

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Automata and Formal Languages

Exam: IN2041 / Retake

Date: Monday 30th March, 2026

Examiner: Prof. Dr. Javier Esparza

Time: 11:00 – 13:00

Working instructions

- This exam consists of **12 pages** with a total of **7 problems**.
Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 75 credits, including 5 bonus credits.
- To pass the exam, 35 credits are *sufficient*.
- Detaching pages from the exam is prohibited.
- Allowed resources:
 - one **analog dictionary** English ↔ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from _____ to _____ / Early submission at _____

Problem 1 Quiz (15 credits)

For each of these statements, decide whether it is true or false. If it is true, give a proof; if it is false, give a counterexample. Otherwise no points will be awarded! We use $\Sigma := \{a, b\}$ as alphabet in this exercise.

0	<input type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>

a)* Let L_1, L_2 be regular languages such that $L_1 \cup L_2$ has a (not necessarily minimal) DFA with 12 states. If the minimal DFA for L_1 has 4 states, then the minimal DFA for L_2 has at most 3 states.

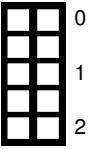
0	<input type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>

b)* Fix a number $n \in \mathbb{N}$. Every language L such that $\bar{L} \subseteq \{w \in \Sigma^n \mid |w|_a \neq |w|_b\}$ is regular.

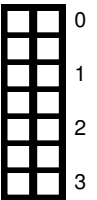
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

c)* Let $L \subseteq \{a, b\}^\omega$ be a nonempty ω -language such that $L = \{a\}L$. True or false: $L = \{a^\omega\}$.

d)* Let R_1, R_2 be two nondeterministic Rabin automata with n_1 and n_2 states, respectively. True or false: There exists a nondeterministic Rabin automaton for $L_\omega(R_1) \cup L_\omega(R_2)$ with $n_1 + n_2$ states.

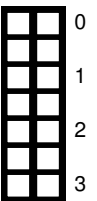


e)* Let q, r be two reachable states of an NBA A such that r is reachable from q , but q is not reachable from r . Consider a run of *NestedDFS* on A . True or false: $d[r] < f[q]$, i.e. the algorithm discovers r before it backtracks from q . If the answer is “true”, prove the statement using the parenthesis theorem given below; if the answer is “false”, give a counterexample.



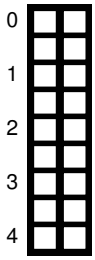
Parenthesis theorem. Let $I(q)$ denote the interval $[d[q], f[q]]$, and let $I(q) \prec I(r)$ denote that $f[q] < d[r]$ holds. In a DFS-tree, for any two states q and r , one of the following four conditions holds: (1) $I(q) \subseteq I(r)$ and q is a descendant of r ; (2) $I(r) \subseteq I(q)$ and r is a descendant of q ; (3) $I(q) \prec I(r)$ and neither is a descendant of the other; (4) $I(r) \prec I(q)$ and neither is a descendant of the other.

f)* Let $AP = \{p, q, r\}$ be a set of atomic propositions and let $\sigma \in (2^{AP})^\omega$ be a computation. True or false: If $\sigma \models \mathbf{G}(p \mathbf{U} q) \wedge \mathbf{G}(q \mathbf{U} r)$, then $\sigma \models \mathbf{G}(p \mathbf{U} r)$.



Problem 2 Regular languages (8 credits)

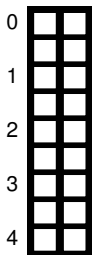
Let Σ be an alphabet.



a)* Give a procedure which takes two NFAs A_1, A_2 over the same alphabet Σ as input and constructs an NFA A such that

$$L(A) = \{u \mid \exists v \in \Sigma^* : uv \in L(A_1) \wedge uv \in L(A_2)\}.$$

You do not need to prove that your procedure is correct.



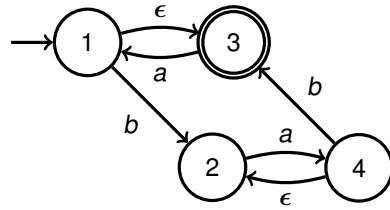
b)* Give a procedure which takes two NFAs A_1, A_2 over the same alphabet Σ as input and constructs an NFA A such that

$$L(A) = \{u \mid \exists v_1, v_2 \in \Sigma^* : uv_1 \in L(A_1) \wedge uv_2 \in L(A_2)\}.$$

You do not need to prove that your procedure is correct.

