

Solutions for Homework Assignment #5

Answer to Question 1.

a. This is false. Here is a counterexample: Let $R = 0$ and $S = 1$. Clearly, $01 \in \mathcal{L}((R + S)^*)$ while $01 \notin \mathcal{L}(R^* + S^*)$. Therefore $(R + S)^* \neq R^* + S^*$.

b. This is false. Here is a counterexample: Let $R = 0^*$, $S = 0$ and $T = (0 + 00)$. We have,

$$\begin{aligned} \mathcal{L}(RT) &= \{0^{n+1} : n \in \mathbb{N}\} \cup \{0^{n+2} : n \in \mathbb{N}\} \\ &= \{0^{n+1} : n \in \mathbb{N}\} \\ &= \mathcal{L}(RS) \end{aligned}$$

and so $RT \equiv RS$. Also, $R \neq \emptyset$. However, $S \neq T$, (because $\mathcal{L}(S) = \{0\}$ while $\mathcal{L}(T) = \{0, 00\}$).

c. This is true. For any string x , $x = \epsilon \circ x \circ \epsilon$. So, if $x \in \mathcal{L}((R + S)^*)$ then $x \in \mathcal{L}(R^*(R + S)^*S^*)$ (because ϵ is in $\mathcal{L}(R^*)$ and $\mathcal{L}(S^*)$). Conversely, if $x \in \mathcal{L}(R^*(R + S)^*S^*)$ then $x \in \mathcal{L}((R + S)^*(R + S)^*(R + S)^*)$ (because the languages denoted by R^* and S^* are subsets of the language denoted by $(R + S)^*$), and so $x \in \mathcal{L}((R + S)^*)$. Therefore $(R + S)^* \equiv R^*(R + S)^*S^*$.

Answer to Question 2.

a. Let $L = \{0^m 1^n : m \cdot n \geq 3\}$. The automaton below accepts L .

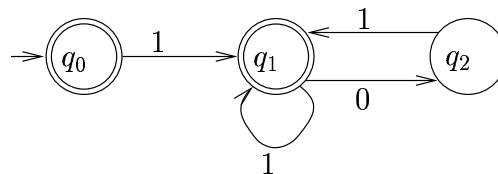


To prove the correctness of the automaton we can prove (by induction on $|x|$) that for any string x of the form $0^m 1^n$ that takes the automaton from its initial state to some state q , the values of m and n for that string satisfy the invariant written inside q .

L is denoted by the regular expression $01111^* + 00111^* + 0000^*11^*$. To see why, note that L is the union of three sets of strings: those that start with a single 0 followed by at least three 1s, denoted by 01111^* ; those that start with two 0s followed by at least two 1s, denoted by 00111^* ; and those that start with at least three 0s followed by at least one 1, denoted by 0000^*11^* .

b. Let $L' = \{x \in \{0, 1\}^* : \text{every } 0 \text{ in } x \text{ is immediately preceded and followed by a } 1\}$

Following is a DFSA M' that accepts L' .



To prove that M' accepts L' we first note that L' consists of the set of strings that:

- do not start with 0;
- do not end with 0; and
- do not contain 00 as a substring.

(A string that violates one of these properties contains a 0 that is not surrounded by 1s. Conversely, if x contains a 0 that is not surrounded by 1s, this 0 is the first or last symbol in x , or it is preceded or followed by another 0. In other words x violates one of the above three properties.)

By inspection of the transition diagram, M' rejects a string x if and only if:

