

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 $\{0, 1\}$. There are infinitely many correct regular expressions for each language.

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 0011 , end with 0011 , and whose length is a multiple of 5.

Solution: Call this language L_a . Every string in L_a has the form $0011 x 0011$ for some string x such that $(|x| \bmod 5) = 2$.

$$L_a = 0011 (0 + 1)^2 ((0 + 1)^5)^* 0011$$

■

- (b) All strings that do not begin with 000 or 111 , but do end with 1010 .

Solution: Let L_b be the above language, and let $x \in L_b$ be a string in the language. We divide into the following cases based on the length of the string x , which cover all possibilities. For each we will construct a regular expression capturing strings that do not begin with 000 or 111 . Note that, the string has to end with 1010 . (i) x has length 4, (ii) x has length 5, (iii) x has length 6, (iv) x has length *at least* 7.

- In case (i), the string has to be exactly 1010 .
- In case (ii), the number of characters before 1010 is exactly one, which is either 0 or 1 .
- In case (iii), the number of characters before 1010 is exactly two, and these should not make 000 or 111 with the end string 1010 . Thus, these can be 00 , 01 , or 10 .
- In case (iv), the number of characters before 1010 is at least three, and the first three characters should not be 000 or 111 . Thus, these can be 001 , 010 , 011 , 100 , 101 , or 110 .

Answer:

$$\begin{aligned} L_b &= 1010 \\ &+ (0 + 1) 1010 \\ &+ (00 + 01 + 10) 1010 \\ &+ (001 + 010 + 011 + 100 + 101 + 110) (0 + 1)^* 1010 \end{aligned}$$

■

- (c) All strings that have an odd number of 1s before the first 0, contain an even number of 0s, and contain 101 as a substring.

Solution: Let L_c be the language described above. A string x in L_c can be divided into two blocks. The first block is the prefix of 1s with odd length. The regular expression for the same is $(1(11)^*)$. This is followed by the second block that starts with a 0 and has an even number of zeros. The corresponding regular expression is $(01^*01^*)^*$. For x to contain string 101, 101 must be in one of the (01^*01^*) within the second block. Based on which of the two zeros it corresponds to in $(0 \cdot 1^* 0 1^*)$, there are three cases:

- (i) using the first 0 immediately after the first run of 1s with odd length: (011^*01^*) ,
- (ii) using the first 0 in (01^*01^*) after its few occurrences: (1011^*01^*) ,
- (iii) using the second 0: (01^*1011^*) .

Answer: [In the solution below, for a regular expression R , R^+ is a shorthand for $(R \cdot R^*)$.]

$$L_c = (1(11)^*)(011^*01^*)(01^*01^*)^* \\ + (1(11)^*)(01^*01^*)^+(1011^*01^*)(01^*01^*)^* \\ + (1(11)^*)(01^*01^*)^*(01^*1011^*)(01^*01^*)^*$$

■

- (d) All strings where every run of 0s with odd length is followed by a run of 1s with odd length. A run is a consecutive sequence of the same alphabet, e.g., 000 is a run of zeros with odd length, while 111111 is a run of ones with even length, and the string 00011100 has one run of 0s with odd length followed by one run of 1s with odd length.

Solution: Let the above language be L_d . A string $x \in L_d$ can be thought of as alternating runs of 0s and 1s. Each run of 0s has length either even or odd. An even-lengthed run of 0s can be followed by a run of 1s of any length. This can be represented by regular-expression $((00)^+1^*)$. An odd-lengthed run of 0s must be followed by an odd-lengthed run of 1s, which can be represented as $(0(00)^*1(11)^*)$. Finally, the string can start with a run of 1s of arbitrary length.

$$L_d = 1^* \left(((00)^+1^*) + (0(00)^*1(11)^*) \right)^*$$

■

- * (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:

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

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.

