

Solutions for Assignment #4

Answer to Question 1.

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

$\mathcal{L}((R + S)^*) \subseteq \mathcal{L}((R^*S^*)^*)$: Suppose $x \in \mathcal{L}((R + S)^*)$. Thus, for some $k \in \mathbb{N}$, $x = x_1x_2 \cdots x_k$, where $x_i \in \mathcal{L}(R)$ or $x_i \in \mathcal{L}(S)$, for each i such that $1 \leq i \leq k$. For every string y , $y = y \cdot \epsilon$ and $y = \epsilon \cdot y$. Furthermore, for every regular expression T , $\epsilon \in \mathcal{L}(T^*)$, and if $y \in \mathcal{L}(T)$ then $y \in \mathcal{L}(T^*)$. Thus, since $x_i \in \mathcal{L}(R)$ or $x_i \in \mathcal{L}(S)$, we have that $x_i \in \mathcal{L}(R^*S^*)$, and therefore $x \in \mathcal{L}((R^*S^*)^*)$.

$\mathcal{L}((R^*S^*)^*) \subseteq \mathcal{L}((R + S)^*)$: Suppose $x \in \mathcal{L}((R^*S^*)^*)$. 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 , $1 \leq i \leq k$. Therefore, by the definition of $\mathcal{L}(R^*S^*)$, for each i such that $1 \leq i \leq k$, $x_i = y_i^1y_i^2 \cdots y_i^{\ell_i}z_i^1z_i^2 \cdots z_i^{m_i}$, for some $\ell_i, m_i \in \mathbb{N}$, where $y_i^j \in \mathcal{L}(R)$, for $1 \leq j \leq \ell_i$, and $z_i^j \in \mathcal{L}(S)$, for $1 \leq j \leq m_i$. So x is the concatenation of strings, each of which is in $\mathcal{L}(R)$ or $\mathcal{L}(S)$, and therefore $x \in \mathcal{L}((R + S)^*)$.

b. In general, this equivalence is false. For example, let $R = 0$ and $S = 1$. Clearly, $01 \in \mathcal{L}((R + S)^*)$ while $01 \notin \mathcal{L}(R^* + S^*)$. Therefore $(R + S)^* \neq R^* + S^*$.

c. This equivalence is true. The left-hand side is equivalent to $(R(S + \epsilon))^*R$ and the right-hand side is equivalent to $R((S + \epsilon)R)^*$. Let's denote the regular expression $(S + \epsilon)$ by T . Therefore, we have to show that $(RT)^*R \equiv R(TR)^*$. If $x \in \mathcal{L}((RT)^*R)$ then $x = x_1y_1x_2y_2 \dots x_ky_kx_{k+1}$ for some $k \in \mathbb{N}$, where each $x_i \in \mathcal{L}(R)$, $1 \leq i \leq k + 1$, and each $y_j \in \mathcal{L}(T)$, $1 \leq j \leq k$. Therefore, each $y_ix_{i+1} \in \mathcal{L}(TR)$, for $1 \leq i \leq k$, and so, $x \in \mathcal{L}(R(TR)^*)$.

Similarly, if $x \in \mathcal{L}(R(TR)^*)$ then $x = x_0y_1x_1y_2x_2 \dots x_ky_k$ for some $k \in \mathbb{N}$, where each $x_i \in \mathcal{L}(R)$, $0 \leq i \leq k$, and each $y_j \in \mathcal{L}(T)$, $1 \leq j \leq k$. Therefore, each $x_iy_{i+1} \in \mathcal{L}(TR)$, $0 \leq i < k$, and so, $x \in \mathcal{L}((RT)^*R)$.

Answer to Question 2. The DFSA whose diagrams are shown in (a) and (b) of Figure 1 accept the languages L and L' , respectively.

To prove that the DFSA (a) accepts L we can show, by a straightforward induction on the length of $x \in \{0, 1\}^*$, that a string x takes the DFSA from the initial state to

- q_0 iff $x = \epsilon$,
- q_1 iff x does not have 00 or 11 as a substring and ends in 0,
- q_2 iff x does not have 00 or 11 as a substring and ends in 1,
- q_3 iff x has 00 or 11 as a substring.

The desired result follows immediately from this since q_0 , q_1 and q_2 are the only accepting states, and the strings that take the FSA to these states are precisely the ones in the language we want to accept.