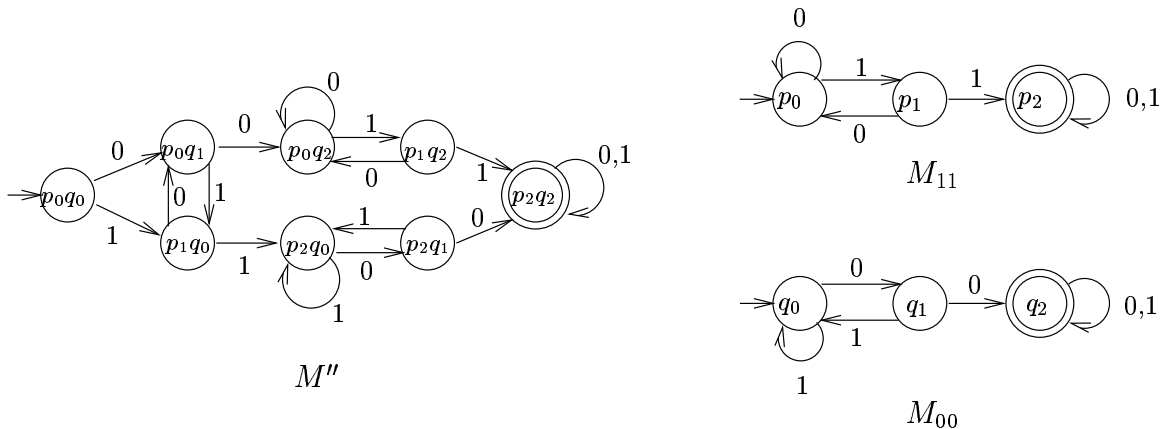
- M' is in q_0 and the next symbol is 0. This means that x starts with 0. (Because there is no transition into q_0 , so M' is in this state q_0 *only* at the start.)
- M' is in state q_1 , and the last symbol of x is 0.
- At some point M' is in state q_2 and the next symbol is 0. This means that the previous symbol seen by M' was 0 (because the only transition into q_2 is on 0). Therefore x contains 00 as a substring.

In other words, M' rejects x if and only if x is not in L' . Therefore M' accepts L .

A regular expression that denotes L' is $\epsilon + 1(1^*01)^*1^*$. To see why, note that every nonempty string in L' begins with 1, is followed by zero or more “blocks” each of which starts with some number (possibly zero) of 1s and ends with 01, and ends with a (possibly empty) string of 1s.

c. Let $L'' = \{x : \text{both } 00 \text{ and } 11 \text{ appear as substrings of } x\}$

Shown below (on the left) is a DFSA M'' that accepts L'' .



To see that M'' accepts L'' , observe that M'' is the Cartesian product of two DFSAs denoted M_{11} and M_{00} shown on the right-hand side of the above figure. M_{11} accepts the strings that contain 11 as a substring, and M_{00} accepts the strings that contain 00 as a substring — thus, their Cartesian product accepts the strings that contain both 00 and 11 as substrings. To prove that M_{11} accepts the set of strings that contain 11 as a substring we can prove by induction on the length of $x \in \{0, 1\}^*$ (left as an exercise) that x takes the DFSA from the initial state to

- p_0 iff $x = \epsilon$ or x does not contain 11 as a substring and ends in 0,
- p_1 iff x does not contain 11 as a substring and ends in 1,
- p_2 iff x contains 11 as a substring.

Given this, M_{11} accepts the strings that contain 11 as a substring since p_2 is its only accepting state. A similar argument shows that M_{00} accepts the strings that contain 00 as a substring.

A regular expression that denotes L'' is $(0+1)^*00(0+1)^*11(0+1)^* + (0+1)^*11(0+1)^*00(0+1)^*$. To see why, note that L'' is the union of two sets of strings: those that contain an occurrence of 00 before an occurrence of 11 and those that contain an occurrence of 11 before an occurrence of 00 (note that these two sets of strings are not disjoint).

Answer to Question 3. Let $M = (Q, \Sigma, \delta, s, F)$ be a DFSA that accepts L .

a. We describe a NFSA M_p that accepts **Prefix**(L). Let F_p be the states in Q from which some accepting state of M is reachable. More precisely, $F_p = \{q \in Q : \text{for some } y \in \Sigma^*, \delta^*(q, y) \in F\}$. We claim that the DFSA $M_p = (Q, \Sigma, \delta, s, F_p)$ accepts **Prefix**(L). This can be proved as follows:

$$\begin{aligned} x \in \mathcal{L}(M_p) &\Leftrightarrow \delta^*(s, x) \in F_p \\ &\Leftrightarrow \text{for some } y \in \Sigma^*, \delta^*(\delta^*(s, x), y) \in F \\ &\Leftrightarrow \text{for some } y \in \Sigma^*, \delta^*(s, xy) \in F \\ &\Leftrightarrow \text{for some } y \in \Sigma^*, xy \in L \\ &\Leftrightarrow x \in \mathbf{Prefix}(L) \end{aligned}$$

b. We describe a NFSA M^r that accepts **Reversal**(L). M^r is obtained from M by

- (i) reversing the direction of each transition;
- (ii) adding a new start state s^r with ϵ transitions to each of M 's accepting states; and
- (iii) making M 's start state M^r 's unique accepting state.

