Computer Science 238 Fall 1998
University of Toronto

Answers to Homework Assignment #5

ANSWER TO QUESTION 1.

(a) This equivalence is true. To prove this, we prove mutual containment of the languages denoted by the two expressions.

$\mathcal{L}\big((R+S)^*\big) \subseteq \mathcal{L}\big((R^*S^*)^*\big)$: Suppose $x \in \mathcal{L}\big((R+S)^*\big)$. Thus, for some $k \in \mathbb{N}$, $x = x_1x_2\cdots x_k$, where $x_i \in \mathcal{L}(R+S)$, for each x_i, $1 \le i \le k$. Therefore, by the definition of $\mathcal{L}(R+S)$, for each $1 \le i \le k$, either $x_i \in \mathcal{L}(R)$ or $x_i \in \mathcal{L}(S)$.

Case 1. $x_i \in \mathcal{L}(R)$. Then $x_i \in \mathcal{L}(R^*)$; since $x_i = x_i \cdot \epsilon$ and $\epsilon \in \mathcal{L}(S^*)$, it follows that $x_i \in \mathcal{L}(R^*S^*)$.

Case 2. $x_i \in \mathcal{L}(S)$. Then $x_i \in \mathcal{L}(S^*)$; since $\epsilon \in \mathcal{L}(R^*)$ and $x_i = \epsilon \cdot x_i$, it follows that $x_i \in \mathcal{L}(R^*S^*)$.

In either case, $x_i \in \mathcal{L}(R^*S^*)$, for $1 \le i \le k$, and therefore $x = x_1x_2\cdots x_k \in \mathcal{L}\big((R^*S^*)^*\big)$.

$\mathcal{L}\big((R^*S^*)^*\big) \subseteq \mathcal{L}\big((R+S)^*\big)$: Suppose $x \in \mathcal{L}\big((R^*S^*)^*\big)$. Thus, for some $k \in \mathbb{N}$, $x = x_1x_2\cdots x_k$, where $x_i \in \mathcal{L}(R^*S^*)$, for each x_i, $1 \le i \le k$. Therefore, by the definition of $\mathcal{L}(R^*S^*)$, for each i such that $1 \le i \le k$, $x_i = y_i^1y_i^2\cdots y_i^{\ell_i}z_i^1z_i^2\cdots z_i^{m_i}$, for some $\ell_i, m_i \in \mathbb{N}$, where $y_i^j \in \mathcal{L}(R)$, for $1 \le j \le \ell_i$, and $z_i^j \in \mathcal{L}(S)$, for $1 \le j \le m_i$. Therefore,

$$x = y_1^1y_1^2\cdots y_1^{\ell_1}z_1^1z_1^2\cdots z_1^{m_1}y_2^1y_2^2\cdots y_2^{\ell_2}z_2^1z_2^2\cdots z_2^{m_2}\cdots\cdots\cdots y_k^1y_k^2\cdots y_k^{\ell_k}z_k^1z_k^2\cdots z_k^{m_k}$$

where each $y_i^j \in \mathcal{L}(R)$ and each $z_i^j \in \mathcal{L}(S)$. Thus, $x \in \mathcal{L}\big((R+S)^*\big)$.

(b) This putative "Kleene closure cancellation law" in invalid. For example, if $R = 0$ and $S = 0^*$, by the idempotence of Kleene closure we have that $R^* \equiv S^*$, yet it is certainly not the case that $R \equiv S$ because, for example, $\epsilon \in \mathcal{L}(S)$, while $\epsilon \notin \mathcal{L}(R)$.

(c) This putative equivalence is not valid. For *any* regular expressions R and S, $\epsilon \in \mathcal{L}\big(((\epsilon+S)R)^*\big)$; but if $\epsilon \notin \mathcal{L}(R)$, then $\epsilon \notin \mathcal{L}\big((R+S)^*R\big)$. Thus, for any regular expression R such that $\epsilon \notin \mathcal{L}(R)$, and any regular expression S, $((\epsilon+S)R)^* \not\equiv (R+S)^*R$.

