Chair for Foundations of Software Reliability and Theoretical Computer Science
School of Computation, Information and Technology
Technical University of Munich

Esolution
Place student sticker here

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 / Endterm	**Date:**	Wednesday 28th February, 2024
Examiner:	Prof. Javier Esparza	**Time:**	10:30 – 12:30

	P 1	P 2	P 3	P 4	P 5	P 6	P 7
I							

Working instructions

- This exam consists of **16 pages** with a total of **7 problems** including one **bonus problem**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 80 credits.
- To pass the exam, 35 credits are *sufficient*.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - one **analog dictionary** English $\leftrightarrow$ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- The points of the bonus problem count for your grade, but we disregard them when calculating the grading scheme. In particular, to receive the best grade it suffices to achieve all non-bonus points.
- **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 (22 credits)

Please answer the following questions. For true/false questions, provide a justification of your answer for “true” answers, and a counterexample for “false” answers. Otherwise **no points will be awarded!** We use $\Sigma := \{a, b\}$ as alphabet in this exercise.

a)* Let $L \subseteq \Sigma^*$. True or false: if $L = L^R$, then L is regular.

Note: For a word $w = a_1...a_n$, where $a_1, ..., a_n \in \Sigma$, we write $w^R := a_n...a_1$ for the *reverse* of w, and define $L^R := \{w^R : w \in L\}$.

☐ True [✓ False]

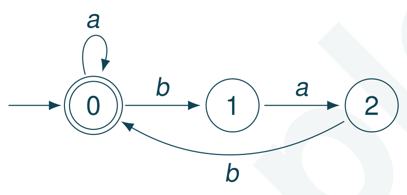
Consider the language $L = \{w : |w|_a = |w|_b\}$, i.e. the words that contain the same number of a and b. Then $L^R = L$, but L is not regular.

b)* True or false: every NFA M accepting the language $L := \{a, ab\}^{2024}$ has at least 4048 states.

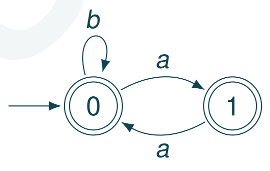
[✓ True] ☐ False

As $w := (ab)^{2024} \in L$, M has an accepting run on w. Since L is finite, the run does not contain a cycle, and so it visits every state at most once. Since the run traverses $|w| + 1 = 4049$ states, the automaton has at least 4048 states.

c)* Construct a DFA with at most 5 states recognising the language of $(a + bab)^*$.



d)* Let $L \subseteq \Sigma^*$ denote the language of words where no b is preceded by an odd-length sequence of a, i.e. L is generated by the RE $((aa)^*b)^*a^*$. For example, $\varepsilon, aabbba, baab \in L$, and $aabab \notin L$.
Construct an NFA N with $L(N) = L$ and at most 5 states, such that all states of N are accepting.



e)* Let AP = $\{p, q\}$ be a set of atomic propositions. Give an LTL formula φ over AP such that $L(\varphi)$ is the set of all computations where every occurrence of $\{p\}\{p\}$ is followed by some later occurrence of $\emptyset\ \emptyset$.

☐ 0 ☐ 1 ☐ 2

$G(\ ((p \wedge \neg q) \wedge X(p \wedge \neg q)) \ \rightarrow\ F((\neg p \wedge \neg q) \wedge X(\neg p \wedge \neg q))\ )$

f)* Let L denote an ω-regular (!) language. True or false: if $L = \Sigma L$, then $L = \emptyset$ or $L = \Sigma^\omega$.

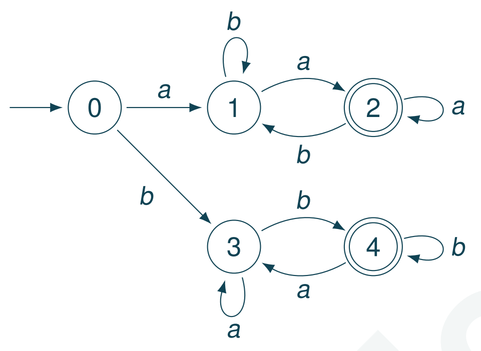
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4

True [✓ False]

A counterexample is $((a + b)^* a)^\omega$, i.e. the language of words containing infinitely many a.

g)* Let $L_1 \subseteq \Sigma^\omega$ denote the language of ω-words w such that the first letter of w appears infinitely often in w. Give a Büchi automaton accepting L_1 with at most 5 states.

