

**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 30<sup>th</sup> 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.

False. Let  $L_1$  be any language where the minimal DFA has 4 states, e.g.  $L_1 = \{w \in \Sigma^* \mid |w| = 0 \pmod{4}\}$ , and  $L_2 = \overline{L_1}$ . Then the minimal DFA for  $\overline{L_1}$  has 4 states, and there exists a DFA for  $L_1 \cup L_2 = \Sigma^*$  with 1 state, so there also exists one with 12.

Alternatively, one can set  $L_2 = L_1$ .

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  $\overline{L} \subseteq \{w \in \Sigma^n \mid |w|_a \neq |w|_b\}$  is regular.

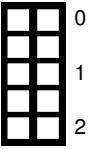
True.  $\overline{L}$  is finite, so  $\overline{L}$  is regular, so  $L$  is regular.

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  $\omega$ -language such that  $L = \{a\}L$ . True or false:  $L = \{a^\omega\}$ .

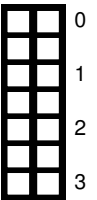
True. Let  $w \in L$  be the word in  $L$  where a  $b$  appears in the earliest position. We then have  $w = a^i b w'$  for some  $i \in \mathbb{N}$  and  $w' \in \{a, b\}^\omega$ . Since  $L = \{a\}L$ , we have  $w = a^i b w' \in \{a\}L$ , so  $a^{i-1} b w' \in L$ . This contradicts our choice of  $w$ .

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.



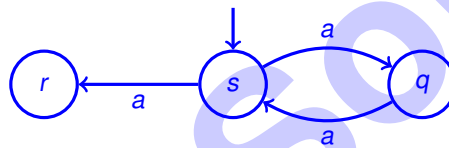
True. One can put the automata side by side and take the union of the Rabin pairs.

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.



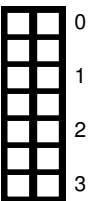
*Parenthesis theorem.* Let  $I(q)$  denote the interval  $[d[q], f[q]]$ , and let  $I(q) \prec I(r)$  denote that  $f[q] < d[r]$  holds. In a DFS-tree, for any two states  $q$  and  $r$ , one of the following four conditions holds: (1)  $I(q) \subseteq I(r)$  and  $q$  is a descendant of  $r$ ; (2)  $I(r) \subseteq I(q)$  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.

False. Counterexample:



Here, if the algorithm chooses  $q$  before  $r$ , it backtracks from  $q$  before discovering  $r$ .

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  $\sigma \models \mathbf{G}(p \mathbf{U} r)$ .



False. A counterexample is e.g.  $\{q\}\{q, r\}^\omega$  or  $\sigma = (\{q\}\{p, r\})^\omega$ .

## Problem 2 Regular languages (8 credits)

Let  $\Sigma$  be an alphabet.

a)\* Give a procedure which takes two NFAs  $A_1, A_2$  over the same alphabet  $\Sigma$  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.

We have

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

We first construct an NFA  $B = (Q, \Sigma, \delta, q_0, F)$  for  $L(A_1) \cap L(A_2)$  by applying the pairing construction to  $A_1$  and  $A_2$ . We compute the set of states  $G$  which can reach a final state (in 0 or more steps):

$$G := \{q \in Q : \exists w \in \Sigma^*, q' \in F : q \xrightarrow{w} q'\}$$

The final automaton is  $A := (Q, \Sigma, \delta, q_0, G)$ .

Alternatively, we can remove all states of  $B$  which cannot reach a final state and then make all remaining states final.

b)\* Give a procedure which takes two NFAs  $A_1, A_2$  over the same alphabet  $\Sigma$  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.

