Exercise 2.1.

Consider the regular expression \( r = (a + ab)^* \).

(a) Convert \( r \) into an equivalent NFA-\( \epsilon \) \( A \).

(b) Convert \( A \) into an equivalent NFA \( B \). (It is not necessary to use algorithm \( NFA_{\epsilon} toNFA \))

(c) Convert \( B \) into an equivalent DFA \( C \).

(d) By inspecting \( B \), give an equivalent minimal DFA \( D \). (No algorithm needed).

(e) Convert \( D \) into an equivalent regular expression \( r' \).

(f) Prove formally that \( L(r) = L(r') \).

Solution.

(a)

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Automaton obtained</th>
<th>Rule applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p \xrightarrow{(a + ab)^*} q )</td>
<td>Initial automaton from reg. expr.</td>
</tr>
<tr>
<td>2</td>
<td>( p \xrightarrow{\varepsilon} q \xrightarrow{\varepsilon} r )</td>
<td>( p \xrightarrow{\varepsilon} q )</td>
</tr>
<tr>
<td>3</td>
<td>( p \xrightarrow{\varepsilon} q \xrightarrow{\varepsilon} r )</td>
<td>( p \xrightarrow{r_1 + r_2} q )</td>
</tr>
</tbody>
</table>
(b)

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Automaton obtained</th>
<th>Rule applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Automaton" /></td>
<td><img src="image2" alt="Rule" /></td>
</tr>
<tr>
<td></td>
<td>where $\sigma \in \Sigma \cup {\varepsilon}$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3" alt="Automaton" /></td>
<td><img src="image4" alt="Rule" /></td>
</tr>
</tbody>
</table>

Initial states that can reach a final state through $\varepsilon$-transitions are made final.
Remove ε-transitions.
Remove states non reachable from initial state.

(c) States \( \{p\} \) and \( \{q, r\} \) have the exact same behaviours, so we can merge them. Indeed, both states are final and \( \delta(\{p\}, \sigma) = \delta(\{q, r\}, \sigma) \) for every \( \sigma \in \{a, b\} \). We obtain:

(e) Iter. Automaton obtained Rule applied

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Automaton obtained</th>
<th>Rule applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Automaton" /></td>
<td>Add single initial and final states.</td>
</tr>
<tr>
<td>2</td>
<td>$i \xrightarrow{a} q \xrightarrow{b} ba \xrightarrow{\varepsilon} f$</td>
<td>$i \xrightarrow{\varepsilon} p \xrightarrow{a} q \xrightarrow{b} \varepsilon \xrightarrow{f}$</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>$i \xrightarrow{a} q \xrightarrow{\varepsilon + b} f$</td>
<td>$i \xrightarrow{a + ba} p \xrightarrow{r_1 + r_2} q$</td>
</tr>
<tr>
<td>4</td>
<td>$i \xrightarrow{\varepsilon} f$</td>
<td>$i \xrightarrow{a} q \xrightarrow{\varepsilon + b} f$</td>
</tr>
<tr>
<td>5</td>
<td>$i \xrightarrow{\varepsilon + a(a + ba)^*(\varepsilon + b)} f$</td>
<td>$i \xrightarrow{a + ba} p \xrightarrow{r_1 + r_2} q$</td>
</tr>
<tr>
<td>6</td>
<td>$\varepsilon + a(a + ba)^*(\varepsilon + b)$</td>
<td>Extract regular expression from the unique transition.</td>
</tr>
</tbody>
</table>
Let us first show that \( a(a + ba)^i = (a + ab)^i a \) for every \( i \in \mathbb{N} \). We proceed by induction on \( i \). If \( i = 0 \), then the claim trivially holds. Let \( i > 0 \). Assume the claims holds at \( i - 1 \). We have
\[
\begin{align*}
a(a + ba)^i &= a(a + ba)^{i-1}(a + ba) \\
&= (a + ab)^{i-1}a(a + ba) \quad \text{(by induction hypothesis)} \\
&= (a + ab)^{i-1}(aa + aba) \quad \text{(by distributivity)} \\
&= (a + ab)^{i-1}(a + ab)a \quad \text{(by distributivity)} \\
&= (a + ab)^i a.
\end{align*}
\]
This implies that
\[
a(a + ba)^* = (a + ab)^* a. \tag{1}
\]
We may now prove the equivalence of the two regular expressions:
\[
\begin{align*}
\varepsilon + a(a + ba)^*(\varepsilon + b) &= \varepsilon + (a + ab)^*a(\varepsilon + b) \quad \text{(by (1))} \\
&= \varepsilon + (a + ab)^*(a + ab) \quad \text{(by distributivity)} \\
&= \varepsilon + (a + ab)^+ \\
&= (a + ab)^*.
\end{align*}
\]

