1. For each of the following languages, describe an equivalent regular expression, and ***briefly explain in English*** why your regular expression is correct. Unless specified otherwise, all languages are over the binary alphabet $\Sigma = \{0, 1\}$.

Rubric: 2½ points for each part = 1½ for correctness + 1 for English explanation (standard regular expression rubric, scaled and rounded). Every subproblem has infinitely many correct solutions!

(a) All strings that start with the substring 11, end with the substring 11, and have even length.

Solution: Call this language L_a. We have $11 \in L_a$, and every other string in L_a has the form $11x11$ where x is even length. Therefore, L_a is represented by

$$11 + 11((0+1)(0+1))^* 11.$$

■

(b) All strings that do not contain the *subsequence* 0010.

Solution: Call this language L_b. We'll partition the strings w in L_b by whether they contain zero, one, or at least two 0s. In the last case, we can have an arbitrary number of 0s starting with the second one, but as soon as a 1 appears, no more 0s can appear.

Language L_b is represented by

$$\begin{aligned} &1^* && \#(0,w) = 0 \\ &+ 1^*01^* && \#(0,w) = 1 \\ &+ 1^*01^*00^*1^* && \#(0,w) \geq 2 \end{aligned}$$

or more compactly

$$1^*(\varepsilon + 0)1^*0^*1^*.$$

In hindsight, we can interpret the second regular expression as expressing *after* the *first* 0, you can have at most one non-empty maximal substring of 0s. ■

(c) All strings that do not contain the *substring* 011.

Solution: Call this language L_c. After the first (possibly empty) substring of 0s, every individual 1 must either be the last symbol in the string or be immediately followed by a (non-empty) run of 0s.

Language L_c is represented by

$$\underbrace{1^*}_{\text{optional run of 1s}} \cdot \underbrace{0^*}_{\text{oh no 0s}} \cdot \Big(\underbrace{100^*}_{\text{each 1 followed by a run of 0s}} \Big)^* \cdot \Big(\underbrace{\varepsilon + 1}_{\text{optional 1 to end}} \Big).$$

■

(d) All strings in which each substring 01 is immediately followed by an odd-length run of 0s.

Solution: Call this language L_d. After the first (possibly empty) substring of 0s, each 1 should be followed by an odd-length run of 0s. Such a run ends with a 0, meaning either the string is complete or the next symbol is a 1 again followed by an odd-length run of 0s.

Language L_d is represented by

$$\underbrace{1^*}_{\text{optional run of 1s}} \cdot \underbrace{0^*}_{\text{oh no 0s}} \cdot \Big(\underbrace{\varepsilon}_{\text{no more string}} \underbrace{+}_{\text{or}} \underbrace{(10(00)^*)^*}_{\text{each 1 followed by run of 0s}} \Big).$$

■

*(e) All strings ***over the alphabet {0, 1, 2, 3}*** in which every pair of adjacent symbols differs by exactly 1.

Solution: Let's call this language L_e. Just like in part (d), we'll use 1s as signposts; the definition of L_e constrains what substrings can appear before, after, and between 1s. Our regular expression has the following high-level structure:

$$L_e = \langle \textit{no 1s} \rangle \;+\; \langle \textit{before first 1} \rangle \cdot \big(1 \cdot \langle \textit{between 1s} \rangle\big)^* \cdot 1 \cdot \langle \textit{after last 1} \rangle$$

The substring between two consecutive 1s is either a single 0 or a substring of alternating 2s and 3s that begins and ends with 2.

$$\langle \textit{between 1s} \rangle \;=\; 0 + 2(32)^*$$

The prefix before the first 1 is either empty, a single 0, or a string of alternating2s and 3s that *ends* with 2.

$$\langle \textit{before first 1} \rangle \;=\; 0 + (32)^* + (23)^*2$$

The suffix after the last 1 is either empty, a single 0, or a string of alternating 2s and 3s that *begins* with 2.

$$\langle \textit{after last 1} \rangle \;=\; 0 + (23)^* + 2(32)^*$$

Finally, a string in L_e with no 1s at all is either empty, a single 1, or alternating 2s and 3s.

$$\langle \textit{no 1s} \rangle \;=\; 0 + (23)^* + (32)^* + (23)^*2 + (32)^*3$$

Putting all the pieces together, we obtain our final regular expression:

$$\begin{aligned} L_e = {} & 0 + (23)^* + (32)^* + (23)^*2 + (32)^*3 \\ & + \big(0 + (32)^* + (23)^*2\big) \cdot \big(1 \cdot (0 + 2(32)^*)\big)^* \cdot 1 \cdot \big(0 + (23)^* + 2(32)^*\big) \end{aligned}$$

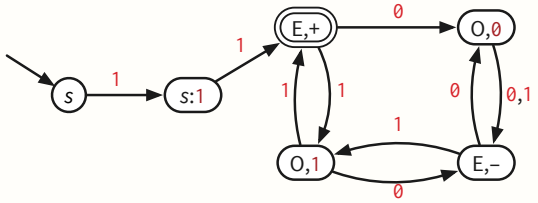
■

2. For each of the following languages over the binary alphabet $\Sigma = \{0, 1\}$, describe a DFA that accepts the language, and ***briefly explain in English*** the purpose of each state. You can describe your DFA using a drawing, formal mathematical notation, or a product construction; see the standard DFA rubric.

