EXAM AUTOMATA THEORY
Monday 19 December 2022, 09.00 - 12.00

This exam consists of eight exercises, where [x pt] indicates how many points can be earned per exercise. A total of 100 points can be earned.
It is important to provide an explanation or motivation when a question asks for it.
A finite automaton in this exam (without further addition), refers to a deterministic finite automaton without Λ-transitions (which is elsewhere called *DFA*).

---

1. [8 pt] Let

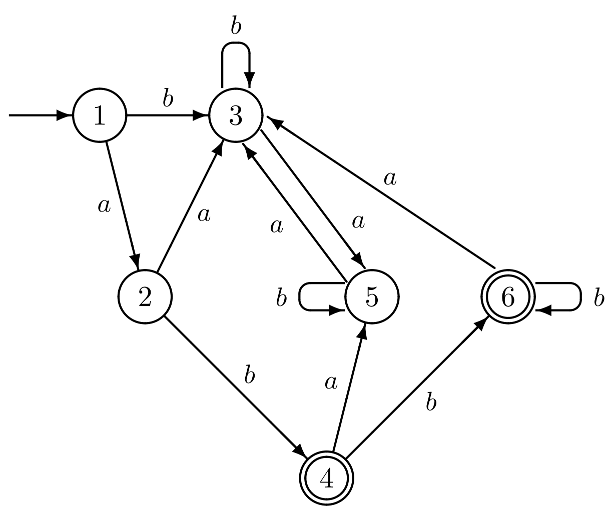
$$L = \{x \in \{a,b\}^* \mid \ x \text{ begins with } aa \text{ or } ab, \text{ and } x \text{ ends with } aa \text{ or } bb\}$$

For example, $aa \in L$, $abb \in L$, but $abbbab \notin L$.

Draw a finite automaton M, such that $L(M) = L$.

---

2. [9 pt] Consider the following finite automaton M_1:



Apply the minimization algorithm on M_1 to find a minimal finite automaton M_2 (a finite automaton with as few states as possible), such that $L(M_2) = L(M_1)$. Which means:

(a) Loop repeatedly, column by column (columns from left to right), through the triangle below, and fill in the numbers indicating in which iteration of the algorithm it is established that states i and j cannot be merged.

$i = 2$	·				
3	·	·			
4	·	·	·		
5	·	·	·	·	
6	·	·	·	·	·
	$j =$				
	1	2	3	4	5

Give the resulting table as your answer.

(b) Draw the resulting finite automaton M_2 where states are merged following your answer to (a). Remark: the names of the states of M_1 must be recognizable in the names of the states of M_2.

---

3. [17 pt]

(a) Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ be an arbitrary finite automaton, and let $L_1 = L(M_1)$.

How can you convert M_1 into a finite automaton $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$, such that $L(M_2) = L_1'$ (the complement of L_1)?

Give as your answer the four components Q_2, q_2, A_2 and δ_2 of M_2, expressed in terms of the components of M_1.

If you do not know the answer to this question, you can 'buy' it from the lecturer. Perhaps you can then solve (b).

(b) The construction from part (a) can not be transferred to an arbitrary **pushdown** automaton. Give two reasons why you could not convert an arbitrary pushdown automaton M_1 into a pushdown automaton M_2 such that $L(M_2) = L(M_1)'$ in the same way.

Illustrate each reason with a concrete example of a pushdown automaton M_1 with at most four states, in which the construction yields and automaton M_2 with $L(M_2) \neq L(M_1)'$. In addition, give for each example pushdown automaton a concrete string x, such that x is accepted by both M_1 and M_2, or is not accepted by both automata.

(c) Indeed, the class of context-free languages is not closed under complement. Give an example of a context-free language L_1 whose complement L_1' is not context-free.

If you mention a language L_1, for which it was explained in the lectures that L_1 is context-free and L_1' is not, then you do not need to prove this further. Otherwise, you do have to prove this.

---

4. [12 pt] This exercise concerns regular languages over the alphabet $\{a, b, c\}$. Let $L_1 = \{a^i b^j c^k \mid i, j, k \geq 1\}$.

(a) Give a regular expression for the language L_1.

(b) Give a regular expression for the language L_1' (the complement of L_1).

