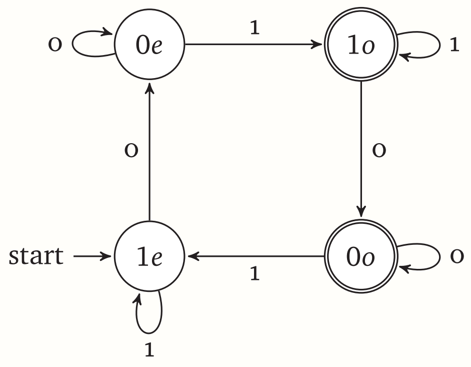
1. This is a two part question. For the first part you'll be asked to come up with several (semi-)simple DFAs and for the next you'll be asked to formally combine those DFAs into one solution. Read the whole question before starting.

   (a) Describe the DFA that describes the following languages ($\Sigma = \{0, 1\}$). Formally define the DFAs and make sure their definitions are unique w.r.t. the other languages in 1.a (will make sense when doing part b):

      i. L_1 contains all strings where the substring 01 appears an odd number of times.

Solution: The DFA is drawn below.



The DFA accepts all strings in which the substring 01 appears an odd number of times. Each state records the last symbol read (1 if nothing has been read yet), and whether the DFA has read an even or odd number of 01s.

Formal Description:

$$
\begin{aligned}
Q &= \{0, 1\} \times \{0, 1\} \\
s &= (1, 0) \\
A &= \{(a, b) \mid b \neq 0\} \\
\delta((0, b), 0) &= (0, b) \\
\delta((0, b), 1) &= (1, (b + 1) \bmod 2) \\
\delta((1, b), 0) &= (0, b) \\
\delta((1, b), 1) &= (1, b)
\end{aligned}
$$

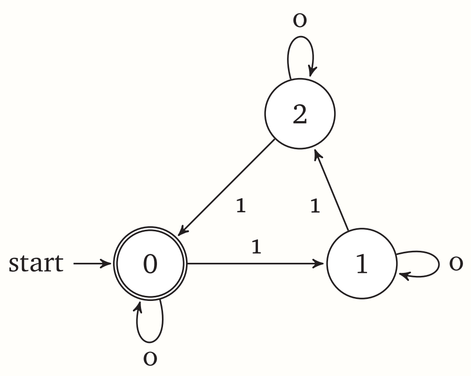
The state (a, b) indicates the following:

- a is the last symbol read by the DFA, or 1 if the DFA hasn't read anything yet.
- b is the number of times the DFA has read the substring 01, modulo 2.

■

ii. L_2 contains all strings where $\#(1, w)$ is divisible by three.

Solution: The DFA is drawn below.



The DFA accepts all strings where the number of 1s is divisible by 3. Each state records the number of 11s (mod 3) read so far).

Formal Description:

$$\begin{aligned} Q &= \{0, 1, 2\} \\ s &= 0 \\ A &= \{c \mid c = 0\} \\ \delta(c, 0) &= c \\ \delta(c, 1) &= (c + 1) \bmod 3 \end{aligned}$$

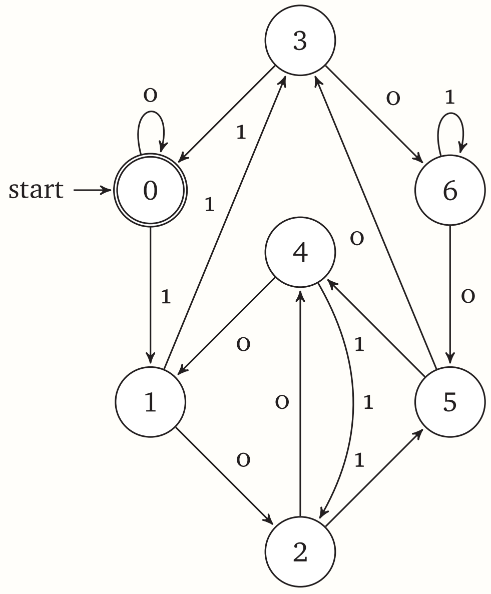
The state c indicates the following:

- c is the number of times the DFA has read the symbol 1, modulo 3.

