1. In class, we proved the following theorem:

**Theorem.** If $M$ is an NFA accepting a language $L$, then there is an NFA $M'$ accepting the language $L^*$.

This shows that the set of regular languages is closed under the Kleene star operation. Consider the following “attempt” at constructing an NFA $M'$ from $M$ to prove this theorem.

Let $M = (\Sigma, Q_0, q_0, F_0, \delta_0)$ be a description of an NFA with the usual components (alphabet, states, start state, accept states, and transition function, respectively). Define $M' = (\Sigma, Q_1, q_1, F_1, \delta_1)$ as follows:

- $Q_1 = Q_0$: The states of $M'$ are the same as $M$.
- $q_1 = q_0$: The start state of $M'$ is the same as the start state of $M$.
- $F_1 = F_0 \cup \{q_1\}$: keep all the accepting states of $M$, and also make the start state an accepting state.
- Define $\delta_1 : Q_1 \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q_1)$ as follows. For each state $q \in Q_1$ and character $a \in \Sigma \cup \{\varepsilon\}$,

$$\delta_1(q, a) = \begin{cases} 
\delta_0(q, a) & \text{if } q \text{ is not an accepting state or } a \neq \varepsilon \\
\delta_0(q, a) \cup \{q_1\} & \text{if } q \text{ is an accepting state and } a = \varepsilon.
\end{cases}$$

That is, $\delta_1$ is the same as $\delta_0$ except we add $\varepsilon$ transitions from the final states of $M'$ to the start state of $M'$.

Show that this construction does not prove the above theorem. That is, give a language $L$ with an NFA $M$ accepting it such that the NFA $M'$ obtained by the above construction does not accept $L^*$.

**Solution.** The following DFA accepts the language $\{x \in \{0, 1\}^* \mid x \text{ contains } 001\}$.

Applying the construction yields an NFA which accepts $01$, which is clearly not in $L^*$.
2. Give regular expressions for the following languages over the alphabet \( \Sigma = \{0, 1\} \).

(a) \( \{ x \mid \text{every odd character of } x \text{ is a 1}\} \).

**Solution.** \( 1((0 \cup 1)1)^* (\varepsilon \cup 0 \cup 1) \).

(b) \( \{ x \mid x \text{ does not contain the substring 10}\} \).

**Solution.** \( 0^*1^* \).

(c) (Extra Practice) \( \{ x \mid x \text{ contains at least three 1s}\} \).

(d) (Extra Practice) \( \{0, 1\}^* \setminus \{\varepsilon\} \).

3. For any string \( w = w_1w_2 \cdots w_n \) the **reverse** of \( w \), denoted \( w^R \), is (unsurprisingly) the string \( w_nw_{n-1} \cdots w_2w_1 \). Given a language \( A \), let \( A^R = \{ w^R : w \in A \} \). Show that if \( A \) is regular then so is \( A^R \).

**Solution.** Let \( M \) be a DFA computing \( A \), and let \( M' \) be the NFA obtained from \( M \) by creating a new unique accept state \( q_{acc} \), and adding \( \varepsilon \)-transitions from all old accept states of \( M \) to \( q_{acc} \). Clearly this new NFA accepts the same language as \( M \). Then, define \( M^R \) to be the NFA obtained from \( M \) by swapping the start and accept state of \( M' \) and then reversing all transitions in \( M' \) — so, for each transition from state \( q \) to state \( r \) labelled with a symbol \( a \), remove it and add the transition from \( r \) to \( q \) labelled with \( a \). We claim that \( M^R \) accepts \( A^R \).

