Universität Zürich
Institut für Informatik

Autumn 2020
Sven Seuken
Dennis Komm

Student Name:
Student Number:

Foundations of Computing II
Assignment 2 – Solutions

Deterministic and Non-deterministic Automata and Regular Languages

---

Distributed: 05.10.2020 – Due Date: 25.10.2020

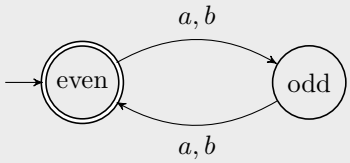
Upload your solutions to the OLAT system.

2.1 Deterministic Finite Automata

For a word w, we denote the number of occurrences of the letter a in w by $|w|_a$; for instance, we have $|1010110|_0 = 3$ and $|xyzxzzy|_x = 2$. Draw DFAs for the following languages.

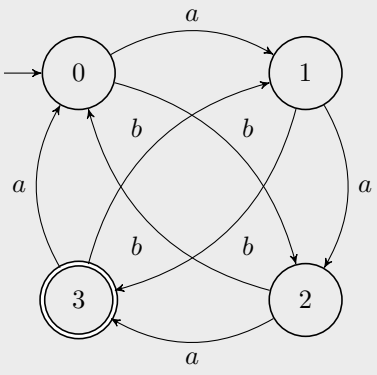
a) $L_1 = \{w \in \{a, b\}^* \mid |w|_a + |w|_b \text{ is even}\}$,

Since words in L_1 only consist of as and bs, the condition "$|w|_a + |w|_b$ is even" simply means that $|w|$ is even.



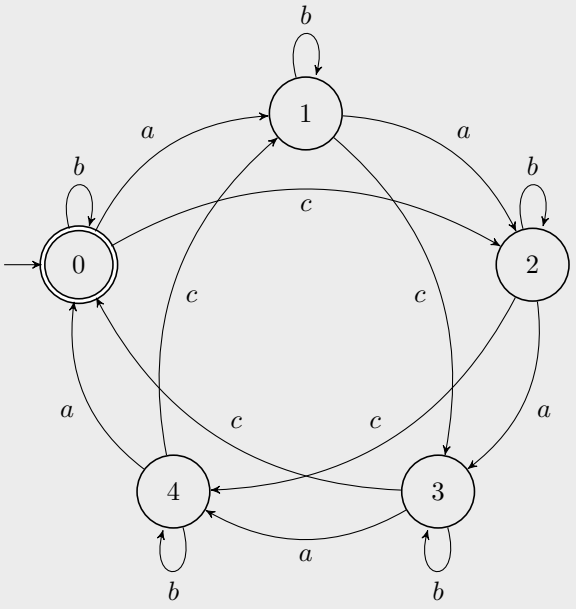
b) $L_2 = \{w \in \{a, b\}^* \mid (|w|_a + 2|w|_b) \bmod 4 = 3\}$,

A state is labeled i if, for the current word x, it holds that $(|x|_a + 2|x|_b) \bmod 4 = i$.



c) $L_3 = \{w \in \{a,b,c\}^* \mid (|w|_a + 2|w|_c) \bmod 5 = 0\}$.

A state is labeled i if, for the current word x, it holds that $(|x|_a + 2|x|_c) \bmod 5 = i$. Note that reading bs does not influence this equality.



2.2 Size of Deterministic Finite Automata

Consider the language

$$L = \{w \in \{a,b\}^* \mid w = xay \text{ with } x \in \{a,b\}^* \text{ and } y \in \{a,b\}^2\},$$

that is, the language that contains all words over the alphabet $\{a,b\}$ which have at least three letters and an a at the third-to-last position. Prove that any DFA for L has to have at least four states in the following way.

Take a set of four words (that are not necessarily in L) and show that no two of them are allowed to end in the same state of any DFA for L. To this end, for any two words w_1 and w_2, supply a *shortest suffix* that implies that one of the two words has to end in an accepting state while the other one must not end in an accepting state.

