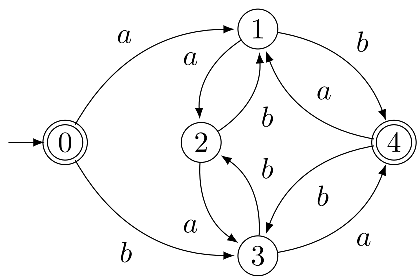
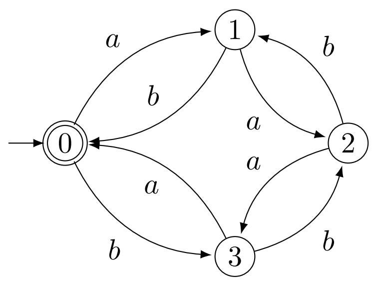
EXAM SOLUTION AUTOMATA THEORY
Thursday 21 December 2023, 09:00 - 12:00

1. [8 pt] An automaton with 5 states:



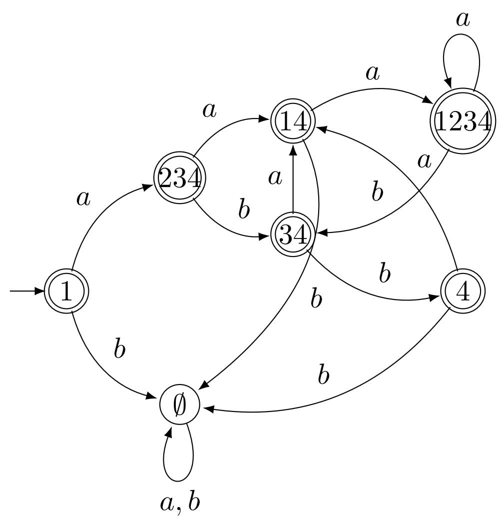
An automaton with 4 states (min nr. of states):



How-to: We consider an a having a weight of 1 while a b having a weight of 3. Summing up the two and dividing the resulting number by 4, if we obtain a remainder of 0, the string belongs to the language, i.e., the state is accepting. In the second automaton, the name of a state represents the remainder when the number is divided by 4. We may start the construction at state 0, where no symbols are read so far, and consider what happens when we read an a, respectively a b.

2. [9 pt]

state q	$\delta(q, a)$	$\delta(q, b)$
1	234	$\emptyset$
234	14	34
14	1234	$\emptyset$
34	14	4
1234	1234	34
4	14	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$



How-to: Let us start at the same initial state as the given automaton, i.e., state $\{1\}$. Where could we go via an a transition? To states 2, 3 or 4. If we take the union of these, we obtain state $\{2, 3, 4\}$. Similarly, where could we go via a b transition? To no state, i.e., $\emptyset$. Now consider state $\{2, 3, 4\}$. For each of the states in this set, where could we go via an a transition? Again take the union of the resulting states. ... Mark as accepting all states which contain a state that is accepting in the given automaton.

---

3. [20 pt]

(a) x_1 is not suitable, e.g., $u = \lambda, v = a^2, w = a^{n-1}b^n$
x_2 is suitable
x_3 is not suitable, it does not necessarily have length $\geq n$
x_4 is not suitable, e.g., $u = \lambda, v = b, w = b^{2n-1}$

(b) $\{b^k \mid k \geq 1\}$

(c) $\{a^i b^i \mid i \geq 1\}$.

How-to:

(a) For contradicting the pumping lemma, we need to prove the negation:

If
$\forall$ for every $n \geq 1$
$\exists$ there exists $x \in L$
with $|x| \geq n$
such that
$\forall$ for every decomposition $x = uvw$
with (1) $|uv| \leq n$,
and (2) $|v| \geq 1$
$\exists$ (3) there exists $m \geq 0$,
such that $uv^m w \notin L$
then L is not a regular language.

A suitable string, x_i, is one which satisfies the above.