More formally, let s^r be a state that does not appear in Q ; the new automaton is $M^r = (Q \cup \{s^r\}, \Sigma, \delta^r, s^r, \{s\})$, where δ^r is defined as follows:

$$\delta^r(q, a) = \begin{cases} \{q' \in Q : \delta(q', a) = q\}, & \text{if } q \in Q \text{ and } a \in \Sigma \\ F, & \text{if } q = s^r \text{ and } a = \epsilon \\ \emptyset, & \text{if } q \in Q \text{ and } a = \epsilon, \text{ or } q = s^r \text{ and } a \in \Sigma \end{cases}$$

It is easy to see that M accepts x if and only if M^r accepts the reversal of x . Since M accepts L , M^r accepts **Reversal**(L).

c. We describe a NFSA $M_a = (Q_a, \Sigma, \delta_a, s_a, F_a)$ that accepts **Insert** _{a} (L). Intuitively, M_a consists of two copies of M , which for convenience we will call M^1 and M^2 . M_a has the following transitions in addition to those dictated by M : From any state of M^1 , on input a , the automaton can jump to the corresponding state of M^2 . Thus, M_a is nondeterministic because from each state q of M^1 there are two transitions on input a : one leading to some state of M^1 (as dictated by the transition function of M) and one leading to the second copy of state q in M^2 . The start state of M_a is the start state of M^1 and the accepting states of M_a are the accepting states of M^2 .

Thus, a string x is accepted by M_a if and only if it initially follows a computation path from the start state of M^1 to some state q of M^1 , then jumps on input a to the copy of q in M^2 and proceeds from there to an accepting state. In other words, x is accepted by M_a if and only if there exist strings y and z such that $x = yaz$ and yz is accepted by M . (Intuitively, y is the portion of x in which the computation follows transitions in M^1 , a is the inserted symbol that causes the jump from M^1 to M^2 , and z is the portion of x in which the computation follows transitions in M^2 .) More formally,

- The set of states of M_a is $Q_a = Q \times \{1, 2\}$. (Intuitively, $(q, 1)$ corresponds to the state q in the first copy of M and $(q, 2)$ corresponds to the state q in the second copy of M .)
- The start state of M_a is $s_a = (s, 1)$ (the start state of the first copy of M).

- The set of accepting states of M_a is $F_a = F \times \{2\}$ (the accepting states of the second copy of M).
- The nondeterministic transition function δ_a of M_a is defined as follows:

$$\begin{aligned} \delta_a((q, i), b) &= \{(\delta(q, b), i)\}, & \text{for all } q \in Q, i \in \{1, 2\}, b \in \Sigma, \text{ except } i = 1 \text{ and } b = a \\ \delta_a((q, 1), a) &= \{(\delta(q, a), 1)\} \cup \{(q, 2)\}, & \text{for all } q \in Q \end{aligned}$$

The first of these equations says that transitions within each copy of M are as dictated by the transition function of that automaton, except for transitions from states in the first copy on input a . The second equation says that from each state of the first copy of M there are two transitions on a : one to the appropriate state in the first copy of M and one to the corresponding state of the second copy of M .

Given this construction we can prove that M_a accepts $\mathbf{Insert}_a(L)$ or, equivalently, that

$$x \in \mathcal{L}(M_a) \Leftrightarrow x \in \mathbf{Insert}_a(L)$$

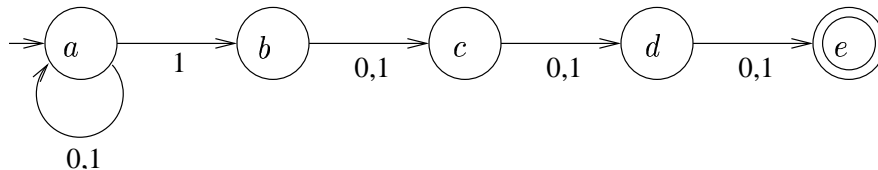
[\Rightarrow] Suppose $x \in \mathcal{L}(M_a)$. This means that in M_a there is a computation path from $(s, 1)$ to $(f, 2)$ on input x , for some $f \in F$. Since the only transitions in M_a from states of the first copy of M to states of the second copy of M are transitions from $(q, 1)$ to $(q, 2)$ on input a , and since there are no transitions from states of the the second copy of M to states of the first copy of M , it follows that the above computation path in M_a consists of: (a) a computation path from $(s, 1)$ to some $(q, 1)$ on input y , (b) a single transition from $(q, 1)$ to $(q, 2)$ on input a , and (c) a computation path from $(q, 2)$ to $(f, 2)$ on input z , where y and z are strings such that $x = yaz$. By construction of δ_a , $\delta^*(s, y) = q$ and $\delta^*(q, z) = f$. Thus, $\delta^*(s, yz) \in F$ which means that $yz \in L$. But since $x = yaz$, $x \in \mathbf{Insert}_a(L)$.