ANSWER TO QUESTION 2. Let P be the following predicate about regular expressions:

$P(R)$: $R \equiv R'$, for some R' which is either a $\emptyset$-free regular expression or the regular expression $\emptyset$

We use structural induction on R to prove that $P(R)$ holds, for every regular expression R.

Basis: R is the regular expression $\emptyset$, ϵ or a, for some $a \in \Sigma$. The fact that $P(R)$ holds in this case follows immediately from the following two obvious facts: (a) a regular expression is equivalent to itself, and (b) ϵ and a (for any $a \in \Sigma$) are both $\emptyset$-free.

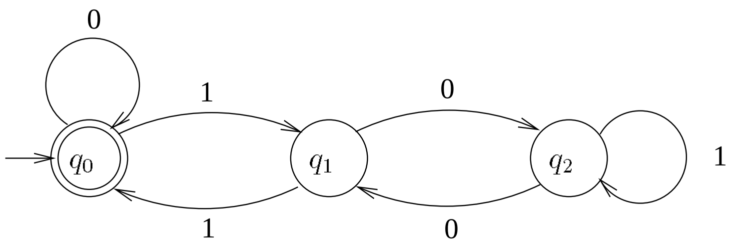
Induction Step: R is constructed from subexpressions S and T. We assume (inductively) that $P(S)$ and $P(T)$ hold. That is, $S \equiv S'$ and $T \equiv T'$, where each of S' and T' is either $\emptyset$-free or the regular expression $\emptyset$. We will prove that $P(R)$ holds as well for every possible way of constructing R out of S and T.

Case 1. $R = S + T$. Since $S \equiv S'$ and $T \equiv T'$, we have that $S + T \equiv S' + T'$. If both S' and T' are $\emptyset$-free, then so is $S' + T'$; in this case, $S + T$ (i.e., R) is equivalent to the $\emptyset$-free regular expression $S' + T'$. If S' is $\emptyset$-free and $T' = \emptyset$, then $S' + T' \equiv S'$; in this case, $S + T$ (i.e., R) is equivalent to the $\emptyset$-free regular expression S'. If $S' = \emptyset$ and T' is $\emptyset$-free, then $S' + T' \equiv T'$; in this case $S + T$ (i.e., R) is equivalent to the $\emptyset$-free regular expression T'. Finally, if $S' = T' = \emptyset$, then $S' + T' \equiv \emptyset$; in this case $S + T$ (i.e., R) is equivalent to $\emptyset$.

Case 2. $R = ST$. Since $S \equiv S'$ and $T \equiv T'$, we have that $ST \equiv S'T'$. If neither one of S' or T' is $\emptyset$, then the regular expression $S'T'$ is $\emptyset$-free, and so ST (i.e., R) is equivalent to the $\emptyset$-free regular expression $S'T'$. Otherwise, $S'T' \equiv \emptyset$, and thus ST (i.e., R) is equivalent to $\emptyset$.

Case 3. $R = S^*$. Since $S \equiv S'$, we have that $S^* \equiv S'^*$. If S' is $\emptyset$-free, then so is S'^*, and so S^* (i.e., R) is equivalent to the $\emptyset$-free regular expression S'^*. Otherwise, $S' = \emptyset$, and thus $S'^* \equiv \epsilon$, and therefore S^* (i.e., R) is equivalent to $\emptyset$-free regular expression ϵ.

ANSWER TO QUESTION 3. A DFSA that accepts L_1 is shown in the diagram below.



For convenience define $\mathbf{int}(\epsilon) = 0$ — i.e., the empty string represents the number 0. From the definition of $\mathbf{int}(x)$ it follows immediately that:

$$\mathbf{int}(xa) = 2 \cdot \mathbf{int}(x) + a, \qquad \text{for any } x \in \{0,1\}^* \text{ and any } a \in \{0,1\} \tag{1}$$

We claim that, for each i such that $0 \leq i \leq 2$, state q_i satisfies the following invariant:

$$\delta^*(q_0, x) = q_i \quad \text{if and only if} \quad \mathbf{int}(x) \bmod 3 = i$$

In other words, string x takes the DFSA from the initial state to q_i if and only if i is the remainder of the division by 3 of the number represented in binary by x. We now prove that this state invariant does, indeed, hold.

