1 Question 1

Let $M$ be the DFA with input alphabet $\{a, b, c\}$ and transition function:

<table>
<thead>
<tr>
<th>State</th>
<th>Symbol</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>start</td>
<td>$q_1$</td>
<td>$q_2$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>final</td>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>final</td>
<td>$q_2$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

Using the GNFA method, produce a regular expression that describes the language recognized by $M$. Draw each GNFA produced in the process.

Solution: Here is the initial GNFA:
\[ R_1 = a \cup c(b \cup c)^* a \]

\[ R_2 = (b \cup c) \cup a(b \cup c)^* a \]

\[ R_3 = a(b \cup c)^* \]

\[ R_4 = \varepsilon \cup c(b \cup c)^* \]

Rip \( q_2 \):

\[ R_2(R_1)^* b \]

Rip \( q_0 \):

\[ (R_2(R_1)^* b)^* (R_3 \cup R_2(R_1)^* R_4) \]

So the final regex is: \( (R_2(R_1)^* b)^* (R_3 \cup R_2(R_1)^* R_4) \).

### 2 Question 2

Let \( M \) be the NFA with transition function:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition on ( \rightarrow )</th>
<th>0</th>
<th>1</th>
<th>( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 ) – start</td>
<td>( { q_1, q_5 } ) ( { q_0 } ) ( { q_2 } )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( { q_0, q_2 } )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \emptyset ) ( { q_2, q_3 } ) ( { q_3 } )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 ) – final</td>
<td>( \emptyset ) ( { q_1, q_4 } ) ( { q_4 } )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 ) – final</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( { q_3, q_5 } )</td>
<td>( \emptyset )</td>
<td>( { q_1, q_4 } )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

Using the powerset method from class, produce a DFA that is equivalent to \( M \). (Do not make all \( 2^6 = 64 \) states, rather make only those that you need! Explain your solution.)

**Solution:** Here is the NFA as a state diagram:
Here is the converted DFA:

The DFA was generated using the powerset construction as given in lecture. We compute the \( \varepsilon \)-closure of \( q_0 \) in the NFA as a state in the DFA. As long as there is an unfilled transition from a state \( S \) in the DFA, we compute what states the NFA could have reached from the states described in \( S \) (of the NFA), and then compute the \( \varepsilon \)-closure once again. We continue until there is no unfilled transition of the DFA.

3 Question 3

Using the GNFA method from class, produce a regular expression that describes the language recognized by the NFA \( M \) from Question 2. Give the steps involved. (You may start either with the original NFA or with the DFA produced in your answer to Question 2.)

Solution: We produce a regex for the produced DFA. Note that since there is a “dead” state in this DFA, we can remove it immediately for the purposes of the GNFA method (ripping it will not change the produced regex at the end). However, for correctness, we will add an explicit step to rip it.

Here is the starting GNFA:
Rip $\emptyset$:  

Rip 0234:
4 Question 4

Devise a method which, given a regular expression $R$ whose language is $L$, produces a regular expression whose language is exactly the complement of $L$. Show that your method is correct. Then apply your method to determine a regular expression for the complement of $\epsilon a$ with alphabet $\Sigma = \{a, b\}$. Show all steps.

Solution: The intended strategy is the following: (1) convert the regex to an NFA $N$, (2) convert $N$ to a DFA $D$, (3) convert $D$ into a DFA for its complement $\bar{D}$, and (4) convert $\bar{D}$ into an equivalent regex.

Step (1): convert the regex into NFA $N$:
Step (2): convert NFA $N$ into a DFA $D$:

Step (3): complement non-final and final states of $D$:

Step (4): convert $D$ into an equivalent regex:

Step (4a): create the starting GNFA:

Step (4b): rip state $\{2\}$:
Step (4c): rip state \( \{0,1,3\} \):

\[
\begin{array}{c}
\text{Step (4d): rip state } \emptyset: \\
\end{array}
\]

Thus the final regex is: \((b \cup a(a \cup b))(a \cup b)^*\) (i.e., all strings having at least one character, start with \(b\), or start with \(a\) and at least one more character).

However, in general, step (2) can possibly introduce many states for \(D\), which corresponds to, in step (4), the regex being extremely long. A question for thought: can one avoid step (2) (i.e., generate the complement NFA directly)?

5 Question 5

Let \(\Sigma = \{0,1\}\) and let \(w_1, w_2 \in \Sigma^+ = \Sigma\Sigma^*\). Let \(L_{w_1, w_2}\) be the language of all strings in \(\Sigma^*\) that have the same numbers of substrings equal to \(w_1\) and substrings equal to \(w_2\).

1. Is \(L_{0,1}\) regular? Show that your answer is correct.
   
   **Solution:** Consider \(L_{0,1} \cap 0^*1^*\); it is \(L = \{0^n1^n : n \geq 0\}\). If \(L_{0,1}\) were regular, then the intersection would be regular. However, we know from class that \(L\) is not regular, so therefore \(L_{0,1}\) is not regular.

2. Is \(L_{0,01}\) regular? Show that your answer is correct.
   
   **Solution:** If \(z \in L_{0,01}\), then every occurrence of 0 has a 1 immediately appear after it. Therefore, this language can be described by the regular expression \(1^*(01)^*\).

3. Is \(L_{0,001}\) regular? Show that your answer is correct.
   
   **Solution:** For a direct solution, if \(z \in L_{0,001}\), then there can be no occurrences of 0 in \(z\). If there are \(s \geq 1\) occurrences of 0 in \(z\), then there needs to be \(s\) occurrences of 001, which implies there are \(2s > s\) occurrences of 0, a contradiction. So the regular expression for this language is \(1^*\).

4. Is \(L_{10,01}\) regular? Show that your answer is correct.
   
   **Solution:** Suppose \(x \in L_{10,01}\). Then it must be the case that \(x\) starts with a 0 and ends with a 0, or starts with a 1 or ends with a 1, or is \(\varepsilon\). If not, without loss of generality suppose that \(x = 0u1\), and \(x\) is the shortest string in \(L_{10,01}\) with this property. If \(u \in L_{10,01}\), then since \(x\) was the shortest string with that property. If \(1\) \(u = \varepsilon\), then \(x = 01 \notin L_{10,01}\), a contradiction; \(2\) \(u = 0\ldots0\) so \(x = 00\ldots01 \notin L_{10,01}\), a contradiction (there is 1 more occurrence of 01 than 10); and \(3\) \(u = 1\ldots1\), which is similar to case \(2\). This is a similar proof if \(x = 1u0\) instead.

Call the language described above \(L = \{w : w\) starts with a 0 and ends with a 0, or starts with a 1 or ends with a 1, or is \(\varepsilon\}\). A DFA for \(L\) can be constructed on 5 states:
5. Is $L_{100,001}$ regular? Show that your answer is correct.

**Solution:** This is regular. Note that all strings of length at most 2 are in $L = L_{100,001}$. Think of a string in this language. If the string entirely consists of 0’s, we accept. Suppose we have seen at least two 0’s and finally encounter a 1 - we have more occurrences of 001 than 100, so we must now be in a non-final state $s$. If we are in state $s$, we need to see two 0’s in a row to get to a final state. The idea is symmetric for seeing 100’s.

A DFA for this language is: