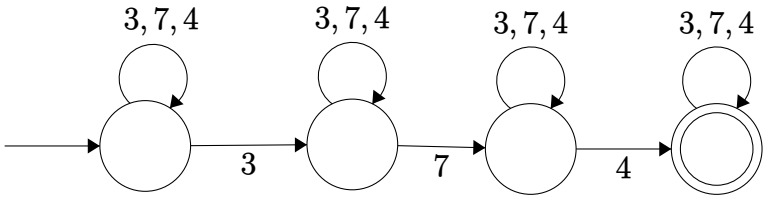
7 (100 PTS.) NFAs

For each of the following languages over $\Sigma = \{3, 7, 4\}$, draw an NFA that accepts them. Your NFA should have a small number of states (at most say 14 states). Provide a brief explanation for your solution.

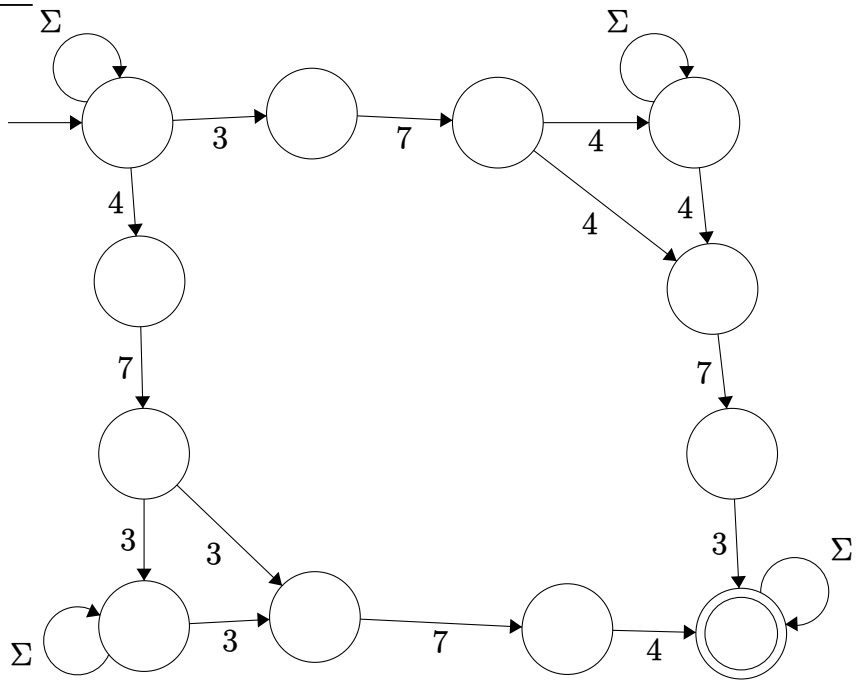
7.A. (20 PTS.) $\Sigma^* 3 \Sigma^* 7 \Sigma^* 4 \Sigma^*$

Solution:



7.B. (20 PTS.) All strings in Σ^* that contain the substrings 374 and 473.

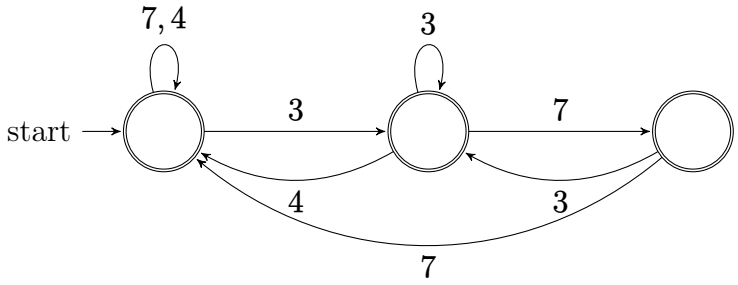
Solution:



An alternative solution is to build a DFA using product construction (over the two automatas accepting 374 and 473). It looks more complicated, but it has the same number of states as the NFA construction.

