1. For each of the following languages over the alphabet $\{0,1\}^*$, describe an equivalent regular expression, and briefly explain why your regular expression is correct. There are infinitely many correct answers for each language.

Rubric: 10 points = 2 points for each part = 1 for correctness + 1 for English explanation (standard regular expression rubric, scaled and rounded).

In all subproblems, these are not the only correct solutions.

(a) All strings in 1^*01^* whose length is a multiple of 3.

Solution: Call this language L_a. Every string in L_a has the form 1^i01^j, where $(i \bmod 3) + (j \bmod 3) = 2$.

$$\begin{aligned} L_a = (111)^*0(111)^*11 \qquad & i \bmod 3 = 0 \text{ and } j \bmod 3 = 2 \\ + 1(111)^*0(111)^*1 \qquad & i \bmod 3 = 1 \text{ and } j \bmod 3 = 1 \\ + 11(111)^*0(111)^* \qquad & i \bmod 3 = 2 \text{ and } j \bmod 3 = 0 \end{aligned}$$

More compactly, we have $L = (111)^*(011 + 010 + 110)(111)^*$. ■

(b) All strings that begin with the prefix 001, end with the suffix 100, and contain an odd number of 1s.

Solution: Let's call this language L_b. The shortest string in L_a is 00100. In every other string in L_a, the prefix 001 and the suffix 100 are disjoint, and the substring between them has an odd number of 1s. Moreover, *any* string with an odd number of 1s can appear between 001 and 100.

$$L_b = 00100 + 001 \cdot \langle \textit{odd\#1s} \rangle \cdot 100$$

A string with an odd number of 1s can have any number of 0s before the first 1, any number of 0s between any two 1s, and any number of 0s after the last 1.

$$\langle \textit{odd\#1s} \rangle = 0^*1(0^*10^*1)^*0^*$$

We conclude that

$$L_b = 00100 + 001 \cdot 0^*1(0^*10^*1)^*0^* \cdot 100$$

(The dots $\cdot$ aren't strictly necessary, but I think they help show the structure of the regular expression.) ■

(c) All strings that contain both 0011 and 1100 as substrings.

Solution: Let's call this language L_c. In every string $w \in L_c$, either the substrings 0011 and 1100 overlap, or they do not. If they overlap, then w contains one of the following substrings:

$$001100 \quad 0011100 \quad 110011 \quad 1100011$$

If they do not overlap, then substring 0011 appears either before or after the substring 1100.

$$\begin{aligned} L_c = (0+1)^*\left(001100 + 0011100 + 110011 + 1100011\right)(0+1)^* \\ + (0+1)^* 0011 (0+1)^* 1100 (0+1)^* \\ + (0+1)^* 1100 (0+1)^* 0011 (0+1)^* \end{aligned}$$

or slightly more compactly:

$$\begin{aligned} L_c = (0+1)^*\left(001(\varepsilon + 1 + 1(0+1)^*1)100\right)(0+1)^* \\ + (0+1)^*\left(110(\varepsilon + 0 + 0(0+1)^*0)011\right)(0+1)^* \end{aligned}$$

■

(d) All strings that contain the substring 01 an odd number of times.

Solution: Let's call this language L_c. Every substring 01 appears at boundary between a run of 0s and a run of 1s. Thus, every string in L_c consists of

- an optional run of 1s,
- an odd number of substrings of the form ⟨*run of* 0*s*⟩⟨*run of* 1*s*⟩.
- an optional run of 0s.

Let A denote the set of strings of the form ⟨*run of* 0*s*⟩⟨*run of* 1*s*⟩. Then we have

$$L_c = 1^* A (AA)^* 0^*$$

and

$$A = 0^+ 1^+ = 0^* \underline{01} 1^*$$

and therefore

$$L_c = 1^* 0^* \underline{01} 1^* (0^* \underline{01} 1^* 0^* \underline{01} 1^*)^* 0^*$$

(I've underlined every occurrence of the substring 01 for emphasis.) ■

(e) $\{0^a 1^b 0^c \mid a \geq 0 \text{ and } b \geq 0 \text{ and } c \geq 0 \text{ and } a \equiv b + c \pmod 2\}$.

