Exercise 1: Drawing DFAs  

(7 Points)

Construct DFAs that recognizes the following languages. The alphabet set is $\Sigma = \{0, 1\}$.

1. $L_1 = \{w \mid w$ is any string except 11 and 111$\}$.
2. $L_2 = \{w \mid w$ contains at least two 0s and at most one 1$\}$.
3. Construct a DFA which accepts the language $L_2 \setminus L_1 = \{w \mid w \in L_2$ and $w \notin L_1\}$.

Sample Solution

(a) The following DFA accepts $L_1$:

(b) The following DFA accepts $L_2$:
(c) Notice that \( L_2 \subseteq L_1 \). Therefore, \( L_2 \setminus L_1 = \emptyset \) i.e. the empty language (you can also write \( \emptyset := \{ \} \)). The following DFA accepts the empty language:

\[
\Sigma = \{0, 1\}
\]

![DFA Diagram]

**Remark:** There’s a difference between the following two languages \( L_1 := \emptyset \) and \( L_2 := \{ \varepsilon \} \), where the empty string \( \varepsilon \) is defined as a string of length \( |\varepsilon| = 0 \).

The empty language \( \emptyset \) is a set containing no strings, while \( L_2 = \{ \varepsilon \} \) is a set containing \( \varepsilon \), while \( \varepsilon \) is just a string but a string containing no symbols. So, \( L_1, L_2 \) are different languages since \( L_2 \) contains a string while \( L_1 \) is empty (\( 0 = |L_1| \neq |L_2| = 1 \)).

**Exercise 2: Closure under Set Difference**

Let \( L, L_1, L_2 \) be regular languages. Show that both \( \overline{L} := \Sigma^* \setminus L \) and \( L_1 \cap L_2 \) are regular as well by constructing the corresponding DFAs. Deduce that \( L_1 \setminus L_2 \) is also regular.

**Remark:** No need for drawing state diagrams. Show how a DFA for the language in question can be constructed presuming the existence of DFAs for \( L, L_1, L_2 \).

**Sample Solution**

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be the DFA recognizing \( L \). We define the DFA \( \overline{M} := (Q, \Sigma, \delta, q_0, \overline{F}) \) by inverting the set of accepting states of \( M \), i.e. \( \overline{F} := Q \setminus F \). We show that \( \overline{M} \) recognizes \( \overline{L} \).

If \( w \in \overline{L} \), then \( w \notin L \) and so \( M \) halts in an non accepting state \( q \) when processing \( w \). \( \overline{M} \) will halt in the same state (because we only changed the set of accepting states), but here \( q \) is an accepting state. Analogously, if \( w \notin \overline{L} \), then \( w \in L \) and so \( M \) halts in an accepting state when processing \( w \). \( \overline{M} \) will again halt in the same state, but here \( q \) is a non accepting state. So we have that \( \overline{M} \) halts in an accepting state when processing \( w \) if and only if \( w \in \overline{L} \). Thus \( \overline{M} \) recognizes the language \( \overline{L} \) which is therefore regular.

For proving the regularity of \( L_1 \cap L_2 \), we construct the product automaton like done in the lecture (Theorem 1.25. p. 30) for \( L_1 \cup L_2 \), with the difference that we set \( F := F_1 \times F_2 \) as the set of accepting states, where \( F_1 \) and \( F_2 \) are the sets of accepting states of the DFAs for \( L_1 \) and \( L_2 \).

**Alternative approach:** using De Morgan’s law we obtain: \( L_1 \cap L_2 = (L_1^c \cup L_2^c) \). Thus \( L_1 \cap L_2 \) is regular, since we already know that regularity is conserved by complementation and a finite number of unions of regular languages (cf. lecture).

Finally, we can easily see that since \( L_1 \setminus L_2 = L_1 \cap \overline{L_2} \) and after showing that regular languages are closed under intersection and complement, it follows that regular languages are also closed under set difference.
Exercise 3: From NFA to DFA  

Consider the following NFA.

(a) Give a formal description of the NFA by giving the alphabet, state set, transition function, start state and the set of accept states.

(b) Construct a DFA which is equivalent to the above NFA by drawing the corresponding state diagram.

(c) Explain which language the automaton accepts.

Sample Solution

(a) The set of states is \( Q = \{ q_0, q_1, q_2 \} \); the alphabet \( \Sigma = \{ a, b \} \); the starting state is \( q_0 \); the set of accept states is \( F = \{ q_2 \} \); the transition function is shown in the following table.

<table>
<thead>
<tr>
<th></th>
<th>( q_0 )</th>
<th>( q_1 )</th>
<th>( q_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( { q_1 } )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( b )</td>
<td>( { q_0, q_1 } )</td>
<td>( { q_0, q_2 } )</td>
<td>( { q_0 } )</td>
</tr>
</tbody>
</table>

(b) After performing the algorithm from the lecture we obtain the following DFA. All transitions which are not in the picture go to the garbage state \( \emptyset \).

(c) The recognized language contains words of length at least two. Furthermore any \( a \) is immediately followed by a \( b \). The number of \( b \)'s after the last \( a \) must not be two.