**Exercise 2.2.**
Prove or disprove the following.

(a) If \( L_1 \) and \( L_1 \cup L_2 \) are regular, then \( L_2 \) is regular.
(b) If \( L_1 \) and \( L_1 \cap L_2 \) are regular, then \( L_2 \) is regular.
(c) If \( L_1 \) and \( L_1 L_2 \) are regular, then \( L_2 \) is regular.
(d) If \( L^* \) is regular, then \( L \) is regular.

**Solution.** All of these claims are false. Let \( \Sigma = \{a\} \). Note that since there are an uncountable number of languages over \( \Sigma \) which contain the words \( \varepsilon \) and \( a \), but only a countable number of DFAs, it follows that there must be a non-regular language \( L' \) such that \( \varepsilon, a \in L' \).

(a) Let \( L_1 = \Sigma^* \) and \( L_2 = L' \). Since \( L_1 \cup L_2 = \Sigma^* \), the claim is false.
(b) Let \( L_1 = \emptyset \) and \( L_2 = L' \). Since \( L_1 \cap L_2 = \emptyset \), the claim is false.
(c) Let \( L_1 = \Sigma^* \) and \( L_2 = L' \). Since \( \varepsilon \in L' \), it follows that \( L_1 L_2 = \Sigma^* \) and so the claim is false.
(d) Let \( L = L' \). Since \( a \in \Sigma \), it follows that \( L^* = \Sigma^* \) and so the claim is false.
Exercise 2.3.

Recall that a nondeterministic automaton $A$ accepts a word $w$ if at least one of the runs of $A$ on $w$ is accepting. This is sometimes called the existential accepting condition. Consider the variant in which $A$ accepts $w$ if all runs of $A$ on $w$ are accepting (in particular, if $A$ has no run on $w$ then it accepts $w$). This is called the universal accepting condition and such automata will be referred to as a co-NFA.

Intuitively, we can visualize a co-NFA as executing all runs in parallel. After reading a word $w$, the automaton is simultaneously in all states reached by all runs labelled by $w$, and accepts if all those states are accepting.

(a) Suppose $A_1$ and $A_2$ are two co-NFA which accept languages $L_1$ and $L_2$ respectively. Let $n_1$ and $n_2$ be the number of states of $A_1$ and $A_2$ respectively. Show that there is a co-NFA $B$ over $n_1 + n_2$ states which accepts $L_1 \cap L_2$.

(b) Give an algorithm that transforms a co-NFA into a DFA recognizing the same language. This shows that automata with universal accepting condition recognize the regular languages.

Let $\Sigma = \{a, b\}$. Given a word $w = a_1 a_2 ... a_n$ where each $a_i \in \Sigma$, let $w^R = a_n a_{n-1} ... a_1$ denote the reverse of $w$. For any $n \in \mathbb{N}$, consider the language $L_n := \{ww^R \in \Sigma^{2n} \mid w \in \Sigma^n\}$.

(c) Give a co-NFA with $O(n^2)$ states that recognizes $L_n$.

(d) Prove that every NFA (and hence also every DFA) recognizing $L_n$ has at least $2^n$ states.

Solution.

(a) Let $A_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$ be the given two co-NFAs. Let $B$ be the co-NFA given by $B = (Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2, I_1 \cup I_2, F_1 \cup F_2)$. Notice that if $|Q_1| = n_1$ and $|Q_2| = n_2$, then the number of states of $B$ is $n_1 + n_2$. Further, note that if $\rho$ is a run of $B$ on a word $w$ if and only if $\rho$ is either a run of $A_1$ on $w$ or $\rho$ is a run of $A_2$ on $w$. It follows that all runs of $B$ on a word $w$ are accepting if and only if all runs of $A_1$ and $A_2$ on $w$ are accepting. Hence, $B$ accepts $L_1 \cap L_2$.

(b) Let $A = (Q, \Sigma, \delta, Q_0, F)$ be a co-NFA. We do the same powerset construction that we do for NFAs to get a DFA $B = (Q, \Sigma, \Delta, q_0, F)$ except we now set $F = \{Q' \in Q : Q' \subseteq F\}$. All the other elements are defined in exactly the same way as is done for the powerset construction.

