1. **(10 points)** Draw the state diagram of the DFA that recognizes the language over $\Sigma = \{0, 1\}$

$$A = \{w \in \{0, 1\}^*: w \text{ does not contain the string } 1101 \text{ as a substring}\}.$$ 

For full credit your DFA should have no more than five states.

**solution:** The easiest way to solve this problem is with the complement language, $\overline{A} = \{w \in \{0, 1\}^*: w \text{ has } 1101 \text{ as a substring}\}$, then flip the accept/reject states.

![State Diagram](image)

2. **(10 points)** Draw the state diagram of a DFA over the alphabet $\Sigma = \{0, 1\}$ that recognizes the language

$$B = \{1^n0^m | n + m \text{ is an odd positive integer}\}.$$ 

For full credit your machine should have at most six states. Hint: this language can be seen as an intersection of two simpler languages.

**solution:** $B$ can be seen as the intersection of the following two languages:

$$B_1 = \{w \in \{0, 1\}^* : w \text{ has odd length}\}, B_2 = \{1^n0^m : n, m \geq 0\}.$$ 

These have the respective DFAs:
Now we use the product construction on the following two DFAs. This yields the answer below.

Note, we can combine the right most two nodes into one for a DFA with 5 states.

3. (10 points) Recall, for a language \( L \subseteq \Sigma^* \) its complement is the set of strings over \( \Sigma \) not in \( L \), denoted as \( \overline{L} = \{ w \notin L \} \subseteq \Sigma^* \). Let \( A \) be the language above and let \( C = \{ w \in \{0, 1\}^* : w \text{ has even length} \} \). Draw the state diagrams of the DFA of the following language. Hint: use the construction from the book proving regular languages are closed under union.

(a) \( A \cup C \).

(b) \( A \cap C \)

\textbf{solution:} The language for \( C \) is the same as \( \overline{B}_1 \):

We labeled the states in the DFAs for \( C \) and \( A \) from left to right in increasing order (the DFA for \( C \) has start state \( c_0 \), for example).

\textbf{The DFA for} \( A \cup C \) \textbf{is as below}.
4. (10 points) We first review some definitions.

- The concatenation of two languages \( L_1, L_2 \) over \( \Sigma \) is \( L_1 \circ L_2 = \{x_1 x_2 : x_i \in L_i\} \).
- Lastly, the language \( L^* = \{x_1 x_2 \ldots x_k : x_i \in L, k \geq 0\} \).

Let \( A \) and \( B \) be the languages above. Draw the NFA state diagrams of the following languages:

(a) \( \overline{A} \circ B \)
(b) \( (\overline{A})^* \circ B \)

**solution:** The first NFA is:
5. (10 points) In this problem we are going to construct one regular language over alphabet $\Sigma_2$ from
another regular language over the alphabet $\Sigma_1$. First, let

$$f : \Sigma_1 \cup \{\varepsilon\} \rightarrow \Sigma_2 \cup \{\varepsilon\}$$

be any function that maps the empty string to the empty string $f(\varepsilon) = \varepsilon$. We can extend the function to another function on the sets of strings over these alphabets

$$f^* : \Sigma_1^* \rightarrow \Sigma_2^*$$

by applying the function to each character separately

$$f^*(w) = f^*(x_1x_2 \ldots x_k) = f(x_1)f(x_2) \ldots f(x_k).$$

Here are some examples:

- $f^*$ given by $f : \{a, b, c\} \cup \{\varepsilon\} \rightarrow \{0, 1\} \cup \{\varepsilon\}$ where $f(a) = 0$, $f(b) = 1$, $f(c) = \varepsilon$ maps $acb$ to 01.
- $f^*$ given by $f : \{0, 1\} \cup \{\varepsilon\} \rightarrow \{0, 1\} \cup \{\varepsilon\}$ where $f(1) = 1$, $f(0) = 1$ maps the language $B$ from problem 2 to the language $f^*(B) = \{1^l : l \geq 1\}$.

Now let $f : \Sigma_1 \cup \{\varepsilon\} \rightarrow \Sigma_2 \cup \{\varepsilon\}$ be a function where $f(\varepsilon) = \varepsilon$. Prove that if $L \subset \Sigma_1^*$ is a regular language then its image under $f^*$, $f^*(L) = \{f^*(w) : w \in L\} \subseteq \Sigma_2^*$, is a regular language in $\Sigma_2^*$. Hint: show how, starting from a DFA for $L$, you can construct an NFA for $f^*(L)$. You can use the theorem from class saying that if an NFA recognizes a language then it is regular.

**solution:** Recall, a language is regular if and only if there exists an NFA or DFA that accepts it.

We prove this by applying the function to the edges of a DFA/NFA for $L$ over $\Sigma_1$. Call this DFA “D.” Apply the function $f(\cdot)$ to the edges of $D$ to form an NFA. Call this NFA “N.”

Say $w = x_1 \ldots x_n \in L$. Then, the path in $D$, the DFA for $L$, reading $w$ ends in an accept state.

By following the same edges on the NFA $N$, we get a path representing the string

$$f^*(w) = f(x_1) \ldots f(x_2)$$

ending in an accept state for the NFA!

Therefore, the NFA accepts $f^*(L)$. Now, we must make sure the NFA accepts no strings outside of $f^*(L)$. Say we pick out a path from the start state to the accept state in the NFA. This path represents a string, $y_1 \ldots y_m$. However, this string is a string of characters from the function:

$$y_i = f(x_i).$$

Looking back at the DFA for $L$, we have the same path! Therefore $y_1 \ldots y_m \in f^*(L)$ and $x_1 \ldots x_m \in L$.

For completeness, we give the DFA to NFA transformation in terms of their exact definitions.

The DFA $D = (Q, \Sigma_1, \delta, q_0, F)$ is transformed to the NFA $N = (Q, \Sigma_2, \delta_2, q_0, F)$. All that is left is to do is to define the function $\delta_2 : Q \times \Sigma_2 \rightarrow P(Q)$. This is $\delta_2(q, y) = \{\delta(q, x) : y = f(x)\}$. 