■

iii. L_3 contains all strings where the binary value of w is divisible by seven.

Solution: The DFA is drawn below.



The DFA accepts all strings whose binary value is divisible by 7. Each state records the binary value (mod 7) of the string read so far).

Formal Description:

$$
\begin{aligned}
Q &= \{0,1,2,3,4,5,6\} \\
s &= 0 \\
A &= \{d \mid d = 0\} \\
\delta(d, 0) &= (2d) \bmod 7 \\
\delta(d, 1) &= (2d+1) \bmod 7
\end{aligned}
$$

The state d indicates the following:

- d is the binary value of the string red so far, modulo 7.

■

(b) Let L denote the set of all strings $w \in \{0, 1\}^*$ that are in **at most two** of the languages in part (a). Formally describe a DFA with input alphabet $\Sigma = \{0, 1\}$, that accepts the language L, by explicitly describing the states Q, the start state s, the accepting states A, and the transition function δ. Do not attempt to draw this DFA. At minimum, the smallest DFA for this language has 84 states.

Argue your machine is correct by *concisely* explaining explaining your DFA (it's formal definition).

Solution:

$$\begin{aligned}
Q &= \{0,1\} \times \{0,1\} \times \{0,1,2\} \times \{0,1,2,3,4,5,6\}\\
s &= (1,0,0,0)\\
A &= \{(a,b,c,d) \mid b = 0 \text{ or } c \neq 0 \text{ or } d \neq 0\}\\
\delta((0,b,c,d),0) &= (0,b,c,(2d) \bmod 7)\\
\delta((0,b,c,d),1) &= (1,(b+1) \bmod 2,(c+1) \bmod 3,(2d+1) \bmod 7)\\
\delta((1,b,c,d),0) &= (0,b,c,(2d) \bmod 7)\\
\delta((1,b,c,d),1) &= (1,b,(c+1) \bmod 3,(2d+1) \bmod 7)
\end{aligned}$$

The state (a,b,c,d) indicates the following:

- a is the last symbol read by the DFA, or 1 if the DFA hasn't read anything yet.
- b is the number of times the DFA has read the substring 01, modulo 2.
- c is the number of times the DFA has read the symbol 1, modulo 3.
- d is the binary value of the string red so far, modulo 7.

■

2. Let

$$\Sigma = \left\{ \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix} \right\}$$

Consider each row to be a binary number and let

$$C = \{w \in \Sigma^* \mid \text{the bottom row of } w \text{ is three tims the top row.}\}$$

For example

$$\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} \in C, \text{ but } \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} \notin C.$$

Show that C is regular.

Solution: Idea: In order to to prove that C is regular, we need to construct a DFA that recognizes this language. To do so, we can use the property that regular languages are closed under reversal and read the input backward, i.e. start with the low order bits. Multiplying by 3 in binary is equivalent to adding the multiplicand with the result of shifting the multiplicand itself to the left by one bit. For example, three times 111 (7) is equal to 111 plus 1110 (14) which is 10101 (21). So, if we represent the top and bottom rows of a string w that is in the language by $T_nT_{n-1}\dots T_1T_0$ and $B_nB_{n-1}\dots B_1B_0$, respectively, they must satisfy the following condition:

$$\begin{array}{cccc} T_n & T_{n-1} & \dots & T_1 & T_0 \\ T_{n-1} & T_{n-2} & \dots & T_0 & 0 \\ \hline B_n & B_{n-1} & \dots & B_1 & B_0 \end{array}$$