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

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

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

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

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

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

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

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

$$L_e = 0 + (23)^* + (32)^* + (23)^*2 + (32)^*3 \\ + (0 + (32)^* + (23)^*2) \cdot (1 \cdot (0 + 2(32)^*))^* \cdot 1 \cdot (0 + (23)^* + 2(32)^*)$$

■

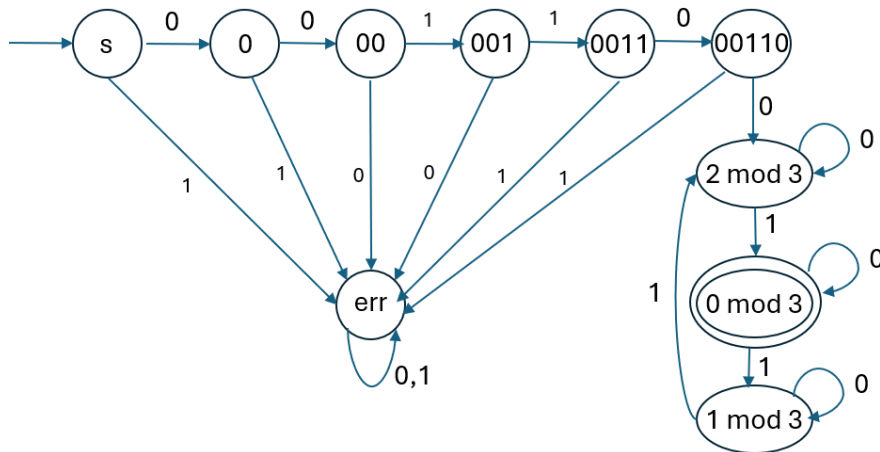
2. For each of the following languages over the alphabet $\{0, 1\}$, describe a DFA that accepts the language, and briefly describe the purpose of each state. You can describe your DFA using a drawing, or using formal mathematical notation, or using a product construction; see the standard DFA rubric.

Rubric: 60% for correctness + 40% 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) (2 points) All strings that start with 001100 and have the number of 1 s divisible by 3.

Solution: • s is the start state; no symbols have been read.

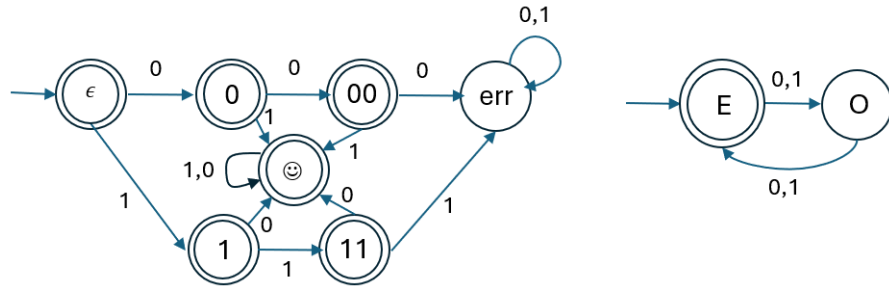
- First 5 states, namely 0, 00, 001, 0011, and 00110 indicates the string read so far.
- The *Err* state indicates an error, from which accept state cannot be reached.
- Each of the $(i \bmod 3)$ states for $i = 0, 1, 2$ indicates that we have seen $i \bmod 3$ number of ones. Therefore, at state 0011 we have to get a 0, and at this point we have seen $2 \bmod 3$ number of ones.
- the remaining transitions keep track of $\#1$'s mod 3.
- $0 \bmod 3$ is the accept state.



(b) (4 points) All even length strings that do not start with 000 or 111.

Solution (Use product construction): The left DFA below accepts all strings that do not start with 000 or 111. State ϵ , 0, 1, 00, and 11 represent the string seen so far. State *err* is the error state indicating that the string started with either 000 or 111. The “Happy state” is reached when 000 and 111 are avoided at the start of the string, since now anything can follow after that. Clearly, 000 and 111 are avoided when we are in either of the “Happy state”, ϵ , 0, 00, 1 or 11 states, and hence these are the accepting states.