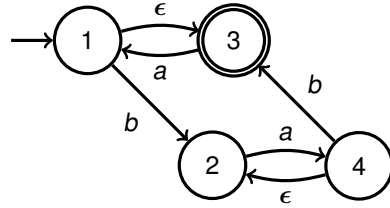
We mark all states of  $A_1$  and  $A_2$  which can reach a final state (in 0 or more steps) as final, resulting in NFAs  $B_1$  and  $B_2$  and then apply the pairing construction to obtain the NFA  $A$  for  $L(B_1) \cap L(B_2)$ .

Alternatively, we remove all states from  $A_1$  and  $A_2$  which cannot reach a final state. Then, we construct an NFA  $B$  by applying the pairing construction to the two resulting NFAs, and make all states of  $B$  final. The resulting NFA is  $A$ .

Applying the pairing construction first and then marking all states  $(q_1, q_2)$  which can reach both a state in  $F_1 \times Q_2$  and a state in  $Q_1 \times F_2$  as final does not work. It could be that  $q_1$  can reach  $F_1$  in  $A_1$ , but  $(q_1, q_2)$  cannot reach  $F_1 \times Q_2$  in  $B$  (this is the same problem you get when you apply pairing construction for NFA union). However, this works if trap states are added to  $A_1$  and  $A_2$  first.

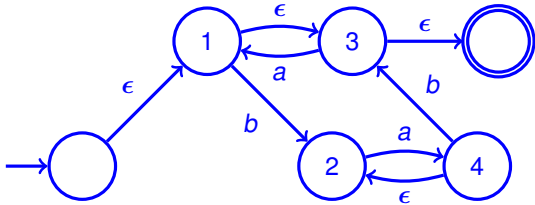
### Problem 3 NFA $\epsilon$ to Regular Expression (6 credits)

Consider the NFA $\epsilon$   $N$  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  $\alpha$ .) Remove the states **in ascending order**.

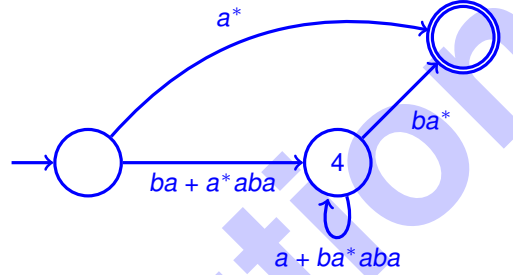


	0
	1
	2
	3
	4
	5
	6

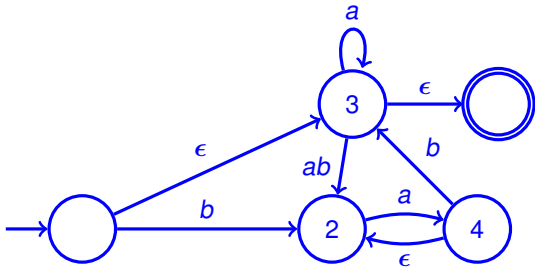
Preprocessing:



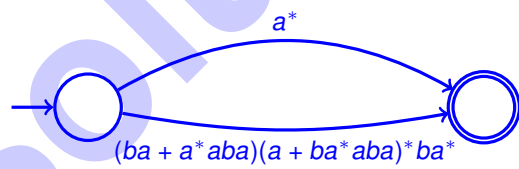
Combine:



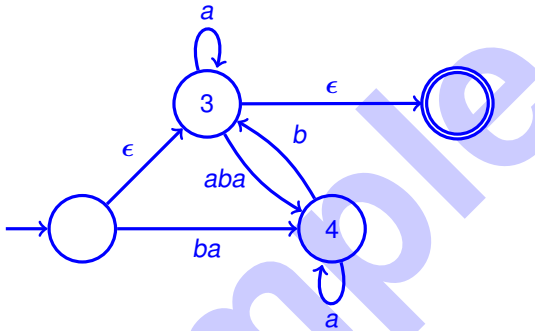
Remove 1:



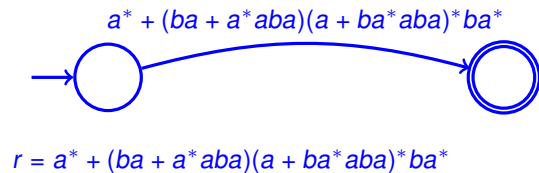
Remove 4:



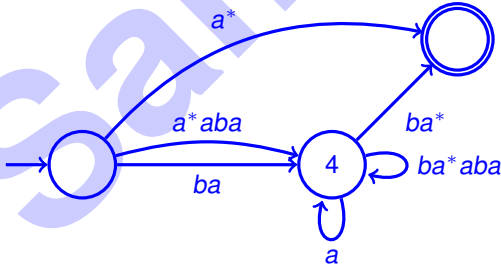
Remove 2:



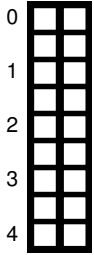
Combine:



Remove 3:

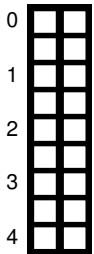
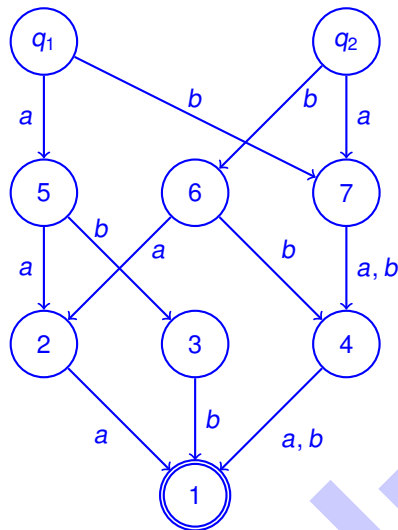


### Problem 4 Fixed-length languages (8 credits)

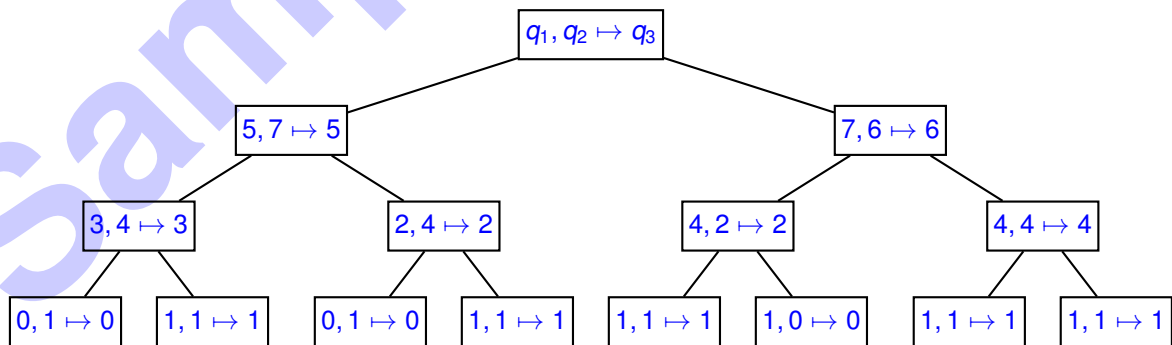


a)\* Let  $\Sigma = \{a, b\}$ . Construct the multi-DFA (i.e., the fragment of the fixed-length master automaton) for the set of languages  $\{L_1, L_2\}$ , where  $L_1 = L(aaa + abb + b\Sigma)$  and  $L_2 = L(baa + bb\Sigma + a\Sigma\Sigma)$ .

The states recognizing  $L_1$  and  $L_2$  are  $q_1$  and  $q_2$ , respectively.



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* 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.



$q_3$  is a new state with state 5 and 6 as  $a$ - and  $b$ -successor, respectively.