☐ 0 ☐ 1 ☐ 2



h)* Consider the language L_1 defined in exercise g). True or false: every Büchi automaton A for L_1 has at least two **accepting** states (and any number of non-accepting states).

Note: If your automaton in exercise g) already accepts L_1 with only one accepting state, you are automatically awarded the points for this exercise.

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4

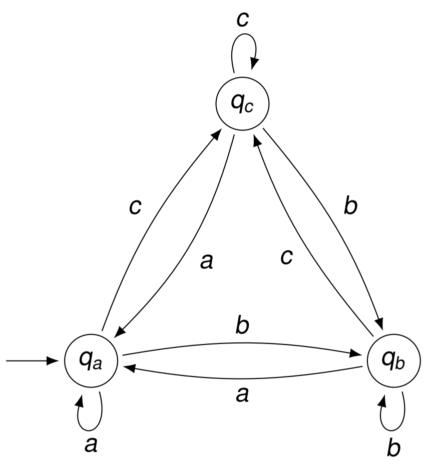
[✓ True] False

Let q_f denote the unique accepting state of A. We have $a^\omega, b^\omega \in L$, so A must have an accepting run for both of these words. The state q_f must appear infinitely often in both of these runs, so there is some i such that A can reach q_f after reading a^i, and starting from q_f there is some accepting run reading b^ω. So the word $a^i b^\omega \notin L_1$ is accepted, a contradiction.

Problem 2 Omega-Automata (8 credits)

0 1 2 3 4

a)* Consider the following semi-automaton with $\Sigma = \{a, b, c\}$:



For each one of the languages L_1, L_2, and L_3 below, decide whether there exists a Rabin condition with only one Rabin pair such that the above semi-automaton recognizes the language. If it does, give such a condition in the format $\langle F, G\rangle$. If it does not, justify why no such condition exists.
Reminder: A run satisfies the Rabin condition $\langle F, G\rangle$ if it visits the set F of states infinitely often and the set of states G finitely often.

$L_1 = \{w \in \Sigma^\omega \mid w$ contains infinitely many a and finitely many $b\}$

[✓ Exists] Does not exist

$L_2 = \{w \in \Sigma^\omega \mid w$ contains finitely many b and finitely many $c\}$

[✓ Exists] Does not exist

$L_3 = \{w \in \Sigma^\omega \mid w$ contains infinitely many a, infinitely many b and infinitely many $c\}$

Exists [✓ Does not exist]

L_1 and L_2 are possible. L_1 is the result when choosing the Rabin pair $(\{p_a\}, \{p_b\})$, L_2 is the result when choosing any Rabin pair $(F, \{p_b, p_c\})$ with $p_a \in F$.
L_3 is impossible: Namely we claim that any single Rabin pair would already accept too many words, and hence any union of Rabin conditions would also accept too much. If the Rabin pair (F, G) fulfills $G \neq \emptyset$ (and is non-contradictory, i.e. $F \not\subseteq G$), then it accepts wrong words. Hence $G = \emptyset$. The only reasonable choice is $F = Q$, but this only guarantees that *some* letter is seen infinitely often, hence wrong words are accepted.

b)* Consider the following non-deterministic semi-automaton with $\Sigma = \{a, b\}$:

0 1 2 3 4

$$\begin{tikzcd} {} \arrow[r] & p \arrow[loop above, "{a,b}"] \arrow[r, bend left, "a"] & q \arrow[loop above, "a"] \arrow[l, bend left, "a"] \end{tikzcd}$$

Which of the following languages can be recognized by this semi-automaton by using an appropriate Büchi-condition? If such a condition exists, give it. If it does not, justify why no such condition exists.

$L_1 = \{w \in \Sigma^\omega \mid w \text{ contains infinitely many } a\}$

Exists [✓ Does not exist]

$L_2 = \{w \in \Sigma^\omega \mid w \text{ contains finitely many } b\}$

Exists [✓ Does not exist]

$L_3 = \{w \in \Sigma^\omega \mid w \text{ contains } aa \text{ infinitely often}\}$

[✓ Exists] Does not exist

Only L_3. If $p \in F$, then the automaton accepts Σ^ω, hence only $F = \{q\}$ is interesting. In order for a run to be infinite, it can only move to state q if it reads a and knows that another a is coming. I.e. at the start of an infix aa. The corresponding run moving to q whenever we read a and when another a is coming next is also feasible. Hence q can be visited infinitely often if and only if w contains aa infinitely often.

