Automata and Formal Languages — Homework 5

Due 18.11.2016

Exercise 5.1

Let $L_n \subseteq \{a, b\}^*$ be the language described by the regular expression $(a + b)^* a(a + b)^n b(a + b)^*$.

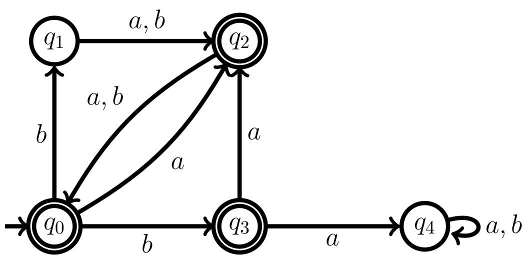
(a) Give an NFA with $n + 3$ states that accepts L_n.

(b) Show that for every $w \in \{a, b\}^*$, if $|w| = n + 1$, then $ww \notin L_n$.

(c) Show that any NFA accepting $\overline{L_n}$ has at least 2^{n+1} states. (Hint: use (b) and the pigeonhole principle.)

Exercise 5.2

Use the algorithm *UnivNFA* to test whether the following NFA is universal.



Exercise 5.3

(a) Build B_p and C_p for the word pattern $p = mammamia$.

(b) How many transitions are taken when reading $t = mami$ in B_p and C_p?

(c) Let $n > 0$. Find a text $t \in \{a, b\}^*$ and a word pattern $p \in \{a, b\}^*$ such that testing whether p occurs in t takes n transitions in B_p and $2n - 1$ transitions in C_p.

Exercise 5.4

Two-way DFAs are an extension of lazy automata where the reading head is also allowed to move left. Formally, a *two-way DFA (2DFA)* is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times \{L, S, R\}$. Given a word $w \in \Sigma^*$, A starts in q_0 with its reading tape initialized with $\vdash w \dashv$, and its reading head pointing on $\vdash$. When reading a letter, A moves the head according to δ (*L*eft, *S*tationnary, *R*ight). Moving left on $\vdash$ or right on $\dashv$ does not move the reading head. A accepts w if, and only if, it reaches $\dashv$ in a state of F.

(a) Let $n \in \mathbb{N}$. Give a 2DFA that accepts $(a + b)^* a(a + b)^n$.

(b) Give a 2DFA that does not terminate on any input.

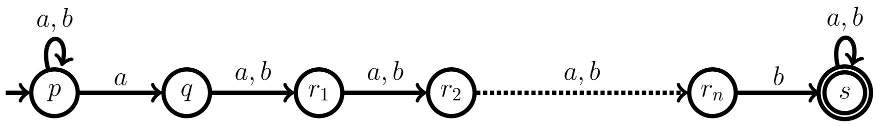
(c) Describe an algorithm to test whether a given 2DFA A accepts a given word w.

(d) Let $A_1, A_2, \ldots, A_n$ be DFAs over a common alphabet. Give a 2DFA B such that

$$L(B) = L(A_1) \cap L(A_2) \cap \cdots \cap L(A_n) \ .$$

Solution 5.1

(a)



(b) 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

$$i + 1 + n + 1 + j = 2(n+1) \ ,$$

hence $i + j = n$. Therefore, $w_{i+1} = a$ and $w_{n+1-j} = b$. We have $n + 1 - j = i + 1$. This implies that $a = w_{i+1} = w_{n+1-j} = b$ which is a contradiction. $\square$

(c) Assume there exists an NFA $A_n = (Q, \{a,b\}, \delta, Q_0, F)$ such that $L(A_n) = \overline{L_n}$ 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 exist $1 \leq i \leq n+1$ such that $w_i \neq w'_i$. Without loss of generality, $w_i = a$ and $w'_i = b$. Thus, $ww' = uau'vbv'$. Moreover $|u'| = n - i + 1$ and $|v| = i - 1$. Therefore, $|u'v| = n$ which implies that $ww' \in L_n$. This is a contradiction, since $p_w \xrightarrow{w} q_{w'} \xrightarrow{w'} r_{w'}$ and $r_{w'} \in F$. $\square$

