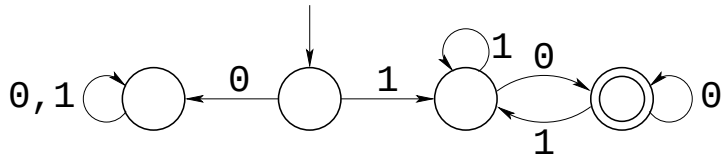
1. **(25 points)**: (from Sipser, problem 1.6) Give state diagrams of DFAs recognizing the following languages. In all parts the alphabet is $\{0, 1\}$.

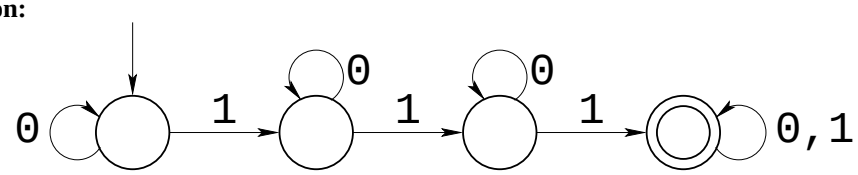
   (a) $\{w \mid w$ begins with a 1 and ends with a $0\}$.

   Solution:



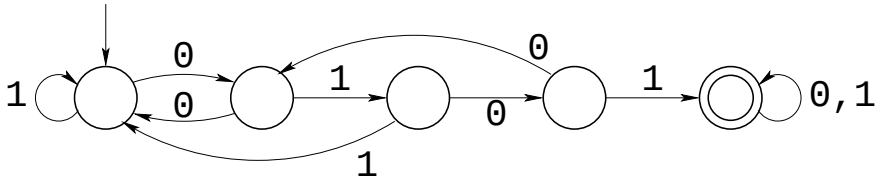
   (b) $\{w \mid w$ contains at least three 1s$\}$.

   Solution:



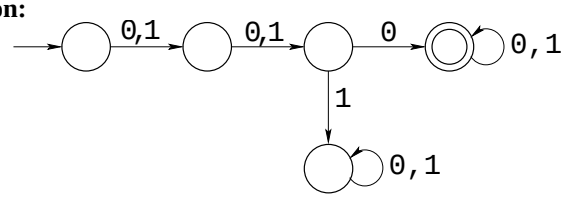
   (c) $\{w \mid w$ contains the substring 0101, i.e., $w = x0101y$ for some x and $y\}$.

   Solution:



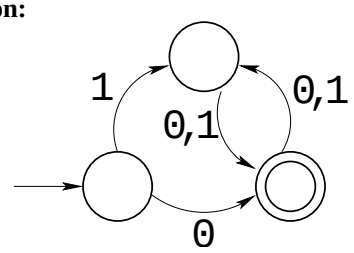
   (d) $\{w \mid w$ has length at least 3 and its third symbol is a $0\}$.

   Solution:



   (e) $\{w \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$.

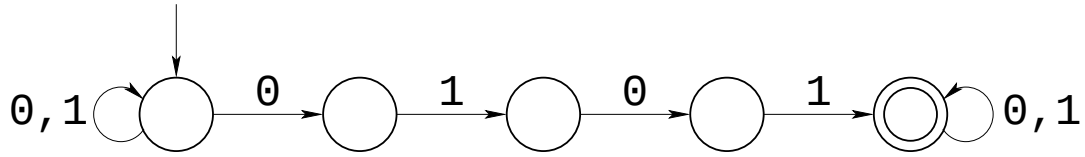
   Solution:



2. **(30 points)**: (from Sipser, problem 1.7) Give state diagrams of NFAs with the specified number of states recognizing each of the following languages. In all parts the alphabet is $\{0, 1\}$.

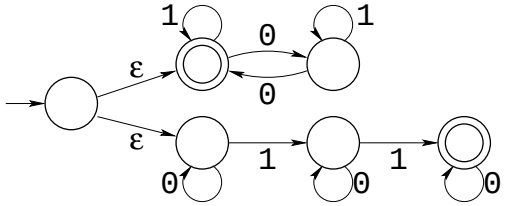
   (a) The language of $\{w \mid w$ contains the substring 0101, i.e., $w = x0101y$ for some x and $y\}$ with five states.

Solution:

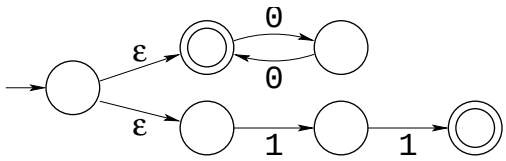


(b) The language of $\{w \mid w$ contains an even number of 0s, or contains exactly two 1s$\}$ with six states.