Rubric: 10 points = 2½ points for each part = 1½ for correctness + 1 for English explanation of states (standard DFA rubric, scaled and rounded). Again, every subproblem has infinitely many correct solutions. These solutions are more detailed than necessary for full credit.

(a) All strings that start with the substring 11, end with the substring 11, and have even length.

Solution:



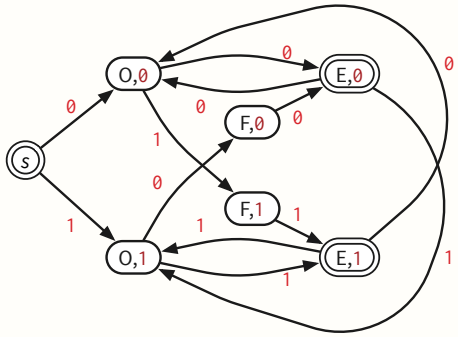
All missing transitions lead to a hidden dump state. States are labeled as follows:

- State s is the start state; no symbols have been read.
- s:1 — The first input symbol was 1.
- O,a — The first two input symbols were 11, we've read an Odd number of symbols, and the previous input symbol was a.
- E,+ — The first two input symbols were 11, we've read an Even number of symbols, and the previous two input symbols were 11.
- E,– — The first two input symbols were 11, we've read an Even number of symbols, and the previous two input symbols were *not* 11.

■

(b) All strings in which each odd-length run is immediately followed by a run of length at least 2.

Solution:



All missing transitions lead to a hidden dump state. States are labeled as follows:

- State s is the start state; no symbols have been read.
- C,a — The previous input symbol was a, and it was the first symbol Following an odd-length run of its complementary symbol.
- E,a — The previous input symbol was a, and its run has had even length up to this point.
- O,a — The previous input symbol was a, its run has had odd length up to this point, and it is *not* the first symbol following an odd-lenght run of its complementary symbol.

■

(c) All strings of length at least 8 whose last eight symbols contain exactly three 1s.

Solution: We formally define a DFA (Q, δ, s, A) over the alphabet $\Sigma = \{0, 1\}$ as follows; the notation Σ^n denotes any string over Σ of length exactly n.

$$Q = \{w \in \Sigma^* \mid |w| \le 8\}$$

$$\delta(w,a) = \begin{cases} wa & \text{if } |w| < 8 \\ xa & \text{if } w = bx \text{ for some } b \in \Sigma \text{ and } x \in \Sigma^7 \end{cases}$$

$$s = \varepsilon$$

$$A = \left\{ w \in \Sigma^8 \;\middle|\; \#(1, w) = 3 \right\}$$

Each state stores the last eight symbols read or all symbols read if fewer than 8 symbols have been read. ■

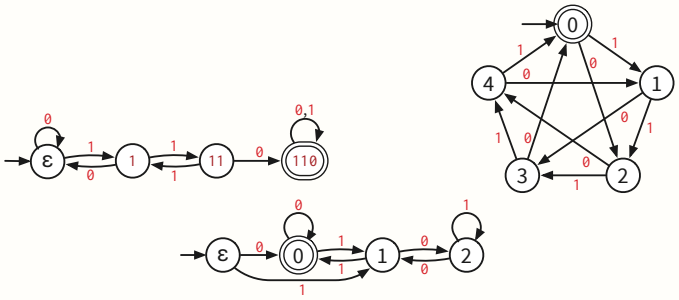
(d) All strings w that satisfy *exactly two* of the following conditions:

i. w contains the substring 110

ii. $|w| + \#(0, w)$ is a multiple of 5

iii. w represents a multiple of 3 in binary

Solution (product construction): We construct the product M of *three* DFAs.



- The first DFS M_1 recognizes all strings that contain the substring 110. The first three states are labeled with the longest suffix of the bits read so far that is also a prefix of 110; only strings that do not contain 110 can reach these states.
- The second DFA M_2 recognizes strings w such that $|w| + \#(0, w)$ is a multiple of 5. Each state is labeled with the value of $|w| + \#(0, w)$ mod 5, where w is the prefix of bits read so far.
- The third DFA M_3 recognizes binary representations of multiples of 3. Each state other than the start state stores the binary value modulo 3 of the bits read so far.

