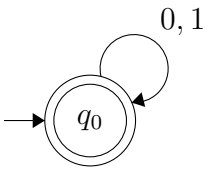
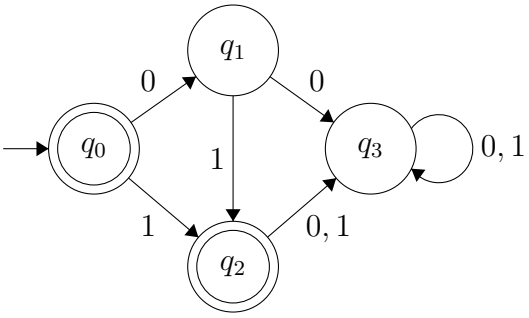
Solution sketch 1 - Computational Models - Fall 2017

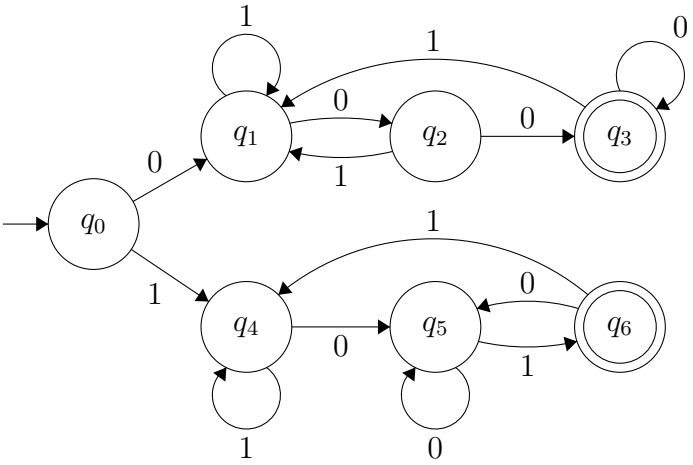
1. (a) Σ^*



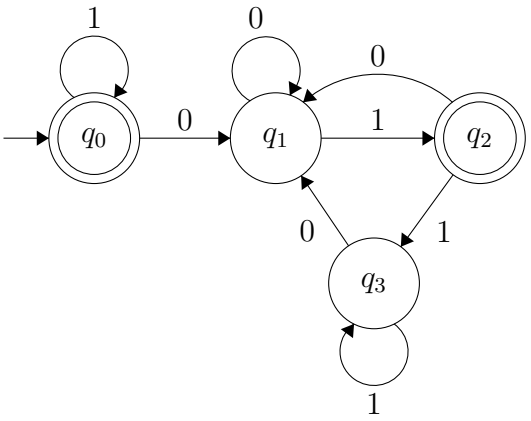
(b) $\{\varepsilon, 1, 01\}$



(c) $\{\sigma w 0 \sigma \mid \sigma \in \Sigma, w \in \Sigma^*\}$



(d) $\{w \mid w$ does not contain 0 or w ends with $01\}$

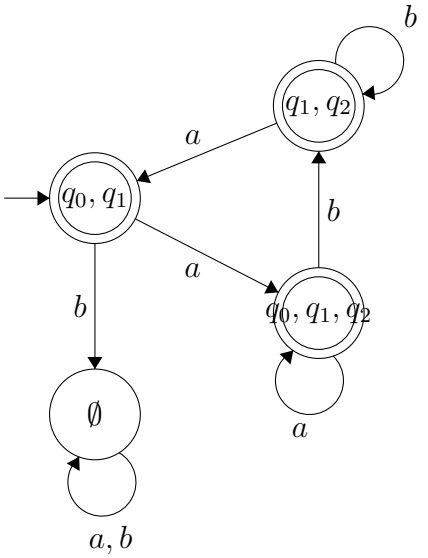


2. DFA: $A = (Q, \Sigma, \delta, q_0, F)$. $Q = \{x_1 x_2 \cdots x_n \mid \forall i. x_i \in \{0,1\}\}$, $\Sigma = \{0,1\}$, $q_0 = 1^n$, $F = \{x_1 x_2 \cdots x_n \mid x_1 = 0\}$, $\forall q \in Q, \sigma \in \Sigma, \delta(q,\sigma) = \delta(x_1 x_2 \cdots x_n, \sigma) = x_2 \cdots x_n \sigma$.

3. (a) $M = (Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}, \delta, q_0, \{q_1\})$, δ :

	a	b	ε
q_0	$\{q_2\}$	$\emptyset$	$\{q_1\}$
q_1	$\{q_0\}$	$\emptyset$	$\emptyset$
q_2	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$

(b) The equivalent DFA



4. (a) $1^*01^*01^*01^*01^*$
(b) $(0 \cup 1)^*000(0 \cup 1)^*$

5. Can be proved using the claim:

   Claim. *For any* $w \in \Sigma^*$ *and* $q \in Q$ *it holds that if* $q \in \widehat{\delta}(S, w) \Rightarrow \exists a = (a_1 a_2 \dots a_k) \in (\Sigma_\varepsilon)^k$ *and* $r_0, \dots, r_k \in Q$ *s.t.,*

   - $w = d(a)$
   - $r_0 \in S$
   - $r_k = q$
   - $r_{i+1} \in \delta(r_i, a_{i+1})$, *for all* $0 \leq i < k$

   The above claim can proved by induction on word length.

6. Let $M_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B)$ be two DFAs that recognize A and B, respectively. We build an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes the $perfect\ shuffle$ of A and B.

   The key idea is to design N to alternately switch from running M_A and running M_B after each character is read. Therefore, at any time, N needs to keep track of (i) the current states of M_A and M_B and (ii) whether the next character of the input string should be matched in M_A or in M_B. Then, when a character is read, depending on which DFA should match the character, N makes a move in the corresponding DFA accordingly. After the whole string is processed, if both DFAs are in the accept states, the input string is accepted; otherwise, the input string is rejected. Formally, the NFA N can be defined as follows:

   - $Q = Q_A \times Q_B \times \{A, B\}$, which keeps track of all possible current states of M_A and M_B, and which DFA to match.
   - $q_0 = (q_0^A, q_0^B, A)$, which states that N starts with M_A in q_0^A, M_B in q_0^B, and the next character read should be in M_A.
   - $\forall \sigma \in \Sigma, q_a \in A, q_b \in B$ δ is as follows:
     - $\delta((q_a, q_b, A), \sigma) = \{(\delta_A(q_a, \sigma), q_b, B)\}$, which states that if current state of M_A is q_a, the current state of M_B is q_b, and the next character read is in M_A, then when σ is read as the next character, we should change the current state of M_A to $\delta_A(q_a, \sigma)$, while the current state of M_B is not changed, and the next character read will be in M_B.
     - Similarly, $\delta((q_a, q_b, B), \sigma) = \{(q_a, \delta_B(q_b, \sigma)A)\}$.
   - $F = F_A \times F_B \times \{A\}$, which states that N accepts the string if both M_A and M_B are in accept states, and the next character read should be in M_A (i.e., last character was read in M_B).

You should prove that $L(N)$ is equal to the $perfect\ shuffle$ of A and B.