Exercise 4.1
Let $A$ and $B$ be respectively the following NFAs:

(a) Compute the coarsest stable refinements (CSR) of $A$ and $B$.
(b) Construct the quotients of $A$ and $B$ with respect to their CSRs.
(c) Show that
   \[ L(A) = \{ w \in \{a, b\}^* : w \text{ contains an occurrence of the subword } ab \} \]
   \[ L(B) = \{ w \in \{a, b, c\}^* : w \text{ starts with } bc \text{ and ends with } a \} \]
(d) Are the automata obtained in (b) minimal?

Exercise 4.2
Let $\Sigma = \{a, b\}$. For any $n \in \mathbb{N}$, let $L_n := \{ww^R : w \in \Sigma^n\}$, where $w^R$ is the reverse of $w$, e.g. $(abc)^R = cba$. In Exercise 2.3, we have shown that every NFA (and hence also every DFA) recognizing $L_n$ must have at least $2^n$ states. We refine this bound here for DFAs.

(a) Construct $A_2$, the minimal DFA for $L_2$.
(b) What are the residuals of $L_2$? Assign them to the states of the DFA you gave for (a).
(c) Give a construction for a DFA that accepts $L_n$. 
(d) How many states does the minimal DFA for $L_n$ contain, for $n \geq 2$?

**Exercise 4.3**
Consider the following DFAs $A$, $B$, $C$ and $D$:

(a) Use pairings to decide *algorithmically* whether $L(A) \cap L(B) \subseteq L(C)$.
(b) Use pairings to decide *algorithmically* whether $L(D) \subseteq L(A) \cap L(B)$.
Solution 4.1
A) (a)

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Block to split</th>
<th>Splitter</th>
<th>New partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>—</td>
<td>—</td>
<td>{q_0, q_1, q_2, q_3, q_4, q_5}</td>
</tr>
<tr>
<td>1</td>
<td>{q_0, q_1, q_2, q_3, q_4}</td>
<td>(b, {q_5})</td>
<td>{q_0}, {q_1, q_2, q_3, q_4}, {q_5}</td>
</tr>
<tr>
<td>2</td>
<td>none, partition is stable</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

The CSR is \( P = \{\{q_0\}, \{q_1, q_2, q_3, q_4\}, \{q_5\}\} \).

(b)

(c) The automaton \( A \) and the automaton obtained from (b) accept the same language. Notice that in the automaton from (b), there is an accepting run for a word \( w \) which visits the final state exactly once if and only if \( w \in \Sigma^*ab \). Since there are self-loops at the final state for both \( a \) and \( b \), it follows that the language of this automaton is \( \Sigma^*ab\Sigma^* \).

(d) Yes. By (c), the language accepted by \( A \) is \( \Sigma^*ab\Sigma^* \). An NFA with one state can only accept \( \emptyset, \{\varepsilon\}, a^*b^* \) and \( \{a,b\}^* \). Suppose there exists an NFA \( A' = (\{q_0, q_1\}, \{a, b\}, \delta, Q_0, F) \) accepting \( L(A) \). Without loss of generality, we may assume that \( q_0 \) is initial. \( A' \) must respect the following properties:

- \( q_0 \notin F \), since \( \varepsilon \notin L(A) \),
- \( q_1 \in F \), since \( L(A) \neq \emptyset \),
- \( q_1 \notin Q_0 \), since \( \varepsilon \notin L(A) \),
- \( \delta(q_0, a) \) is non-empty, otherwise it is impossible to accept \( ab \). Further, \( q_1 \notin \delta(q_0, a) \), otherwise it is possible to accept \( a \). Hence, \( \delta(q_0, a) = \{q_0\} \).
- \( q_1 \in \delta(q_0, b) \), otherwise it is impossible to accept \( ab \).

This implies that \( A' \) accepts \( b \), yet \( b \notin L(A) \). Therefore, no NFA with two states can accept \( L(A) \). □

B) (a)

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Block to split</th>
<th>Splitter</th>
<th>New partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>—</td>
<td>—</td>
<td>{q_0, q_1, q_2, q_3, q_4}, {q_5}</td>
</tr>
<tr>
<td>1</td>
<td>{q_0, q_1, q_2, q_3, q_4}</td>
<td>(a, {q_5})</td>
<td>{q_0, q_1, q_2, q_4}, {q_5}</td>
</tr>
<tr>
<td>2</td>
<td>{q_2, q_4}</td>
<td>(b, {q_0, q_1, q_3})</td>
<td>{q_0, q_1, q_3}, {q_2, q_4}, {q_5}</td>
</tr>
<tr>
<td>3</td>
<td>{q_0, q_1, q_3}</td>
<td>(c, {q_4})</td>
<td>{q_0, q_1}, {q_2}, {q_4}, {q_5}</td>
</tr>
<tr>
<td>4</td>
<td>{q_0, q_1}</td>
<td>(c, {q_2})</td>
<td>{q_0}, {q_1}, {q_2}, {q_4}, {q_5}</td>
</tr>
</tbody>
</table>

The CSR is \( P = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_4\}, \{q_5\}\} \).

(b) The automaton remains unchanged.

(c) \( \supseteq \) Suppose \( w \) starts with \( bc \) and ends with \( a \). If \( w = w_1w_2 \ldots w_n \), then \( q_0, q_1, \ldots, q_2, \ldots, q_5 \) is a valid accepting run for \( w \).

\( \subseteq \) Let \( w \in L(B) \). Note that every outgoing edge from \( q_0 \) is labelled by a \( b \) and goes to either \( q_1 \) or \( q_3 \) and every outgoing edge from both \( q_1 \) and \( q_3 \) is labelled by a \( c \). It follows that any path from \( q_0 \) to \( q_5 \) must involve reading a \( bc \) at the beginning. Further, all the incoming edges to \( q_5 \) are labelled by an \( a \). It follows that any path from \( q_0 \) to \( q_5 \) must involve reading an \( a \) at the end. Since \( w \in L(B) \), it then follows that \( w \) must begin with \( bc \) and end with \( a \). □

(d) No. The following NFA with four states accepts the same language.
Solution 4.2

(a) The trap state is omitted for the sake of readability:

(b) We have \( L_2 = \{aaaa, abba, baab, bbbb\} \). We compute the residuals \( L^w \) for all words \( w \) by increasing length of \( w \).

- \(|w| = 0\): \( L^\varepsilon = \{aaaa, abba, baab, bbbb\} \).
- \(|w| = 1\): \( L^a = \{aaa, abba, baab, bbbb\} \) and \( L^b = \{aab, bbb\} \).
- \(|w| = 2\): \( L^{aa} = \{aa\}, L^{ab} = \{ba\}, L^{ba} = \{ab\} \) and \( L^{bb} = \{bb\} \).
- \(|w| = 3\): \( L^{aaa} = \{a\} = L^{abb} \) and \( L^{baa} = \{b\} = L^{bbb} \).
- \(|w| \geq 4\): \( L^w = \{\varepsilon\} \) if \( w \in L_k \), \( \emptyset \) otherwise.

(c) Notice that \( L_{k+1} \) is simply \( aL_k a + bL_k b \) for any \( k \geq 2 \). Using this observation, we generalize the construction given in (a) for \( k = 2 \), by induction on \( k \). The base case of \( k = 2 \) has been done already. Suppose we have already constructed \( A_k = (Q_k, \{a,b,\}, \delta_k, q_0^k, \{f, trap_k\}) \) with the property that it has exactly one initial state, one final state and one trap state \( trap_k \) (Note that \( A_2 \) satisfies this property). We now construct \( A_{k+1} = (Q_{k+1}, \{a,b,\}, \delta_{k+1}, q_0^{k+1}, \{f, trap_{k+1}\}) \) as follows:

The set of states \( Q_{k+1} \) is taken to be \( \{q_0^{k+1}, q_f^{k+1}, trap_{k+1}\} \cup (Q_k \setminus \{trap_k\}) \times \{1,2\} \), where \( q_0^{k+1}, q_f^{k+1}, trap_{k+1} \) are three fresh states. Intuitively we add a fresh initial state, a fresh final state, a fresh trap state and take two copies of the states of \( A_k \) while removing \( trap_k \).

The transition function \( \delta_{k+1} \) is defined as follows:

- \( \delta_{k+1}(q_0^{k+1}, a) = (q_f^k, 1) \) and \( \delta_{k+1}(q_0^{k+1}, b) = (q_f^k, 2) \). Intuitively, upon reading an \( a \) (resp. \( b \)) from the initial state of \( A_{k+1} \), we move to the initial state of the first (resp. second) copy of \( A_k \).
- \( \delta_{k+1}(q_f^k, a) = q_f^{k+1} \) and \( \delta_{k+1}(q_f^k, b) = q_f^{k+1} \). Intuitively, upon reading an \( a \) (resp. \( b \)) from the final state of the first (resp. second) copy of \( A_{k+1} \), we move to the final state of \( A_{k+1} \).
- \( \delta_{k+1}((q, i), a) = p \) where \( p = (\delta_k(q, a), i) \) if \( \delta_k(q, a) \neq trap_k \) and otherwise \( p = trap_{k+1} \). Intuitively, within a copy of \( A_k \), we follow the transitions of \( A_k \) and stay within that copy itself if the state that we are supposed to go to is not the trap state of \( A_k \). Otherwise, instead of going to the trap state of \( A_k \), we go to the trap state of \( A_{k+1} \).
Assuming that $A_k$ recognizes $L_k$, we can then show that $A_{k+1}$ recognizes $L_{k+1}$. By induction, this will show that our construction is correct.

(d) Note that if $f(k)$ is the number of states that $A_k$ has, (where $A_k$ is the DFA defined in the previous subproblem), then $f(2) = 11$ and $f(k+1) = 2(f(k)−1)+3 = 2f(k)+1$. Solving this, we get $f(k) = 3·2^k−1$. We claim that $A_k$ is a minimal DFA, by induction on $k$. The base case of $k = 2$ is already done. For the induction step, suppose $p, q$ are two distinct states of $A_{k+1}$. We will show that $L_{A_{k+1}}(p) \neq L_{A_{k+1}}(q)$.

Notice that the initial state $q_{0}^{k+1}$ recognizes only strings of length $2k + 2$ and the final state $q_{f}^{k+1}$ recognizes only $\epsilon$, whereas the other states of $A_{k+1}$ do not recognize any of these strings. This implies that the languages of the initial and the final states are different from the rest. Similarly, the language of the trap state is also different from the rest.

Hence, we can assume that $p = (p', i)$ and $q = (q', j)$ for some $p', q' \in Q_k$ and some $i, j \in \{1, 2\}$. If $i \neq j$, then $p$ and $q$ belong to different copies of $A_k$. Let $i = 1$ and $j = 2$. Notice that $L_{A_{k+1}}(p) = L_{A_k}(p')a$ and $L_{A_{k+1}}(q) = L_{A_k}(q')b$. Hence $L_{A_{k+1}}(p) \neq L_{A_{k+1}}(q)$.

The only case left is when $i = j$. In this case notice that $L_{A_{k+1}}(p) = L_{A_k}(p')c$ and $L_{A_{k+1}}(q) = L_{A_k}(q')c$ where $c$ is either $a$ or $b$, depending on whether $i$ is 1 or 2. By induction hypothesis, $A_k$ is the minimal DFA for $L_k$ and so $L_{A_k}(p')$ and $L_{A_k}(q')$ are different. Hence $L_{A_{k+1}}(p) \neq L_{A_{k+1}}(q)$, thereby concluding the proof.

Solution 4.3

(a) We first build the pairing accepting $L(A) \cap L(B)$. Note that it is not necessary to explore the implicit trap states of $A$ and $B$ as they cannot lead to final states in the pairing. We obtain:

Now, we build the pairing accepting $(L(A) \cap L(B)) \setminus L(C)$, or equivalently $(L(A) \cap L(B)) \cap \overline{L(C)}$, from the above automaton and $C$. Recall that the complement of $C$ is the following automaton:

Once again, it is not necessary to explore the implicit trap states of the automaton for $L(A) \cap L(B)$. The following automaton is the pairing accepting $(L(A) \cap L(B)) \cap L(C)$:

Since the above automaton does not contain final states, its language is empty and hence $L(A) \cap L(B) \subseteq L(C)$.

(b) This time we want to check whether $L(D) \setminus (L(A) \cap L(B))$ is empty. That is, we need to construct the pairing $L(D) \cap (L(A) \cap L(B))$. Thus, it is not necessary to explore the implicit trap states of the automaton $D$, but it is necessary for $A$ and $B$, as their trap states may be part of final states in the pairing. First we obtain the automaton accepting $(L(A) \cap L(B))$: 
Now, we build the pairing accepting $L(D) \cap (L(A) \cap L(B))$. We obtain:

Since the above automaton contains a final state, it means that there is a word in the language $L(D) \setminus (L(A) \cap L(B))$, that is, there is a word accepted by $D$, but not by $A$ and $B$. For example, any word starting with a letter $b$. Therefore, it is not true that $L(D) \subseteq L(A) \cap L(B)$. 