Technische Universität München
Winter term 2020/21, 03.03.2020
Prof. J. Esparza / M. Lazić / C. Weil-Kennedy

Automata and Formal Languages — Endterm Exam

- You have 120 minutes to complete the exam.
- Answers must be written in a separate booklet. Do not answer on the exam.
- Please let us know if you need more paper.
- Write your name and Matrikelnummer on every sheet.
- Write with a non-erasable pen. Do not use red or green.
- You can obtain 40 points. You need 17 points to pass. There are 4 bonus points in the last exercise.
- The ★ symbol indicates a more challenging question.

Question 1 (2 + 3 + 2+ 2 + 3 = 12 points)

a. Is there an NFA for $(a+b)c^*$ satisfying **all** the following conditions? If so, give one. If not, give a proof.

- No initial state has an incoming transition.
- No final state has an outgoing transition.
- For every state q, all transitions starting at q (if any) are labelled with the same letter.
- For every state q, all transitions ending at q (if any) are labelled with the same letter.

b. Give a transducer over alphabet $\{0,1\}^3$ accepting all least significant bit first (lsbf) encodings of pairs $(x, y, z) \in \mathbb{N}^3$ such that $x > 0, y = x - 1$ and $z = x + 1$. For example, $(0101, 1001, 1101)$ encodes $(10, 9, 11)$ and should be accepted, while $(110, 101, 011)$ encodes $(3, 5, 6)$ and should be rejected.

c. Recall that $\inf(w)$ denotes the set of letters occurring infinitely often in the infinite word w. Give a Büchi automaton and an ω-regular expression for the following ω-language over $\Sigma = \{a, b, c\}$:

$$L = \{w \in \Sigma^\omega : a \in \inf(w) \Rightarrow b \in \inf(w)\}.$$

d. Let $f : 2^{\mathbb{N}} \to \mathbb{N}$ be a surjective function.

Assume you are given an MSO formula $\text{Sum}(X, Y, Z)$ for X, Y, Z in $2^{\mathbb{N}}$ that is true if and only if $f(X) + f(Y) = f(Z)$. Give an MSO formula $\varphi(X, Y, Z)$ that is true if and only if $f(X) + f(Y) \leq f(Z)$.

e. Given languages L_1, L_2 over alphabet $\Sigma \neq \emptyset$, the *2-shuffle* of L_1 and L_2 is the language

$$L_1 \sqcup\!\sqcup L_2 := \{u_1 v_1 u_2 v_2 \mid u_1 u_2 \in L_1 \wedge v_1 v_2 \in L_2\}\,.$$

Let $\mathcal{K} = (Q_K, \Sigma, \delta_K, q_0^K, F_K)$ and $\mathcal{L} = (Q_L, \Sigma, \delta_L, q_0^L, F_L)$ be NFAs recognizing languages K and L, respectively. Give a tuple $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ such that $\mathcal{M}$ is an ε-NFA recognizing $K \sqcup\!\sqcup L$.

Question 2 (4 points)

Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$:

- $R \subseteq \Sigma^*$ is the language of all words of odd length, where the first and the last letter coincide. For example, $aba, abcca \in R$ and $bcab, baa \notin R$.

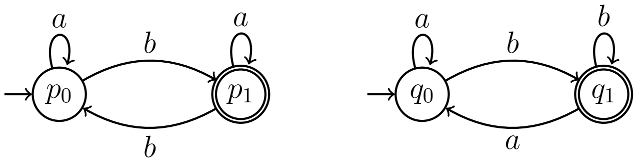
- $S \subseteq \Sigma^*$ is the language of all words 'w such that $|w|_a \leq |w|_b$ and $|w|_c \leq |w|_b$, where for every $\sigma \in \Sigma$ the expression $|w|_\sigma$ denotes the number of times that σ occurs in w. For example, $abbacc, bcab, cbcaacabb \in S$ and $aab, ccba \notin S$.

