (Q_i, \Sigma_2, \delta_i, q_{i,1}, F_i)$ is a DFA for $s(a_i)$ (recall that each $s(a_i)$ is a regular language in Σ_2^*).
  Without loss of generality, let $Q_i = \{q_{i,1}, \ldots, q_{i,m_i}\}$, where $|Q_i| = m_i$. State $q_{i,1}$ is the starting state, while the others are non-starting states of M_i.
- $P_{i,k} = (Q_i \times Q_0, \Sigma_2, \delta_{i,0}, (q_{i,1}, q_{0,k}), F_i \times Q_0)$ is a product of M_i and M_0. Here, $\delta_{i,0}((u,v),b) = (\delta_i(u,b), \delta_0(v,b))$ for all $u \in Q_i$, $v \in Q_0$ and $b \in \Sigma_2$. There are three points to note in this definition:
  - The transition function $\delta_{i,0}$ is defined in the usual way for product automata.
  - The starting state of $P_{i,k}$ is not the usual pair of starting states, $(q_{i,1}, q_{0,1})$. Instead, it is $(q_{i,0}, q_{0,k})$.
  - The accepting states of $P_{i,k}$ are those which have an accepting state of M_i as their first component.

We are now ready to construct an NFA N for $\tilde{s}^{-1}(L)$. In particular, $N = (Q_0, \Sigma_1, \delta_N, q_{0,1}, F_0)$. Thus, N differs from M_0 only in the definitions of the alphabet and δ_N.

We define δ_N in much the same way as we defined state transitions when proving closure of regular languages under inverse homomorphism. Thus, there exists a transition labeled a_j ($\in \Sigma_2$) from state $q_{0,i}$ to state $q_{0,k}$ whenever there is a string in the language $s(a_j)$ that takes M_0 from $q_{0,i}$ to $q_{0,k}$. This ensures that $\delta_N(q_{0,1}, a_j) = \bigcup_{x \in s(a_j)} \hat{\delta_0}(q_{0,1}, x)$. It then follows from simple induction on the length of a string w that for all $w \in \Sigma_2^*$, $\hat{\delta_N}(q_{0,1}, w) = \bigcup_{x \in \hat{s}(w)} \hat{\delta_0}(q_{0,1}, x)$.

While this seems similar to the construction we used when proving closure under inverse homomorphism, there is a key difference. We could now have infinite strings in the language $s(a_j)$. So, how do we find out if we can insert a transition labeled a_j from $q_{0,i}$ to $q_{0,k}$? This is where the product automata come to our help.

We consider the automaton $P_{j,i}$ with starting state $(q_{j,1}, q_{0,i})$, and determine which of its final states are reachable from this starting state. By definition, each final state of $P_{j,i}$ has a final state of M_j as its first component. However, not all states that have a final state of M_j as its first component may be reachable from $(q_{j,1}, q_{0,i})$. We focus on only those final states that are reachable from the starting state, and pick up their second components to form a set $S_{j,i}$. Formally, $S_{j,i} = \{q_{0,l} \mid \exists q_{j,r} \in F_j$ such that $(q_{j,r}, q_{0,l})$ is reachable from $(q_{j,1}, q_{0,i})$ in $P_{j,i}\}$.

We can now define $\delta_N(q_{0,i}, a_j) = S_{j,i}$ for $i \in \{1, \ldots, m_0\}$ and $j \in \{1, \ldots n\}$.

We claim that $\tilde{s}^{-1}(L) = L^{\#}(N)$. Recall from problem 1 that for an NFA N, $L^{\#}(N)$ is the set of strings w such that $\hat{\delta_N}(q_{0,1}, w) \subseteq F_0$ and $\hat{\delta_N}(q_{0,1}, w) \neq \phi$. For purposes of this problem, we relax the second restriction, allowing $\hat{\delta_N}(q_{0,1}, w)$ to be empty as well. It is easy to see from the solution of problem 1(b) that $L^{\#}(N)$ with this relaxation is still regular (include ϕ in the set of accepting states during the subset construction).

To prove our claim, we show set inclusion in both directions.

$\mathbf{\tilde{s}^{-1}(L) \subseteq L^{\#}(N)}$: Let w be in $\tilde{s}^{-1}(L)$. By definition, $\hat{s}(w) \subseteq L$. This implies that $\bigcup_{x \in \hat{s}(w)} \hat{\delta_0}(q_{0,1}, x) \subseteq F_0$. It follows from the definition of δ_N that $\hat{\delta_N}(q_{0,1}, w) \subseteq F_0$. Therefore, $w \in L^{\#}(N)$.

$L^{\#}(N) \subseteq \mathbf{\tilde{s}^{-1}(L)}$: Let w be in $L^{\#}(N)$. By definition of $L^{\#}(N)$, we have $\hat{\delta_N}(q_{0,1}, w) \subseteq F_0$. By definiton of δ_N, we then have $\bigcup_{x \in \hat{s}(w)} \hat{\delta_0}(q_{0,1}, x) \subseteq F_0$. Therefore, $\hat{s}(w) \subseteq L$. Finally, by definition of $\tilde{s}^{-1}$, it follows that $w \in \tilde{s}^{-1}(L)$.