Problem 3 FO/MSO on words (10 credits)

In the following questions we consider MSO(Σ), monadic second order logic on words, over the alphabet $\Sigma = \{a, b\}$. The first position in a word w has index 1. You are allowed to use the macros *first* and $x = y + 1, x = y + 2, \ldots$ from the lecture. You are not allowed to use any other macros.

0 1 2 3

a)* Give a formula $\mathit{EveryThirdPosition}(X)$ which is true if and only if X contains exactly every third position, starting from the first position. For example, if the word has length 5, then $X = \{1, 4\}$ is the only model.

$\mathit{EveryThirdPosition}(X) = \forall x(x \in X) \leftrightarrow (\mathit{first}(x) \vee \exists y(y \in X \wedge x = y + 3))$.

0 1 2 3

b)* Using the formula $\mathit{EveryThirdPosition}(X)$ as a macro, give a formula $\mathit{RepeatingABB}$, which is true for w if and only if w is a finite prefix of $(abb)^\omega$, i.e. if and only if $w \in (abb)^*(\varepsilon \mid a \mid ab)$.

$\mathit{RepeatingABB} = \forall X\colon\ \mathit{EveryThirdPosition}(X) \rightarrow (\forall x\colon (x \in X) \leftrightarrow Q_a(x))$.

Define for every $k \in \mathbb{N}_{>0}$ the macros

$$\mathit{PositionsGreaterK}(X) := \forall x\Big((x \in X) \leftrightarrow \big(\exists z_1 \exists z_2 \ldots \exists z_k \bigwedge_{i=1}^{k-1}(z_i < z_{i+1}) \wedge z_k < x\big)\Big)$$

$$\varphi_k(x) := \exists X(\mathit{PositionsGreaterK}(X) \wedge x \in X)$$

0 1 2

c)* True or false: For every $k \in \mathbb{N}_{>0}$ there exists a formula in FO(Σ) (first-order logic on words) equivalent to the formula φ_k.
If yes, give a family of such formulas. If not, justify your answer.

[✓ Yes] No

An equivalent formula is $\exists z_1 \exists z_2 \ldots \exists z_k \big(\bigwedge_{i=1}^{k-1}(z_i < z_{i+1}) \wedge z_k < x\big)$.
(Not necessary for points): To see this, observe that in $\exists X\colon\ \mathit{PositionsGreaterK}(X)$ there is exactly one choice for X in the quantifier, namely the set X of all positions $> k$. Then the statement becomes "x is in the set X of all positions $> k$", i.e. $x > k$.

0 1 2

d)* Given $k \in \mathbb{N}_{>0}$, define $\psi_k := \forall x(\varphi_k(x) \rightarrow Q_b(x))$. True or false: there is a formula of MSO(Σ) such that for every word $w \in \Sigma^*$, w satisfies the formula iff w satisfies ψ_k for at least one $k \in \mathbb{N}_{>0}$?
If yes, give an example of such a formula. If not, justify your answer.

[✓ Yes] No

Yes, an example formula is $\forall x \mathit{False} \vee \exists y \forall x(y < x \rightarrow Q_b(x))$. The second part is the important one, the first part is only to ensure the empty word is accepted.
Another solution is to observe that the disjunction is in fact a tautology, hence True is also a solution.

Problem 4 LTL (10 credits)

a)* Consider the set of atomic propositions AP = $\{p, q\}$ and the LTL formula $\varphi := p \, \mathrm{U}(q \, \mathrm{U} \, \mathrm{G}(q \wedge p))$. Give an ω-regular expression s over the alphabet $\{\emptyset, \{p\}, \{q\}, \{p, q\}\}$ such that the languages of s and φ are equal. You do not need to justify your answer.

☐ 0 ☐ 1 ☐ 2 ☐ 3

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

b)* Which of the following formulas are equivalent to the formula φ from the previous question?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4

☐ $(p \wedge q) \, \mathrm{U} \, \mathrm{G}(q \wedge p)$

☒ $p \, \mathrm{U}(\mathrm{G}q \wedge \mathrm{FG}p)$

☐ $p \, \mathrm{U}(\mathrm{FG}(q \wedge p))$

Give a counter-example for the ones that are not equivalent. You do not need to provide an explanation for the ones that are equivalent.