Solution: I took the phrase "contains an even number of 0s" to mean that any string with an even number of zeros and any number of ones is in the language. Likewise, I took "or contains exactly two ones" to mean any string with exactly two ones and any number of zeros. Here's my NFA.

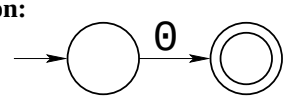


You could also take "an even number of 0s" to mean an even number of zeros and nothing else (i.e. no ones) and likewise for "any number of 1s." The NFA for that interpretation is the same as the one above without the self-loops:



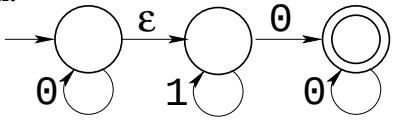
(c) The language $\{0\}$ with two states.

Solution:



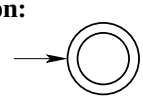
(d) The language $0^*1^*0^+$ with three states.

Solution:



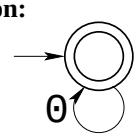
(e) The language $\{\epsilon\}$ with one state.

Solution:



(f) The language 0^* with one state.

Solution:



Note: 0^* is a string of *zero* or more 0s.
0^+ is a string of *one* or more 0s.

3. **(25 points)**: Closure properties of regular languages.

(a) **(10 points)** Prove that the regular languages are closed under complement. In other words, show that if L is a regular language, then the language $\overline{L}$ is regular as well.

Solution 1: Construct a DFA.

Let L be a regular language and $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes L. Let $\overline{M} = (Q, \Sigma, \delta, q_0, \overline{F})$. $\overline{M}$ recognizes $\overline{L}$.

Proof 1: As Sipser would say, the construction is obviously correct. Since this is good enough for Sipser, it's good enough for a solution in this class.

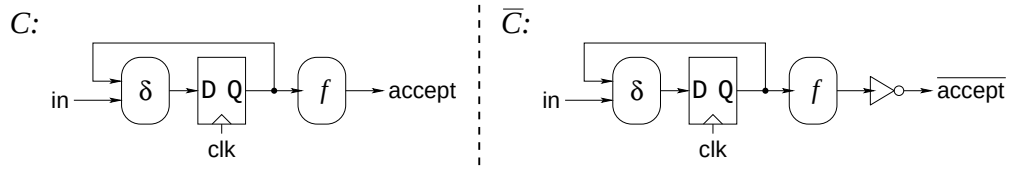
Proof 2: Let $w \in \Sigma^*$ be a string.

$$\begin{aligned} & w \in \overline{L} \\ \Leftrightarrow \quad & w \notin L \\ \Leftrightarrow \quad & w \notin L(M) \\ \Leftrightarrow \quad & \delta(q_0, w) \notin F \\ \Leftrightarrow \quad & \delta(q_0, w) \in \overline{F} \\ \Leftrightarrow \quad & w \in L(\overline{M}) \end{aligned}$$

Thus, DFA $\overline{M}$ recognize language $\overline{L}$. Because $\overline{L}$ is recognized by a DFA, $\overline{L}$ is regular.

Solution 2: Construct a Sequential Circuit.

Let C be a sequential circuit that asserts its accept output after reading a string in L. Use an inverter to produce $\overline{\text{accept}}$. Call the resulting circuit $\overline{C}$. $\overline{C}$ is a sequential circuit, and it recognizes $\overline{L}$. As Sipser would say, the correctness of the construction is obvious.



(b) **(15 points)** Prove that the regular languages are closed under intersection. In other words, show that if L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is regular as well.

Solution 1: Construct a DFA.

Let $L_1, L_2 \subseteq \Sigma^*$ be regular languages. Let $M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ be a DFAs that recognize L_1 and L_2 respectively. As in the proof (from class or Sipser) that regular languages are closed under union, we will construct a product machine. Let $M_\cap = (Q_\cap, \Sigma, \delta_\cap, q_{0,\cap}, F_\cap)$ be a DFA where

$$\begin{aligned} Q_\cap &= Q_1 \times Q_2 \\ \delta_\cap((q_1, q_2), c) &= (\delta_1(q_1, c), \delta_2(q_1, c)) \\ q_{0,\cap} &= (q_{(0,1)}, q_{(0,2)}) \\ F_\cap &= \{(q_1, q_2) \mid (q_1 \in F_1) \wedge (q_2 \in F_2)\} \\ &= F_1 \times F_2 \end{aligned}$$

$M_\cap$ recognizes $L_1 \cap L_2$.