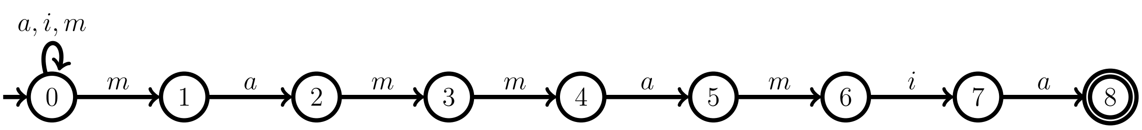
Solution 5.2

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

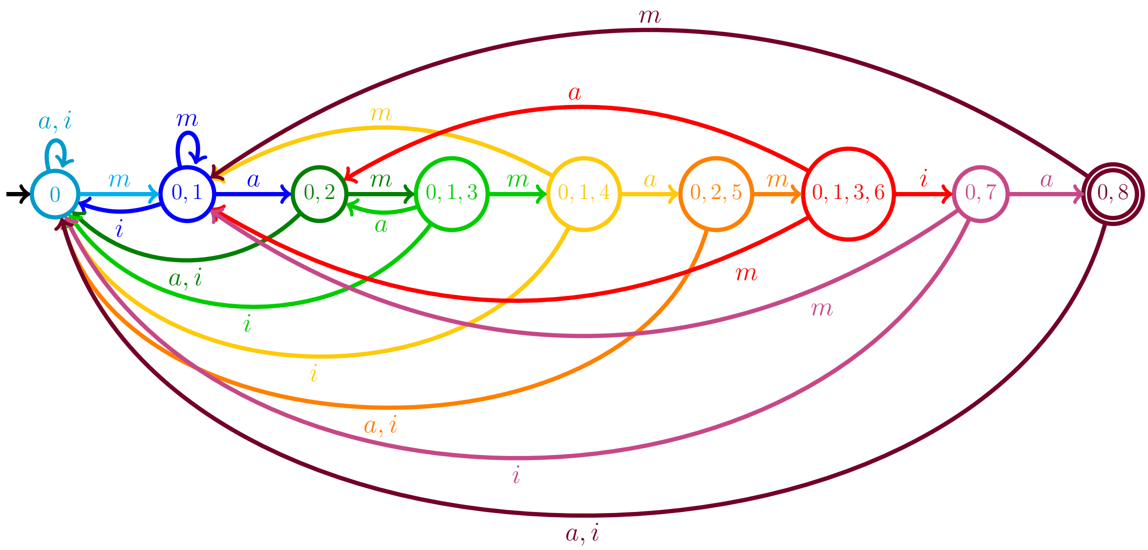
The algorithm returns true, hence the NFA accepts $\{a,b\}^*$.

Solution 5.3

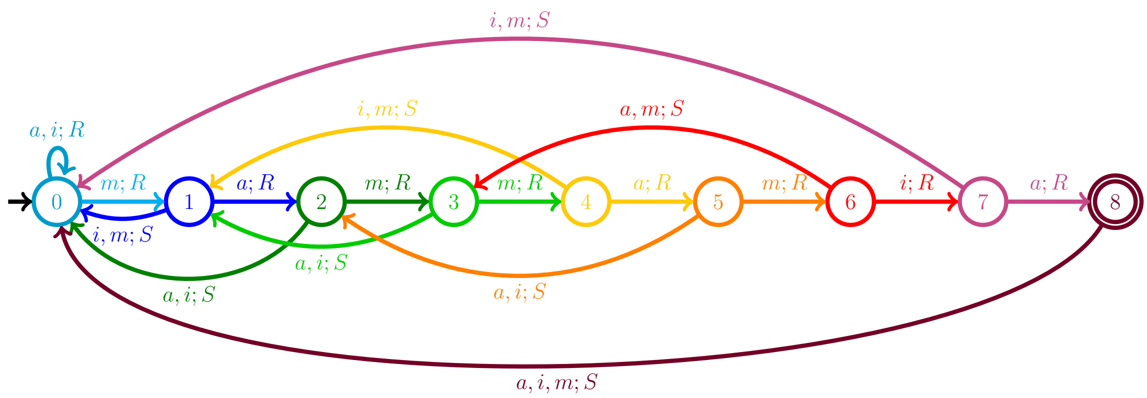
(a) A_p :