(b) Which strings $z \in \Sigma^*$ could we concatenate with a^2b^2 such that $a^2b^2 \cdot z$ belongs to L_1?

(c) Which strings $x \in \Sigma^*$ have the same L_1/x as a^2b^2, i.e., $L_1/x = L_1/a^2b^2$?

---

4. [9 pt]

(a) abb

(b) a

(c) b

(d) λ

How-to: One way to approach this exercise is by considering strings of increasing length: $\lambda, a, b, aa, ab, ba, bb, aaa, \ldots$, and for each of them check whether it satisfies r and/or s.

---

5. [14 pt]

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

$$\begin{array}{ll} S \to S_1 \mid S_2 & S_1 \text{ for } i > j,\ S_2 \text{ for } i < j \\ S_1 \to aS_1b \mid T_1 & T_1 \text{ for extra } a\text{'s} \\ T_1 \to aT_1 \mid a & \\ S_2 \to aS_2b \mid T_2 & T_2 \text{ for extra } b\text{'s} \\ T_2 \to T_2b \mid b & \end{array}$$

This context-free grammar is unambiguous.

(b)

$$S \Rightarrow S_2 \Rightarrow aS_2b \Rightarrow aT_2b \Rightarrow aT_2bb \Rightarrow abbb$$

---

6. [11 pt]

Given that $V = \{S\}$ and $\Sigma = \{a, b\}$, the only possible productions in P are $S \to SS$ and $S \to a \mid b$.

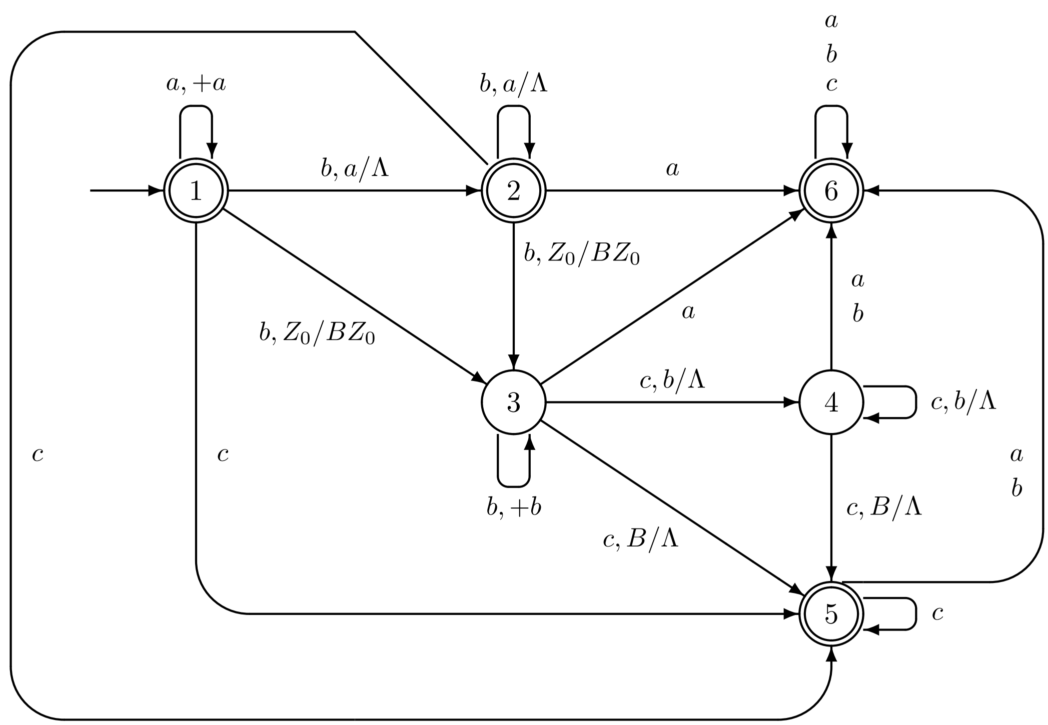
- If G does not have any of the productions $S \to a$ and $S \to b$, then G cannot derive any terminal string: $L(G) = \emptyset$.
- If G contains at least one of the productions $S \to a$ and $S \to b$, then let $\Sigma' = \{\sigma \in \Sigma \mid S \to \sigma \in P\}$ (the terminals that can actually be generated).
  - If G does not contain the production $S \to SS$, then G can only directly generate the string(s) consisting of a single terminal from Σ': $L(G) = \Sigma'$.
  - If G does contain the production $S \to SS$, then G can generate all non-empty strings over Σ': $L(G) = (\Sigma')^* \setminus \{\Lambda\} = \Sigma' \cdot (\Sigma')^*$.

