(1) Let $M$ be the DFA with transition function:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition on 0</th>
<th>Transition on 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$ - start, final</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$ - final</td>
<td>$q_2$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_2$ final</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

Using the GNFA method from class, produce a regular expression that describes the language recognized by $M$. Show the steps involved.

**Answer:** The first step is to add a new start and final state, producing the following GNFA (Transitions on $\emptyset$ are omitted):

![GNFA Diagram 1](image1)

Then $q_2$ is removed resulting in the following GNFA (Note: removing the states in a different order will produce a different, but equivalent, regular expression):

![GNFA Diagram 2](image2)
Then $q_0$ is removed resulting in the following GNFA:

Finally $q_1$ is removed resulting in the final GNFA:

Therefore, a regular expression that describes the language of $M$ is

$$0^* \cup 0^* 1 (01^* 0 \cup 10^*)^* (\epsilon \cup 10^*).$$

(Note: If you removed the states in a different order, you would have obtained a different, but equivalent, final regular expression).
(2) Let \( M \) be the NFA with transition function:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition on 0</th>
<th>Transition on 1</th>
<th>Transition on ( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 ) – start</td>
<td>( {q_0, q_1} )</td>
<td>( \emptyset )</td>
<td>( {q_3} )</td>
</tr>
<tr>
<td>( q_1 ) – final</td>
<td>( \emptyset )</td>
<td>( {q_2} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \emptyset )</td>
<td>( {q_1} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( \emptyset )</td>
<td>( {q_4} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( \emptyset )</td>
<td>( {q_5} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( \emptyset )</td>
<td>( {q_6} )</td>
<td>( {q_3} )</td>
</tr>
<tr>
<td>( q_6 ) – final</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

Using the power set method from class, produce a DFA that is equivalent to \( M \).

**Answer:** The DFA that is equivalent to \( M \) generated by using the power set method is given below (Only states reachable from the start state are shown):

(3) Let \( M \) be the NFA in Question 2. Using the GNFA method from class, produce a regular expression that describes the language recognized by \( M \). Show the steps involved.

**Answer:** The first step is to add a new start and final state, producing the following GNFA (Transitions on \( \emptyset \) are omitted):

(3) Let \( M \) be the NFA in Question 2. Using the GNFA method from class, produce a regular expression that describes the language recognized by \( M \). Show the steps involved.

**Answer:** The first step is to add a new start and final state, producing the following GNFA (Transitions on \( \emptyset \) are omitted):
Then $q_2$, $q_4$ and $q_6$ are removed resulting in the following GNFA (Note: removing the states in a different order will produce a different, but equivalent, regular expression):

![GNFA Diagram](image)

Then $q_1$ and $q_5$ are removed resulting in the following GNFA:

![GNFA Diagram](image)

Next $q_3$ is removed resulting in the following GNFA:

![GNFA Diagram](image)

Finally $q_0$ is removed resulting in the final GNFA:

![GNFA Diagram](image)

Therefore, a regular expression that describes the language of $M$ is

$$0^* (0(11)^* \cup (11)^* 111)$$

(Note: If you removed the states in a different order, you would have obtained a different, but equivalent, final regular expression).
(4) Let $w \in \Sigma^*$. Write $w = w_1 \cdots w_n$ with $w_i \in \Sigma$ for $1 \leq i \leq n$. The reverse of $w$ is the string $w_n \cdots w_1$. For language $L \subseteq \Sigma^*$, the reverse of $L$ is $\{\text{reverse}(w) : w \in L\}$.

Show that $L$ is regular if and only if $\text{reverse}(L)$ is regular.

**Answer:** First, we'll assume $L$ is regular and show $\text{reverse}(L)$ is regular. Since $L$ is regular, there exists a DFA, $D = (Q, \Sigma, \delta, q_0, F)$ such that $L(D) = L$. We will construct an NFA $N$ to recognize $\text{reverse}(L)$ as follows:

- Add a new state to the states of $D$ and make this the start state of $N$. Add $\epsilon$-transitions from this new start state to all the final states of $D$. This allows a computation of $N$ on input $\text{reverse}(w)$ to reverse the computation of $D$ on input $w$, thereby effectively showing the transitions of $D$ from a state backwards to all possible ways the state could have been reached on the given input.

