Describe deterministic finite-state automata that accept each of the following languages over the alphabet $\Sigma = \{0, 1\}$. You may find it easier to describe these DFAs formally than to draw pictures. Either drawings or formal descriptions are acceptable, as long as the states Q, the start state s, the accept states A, and the transition function δ are all be clear. Try to keep the number of states small.

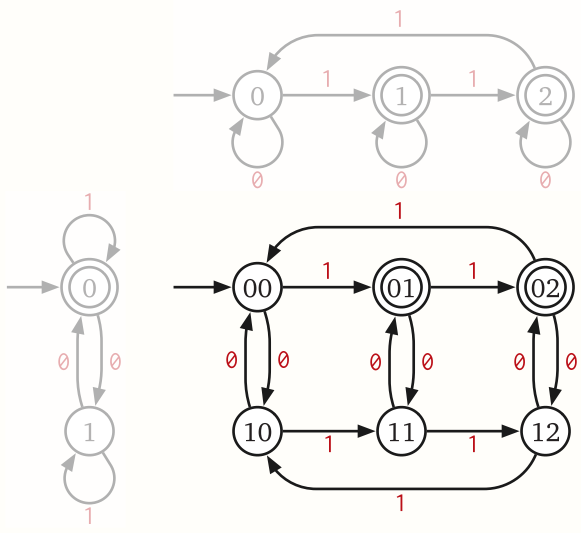
1. All strings in which the number of 0s is even **and** the number of 1s is *not* divisible by 3.

Solution: We use a standard product construction of two DFAs, one accepting strings with an even number of 0s, and the other accepting strings where the number of 1s is not a multiple of 3. The product DFA has six states, each labeled with a pair of integers, one indicating the number 0s read modulo 2, the other indicating the number of 1s read modulo 3.

$$Q := \{0,1\} \times \{0,1,2\}$$
$$s := (0,0)$$
$$A := \{(0,1),(0,2)\}$$

$$\delta((q,r),0) := (q+1 \bmod 2,\ r)$$
$$\delta((q,r),1) := (q,\ r+1 \bmod 3)$$

In this case, the product DFA is simple enough that we can just draw it out in full. I've drawn the two factor DFAs (in gray) to the left and above for reference.



■

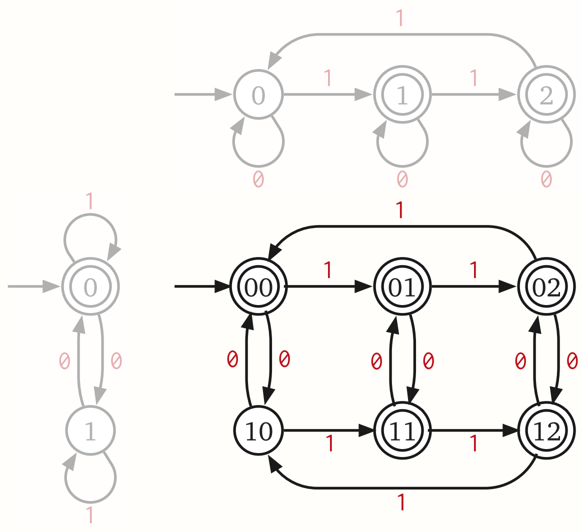
2. All strings in which the number of `0`s is even **or** the number of `1`s is *not* divisible by 3.

Solution: We use a standard product construction of two DFAs, one accepting strings with an even number of `0`s, and the other accepting strings where the number of `1`s is not a multiple of 3. The product DFA has six states, each labeled with a pair of integers, one indicating the number `0`s read modulo 2, the other indicating the number of `1`s read modulo 3.

$$Q := \{0,1\} \times \{0,1,2\}$$
$$s := (0,0)$$
$$A := \{(0,0),(0,1),(0,2),(1,1),(1,2)\}$$

$$\delta((q,r),\mathtt{0}) := (q+1 \bmod 2,\ r)$$
$$\delta((q,r),\mathtt{1}) := (q,\ r+1 \bmod 3)$$

In this case, the product DFA is simple enough that we can just draw it out in full. I've drawn the two factor DFAs (in gray) to the left and above for reference.