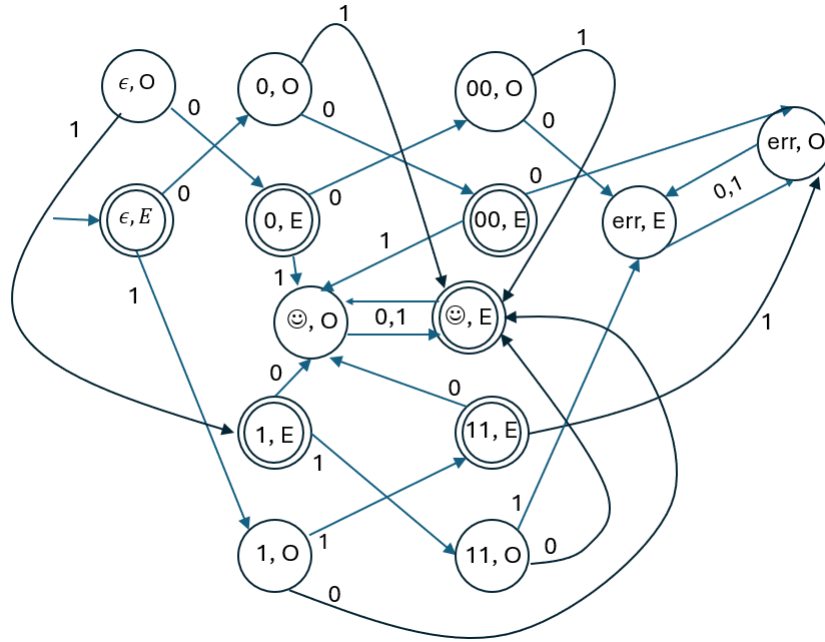
The right DFA accepts all even-length strings, where state *E* represents even-length and *O* represents odd-length. The transitions go between the two, and *E* is the accept state.



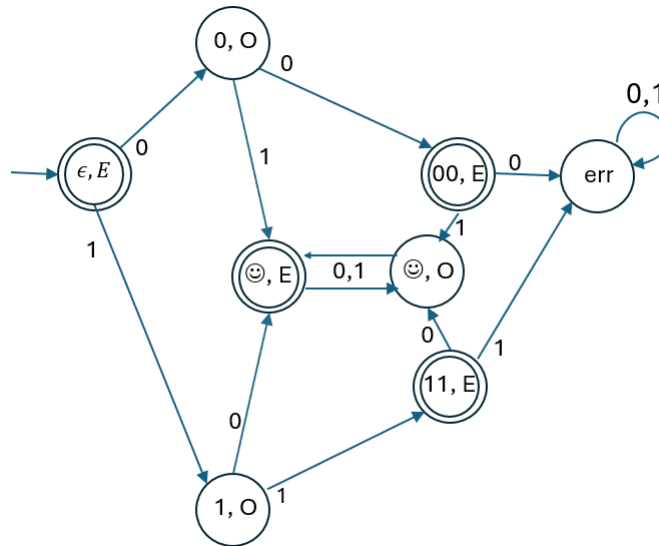
☺: Happy state

Let the left and the right DFA above be denoted by $(Q_1, s_1, A_1, \delta_1)$ and $(Q_2, s_2, A_2, \delta_2)$ respectively. The following DFA is a product of these two, which can be defined as follows: (Q, s, A, δ) such that

- $Q = Q_1 \times Q_2$,
- $s = (s_1, s_2)$,
- $A = A_1 \times A_2$, and
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ for all $(q_1, q_2) \in Q_1 \times Q_2$ and $a \in \{0, 1\}$.



Note that several states in the above DFA are unreachable, and it can be simplified to the following. The following DFA is a product of the above two.



(c) (4 points) All strings such that **exactly one of** the following is true:

- Every run of 0s with length divisible by 374 is followed by a run of 1s with length divisible by 374.
- Contains substring 01 a number of times divisible by 374.

Solution: We will define DFA for each of the two parts above separately and then use product construction.

[We note that, to keep the exposition clear, the solution is not optimized in terms of number of states, etc. And there are more succinct solutions.]

Part I. Let L_1 be the language containing all string where every run of 0 s with length divisible by 374 is followed by a run of 1 s with length divisible by 374. Note that ϵ is in L_1 , and if a string start with a run of 1 s then this particular can have any length. Let the DFA for L_1 be $(Q_1, s_1, A_1, \delta_1)$ over the alphabet set $\Sigma = \{0, 1\}$ be as follows:

- Q_1 : We have *start*, *err*, and $(i, j, prev)$ states. States *start* and *err* are self-explanatory. Each state of the form $(i, j, prev)$ stores $(\#0 \bmod 374, \#1 \bmod 374, \text{previous symbol})$. The previous symbol is either 0 or 1 .
- A_1 : Since $\epsilon \in L_1$, *start* state is included. And any state of the form $(i, j, prev)$ such that either $i \neq 0$ indicating the last runs of 0 s did not have length $0 \bmod 374$, or is $(0, 0, 1)$ indicating the condition is met by the last run of 1 s.
- δ_1 : Essentially, if the current symbol is a , and it matches with the previous symbol, then we increment $(\#a \bmod 374)$ by one. Else, reset $(\#a \bmod 374)$ to one, unless the condition is violated. That is, if we get a zero from any state of the form $(0, j, 1)$ then we transition to the *err* state because we have encountered a run of 1 s with length not equals $(0 \bmod 374)$ while the previous run of 0 s had length $(0 \bmod 374)$ violating the condition.