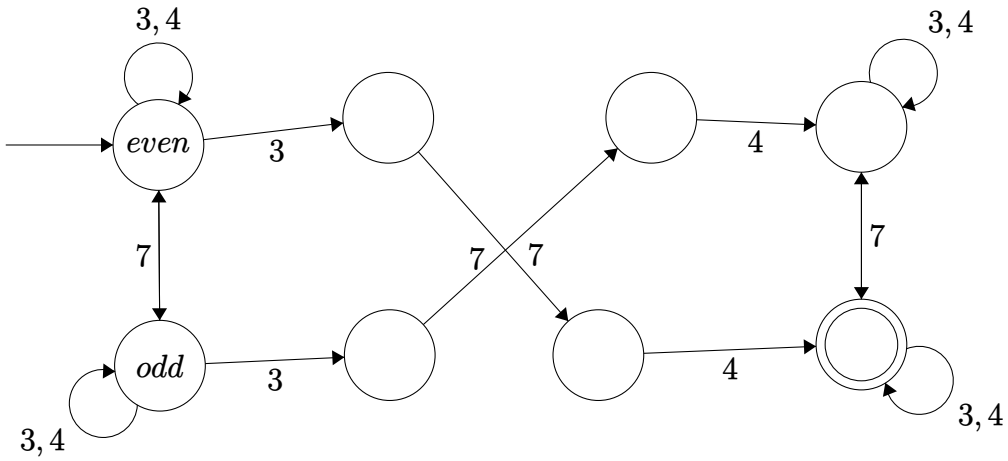
7.C. (20 PTS.) All strings in Σ^* that do not contain 374 as a substring.

Solution:



7.D. (20 PTS.) All strings in Σ^* that contain the substring 374 and an odd number of 7s.

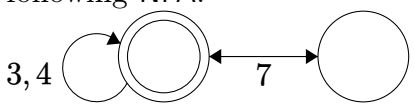
Solution:



This is essentially the product construction of the two natural DFAs accepting 374, and counting 7 modulate 2. Since it is an NFA, one can drop a few of the transitions.

7.E. (20 PTS.) All strings in Σ^* such that every maximal substring of consecutive 7s is even in size.

Solution: Consider the DFA that counts 7, and accept if their number is even. Now, observe that any transition from the odd state that is not 7, should lead to the reject state. Since we are building an NFA, we can just remove these transitions, and we get the following NFA.



8 (100 PTS.) DFAs to NFAs

Given a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts L, construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts the following languages. You can assume $\Sigma = \{0, 1\}$ in **8.A.** and **8.C.**. Provide a brief explanation for your solution.

8.A. (30 PTS.) $\textsf{DelOnes}(L) := \left\{0^{\#_0(w)} \;\middle|\; w \in L\right\}$; i.e., removes all 1s from the strings of L.

Solution:

The main idea is that we use epsilon transitions to emulate processing 1 symbols even without reading them. As such, the new NFA is defined as follows

- States: $Q' = Q$.
- Start state: $s' = s$.
- Accept states: $A' = A$.
- Transition function:

$$\delta'(q, c) = \begin{cases} \{\delta(q, 0)\} & c = 0 \\ \emptyset & c = 1 \\ \{\delta(q, 1)\} & c = \epsilon. \end{cases}$$

To see correctness, consider a string $w \in L$, and its walk in the original automata that accepts it. This corresponds to a walk in the resulting NFA that reads only the zeros of w. Similarly, any walk that accepts in the NFA, maps to an accepting walk in the DFA that uses 1 instead of ϵ-transitions.

8.B. (30 PTS.) $\textit{ThereAndBack}(L) := \left\{xy \;\middle|\; x \in L \text{ and } y^R \in L\right\}$

Solution:

We saw in the discussion section, how to construct an NFA N^R that accepts $\textsf{reverse}(L) = \left\{y^R \;\middle|\; y \in L\right\}$. Now, convert M and N^R into NFAs with normal form as seen in class – that is, with a unique start state that contains only an ϵ-transition out of it, and a unique accept state (which it is being transition into it is only via ϵ-transitions). Let N_1 and N_2 be the two resulting NFAs.

Now, we need to concatenate $L(N_1)$ and $L(N_2)$, and we had seen in class how to "concatenate" two such NFA, and the resulting NFA is the desired NFA.

Solution:

The main idea is we have to "guess" where x stops and y begins. To do this we will need two copies of the states Q. The other main idea is to use NFA to evaluate a machine in reverse. When we start processing y, we guess which accepting state y^R ends in, and step

backwards until we end at the start s. Formally, we have

$$
\begin{aligned}
Q' &= \{(q,0) \mid q \in Q\} \cup \{(q,1) \mid q \in Q\} \\
s' &= (s,0) \\
\delta'((q,\ell),c) &= \begin{cases} \{(\delta(q,c),0)\} & \ell = 0 \text{ and } c \neq \epsilon \\ \{(q',1) \mid q' \in A\} & \ell = 0, c = \epsilon \text{ and } q \in A \\ \{(q',1) \mid q', q \in Q \text{ s.t. } \delta(q',a) = q\} & \ell = 1 \text{ and } c \neq \epsilon. \end{cases} \\
A' &= \{(s,1)\}.
\end{aligned}
$$

8.C. (40 PTS.) $XOR(L) := \{z \mid z = \texttt{XOR}(x,y) \text{ for some } x \in L, y \in L, \text{ such that } |x| = |y| = |z|\}$, where $\texttt{XOR}(x,y)$ computes the element-wise XOR of x and y (so for each index i, $z_i = x_i \texttt{ XOR } y_i$).