Since N is an NFA and $L^{\#}(N)$ is regular, it follows that $\tilde{s}^{-1}(L)$ is also regular.

3. As an illustration of different techniques, I will use Pumping Lemma in one case and Myhill-Nerode Theorem in the other. You could use either technique in each subproblem to prove that the corresponding language is non-regular.

   (a) Let's use Myhill-Nerode Theorem in this case. Let $L = \{0^i 1^j | gcd(i, j) = 1\}$. We wish to show that the equivalence relation $\sim_L$ induces an infinite number of equivalence classes. In order to prove this, consider the strings 0^p and 0^q, where p and q are distinct prime numbers. Clearly, the string $z = 1^{p^2}$ when appended to 0^p gives a string that is not in L. However, when z is appended to 0^q, we get a string in L. Thus, there exists a string z such that $0^p.z \notin L$ and $0^q.z \in L$. Therefore, 0^p and 0^q must be in different equivalence classes. Since p and q are arbitrary distinct primes, it follows that the number of equivalence classes is at least as large as the number of primes (countably infinite). It follows that L is not regular.

   (b) Let's use the Pumping Lemma in this case. Let $L = \{0^i 1^j 2^k \mid i < j \text{ or } j < k\}$. Define $L^c = \{0, 1, 2\}^* - L$. Since regular languages are closed under complementation, it suffices to show that L^c is not regular.

   If possible, let L^c be regular. Let $n(\geq 1)$ be the Pumping Lemma constant for L^c. Now consider the string $0^n 1^n 2^n$, which is clearly in L^c. If L^c was indeed regular, we would be able to find x, y, z such that $xyz = 0^n 1^n 2^n$, $|xy| \leq n$ and $|y| \geq 1$. It follows that $y = 0^i$ where $1 \leq i \leq n$. Therefore, $xz = 0^{n-i} 1^n 2^n$. By the Pumping Lemma, if L^c is regular, then $xy^0 z = xz$ should also be in L. However, since $n - i < n$, we know that $xz \in L$ and is therefore $\notin L^c$. Thus, we have shown that L^c doesn't satisfy the Pumping Lemma. It follows that L^c and hence L is not regular.

4. (a) Define $L_1 = (\mathbf{0} + \mathbf{1})^* \mathbf{111}$, $L_2 = (\mathbf{0} + \mathbf{1})^* \mathbf{000}$, and $L_3 = (\mathbf{0} + \mathbf{1})^* \mathbf{000} (\mathbf{0} + \mathbf{1})^*$.

Thus, L_1 (L_2) is the regular language comprised of all strings of 0's and 1's that end in three consecutive 1's (0's). Similarly, L_3 is the regular language comprised of all strings of 0's and 1's that contain three consecutive $0's$.

We claim that the required language is $L = L_3^c \bigcup (L_2.L_1)^+.L_3^c$. To prove this, we show set containment in both directions:

$\mathbf{L \subseteq L_3^c \bigcup (L_2.L_1)^+.L_3^c}$: Let $w \in L$. There are two possibilities for w:

- w does not contain three or more consecutive 0's. Therefore, $w \in L_3^c$.
- w contains three or more consecutive 0's. In this case, we decompose w as $x.y.z$, where:
  - x is the prefix of w that ends in the *last* run of three or more 0's.
  - y is the subsequent part of w that ends in the *last* run of three or more 1's.
  - The remaining part of w (containing no runs of three or more 0's, or three or more 1's) constitutes z.

  Then $x \in L_2$, $y \in L_1$ and $z \in L_3^c$. Therefore, $w \in L_2.L_1.L_3^c$ and hence $\in (L_2.L_1)^+.L_3^c$.

$\mathbf{L_3^c \bigcup (L_2.L_1)^+.L_3^c \subseteq L}$: Let $w \in L_3^c$. It follows from the definition of L that $w \in L$.

Let $w \in (L_2.L_1)^+.L_3^c$. Then w can be written as $x.y.z$ where $x \in (\mathbf{0}+\mathbf{1})^*$, $y \in L_1$, and $z \in L_3^c$. Since y ends in three or more 1's and z does not have any run of three or more 0's, it follows that all runs of three or more 0's in w are followed (not necessarily immediately) by a run of three or more 1's. Therefore, $w \in L$.

We have thus shown that $L = L_3^c \bigcup (L_2.L_1)^+.L_3^c$. Since L_1, L_2, L_3 are regular, so is L.

(b) The required language can be decomposed as $L = \epsilon + \mathbf{0}^*.\mathbf{1}^* + (\mathbf{00})^*.(\mathbf{11})^* + \cdots + (\mathbf{00000})^*.(\mathbf{11111})^*$. It is trivial to see that this is nothing but $\mathbf{0}^*.\mathbf{1}^*$, which is regular.