Consider the four strings aa, ab, ba, and bb. Let us have a look at the first two words aa and ab. For a contradiction, suppose that there is a DFA A for L such that aa and ab end in the same state q of A. If we append the word bb, we obtain the two words $aabb$ and $abbb$; the former has to be accepted by A, the latter must not be accepted. However, since A is in state q after reading either aa or ab, it will also be in the same state q' after reading the two words of length four, since the same two letters bb are appended. Thus, q' needs to be accepting and non-accepting at the same time, which is a contradiction, and it follows that A is in a different state when reading aa than when reading ab. We continue in this fashion with the other words; for ease of presentation,

we arrange the suffixes in a table

	ab	ba	bb
aa	bb	b	b
ab		b	b
ba			bb

It follows that no two of these four words are allowed to end in the same state of A. Since, as a consequence, there needs to be at least one state for each such word, A has to have at least four states.

2.3 Non-Deterministic Finite Automata

a) Construct an NFA for the following languages.

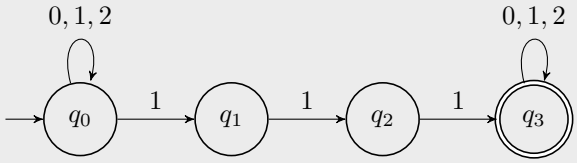
(i) $L_1 = \{w \in \{0,1,2\}^* \mid w \text{ contains the string } 111\}$,

(ii) $L_2 = \{w \in \{0,1,2\}^* \mid w \text{ ends with the string } 111\}$,

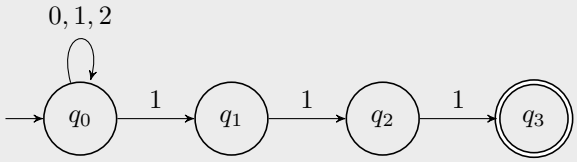
(iii) L_3, which is matched by the regular expression $(1+0)^*1(0+1)11(0+1)(0+1)00$.

Hint: Here, you do not have to use the construction from the lecture.

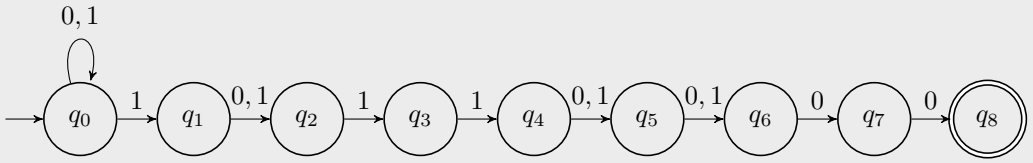
(i) NFA for L_1:



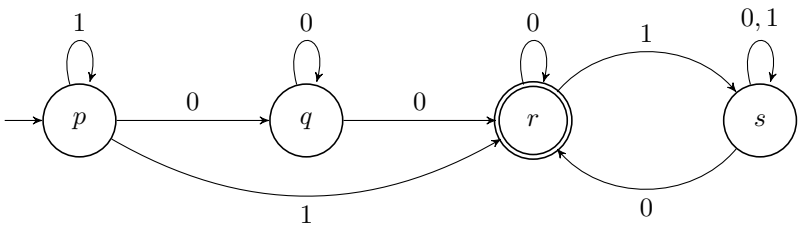
(ii) NFA for L_2:



(iii) NFA for L_3:



b) Consider the following NFA.