This is exactly the same DFA as problem 1, except for the accepting states. Similar standard product constructions yield DFAs for any other boolean combination of these two factor languages, including the following:

- $\{w \mid \#(\mathtt{0},w)$ is even and $\#(\mathtt{1},w)$ is divisible by 3$\}$
- $\{w \mid \#(\mathtt{0},w)$ is odd or $\#(\mathtt{1},w)$ is divisible by 3$\}$
- $\{w \mid$ if $\#(\mathtt{0},w)$ is even, then $\#(\mathtt{1},w)$ is not divisible by 3$\}$
- $\{w \mid$ if $\#(\mathtt{1},w)$ is not divisible by 3, then $\#(\mathtt{0},w)$ is even$\}$
- $\{w \mid \#(\mathtt{0},w)$ is even if and only if $\#(\mathtt{1},w)$ is not divisible by 3$\}$
- $\{w \mid \#(\mathtt{0},w)$ is odd if and only if $\#(\mathtt{1},w)$ is divisible by 3$\}$

This is why, whenever you describe a product construction, especially for a homework or exam problem, you ***must*** specify the set of accepting states. ■

3. Given DFAs M_1 and M_2, all strings in $\overline{L(M_1)} \oplus L(M_2)$.

Solution: We use a standard product construction of two DFAs, $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$. As observed in the solution the problem 2, the major difference is the set of accepting states. Here

$$\begin{aligned} w \in \overline{L(M_1)} \oplus L(M_2) &\iff w \in \overline{L(M_1)} \textbf{\textit{ xor }} w \in L(M_2) \\ &\iff w \notin L(M_1) \textbf{\textit{ xor }} w \in L(M_2). \end{aligned}$$

so a state (q, r) should be accepting if and only if $q \notin A_1$ ***xor*** $r \in A_2$.

$$\begin{aligned} Q &:= Q_1 \times Q_2 \\ s &:= (s_1, s_2) \\ A &:= \{(q, r) \in Q \mid q \notin A_1 \textbf{\textit{ xor }} r \in A_2\} \end{aligned}$$

$$\delta((q, r), a) := (\delta_1(q, a), \delta_2(r, a))$$

Since M_1 and M_2 were given abstractly, the product DFA can only be specified mathematically. ■

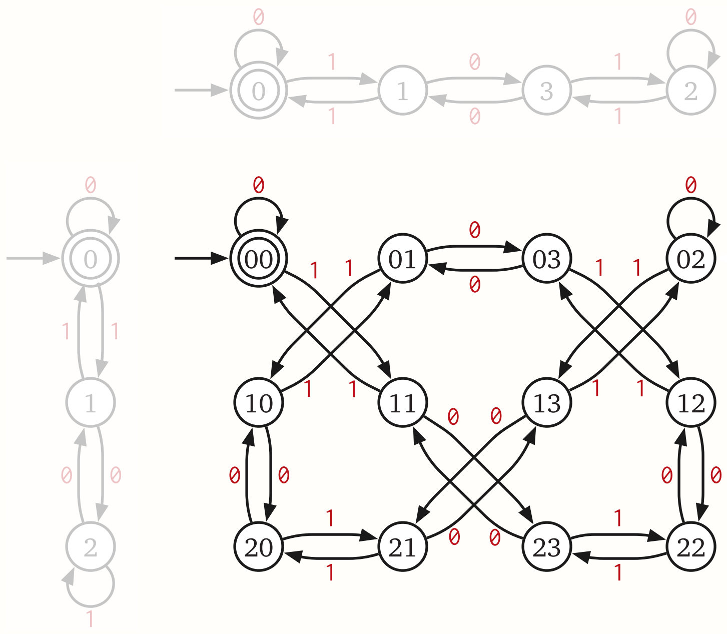
4. All strings that are **both** the binary representation of an integer divisible by 3 **and** the ternary (base-3) representation of an integer divisible by 4. For example, the string `1100` is an element of this language, because it represents $2^3 + 2^2 = 12$ in binary and $3^3 + 3^2 = 36$ in ternary.

Solution: Again, we use a standard product construction of two DFAs, one accepting binary strings divisible by 3, the other accepting ternary strings divisible by 4. The product DFA has twelve states, each labeled with a pair of integers: The binary value read so far modulo 3, and the ternary value read so far modulo 4.

$$Q := \{0,1,2\} \times \{0,1,2,3\}$$
$$s := (0,0)$$
$$A := \{(0,0)\}$$

$$\delta((q,r),\mathtt{0}) := (2q \bmod 3, \quad 3r \bmod 4)$$
$$\delta((q,r),\mathtt{1}) := (2q+1 \bmod 3, \ 3r+1 \bmod 4)$$

For reference, here is a drawing of the DFA, with the two factor DFAs (in gray) to the left and above. We wouldn't expect you to draw this, especially on exams. Or more accurately: We would expect you ***not*** to draw this, ***especially*** on exams. The states of the factor DFA that maintains ternary-value-mod-4 are deliberately "out of order" to simplify the drawing.