Problem 3 NFA ϵ to Regular Expression (6 credits)

Consider the NFA ϵ N on the right. Using the algorithm from the lecture, compute a regular expression r with $L(r) = L(N)$. Give the automaton after **every** transformation and **do not** simplify the regular expressions. (Exception: You may simplify $\epsilon\alpha$ or $\alpha\epsilon$ to α .) Remove the states **in ascending order**.



	0
	1
	2
	3
	4
	5
	6

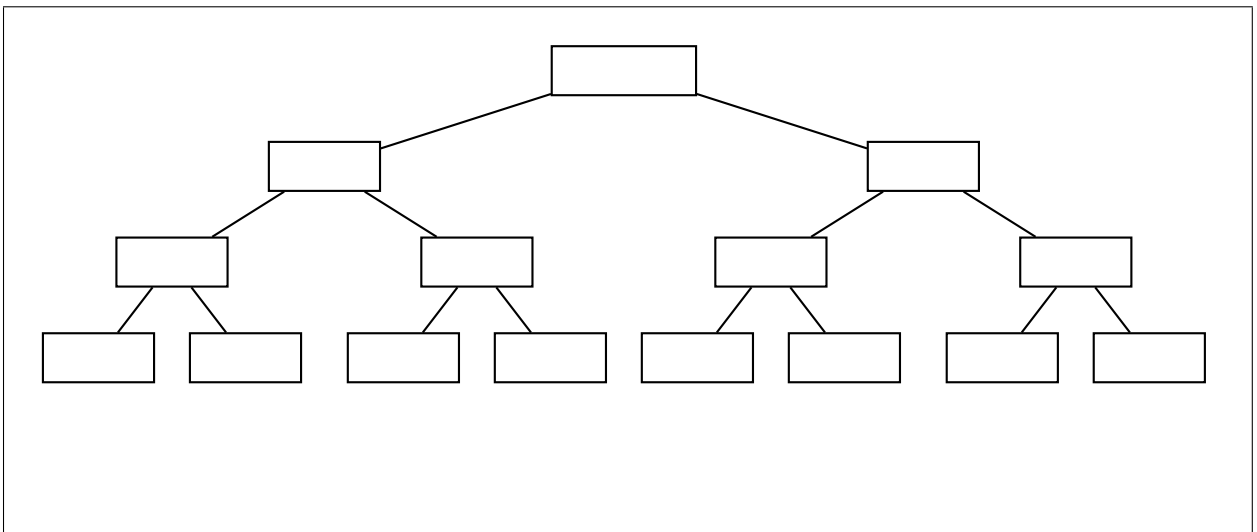
Problem 4 Fixed-length languages (8 credits)

0		
1		
2		
3		
4		

a)* Let $\Sigma = \{a, b\}$. Construct the multi-DFA (i.e., the fragment of the fixed-length master automaton) for the set of languages $\{L_1, L_2\}$, where $L_1 = L(aaa + abb + b\Sigma\Sigma)$ and $L_2 = L(baa + bb\Sigma + a\Sigma\Sigma)$.

0		
1		
2		
3		
4		

b) Let q_1, q_2 be the states of the fixed-length master automaton for L_1, L_2 in your solution to a). Compute the state q_3 for the language $L_3 := L_1 \cap L_2$ by executing the recursive algorithm *inter*(q_1, q_2) from the lecture. For that, fill the tree of recursive calls to *inter* in the box below. Recall that *inter* uses memoization (that is, it stores the results of previous calls to avoid recomputing them). You can use optimized versions of *inter*, but then you must explain the optimizations.