[\Leftarrow] Suppose $x \in \mathbf{Insert}_a(L)$. By definition, there are $y, z \in \Sigma^*$ such that $x = yaz$ and $yz \in L$. Let $q = \delta^*(s, y)$ and $f = \delta^*(q, z)$. Since $yz \in L$, $f \in F$. By construction of δ_a , in M_a there is (a) a computation path from $(s, 1)$ to $(q, 1)$ on input y , (b) a transition from $(q, 1)$ to $(q, 2)$ on input a , and (c) a computation path from $(q, 2)$ to $(f, 2)$ on input z . Therefore in M_a there is a computation path from $(s, 1)$ to $(f, 2)$ on input yaz . Since $(f, 2) \in F_a$ and $yaz = x$, M_a has an accepting computation path on input x . Thus, $x \in \mathcal{L}(M_a)$.

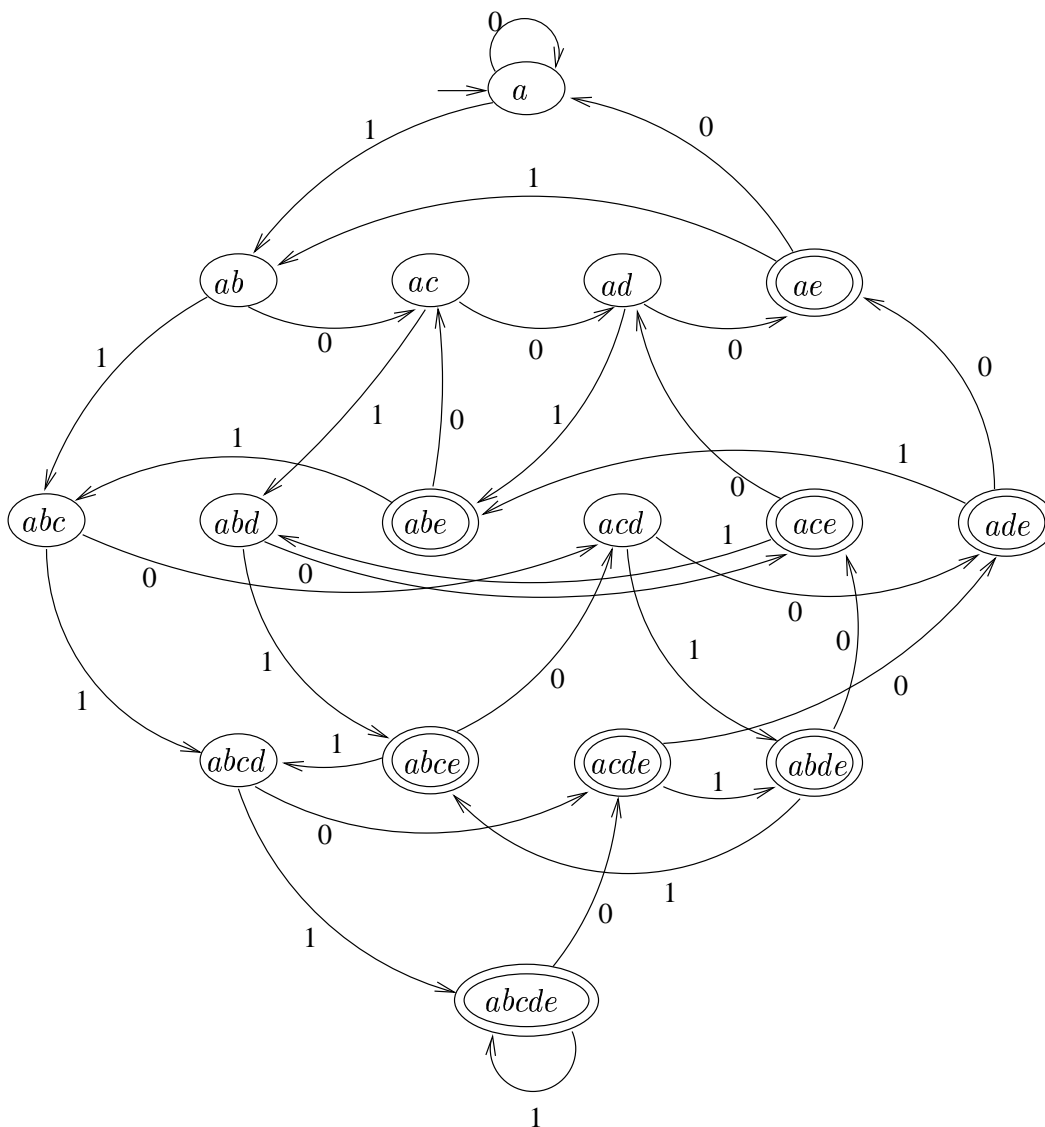
Answer to Question 4.

