

Automata and Formal Languages — Exercise Sheet 1

You can find the Automata Tutor course name and password on the Moodle website for “Exercise - Automata and Formal Languages (IN2041)”. If you are enrolled for the course “Exercise - Automata and Formal Languages (IN2041)” in TUM online, you automatically have access to the Moodle website.

Exercise 1.1

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

1. ... with the second letter from the right being an a .
2. ... beginning with ab or ending with ba .
3. ... with at least one occurrence of a and at least one occurrence of b .
4. ... with no occurrence of the subword abb .

You can do this exercise in Automata Tutor to get feedback on your answer. The NFA constructions are under “NFA Construction” and called AFL 1.1 (i), while the regular expression are under “RE Construction” and called AFL 1.1 (i) RE.

Exercise 1.2

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

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

You can do this exercise in Automata Tutor to get feedback on your answer. The first two parts are under “NFA Construction”, the third is under “DFA Construction”.

Exercise 1.3

Consider the regular expression $r = (a + ab)^*$.

- (a) Convert r into an equivalent NFA- ε A .
- (b) Convert A into an equivalent NFA B . (It is not necessary to use algorithm *NFA ε toNFA*)
- (c) Convert B into an equivalent DFA C .
- (d) By inspecting B , give an equivalent minimal DFA D . (No algorithm needed).
- (e) Convert D into an equivalent regular expression r' .
- (f) Prove formally that $L(r) = L(r')$.

Exercise 1.4

Given $n \in \mathbb{N}_0$, let $\text{MSBF}(n)$ be the set of *most-significant-bit-first* encodings of n , i.e., the words that start with an arbitrary number of leading zeros, followed by n written in binary. For example:

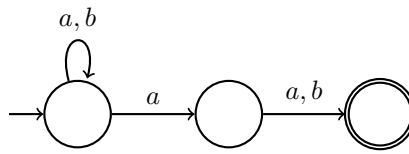
$$\text{MSBF}(3) = 0^*11 \quad \text{and} \quad \text{MSBF}(9) = 0^*1001 \quad \text{MSBF}(0) = 0^*.$$

1. Construct and compare DFAs recognizing the encodings of the even numbers $n \in \mathbb{N}_0$ w.r.t. the unary encoding, where n is encoded by the word 1^n , and the MSBF-encoding.
2. Same for the set of numbers divisible by 3, and for the set of numbers divisible by 4.
3. Give regular expressions corresponding to the languages in 2.

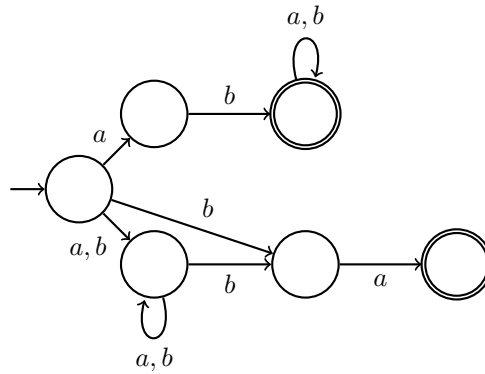
Solution 1.1

We write Σ^* for $(a + b)^*$.