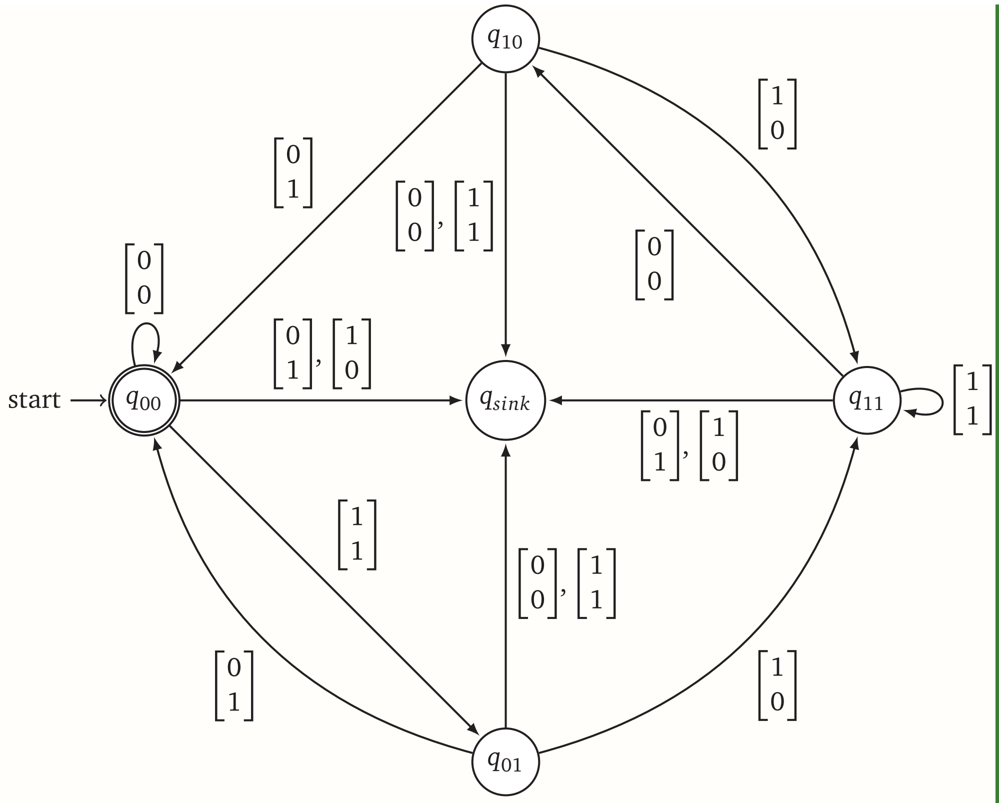
We can see that any valid input string must have $B_0 = T_0$. For the relations between B_i and T_i for $i \geq 1$, a valid value for B_i will depend on the values of T_i and T_{i-1} as well as whether or not there exists a carry-in c_i^{in} (which is equal to the carry-out c_{i-1}^{out} from the previous sum). The following logical equations describe exactly how they are related for $i \geq 1$:

$$\begin{aligned} B_i &= T_i \oplus T_{i-1} \oplus c_i^{in} \\ c_{i+1}^{in} &= (T_i \wedge T_{i-1}) \vee \big((T_i \vee T_{i-1}) \wedge c_i^{in}\big) = c_i^{\text{out}} \end{aligned}$$

Besides that, it can be seen that a valid input also requires that $c_{n+1}^{in} = c_n^{\text{out}} = 0$ and $T_n = 0$.

Therefore, in order to prove that C is regular, we need a DFA with a total of 4 states (not taking into account the sink state) to keep track of all possible combinations of c_i^{in} and T_{i-1} and, for each of these states, there will be exactly two possible valid output transitions depending on whether T_i is equal to 0 or 1 (the corresponding value of B_i will be given by the above equation).

State Diagram: Let q_{ij} denote the state for which the carry-in is equal to i and the previous symbol seen at the top is equal to j and let q_{sink} denote the sink state. Then, the following DFA recognizes C.


q_{10}
$\begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}$
start
q_{00}
q_{sink}
q_{11}
$\begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\end{bmatrix}$
q_{01}

*the transition from the sink state to itself was omitted ■

3. Let B and C be languages over $\Sigma = \{0, 1\}$. Define:

$$B \xrightarrow{0} C = \{w \in C |\text{ for some } x \in B \text{ strings } w \text{ and } x \text{ contain an equal number of } 0\text{'s}\} \quad (1)$$

Show that the class of regular languages is closed under the $\xrightarrow{0}$ operation.

Solution: Let $MB = (QB, sB, AB, \delta B)$ be a DFA that accepts the language B and $MC = (QC, sC, AC, \delta C)$ be a DFA that accepts the language C. We construct a new NFA $M' = (Q', s', A', \delta')$ that accepts $B \xrightarrow{0} C$ as follows.