Proof 1: As Sipser would say, the construction is obviously correct. Since this is good enough for Sipser, it's good enough for a solution in this class.

Proof 2: Let $w \in \Sigma^*$ be a string.

	$w \in L_1 \cap L_2$	
$\Leftrightarrow$	$(w \in L_1) \wedge (w \in L_2)$	def. intersection
$\Leftrightarrow$	$(w \in L(M_1)) \wedge (w \in L(M_2))$	choice of M_1 and M_2
$\Leftrightarrow$	$\delta_1(q_{0,1}, w) \in F_1) \wedge \delta_1(q_{0,2}, w) \in F_2)$	def. L(DFA)
$\Leftrightarrow$	$(\delta_1(q_{0,1}, w), \delta_1(q_{0,2}, w)) \in F_\cap)$,	def. $F_\cap$
$\Leftrightarrow$	$\delta_\cap((q_{0,1}, q_{0,2}), w) \in F_\cap)$,	see lemma 1 below
$\Leftrightarrow$	$\delta_\cap(q_{0,\cap}, w) \in F_\cap)$,	def. $q_{0,\cap}$
$\Leftrightarrow$	$w \in L(M_\cap)$,	def. L(DFA)

Thus, $L(M_\cap) = L_1 \cap L_2$. Because $L_1 \cap L_2$ is recognized by a DFA, $L_1 \cap L_2$ is regular.
Now, I left one detail for a lemma:

$$(\delta_1(q_{0,1}, w), \delta_1(q_{0,2}, w)) \quad = \quad \delta_\cap((q_{0,1}, q_{0,2}), 2)$$

I'll accept a solution that states this is obvious (or just assumes it). The formal proof is by induction on w:

Base case, $w = \epsilon$:

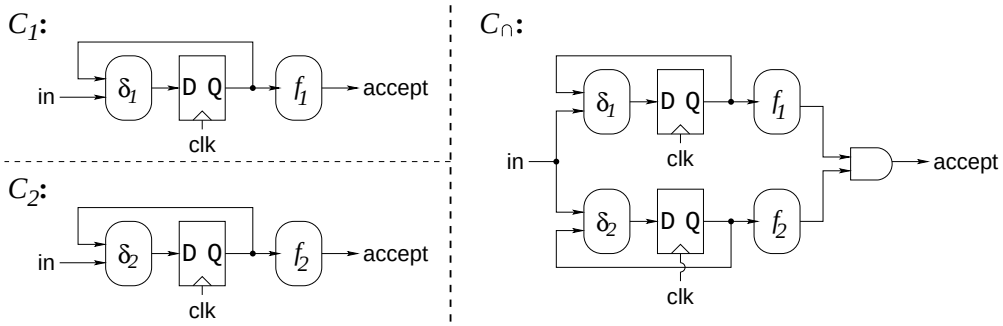
$$\begin{aligned}(\delta_1(q_{0,1}, w), \delta_1(q_{0,2}, w)) &= (\delta_1(q_{0,1}, \epsilon), \delta_2(q_{0,2}, \epsilon)) \\ &= (q_{0,1}, q_{0,2}) \\ &= q_{0,\cap} \\ &= \delta_\cap(q_{0,\cap}, \epsilon)\end{aligned}$$

Induction step, $w = x \cdot c$, where $x \in \Sigma^*$ and $c \in \Sigma$:

$(\delta_1(q_{0,1}, w), \delta_1(q_{0,2}, w))$	$=$	$(\delta_1(q_{0,1}, x \cdot c), \delta_2(q_{0,2}, x \cdot c))$,	$w = x \cdot c$
	$=$	$(\delta_1(\delta_1(q_{0,1}, x), c), \delta_2(\delta_2(q_{0,1}, x), c))$,	def. $\delta(q, string)$
	$=$	$\delta_\cap((\delta_1(q_{0,1}, x), \delta_2(q_{0,2}, x)), c)$,	def. $\delta_\cap$
	$=$	$\delta_\cap(\delta_\cap((q_{0,1}, q_{0,2}), x), c)$,	induction hypothesis
	$=$	$\delta_\cap(\delta_\cap(q_{0,\cap}, x), c)$,	def. $q_{0,\cap}$
	$=$	$\delta_\cap(q_{0,\cap}, w)$,	def. $\delta(q, string)$

Solution 2: Construct a Sequential Circuit.