First, suppose that \( x \) is accepted by \( M' \) and we show \( x^R \) is accepted by \( M^R \). This is easy to see: writing \( x = x_1x_2 \cdots x_n \) where each \( x_i \) is from the underlying alphabet, since \( x \) is accepted by \( M \) there is an accepting computation \( q_0, q_1, \ldots, q_n, q_{acc} \) where \( q_0 \) is the start state, \( q_i \in \delta(q_{i-1}, x_i) \) for each \( i = 1, 2, \ldots, n \), and \( q_{acc} \). We claim that \( q_{acc}, q_n, q_{n-1}, \ldots, q_0 \) is an accepting computation of \( M^R \) on \( x^R = x_nx_{n-1} \cdots x_1 \). This is clear: \( q_{acc} \) is the start state of \( M^R \), the first transition is an \( \varepsilon \)-transition, \( q_0 \) is an accept state, and each transition is available since the reverse transition was available in the original DFA.

Now, suppose \( x \) is accepted by \( M^R \), and we show \( x^R \) is accepted by \( M \). The only \( \varepsilon \)-transitions of \( M^R \) are from the start state \( q_{acc} \) to the old final states, so any accepting computation will be of the form \( q_{acc}, q_1, q_2, \ldots, q_{n-1}, q_0 \), since \( q_0 \) is the unique accept state of
Let $x_n$ be the last character read by $M^R$. Since each transition of $M^R$ was obtained by reversing a transition of $M$, which was a DFA, it follows that $\delta(q_0, x_n) = q_{n-1}$ was the unique transition from $q_0$ to $q_{n-1}$ on input $x_R$. By induction, the same argument holds for each state, and it follows that on input $x_R$ the DFA $M$ will proceed through the states $q_0, q_{n-1}, q_{n-2}, \ldots, q_1$. But $q_1$ is an accept state of $M$ by the construction of $M^R$.

4. In this question we will continue the development of the “distinguishability” relation, from Assignment 2. If $L$ is a language and $x, y$ are strings recall that $x$ and $y$ are distinguishable by $L$ if there is a string $z$ such that $xz \in L, yz \not\in L$.

![Figure 1: The DFA $M$](image)

(a) Consider the DFA $M$ drawn above, which accepts the language

$$L = \{x \mid x \text{ contains the substring } aaab\}.$$ 

For any state $q$ in the automata, let $S(q)$ be the set of strings defined by

$$S(q) = \{x \mid \text{the computation of } M \text{ on } x \text{ ends at } q\}.$$ 

For example, $S(q_{aa}) = L$, and a regular expression for $S(q_{a})$ is $(b^*a)[(b \cup ab)b^*a]^*$. Give regular expressions for $S(q_{e})$ and for $S(q_{aaa})$.

**Solution.** A general trick for these solutions is as follows, using $q_{e}$ as an example: change the DFA so that the only final state is $q_{e}$, and then run the procedure for converting a DFA into a regular expression.

A regular expression for $S(q_{e})$ is $(b \cup (a(b \cup ab)))^*$. A regular expression for $S(q_{aaa})$ is $(b \cup (a(b \cup ab)))^*aaaa^*$

(b) Let $x, y \in S(q_{a})$ be any two strings that end up at the state $q_{a}$ when run on $M$. Show that $x \equiv_L y$.

**Solution.** We actually prove the “Extra Practice” question below, as it is easier. We need to show that $x \equiv_L y$ for all $x, y \in S(q_{a})$. So, consider any string $z$, and we show that either $xz, yz \in L$ or $xz, yz \not\in L$. This is straightforward: the machine $M$ ends up at the same state $q_{a}$ on either $x$ or $y$. Since $M$ is a DFA, there is a unique computation path on $z$ from the state $q_{a}$ to some state $q_{b}$. If $q_{b}$ is an accept state, then both $xz, yz \in L$; otherwise, neither $xz$ nor $yz$ are in $L$. 

3
(c) (Extra Practice). Let $L$ be a regular language and let $M$ be a DFA computing $L$. Let $q$ be any state in $M$. Show that, for any $x, y \in S(q)$, $x \equiv_L y$. (Observe that this implies the number of equivalence classes of $L$ is at most the number of states in the smallest DFA computing $L$!).