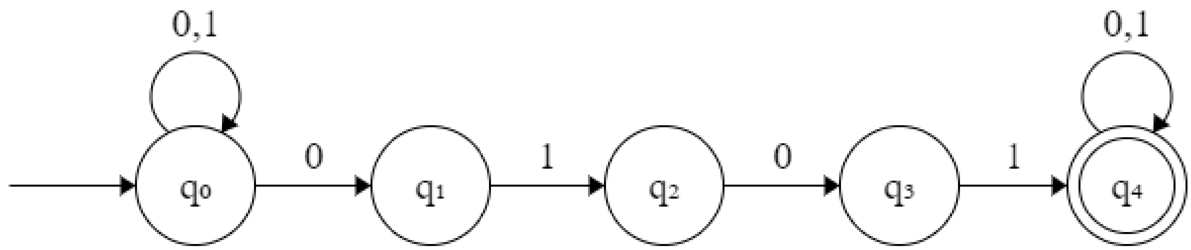
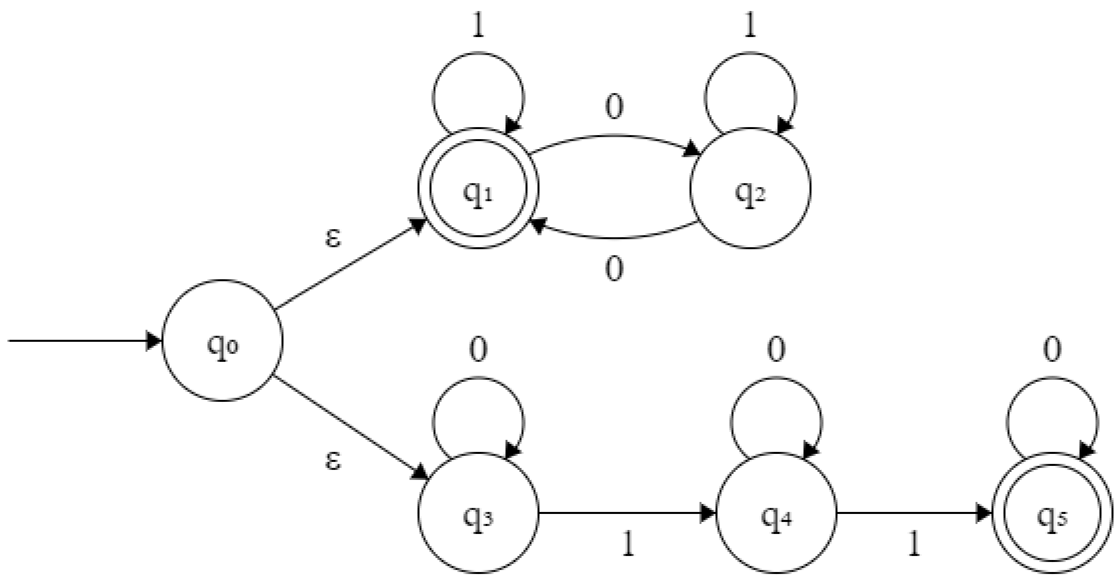
CMPS 130 Homework 3

Textbook Exercises

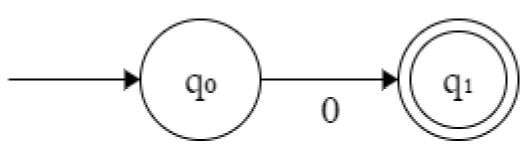
1.7 b. {w| w contains the substring 0101}



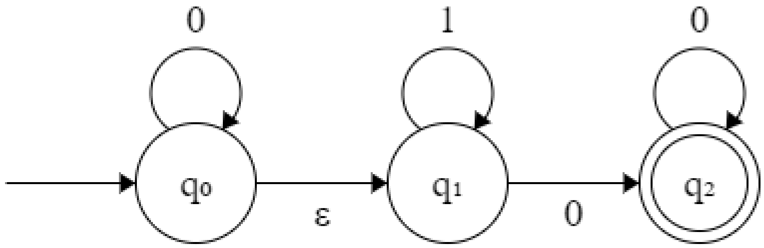
c. {w| w contains an even number of 0s, or contains exactly two 1s}