A regular expression that denotes L is $(01)^*(0 + \epsilon) + (10)^*(1 + \epsilon)$. To see why, we observe that a string that doesn't contain either 00 or 11 as a substring must consist of an alternating sequence of 0s and 1s, starting with either 0 or 1. The first term of the regular expression denotes alternating sequences of 0s and 1s starting with 0, and the second term denotes alternating sequences of 0s and 1s starting with 1.

DFSA (b) accepts L' because it is the Cartesian product of the two DFSA shown in (c). The top of these accepts the strings that contain 11 as a substring, and the bottom accepts the strings that contain 00 as a substring. Thus their Cartesian product recognises the strings that contain both 00 and 11 as

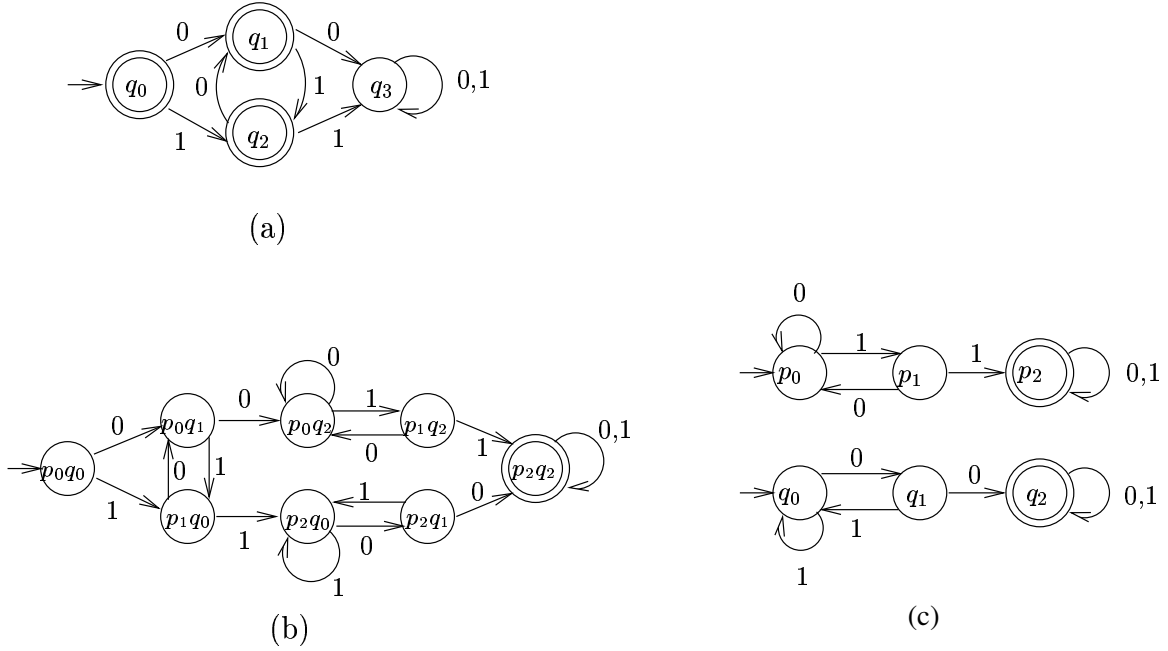


Figure 1: DFSA for Question 2

substrings. To prove that the FSA at the top of (c) accepts the set of strings that contain 11 as a substring we can prove, by induction on the length of $x \in \{0,1\}^*$, 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.

The DFSA accepts the strings that contain 11 as a substring since p_2 is its only final state. A similar argument shows that the FSA at the bottom accepts the strings that contain 00 as a substring.

A regular expression that denotes L' is $(0+1)^*00(0+1)^*11(0+1)^* + (0+1)^*11(0+1)^*00(0+1)^*$. The first term denotes strings that contain both required substrings where some 00 appears before some 11, while the second term denotes strings that contain both required substrings where some 11 appears before some 00.