Intuitively, M' receives some string w as input,inorder to decide whether the input w is in $B \xrightarrow{0} C$ the machine M' checks that $w \in C$ and checks if the string x which has equivalent number of 0s as w belongs to B .

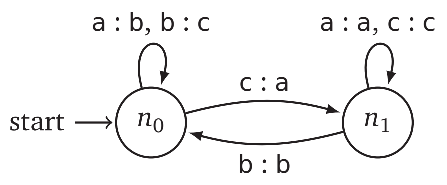
M' would simulate two machines in parallel MC, and MB.It would simulate MC to check if $w \in C$ for some input w. M' would simulate MB to check if the string x with same of of 0s as w is accepted by MB.This would be done by replacing the 1s with ε. Since,M' would have ε-transitions it would be a NFA.

So,Formally we construct NFA $M' := (\Sigma, Q', s', A', \delta')$ as follows:

$$\begin{aligned}
Q' &= QC \times QB \\
s' &= (sC, sB) \\
A' &= AC \times AB \\
\delta'((q, r), 0) &= \{(\delta C(q, 0),\ \delta B(r, 0))\} \\
\delta'((q, r), 1) &= (\delta C(q, 1), r) \\
\delta'((q, r), \varepsilon) &= (q, \delta B(r, 1))
\end{aligned}$$

■

4. A *finite-state transducer* (FST) is a type of deterministic finite automaton whose output is a string instead of just *accept* or *reject*. The following is the state diagram of finite state transducer FST_0.



Each transition of an FST is labeled at least an input symbol and an output symbol, separated by a colon (:). There can also be multiple input-output pairs for each transitions, separated by a comma (,). For instance, the transition from n_0 to itself can either take a or b as an input, and outputs b or c respectively.

When an FST computes on an input string $s := \overline{s_0 s_1 \ldots s_{n-1}}$ of length n, it takes the input symbols $s_0, s_1, \ldots, s_{n-1}$ one by one, starting from the starting state, and produces corresponding output symbols. For instance, the input string abccba produces the output string bcacbb, while cbaabc produces abbbca.

(a) Assume that FST_1 has an input alphabet Σ_1 and an output alphabet Γ_1, give a formal definition of this model and its computation. (Hint: An FST is a 5-tuple with no accepting states. Its transition function is of the form $\delta : Q \times \Sigma \to Q \times \Gamma$.)

Solution: Formal definition: $\mathsf{FST}_1 := (\Sigma_1, \Gamma_1, Q, \delta, s)$, where

- Σ_1 is the input alphabet,
- Γ_1 is the output alphabet,
- Q is the set of all states,
- $s \in Q$ is the start state,
- $\delta : Q \times \Sigma_1 \to Q \times \Gamma_1$ is the transition function. $\forall q_1, q_2 \in Q$, denote the transition between them as $a : b$, where $a \in \Sigma_1, b \in \Gamma_1$, and

$$\delta(q_1, a) = (q_2, b)$$

■

(b) Give a formal description of FST_0.

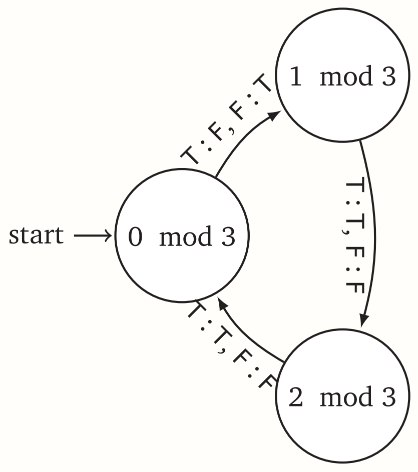
Solution: $\mathsf{FST}_0 := (\Sigma_0, \Gamma_0, Q_0, \delta_0, s_0)$, where

- $\Sigma_0 := \{a, b, c\}$,
- $\Gamma_0 := \{a, b, c\}$,
- $Q_0 := \{n_0, n_1\}$,
- $s_0 := n_0$ is the start state.
- $\delta_0 : Q_0 \times \Sigma_0 \to Q_0 \times \Gamma_0$ is defined as,

$$\begin{aligned} &\delta_0(n_0, a) = (n_0, b), \quad &&\delta_0(n_0, b) = (n_0, c), \quad &&\delta_0(n_0, c) = (n_1, a), \\ &\delta_0(n_1, a) = (n_1, a), \quad &&\delta_0(n_1, b) = (n_0, b), \quad &&\delta_0(n_1, c) = (n_1, c). \end{aligned}$$

■

(c) Give a state diagram of an FST with the following behavior. Its input and output alphabets are {T, F}. Its output string is inverted on the positions with indices divisible by 3 and is identical on all the other positions. For instance, on an input TFTTFTFT it should output FFTFFTTT.