## 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  $u = u_1 \cdots u_n$ .

For each state  $q$  we construct a transducer  $T_q$  over the alphabet  $\Sigma \times \Sigma$  recognizing all pairs of words  $(w_1 \cdots w_n, u_1 \cdots u_n)$  such that

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

The states of  $T_q$  are all pairs of states of  $Q$ . The initial state is  $(q_0, q)$ , the final state is  $(q, q_f)$ . There is

a transition  $(r_1, r_2) \xrightarrow{\begin{matrix} a_1 \\ a_2 \end{matrix}} (r'_1, r'_2)$  iff  $r_1 \xrightarrow{a_1} r'_1$  and  $r_2 \xrightarrow{a_2} r'_2$  are transitions of  $A$ .

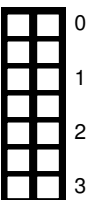
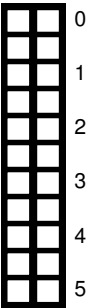
We have  $P_A = \bigcup_{q \in Q} L(T_q)$  and so we can construct a transducer for  $P_A$ .

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.

$H_L$  is the result of projecting the language  $P_A$  from a) onto the first component. So we can apply the operation for projection of transducers.



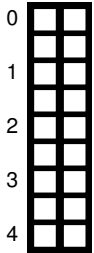
## Problem 6 Alternating letters (11 credits)

Let  $L_1$  and  $L_2$  be  $\omega$ -languages over an alphabet  $\Sigma$ . 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  $\{(ab)^\omega\} \sim \{(ba)^\omega\} = \{a^\omega\}$ .

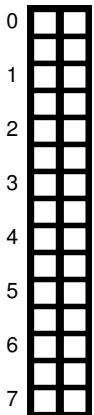
a)\* Let  $\Sigma = \{a, b\}$ , let  $L_1$  be the language of all words with infinitely many  $a$ 's, and let  $L_2$  be the language of all words with infinitely many  $b$ 's. Give an  $\omega$ -regular expression  $r$  for  $L_1 \sim L_2$ , and give a proof that your  $\omega$ -regular expression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



$$r = \Sigma^\omega$$

Proof: Let  $w = w_1 w_2 w_3 \dots \in \Sigma^\omega$ . Define  $u_1 = w_1 a w_3 a w_5 a \dots$  and  $u_2 = b w_2 b w_4 b w_6 \dots$ . We have  $u_1 \in L_1$  and  $u_2 \in L_2$ ; therefore  $w \in L_1 \sim L_2$ .

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  $\omega$ -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.



$N = (Q, \Sigma, \delta, Q_0, \mathcal{F})$  is an NGA with

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$Q_0 = Q_{01} \times Q_{02} \times \{1\}$$

$$\mathcal{F} = \{F_1 \times Q_2 \times \{1, 2\}, Q_1 \times F_2 \times \{1, 2\}\}$$

$$\delta((q_1, q_2, 1), a) = \delta_1(q_1, a) \times \bigcup_{b \in \Sigma} \delta_2(q_2, b) \times \{2\}$$

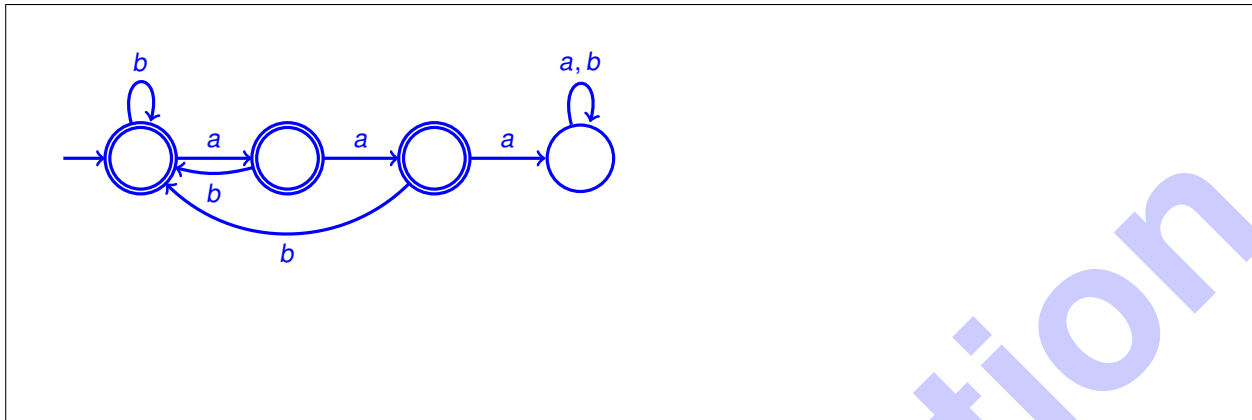
$$\delta((q_1, q_2, 2), a) = \bigcup_{b \in \Sigma} \delta_1(q_1, b) \times \delta_2(q_2, a) \times \{1\}$$