Answer to Question 3.

a. By definition, $\mathbf{Odd}(L) = L \cap \{x \in \Sigma^* : |x| \text{ is odd}\}$. The language $\{x \in \Sigma^* : |x| \text{ is odd}\}$ is obviously accepted by a DFSA (with just two states). (Alternatively, we can observe that this language is denoted by the regular expression $(a_1 + \dots + a_n)((a_1 + \dots + a_n)(a_1 + \dots + a_n))^*$, where $\Sigma = \{a_1, \dots, a_n\}$, and so it is accepted by a FSA.) So $\mathbf{Odd}(L)$ is the intersection of two languages each of which is accepted by a FSA, so $\mathbf{Odd}(L)$ is itself accepted by a FSA.

b. It is easy to see that $L \bowtie L' = (L \circ L')^*$: If $x \in L \bowtie L'$, there are $y_1, y_2, \dots, y_k \in L$ and $y'_1, y'_2, \dots, y'_k \in L'$ so that $x = y_1 y'_1 y_2 y'_2 \dots y_k y'_k$. Each $y_i y'_i \in L \circ L'$, and so $x = y_1 y'_1 y_2 y'_2 \dots y_k y'_k \in (L \circ L')^*$. Conversely, if $x \in (L \circ L')^*$, there are $z_1, z_2, \dots, z_k \in L \circ L'$ so that $x = z_1 z_2 \dots z_k$. But since each $z_i \in L \circ L'$, there are $y_i \in L$ and $y'_i \in L'$ so that $z_i = y_i y'_i$. So, $x = y_1 y'_1 y_2 y'_2 \dots y_k y'_k$, i.e., $x \in L \bowtie L'$. The result now follows immediately from the fact that FSA-accepted languages are closed under concatenation and under the Kleene star.

Answer to Question 4.

a. The diagram of a NFSA that accepts L_3 is shown in Figure 2. Intuitively, it starts in state 0 and nondeterministically “guesses” if 1, 2 or 3 occurs an odd number of times in the input string, jumping to states 1, 2 or 3 accordingly. It then alternates between state i and state \bar{i} , for $i = 1, 2, 3$, keeping track of the parity of the number of occurrences of i so far.

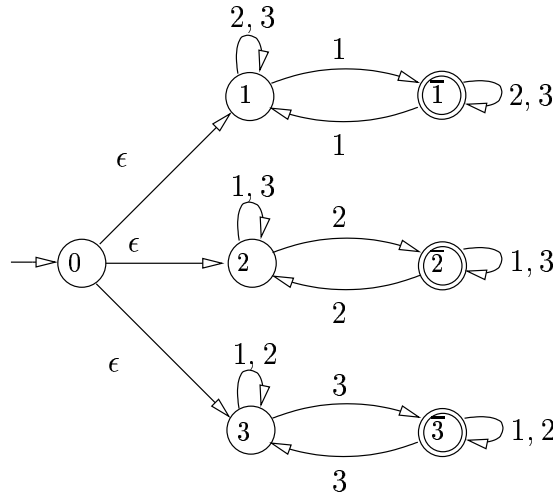


Figure 2: NFSA for Question 4(a)

b. In Figure 3 we show the diagram of the DFSA that results from the above NFSA if we apply the subset construction. Actually, not quite: If we had followed the subset construction precisely, we would have obtained a nine-state DFSA, with an additional state, labeled 0123, which would be the automaton’s initial state. From this state, the automaton would move to $\bar{1}23$, $1\bar{2}3$ or $12\bar{3}$, depending on whether the first symbol of the input string is 1, 2 or 3, respectively. The rest of the automaton would be as shown below. However, it is easy to see that state 0123 merely duplicates the role of 123 initially. (The role of these states is to “remember” that the portion of the input string “consumed” so far contains an even number of 1s, 2s and 3s.) We show the simplified DFSA, where we combine these two states into one.

c. Let $a \in \{1, 2, 3\}$ and $x, x' \in \{1, 2, 3\}^*$ be such that x and x' differ (at least) in the parity of the number of occurrences of a . Without loss of generality, assume that a occurs an odd number of times in x and an even number of times in x' . Let y be any string in $\{1, 2, 3\}^*$ consisting of one occurrence of each symbol in $\{1, 2, 3\}$ that occurs an odd number of times in x . Note that y contains one occurrence of a (and possibly of other symbols). Obviously $xy \notin L_3$ (since every symbol occurs an even number of times in xy) while $x'y \in L_3$ (since at least a occurs an odd number of times in $x'y$). Thus, if s is the start state and δ is the transition function of any DFSA that accepts L_3 , $\delta^*(s, xy) \neq \delta^*(s, x'y)$ (since the left hand side is a non-accepting state, while the right hand side is an accepting state). Therefore, $\delta^*(s, x) \neq \delta^*(s, x')$.