Only the second is equivalent. The first forbids $\{p\}\{p, q\}^\omega$, the last allows $\{p\}\emptyset\{p, q\}^\omega$.
(Not necessary for the points): To see the equivalence, use the ω-regular expression of subproblem a). This implies that the second formula is equivalent, since $\mathrm{G}q \wedge \mathrm{FG}p$ also holds exactly for traces of the form $(\{q\} + \{p, q\})^*\{p, q\}^\omega$.

c)* Are the formulas $\psi_1 := p \, \mathrm{U} \, (\mathrm{F}q)$ and $\psi_2 :=$ **true** $\mathrm{U} \, (\mathrm{F}q)$ equivalent?
If yes, prove that they are equivalent. If no, provide a suitable computation as counterexample.

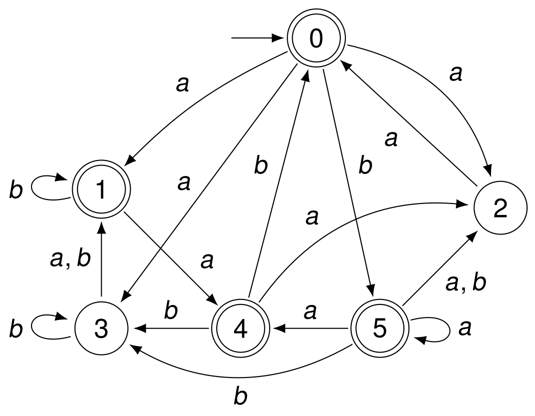
☐ 0 ☐ 1 ☐ 2 ☐ 3

☒ Equivalent ☐ Not equivalent

Proof: Clearly ψ_1 implies ψ_2, hence it suffices to prove the other direction.
Let $\sigma = \sigma_0\sigma_1 \ldots$ be a computation satisfying $\psi_2 = \mathrm{True} \, \mathrm{U} \, \mathrm{F}q$. By definition of until and $\mathrm{F}q$ there exists $k \in \mathbb{N}$ such that $\sigma^k = \sigma_k\sigma_{k+1} \ldots$ fulfils $q \in \sigma_k$. This implies that $\sigma = \sigma^0$ fulfils $\mathrm{F}q$. By choosing $n = 0$ for the until, σ fulfils $p \, \mathrm{U} \, \mathrm{F}q$.

Problem 5 NFA Universality (8 credits)

a)* Consider the following NFA N:



Decide whether N is universal, i.e. whether $L(N) = \{a, b\}^*$, using the algorithm UnivNFA from the lecture with the subsumption check. (If you execute the algorithm without the subsumption check, you can still receive up to 3 points.)

Whenever choosing which item to remove from the workset $\mathcal{W}$, choose the item that has been in the workset the longest. Consider always letter a before b.

While executing the algorithm, fill out the table below. Enter each item removed from the workset in the first column. Use the other two columns to enter the items (if any) that are added to the workset in this iteration. (If you wish, you can also note items that you considered adding to the workset, but did not add, e.g. because they failed the subsumption test. Mark these items with $\times$.)

Universal [✓ Not universal]

With subsumption:

removed from $\mathcal{W}$	added to $\mathcal{W}$ (a)	added to $\mathcal{W}$ (b)
$\{0\}$	$\{1,2,3\}$	$\{5\}$
$\{1,2,3\}$	$\{0,1,4\}$ $\times$	$\{1,3\}$
$\{5\}$	$\{2,4,5\}$ $\times$	$\{2,3\}$
$\{1,3\}$	$\{1,4\}$	$\{1,3\}$ $\times$
$\{2,3\}$		

Without subsumption:

removed from $\mathcal{W}$	added to $\mathcal{W}$ (a)	added to $\mathcal{W}$ (b)
$\{0\}$	$\{1,2,3\}$	$\{5\}$
$\{1,2,3\}$	$\{0,1,4\}$	$\{1,3\}$
$\{5\}$	$\{2,4,5\}$	$\{2,3\}$
$\{0,1,4\}$	$\{1,2,3,4\}$	$\{0,1,3,5\}$
$\{1,3\}$	$\{1,4\}$	$\{1,3\}$ $\times$
$\{2,4,5\}$	$\{0,2,4,5\}$	$\{0,2,3\}$
$\{2,3\}$		

b) If N is universal, give a transition such that N is no longer universal when that transition is deleted. If N is not universal, give a word $w \in \{a, b\}^* \setminus L(N)$.