We alternate between making steps of  $N_1$  and  $N_2$ , starting with  $N_1$ . When we make a step of  $N_1$ ,  $N_2$  makes a step with all possible letters, and vice versa. We accept if we visit an accepting state of  $N_1$  infinitely often **and** an accepting state of  $N_2$  infinitely often. (We do not need to visit an accepting state of  $N_1$  and  $N_2$  simultaneously.)

### 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  $L(\Sigma^*aaa\Sigma^*)$ .



0
1
2
3

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.

A correct regular expression is e.g.  $(b + ab + aab)^*(a + \epsilon)(a + \epsilon)$ .

0
1
2

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.

- $L((a + b)^*a(b + c)^*)$
- $L((a^*b^*c^*)^*c)$
- $L((ab + c)^*)$

0
1
2
3
4

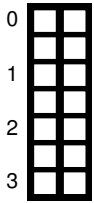
- $\exists x \left( Q_a(x) \wedge \forall y ((y < x \rightarrow (Q_a(y) \vee Q_b(y))) \wedge (x < y \rightarrow (Q_b(y) \wedge Q_c(y))) \right)$
- This is the language  $\Sigma^*c$ , and so a possible formula is  $\exists x(\text{last}(x) \wedge Q_c(x))$ .
- $\forall x \left( (Q_a(x) \rightarrow \exists y (y = x + 1 \wedge Q_b(y))) \wedge (Q_b(x) \rightarrow \exists y (x = y + 1 \wedge Q_a(y))) \right)$

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

The computations are those that satisfy  $Gp$  until they satisfy  $Gq$ . This means that either  $Gq$  holds initially (yielding  $(\{q\} + \{p, q\})^\omega$ ), or  $Gp$  holds until  $Gq$  holds (yielding  $(\{p\} + \{p, q\})^* \{p, q\}^\omega$ ). Together we get

$$(\{q\} + \{p, q\})^\omega + (\{p\} + \{p, q\})^* \{p, q\}^\omega$$

0
1
2
3

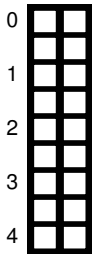


e)\* Let  $L \subseteq \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}^*$  be the language encoding the solutions of the Presburger formula  $y = x + 3$  in least significant bit encoding.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

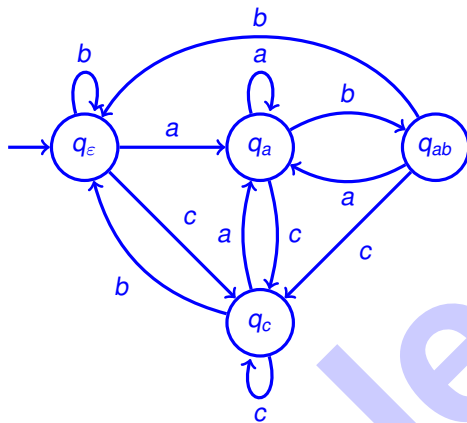
Let  $L'$  be the residual with respect to  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  and let  $w_x$  denote that  $w$  is a lsbf encoding of  $x$ . We have

$$\begin{aligned}
 & (w_x, w_y) \in L' \\
 \text{iff } & (0w_x, 1w_y) \in L \quad (\text{definition of residual w.r.t. } \begin{bmatrix} 0 \\ 1 \end{bmatrix}) \\
 \text{iff } & (w_{2x}, w_{2y+1}) \in L \quad (\text{lsbf encoding}) \\
 \text{iff } & 2y + 1 = 2x + 3 \quad (\text{by the definition of } L, (w_a, w_b) \in L \text{ iff } b = a + 3) \\
 \text{iff } & y = x + 1
 \end{aligned}$$

So the formula is  $y = x + 1$ .