Solution: Let's call our target language L_e. We can split this language into four subsets according to the legal parities of the exponents a, b, and c as follows:

$$\begin{aligned} L_e = (00)^*(11)^*(00)^* & \qquad [0^{\text{even}}1^{\text{even}}0^{\text{even}}] \\ + (00)^*(11)^*10(00)^* & \qquad [0^{\text{even}}1^{\text{odd}}0^{\text{odd}}] \\ + (00)^*0(11)^*0(00)^* & \qquad [0^{\text{odd}}1^{\text{even}}0^{\text{odd}}] \\ + (00)^*0(11)^*1(00)^* & \qquad [0^{\text{odd}}1^{\text{odd}}0^{\text{even}}] \end{aligned}$$

Or more succinctly: $L_e = (00)^*(\varepsilon + 01)(11)^*(\varepsilon + 10)(00)^*$

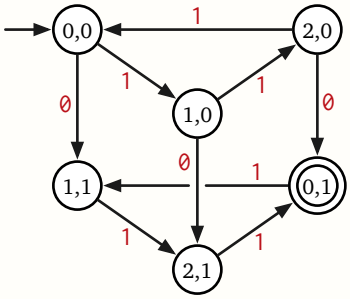
Or equivalently: $L_e = \varepsilon + (00)^*(0 + 1)(11)^*(0 + 1)(00)^*$ ■

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

Rubric: 10 points = 2 points for each part = 1 for correctness + 1 for English explanation of states (standard DFA rubric, scaled and rounded). Again, for all subproblems, these re not the only correct solutions.

(a) All strings in 1^*01^* whose length is a multiple of 3.

Solution (explicit drawing):

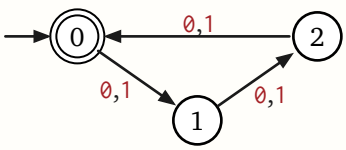


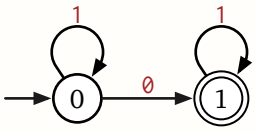
All missing transitions lead to a hidden dump state. Every (non-dump) state is labeled with a pair of integers (i, j) where

- i is the number of symbols read so far, modulo 3, and
- j is the number if 0s read so far.

■

Solution (product): We construct the product of two DFAs:

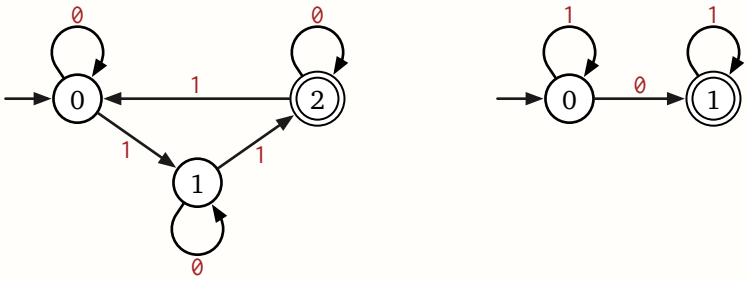




- The first DFA accepts all strings whose length is divisible by 3. States are labeled with the number of symbols read so far, modulo 3.
- The second DFA accepts all strings with exactly one 0. States are labeled with the number of 0s read so far. Missing transitions lead to a hidden dump state.

The only accepting state in the product DFA is $(0, 1)$. ■

Solution (different product): We construct the product of two DFAs:



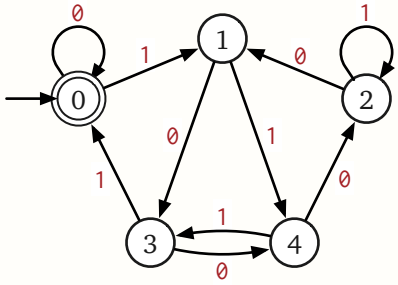
- The first DFA accepts all strings w such that $\#(1,w) \bmod 3 = 2$. States are labeled with $\#(1,w) \bmod 3$.
- The second DFA accepts all strings w such that $\#(0,w) = 1$. States are labeled with $\#(0,w)$.

In both DFAs, missing transitions lead to a hidden dump state.

The only accepting state in the product DFA is $(2,1)$. ■

(b) All strings that represent a multiple of 5 in base 3. For example, this language contains the string 10100, because $10100_3 = 90_{10}$ is a multiple of 5. (Yes, base 3 allows the digits 0, 1, and 2, but your input string will never contain a 2.)

Solution:



Each state is labeled with the base-3 value (modulo 5) of the bits read so far.[a] ■

[a] This DFA is isomorphic to the DFA in the notes that recognizes *binary* numbers divisible by 5, but with all arrows reversed and with different state names. It follows that a binary number is divisible by 5 if and only its reversal, interpreted in base 3, is also divisible by 5!