$bb \notin L(N)$

Problem 6 The infinite master automaton (12 credits)

We consider the master automaton M for fixed-length languages, as defined in the lecture: $M := (Q, \Sigma, \delta, F)$, where Q is the set of all fixed-length languages, $\Sigma := \{a, b\}$, $\delta(L, x) := L^x$ for $L \in Q, x \in \Sigma$ and $F := \{\{\varepsilon\}\}$. (Recall that M has infinitely many states.)

a)* Prove that there is no infinite, simple, directed path in M, i.e. an infinite sequence of pairwise distinct $L_1, L_2, ... \in Q$ such that for each i there is an $x \in \Sigma$ with $L_{i+1} = \delta(L_i, x)$.

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4

Assume that such a path exists. If there is an i with $L_i = \emptyset$, then we would also get $L_{i+1} = \emptyset$, contradicting the requirement that $L_1, L_2, ...$ are pairwise distinct.
We can thus define $n_i := \max\{|w| : w \in L_i\}$ as the maximum length of words in L_i. (Of course, all words in L_i have the same length, so it does not really matter that we take the maximum.) We now find $n_{i+1} < n_i$ for all i, as $L_{i+1} = L_i^x$ for some $x \in \Sigma$. But this is a contradiction as well, since there cannot be an infinitely descending sequence of natural numbers.

b)* Prove or disprove: the nonempty languages of M are connected when viewed as undirected graph. More precisely, show that for every pair of nonempty, fixed-length languages $R, S \in Q \setminus \{\emptyset\}$ there is a finite sequence $L_1, ..., L_k \in Q \setminus \{\emptyset\}$ with $L_1 = R$, $L_k = S$ and for each $i \in \{1, ..., k-1\}$ there is an $x \in \Sigma$ such that either $L_{i+1} = \delta(L_i, x)$ or $L_i = \delta(L_{i+1}, x)$.
Note: Such a sequence $L_1, ..., L_k$ is a *path connecting R and S*.

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4

[✓ True] ☐ False

Let n, m such that $R \subseteq \Sigma^n$ and $S \subseteq \Sigma^m$. We assume $n \leq m$ wlog. We proceed by strong induction on $m - n$.
In the base case, $m - n = 0$. Let $L := \{a\}R \cup \{b\}S$. We have $L \subseteq \Sigma^{n+1}$, so L is a fixed-length language. Additionally, $L^a = R$ and $L^b = S$, so R, L, S is a path.
For the induction step, let $L := \{a\}R$. So $L \subseteq \Sigma^{n+1}$ and by induction hypothesis there is some path $L_1, ..., L_k$ connecting S and L. As $R = L^a$, we have that $L_1, ..., L_k, R$ connects S and R. (Note $L_1 = S$, $L_k = L$.)

0 ☐
1 ☐
2 ☐
3 ☐
4 ☐

c)* We now consider a variant of M, which we refer to as M'. In M' we set $Q := 2^{\Sigma^*}$ (that is, to the set of all languages over Σ, fixed-length or not and regular or not), define δ as for M, and define F as the set of all languages containing ε.
Is the statement of part a) still true in M'? More precisely, prove or disprove: there is an infinite, simple, directed path in M'.

[✓ True] False

Let $L \subseteq \{a\}^*$ be a non-regular language. (We know that non-regular unary languages exist, as there is an uncountable number of unary languages, but only a countable number of regular languages.) We set $L_i := L^{a^i}$ and claim that $L_1, L_2, ...$ is an infinite, simple, directed path in M'. Clearly, $L_{i+1} = L_i^a$ for all i, so it remains to show that $L_1, L_2, ...$ are pairwise distinct. First, we observe that $S = \{L_i : i \in \mathbb{N}\}$ are precisely the residual languages of L over the alphabet $\{a\}$. As L is not regular, $|S| = \infty$. But if $i < j$ with $L_i = L_j$ were to exist, we would have $S = \{L_1, ..., L_{j-1}\}$, a contradiction.

Problem 7 **Bonus question: Fully connected (10 credits)**

Let $R \subseteq \mathbb{N} \times \mathbb{N}$ be a relation on natural numbers. We say that a number n is *fully connected* w.r.t. R if $(n, m) \in R$ for every $m \in \mathbb{N}$. We let FC_R denote the set of all fully connected numbers w.r.t. R.

a)* Give an algorithm satisfying the following specification:

0 1 2 3 4 5 6

- Input: a well-formed deterministic transducer recognizing a relation $R \subseteq \mathbb{N} \times \mathbb{N}$ in lsbf encoding.