Let P be the following predicate on strings:

$$P(x): \qquad \delta^*(q_0, x) = q_i, \text{ if } \mathbf{int}(x) \bmod 3 = i$$

We prove that $P(x)$ holds for every string x by structural induction on x.

Basis: $x = \epsilon$. We have $\delta^*(q_0, x) = q_0$, and $\mathbf{int}(x) \bmod 3 = 0 \bmod 3 = 0$, as wanted.

Induction Step: $x = ya$, for some $y \in \{0,1\}^*$ and $a \in \{0,1\}$. Suppose, by induction, that $P(y)$ holds. That is, $\delta^*(q_0, y) = q_i$ if $\mathbf{int}(y) \bmod 3 = i$. We have

$$
\begin{aligned}
\delta^*(q_0, x) &= \delta^*(q_0, ya) && \text{[since } x = ya\text{]} \\
&= \delta(\delta^*(q_0, y), a)) && \text{[by def. of } \delta^*\text{]} \\
&= \begin{cases} \delta(q_0, a), & \text{if } \mathbf{int}(y) \bmod 3 = 0 \\ \delta(q_1, a), & \text{if } \mathbf{int}(y) \bmod 3 = 1 \\ \delta(q_2, a), & \text{if } \mathbf{int}(y) \bmod 3 = 2 \end{cases} && \text{[by induction hypothesis]} \\
&= \begin{cases} \delta(q_0, a), & \text{if } \mathbf{int}(x) \bmod 3 = a \\ \delta(q_1, a), & \text{if } \mathbf{int}(x) \bmod 3 = (2+a) \bmod 3 \\ \delta(q_2, a), & \text{if } \mathbf{int}(x) \bmod 3 = 1 + a \end{cases} && \text{[by (1), and arithmetic mod 3]} \quad (2)
\end{aligned}
$$

There are two cases, depending on whether a is 0 or 1.

CASE 1. $a = 0$. Then,

$$
\begin{aligned}
\delta^*(q_0, x) &= \begin{cases} \delta(q_0, 0), & \text{if } \mathbf{int}(x) \bmod 3 = 0 \\ \delta(q_1, 0), & \text{if } \mathbf{int}(x) \bmod 3 = 2 \\ \delta(q_2, 0), & \text{if } \mathbf{int}(x) \bmod 3 = 1 \end{cases} && \text{[by substituting } a = 0 \text{ in (2)]} \\
&= \begin{cases} q_0, & \text{if } \mathbf{int}(x) \bmod 3 = 0 \\ q_2, & \text{if } \mathbf{int}(x) \bmod 3 = 2 \\ q_1, & \text{if } \mathbf{int}(x) \bmod 3 = 1 \end{cases} && \text{[by inspection of the transition diagram]}
\end{aligned}
$$

CASE 2. $a = 1$. Then,

$$
\begin{aligned}
\delta^*(q_0, x) &= \begin{cases} \delta(q_0, 1), & \text{if } \mathbf{int}(x) \bmod 3 = 1 \\ \delta(q_1, 1), & \text{if } \mathbf{int}(x) \bmod 3 = 0 \\ \delta(q_2, 1), & \text{if } \mathbf{int}(x) \bmod 3 = 2 \end{cases} && \text{[by substituting } a = 1 \text{ in (2)]} \\
&= \begin{cases} q_1, & \text{if } \mathbf{int}(x) \bmod 3 = 1 \\ q_0, & \text{if } \mathbf{int}(x) \bmod 3 = 0 \\ q_2, & \text{if } \mathbf{int}(x) \bmod 3 = 2 \end{cases} && \text{[by inspection of the transition diagram]}
\end{aligned}
$$

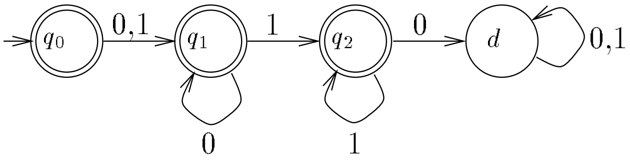
So, in both cases, $\delta^*(q_0, x) = q_i$ if $\mathbf{int}(x) \bmod 3 = i$. This completes the proof that $P(x)$ holds for every string x.

