Exercise 1:

Give algorithms to tell whether

a) a given regular language \( L \) is universal.
(i.e. \( L = \Sigma^* \))

b) two regular languages have at least one string in common.

Exercise 2:

Using the characterization of regular languages in terms of DFAs, show the following:

If \( L_1 \) and \( L_2 \) are regular, then so is \( L_1 \cap L_2 \).

Do not rely on De Morgan's law \( L_1 \cap L_2 = \overline{L_1} \cup \overline{L_2} \).

Apply the construction of a DFA for \( L_1 \cap L_2 \) to the following DFAs \( A_1 \) for \( L_1 \) and \( A_2 \) for \( L_2 \):

\[ A_1 : \]

\[ A_2 : \]

Note: we can assume that \( L_1 \) and \( L_2 \) are RLs over the same alphabet \( \Sigma \).
SOLUTIONS

1) a) \( L = \Sigma^* \), then \( \overline{L} = \Sigma^\star = \emptyset \)

Hence, we need to check whether \( \overline{L} \) is empty.

Algorithm when \( L \) is given as a DFA \( D_L \):

1) Construct a DFA \( D_{\overline{L}} \) s.t.: \( L(D_{\overline{L}}) = \overline{L} \) by swapping final and non-final states of \( D_L \).

2) Check whether \( D_{\overline{L}} \) is empty (by constructing the set of states reachable from the initial state, and checking whether it contains at least one final state).

Algorithm when \( L \) is given as an NFA \( N_L \):

1) Determine \( N_{\overline{L}} \), i.e. construct a DFA \( D_{\overline{L}} \) s.t.: \( L(D_{\overline{L}}) = L(N_{\overline{L}}) \) (Note: \( D_{\overline{L}} \) might have a number of states that is exponential in the number of states of \( N_L \)).

2) Proceed as in the case of a DFA

Algorithm when \( L \) is given as a RE \( E_L \):

1) Construct an \( \epsilon \)-NFA \( N_{\epsilon_L} \) s.t.: \( L(N_{\epsilon_L}) = L(E_L) \)

2) Eliminate \( \epsilon \)-transitions from \( N_{\epsilon_L} \) obtaining an NFA \( N_L \) s.t.: \( L(N_L) = L(N_{\epsilon_L}) \)

3) Proceed as in the case of an NFA
1) b) To check whether two RLs \( L_1 \) and \( L_2 \) have at least one string in common, we can check whether \( L_1 \cap L_2 \) is non-empty.

Algorithm:

1) Construct a DFA/NFA/\( \varepsilon \)-NFA/RE for \( L_1 \cap L_2 \), starting from DFA/NFA/\( \varepsilon \)-NFA/REs for \( L_1 \) and for \( L_2 \).

2) Check whether \( L_1 \cap L_2 \) is non-empty.

Note: to construct a DFA/NFA/\( \varepsilon \)-NFA/RE for \( L_1 \cap L_2 \), we can use De Morgan's law.
- \( L_1 \cap L_2 \) is still a RL, since RLs are closed under intersection.
2) Let $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ be a DFA s.t. $L(A_1) = L_1$. Set $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ s.t. $L(A_2) = L_2$.

Consider a string $w$ accepted by both $A_1$ and $A_2$. Let $w = e_1 e_2 \ldots e_n$. Then we have

\[
\begin{align*}
q_{01} &\xrightarrow{a_1} p_1 & q_{02} &\xrightarrow{a_1} p'_1 \\
q_{01} &\xrightarrow{a_2} p_2 & q_{02} &\xrightarrow{a_2} p'_2 \\
&\quad \vdots & &\quad \vdots \\
q_{01} &\xrightarrow{a_n} p_n & q_{02} &\xrightarrow{a_n} p'_n \in F_1 \\
q_{01} &\xrightarrow{a_1} p_n & q_{02} &\xrightarrow{a_2} p'_n \in F_2
\end{align*}
\]

Hence we can construct a DFA $A = (Q, \Sigma, \delta, q_0, F)$ that simulates the transitions of both $A_1$ and $A_2$:

- Each state of $A$ is a pair of states $(q_1, q_2)$, where $q_1 \in Q_1$ and $q_2 \in Q_2$. Hence $Q = Q_1 \times Q_2$.
- The initial state $q_0$ is the pair of initial states of $Q_1$ and $Q_2$. Hence $q_0 = (q_{01}, q_{02})$.
- The set of final states is such that both $A_1$ and $A_2$ accept if $A$ accepts, hence $F = F_1 \times F_2$.
- The transition function $\delta$ simulates the transitions of both $A_1$ and $A_2$: If $A$ is in state $(q_1, q_2)$, then on input $a$ it goes to a state $(q'_1, q'_2)$, where $q'_1 = \delta_1(q_1, a)$ and $q'_2 = \delta_2(q_2, a)$.

Hence, for all $a \in \Sigma$, $q_1 \in Q_1$, $q_2 \in Q_2$:

$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. 
One can show that $A_\cap$ constructed in this way accepts $L(A_1) \cap L(A_2)$.

$A_\cap$ is called the **product automaton**.

By applying this construction to the automata $A_1$ and $A_2$, we obtain

We have used $q_{ij}$ to denote $(q_i, q_j)$.