---

7. [18 pt] L' consists of strings in $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i + k \geq j\}$ and strings with letters a, b, c 'out of order'.

   (a) The first four elements in the canonical (shortlex) order of L' are: Λ, a, c, aa.

   (b) M could be as follows:



M uses state 6 to accept any string where letters are 'out of order', i.e., a after b (transitions from states 2 and 3), or a or b after c (transitions from states 4 and 5). The other states are meant to accept the strings $a^i b^j c^k$ with $i + k \geq j$.

First, in state 1, we count the a's that we read with symbols a on the stack. At that point, $j = k = 0$, so the string is still in L'.

Upon reading b^j, we remove an a from stack for every b read. Initially, as long as $j \leq i$, in state 2, the string is still in L'.

When we read a b with Z_0 on top of the stack, we have reached $j > i$, and since k is still 0, $i + k < j$. We move to non-accepting state 3, pushing B onto the stack, to indicate the first extra b read. We count subsequent extra b's in the input with b's on the stack.

When we continue with c's, we move to a new non-accepting state, state 4, and remove a b from the stack for every c read. As long as B is on the stack (possibly with b's on top of it), we have $i + k < j$.

When we read c with B on top of the stack, we reach equality: $i+k = j$. We remove B from stack, and move to accepting state 5, where we may continue reading c's as long as we wish.

We do not necessarily visit all five states $1 - 5$. There are shortcuts for cases where we read b or c earlier than described above.

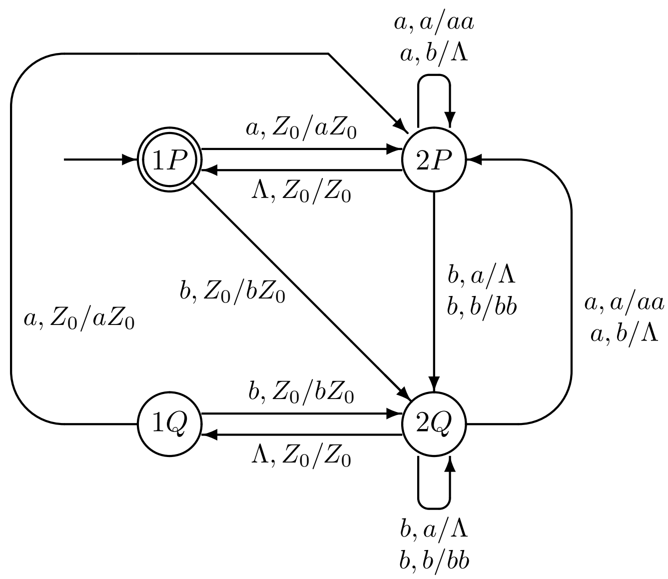
M is indeed deterministic and does not have any Λ-transitions.

8. [11 pt]

(a)

$$
\begin{aligned}
Q &= \{1, 2\} \\
\Sigma &= \{a, b\} \\
\Gamma &= \{a, b, Z_0\} \\
q_0 &= 1 \\
A &= \{1\}
\end{aligned}
$$

(b) We combine the states and the transitions of M_1 and M_2, yielding:



`end of exam solution`