$$Q_1 = \{start, err\} \cup \{(i, j, prev) \mid i, j \in \{0, 1, \dots, 373\} \text{ and } prev \in \{0, 1\}\}$$

$$s_1 = start$$

$$A_1 = \{start\} \cup \{(i, j, prev) \in Q \mid \text{Either } i \neq 0, \text{ or } (i = 0 \text{ and } j = 0 \text{ and } prev = 1)\}$$

$$\delta_1(err, a) = err \quad \forall a \in \{0, 1\}$$

$$\delta_1(start, 0) = (1, 0, 0)$$

$$\delta_1(start, 1) = (1, 1, 1) \quad \text{Here } i = 1 \text{ indicates that this run of } 1\text{s can have any length}$$

$$\delta_1((i, j, 0), 0) = ((i + 1) \bmod 374, j, 0)$$

$$\delta_1((i, j, 0), 1) = (i, 1, 1)$$

$$\delta_1((i, j, 1), 1) = (i, (j + 1) \bmod 374, 1)$$

$$\delta_1((i, j, 1), 0) = \begin{cases} err & \text{if } i = 0 \text{ and } j \neq 0 \\ (1, j, 0) & \text{otherwise} \end{cases}$$

Part II. For the second part, let the language L_2 contain all the strings with substring 01 appearing $0 \bmod 374$ many times. For this, it suffices to keep track of the previous symbol to check if the substring 01 appeared, and the number of times it appeared mod 374. For L_2 , define DFA $(Q_2, s_2, A_2, \delta_2)$ as follows:

- Q_2 : States of the form $(i, prev)$ indicating that we have encountered 01 substring $(i \bmod 374)$ times and the previous symbol was $prev$. And a $start$ state.
- A_2 : Contains $start$ state because $\epsilon \in L_2$, and states $(0, 0)$ and $(0, 1)$.
- δ_2 : Transitions from the $start$ state on symbol a is to $(0, a)$. Transition from state (i, b) for $b \in \{0, 1\}$ on symbol a is to (i, a) , unless $b = 0$ and $a = 1$ in which case we have encountered the substring 01 and hence the counter should be incremented to $((i + 1) \bmod 374)$.

$$Q_2 = \{start\} \cup \{(i, prev) \mid i \in \{0, 1, \dots, 373\} \text{ and } prev \in \{0, 1\}\}$$

$$s_2 = start$$

$$A_2 = \{start, (0, 0), (0, 1)\}$$

$$\delta_2(start, 0) = (0, 0)$$

$$\delta_2(start, 1) = (0, 1)$$

$$\delta_2((i, 0), 0) = (i, 0)$$

$$\delta_2((i, 0), 1) = ((i + 1) \bmod 374, 1)$$

$$\delta_2((i, 1), 0) = (i, 0)$$

$$\delta_2((i, 1), 1) = (i, 1)$$

Final Solution through Product Construction. Let L be the language such that every string $x \in L$ is either in L_1 or in L_2 , but not in both. A DFA (Q, s, A, δ) for L can be constructed by taking a product of the DFAs for L_1 and L_2 as follows:

$$Q = Q_1 \times Q_2$$

$$s = (s_1, s_2)$$

$$A = \{(a_1, q_2) \in A_1 \times Q_2 \mid q_2 \notin A_2\} \cup \{(q_1, a_2) \in Q_1 \times A_2 \mid q_1 \notin A_1\}$$

$$\delta((q_1, q_2), a) = ((q_1, a), (q_2, a)) \quad \forall a \in \{0, 1\}$$

■

Alternate-Solution: Below is an alternate solution for Part I with fewer states. The remaining solution is as above.

