

Automata and Formal Languages — Exercise Sheet 2

Exercise 2.1

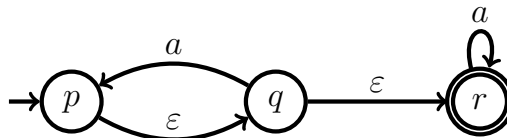
Consider the regular expression $r = (a + ab)^*$.

- (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 2.2

Convert the following NFA- ε to an NFA

- (a) using the naïve method (with saturation, ε -check, ε -transition elimination and normalization).
- (b) using the algorithm *NFA ε toNFA* from the lecture notes (see Sect. 2.3, p. 35). You may verify your answer with the Python program `nfa-eps2nfa`.



Exercise 2.3

Recall that a nondeterministic automaton A accepts a word w if at least one of the runs of A on w is accepting. This is sometimes called the *existential* accepting condition. Consider the variant in which A accepts w if *all* runs of A on w are accepting (in particular, if A has no run on w then it accepts w). This is called the *universal* accepting condition. Notice that a DFA accepts the same language with both the existential and the universal accepting conditions.

Intuitively, we can visualize an automaton with universal accepting condition as executing all runs in parallel. After reading a word w , the automaton is simultaneously in all states reached by all runs labelled by w , and accepts if all those states are accepting.

Consider the family L_n of languages over the alphabet $\{0, 1\}$ given by $L_n = \{ww \in \Sigma^{2n} \mid w \in \Sigma^n\}$.

1. Give an automaton of size $O(n)$ with universal accepting condition that recognizes L_n .
2. Prove that every NFA (and so in particular every DFA) recognizing L_n has at least 2^n states.

3. Give an algorithm that transforms an automaton with universal accepting condition into a DFA recognizing the same language. This shows that automata with universal accepting condition recognize the regular languages.

Exercise 2.4

Prove or disprove:

1. A subset of a regular language is regular.
2. A superset of a regular language is regular.
3. If L_1 and L_1L_2 are regular and $L_1, L_2 \neq \emptyset$, then L_2 is regular.
4. If L_2 and L_1L_2 are regular and $L_1, L_2 \neq \emptyset$, then L_1 is regular.

Exercise 2.5

★ The existential and universal accepting conditions can be combined, yielding *alternating automata*. The states of an alternating automaton are partitioned into *existential* and *universal* states. An existential state q accepts a word w (i.e., $w \in L(q)$) if $w = \varepsilon$ and $q \in F$ or $w = aw'$ and *there exists* a transition (q, a, q') such that q' accepts w' . A universal state q accepts a word w if $w = \varepsilon$ and $q \in F$ or $w = aw'$ and *for every* transition (q, a, q') the state q' accepts w' . The language recognized by an alternating automaton is the set of words accepted by its initial state.

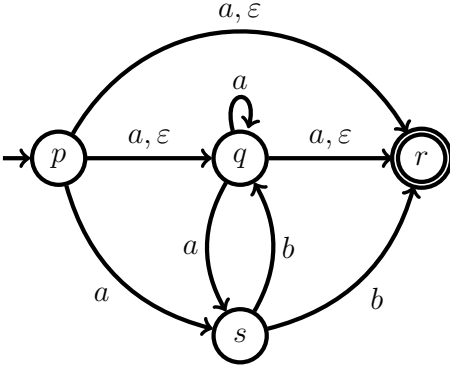
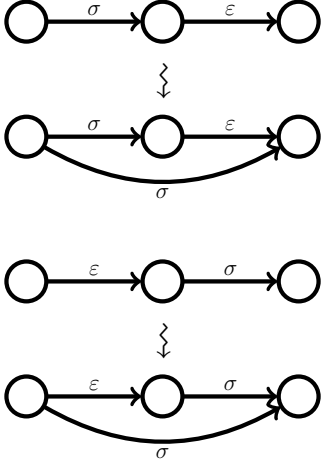
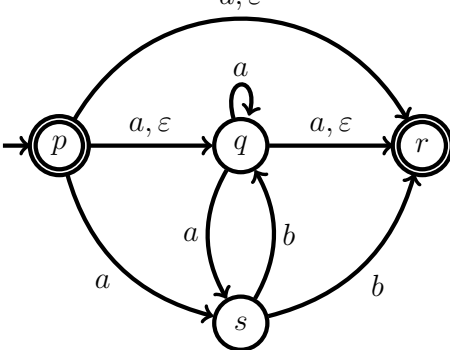
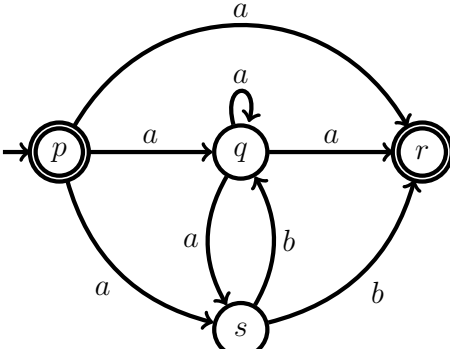
Give an algorithm that transforms an alternating automaton into a DFA recognizing the same language.