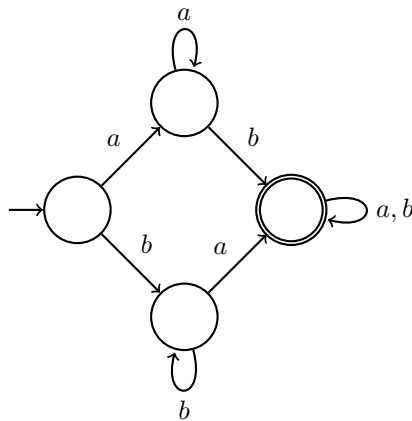
1. $\Sigma^*a\Sigma$



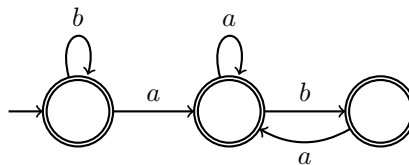
2. $ab\Sigma^* + \Sigma^*ba$



3. $aa^*b\Sigma^* + bb^*a\Sigma^*$

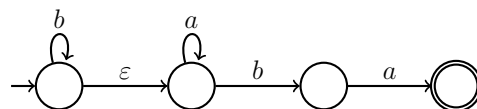


4. $b^*(ab + a)^*$

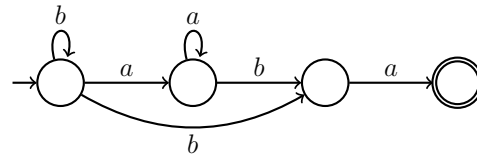


Solution 1.2

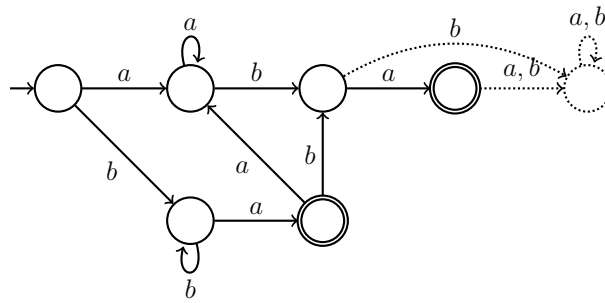
1. NFA- ϵ accepting L :



2. NFA accepting L :



3. DFA accepting L :

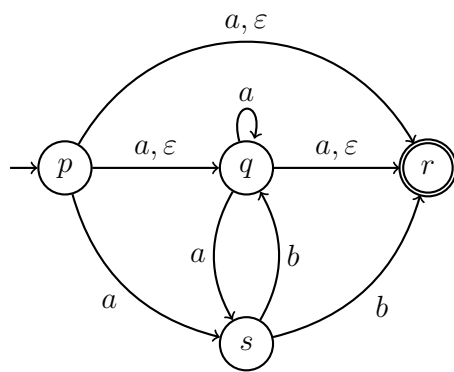
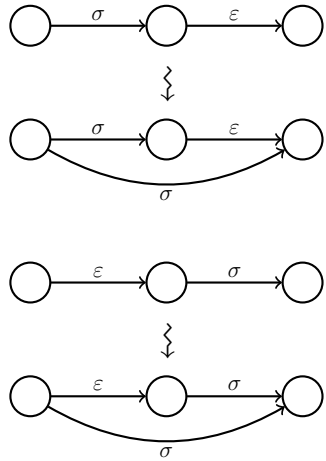
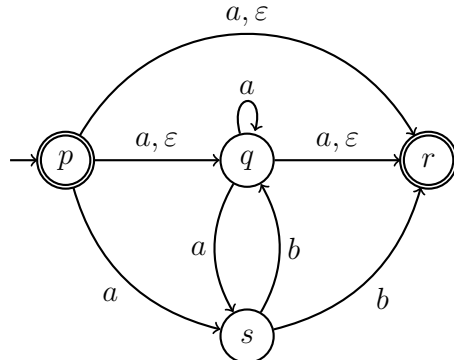
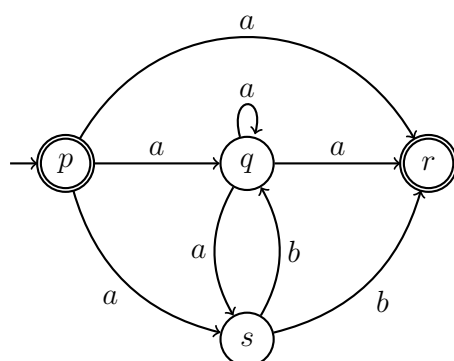


Solution 1.3

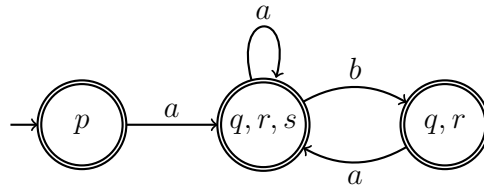
(a)

Iter.	Automaton obtained	Rule applied
1		Initial automaton from reg. expr.
2		
3		
4		