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

b. The NFSA shown below accepts L_4 .



c. After pruning the unreachable states, the FSA resulting from the subset construction is:



d. Let $M = (Q, \{0, 1\}, \delta, q_0, F)$ be any deterministic FSA that accepts L_4 . 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_4 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_4 since it either accepts a string it should reject, or rejects a string it should accept.

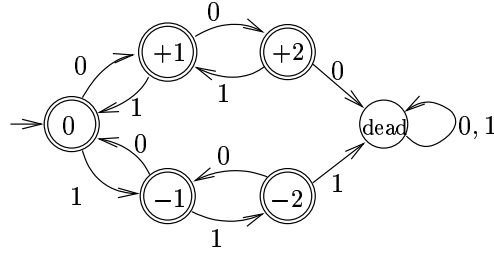
Therefore, any deterministic FSA that accepts L_4 must have at least as many states as there are strings of length four in $\{0, 1\}^*$. Since there are $2^4 = 16$ such strings, we get the desired result. The deterministic FSA that accepts L_3 constructed in part (c) has only 16 states.

e. Let $L_n = \{x \in \{0, 1\}^* : x = y1z, \text{ for some } y, z \in \{0, 1\}^* \text{ such that } |z| = n - 1\}$. That is, L_n is the set of strings with at least n symbols, where the n th symbol from the end is 1.

We can easily generalise the NFA given in part (b) to construct a NFA with $n + 1$ states that accepts L_n . Also, we can generalise the proof in part (d) to show that every DFA that accepts L_n has 2^n states. We do this by proving (just as in part (d)) that in any DFA that accepts L_n , distinct input strings of length n must take the automaton (from the start state) to *distinct* states.

Answer to Question 5.

a. The diagram below is a DFSA that accepts L .



Intuitively the state of the DFSA keeps track of whether the string x processed so far is in L and, if so, the actual (not absolute) difference between the number of 0s and 1s. More precisely, let $\mathbf{zero}(x)$ be the number of 0s in x and $\mathbf{one}(x)$ be the number of 1s in x . The states of the automaton satisfy the following invariants:

$$\delta^*(0, x) = \begin{cases} i, & \text{if } x \in L \text{ and } \mathbf{zero}(x) - \mathbf{one}(x) = i, \text{ for all } i \text{ such that } -2 \leq i \leq 2 \\ \text{dead}, & \text{if } x \notin L \end{cases}$$

This can be shown by a straightforward induction on x . Since all states except the one labeled “dead” are accepting, the above invariant implies that $\delta^*(0, x)$ is accepting if and only if $x \in L$. Thus the DFSA accepts L .

b. Assume, for contradiction, that L' is regular. By the Pumping Lemma there is some n such that for every string $x \in L'$ such that $|x| \geq n$, there are $u, v, w \in \{0, 1\}^*$ such that (a) $x = uvw$, (b) $v \neq \epsilon$, (c) $|uv| \leq n$, and (d) for every $k \in \mathbb{N}$, $uv^k w \in L'$. Let $x = 0^n 1^n$. Obviously $x \in L'$ and $|x| \geq n$. By (a), (b) and (c), $u = 0^i$, $v = 0^j$, where $i \geq 0$ and $j \geq 1$. So, $uvw = 0^i 0^j 0^{n-i-j} 1^n$ and therefore, for all $k \in \mathbb{N}$, $uv^k w = 0^{n+(k-1)j} 1^n$. Therefore, for $k = 4$, we have that $uv^4 w = 0^{n+3j} 1^n$ and since $j \geq 1$, $uv^4 w \notin L'$, contrary to (d).