■

5. All strings in which the subsequence `0101` appears an even number of times.

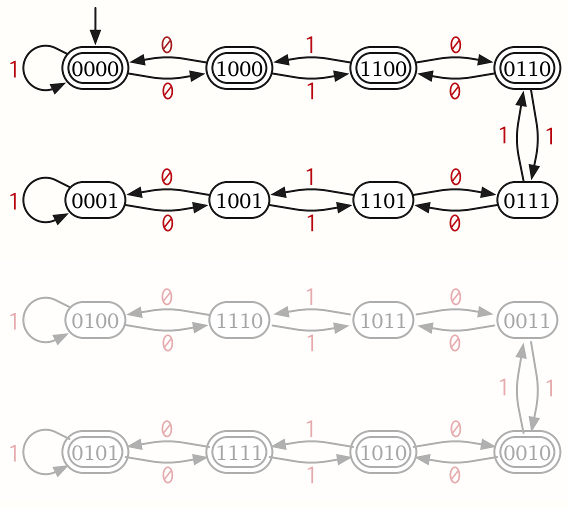
Solution: Our DFA has 16 states, each labeled with a vector (a, b, c, d) of four bits:

- a is the number of times (mod 2) that we have seen `0`.
- b is the number of times (mod 2) we have seen the subsequence `01`.
- c is the number of times (mod 2) we have seen the subsequence `010`.
- d is the number of times (mod 2) we have seen the subsequence `0101`.

$$\begin{aligned} Q &:= \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \\ s &:= (0,0,0,0) \\ A &:= \{(a,b,c,0) \mid a,b,c \in \{0,1\}\} \end{aligned}$$

$$\begin{aligned} \delta((a,b,c,d),\mathtt{0}) &:= (a+1 \bmod 2,\ b,\ c+b \bmod 2,\ d) \\ \delta((a,b,c,d),\mathtt{1}) &:= (a,\ b+a \bmod 2,\ c,\ d+c \bmod 2) \end{aligned}$$

This DFA is small enough that we can draw is out in full. It turns out that only eight of the sixteen states are reachable from the start state; the unreachable states are grayed out in the figure below.



■

6. All strings w such that $\binom{|w|}{2} \bmod 6 = 4$. *[Hint: Maintain both $\binom{|w|}{2} \bmod 6$ and $|w| \bmod 6$.]*

Solution: Our DFA has 36 states, each labeled with a pair of integers representing $\binom{|x|}{2} \bmod 6$ and $|x| \bmod 6$, where x is the prefix of the input read so far.

$$\begin{aligned} Q &:= \{0,1,2,3,4,5\} \times \{0,1,2,3,4,5\} \\ s &:= (0,0) \\ A &:= \{(4,r) \mid r \in \{0,1,2,3,4,5\}\} \end{aligned}$$

$$\begin{aligned} \delta((q,r),\mathtt{0}) &:= (q + r \bmod 6,\ r+1 \bmod 6) \\ \delta((q,r),\mathtt{1}) &:= (q + r \bmod 6,\ r+1 \bmod 6) \end{aligned}$$

The transition function exploits the identity $\binom{n+1}{2} = \binom{n}{2} + n$. ■

Solution (sketch): This is the set of all strings w such that $|w| \bmod 12 \in \{5,8\}$. This language can be accepted using a 12-state DFA. ■

*7. All strings w such that $F_{\#(\texttt{10},w)} \bmod 10 = 4$, where $\#(\texttt{10}, w)$ denotes the number of times `10` appears as a substring of w, and F_n is the nth Fibonacci number:

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

Solution: Our DFA has 200 states, each labeled with three values:

- $F_k \bmod 10$, where k is the number of times we have seen the substring `10`.
- $F_{k+1} \bmod 10$, where k is the number of times we have seen the substring `10`.
- The last symbol read (or `0` if we have read nothing yet)

Here is the formal description:

$$\begin{aligned} Q &:= \{0,1,2,3,4,5,6,7,8,9\} \times \{0,1,2,3,4,5,6,7,8,9\} \times \{\texttt{0},\texttt{1}\} \\ s &:= (0,1,\texttt{0}) \\ A &:= \{(4,r,a) \mid r \in \{0,1,2,3,4,5,6,7,8,9\} \text{ and } a \in \{\texttt{0},\texttt{1}\}\} \end{aligned}$$

$$\begin{aligned} \delta((q,r,\texttt{0}),\texttt{0}) &:= (q,r,\texttt{0}) \\ \delta((q,r,\texttt{1}),\texttt{0}) &:= (r, q+r \bmod 10, \texttt{0}) \\ \delta((q,r,\texttt{0}),\texttt{1}) &:= (q,r,\texttt{1}) \\ \delta((q,r,\texttt{1}),\texttt{1}) &:= (q,r,\texttt{1}) \end{aligned}$$

The transition function exploits the recursive definition $F_{k+1} = F_k + F_{k-1}$. ■

Solution (sketch): The Fibonacci numbers modulo 10 define a repeating sequence with period 60. So this language can be accepted by a DFA with "only" 120 states. ■