Any two distinct strings in the set $\{\epsilon, 1, 2, 3, 12, 13, 23, 123\}$ differ in the parity of the number of occurrences of at least one symbol. By the above result, they must take any DFSA that accepts L_3 from the start state to distinct states. Thus, any DFSA that accepts L_3 must have at least eight states.

d. An NFSA with $2n + 1$ states that accepts L_n has states $\{0\} \cup \{i, \bar{i} : 1 \leq i \leq n\}$. It starts in 0, and has ϵ -transitions to each state i , $1 \leq i \leq n$. For each i such that $1 \leq i \leq n$, state i has self-transitions on all symbols other than i and a transition to \bar{i} on symbol i ; and state \bar{i} has self-transitions on all symbols other than i and a transition to i on symbol i .

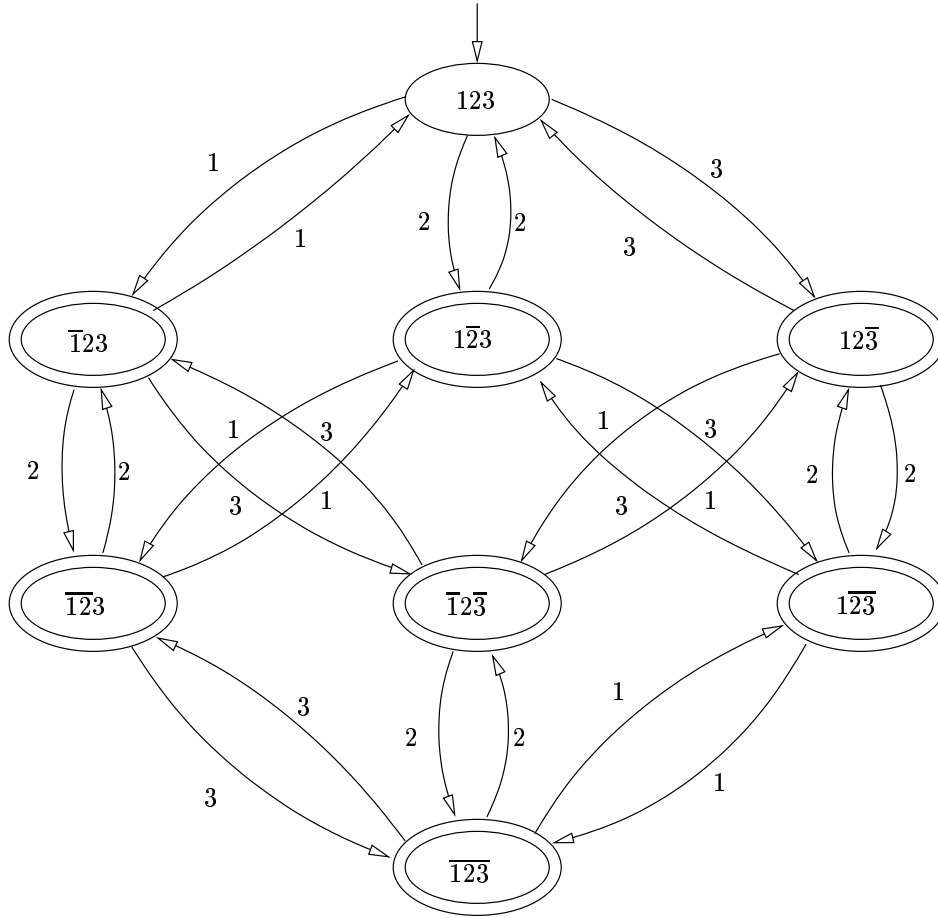


Figure 3: DFSA for Question 4(b)

The intuition is similar as for the special case $n = 3$ in part (a) above: The NFSA nondeterministically guesses which of the symbols occurs an odd number of times, and makes an ϵ -transition to the corresponding symbol. It then verifies that the symbol occurs an odd number of times.

The proof that no DFSA with fewer than 2^n states accepts L_n is a generalisation of the proof in part (c) above: We prove that if two strings differ in the parity of the number of occurrences of some symbol, then these two strings must take the DFSA to distinct states. (Otherwise, by suitable continuation of the two strings, the DFSA would be accepting a string that is not in L_n .) The result now follows by observing that there are 2^n possible combinations of choices for the parities of the n symbols.