If you cannot come up with a suitable regular expression, you can earn part of the points for this part with a regular expression for the language L_0' (the complement of L_0), where $L_0 = \{a^i b^j c^k \mid i, j, k \geq 0\}$. In that case, explicitly state that you opt for L_0'.

Explain why your expression precisely describes the chosen language.

---

5. [18 pt] In homework 3, we asked for a context-free grammar for the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i + k + 2 < j\}$$

The first six elements of L in canonical order are bbb, $bbbb$, $abbbb$, $bbbbb$, $bbbbc$, $abbbbb$.

Consider the following three context-free grammars G_1, G_2 and G_3:

(a) G_1 has start variable S and the following productions:

$$S \to aSb \mid bbbC \qquad C \to bCc \mid bC \mid \Lambda$$

(b) G_2 has start variable S and the following productions:

$$S \to AbbB \mid AbbBC \mid bbBC \qquad A \to aAb \mid ab \qquad B \to bB \mid b \qquad C \to bCc \mid bc$$

(c) G_3 has start variable S and the following productions:

$$S \to AbbbC \qquad A \to aAb \mid \Lambda \qquad C \to bCc \mid B \qquad B \to bB \mid \Lambda$$

For each of these three grammars G_i, answer the following two questions:

(i) Is $L \subseteq L(G_i)$? If not, give a string x that is in L, but not in $L(G_i)$. If yes, then you don't have to explain that.

(ii) Is $L(G_i) \subseteq L$? If not, give a string x that is in $L(G_i)$, but not in L. If yes, then you don't have to explain that.

---

6. [12 pt] Let G_1 be the context-free grammar with start variable S and the following productions:

$$S \to Sa \mid bb \mid AB \qquad A \to aAb \mid BBa \qquad B \to SB \mid a \mid \Lambda$$

In this exercise, we will perform the first steps of an algorithm to convert G_1 in Chomsky normal form.

(a) Determine step by step (hence via $N_0, N_1, N_2, \ldots$) the *nullable* variable(s) in G_1.

(b) Give the context-free grammar G_2 that results by eliminating Λ-productions from G_1.

(c) Give for each variable X in G_2 the set of X-*derivable* variables.

(d) Give the context-free grammar G_3 that results by eliminating unit productions from G_2.

---

7. [11 pt] Consider again

$$L = \{a^i b^j c^k \quad | \quad i, j, k \geq 0 \text{ and } i + k + 2 < j\}$$

Draw a pushdown automaton M, such that $L(M) = L$.

This pushdown automaton must be based directly on the properties of the language. It should, therefore, not be the result of a standard construction for, for example, converting a context-free grammar into a pushdown automaton.

Try to ensure that M is deterministic and does not contain any Λ-transitions. If you do not succeed in this, you can still earn most of the points.

Also explain how M uses its states and stack to accept precisely the right language.

---

8. [13 pt] The pumping lemma for context-free languages is as follows:

> Suppose L is a context-free language.
> Then there exists an integer $n \geq 2$, such that
> for every $u \in L$ with $|u| \geq n$, u can be written as $u = vwxyz$ for some strings v, w, x, y and z such that
> 1. $|wy| \geq 1$ (i.e., $wy \neq \Lambda$).
> 2. $|wxy| \leq n$.
> 3. For every $m \geq 0$ the string $vw^m xy^m z$ also belongs to L.

Now let

$$L_1 = \{x \in \{a, b, c\}^* \quad | \quad n_a(x) = n_c(x) \text{ and } n_a(x) + n_c(x) \leq n_b(x) + 1\}$$

For example, $a^{10}c^{10}b^{19} \in L_1$. Now, let n be the number from the pumping lemma for this language. You can assume that $n \geq 4$.

For each of the following four strings u_1, u_2, u_3, u_4, indicate whether it is suitable for establishing a contradiction with the pumping lemma. Furthermore, for each of the strings u_i that is **not** suitable, indicate why not, for example, via a concrete decomposition $vwxyz$ of u_i that does satisfy the pumping lemma. If u_i is suitable for contradicting the pumping lemma, then you don't have to explain that.

$$u_1 = a^n b^{2n-2} c^n$$
$$u_2 = a^n b^{2n-1} c^n$$
$$u_3 = a^n b^{3n} c^n$$
$$u_4 = (abbc)^n abc$$

`end of exam`