Solution:

For each symbol $c = z_i$ we read, there are two possible choices of $a = x_i$ and $b = y_i$, e.g. if $z_i = 1$ then either $x_i = 0, y_i = 1$ or $x_i = 1, y_i = 0$. We will need to create a product construction to simulate the checking of x and y accordingly. We define $M' = (Q', s', \delta', A')$ as:

$$
\begin{aligned}
Q' &= Q \times Q \\
s' &= (s,s) \\
\delta'((q_x,q_y),1) &= \{(\delta(q_x,1),\delta(q_y,0)),(\delta(q_x,0),\delta(q_y,1))\} \\
\delta'((q_x,q_y),0) &= \{(\delta(q_x,0),\delta(q_y,0)),(\delta(q_x,1),\delta(q_y,1))\} \\
A' &= \{(q_x,q_y) \in Q' \mid q_x \in A, q_y \in A\}
\end{aligned}
$$

8.D. (**Not for submission**[1]) Consider, if you must, the language

$$Middle(L) := \{y \in L \mid xyz \in L \text{ for some } x,z \text{ such that } |x| = |y| = |z|\}.$$

Prove that this language is regular.

Solution:

We sketch a somewhat high level solution using closure properties. The next solution is more direct.

Let $M[q,q']$ be the DFA that accepts a string w, if $\delta(q,w) = q'$. Similarly, let

$$L_*[q,q'] = \{w \in \Sigma^* \mid \exists x \in \Sigma^*, |x| = |w|, \delta(q,x) = q'\}$$

[1] Do not read this problem. If you read it, do not think how to solve it. If you think how to solve it, do not write your solution down. If you write your solution, then do not submit your solution. Because if you submit your solution, we definitely are not going to read it. Not that you are going to have any way of submitting it anyway. Okay, hopefully you got the drift...

be the language of all the strings that if we ignore the exact characters in the string, then there is a way to traverse from q to q' in M using a string of the same length.

It is not hard to construct an NFA $N_*[q.q']$ for $L_*[q, q']$ from M.

Now, back to our original problem. Assume we guess the three states $q_x = \delta(s, x) \in Q$, and $q_{xy} = \delta(s, xy) \in Q$ and $q_{xyz} = \delta(s, xyz) \in A$. Then y is in all the three languages

$$L_*[s, q_x], \quad L[q_x, q_{xy}], \quad \text{and} \quad L_*[q_{xy}, q_{xyz}].$$

Namely y is in the intersection of the three languages. It is not hard to verify that the reverse also holds: If a word y is in the intersection, then it is in $Middle(L)$. Since all three languages are regular, the intersection is also regular, and we have that the language

$$L(q_x, q_{xy}, q_{xyz}) = L_*[s, q_x] \cap L[q_x, q_{xy}] \cap L_*[q_{xy}, q_{xyz}].$$

is regular. The desired language is in the finite union

$$Middle(L) = \bigcup_{q_x \in Q. q_{xy} \in Q, q_{xyz} \in A} L(q_x, q_{xy}, q_{xyz}).$$

Since regular languages are closed under finite union, it follows that this language is regular. Since regular languages have an NFA that accepts them the construction follows.

Implementing this proof to construct the NFA directly is not hard but somewhat tedious. The next solution is somewhat more direct (and thus we omit these low level details).

Solution:

Each time we read a symbol of y, we guess possible next symbols for x and for z. A gotcha here is that both y as well as xyz must be in the language. We define $M' = (Q', s', \delta', A')$ as:

$$Q' = (Q \times Q \times Q \times Q \times Q \times Q) \cup \{s_0\}$$

. The first state is for checking $y \in L$, the next three are for advancing x, y, z separately, and the last two are to store our guesses for the ending state after x and the ending state after xy.

$$s' = s_0$$

$$\delta'(s_0, \epsilon) = \{(s, s, q_y, q_z, q_{xend}, q_{zbegin}) \mid q_y \in Q, q_z \in Q, q_{xend} \in Q, q_{zbegin} \in Q\}$$

$$\delta'((q, q_x, q_y, q_z, q_{xe}, q_{zb}), b) = \{(\delta(q, b), \delta(q_x, a), \delta(q_y, b), \delta(q_z, c), q_{xe}, q_{zb}) \mid a \in \Sigma, b \in \Sigma\}$$

$$A' = \{(q, q_{xe}, q_{zb}, q_z, q_{xe}, q_{zb}) \mid q \in A, q_z \in A\}$$

9 (100 PTS.) Fooling Sets

Prove that the following languages are not regular by providing a fooling set. You need to provide an infinite set and also prove that it is a valid fooling set for the given language.