Solution 2.1

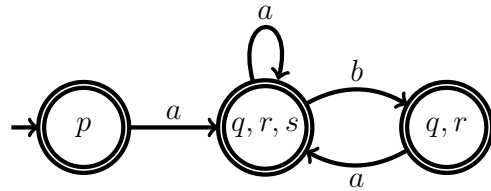
(a)

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

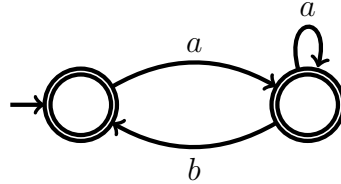
(b)

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	$\varepsilon + a(a + ba)^*(\varepsilon + b)$	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

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

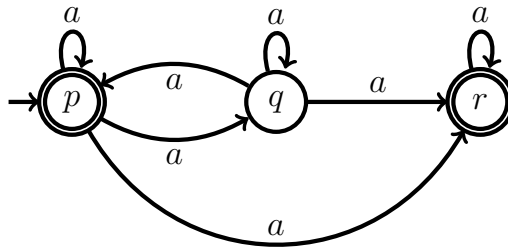
We may now prove the equivalence of the two regular expressions:

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

□

Solution 2.2

(a) The resulting NFA is:



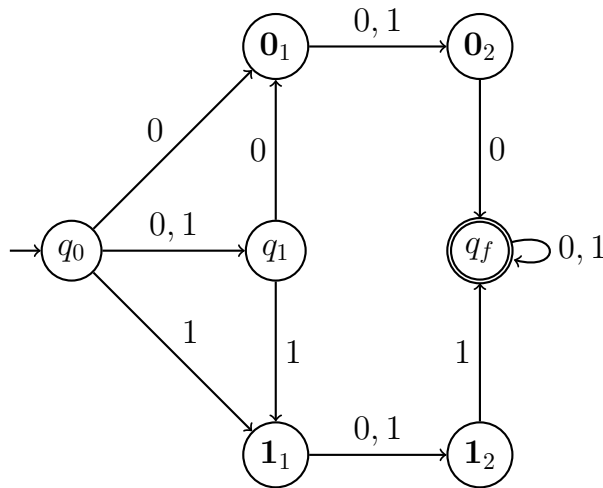
(b) The resulting NFA is the same. And it corresponds to the output of `nfa-eps2nfa` (a python script with the algorithm `NFAεtoNFA` from class and an example input corresponding to the automaton of this exercise).

Solution 2.3

1. We use that $v \in L_n$ iff for every $1 \leq i \leq n$ the i -th and $i + n$ -th letters of v coincide. This is a conjunction of conditions. We construct a universal automaton that has a run on v for each of these conditions, and the run accepts iff the condition holds.