Let C_1 be sequential circuit that asserts its accept output after reading a string in L_1 and likewise for C_2 and L_2. Combine the outputs of these two machines with an AND-gate. Call the resulting circuit $C_\cap$. $C_\cap$ is a sequential circuit taht recognizes $L_1 \cap L_2$. As Sipser would say, the correctness of the construction is obvious.



Solution 3: Use Closure Properties.

Let L_1 and L_2 be regular languages. Let $L' = \overline{\overline{L_1} \cup \overline{L_2}}$. Because the regular languages are closed under union (shown in Sipser and lecture) and intersection (shown above), L' is regular. From De Morgan's Law, $L' = L_1 \cap L_2$. Therefore, the regular languages are closed under intersection.

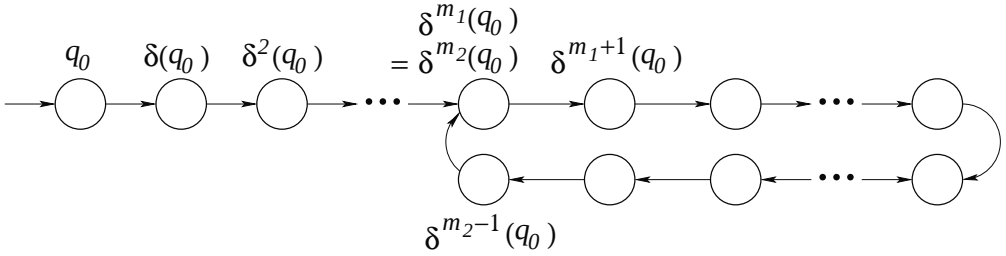


Figure 1: How a DFA with one input symbol goes into a loop

4. **(25 points)**: Let $M = (Q, \{\mathsf{a}\}, \delta, q_0, F)$ be a finite automaton. Note that the alphabet for M has only one symbol, a. All strings in $\{\mathsf{a}\}^*$ have the form a^m for some $m \in \mathbb{N}$.

   Prove that there are sets $A, B \subset \mathbb{N}$, and an integer, k, such that:

   - $(a^m \in L(M)) \quad \Leftrightarrow \quad (m \in A) \vee (\exists i \in B.\ \exists j \in \mathbb{N}.\ m = i + j * k)$.
     In English, this says that the length of any string in $L(M)$ is either given by an element of A or is the sum of an element of B and a multiple of k.
   - Every element of A or B is at most $|Q|$.
   - Likewise, $k \leq |Q|$.

   Hint: Think of what the state-transition diagram for M must look like, and consider what M does while reading a string, a^m.

   Solution: Let $M = (Q, \{\mathsf{a}\}, \delta, q_0, F)$ be a DFA whose input alphabet is $\{\mathsf{a}\}$. Note that each state, q, of M has exactly one successor state, $\delta(q, \mathsf{a})$. Because the input symbol is always a, I'll abbreviate $\delta(q, \mathsf{a})$ as $\delta(q)$. I'll write $\delta^2(q)$ to mean $\delta(\delta(q))$, and $\delta^k(q)$ means k applications of δ to q.

   Because Q is finite, we can find m_1 and m_2 such that $m_1 < m_1$ and $\delta^{m_1}(q) = \delta^{m_2}(q)$. Let $k = m_2 - m_1$. Figure 1 illustrates the operation of the DFA and the choice of m_1 and m_2. For $n \geq m_1$ we note that for any $j \in \mathbb{N}$, $\delta^{n+j*k}(q_0) = \delta^n(q_0)$ (I'll give a formal proof below). Now, let

$$\begin{array}{rcl} A & = & \{i \mid (i < m_1) \wedge (\delta^i(q_0) \in F)\} \\ B & = & \{i \mid (m_1 \leq i < m_2) \wedge (\delta^i(q_0) \in F)\} \end{array}$$

   These are the A, B, and k required by the problem.