B_p :



C_p :

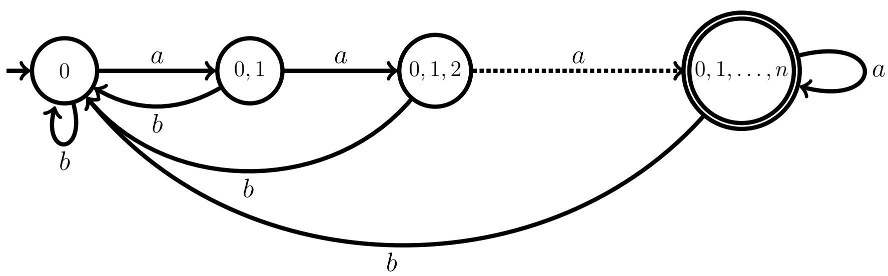


(b) Four transitions taken in B_p: $\{0\} \xrightarrow{m} \{0,1\} \xrightarrow{a} \{0,2\} \xrightarrow{m} \{0,1,3\} \xrightarrow{i} \{0\}$.

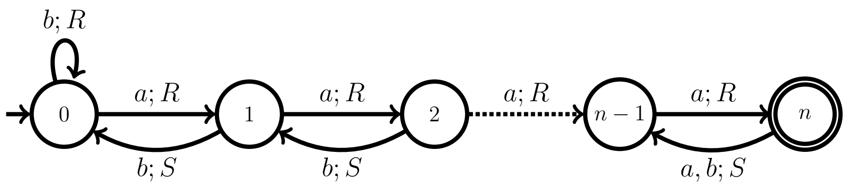
Six transitions taken in C_p: $0 \xrightarrow{m} 1 \xrightarrow{a} 2 \xrightarrow{m} 3 \xrightarrow{i} 1 \xrightarrow{i} 0 \xrightarrow{i} 0$.

(c) $t = a^{n-1}b$ and $p = a^n$. The automata B_p and C_p are as follows:

B_p:



C_p:



The runs over t on B_p and C_p are respectively:

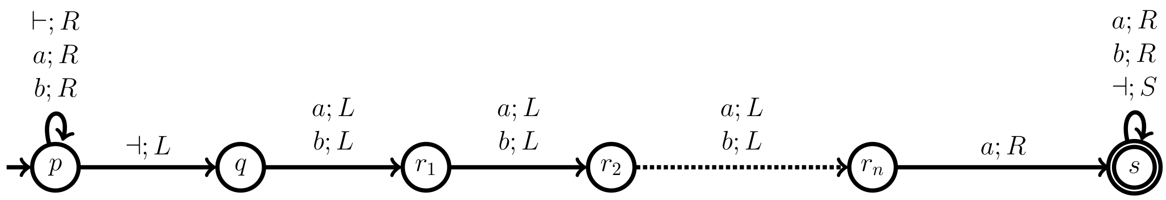
$$\{0\} \xrightarrow{a} \{0,1\} \xrightarrow{a} \{0,1,2\} \xrightarrow{a} \cdots \xrightarrow{a} \{0,1,\ldots,n-1\} \xrightarrow{b} \{0\} ,$$

and

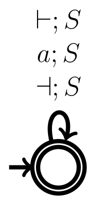
$$0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} \cdots \xrightarrow{a} (n-1) \xrightarrow{b} (n-2) \xrightarrow{b} (n-3) \xrightarrow{b} \cdots \xrightarrow{b} 0 .$$

Solution 5.4

(a) The following 2DFA accepts $(a+b)^* a (a+b)^n$. Transitions not drawn lead to a trap state without moving the head.



(b)



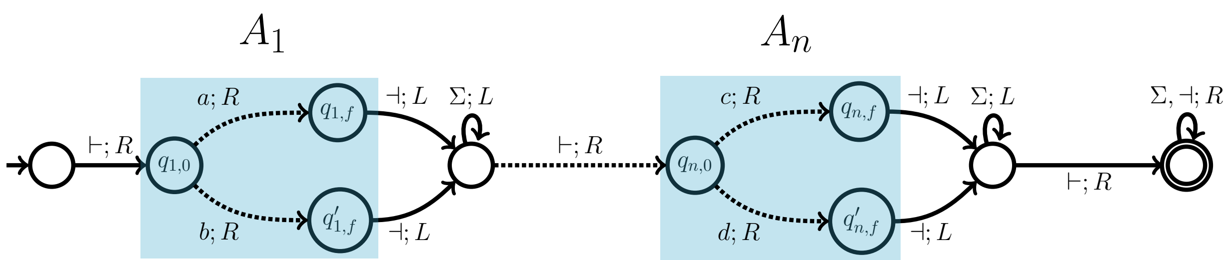
(c) From (b), we know that simply reading an input word is not sufficient since the automaton could loop forever. Instead, we keep track of all configurations that are encountered when reading the input word w. A configuration is a pair (q, i) where q is a state and $0 \leq i \leq |w| + 1$ is a position of the reading head. If $(q_f, |w| + 1)$ where $q_f \in F$ is encountered, then the automaton accepts w. If a configuration is seen twice, then the automaton loops forever.

We obtain the following algorithm:

```
Input: 2DFA A = (Q, Σ, δ, q0, F) and w ∈ Σ*.
Output: w ∈ L(A)?
W ← ∅
q ← q0
i ← 0
while (q, i) ∉ W do
    if q ∈ F and i = |w| + 1 then                 /* Final configuration? */
        return true
    if i = 0 then                                 /* Compute next state */
        q, d ← δ(q, ⊢)
    else if i = |w| + 1 then
        q, d ← δ(q, ⊣)
    else
        q, d ← δ(q, w_i)
    if d = L and i > 0 then                       /* Compute next position */
        i ← i − 1
    else if d = R and i ≤ |w| then
        i ← i + 1
return false
```

(d) We build a 2DFA B that first simulates A_1 on w. If a final state of A_1 is reached in $\dashv$, then B rewinds the tape. B then repeat this process on $A_2, \ldots, A_n$. If every A_i accepts w, then B finally move the reading head to $\dashv$ in a final state.

The construction looks as follows:



Let $A_i = (Q_i, \Sigma, \delta_i, q_{i,0}, F_i)$. Formally, B is defined as $B = (Q, \Sigma, \delta, \{p\}, \{r\})$ where

- $Q = \{p, s\} \cup Q_1 \cup Q_2 \cup \cdots \cup Q_n \cup \{r_i : 1 \leq i \leq n\}$,
- $\delta(q, a) = \begin{cases} (q_{1,0}, R) & \text{if } q = p \text{ and } a = \vdash, \\ (\delta_i(q, a), R) & \text{if } q \in Q_i \text{ and } a \in \Sigma, \\ (r_i, L) & \text{if } q \in F_i \text{ and } a = \dashv, \\ (r_i, L) & \text{if } q = r_i \text{ and } a \in \Sigma, \\ (q_{i+1,0}, R) & \text{if } q = r_i,\ a = \vdash \text{ and } 1 \leq i < n, \\ (s, R) & \text{if } q = r_n,\ a = \vdash, \\ (s, R) & \text{if } q = s,\ a \in \Sigma \cup \{\dashv\}. \end{cases}$

★ It is known that the *intersection problem*, which is defined as follows, is PSPACE-complete [3]:

Given: DFAs $A_1, A_2, \ldots, A_n$,
Decide: whether $L(A_1) \cap L(A_2) \cap \cdots \cap L(A_n)$.

We have seen how to build a 2DFA B such that $L(B) = L(A_1) \cap L(A_2) \cap \cdots \cap L(A_n)$, in polynomial time. Thus, testing emptiness for 2DFAs is "at least as hard" as the intersection problem, i.e. it is PSPACE-hard. In fact, the emptiness problem for 2DFAs is PSPACE-complete [1, 2].

References

[1] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[2] H. B. III Hunt. On the time and tape complexity of languages I. In *Proc. 5th Annual ACM Symposium on Theory of Computing (STOC)*, pages 10–19, 1973. Available online at `https://ecommons.cornell.edu/handle/1813/6007`.

[3] Dexter Kozen. Lower bounds for natural proof systems. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 254–266, 1977. Available online at `http://www.cs.cornell.edu/~kozen/papers/LowerBounds.pdf`.