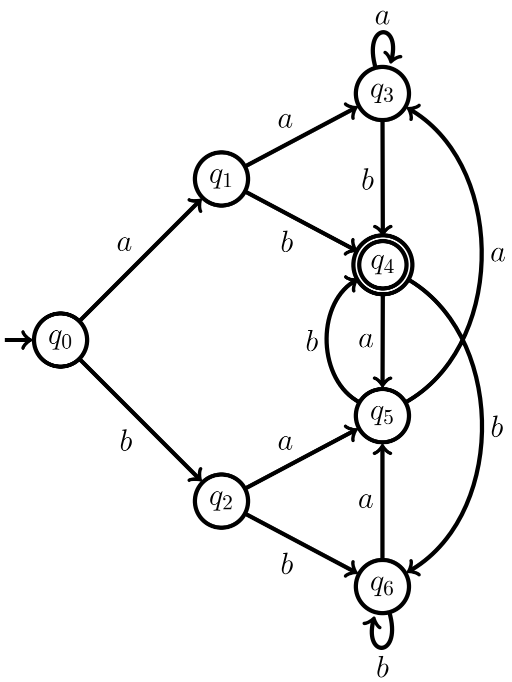
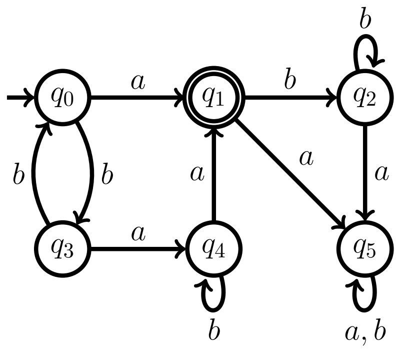
Technische Universität München **Winter term 21/22**
I7
Prof. J. Esparza / A. R. Balasubramanian / C. Weil-Kennedy

Automata and Formal Languages — Exercise Sheet 4

Exercise 4.1
Let A and B be respectively the following DFAs:



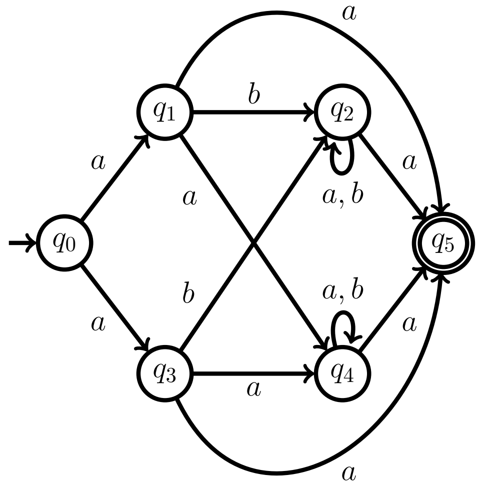


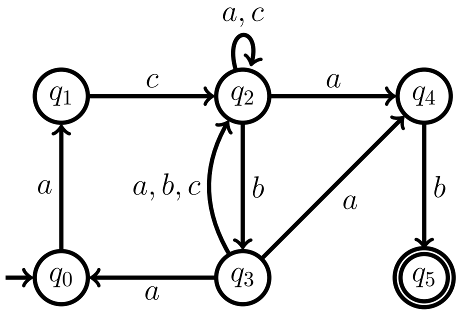
(a) Compute the language partitions of A and B.

(b) Construct the quotients of A and B with respect to their language partitions.

(c) Give regular expressions for $L(A)$ and $L(B)$.

Exercise 4.2
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

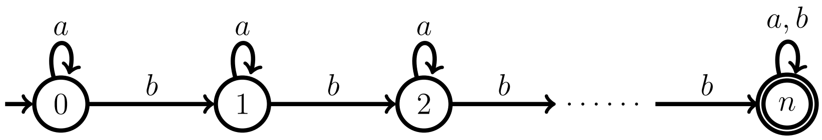
$$L(A) = \{w \in \{a,b\}^* : |w| \geq 2 \text{ and } w \text{ starts and ends with } a\}$$
$$L(B) = \{w \in \{a,b,c\}^* : w \text{ starts with } ac \text{ and ends with } ab\}$$

(d) Are the automata obtained in (b) minimal?