   Proof:

   $((n \in A) \vee (\exists i \in B.\ j \in \mathbb{N}.\ n = i + j * k)) \Rightarrow (\mathsf{a}^n \in L(M))$

   If $n \in A$, then

$$\delta(q_0, \mathsf{a}^n) = \delta^n(q_0) \in F$$

   by the definition of A, and $\mathsf{a}^n \in L(M)$. If $(\exists i \in B.\ j \in \mathbb{N}.\ n = i + j * k))$, then, choose $i \in B$ and $j \in \mathbb{N}$ such that $n = i + j * k$. Then,

$$\begin{array}{rcll} \delta(q_0, \mathsf{a}^n) & = & \delta^n(q_0) & \\ & = & \delta^{i+j*k}(q_0), & n = i + j * k \\ & = & \delta(i, q_0), & \text{lemma 2 below} \\ & \in & F, & i \in B, \text{ def. } B \end{array}$$

   Thus, $\mathsf{a}^n \in L(M)$ as required.

$(\mathsf{a}^n \in L(M)) \Rightarrow ((n \in A) \vee (\exists i \in B.\ j \in \mathbb{N}.\ n = i + j * k))$

Let $n \in \ltimes$ such that $\mathsf{a}^n \in L(M)$. If $n < m_1$, then $n \in A$ by the definition of A and the claim is satisfied. If $m_1 \leq n$, then choose i and j with $m_1 \leq i < m_2$ and $j \in \mathbb{N}$ such that $n = i + j * k$. (Such a choice is always possible. In particular, $i = m_1 + ((n - m_1) \bmod k)$ and $j = (n - i)/k$.) We now have

$$\begin{array}{rcll} \delta(q_0, \mathsf{a}^n) & = & \delta^n(q_0) & \\ & = & \delta^{i+j*k}(q_0), & n = i + j * k \\ & = & \delta(i, q_0), & \text{lemma 2 below} \\ & \in & F, & \mathsf{a}^n \in L(M) \end{array}$$

This means that $i \in B$ by the definition of B, and the claim is satisfied.

Lemma 2: Let $n \in \mathbb{N}$ with $n \geq m_1$ and let $j \in \mathbb{N}$. Then $\delta^{n+j*k}(q_0) = \delta^n(q_0)$.

Proof: It's sufficient to show that for $m_1 \leq n \leq m_2$, $\delta^{n+j*k}(q_0) = \delta^n(q_0)$. Here's why. Let's say we've got some arbitrary $n \geq m_1$. Then, we can let $n_0 = m_1 + ((n - m_1) \bmod k)$, and $j_0 = (n - n_0)/k$ (note that $n - n_0$ is a multiple of k. Once we've shown that $\delta^{n_0+j*k}(q_0) = \delta^{n_0}(q_0)$, we can conclude that

$$\begin{array}{rrcl} & \delta^{n_0}(q_0) & = & \delta^{n_0+j_0*k}(q_0), \\ & & = & \delta^n(q_0) \\ \text{and} & \delta^{n_0}(q_0) & = & \delta^{n_0+(j_0+j)*k}(q_0) \\ & & = & \delta^{n+j*k}(q_0); \\ \text{thus,} & \delta^{n+j*k}(q_0) & = & \delta^n(q_0) \end{array}$$

First, we consider the case where $n = m_1$. We need to prove that $\delta^{m_1+j*k}(q_0) = \delta_1^m(q_0)$. My proof is by induction on j.

Base case, $j = 0$: $\delta^{m_1+j*k}(q_0) = \delta^{m_1+0*k}(q_0) = \delta^{m_1}(q_0)$.

Induction step, assume for j, prove for $j + 1$:

$$\begin{array}{rcll} & = & \delta^{m_1+(j+1)*k}(q_0) & \\ & = & \delta^k(\delta^{m_1+j*k}(q_0)) & \\ & = & \delta^k(\delta^{m_1}(q_0)), & \text{induction hypothesis} \\ & = & \delta^{m_1+k}(q_0) & \\ & = & \delta^{m_2}(q_0), & \text{def. } k \\ & = & \delta^{m_1}(q_0), & \text{choice of } m_1 \text{ and } m_2 \end{array}$$

Now, I'll look at the general case. Let $r = n - m_1$.

$$\begin{array}{rcll} \delta^{n+j*k}(q_0) & = & \delta^{m_1+r+j*k}(q_0) & \\ & = & \delta^r(\delta^{m_1+j*k}(q_0)) & \\ & = & \delta^r(\delta^{m_1}(q_0)), & \text{shown above} \\ & = & \delta^{m_1+r}(q_0) & \\ & = & \delta^n(q_0) & \end{array}$$

This completes the proof of lemma 2.

Note: I'll accept a solution that makes the observation that the machine is in a loop and uses the length of that loop to define k. A solution does not have to state this as a lemma or give a proof to receive full credit. I'm gave a proof here for completeness.