Part I. A binary string can be thought of as alternating runs of 0 s and 1 s. For a string $x \in L_1$, a run of 1 s can have any length if the preceding run of 0 s had length not divisible by 374. Otherwise, it must have a length divisible by 374.

$$Q_1 = \{any\#1, err, start\} \cup \{i\#0, i\#1 \mid i \in \{0, 1, \dots, 373\}\}$$

$$s_1 = start$$

$$A_1 = \{start, any\#1, 0\#1\} \cup \{i\#0 \mid i \in \{1, \dots, 373\}\}$$

$$\delta_1(err, a) = err$$

$$\forall a \in \{0, 1\}$$

$$\begin{aligned}
\delta_1(\text{start}, 0) &= 1\#0 \\
\delta_1(\text{start}, 1) &= \text{any}\#1 \\
\delta_1(\text{any}\#1, 0) &= 1\#0 \\
\delta_1(\text{any}\#1, 1) &= \text{any}\#1 \\
\delta_1(i\#0, 0) &= ((i + 1) \bmod 374)\#0 && \forall i \in \{0, 1, \dots, 373\} \\
\delta_1(0\#0, 1) &= 1\#1 \\
\delta_1(i\#0, 1) &= \text{any}\#1 && \forall i \in \{1, 2, \dots, 373\} \\
\delta_1(0\#1, 0) &= 1\#0 \\
\delta_1(i\#1, 0) &= \text{err} && \forall i \in \{1, 2, \dots, 373\} \\
\delta_1(i\#1, 1) &= ((i + 1) \bmod 374)\#1 && \forall i \in \{0, 1, \dots, 373\}
\end{aligned}$$

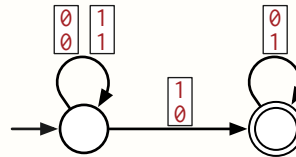
In the above construction, state $i\#0$ indicates we are in a run of 0s and the length seen so far is $i \bmod 374$. If the ongoing run of 0 does not end with a length $0 \bmod 374$, then the run of 1s that follows can have any length, which is captured by the transition to state $\text{any}\#1$. Otherwise, we must keep track of its length $\bmod 374$ through states $i\#1$. The error state err is reached when the length does not follow this condition.

■

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} .

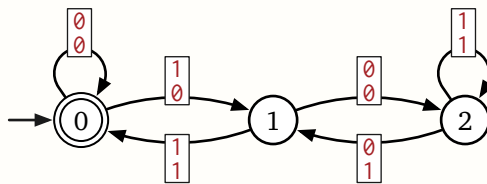
Solution: $(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix})^* \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^*$ ■

(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

$$\delta(\Delta, \begin{bmatrix} a \\ b \end{bmatrix}) = 2\Delta + a - 3b = \begin{cases} 2\Delta & \text{if } a = 0 \text{ and } b = 0 \\ 2\Delta + 1 & \text{if } a = 1 \text{ and } b = 0 \\ 2\Delta - 3 & \text{if } a = 0 \text{ and } b = 1 \\ 2\Delta - 2 & \text{if } a = 1 \text{ and } b = 1 \end{cases}$$

In particular, $2\Delta - 3 \leq \delta(\Delta, \begin{bmatrix} a \\ b \end{bmatrix}) \leq 2\Delta + 1$. 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 $L_{\times 3}$.

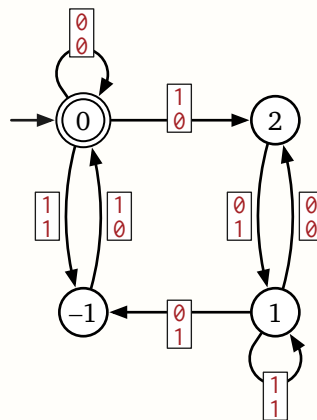
Solution: $(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix})(\begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \end{bmatrix})^* \begin{bmatrix} 1 \\ 1 \end{bmatrix}^*$ ■

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

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

$$\delta(\Delta, \begin{bmatrix} a \\ b \end{bmatrix}) = 2\Delta + 2a - 3b = \begin{cases} 2\Delta & \text{if } a = 0 \text{ and } b = 0 \\ 2\Delta + 2 & \text{if } a = 1 \text{ and } b = 0 \\ 2\Delta - 3 & \text{if } a = 0 \text{ and } b = 1 \\ 2\Delta - 1 & \text{if } a = 1 \text{ and } b = 1 \end{cases}$$

In particular, we have $2\Delta - 3 \leq \delta(\Delta, \begin{bmatrix} a \\ b \end{bmatrix}) \leq 2\Delta + 2$. No state $\Delta \leq -2$ or $\Delta \geq 3$ 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 $L_{\times 3/2}$:

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

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. ■