Using this we can now complete the proof that our DFSA is correct. Let x be any string in L_1. Thus, $\mathbf{int}(x) \bmod 3 = 0$. Since $P(x)$ holds, $\delta^*(q_0, x) = q_0$, and since q_0 is an accepting state, every string in L_1 is accepted by the automaton.

Conversely, let x be any string not in L_1. Thus, $\mathbf{int}(x) \bmod 3 \neq 0$. Since $P(x)$ holds, $\delta^*(q_0, x) \neq q_0$, and therefore $\delta^*(q_0, x) \in \{q_1, q_2\}$. Since neither q_1 nor q_2 is an accepting state, x is not accepted by the automaton.

Therefore the language accepted by the automaton in L_1, as wanted.

ANSWER TO QUESTION 4. A DFSA that accepts L_2 is shown in the diagram below.



Next we show that this automaton does, indeed accept L_2. By inspection of the diagram it is easy to verify that:

(i) For any state q, $\delta^*(q, 010) = \delta^*(q, 110) = d$. In other words, from any state, the forbidden substrings take the automaton to the dead state.

(ii) For any string x such that $\delta^*(s, x) = d$, if y is the shortest prefix of x such that $\delta^*(s, y) = d$, then the last three symbols of y are either 010 or 110.

Let x be an arbitrary string that is not in L_2. By (i) it follows that $\delta^*(s, x) = d$. Since d is a nonaccepting state, x is not accepted by the automaton. Therefore, every string that is not in L_2 is not accepted by the automaton.

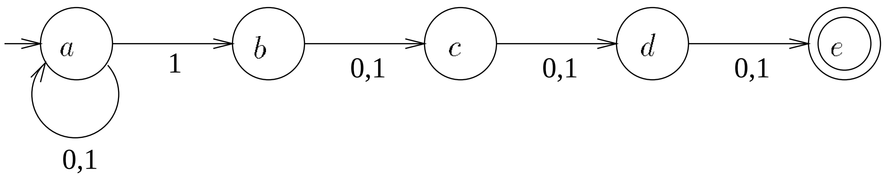
Conversely, let x be an arbitrary string that is not accepted by the automaton. Since d is the only nonaccepting state, $\delta^*(s, x) = d$. By (ii), x has a prefix y that ends in either 010 or 110. Hence, x contains 010 or 110 as a substring, which means that x is not in L_2. Therefore, every string that is not accepted by the automaton is not in L_2.

We have therefore shown that the language accepted by the automaton is L_2.

ANSWER TO QUESTION 5.

(a) $(0+1)^*1(0+1)(0+1)(0+1)$.

