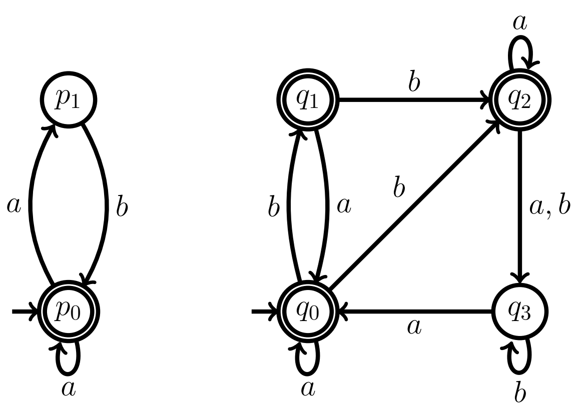
# Automata and Formal Languages — Exercise Sheet 5

**Exercise 5.1**
Consider the following NFAs $A$ and $B$:




(a) Use algorithm *UnivNFA* to determine whether $L(B) = \{a, b\}^*$.

(b) Use algorithm *InclNFA* to determine whether $L(A) \subseteq L(B)$.

**Exercise 5.2**

(a) We have seen that testing whether two NFAs accept the same language can be done by using algorithm *InclNFA* twice. Give an alternative algorithm, based on pairings, for testing equality.

(b) Give two NFAs $A$ and $B$ for which exploring only the minimal states of $[\mathit{NFAtoDFA}(A), \mathit{NFAtoDFA}(B)]$ is not sufficient to determine whether $L(A) = L(B)$.

(c) Show that the problem of determining whether an NFA and a DFA accept the same language is PSPACE-hard.

**Exercise 5.3**
For every $n \in \mathbb{N}$, let $L_n \subseteq \{a, b\}^*$ be the language described by the regular expression $(a+b)^*a(a+b)^n b(a+b)^*$.

(a) Give an NFA $A_n$ with $n+3$ states that accepts $L_n$.

(b) If you swap the final and non final states of $A_n = (Q, \Sigma, \delta, F)$, you will obtain NFA $A'_n = (Q, \Sigma, \delta, Q \setminus F)$. Does $A'_n$ accept $\overline{L_n}$? Justify your answer.

(c) Show that $A'_n$ with *universal accepting condition* does recognize the complement of $A_n$.

**Exercise 5.4**

Let $\Sigma$ be an alphabet, and define the *shuffle operator* $\| \; : \; \Sigma^* \times \Sigma^* \to 2^{\Sigma^*}$ as follows, where $a, b \in \Sigma$ and $w, v \in \Sigma^*$:

$$\begin{array}{lcl} w \parallel \varepsilon & = & \{w\} \\ \varepsilon \parallel w & = & \{w\} \\ aw \parallel bv & = & \{a\}\,(w \parallel bv) \cup \{b\}\,(aw \parallel v) \cup \{bw \mid w \in au||v\} \end{array}$$

For example we have:

$$b||d = \{bd, db\}, \quad ab||d = \{abd, adb, dab\}, \quad ab||cd = \{cabd, acbd, abcd, cadb, acdb, cdab\}.$$

Given DFAs recognizing languages $L_1, L_2 \subseteq \Sigma^*$ construct an NFA recognizing their *shuffle*

$$L_1 \parallel L_2 := \bigcup_{u \in L_1, v \in L_2} u \parallel v \; .$$

**Exercise 5.5**

★ The *perfect shuffle* of two languages $L, L' \in \Sigma^*$ is defined as:

$$L \mathbin{\widetilde{ш}} L' = \{w \in \Sigma^* : \exists a_1, \ldots, a_n, b_1, \ldots, b_n \in \Sigma \text{ s.t. } \begin{array}{l} a_1 \cdots a_n \in L \text{ and} \\ b_1 \cdots b_n \in L' \text{ and} \\ w = a_1 b_1 \cdots a_n b_n \; \} \; . \end{array}$$

Give an algorithm that takes two DFAs $A$ and $B$ in input, and that returns a DFA accepting $L(A) \mathbin{\widetilde{ш}} L(B)$.

**Solution 5.1**

(a) The trace of the execution is as follows:

| **Iter.** | $\mathcal{Q}$ | $\mathcal{W}$ |
|---|---|---|
| 0 | $\emptyset$ | $\{\{q_0\}\}$ |
| 1 | $\{\{q_0\}\}$ | $\{\{q_1, q_2\}\}$ |
| 2 | $\{\{q_0\}, \{q_1, q_2\}\}$ | $\{\{q_2, q_3\}\}$ |
| 3 | $\{\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}\}$ | $\{q_3\}$ |
| 4 | $\{\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}, \{q_3\}\}$ | $\emptyset$ |

At the fourth iteration, the algorithm tests state $\{q_3\}$ which is minimal and non final, and hence it returns *false*. Therefore, $L(B) \neq \{a, b\}^*$.

(b) The trace of the algorithm is as follows:

| **Iter.** | $\mathcal{Q}$ | $\mathcal{W}$ |
|---|---|---|
| 0 | $\emptyset$ | $\{[p_0, \{q_0\}]\}$ |
| 1 | $\{[p_0, \{q_0\}]\}$ | $\{[p_1, \{q_0\}]\}$ |
| 2 | $\{[p_0, \{q_0\}], [p_1, \{q_0\}]\}$ | $\{[p_0, \{q_1, q_2\}]\}$ |
| 3 | $\{[p_0, \{q_0\}], [p_1, \{q_0\}], [p_0, \{q_1, q_2\}]\}$ | $\emptyset$ |

At the third iteration, $\mathcal{W}$ becomes empty and hence the algorithm returns *true*. Therefore $L(A) \subseteq L(B)$.

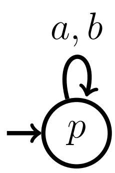
**Solution 5.2**

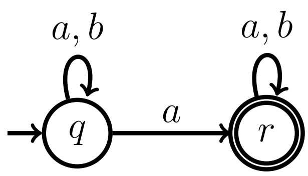
(a) We construct the pairing $[\mathit{NFAtoDFA}(A), \mathit{NFAtoDFA}(B)]$ on the fly. The algorithm returns *false* if it encounters a state $[P, P']$ such that only one of $P$ and $P'$ contains a final state. If no such state is encountered, the algorithm returns *true*.

**Input:** NFAs $A = (Q, \Sigma, \delta, Q_0, F)$ and $A' = (Q', \Sigma, \delta', Q'_0, F')$.
**Output:** $L(A) = L(A')$?

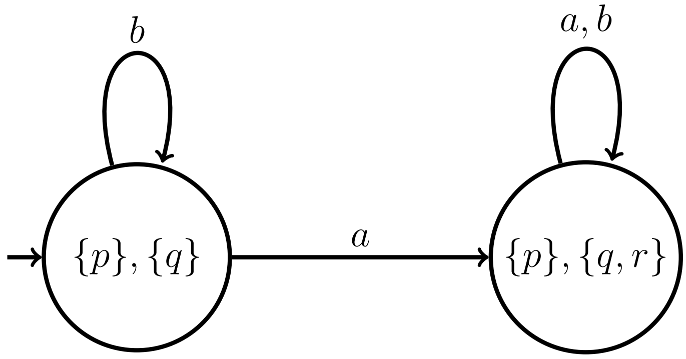
$Q \leftarrow \emptyset$
$W \leftarrow \{[Q_0, Q'_0]\}$
**while** $W \neq \emptyset$ **do**
  **pick** $[P, P']$ **from** $W$
  **if** $(P \cap F = \emptyset) \neq (P' \cap F' = \emptyset)$ **then**
    **return false**
  **for** $a \in \Sigma$ **do**
    $q \leftarrow [\delta(P, a), \delta'(P', a)]$
    **if** $q \notin Q \wedge q \notin W$ **then**
      **add** $q$ **to** $W$
**return true**

(b) Let $A$ and $B$ be the following NFAs:




The pairing of $A$ and $B$ is as follows:




State $[\{p\}, \{q\}]$ does not allow us to conclude anything since both $p$ and $q$ are non final. However, state $[\{p\}, \{q, r\}]$, which is not minimal, allows us to conclude that $L(A) \neq L(B)$ since $r$ is final.