The accepting states of our particular three-way product DFA are all states (q_1, q_2, q_3) such that exactly two of q_1, q_2, q_3 are accepting states in their respective machines. More formally, we have

$$A = \big((A_1 \times A_2 \times Q_3) \cup (A_1 \times Q_2 \times A_3) \cup (Q_1 \times A_2 \times A_3)\big) \setminus (A_1 \times A_2 \times A_3)$$

The product of *any* three DFAs $M_1 = (Q_1, s_1, A_1, \delta_1)$, $M_2 = Q_2, s_2, A_2, \delta_2)$, and $M_3 = (Q_3, s_3, A_3, \delta_3)$ is defined as one would expect:

$$Q = Q_1 \times Q_2 \times Q_3$$
$$s = (S_1, s_2, s_3)$$
$$\delta((q_1, q_2, q_3), a) = \big(\delta_1(q_1, a), \delta_2(q_1, a), \delta_3(q_1, a)\big)$$

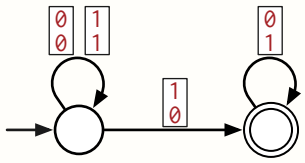
Equivalently, the three-way product is the product of M_1 with the product of M_2 and M_3. The accepting states of the three-way product depend on the target language, just as they do for products of two machines. ■

Rubric: No penalty for interpreting ε as a valid binary representation of 0 or for interpreting later bits as having *higher* significance than those before. No explanation of the three-way product construction is required, but the solution must explicitly describe the accepting states.

3. ***Practice only. Do not submit solutions.***

(a) Describe a DFA that accepts the language $L_{+1} = \{w \in \Sigma_2^* \mid hi(w) = lo(w) + 1\}$.

Solution:



- state 0: $hi(w) = lo(w)$
- state 1: $hi(w) = lo(w) + 1$

Missing transitions lead to a hidden dump state. ■

(b) Describe a regular expression for L_{+1}.

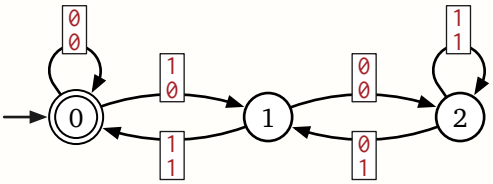
Solution: $\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right] + \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]\right)^* \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right] \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]^*$ ■

(c) Describe a DFA that accepts the language $L_{\times 3} = \{w \in \Sigma_2^* \mid hi(w) = 3 \cdot lo(w)\}$.

Solution: We start with one state for each possible value of the difference $\Delta(w) = hi(w) - 3 \cdot lo(w)$, where w is the string we have read so far. It follows that

$$\delta(\Delta, \left[\begin{smallmatrix}a\\b\end{smallmatrix}\right]) = 2\Delta + a - 3b = \begin{cases} 2\Delta & \text{if } a = 0 \text{ and } b = 0 \\ 2\Delta + 1 & \text{if } a = 1 \text{ and } b = 0 \\ 2\Delta - 3 & \text{if } a = 0 \text{ and } b = 1 \\ 2\Delta - 2 & \text{if } a = 1 \text{ and } b = 1 \end{cases}.$$

In particular, $2\Delta - 3 \le \delta(\Delta, \left[\begin{smallmatrix}a\\b\end{smallmatrix}\right]) \le 2\Delta + 1$. Thus, if Δ is ever negative, it will stay negative forever, and if Δ is ever greater than 2, then Δ will stay greater than 2 forever. So we can collapse all states $\Delta < 0$ and $\Delta > 2$ into a single junk state, leaving us only three interesting states.



■

(d) Describe a regular expression for $L_{\times 3}$.

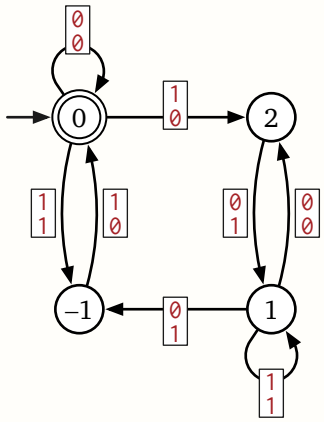
Solution: $\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right] + \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right] \left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right] \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]^* \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\right)^* \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]\right)^*$ ■

*(e) Describe a DFA that accepts the language $L_{\times 3/2} = \{w \in \Sigma_2^* \mid 2 \cdot hi(w) = 3 \cdot lo(w)\}$.

Solution: We start with one state for each possible value of the difference $\Delta(w) = 2 \cdot hi(w) - 3 \cdot low$, where w is the string we have read so far.

$$\delta(\Delta, \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]) = 2\Delta + 2a - 3b = \begin{cases} 2\Delta & \text{if } a = 0 \text{ and } b = 0 \\ 2\Delta + 2 & \text{if } a = 1 \text{ and } b = 0 \\ 2\Delta - 3 & \text{if } a = 0 \text{ and } b = 1 \\ 2\Delta - 1 & \text{if } a = 1 \text{ and } b = 1 \end{cases}.$$

In particular, we have $2\Delta - 3 \le \delta(\Delta, \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]) \le 2\Delta + 2$. No state $\Delta \le -2$ of $\Delta \ge 3$ can ever reach state 0, so we can collapse all such states into a single junk state, leaving us with only four interesting states.



This DFA implies the following regular expression for $L_{\times 3/2}$:

$$L_{\times 3/2} = \left(\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right] + \left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] + \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]\left(\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]^*\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]\right)^*\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]\right)^*$$

---

*Similar constructions imply that for **any** integers a, b, c, the language $\{w \in \Sigma_2^* \mid a \cdot \mathrm{hi}(w) = b \cdot \mathrm{lo}(w) + c\}$ is regular.* ■