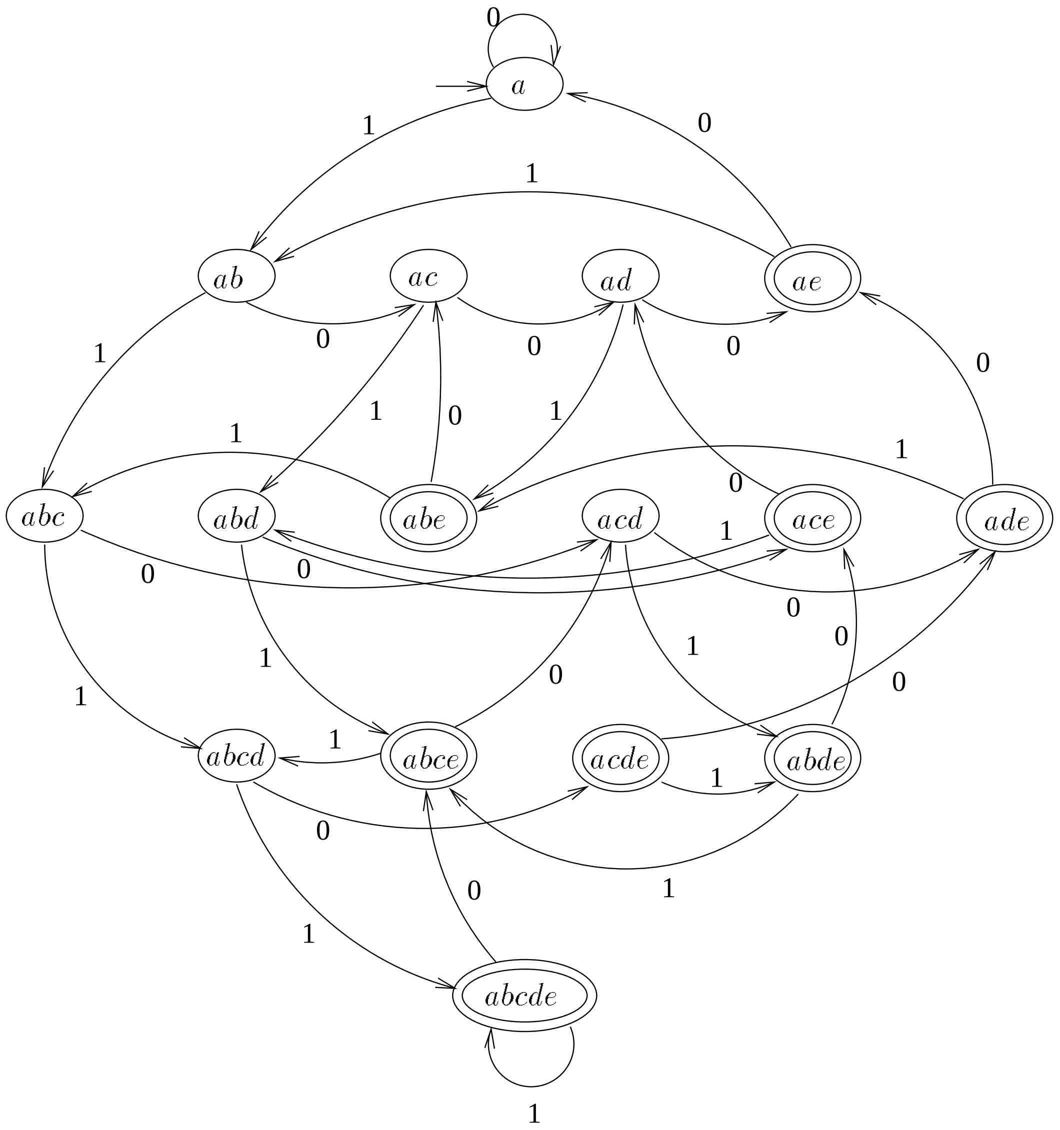
(b) The NFSA shown below accepts L_3.



(c) After pruning the unreachable states, the FSA resulting from the subset construction is shown on the next page.

(d) Let $M = (Q, \{0,1\}, \delta, q_0, F)$ be any deterministic FSA that accepts L_3. Let x, y be any two distinct strings of length 4. We claim that $\delta^*(q_0, x) \neq \delta^*(q_0, y)$. Suppose, for contradiction that $\delta^*(q_0, x) = \delta^*(q_0, y)$. Since $x \neq y$, the two strings differ in at least one position. Let i be the last position where x and y differ, where $1 \leq i \leq 4$. Let z be any string of length exactly $i-1$. One of xz and yz is in L_3 and the other is not, because exactly one of the two has 1 in the fourth position from the end. However, since $\delta^*(q_0, x) = \delta^*(q_0, y)$, we also have that $\delta^*(q_0, xz) = \delta^*(q_0, yz)$. Hence, either M accepts both xz and yz, or it rejects both xz and yz. Thus, M does not accept L_3 since it either accepts a string it should reject, or rejects a string it should accept.