(c) For any $n \in \mathbb{N}$ and any $1 \leq i \leq n$, let

$$L_n^i := \{w : w \in \Sigma^{2n}, \text{ the } i^{th} \text{ letter of } w \text{ and the } (2n-i+1)^{th} \text{ letter of } w \text{ are the same} \}$$

Notice that $L_n = \bigcap_{1 \leq i \leq n} L_n^i$. By a), it follows that if we give a co-NFA of size $O(n)$ for each $L_n^i$, then we have a co-NFA of size $O(n^2)$ for $L_n$.

We now construct a co-NFA of size $O(n)$ for each $L_n^i$, as given by the following illustration.
First, the automaton has a sequence of states \( q_0, q_1, \ldots, q_{i-1} \) with transitions \( q_j \xrightarrow{a,b} q_{j+1} \) for every \( 0 \leq j \leq i - 2 \). Intuitively, these states are simply used to count the number of letters read so far. Hence, upon reaching \( q_j \) for any \( j \leq i - 1 \), we know that we have read \( j \) letters. From here, the automaton has two transitions \( q_{i-1} \xrightarrow{a} q_i \) and \( q_{i-1} \xrightarrow{b} q_i \). Intuitively, these two transitions help us remember the \( i^{th} \) letter of the word.

Then, we have a collection of states \( q^a_{i+1}, q^b_{i+1}, \ldots, q^a_{2n-i}, q^b_{2n-i} \) along with the transitions, \( q^a_j \xrightarrow{a,b} q^{a}_{j+1} \) and \( q^b_j \xrightarrow{a,b} q^{b}_{j+1} \) for every \( i \leq j \leq 2n - i - 1 \). Intuitively, these states are simply used to count the number of letters starting from the \( i^{th} \) letter, while simultaneously remembering the \( i^{th} \) letter. Hence, upon reaching \( q^a_j \) (resp. \( q^b_j \)) for any \( j \leq 2n - i \), we know that we have read \( j \) letters and that the \( i^{th} \) letter that we read was an a (resp. a b). From here, we have two transitions \( q^a_{2n-i} \xrightarrow{a} q_{2n-i+1} \) and \( q^b_{2n-i} \xrightarrow{b} q_{2n-i+1} \). Intuitively, these two transitions force that the \( (2n - i + 1)^{th} \) letter that we read is the same as the \( i^{th} \) letter that we read before.

Finally, we have a sequence of states \( q_{2n-i+1}, q_{2n-i+2}, \ldots, q_{2n} \) with transitions \( q_j \xrightarrow{a,b} q_{j+1} \) for every \( 2n - i + 1 \leq j \leq 2n \). Once again, these states are simply used to count the number of letters read and we can show that if we reach \( q_j \) for any \( j \leq 2n \), then we have read \( j \) letters. We then set the only final state to be \( q_{2n} \).

(d) Suppose \( A \) is some NFA which recognizes \( L_n \). For every \( ww^R \in \Sigma^{2n} \), \( A \) has at least one accepting run on \( ww^R \). Let \( q_w \) be the state reached by this run after reading the prefix \( w \) (If there are multiple such runs, pick any one of them). We claim that if \( w \neq w' \), then \( q_w \neq q_{w'} \). Notice that this claim implies that there are at least \( 2^n \) states in \( A \) and so it simply suffices to prove this claim.

Suppose \( q_w = q_{w'} \) for some pair \( w \neq w' \). Hence, after reading \( w' \) the automaton \( A \) can reach \( q_w \). By definition of \( q_w \), we know that there is a run on the word \( w^R \) starting from \( q_w \) and ending in a final state. This implies that the automaton accepts \( w'w^R \), because first the automaton can reach \( q_w \) by reading \( w' \) and then from \( q_w \) it can reach a final state by reading \( w^R \). But \( w'w^R \notin L_n \), contradicting the fact that \( A \) recognizes \( L_n \).

Puzzle exercise 2.4.

Let \( \Sigma := \{a, b\} \). We say that a regular expression \( r \) is elllone, if \( r \) is of the form \( st, s^*, x, \epsilon, \emptyset, as + bt \), or \( as + bt + \epsilon \), where \( s, t \) are elllone REs and \( x \in \Sigma \).

For example, \((a + b)^* \) and \( ab + b(ab + b \emptyset + \epsilon)^* \) are elllone, but \( (aa + ab)^* \) and \( a + b(ab + (ba)^* + \epsilon) \) are not.

Prove that there is a regular language \( L \), s.t. no elllone RE \( r \) with \( L(r) = L \) exists.
Hint: Consider the language $L$ of words that contain an even number of $a$ or an even number of $b$. 