Individual Homework

Problem (1)

Start by making the NFAs for each part of the regular expression.

NFA for $aa$:

\[
\text{start} \quad q_0 \quad \xrightarrow{a} \quad q_1 \quad \xrightarrow{a} \quad q_2
\]

NFA for $abaa$:

\[
\text{start} \quad q_0 \quad \xrightarrow{a} \quad q_1 \quad \xrightarrow{b} \quad q_2 \quad \xrightarrow{a} \quad q_3 \quad \xrightarrow{a} \quad q_4
\]

NFA for $aaba$:

\[
\text{start} \quad q_0 \quad \xrightarrow{a} \quad q_1 \quad \xrightarrow{a} \quad q_2 \quad \xrightarrow{b} \quad q_3 \quad \xrightarrow{a} \quad q_4
\]
Combine the NFAs and make a new NFA that recognizes $aa \cup abaa \cup aaba$
(same procedure is formalized in 2.a)

To make an NFA that recognises $(aa \cup abaa \cup aaba)^*$ we need to apply the star operation to the previous NFA. This means adding a new start state (which is also an accepting state) and connecting it to the old start state with $\epsilon$. Additionally, we need to connect the accepting states to the new start state with $\epsilon$. 
Problem (2)

(a)

Similar to the second step of the solution to question 1, add a new start state $q_0$ such that $q_0$ is connected to the start states of $M_1$ and $M_2$ through $\epsilon$. This new NFA has $|Q_{M_1}| + |Q_{M_2}| + 1$ states which equals 51. The corresponding DFA would have at most $2^{51}$ states, as it can have at most states representing all the subsets of $Q_{M_1} \cup Q_{M_2} \cup q_0$.

(b)

We can have a DFA were each state represents which state in $M_1$ and which state in $M_2$ is the current state with the same input string (since $M_1$ and $M_2$ are DFAs there can only one current state for any given input string).

The DFA will have states $Q_1 \times Q_2$, meaning every state is a pair $(q_i, q_j)$ where $q_i \in Q_1$ and $q_j \in Q_2$. In this new DFA $(q_i, q_j)$ is an accepting state when either $q_i$ or $q_j$ is an accepting state in $Q_1$ and $Q_2$ respectively. For this new DFA $\delta(\alpha, (q_i, q_j)) = (q'_i, q'_j)$ iff $\delta_{M_1}(\alpha, q_i) = q'_i$ and $\delta_{M_2}(\alpha, q_j) = q'_j$. This DFA has $|Q_1| \times |Q_2| = 600$ states and decides $L_1 \cup L_2$ as we only end in an accepting state for some input string $s$ iff either $s$ is accepted by $M_1$ or by $M_2$.

(c)

For a given input string $s$, run $s$ on $M_1$ first, if $M_1$ accepts, accept $s$, otherwise run $s$ on $M_2$, if $M_2$ accepts, accept $s$, otherwise reject. This approach takes at most $2l$ steps where $l$ is the length of $s$. 
Group Homework

Problem (1)

Note the following accepting futures of the language $L$:

\[ \text{AccFut}_L(\epsilon) = \{a, b\}^*ab\{a, b\}^* \]
\[ \text{AccFut}_L(a) = b \cup \{a, b\}^*ab\{a, b\}^* \]
\[ \text{AccFut}_L(ab) = \{a, b\}^* \]

We note that the language $L$ has at least three distinct accepting futures and therefore by the Myhill-Nerode theorem any DFA that accepts $L$ must have at least three states.

Problem (2)

(a)

Below is a NFA that recognizes $\{0, 1\}^*1\{0, 1\}^3$.

A high level description of the DFA is that $q_0$ is the state corresponding to seeing arbitrarily many zeros and ones, or simply $\{0, 1\}^*$. Then transitioning to state $q_1$ corresponds to seeing a 1 which happens to be the fourth last element of the input string, and then states $q_2, q_3$, and $q_4$ count down to make sure that the element that caused the transition from state $q_0$ to $q_1$ is in fact the fourth last element of the string.