Therefore, any deterministic FSA that accepts L_3 must have at least as many states as there are strings of length four in $\{0,1\}^*$. Since there are $2^4 = 16$ such strings, we get the desired result. The deterministic FSA that accepts L_3 constructed in part (c) (shown below) has only 16 states.



ANSWER TO QUESTION 6. Informally, we can obtain an automaton $M_{\mathbf{min}}$ that accepts $L_{\mathbf{min}}$ by disallowing all transitions from the accepting states of an automaton M that accepts L.

More precisely, we construct $M_{\mathbf{min}}$ as follows. Let $M = (Q, \Sigma, \delta, s, F)$ be a DFSA that accepts L. (We can assume that M is deterministic without loss of generality because we have shown that a language accepted by a nondeterministic automaton is also accepted by a deterministic one.) Let d be a state that is not in Q, and define the DFSA $M_{\mathbf{min}} = (Q \cup \{d\}, \Sigma, \delta_{\mathbf{min}}, s, F)$, where, for any $q \in Q \cup \{d\}$ and any $a \in \Sigma$,

$$\delta_{\mathbf{min}}(q, a) = \begin{cases} \delta(q, a), & \text{if } q \notin F \cup \{d\} \\ d, & \text{if } q \in F \cup \{d\} \end{cases}$$

First we will prove some properties of $M_{\mathbf{min}}$. Then we will use these properties to prove that $M_{\mathbf{min}}$ accepts $L_{\mathbf{min}}$.

Property 1. *Let q be an arbitrary state in Q. For any $x \in \Sigma^*$, if, for every proper prefix z of x, $\delta^*(q, z) \notin F$, then $\delta^*_{\mathbf{min}}(q, x) = \delta^*(q, x)$.*

PROOF. Let P be the following predicate about strings:

$$P(x): \qquad \text{if, for every proper prefix } z \text{ of } x,\ \delta^*(q,z) \notin F, \text{ then } \delta^*_{\mathbf{min}}(q,x) = \delta^*(q,x)$$

We will use structural induction on x to prove that $P(x)$ holds for every string x.

BASIS: $x = \epsilon$. The antecedent of the conditional statement is trivially true because ϵ has no proper prefixes, and the consequent is also true because, by the definition of extended transition function, $\delta^*_{\mathbf{min}}(q,\epsilon) = \delta^*(q,\epsilon) = q$. Thus, $P(\epsilon)$ holds.

INDUCTION STEP: $x = ya$, for some $y \in \Sigma^*$ and some $a \in \Sigma$. We assume, by induction, that $P(y)$ holds. We must prove that $P(x)$ holds as well.

Suppose that for every proper prefix z of x, $\delta^*(q,z) \notin F$ (otherwise, $P(x)$ holds trivially). In particular, (i) for every proper prefix z of y, $\delta^*(q,z) \notin F$, and (ii) $\delta^*(q,y) \notin F$. By (i) and the induction hypothesis, $\delta^*_{\mathbf{min}}(q,y) = \delta^*(q,y)$. From this and (ii), $\delta^*_{\mathbf{min}}(q,y) \notin F \cup \{d\}$. (Recall that d is not a state of M, so since $\delta^*_{\mathbf{min}}(q,y) = \delta^*(q,y)$, $\delta^*_{\mathbf{min}}(q,y)$ cannot be d.) We have:

$$\begin{aligned}
\delta^*_{\mathbf{min}}(q,x) &= \delta^*_{\mathbf{min}}(q,ya) && [\text{because } x = ya] \\
&= \delta_{\mathbf{min}}(\delta^*_{\mathbf{min}}(q,y),a) && [\text{by definition of extended transition function}] \\
&= \delta(\delta^*_{\mathbf{min}}(q,y),a) && [\text{by definition of } \delta_{\mathbf{min}}, \text{ since } \delta^*_{\mathbf{min}}(q,y) \notin F \cup \{d\}] \\
&= \delta(\delta^*(q,y),a) && [\text{because } \delta^*_{\mathbf{min}}(q,y) = \delta^*(q,y), \text{ as shown above}] \\
&= \delta^*(q,ya) && [\text{by definition of extended transition function}] \\
&= \delta^*(q,x) && [\text{because } x = ya]
\end{aligned}$$

Therefore, $P(x)$ holds, as wanted. This completes the proof of Property 1.