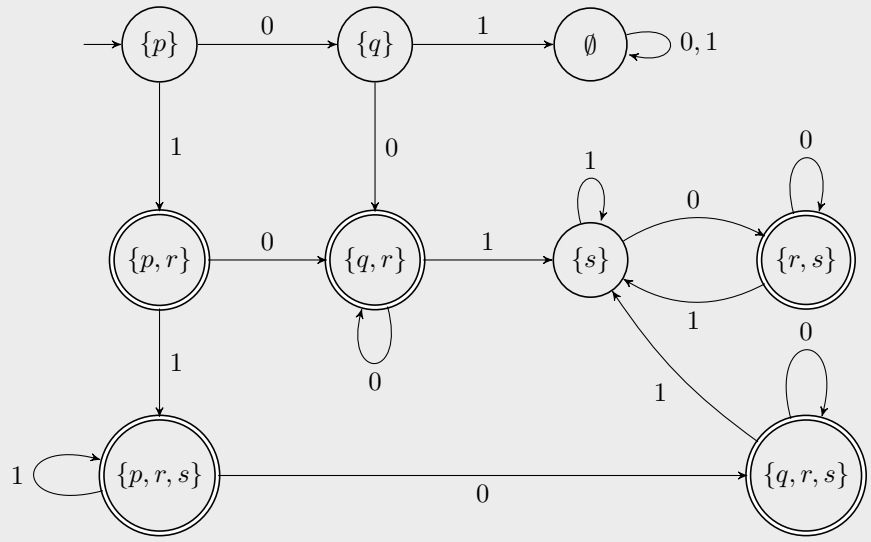


Use the powerset construction to transform this NFA into a DFA.

We obtain the following transition table.

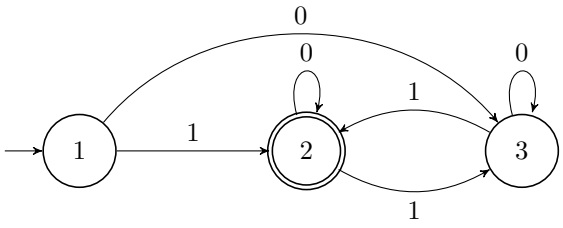
	0	1
$\to \{p\}$	$\{q\}$	$\{p,r\}$
$\{q\}$	$\{q,r\}$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$
$*\{p,r\}$	$\{q,r\}$	$\{p,r,s\}$
$*\{q,r\}$	$\{q,r\}$	$\{s\}$
$*\{p,r,s\}$	$\{q,r,s\}$	$\{p,r,s\}$
$\{s\}$	$\{r,s\}$	$\{s\}$
$*\{q,r,s\}$	$\{q,r,s\}$	$\{s\}$
$*\{r,s\}$	$\{r,s\}$	$\{s\}$

From this, the DFA can be drawn as follows.



2.4 Finite Automata and Regular Expressions

a) Use the method that was introduced in the lecture to transform the following DFA into a regular expression.



Write down all steps and comment on what you are doing.

The method is based on the principle of dynamic programming. It calculates, for each pair of states (i, j), the partial regular expressions $R_{ij}^{(k)}$, which describe the set of all words that lead the automaton from state i to state j and which use only states $\leq k$ as intermediate states. For the initialization, we get the following.

$$
\begin{aligned}
R_{11}^{(0)} &= \varepsilon \\
R_{12}^{(0)} &= 1 \\
R_{13}^{(0)} &= 0 \\
R_{21}^{(0)} &= \emptyset \\
R_{22}^{(0)} &= \varepsilon + 0 \\
R_{23}^{(0)} &= 1 \\
R_{31}^{(0)} &= \emptyset \\
R_{32}^{(0)} &= 1 \\
R_{33}^{(0)} &= \varepsilon + 0
\end{aligned}
$$

Using the formula

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \cdot \left(R_{kk}^{(k-1)}\right)^* \cdot R_{kj}^{(k-1)}$$

for $k = 1$, we can now derive the partial expressions for all ways through the automaton with intermediate state 1.
By this, we get the following table.