Problem 5 Transducers (8 credits)

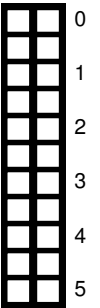
a)* **Bonus points.** Let $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ be a DFA with one single accepting state. Prove that there exists a transducer recognizing the language

$$P_A = \{(w, u) \in \Sigma^* \times \Sigma^* : |w| = |u| \text{ and } wu \in L(A)\}$$

Hint: $|w| = |u|$ and $wu \in L(A)$ holds iff there exists a state $q \in Q$ such that

$$\begin{array}{ccccccc} q_0 & \xrightarrow{w_1} & q_1 & \cdots & q_{n-1} & \xrightarrow{w_n} & q \\ q & \xrightarrow{u_1} & q'_1 & \cdots & q'_{n-1} & \xrightarrow{u_n} & q_f \end{array}$$

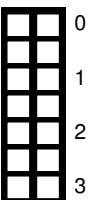
where $w = w_1 \cdots w_n$ and $u = u_1 \cdots u_n$.



b)* Assume the transducer of a) has been constructed. Show that the language of “first halves” of the words of a regular language L , i.e. the language

$$H_L = \{w \in \Sigma^* \mid \exists u \in \Sigma^* : |w| = |u| \text{ and } wu \in L\}$$

is regular.

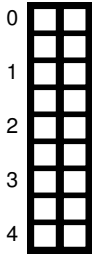


Problem 6 Alternating letters (11 credits)

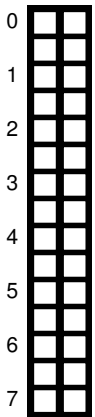
Let L_1 and L_2 be ω -languages over an alphabet Σ . We define

$$L_1 \sim L_2 := \{a_1 b_2 a_3 b_4 a_5 b_6 \dots \mid a_1 a_2 a_3 \dots \in L_1, b_1 b_2 b_3 \dots \in L_2\}.$$

For example, $\{a^\omega\} \sim \{b^\omega\} = \{(ab)^\omega\}$ and $\{(ab)^\omega\} \sim \{(ba)^\omega\} = \{a^\omega\}$.



a)* Let $\Sigma = \{a, b\}$, let L_1 be the language of all words with infinitely many a 's, and let L_2 be the language of all words with infinitely many b 's. Give an ω -regular expression r for $L_1 \sim L_2$, and give a proof that your ω -regular expression is correct.

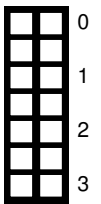


b)* Describe an algorithm which takes two NBAs $N_1 = (Q_1, \Sigma, \delta_1, Q_{01}, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, Q_{02}, F_2)$ as input and outputs an ω -automaton N for $L_\omega(N_1) \sim L_\omega(N_2)$. Give a precise definition of the states, transitions, initial states, and acceptance condition of N . You **may** also give an intuitive explanation of your construction. *Note:* N does not necessarily have to be an NBA.

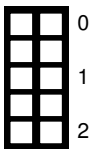
Problem 7 Construction (19 credits)

Answer the following questions. No justification is needed, but a justification can help you to get points if you make a mistake.

a)* Construct the minimal DFA recognizing the language of all words over the alphabet $\Sigma = \{a, b\}$ that do *not* contain *aaa*, i.e., that do *not* belong to $L(\Sigma^*aaa\Sigma^*)$.

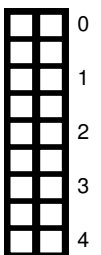


b)* Give a regular expression for the language of the previous question. You may (but don't have to) use the algorithm to convert DFA into regular expressions.

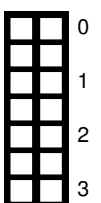


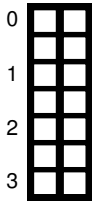
c)* Give formulas of first-order logic on words for the following languages over the alphabet $\{a, b, c\}$. You may use the macros $first(x)$, $last(x)$ and $x = y + 1$, $x = y + 2$, ... without definition. If you use other macros, you must define them first.

1. $L((a + b)^*a(b + c)^*)$
2. $L((a^*b^*c^*)^*c)$
3. $L((ab + c)^*)$

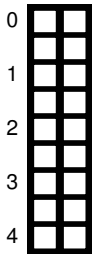


d)* Let $AP = \{p, q\}$. Give an ω -regular expression for the set of all computations over $\Sigma = 2^{AP}$ satisfying $(Gp) \mathbf{U} (Gq)$.





e)* Let $L \subseteq \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}^*$ be the language encoding the solutions of the Presburger formula $y = x + 3$ in least significant bit encoding. Give a Presburger formula for the residual of L with respect to $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.



f)* Give a *deterministic* Rabin automaton for the language of all ω -words over the alphabet $\{a, b, c\}$ that contain infinitely many occurrences of ab or finitely many occurrences of c .

Additional space for solutions—clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