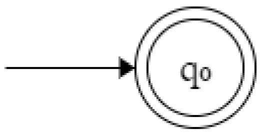
d. The language {0} with two states



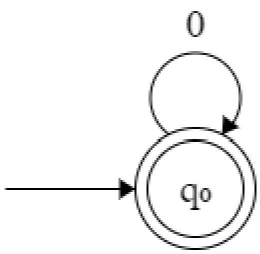
e. The language $0^*1^*0^+$ with three states



g. The language {ε} with one state



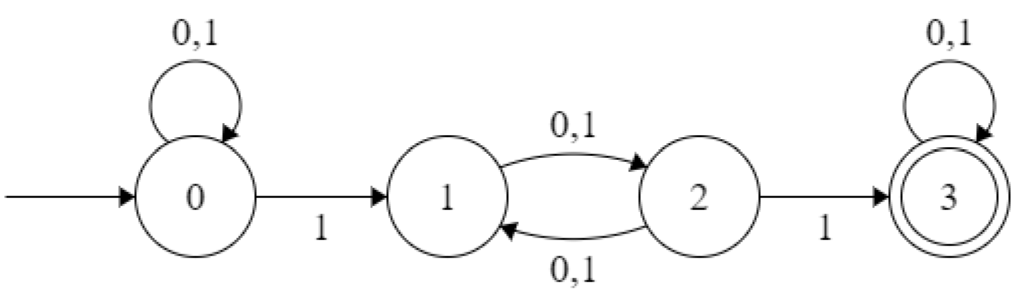
h. The language 0* with one state



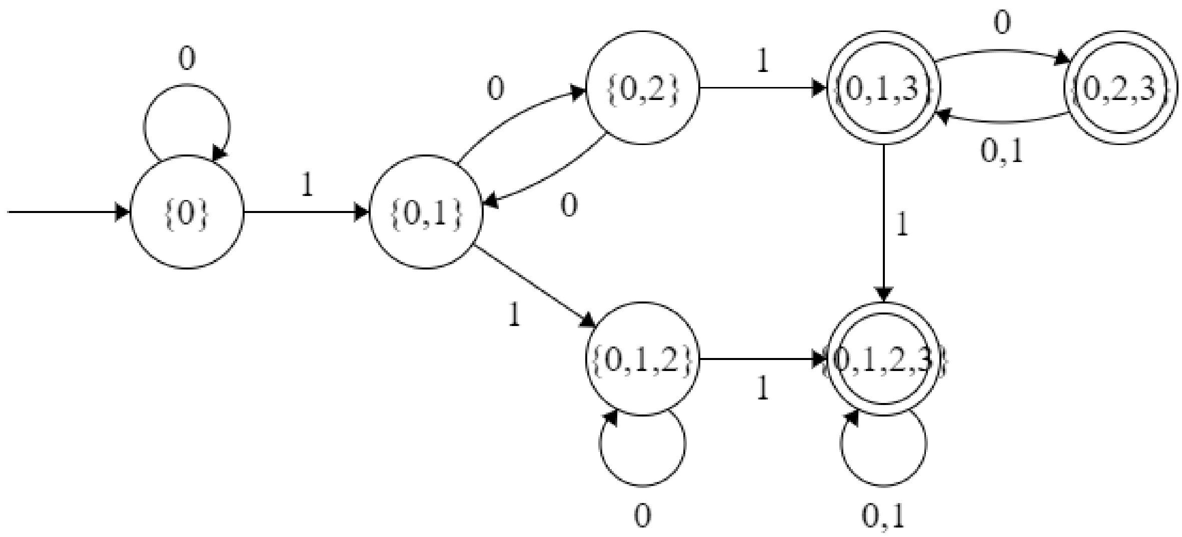
1.13. $F = \{w \in \{0,1\}^* \mid$ w contains a pair of 1s that are NOT separated by an odd number of symbols$\}$

$\overline{F} = \{w \in \{0,1\}^* \mid$ w contains a pair of 1s that are separated by an odd number of symbols$\}$