  Note: Recall that a transducer recognizes R if for every pair $(n, m) \in R$ it accepts every encoding of (n, m), and for every pair $(n, m) \notin R$ it accepts no encoding of (n, m).

- Output: a DFA recognizing the set FC_R.

Hint: $n \in FC_R$ iff $\forall m \in \mathbb{N}\colon (n, m) \in R$ iff $\neg\exists m \in \mathbb{N}\colon (n, m) \notin R$.

We have

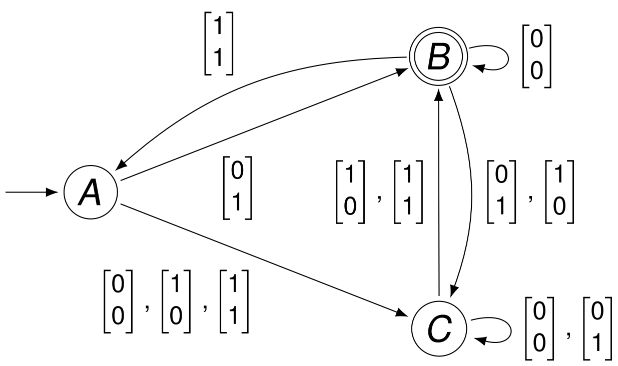
$$\begin{aligned}
& n \in FC_R \\
\text{iff } & \forall m \in \mathbb{N}\colon (n, m) \in R \\
\text{iff } & \neg\exists m \in \mathbb{N}\colon (n, m) \notin R \\
\text{iff } & \neg\exists m \in \mathbb{N}\colon (n, m) \in \textbf{Complement}(R) \\
\text{iff } & \neg(n \in \textbf{Projection}_1(\textbf{Complement}(R))) \\
\text{iff } & n \in \textbf{Complement}(\textbf{Projection}_1(\textbf{Complement}(R)))
\end{aligned}$$

It follows $FC_R = \textbf{Complement}(\textbf{Projection}_1(\textbf{Complement}(R)))$. Therefore, the algorithm is:

- Compute a transducer for **Complement**(R); since the input is a deterministic transducer, it suffices to apply the algorithm for DFAs, that is, swap the accepting and non-accepting states.
- Compute a transducer for $\textbf{Projection}_1$(**Complement**(R)) using the algorithm from the course (remove the second component of each letter and apply the pad closure).
- Complement the resulting NFA by first determinizing the NFA and then swapping the accepting and non-accepting states.

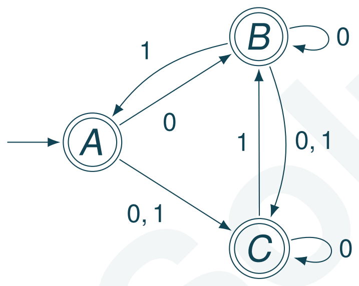
0 ☐
1 ☐
2 ☐
3 ☐
4 ☐

b) Apply the algorithm of part a) to the transducer below, giving enough information about the intermediate steps. Interpret the result by describing FC_R not as a language, but as a set of numbers. (Solutions like "the set of numbers encoded by this language" get no points.)

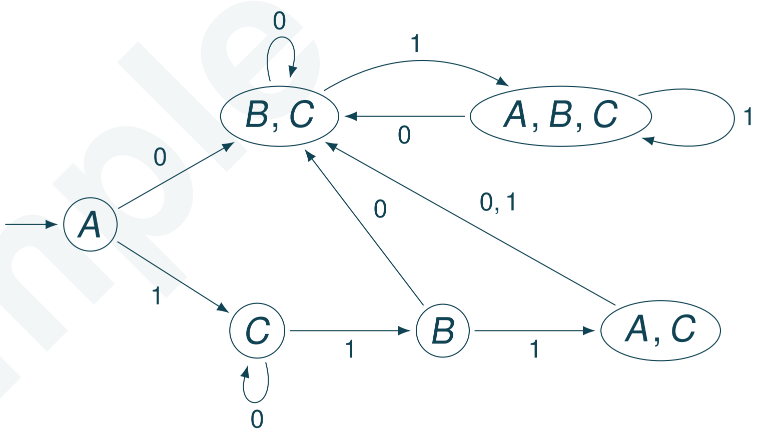


A
B
C
$\begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$

The first step just makes A and C accepting and B non-accepting. After the second step we obtain



B becomes an accepting state because of the padding closure.
Determinizing and then swapping accepting and non-accepting states yields



We get $FC_R = \emptyset$.

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