9.A. (20 PTS.) $L = \{ww^R w \mid w \in \{0,1\}^*\}$.

Solution: An infinite fooling set is $\{0^i 1 1 0^i\}$. Given $x = 0^i 1 1 0^i$ and $y = 0^j 1 1 0^j$, where $i < j$, we can distinguish with the suffix $z = 1 0^i$. The string xz is in the language by construction. yz is not, since even if $|yz|$ is a multiple of 3, then the first third of $|yz|$ must contain only 0 symbols, yet 1 occurs in the second or final third, hence yz cannot be decomposed as $ww^R w$.

9.B. (20 PTS.) $L = \{0^i 1 0^j \mid i \text{ is divisible by } j\}$.

Solution: An infinite fooling set is $\{0^i 1 | i > 0\}$. Given $x = 0^m 1$ and $y = 0^n 1$ (where without loss of generality assume $m < n$), the suffix 0^n distinguishes these since, $yz = 0^n 1 0^n$ is in the language (n divides n) while $xz = 0^m 1 0^n$ is not, since n cannot divide m.

9.C. (20 PTS.) $L = \{a^i b^j \mid i, j \in \mathbb{N}, \text{ and } j = \log_2 i\}$.

Solution: An infinite fooling set can be $\{a^{2^m} \mid m \geq 0\}$. Given $x = a^{2^m}$ and $y = a^{2^n}$ with $m \neq n$, we can distinguish with the suffix $z = b^m$. We have $xz = a^{2^m} b^m$ in the language by construction, but $yz = a^{2^n} b^m$ is not in the language since $2^n \neq 2^m$.

9.D. (20 PTS.) $L = \{0^i 0^j \mid i, j \in \mathbb{N}, \text{ and } j = \sqrt{i}\}$.

Solution: We can restate this language using $n = \sqrt{i}$, $j = (\sqrt{i})^2 = n^2$. As such, for $f(n) = n^2 + n$, $L = \{0^{f(n)} \mid n \geq 0\}$. The gap between consecutive values in this sequence increases with n, as

$$\Delta(n) = f(n+1) - f(n) = (n+1)^2 + (n+1) - (n^2 + n) = n^2 + 3n + 2 - n^2 - n = 2n + 2.$$

Our infinite fooling set is $\{0^i \mid i \geq 0\}$.
Now, given $x = 0^i$ and $y = 0^j$ with $i < j$, let $k = f(3j + 3i) - i$, and $z = 0^k$. Observe that the gap between $f(3j + 3i + 1)$ and $f(3j + 3i)$ is $\Delta(3j + 3i) \geq 6(j + i)$, and as such

$$0^i \cdot 0^k = 0^{f(3j+3i)} \in L \qquad \text{and} \qquad 0^j \cdot 0^k = 0^{f(3j+3i)+j-i} \notin L,$$

since $j - i < \Delta(3j + 3i)$, which implies that $0^j \cdot 0^k$ lies somewhere in between $0^{f(3j+3i)}$ and $0^{f(3j+3i+1)}$ – two consecutive words of L. Thus this is an infinite fooling set for this language.

What is implies by the above is somewhat stronger – for any monotone increasing function $f(i)$, such that $\sum_{i \to \infty} (f(i+1) - f(i)) = \infty$, the language $\{0^{f(i)} \mid i \geq 0\}$ is not regular. And indeed, every semester, we give the same question, with a completely different f.

9.E. (20 PTS.) $L = \{wcd^{\#_a(w)} \mid w \in \{a,b\}^*\}$.

Solution: An infinite fooling set is $a^i c = \{a^i c \mid i \geq 0\}$. For $i \neq j$, consider the strings $x = a^i c$ and $y = a^j c$. A distinguishing suffix is $z = d^i$. We have $xz = a^i c d^i$ in the language, but $yz = a^j c d^i$ is not in the language.