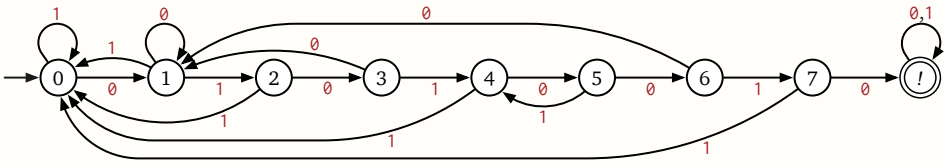
Solution: We define a DFA (Q,s,A,δ) as follows:

$$Q = \{0,1,2,3,4\}$$
$$s = 0$$
$$A = \{0\}$$
$$\delta(q,a) = (3 \cdot q + a) \bmod 5$$

Each state is labeled with the base-3 value (modulo 5) of the bits read so far. In the definition of the transition function δ, we equate the symbols 0 and 1 with their numerical values 0 and 1. ■

(c) All strings containing the substring 01010010. (The required substring is $p_6 = v_6$ from Homework 1.)

Solution:



Each numbered state k indicates that the last k symbols read match the first k symbols of 01010010. The final state *!* indicates that we have read the entire substring 01010010, which means we should (eventually) accept. ■

(d) All strings whose ninth-to-last symbol is 0.

Solution: We define a DFA (Q, s, A, δ) over the alphabet $\Sigma = \{0, 1\}$ as follows:

$$
\begin{aligned}
Q &= \Sigma^9 \\
s &= 111111111 \\
A &= \{0x \mid x \in \{0,1\}^8\} \\
\delta(ax, b) &= xb \qquad \text{for all } a, b \in \Sigma \text{ and } x \in \Sigma^8
\end{aligned}
$$

Each state stores the last nine symbols read, so there are a total of $2^9 = 512$ states. To avoid additional states that only deal with the first eight symbols in the input string, we pretend that we have read nine 1s before the actual input string begins. ■

Solution: We define a DFA (Q, s, A, δ) over the alphabet $\Sigma = \{0, 1\}$ as follows:

$$
\begin{aligned}
Q &= \{w \in \Sigma^* \mid |w| \le 9\} \\
s &= \varepsilon \\
A &= \{0x \mid x \in \{0,1\}^8\} \\
\delta(w, a) &= \begin{cases} wa & \text{if } |w| < 9 \\ xa & \text{if } |w| = 9 \text{ and } w = bx \text{ for some } b \in \Sigma \end{cases}
\end{aligned}
$$

Each state stores the last nine symbols read, or all symbols read if that number is less than 9, so there are a total of $2^0 + 2^1 + \cdots + 2^9 = 1023$ states. ■

(e) All strings w such that $(\#(\texttt{0}, w) \bmod 3) + (\#(\texttt{1}, w) \bmod 7) = (|w| \bmod 4)$.

Solution: We define a DFA (Q, s, A, δ) over the alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$ as follows:

$$Q = \{0,1,2\} \times \{0,1,2,3,4,5,6\} \times \{0,1,2,3\}$$
$$s = (0,0,0)$$
$$A = \{(i,j,k) \in Q \mid i + j = k\}$$
$$\delta((i,j,k), \texttt{0}) = (i+1 \bmod 3,\ j\ ,\ k+1 \bmod 4)$$
$$\delta((i,j,k), \texttt{1}) = (i\ ,\ j+1 \bmod 7,\ k+1 \bmod 4)$$

Each state is represented by a triple of integers (i, j, k), where

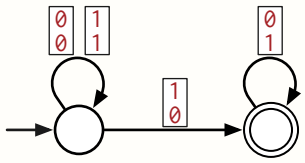
- $i = \#(\texttt{0}, w) \bmod 3$
- $j = \#(\texttt{1}, w) \bmod 7$
- $k = |w| \bmod 4$

where w is the portion of the input string read so far. ■

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}.

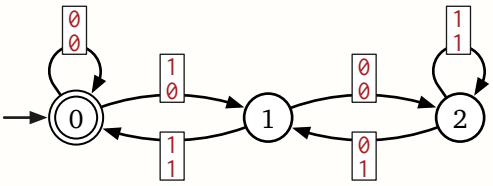
Solution: $\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right] + \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]\right)^* \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right] \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]^*$ ■

(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 $\Delta(w) = hi(w) - 3 \cdot low$, 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 $L_{\times 3}$.

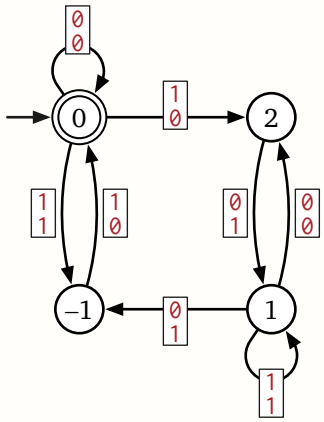
Solution: $\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right] + \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]^*\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\right)^*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

*(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 $\{w \in \Sigma_2^* \mid a \cdot \text{hi}(w) = b \cdot \text{lo}(w) + c\}$ is regular. ■