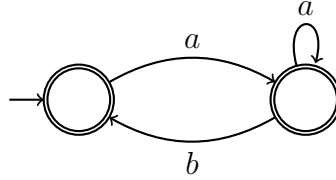
(b)

Iter.	Automaton obtained	Rule applied
1		 <p>where $\sigma \in \Sigma \cup \{\varepsilon\}$</p>
2		<p>Initial states that can reach a final state through ε-transitions are made final.</p>
3		<p>Remove ε-transitions. Remove states non reachable from initial state.</p>

(c)



(d) States $\{p\}$ and $\{q, r\}$ have the exact same behaviours, so we can merge them. Indeed, both states are final and $\delta(\{p\}, \sigma) = \delta(\{q, r\}, \sigma)$ for every $\sigma \in \{a, b\}$. We obtain:



(e)

Iter.	Automaton obtained	Rule applied
1		Add single initial and final states.
2		
3		

4		
5		
6	$\epsilon + a(a+ba)^*(\epsilon+b)$	Extract regular expression from the unique transition.

(f) Let us first show that $a(a+ba)^i = (a+ab)^i a$ for every $i \in \mathbb{N}$. We proceed by induction on i . If $i = 0$, then the claim trivially holds. Let $i > 0$. Assume the claim holds at $i - 1$. We have

$$\begin{aligned}
 a(a+ba)^i &= a(a+ba)^{i-1}(a+ba) \\
 &= (a+ab)^{i-1}a(a+ba) && \text{(by induction hypothesis)} \\
 &= (a+ab)^{i-1}(aa+aba) && \text{(by distributivity)} \\
 &= (a+ab)^{i-1}(a+ab)a && \text{(by distributivity)} \\
 &= (a+ab)^i a.
 \end{aligned}$$

This implies that

$$a(a+ba)^* = (a+ab)^* a. \tag{1}$$

We may now prove the equivalence of the two regular expressions:

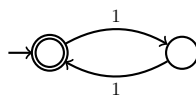
$$\begin{aligned}
 \epsilon + a(a+ba)^*(\epsilon+b) &= \epsilon + (a+ab)^* a(\epsilon+b) && \text{(by (1))} \\
 &= \epsilon + (a+ab)^*(a+ab) && \text{(by distributivity)} \\
 &= \epsilon + (a+ab)^+ \\
 &= (a+ab)^*.
 \end{aligned}$$

□

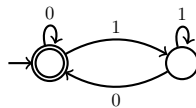
Solution 1.4

1. Here are the DFAs:

- Unary encoding:



- MSBF encoding. The DFA recognizes all strings ending with 0:



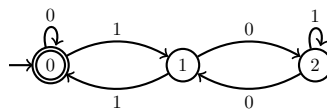
2.

- *Numbers divisible by 3:*

The DFA for the unary encoding is, loosely speaking, a circuit of length three. We now give a DFA for the MSBF encoding. The idea is that the state reached after reading the word u corresponds to the remainder of the number represented by u when dividing by 3. We therefore take as states $\{0, 1, 2\}$ with 0 as both initial and final state. If a word w encodes a number k , then wa encodes the number $2k + a$. So for every state $q \in \{0, 1, 2\}$ we define

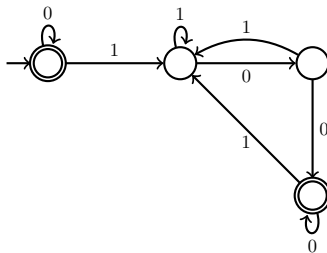
$$\delta(q, a) = 2q + a \pmod{3}.$$

This yields the automaton:



- *Numbers divisible by 4:*

The DFA for the unary encoding is, loosely speaking, a circuit of length four. We now give a DFA for the MSBF encoding. We can do the same as for numbers divisible by 3, with states for the remainders $\{0, 1, 2, 3\}$. Or we can simply rely on the fact that the numbers divisible by 4 are 0 or end with 00 in the MSBF encoding.



3. Regular expressions for the unary encodings are $(1^3)^*$ and $(1^4)^*$, and ones for the MSBF encoding are $(0 + 1(01^*0)^*1)^*$ and $\varepsilon + 0 + (0 + 1)^*00$.