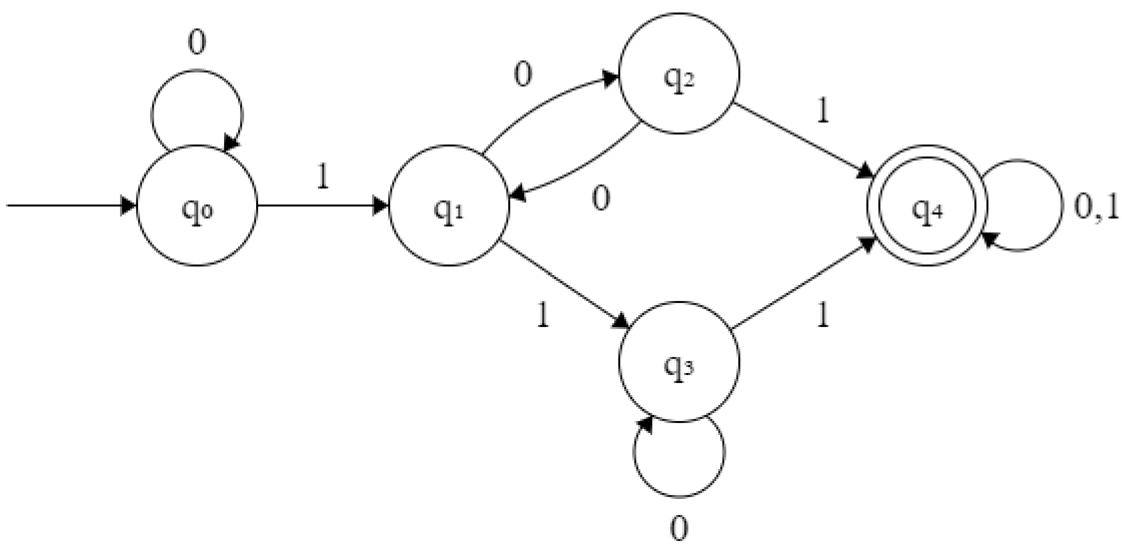
Below is an NFA for $\overline{F}$:



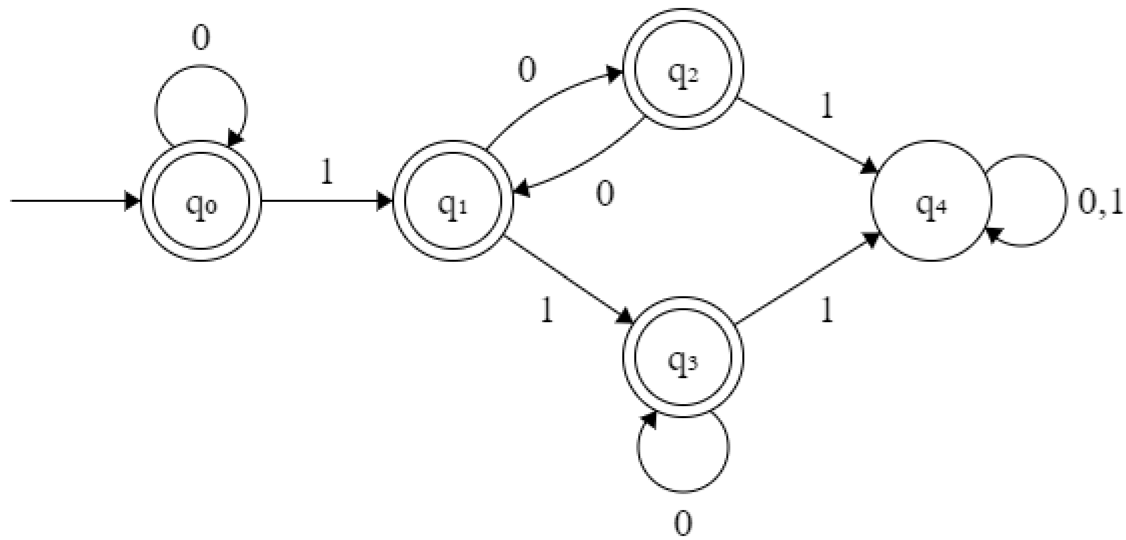
Using the subset construction and only drawing reachable states, we can generate a DFA for $\overline{F}$ from this NFA:



By merging the accept states, we get a DFA with only 5 states for $\bar{F}$:

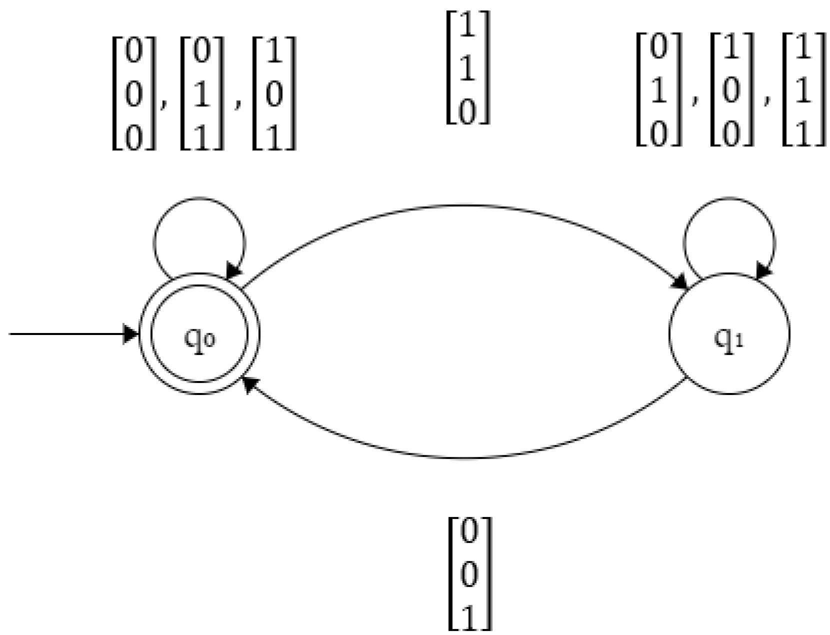


By the “complement construction”, below is the DFA for F:


0
1
0
q2
1
0
q0
q1
q4
0,1
1
1
q3
0

Textbook Problems

1.32. Let us consider an NFA that only accepts strings where each subsequent matrix $\begin{bmatrix}a\\b\\s\end{bmatrix}$ follows valid bit addition rules: $a + b = s$, and changes states with the existence of a carry bit.

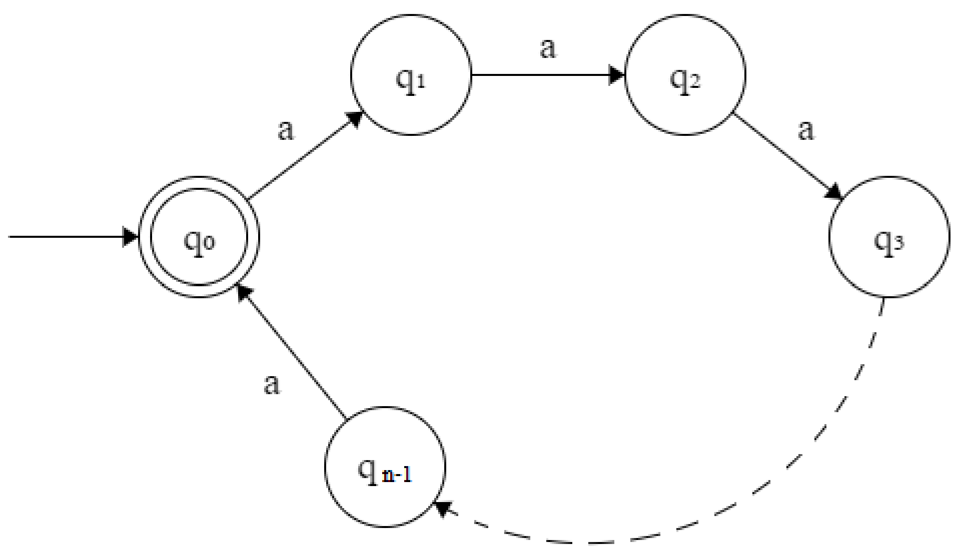


Notice that this NFA accepts all the strings in B^R and only the strings in B^R. Since there is an NFA that accepts B^R, B^R is a regular language. In addition, since the reversal of a language is regular, then $(B^R)^R = B$ is regular.

Note: my NFA argues that the empty string is accepted; if you argue that the empty string isn't accepted, then simply make a separate start state that transitions to q_0 on any input.

1.36. The NFA that accepts $B_n = \{a^k | \text{ k is a multiple of n}\}$ for each $n \geq 1$ is as follows: $N = (Q, \Sigma, \delta, s, F)$, where

$Q = \{q_i \mid 0 \leq i \leq n-1\}$	(there are n states where $q_i = q_{k \bmod n}$)
$a \in \Sigma$	(a is part of the finite alphabet)
$\delta(q_i, a) = q_{(i+1) \bmod n}$	(transition to the next stage in the "cycle")
$s = q_0$	(start with 0 a's; 0 mod n = 0)
$F = \{q_0\}$	(k is a multiple of n iff k mod n = 0)



Since there is an NFA that accepts B_n (the NFA described above), B_n is regular.