	after insertion into the formula	simplified expression
$R_{11}^{(1)}$	$\varepsilon + \varepsilon\varepsilon^*\varepsilon$	ε
$R_{12}^{(1)}$	$1 + \varepsilon\varepsilon^*1$	1
$R_{13}^{(1)}$	$0 + \varepsilon\varepsilon^*0$	0
$R_{21}^{(1)}$	$\emptyset + \emptyset\varepsilon^*\varepsilon$	$\emptyset$
$R_{22}^{(1)}$	$(\varepsilon + 0) + \emptyset\varepsilon^*1$	$\varepsilon + 0$
$R_{23}^{(1)}$	$1 + \emptyset\varepsilon^*0$	1
$R_{31}^{(1)}$	$\emptyset + \emptyset\varepsilon^*\varepsilon$	$\emptyset$
$R_{32}^{(1)}$	$1 + \emptyset\varepsilon^*1$	1
$R_{33}^{(1)}$	$(\varepsilon + 0) + \emptyset\varepsilon^*0$	$\varepsilon + 0$

These expressions can now be reused to get the expressions for the intermediate states 1 and 2. Thereby we obtain the following table.

	after insertion into the formula	simplified expression
$R_{11}^{(2)}$	$\varepsilon + 1(\varepsilon + 0)^*\emptyset$	ε
$R_{12}^{(2)}$	$1 + 1(\varepsilon + 0)^*(\varepsilon + 0)$	10^*
$R_{13}^{(2)}$	$0 + 1(\varepsilon + 0)^*1$	$0 + 10^*1$
$R_{21}^{(2)}$	$\emptyset + (\varepsilon + 0)(\varepsilon + 0)^*\emptyset$	$\emptyset$
$R_{22}^{(2)}$	$(\varepsilon + 0) + (\varepsilon + 0)(\varepsilon + 0)^*(\varepsilon + 0)$	0^*
$R_{23}^{(2)}$	$1 + (\varepsilon + 0)(\varepsilon + 0)^*1$	0^*1
$R_{31}^{(2)}$	$\emptyset + 1(\varepsilon + 0)^*\emptyset$	$\emptyset$
$R_{32}^{(2)}$	$1 + 1(\varepsilon + 0)^*(\varepsilon + 0)$	10^*
$R_{33}^{(2)}$	$(\varepsilon + 0) + 1(\varepsilon + 0)^*1$	$(\varepsilon + 0) + 10^*1$

In the last iteration, in which all states are allowed as intermediate states, we do not have to calculate all expressions. The regular expression equivalent to the DFA

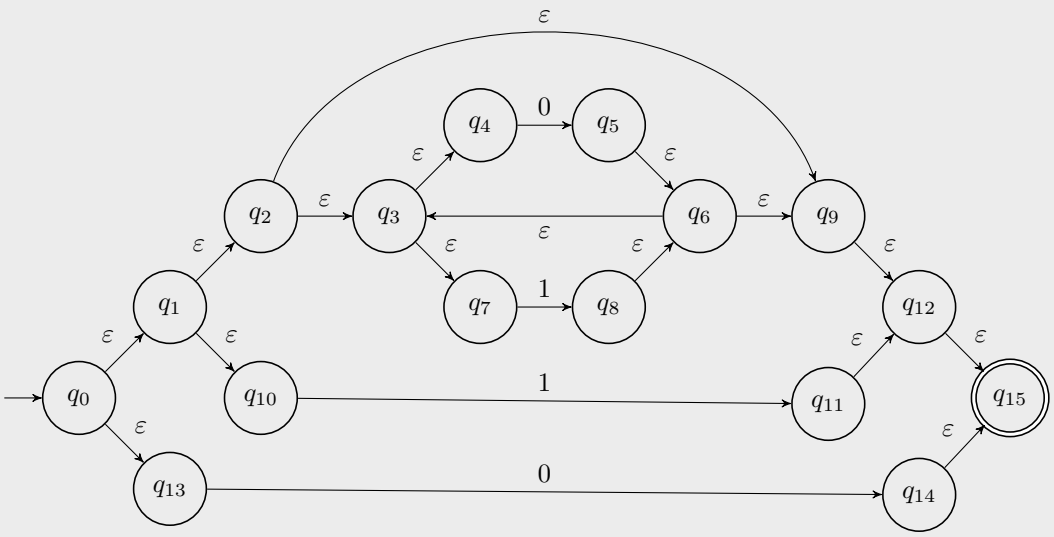
$$R = R_{12}^{(3)}$$

is the only way to reach the accepting state from the starting state. By applying the formula, we get