Property 2. *Let q be an arbitrary state in Q. For any $x \in \Sigma^*$, if $\delta^*_{\mathbf{min}}(q,x) \neq d$, then $\delta^*(q,z) \notin F$, for every proper prefix z of x.*

PROOF. We will prove the contrapositive. Assume that $\delta^*(q,z) \in F$, for some proper prefix z of x. We will prove that $\delta^*_{\mathbf{min}}(q,x) = d$.

Without loss of generality, we may assume that for every proper prefix z' of z, $\delta^*(q,z') \notin F$ (otherwise, we can pick an even shorter prefix z' of x with the desired property). Therefore, by Property 1, $\delta^*_{\mathbf{min}}(q,z) = \delta^*(q,z)$. Since $\delta^*(q,z) \in F$, it follows that $\delta^*_{\mathbf{min}}(q,z) \in F$. Since z is a proper prefix of x, there is some $a \in \Sigma$ and $z' \in \Sigma^*$ so that $x = zaz'$. Because $\delta^*_{\mathbf{min}}(q,z) \in F$, it follows from the definition of $\delta_{\mathbf{min}}$ that $\delta_{\mathbf{min}}(\delta^*_{\mathbf{min}}(q,z),a) = d$, and therefore that $\delta^*_{\mathbf{min}}(q,zaz') = d$. Hence, $\delta^*_{\mathbf{min}}(q,x) = d$.

This completes the proof of Property 2.

We are now ready to prove that $L_{\mathbf{min}} = \mathcal{L}(M_{\mathbf{min}})$, by proving mutual containment of the two sides.

$$\begin{aligned}
x \in L_{\mathbf{min}} &\Rightarrow x \in L, \text{ and } z \notin L, \text{ for every proper prefix } z \text{ of } x \\
&\Rightarrow \delta^*(s,x) \in F, \text{ and } \delta^*(s,z) \notin F, \text{ for every proper prefix } z \text{ of } x \\
&\Rightarrow \delta^*(s,x) \in F, \text{ and } \delta^*_{\mathbf{min}}(s,x) = \delta^*(s,x) \quad [\text{by Property 1}] \\
&\Rightarrow \delta^*_{\mathbf{min}}(s,x) \in F \\
&\Rightarrow x \in \mathcal{L}(M_{\mathbf{min}})
\end{aligned}$$

$$
\begin{aligned}
x \in \mathcal{L}(M_{\mathbf{min}}) &\Rightarrow \delta^*_{\mathbf{min}}(s,x) \in F \\
&\Rightarrow \delta^*_{\mathbf{min}}(s,x) \neq d && \text{[since } d \notin F\text{]} \\
&\Rightarrow \delta^*(s,z) \notin F, \text{ for every proper prefix } z \text{ of } x && \text{[by Property 2]} \\
&\Rightarrow \delta^*_{\mathbf{min}}(s,x) = \delta^*(s,x), \text{ and} && \text{[by Property 1]} \\
&\ \delta^*(s,z) \notin F, \text{ for every proper prefix } z \text{ of } x \\
&\Rightarrow \delta^*(s,x) \in F, \text{ and} && \text{[since } \delta^*_{\mathbf{min}}(s,x) \in F\text{]} \\
&\ \delta^*(s,z) \notin F, \text{ for every proper prefix } z \text{ of } x \\
&\Rightarrow x \in L, \text{ and } z \notin L, \text{ for every proper prefix } z \text{ of } x \\
&\Rightarrow x \in L_{\mathbf{min}}
\end{aligned}
$$

This completes the proof that $\mathcal{L}(M_{\mathbf{min}}) = L_{\mathbf{min}}$.

THAT'S IT WITH HOMEWORK SOLUTIONS, FOLKS!