1.37. The NFA that accepts $C_n = \{x |$ x is a binary multiple of n$\}$ for each $n \geq 1$ is as follows: $N = (Q, \Sigma, \delta, s, F)$, where

$Q = \{q_i \mid 1 \leq i \leq n-1\}$	(there are n states where $q_i = q_{x \bmod n}$)
$\Sigma = \{0,1\}$	
$\delta(q_i, 0) = q_{(2i) \bmod n}$	(concatenating a 0 to a binary number is equivalent to multiplying it by 2)
$\delta(q_i, 1) = q_{(2i+1) \bmod n}$	(concatenating a 1 to a binary number is equivalent to multiplying it by 2 and adding 1)
$s = q_0$	(0 is a multiple of every natural number)
$F = \{q_0\}$	(x is a multiple of n iff $x \bmod n = 0$)

Transition function comments: you may have noticed that I am multiplying the current state (current "mod") by 2 (and adding 1 if scanning symbol "1"), instead of multiplying the value of the current string scanned. Let s = current string scanned. Then s is equivalent to some multiple of n added to s mod n:

$s = kn + s \bmod n$

$2s = 2kn + 2(s \bmod n)$, and $2s + 1 = 2kn + 2(s \bmod n) + 1$

Notice that 2kn is still a multiple of n, so there is nothing wrong with ignoring it.

Since there is an NFA that accepts C_n (the NFA described above), C_n is regular.

1.41. Let $A \subseteq \{a_1, \dots, a_k\}^*$ and $B \subseteq \{b_1, \dots, b_k\}^*$ be regular languages. The DFA that accepts $D_A = (Q_A, \Sigma_A, \delta_A, s_A, F_A)$ and the DFA that accepts $D_B = (Q_B, \Sigma_B, \delta_B, s_B, F_B)$.

The DFA that accepts the perfect shuffle of A and B: $\{w \mid w = a_1 b_1 \dots a_k b_k\}$ is as follows: $D = (Q, \Sigma, \delta, s, F)$, where

$Q = Q_A \times Q_B \times \{A, B\} \cup q_{reject}$
$\Sigma = \Sigma_A \cup \Sigma_B = \{a_1, \dots, a_k\} \cup \{b_1, \dots, b_k\}$
$\delta\big((q_A, q_B, A), a_i\big) = (\delta_A(q_A, a_i), q_B, B)$
$\delta\big((q_A, q_B, B), b_i\big) = (q_A, \delta_B(q_B, b_i), A)$
$\delta\big((q_A, q_B, A), b_i\big) = q_{reject}$
$\delta\big((q_A, q_B, A), b_i\big) = q_{reject}$
$\delta\big(q_{reject}, a_i\big) = q_{reject}$
$\delta\big(q_{reject}, b_i\big) = q_{reject}$
$s = (s_A, s_B, A)$
$F = F_A \times F_B \times \{A\}$

Picture the two DFAs for A and B (D_A and D_B), and two pebbles – one that starts in $s_A \in Q_A$ and $s_B \in Q_B$. The pebbles know that they must alternate their moves. First, the pebble in D_A scans the first symbol and moves to the corresponding state, then the pebble in D_B scans the next symbol and moves to the corresponding state, and so on and so forth. When the string has been fully read, it is accepted if both pebbles in D_A and D_B land on accept states. If at any time the DFAs read letters that aren't in their alphabet, they go to a reject state and stay there no matter what symbol they read next.

1.41. Note: Perhaps it is easier to visualize my answer when you assume A and B have the same alphabet. In that case, the DFA that accepts the perfect shuffle of A and B is as follows:

$D = (Q, \Sigma, \delta, s, F)$, where

$$Q = Q_A \times Q_B \times \{A, B\}$$
$$\Sigma = \Sigma_A = \Sigma_B$$
$$\delta\big((q_A, q_B, A), a_i\big) = (\delta_A(q_A, a_i), q_B, B)$$
$$\delta\big((q_A, q_B, B), b_i\big) = (q_A, \delta_B(q_B, b_i), A)$$
$$s = (s_A, s_B, A)$$
$$F = F_A \times F_B \times \{A\}$$