Automata and Formal Languages — Homework 5

Due 21.11.2017

Exercise 5.1
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.

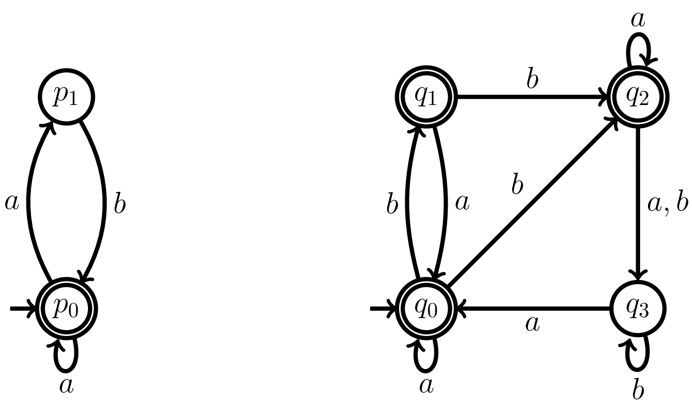
(b) Decide *algorithmically* whether $baabba \in L(A_2)$ and $baabaa \in L(A_2)$.

(c) If you make final and non final states of A_n respectively non final and final, do you obtain an NFA that accepts $\overline{L_n}$? Justify your answer.

(d) Show that $ww \notin L_n$ for every $w \in \{a, b\}^{n+1}$.

(e) Show that any NFA accepting $\overline{L_n}$ has at least 2^{n+1} states. [Hint:]

Exercise 5.2
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 5.3

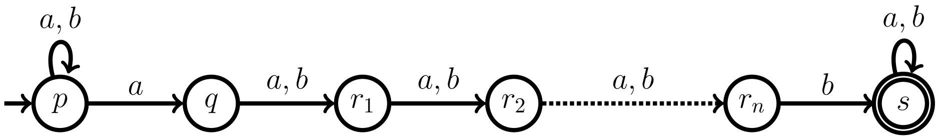
(a) We have seen that testing whether two NFAs accept the same language can be done by using algorithm *InclNFA* twice. Give an alternative algorithm, based on pairings, for testing equality.

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

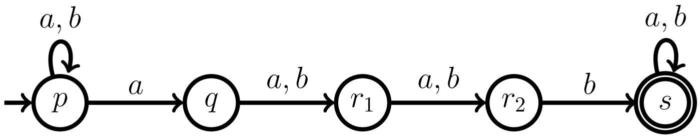
(c) Show that the problem of determining whether an NFA and a DFA accept the same language is PSPACE-hard.

Solution 5.1

(a)



(b) The automaton A_2 is as follows:



Automaton A_2 accepts $w = baabba$ since reading w in the DFA obtained from A_2 yields:

$$\{p\} \xrightarrow{b} \{p\} \xrightarrow{a} \{p, q\} \xrightarrow{a} \{p, q, r_1\} \xrightarrow{b} \{p, r_1, r_2\} \xrightarrow{b} \{p, r_2, s\} \xrightarrow{a} \{p, q, s\}$$

where s is final. However, A_2 rejects $w' = baabaa$ since reading w' in the DFA obtained from A_2 yields:

$$\{p\} \xrightarrow{b} \{p\} \xrightarrow{a} \{p, q\} \xrightarrow{a} \{p, q, r_1\} \xrightarrow{b} \{p, r_1, r_2\} \xrightarrow{a} \{p, q, r_2\} \xrightarrow{a} \{p, q, r_1\}$$

where none of p, q and r_1 are final.

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

(d) Let $w \in \{a, b\}^*$ be such that $|w| = n + 1$. Assume for the sake of contradiction that $ww \in L_n$. There exist $x, y, z \in \{a, b\}^*$ such that $ww = xaybz$ and $|y| = n$. Let $i = |x|$ and $j = |z|$. We have $w_{i+1} = a$ and $w_{|w|-j} = b$. Moreover,

$$i + 1 + n + 1 + j = |xaybz| = |ww| = 2(n + 1).$$

Therefore, $i + 1 = n + 1 - j = |w| - j$, which leads to a contradiction since $a = w_{i+1} = w_{|w|-j} = b$. □

(e) Assume there exists an NFA $B_n = (Q, \{a, b\}, \delta, Q_0, F)$ such that $L(B_n) = \overline{L_n}$ and $|Q| < 2^{n+1}$. Let $W = \{w \in \{a, b\}^* : |w| = n + 1\}$. By (b), $ww \in \overline{L_n}$ for every word $w \in W$. Therefore, for every $w \in W$, there exist $p_w \in Q_0$, $q_w \in Q$ and $r_w \in F$ such that

$$p_w \xrightarrow{w} q_w \xrightarrow{w} r_w.$$

Since $|W| = 2^{n+1}$, by the pigeonhole principle, there exist $w, w' \in W$ such that $w \neq w'$ and $q_w = q_{w'}$. Since $w \neq w'$, there exists $1 \leq i \leq n+1$ such that $w_i \neq w'_i$. Without loss of generality, $w_i = a$ and $w'_i = b$. Thus, $ww' = uavu'bv'$ for some $u, v, u', v' \in \{a, b\}^*$ such that $|v| = n + 1 - i$ and $|u'| = i - 1$. Therefore, $|vu'| = n$ which implies that $ww' \in L_n$. This is a contradiction since

$$p_w \xrightarrow{w} q_w = q_{w'} \xrightarrow{w'} r_{w'} \text{ and } r_{w'} \in F.$$

□

Solution 5.2

(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	$\{\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}\}$	$\emptyset$

At the third iteration, the algorithm encounters state $\{q_3\}$ which is 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 $L(A) \subseteq L(B)$.

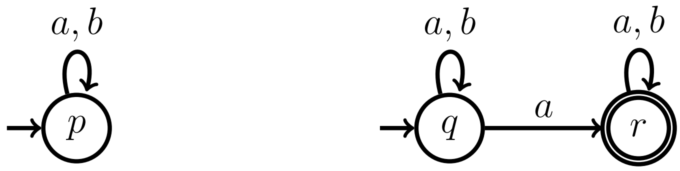
Solution 5.3

(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*.

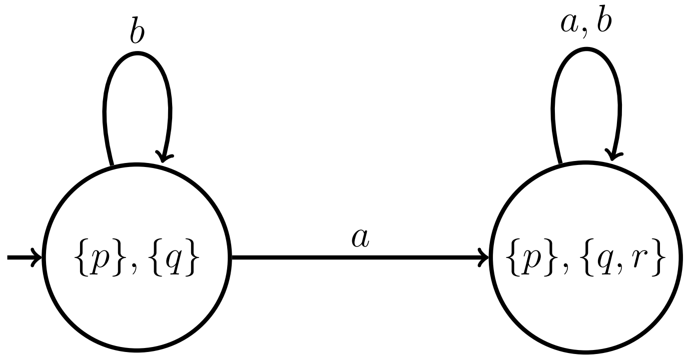
Input: NFAs $A = (Q, \Sigma, \delta, Q_0, F)$ and $A' = (Q', \Sigma, \delta', Q'_0, F')$.
Output: $L(A) = L(A')$?

$Q \leftarrow \emptyset$
$W \leftarrow \{[Q_0, Q'_0]\}$
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 Σ^*. 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.