$$R_{12}^{(3)} = 10^* + (0 + 10^*1)(\varepsilon + 0 + 10^*1)^*10^* = 10^* + (0 + 10^*1)(0 + 10^*1)^*10^* \ .$$

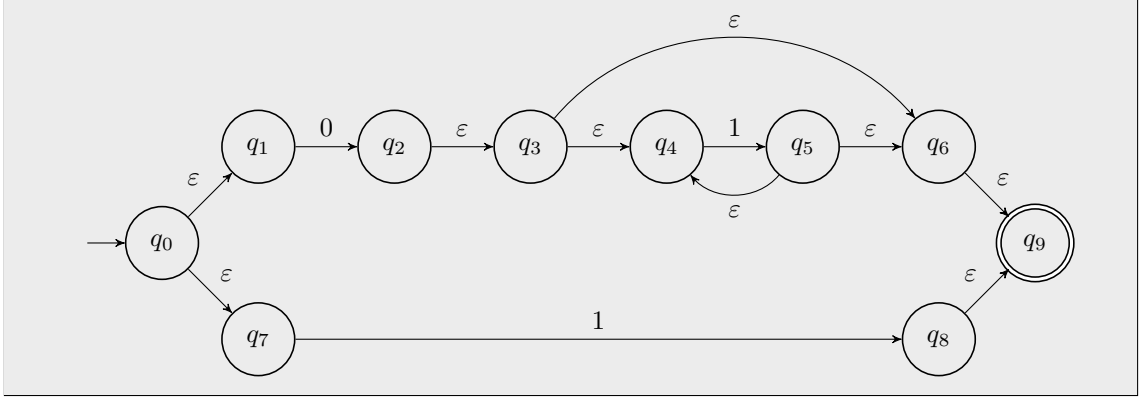
b) Convert the regular expression $1 + (0 + 1)^* + 0$ into an NFA with ε-transitions.

We get the following ε-NFA.



c) Convert the regular expression $01^* + 1$ into an NFA with ε-transitions.

We get the following ε-NFA.