The automaton has a spine of states q_0, q_1, \dots, q_{n-1} , with transitions $q_i \xrightarrow{0,1} q_{i+1}$ for every $0 \leq i \leq n - 2$. At every state q_i the automaton can leave the spine remembering the $(i + 1)$ -th letter by means of transitions $q_i \xrightarrow{0} \mathbf{0}_1$ and $q_i \xrightarrow{1} \mathbf{1}_1$. The automaton then reads the next $n - 1$ letters by transitions $\mathbf{0}_i \xrightarrow{0,1} \mathbf{0}_{i+1}$ and $\mathbf{1}_i \xrightarrow{0,1} \mathbf{1}_{i+1}$ for every $1 \leq i \leq n - 1$, and checks whether the $(i + n)$ -th letter matches the $(i + 1)$ -th letter by transitions $\mathbf{0}_n \xrightarrow{0} q_f$ and $\mathbf{1}_n \xrightarrow{1} q_f$, where q_f is the unique final state.

Below is the automaton for $n = 2$.



2. Let A be an NFA recognizing L_n . Then, for every $ww \in \Sigma^{2n}$, the automaton A has at least one accepting run on ww . Let q_w be the state reached by this run (if there are several accepting runs pick anyone). We claim that for any two different words w, w' of length n the states $q_w, q_{w'}$ are also different. Assume $q_w = q_{w'}$. Then, A has an accepting run on ww' , obtained by concatenating the first half of the accepting run on ww and the second half of the accepting run on ww' . But $ww' \notin L_n$, contradicting the assumption that A recognizes L_n , and the claim is proved. So A has a different state q_w for each word w of length n , and so it has at least 2^n states.

3. It suffices to replace line 6 of `NFAtoDFA` by : **if** $Q' \subseteq F$ **then add** Q' to \mathcal{F} .

Solution 2.4

All statements are false. Since \emptyset and Σ^* are both regular, any of the first two statements would imply that every language is regular, which is certainly not the case. For the third statement, take $L_1 = a^*$ and take for

L_2 any non-regular language over $\{a\}$ (for instance, $L_2 = \{a^{n^2} \mid n \geq 0\}$). Then $L_1L_2 = a^*$, which is regular. For the fourth statement, take $L_1 = \{a^{n^2} \mid n \geq 0\}$ and $L_2 = a^*$.

Solution 2.5

Let $Q = \{q_1, \dots, q_n\}$ be the set of states of the alternating automaton. In an NFA, after reading a word the automaton is in one of the states reached by the runs labelled by w . Imagine these states are $\{q_1, q_2, q_3\}$. Then we say that the automaton is currently at $(q_1 \vee q_2 \vee q_3)$. If the automaton has the universal accepting condition, then intuitively it is simultaneously in all three states, and we write $(q_1 \wedge q_2 \wedge q_3)$.

An alternating automaton can also be at $q_1 \vee (q_2 \wedge q_3)$. For instance, this occurs for the word ab if the initial state q_0 and q_1 are both existential states with transitions (q_0, a, q_1) , (q_0, a, q_4) , and (q_1, b, q_1) , and q_4 is a universal state with transitions (q_4, b, q_2) and (q_4, b, q_3) .

This suggests to define the states of the DFA as the positive boolean formulas with Q as set of variables. However, since there are infinitely many such formulas, we define the states as the equivalence classes of formulas (where, as usual, two formulas are equivalent if they are true for the same valuations of the variables).

The initial state is the (equivalence class of) the formula q_0 . The final states are the formulas that are true when all final states are set to true, and all non-final states to false. Given a formula f , the unique formula f' such that (f, a, f') belongs to the transition relation is defined as follows. For each state q :

- If q is existential and $(q, a, q_1), \dots, (q, a, q_n)$ are the output transitions of q , then replace every occurrence of q in f by $(q_1 \vee \dots \vee q_n)$. If $n = 0$, then replace it by **false**.
- If q is universal and $(q, a, q_1), \dots, (q, a, q_n)$ are the output transitions of q , then replace every occurrence of q in f by $(q_1 \wedge \dots \wedge q_n)$. If $n = 0$, then replace it by **true**.