Exercise 4.3
We consider Hopcroft's algorithm for minimization of DFAs, except we replace all occurences of 'min' in the algorithm with 'max', i.e., in Line 3, we initialize the workset as $\{(a, \max\{F, Q \setminus F\}) : a \in \Sigma\}$ and in Line 10, we add $(b, \max\{B_0, B_1\})$ to the workset. Call this algorithm *MinMax*.

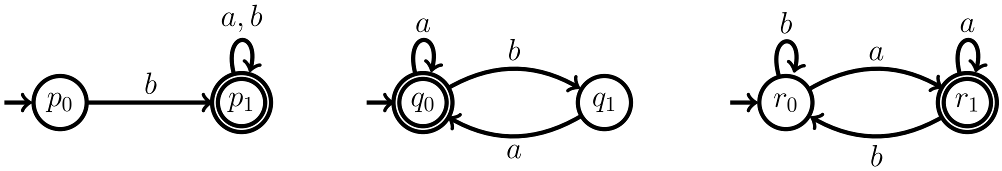
The size of a splitter (a, B) is the size of the set B. For both Hopcroft's algorithm and the MinMax algorithm, the *amount of work* done on a DFA A is defined as the size of all splitters that were added to the workset at any point during the execution of the algorithm on the DFA A. (Assume that in both algorithms the workset is maintained as a queue).

Let $n \geq 1$ and consider the following DFA A_n:



Show that the amount of work done by MinMax on the DFA A_n is $\Theta(n^2)$, whereas the amount of work done by Hopcroft's algorithm on A_n is $\Theta(n)$.

Exercise 4.4
Consider the following DFAs A, B and C:



Use pairings to decide *algorithmically* whether $L(A) \cap L(B) \subseteq L(C)$.

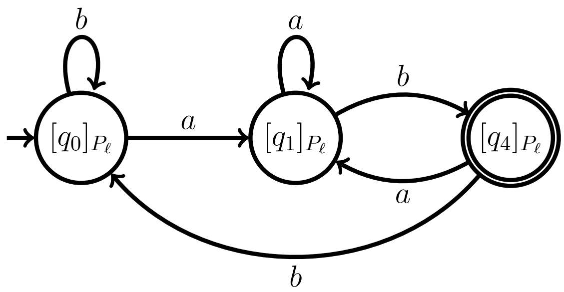
Solution 4.1

A) (a)

Iter.	Block to split	Splitter	New partition
0	—	—	$\{q_0, q_1, q_2, q_3, q_5, q_6\}, \{q_4\}$
1	$\{q_0, q_1, q_2, q_3, q_5, q_6\}$	$(b, \{q_4\})$	$\{q_0, q_2, q_6\}, \{q_1, q_3, q_5\}, \{q_4\}$
2	none, partition is stable	—	—

The language partition is $P_\ell = \{\{q_0, q_2, q_6\}, \{q_1, q_3, q_5\}, \{q_4\}\}$.

(b)



(c) In the above automaton, notice that $\delta(q, a) = [q_1]_{P_\ell}$ for every state q and $\delta([q_1]_{P_\ell}, b) = [q_4]_{P_\ell}$. Hence, every word in $(a + b)^*ab$ is accepted by this automaton.

Further, notice that the only way to reach $[q_4]_{P_\ell}$ is by reading a b from $[q_1]_{P_\ell}$ and the only way to reach $[q_1]_{P_\ell}$ from any state is by reading an a. It follows that if a word is accepted by this automaton, then that word must belong to the language of $(a + b)^*ab$.

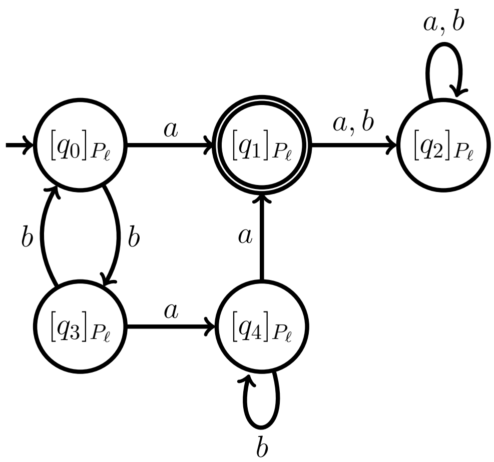
It follows that $(a + b)^*ab$ is a regular expression for this automaton.