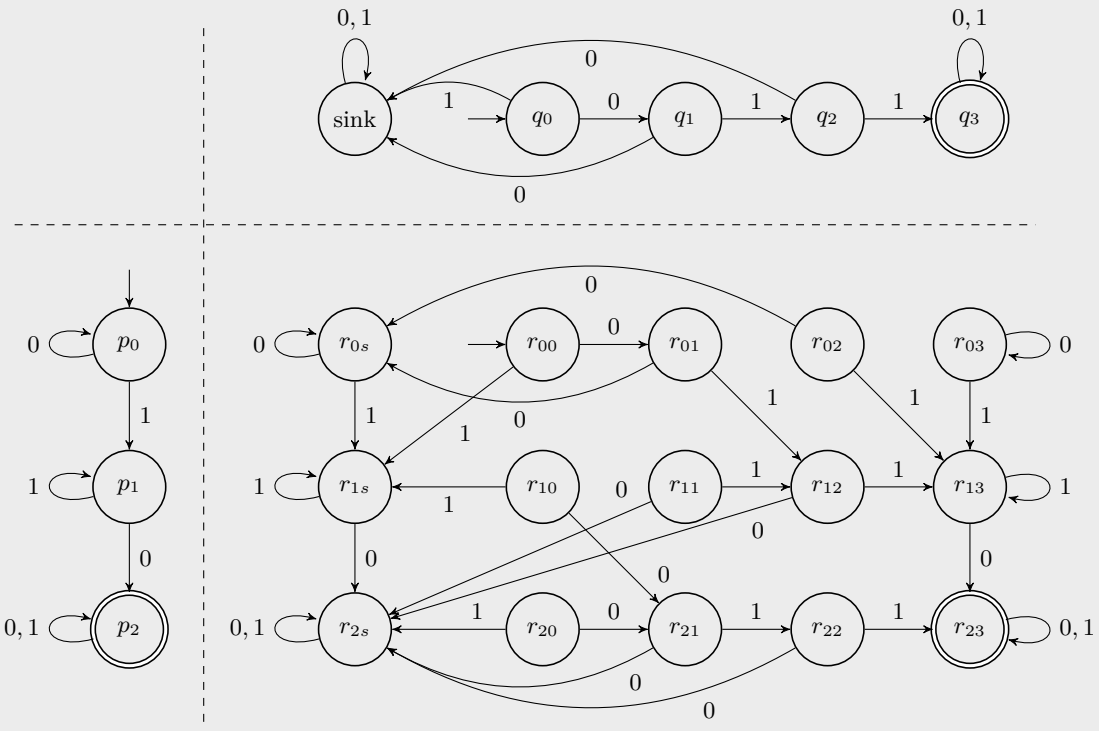
2.5 The Product Automaton

Consider the two languages

$$L_{10} = \{w \in \{0,1\}^* \mid w \text{ contains } 10\} \text{ and}$$
$$L_{011} = \{w \in \{0,1\}^* \mid w \text{ starts with } 011\}.$$

Construct the product automaton for $L_{10} \cap L_{011}$ with the technique described in the lecture.

The construction of the product automaton results in the following DFA for $L_{10} \cap L_{011}$.



2.6 Non-Regularity

a) In the last assignment (more specifically, exercise 1.2(b)iv), we already sketched that there is a problem when designing a DFA for the language $L_{01} = \{0^k 1^k \mid k \in \mathbb{N}\}$. Now use the pumping lemma to prove that this language is indeed not regular.

We prove the claim by contradiction using the pumping lemma. Let n_0 be the constant from the pumping lemma. We pick the word $w = 0^{n_0}1^{n_0}$; clearly, $w \in L_{01}$ and $|w| \geq n_0$. As a consequence, there exist x, y, and z with $w = xyz$ such that the conditions 1, 2, and 3 of the pumping lemma are satisfied. Due to 2, we know that $|xy| \leq n_0$, and therefore y consists only of 0s; due to 2, y consists of at least one 0. Thus, xz, which must be in L_{01} if L_{01} is regular, consists of fewer than n_0 0s, followed by exactly n_0 1s. This word is clearly not in L_{01}, which contradicts the assumption that L_{01} is regular.

b) Again, using the pumping lemma, prove that the language $L_{\text{sq}} = \{0^k \mid k \text{ is a square}\}$ is not regular.

We prove the claim by contradiction using the pumping lemma. Let n_0 be the constant from the pumping lemma. We pick the word $w = 0^{n_0^2}$, that is, n_0^2 0s; clearly, $w \in L_{\text{sq}}$ and $|w| \geq n_0$. Therefore, there exist x, y, and z with $w = xyz$ such that the conditions 1, 2, and 3 of the pumping lemma are satisfied. Due to 1 and 2, we know that y consists of between 1 and n_0 0s. Thus, $xyyz$ has a length between $n_0^2 + 1$ and $n_0^2 + n_0$. Since the next square after n_0^2 is $(n_0 + 1)^2 = n_0^2 + 2n_0 + 1$, we know that the length of $xyyz$ lies strictly between the consecutive squares n_0^2 and $(n_0 + 1)^2$. Thus, the length of $xyyz$ cannot be a square. But if L_{sq} were regular, then $xyyz$ would be in the language, which contradicts the assumption that L_{sq} is regular.