(b)
In part (a) we gave a five state NFA that recognizes \( \{0,1\}^*1\{0,1\}^3 \), where states \( q_2, q_3, \) and \( q_4 \) count the number of remaining elements in order to ensure that the 1 that triggered the transition from \( q_0 \) to \( q_1 \) is in fact the fourth last element of the string. If \( k = 0 \) then we simply have a two state NFA, taking the NFA in part (a) and using only states \( q_0 \) and \( q_1 \) and making \( q_1 \) the only accepting state. If \( k \geq 1 \) then we take the aforementioned two-state NFA and add states \( q_2, \ldots, q_{k+1} \) to the NFA with a deterministic transition for each \( i \in \{1, \ldots, k\} \) from \( q_i \) to \( q_{i+1} \) upon seeing a character in \( \{0,1\} \).

These states \( q_2, \ldots, q_{k+1} \) ensure that the 1 that triggered the transition from \( q_0 \) to \( q_1 \) is in fact the \( k+1 \) last element of the string. We finally make \( q_{k+1} \) the only accepting state of the NFA.

**Problem (3)**

(a)

We list the accepting futures of 00, 01, 10, and 11:

\[
\begin{align*}
\text{AccFut}_L(00) &= \{0,1\}^*1\{0,1\}^1 \\
\text{AccFut}_L(01) &= \{0,1\}^1 \cup \{0,1\}^*1\{0,1\}^1 \\
\text{AccFut}_L(10) &= \epsilon \cup \{0,1\}^*1\{0,1\}^1 \\
\text{AccFut}_L(11) &= \epsilon \cup \{0,1\}^1 \cup \{0,1\}^*1\{0,1\}^1
\end{align*}
\]

We note that the language \( L \) has at least four distinct accepting futures and therefore by the Myhill-Nerode theorem any DFA that accepts \( L \) must have at least four states.

Next, we show that any other accepting future for \( L \) must be equivalent to one of the four accepting futures listed above. We note that \( \text{AccFut}_L(\epsilon) = \text{AccFut}_L(0) = \{0,1\}^*1\{0,1\}^1 \), which is equivalent to the accepting future of 00. Similarly, \( \text{AccFut}_L(1) = \{0,1\}^1 \cup \{0,1\}^*1\{0,1\}^1 \), which is equivalent to the accepting future of 01.

As for an arbitrary string \( s \in \{0,1\}^* \) where the length of \( s \) is at least three, the accepting future of \( s \) depends only on the last two elements of \( s \) and \( \text{AccFut}_L(s) \) is equivalent to one of the accepting futures of 00, 01, 10, and 11. Therefore, the language \( L \) has exactly four distinct accepting futures and so by the Myhill-Nerode theorem there exists a DFA with four states that
recognizes \( L \). Hence, the minimum number of states for a DFA to recognize \( L \) is four.

(b)

Below is a DFA with four states that recognizes \( L \).

\[ q_0 \quad q_1 \quad q_2 \quad q_3 \]

\[ \begin{array}{ccc}
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
& & 1
\end{array} \]

- \( q_0 \) corresponds to the accepting futures of the strings \( \epsilon, 0 \), and those for which the last two elements are 00.
- \( q_1 \) corresponds to the accepting futures of the strings 1 and those for which the last two elements are 01.
- \( q_2 \) corresponds to the accepting futures of the strings for which the last two elements are 11.
- \( q_2 \) corresponds to the accepting futures of the strings for which the last two elements are 10.

(c)

We claim that the a DFA with minimal states that recognizes \( \{0, 1\}^1 \{0, 1\}^k \) has \( 1 + \sum_{i=0}^{k} 2^i \) states. We can construct such a DFA by starting with an initial state \( q_0 \) and transitioning to a state \( q_1 \) if we see a 1, otherwise if we see a 0 then we remain at state \( q_0 \). Using \( q_1 \) as a root we create a binary tree of
states of depth $k + 1$, where the depth includes the root $q_1$, and we transition down the binary tree, going to the right sub-node if we see a 1 and to the left sub-node if we see a zero. The leaves of this binary tree are the accepting states corresponding to all combinations of the last $k + 1$ elements such that the $k + 1$ last element is a 1. Omitting much of the details, we can create appropriate transitions back up the binary tree and to $q_0$ accordingly from the leaves. The non-leaf nodes of this binary tree correspond to accepting futures of the various classes of strings for which the $k + 1$ last element is not a 1. This binary tree has $\sum_{i=0}^{k} 2^i$ nodes, including $q_1$. 