B) (a)

Iter.	Block to split	Splitter	New partition
0	—	—	$\{q_0, q_2, q_3, q_4, q_5\}, \{q_1\}$
1	$\{q_0, q_2, q_3, q_4, q_5\}$	$(a, \{q_1\})$	$\{q_0, q_4\}, \{q_2, q_3, q_5\}, \{q_1\}$
2	$\{q_2, q_3, q_5\}$	$(a, \{q_0, q_4\})$	$\{q_0, q_4\}, \{q_2, q_5\}, \{q_3\}, \{q_1\}$
3	$\{q_0, q_4\}$	$(b, \{q_3\})$	$\{q_0\}, \{q_4\}, \{q_2, q_5\}, \{q_3\}, \{q_1\}$
4	none, partition is stable	—	—

The language partition is $P_\ell = \{\{q_0\}, \{q_1\}, \{q_2, q_5\}, \{q_3\}, \{q_4\}\}$.

(b)



(c) One can show that any word in $(bb)^*a + b(bb)^*ab^*a$ is accepted by this automaton. Further, there are only way two ways to reach the final state from the initial state: Either alternate between $[q_0]_{P_\ell}$ and $[q_3]_{P_\ell}$ by reading b's and then move to $[q_1]_{P_\ell}$ from $[q_0]_{P_\ell}$ by reading an a, or first alternate between

$[q_0]_{P_\ell}$ and $[q_3]_{P_\ell}$ by reading b's, then move to $[q_4]_{P_\ell}$ from $[q_3]_{P_\ell}$ by reading an a, then use the self-loop at $[q_4]_{P_\ell}$ to stay there by reading b's and then finally move to $[q_1]_{P_\ell}$ from $[q_4]_{P_\ell}$ by reading an a. The former gives rise to words in $(bb)^*a$ and the latter gives rise to words in $b(bb)^*ab^*a$. Hence, $(bb)^*a + b(bb)^*ab^*a$ is a regular expression for this automaton.

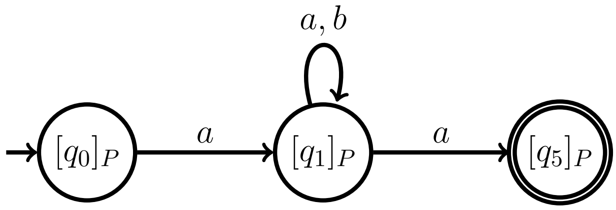
Solution 4.2

A) (a)

Iter.	**Block to split**	**Splitter**	**New partition**
0	—	—	$\{q_0, q_1, q_2, q_3, q_4\}, \{q_5\}$
1	$\{q_0, q_1, q_2, q_3, q_4\}$	$(a, \{q_5\})$	$\{q_0\}, \{q_1, q_2, q_3, q_4\}, \{q_5\}$
2	none, partition is stable	—	—

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

(b)



(c) It follows immediately from the fact that A accepts the same language as the automaton obtained in (b).

(d) Yes. By (c), the language accepted by A is $a(a+b)^*a$. 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 $\varepsilon \notin L(A)$,
- $q_1 \in \delta(q_0, a)$, otherwise it is impossible to accept aa which is in $L(A)$.

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

B) (a)

Iter.	**Block to split**	**Splitter**	**New partition**
0	—	—	$\{q_0, q_1, q_2, q_3, q_4\}, \{q_5\}$
1	$\{q_0, q_1, q_2, q_3, q_4\}$	$(b, \{q_5\})$	$\{q_0, q_1, q_2, q_3\}, \{q_4\}, \{q_5\}$
2	$\{q_0, q_1, q_2, q_3\}$	$(a, \{q_4\})$	$\{q_0, q_1\}, \{q_2, q_3\}, \{q_4\}, \{q_5\}$
3	$\{q_0, q_1\}$	$(c, \{q_2, q_3\})$	$\{q_0\}, \{q_1\}, \{q_2, q_3\}, \{q_4\}, \{q_5\}$
4	$\{q_2, q_3\}$	$(a, \{q_0\})$	$\{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_4\}, \{q_5\}$