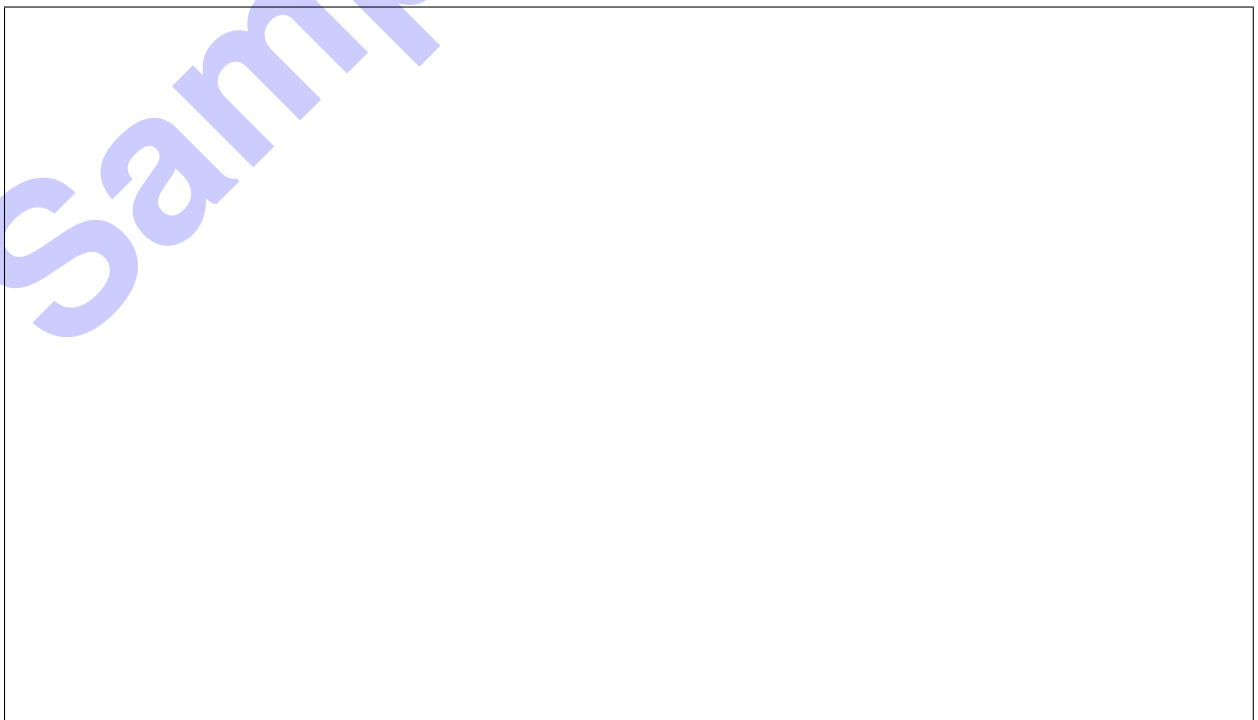


f)\* Give a *deterministic* Rabin automaton for the language of all  $\omega$ -words over the alphabet  $\{a, b, c\}$  that contain infinitely many occurrences of  $ab$  or finitely many occurrences of  $c$ .



Acceptance condition:  $\{(\{q_{ab}\}, \emptyset), (Q, \{q_c\})\}$

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



Sample Solution

Sample Solution