EXAM AUTOMATA THEORY
Thursday 19 December 2024, 09:00 - 12:00

This exam consists of nine 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] Consider the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } j \text{ is even}\}$$

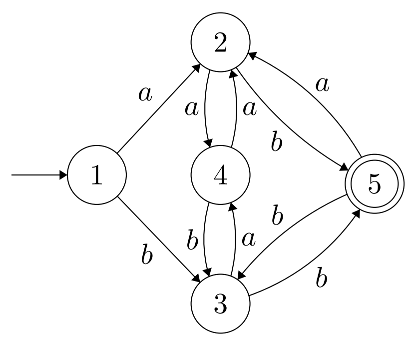
Construct a deterministic finite automaton that accepts L.

---

2. [14 pt] Consider the language

$$L = \{x \in \{a, b\}^* \mid x \text{ ends in } b \text{ and } |x| \text{ is even}\}$$

The following deterministic finite automaton M accepts L:



(a) For each of the following strings x_i, determine the state q_i such that $\delta^*(1, x_i) = q_i$. You do not need to explain your answers. Note that all five states occur as answer once.

(i) $x_1 = \Lambda$
(ii) $x_2 = a$
(iii) $x_3 = b$
(iv) $x_4 = aa$
(v) $x_5 = ab$

(b) For each of the strings in (a), determine the equivalence class $[x_i]$, i.e., the set of strings indistinguishable from x_i with respect to L. Note that your answer should be the same for (at least) two of the strings.

(c) Does M contain a minimal amount of states? Explain your answer.

---

3. [14 pt] Consider the language L described by the regular expression

$$(ab + a)^*(\Lambda + bb)$$

   (a) For each of the following strings, determine whether or not they are an element of L. You do not need to explain your answers.
      (i) Λ
      (ii) bbb
      (iii) $aaba$
      (iv) $ababb$

   (b) Construct a non-deterministic finite automaton accepting L. The regular expression should be recognizable in the automaton.

---

4. [10 pt]

   Consider the language

$$L = \{x \in \{a, b\}^* \mid n_a(x) \geq n_b(x) \geq n_a(x) - 2\}$$

   Hence, L contains the strings in which there are 0, 1 or 2 fewer b's than a's. For example, $aaba \in L$ and $bbabaa \in L$, but $aabaa \notin L$ because there are too few b's, and $bba \notin L$ because there are too many b's.

   Prove that the language L cannot be accepted by a finite automaton by using the pumping lemma for regular languages.

---

5. [7 pt] Each regular language is also context-free, i.e., it can be generated by a context-free grammar. However, not all context-free languages are also regular. For each of the following context-free grammars G, with start variable S, indicate whether or not $L(G)$ is regular. You do not need to explain your answers.

   (a) G has productions

$$S \to abA \mid bB \mid aba \qquad A \to b \mid aB \mid bA \qquad B \to aB \mid bA$$

   (b) G has productions

$$S \to aS \mid Sb \mid a \mid b$$

   (c) G has productions

$$S \to aA \mid b \qquad A \to Sb \mid a$$

   (d) G has productions

$$S \to aA \mid bB \qquad A \to aB \mid bA \mid bS \qquad B \to bS \mid \Lambda$$

---

6. [12 pt] Let
$$L = \{a^i b^j a^k \mid \ i, j, k \geq 0 \text{ and } j < i + k\}$$
   (a) Give the first six elements in the canonical (shortlex) order of L.

   (b) Give a context-free grammar G, such that $L(G) = L$. Try to ensure that G is unambiguous. If you do not succeed in this, then you can still earn most of the points.

   If your context-free grammar is ambiguous, then give two different derivation trees for a string $x \in L$.

---

7. [13 pt] A context-free grammar $G = (V, \Sigma, S, P)$ is said to be in *Chomsky normal form*, if each production in G is of one the following two forms:
$$\begin{array}{lll} A & \rightarrow & BC \qquad\qquad \text{with } A, B, C \in V \\ A & \rightarrow & \sigma \qquad\qquad\; \text{with } A \in V \text{ and } \sigma \in \Sigma \end{array}$$
   Now let G_1 be the context-free grammar with start variable S and the following productions:
$$S \rightarrow XBBb \mid XB \qquad X \rightarrow aX \mid ab \qquad B \rightarrow bB \mid \Lambda$$
   In this question, we will convert this grammar into Chomsky normal form, using the constructions discussed in our lectures and exercise classes. You do not need to explain your answers.

   (a) Give the set of nullable variables in G_1.

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

   (c) Give the context-free grammar G_3 resulting from G_2 by eliminating unit productions.

   (d) Give the context-free grammar G_4 resulting from G_3 by introducing for every terminal symbol σ a variable X_σ (with a corresponding production), and substituting this variable for occurrences of σ where necessary in the righthand side of productions.

   (e) Give the context-free grammar G_5 resulting from G_4 by splitting the righthand side of productions which are too long.

---

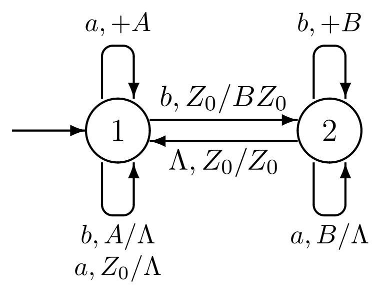
8. [7 pt] Let $M_1 = (Q_1, \Sigma, \Gamma_1, q_1, Z_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, q_2, Z_2, A_2, \delta_2)$ be two pushdown automata, such that $L(M_1) = L_1$ and $L(M_2) = L_2$ for two languages L_1 and L_2 over the same alphabet Σ.

   Describe a general procedure for constructing a pushdown automaton M, such that $L(M) = L_1 \cdot L_2$ (the concatenation of the two languages).

   Your description may consist of words, formulas and/or pictures. Just make sure that it is clear and complete. You may assume that the sets of states Q_1 and Q_2 do not overlap, and that also the stack alphabets Γ_1 and Γ_2 do not overlap. You do not need to prove that the construction is correct.

P.T.O

9. [15 pt] Consider the following pushdown automaton M_1 (without indication of accepting states):



(a) List all occurrences of non-determinism (if any) in M_1. In particular, for each occurrence, mention the state, the input(s) and the stack symbol involved.

(b) A string x is accepted by a pushdown automaton M *by empty stack*, if there exists a computation in M for input x leading to a (completely) empty stack after reading x entirely. In formal terms: if $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, then $(q_0, x, Z_0) \vdash^* (q, \Lambda, \Lambda)$ for some $q \in Q$.

The empty-stack language $L_e(M)$ of a pushdown automaton M is the set of all strings that are accepted by M by empty stack.

What is the empty-stack language $L_e(M_1)$ of the concrete pushdown automaton M_1 above?

Explain how M_1 uses its states and stack (symbols) to accept precisely all strings in this language.

```
end of exam
```