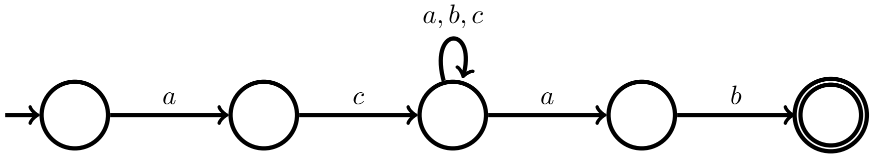
The CSR is $P = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_4\}, \{q_5\}\}$.

(b) The automaton remains unchanged.

(c) $\subseteq$) Let $w \in L(B)$. Every path from q_0 to q_5 first goes through q_1 and q_2 and ends up going through q_4 and q_5. This implies that $w \in L(ac(a+b+c)^*ab)$.

$\supseteq$) First note that for every $u \in \{a, b, c\}^*$, there exists $q \in \{q_2, q_3\}$ such that $q_2 \xrightarrow{u} q$. This can be shown by induction on $|u|$. Let $w \in L(ac(a+b+c)^*ab)$. There exists $u \in \{a, b, c\}^*$ such that $w = acuab$. Let $q \in \{q_2, q_3\}$ be such that $q_2 \xrightarrow{u} q$. We have $q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_2 \xrightarrow{u} q \xrightarrow{a} q_4 \xrightarrow{b} q_5$. Therefore, $w \in L(B)$. □

(d) No. The following NFA with five states accepts the same language.



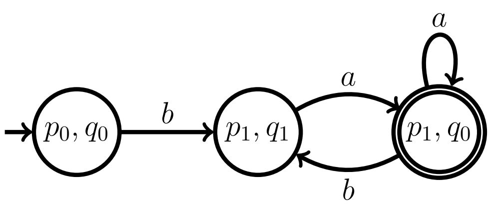
Solution 4.3

Let us consider Hopcroft's algorithm. Initially, the workset contains the pairs $(a, \{n\})$ and $(b, \{n\})$. Then irrespective of the order in which we pick these splitters, the next pairs to be added are $(a, \{n-1\})$ and $(b, \{n-1\})$. Once again, irrespective of the order in which we pick these two splitters, the next pairs to be added are $(a, \{n-2\})$ and $(b, \{n-2\})$ and so on. In this way, after n such additions we will arrive at the partition $\{0\}, \{1\}, \{2\}, \ldots, \{n\}$. Hence, the amount of work done by Hopcroft's algorithm is $\Theta(n)$.

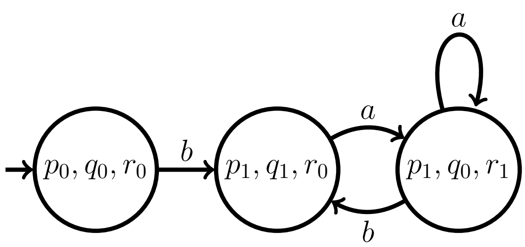
Let us now consider MinMax. Initially, the workset contains the pairs $(a, \{1, 2, \ldots, n-1\})$ and $(b, \{1, 2, \ldots, n-1\})$. Then irrespective of the order in which we pick these splitters, the next pairs to be added are $(a, \{1, 2, \ldots, n-2\})$ and $(b, \{1, 2, \ldots, n-2\})$. Once again, irrespective of the order in which we pick these two splitters, the next pairs to be added are $(a, \{1, 2, \ldots, n-3\})$ and $(b, \{1, 2, \ldots, n-3\})$ and so on. In this way, after n such additions we will arrive at the partition $\{0\}, \{1\}, \{2\}, \ldots, \{n\}$. Hence, the amount of work done by MinMax is $\Theta(n^2)$.

Solution 4.4

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 $(L(A) \cap L(B)) \setminus L(C)$ from the above automaton and C. Once again, it is not necessary to explore the implicit trap states of the automaton for $L(A) \cap L(B)$. We obtain:



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