Since $L$ is regular and $\text{reverse}(L)$ is regular, there exists a DFA, $D = (Q, \Sigma, \delta, q_0, F)$ such that $L(D) = L$. We will use this DFA to construct an NFA $N$ that recognizes $\text{reverse}(L)$ as follows:

- First, we'll assume $L$ is a language and $\text{reverse}(L)$ is regular and show that $L$ is then regular. We have just shown for any regular language, the reverse of that language is regular. Therefore, since $\text{reverse}(L)$ is regular, $\text{reverse}(\text{reverse}(L))$ is regular. However, $\text{reverse}(\text{reverse}(L)) = L$ since $w = w_1w_2\cdots w_n \in L$ iff $\text{reverse}(w) = w_n\cdots w_2w_1 \in \text{reverse}(L)$ iff $\text{reverse}(\text{reverse}(w)) = w_1w_2\cdots w_n = w \in \text{reverse}(\text{reverse}(L))$. Therefore, $L$ is regular.

Thus, we have shown that $L$ is regular if and only if $\text{reverse}(L)$ is regular.

A more formal proof for the forward direction follows for those interested, but is not required. We will define the NFA $N = (Q \cup \{q_0\}, \Sigma, \delta', q_0', F_N = \{q_0\})$ with

$$
\delta_N(q, a) = \begin{cases} 
\{q' \in Q : \delta(q', a) = q\} & \text{if } q \in Q, a \in \Sigma \\
F & \text{if } q = q'_0, a = \epsilon \\
\emptyset & \text{otherwise}
\end{cases}
$$

Next we will show $L(N) = \text{reverse}(L)$. $\text{reverse}(w) = w_n\cdots w_2w_1 \in \text{reverse}(L)$ iff $w = w_1w_2\cdots w_n \in L$ (by definition of $\text{reverse}$) iff $w \in L(D)$ (since $L(D) = L$ by assumption) iff there exists an accepting computation $r_0r_1r_2\cdots r_n$ with $r_0 = q_0$,
\[ \delta(r_i, w_{i+1}) = r_{i+1} \text{ for } 0 \leq i \leq n-1, \text{ and } r_n \in F \text{ on input } w \text{ (by definition of DFA acceptance) iff } r_n \in \delta_N(r'_0, \epsilon) \text{ where } r'_0 = q'_0, r_{i-1} \in \delta_N(r_i, w_i) \text{ for } 1 \leq i \leq n, \text{ and } r_0 = q_0 \in F_N \text{ (by definition of } \delta_N \text{ above) iff } r'_0r_n \ldots r_2r_1r_0 \text{ is an accepting computation for } N \text{ on input } w_n \ldots w_2w_1 = \text{reverse}(w) \text{ (by definition of NFA acceptance) iff } \text{reverse}(w) \in L(N). \] Thus, \( L(N) = \text{reverse}(L) \) and therefore, \( \text{reverse}(L) \) is regular.

(5) (not to be graded) Sometimes students make the following argument.

(a) A DFA can only have a finite set of states.

(b) A DFA with \( n \) states cannot remember an integer value if it can take on any value between 0 and \( n \). Informally, a DFA could only ‘count’ up to a finite number less than its number of states.

(c) Therefore any language whose recognition requires counting the occurrences of a substring, where that number can be arbitrarily large, cannot be a regular language.

(d) Recognition of the language \( L = \{ w \in \{a, b\}^* : \text{the number of occurrences of } ab \text{ is the same as the number of occurrences of } ba \} \) involves counting occurrences of \( ab \) and \( ba \), and these counts can be arbitrary large.

(e) So \( L \) cannot be regular.

Is this a valid argument? Explain as precisely as you can why, or why not.

**Answer:** This argument is not valid. In fact \( L \) is regular, which we’ll show shortly. While points (a)-(c) above are valid, the assumption in point (d) that recognizing \( L \) “involves counting occurrences of \( ab \) and \( ba \)” is not correct and, therefore, the conclusion in (e) is invalid. To see why the assumption in (d) is not correct, we will assume \( w \in L \) and, without loss of generality, \( w \) starts with an \( a \) (the symmetrical arguments hold if \( w \) started with a \( b \)). If \( w \) eventually contains a \( b \) (that is an occurrence of \( ab \)), then \( w \) must contain another \( a \) (giving the countering occurrence of \( ba \)). Additionally, it is impossible to have two occurrences of \( ab \) without an intermediary occurrence of \( ba \) in the input string (since the alphabet only contains \( a \) and \( b \)). Therefore, to recognize \( L \) we can just ‘match’ the next occurrence of an \( ab \) in \( w \) with the following occurrence of \( ba \) and don’t have to try and count all occurrences as stipulated in (d). Since, whatever symbol \( w \) starts with, if it ever contains a change of symbol, then the starting symbol must eventually follow, we see that \( L \) ends up being the same as the language \( \{ w \in \{a, b\}^* : w \text{ starts and ends with the same symbol} \} \) (an NFA for the related language over \( \{0, 1\} \) is given in the solutions to Homework 1 question 2e). A DFA is given for \( L \) below. Thus, \( L \) is regular and we see (e) is invalid.