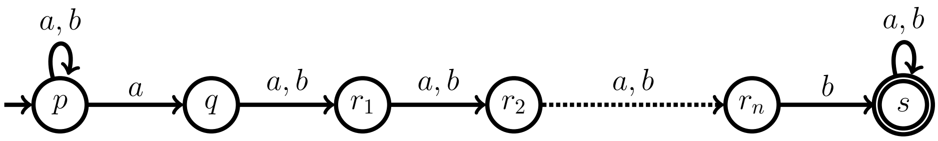
(c) To show PSPACE-hardness, it suffices to give a reduction from NFA universality. Let $A$ be an NFA. Let $B$ the one state DFA that accepts $\Sigma^*$. The following holds:

$$L(A) = \Sigma^* \iff L(A) = L(B).$$

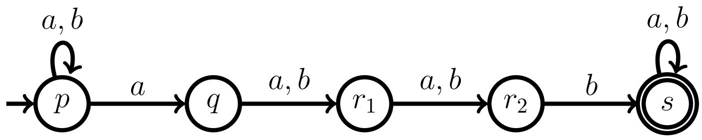
Therefore, $\langle A \rangle \mapsto \langle A, B \rangle$ is a reduction from NFA universality to NFA/DFA equality.

**Solution 5.3**

(a)




For example, the automaton $A_2$ is as follows:




(b) No, it would accept $\{a, b\}^*$ since every word could be accepted in state $p$.

(c) Observe that $A_n$ and $A'_n$ have exactly the same runs on a given word $w$. We have

$$\begin{array}{ll} & A_n \text{ accepts } w \\ \text{iff} & \text{some run of } A_n \text{ on } w \text{ leads to a state of } F \\ \text{iff} & \text{it is not the case that all runs of } A'_n \text{ lead to a state of } Q \setminus F \\ \text{iff} & A'_n \text{ does not accept } w \end{array}$$

**Solution 5.4**

Let $A_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be a DFA with $L_i = L(A_i)$ (for $i = 1, 2$). We use a variation of the pairing construction, i.e., we construct an automaton with states $Q_1 \times Q_2$. While in the standard pairing construction both automata move when a symbol is read, we now choose nondeterministically one of the two automata, which is to move accordingly to the symbol read, while the other one does not change its state, i.e.,

$$\delta((q, q'), a) := \{(\delta_1(q, a), q'), (q, \delta_2(q', a))\}.$$

It is left to the reader to show that this automaton indeed accepts exactly $L_1 \parallel L_2$.

**Solution 5.5**

Let $A = (Q, \Sigma, \delta, q_0, F)$ and $B = (Q', \Sigma, \delta', q'_0, F')$. Intuitively, we build a DFA $C$ that alternates between reading a letter in $A$ and reading a letter in $B$. To do so, we build two copies of the product of $A$ and $B$. Reading a letter $a$ in the first copy simulates reading $a$ in $A$ and then goes to the bottom copy, and vice versa. A word is accepted if it ends up in a state $(p, q)$ of the top copy such that $p \in F$ and $q \in F'$.

Formally, $C = (Q'', \Sigma, \delta'', q''_0, F'')$ where

- $Q'' = Q \times Q' \times \{\top, \bot\}$,
- $\delta(p, a) = \begin{cases} (\delta(q, a), q', \bot) & \text{if } p = (q, q', r) \text{ and } r = \top, \\ (q, \delta'(q', a), \top) & \text{if } p = (q, q', r) \text{ and } r = \bot, \end{cases}$
- $F'' = \{(q, q', \top) : q \in F \text{ and } q' \in F'\}$.

As for most constructions, some states of $C$ may be non reachable from the initial state. We give an algorithm that avoids this:

---

**Input:** DFAs $A = (Q, \Sigma, \delta, q_0, F)$ and $B = (Q', \Sigma, \delta', q'_0, F')$.
**Output:** A DFA $C = (Q'', \Sigma, \delta'', q''_0, F'')$ such that $L(C) = L(A) \mathbin{\widetilde{\sqcup\!\sqcup}} L(B)$.

$Q'' \leftarrow \emptyset$
$\delta'' \leftarrow \emptyset$
$F'' \leftarrow \emptyset$
$W \leftarrow \{(q_0, q'_0, \top)\}$
**while** $W \neq \emptyset$ **do**
    **pick** $p = (q, q', r)$ **from** $W$
    **add** $p$ **to** $Q''$
    **if** $q \in F$, $q' \in F'$ and $r = \top$ **then**
        **add** $p$ **to** $F''$
    **for** $a \in \Sigma$ **do**
        **if** $r = \top$ **then**
            $p' \leftarrow (\delta(q, a), q', \bot)$
        **else if** $r = \bot$ **then**
            $p' \leftarrow (q, \delta(q', a), \top)$
        **add** $(p, a, p')$ **to** $\delta''$
        **if** $p' \notin Q''$ **then add** $p'$ **to** $W$
**return** $(Q'', \Sigma, \delta'', (q_0, q'_0, \top), F'')$

---