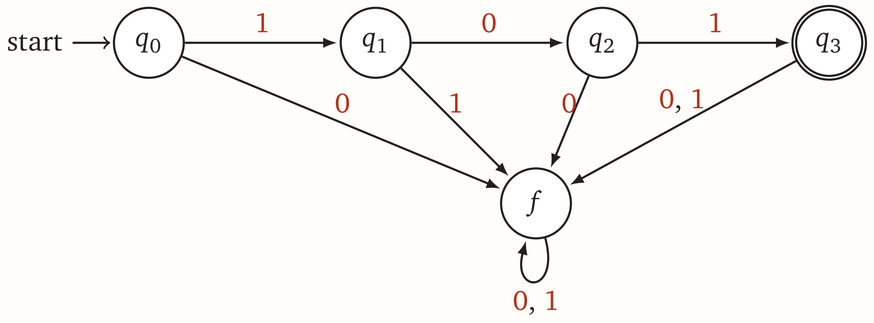
Solution:

■

5. **Another language transformation:** Given an arbitrary regular language L on some alphabet Σ, prove that it is closed under the following operation:

$$\mathsf{cycle}(L) := \{xy | x, y \in \Sigma^*, yx \in L\} \tag{2}$$

Solution: The given language $\mathsf{cycle}(L)$ is a set of strings that can be obtained by spliting a string $w \in L$ into two parts and swapping the order of the parts. As an example, if $L = \{101\}$, then $\mathsf{cycle}(L) = \{101, 011, 110\}$. To get the idea, consider the following DFA $M = (\Sigma, Q, s, A, \delta)$ for the langauge L.



Suppose we start from the state q_2 instead of q_0, traverse through the DFA to reach q_3, take an ϵ-transition to q_0, then continue traversal until reaching back to q_2. This traversal would represent the string 110, which is in $\mathsf{cycle}(L)$. Therefore, if we could start from an arbitrary state $q \in Q$ and traverse the DFA in a similar way as presented above, the traversals would represent the language $\mathsf{cycle}(L)$.

At a high-level, we construct an NFA with $|Q|$ different copies of a pair of M(therefore, it would be the total of $2|Q|$ copies of M). Each pair would correspond to a certain starting state, among all states in Q. For each pair, one copy of M corresponds to pre-cycle, and the other corresponds to post-cycle. We also add a pseudo start state s' that can ϵ-transition to one of the copies. Then, we modify the transition function so it allows the traversal explained above.

Formally, we construct NFA $M' := (\Sigma, Q', s', A', \delta')$, where

- $Q' := (Q \times Q \times \{pre, post\}) \cup \{s'\}$
- $A' := \{(q, q, post) \mid q \in Q\}$
- The transition function δ' is defined as follows,

$$\delta'(s', \epsilon) = \{(q, q, pre) \mid q \in Q\}$$

$$\delta'((q_i, q_j, pre), x) = \begin{cases} (q_i, s, post) & \text{if } q_j \in A, x = \epsilon \\ (q_i, \delta(q_j, x), pre) & otherwise \end{cases}$$

$$\delta'((q_i, q_j, post), x) = (q_i, \delta(q_j, x), post)$$

A state $q' = (q_i, q_j, pre)$, for an example, represents that the traversal started from q_i, so far the input string led to q_j, and we haven't cycled yet. Once we reach one of the original accepting states within a pre-cycle copy, we can take an ϵ-transition to the original starting state s of the corresponding post-cycle copy, and then continue traversal. We accept when we reach the state from which we started the traversal within the post-cycle copy. ■