Automata and Formal Languages — Exercise Sheet 1

Exercise 1.1
Give a regular expression and a NFA for the language of all words over $\Sigma = \{a, b\}$ ...

1. ...beginning and ending with different letters.
2. ...with the third letter from the right being an a.
3. ...with no occurrences of the subword aa.
4. ...containing at most one occurrence of aa.
5. ...that can be obtained from $bbaba$ by deleting letters.

Exercise 1.2

1. Let A and B be two languages. Prove $A \subseteq B \Rightarrow A^* \subseteq B^*$.
2. Prove that the languages of the regular expressions $((a + ab)^* + b^*)^*$ and Σ^* are equal, where $\Sigma = \{a, b\}$ and we write Σ^* for $(a + b)^*$.

Exercise 1.3
Consider the language $L \subseteq \{a, b\}^*$ given by the regular expression $a^*b^*a^*a$.

1. Give an NFA-ε that accepts L.
2. Give an NFA that accepts L.
3. Give a DFA that accepts L.

Exercise 1.4
The *reverse* of a word $w \in \Sigma^*$ is defined as

$$w^R = \begin{cases} \varepsilon & \text{if } w = \varepsilon, \\ a_n a_{n-1} \cdots a_1 & \text{if } w = a_1 a_2 \cdots a_n \text{ where each } a_i \in \Sigma. \end{cases}$$

The *reverse* of a language $L \subseteq \Sigma^*$ is defined as $L^R = \{w^R \mid w \in L\}$.

(a) Give a regular expression for the reverse of $\big((a + ba)^* ba(a + b)\big)^* ba$.

(b) Give an algorithm that takes as input a regular expression r and returns a regular expression r^R such that $\mathcal{L}(r^R) = \big(\mathcal{L}(r)\big)^R$.

(c) Let A be an NFA. Describe an NFA B such that $L(B) = L(A)^R$.

(d) Does your construction in (c) work for DFAs as well? More precisely, does it preserve determinism?

Solution 1.1

We write Σ^* for $(a+b)^*$.

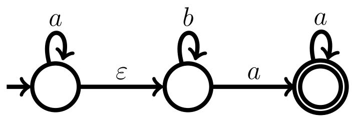
1. $a\Sigma^*b + b\Sigma^*a$
2. $\Sigma^*a\Sigma\Sigma$
3. $(b+ab)^*(\varepsilon+a)$
4. $(b+ab)^*\,\big(aa+\varepsilon\big)\,\big(\varepsilon+b(b+ab)^*(\varepsilon+a)\big)$
5. $(b+\varepsilon)(b+\varepsilon)(a+\varepsilon)(b+\varepsilon)(a+\varepsilon)$

Solution 1.2

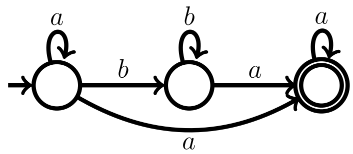
1. Let $w \in A^*$. If $w = \varepsilon$ then it is trivially in B^*. Otherwise, there exists an index $n > 0$ and words $v_1, \ldots, v_n \in A$ such that $w = v_1 \ldots v_n$. Since $A \subseteq B$, we know that for every $1 \leq i \leq n$, v_i is also in B and so $w = v_1 \ldots v_n \in B^*$.
2. The language Σ^* contains all the words written over alphabet Σ so in particular the language of regular expression $((a+ab)^* + b^*)^*$. Now for the other direction, we use the result of 1. with $A = L(a+b)$ and $B = L((a+ab)^* + b^*)$. It is easy to see that the two words of $A = \{a, b\}$ are in language B.

Solution 1.3

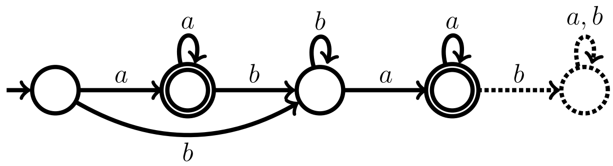
1. NFA-ε accepting L:



2. NFA accepting L:



3. DFA accepting L:



Solution 1.4

(a) $ab\big((a+b)ab(a+ab)^*\big)^*$

(b) We define r^R inductively, which immediately yields a recursive algorithm.

- If $r = \emptyset$, $r = \varepsilon$, or $r = a$ for some letter a, then $r^R = r$.
- If $r = r_1 + r_2$, then $r^R = r_1^R + r_2^R$.
- If $r = r_1 r_2$ then $r^R = r_2^R r_1^R$.
- If $r = r_1^*$, then $r^R = (r_1^R)^*$.

The proof of $L(r^R) = \big(L(r)\big)^R$ is an easy induction.

(c) We reverse the transitions of A and swap its initial and final states. More formally, let $A = (Q, \Sigma, \delta, Q_0, F)$. We define B as $B = (Q, \Sigma, \delta', F, Q_0)$ where $\delta'(p, a) = \{q \in Q \mid p \in \delta(q, a)\}$.

(d) No, if A is deterministic, then B is not necessarily deterministic. For example, the construction applied to the DFA of #1.2(a) for M_2 does not yield a DFA.