**Problem 1**

(25 points) Give an NFA (both a state diagram and a formal description) recognizing the language \(0^*1^*0^+\) with three states. The alphabet is \(\{0, 1\}\).

**Solution.**

State Diagram:

```
0 \quad \epsilon \quad 1 \quad 0
\quad q_0 \quad q_1 \quad q_2
```

Formal Description:

\[ N = (Q, \Sigma, \delta, q_0, F) \] where

- \(Q = \{q_0, q_1, q_2\}\)
- \(\Sigma = \{0, 1\}\)
- \(\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q, \ (q, \sigma) \mapsto S \subseteq Q\) where:

\[
\delta(q, \sigma) =
\begin{cases}
\{q_0\} & \text{if } \sigma = 0 \text{ and } q = q_0 \\
\{q_1\} & \text{if } \sigma = \epsilon \text{ and } q = q_0 \\
\{q_1\} & \text{if } \sigma = 1 \text{ and } q = q_1 \\
\{q_2\} & \text{if } \sigma = 0 \text{ and } q = q_1 \\
\{q_2\} & \text{if } \sigma = 0 \text{ and } q = q_2 \\
\emptyset & \text{otherwise}
\end{cases}
\]

**Problem 2**

(25 points) This question studies the number of states in a DFA equivalent to an NFA. Recall that in class we showed an NFA with 4 states that recognizes the language which consists of all binary strings that have a 1 in the third position from the end. For any integer \(k\), it is easy to generalize this construction to an NFA with \(k + 1\) states that recognizes the language which consists of all binary strings that have a 1 in the \(k^{\text{th}}\) position from the end. The general transformation from an NFA to a DFA will give us a DFA with at most \(2^{k+1}\) states recognizing the same language. Show that, any DFA that recognizes the same language must have at least \(2^k\) states.

**Solution.** Assume for the sake of contradiction that a DFA exists which accepts the language described above. Consider the set of all binary strings with length \(k\), \(\{0, 1\}^k\). \(|\{0, 1\}^k| = 2^k\) since \(\{0, 1\}^k\) is a set of length \(k\) strings where each position in the string can take on two values, 0, or 1. Consider using a DFA with less than \(2^k\) states to compute each of these strings. By the pigeonhole principle, there must be two strings \(w_1\) and \(w_2\) that end on the same state, e.g., in the sequence of states visited in our DFA while computing the strings \(w_1\) and \(w_2\) are \(r_1^1, \ldots, r_1^k\) and \(r_0^1, \ldots, r_0^k\) respectively, where \(r_k^1 = r_k^2\). Let \(j\) be the first position where \(w_1\) and \(w_2\) differ, and consider the strings \(e_1 = w_1 \circ 1^j\) and \(e_2 = w_2 \circ 1^j\). Consider the sequence
of states visited in our DFA while computing these strings, \( r^1_0, \ldots, r^1_k, s^1_1, \ldots, s^1_j \) and \( r^2_0, \ldots, r^2_k, s^2_1, \ldots, s^2_j \). Since \( r^1_k = r^2_k \), and at this point in each computation we have consumed the first \( k \) symbols of each string and thus only have a string of \( j \)'s remaining, we must have that \( s^1_i = s^2_i \) for all \( i \in \{1, \ldots, j\} \) because of the determinism of the DFA we are working with. Thus, the computations of \( e_1 \) and \( e_2 \) end in the same state. This is impossible, since \( e_1 \) and \( e_2 \) have different symbols in their respective \( k \)th positions from the end, so one should be in an accept state and one should be in a reject state. Thus, no DFA with less than \( 2^k \) states can accept this language.

\[ \square \]

**Problem 3**

(25 points) Say that string \( x \) is a prefix of string \( y \) if string \( z \) exists where \( xz = y \) and that \( x \) is a proper prefix of \( y \) if in addition \( x \neq y \). Let \( A \) be a regular language. Show that the class of regular languages is closed under the following operation:

\[
\text{NOEXTEND}(A) = \{w \in A \mid w \text{ is not a proper prefix of any string in } A\}
\]

**Solution.** As \( A \) is regular, there is a DFA \( M = (Q, \Sigma, \delta, p_0, F) \) that recognizes \( A \).

We construct an DFA \( M' = (Q, \Sigma, \delta, p_0, F') \) that recognizes \( \text{NOEXTEND}(A) \). Notice that \( M' \) and \( M \) are only different in their sets of accept states.

We define \( F' \subseteq F \) as follows. For every state \( p \in F, p \) is in \( F' \) if and only if \( \forall q \in F \) (including the case \( q = p \)), there is no path from \( p \) to \( q \) in the diagram of \( M \).

Next we prove that \( M' \) recognizes \( \text{NOEXTEND}(A) \).

For every string \( x \in \text{NOEXTEND}(A) \), according to the definition of \( \text{NOEXTEND}(A) \), we know \( x \in A \). By reading \( x \), starting from \( p_0 \), \( M \) ends at a state \( u \in F \). As \( x \) is not the proper prefix of any string in \( A \), there is no path from \( u \) to any \( q \in F \) in the diagram of \( M \). So \( u \in F' \). As a result, \( x \) is accepted by \( M' \).

For every string \( x \) that is accepted by \( M' \). Starting from \( p_0 \), following \( x \), \( M' \) ends at a state \( u \in F' \). Since \( F' \subseteq F \), we have \( x \in A \). Suppose \( x \) is a proper prefix of a string \( y \in A \). Assume \( y = x \circ z \). As \( y \in A \), starting from \( u \), following \( z \), \( M \) ends at a state \( q \in F \). However this contradicts the definition of \( F' \). So \( x \) is not a proper prefix of any string in \( A \). So \( x \in \text{NOEXTEND}(A) \).

This proves that \( M' \) recognizes \( \text{NOEXTEND}(A) \).

\[ \square \]

**Problem 4**

(25 points). Let \( \Sigma = \{0, 1\} \).

1. Write a regular expression for the language \( L \) consisting of all strings in \( \Sigma^* \) with exactly one occurrence of the substring \( 111 \).
2. Write a regular expression for the language \( L \) consisting of all strings in \( \Sigma^* \) that do not end with \( 10 \).

**Solution.**

1. \[(0 \cup 10 \cup 110)^*111(0 \cup 01 \cup 011)^*\]

2. \[((0 \cup 1)^*(01 \cup 11 \cup 00)) \cup 0 \cup 1 \cup \varepsilon\]

\[ \square \]