- Let $w = a_1 a_2 \ldots a_n \in \Sigma^*$. A *switch from a to b* in w is a pair of indices $1 \leq i < j \leq n$ such that $a_i = a$, $a_{i+1} = \cdots = a_{j-1} = c$, and $a_j = b$. Similarly, a *switch from b to a* in w is a pair of indices $1 \leq i < j \leq n$ such that $a_i = b$, $a_{i+1} = \cdots = a_{j-1} = c$, and $a_j = a$. For example, in $w = accbcaacbcba$ there are 2 switches from a to b ($\boldsymbol{accb}ca\boldsymbol{acb}cba$) and 2 from b to a ($acc\boldsymbol{bca}acbc\boldsymbol{ba}$). In $wb = accbcaacbcbab$ there are 3 switches from a to b ($\boldsymbol{accb}ca\boldsymbol{acb}c\boldsymbol{bab}$), but only 2 from b to a ($acc\boldsymbol{bca}acbc\boldsymbol{ba}b$).

  $T \subseteq \Sigma^*$ is the language of all words that have the same number of switches from a to b and switches from b to a. For example, $w = accbcaacbcba \in T$, but $wb = accbcaacbcbab \notin T$.

For each of the languages R, S, and T, decide if it is regular or not. If a language is regular, give a NFA that recognizes it. If it is non-regular, prove this by analyzing its residuals.

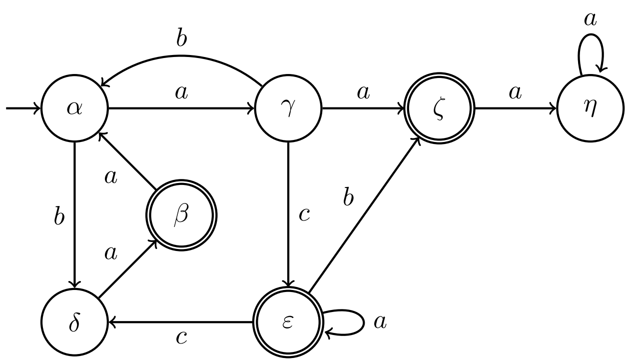
Question 3 (3 points)
Consider the two following NBAs $\mathcal{B}_1$ and $\mathcal{B}_2$:



(a) Give ω-regular expressions for the languages of the NBAs $\mathcal{B}_1$ and $\mathcal{B}_2$.

(b) Give the NBA $\mathcal{B}_1 \cap \mathcal{B}_2$ produced using the algorithm *IntersNBA* seen in class.

Question 4 (4 points)
Let $\mathcal{B}$ the following Büchi automaton.

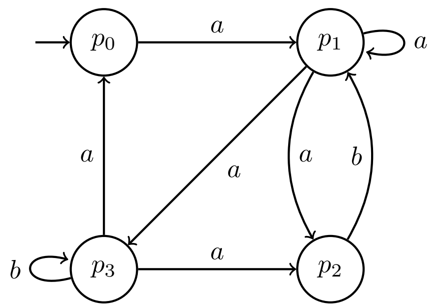


The language of $\mathcal{B}$ is not empty. Consider the algorithm *NestedDFS* seen in class, with procedures dfs_1 and dfs_2.

(a) Give the discovery and finishing times assigned by $dfs_1(q_0)$ to every state, starting from 1. Assume that, at every state, dfs_1 explores transitions labelled by a before transitions labelled by b, and transitions labelled by b before transitions labelled by c.

(b) Give the times at which dfs_2 is called on the final states p_2, p_4, p_6 (if at all). For each such procedure call, give the discovery and finishing times assigned by dfs_2 to each state it explores. Assume that calls to dfs_2 start at time 1, and that they also explore transitions labelled by a before transitions labelled by b, and transitions labelled by b before transitions labelled by c.

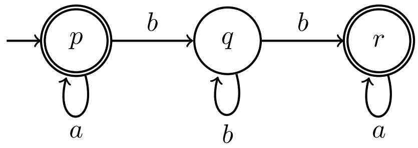
Question 5 (4 points)
Recall: a nondeterministic Muller automaton (NMA) is *empty* if it has no accepting run. Consider the following automaton A:



(a) Does the acceptance condition $F_1 = \{\{p_1, p_3\}\}$ give an empty Muller automaton? Justify your answer: if it is empty prove that no run is accepting, and if it is non-empty give an example of accepting run. Additionally, if it is non-empty, construct a non-deterministic Büchi automaton that recognizes the same language as A.

(b) Does the acceptance condition $F_2 = \{\{p_1, p_2\}\}$ give an empty Muller automaton? Justify your answer: if it is empty prove that no run is accepting, and if it is non-empty give an example of accepting run. Additionally, if it is non-empty, construct a non-deterministic Büchi automaton that recognizes the same language as A.

Question 6 (5 points)
Let $\mathcal{A}$ be the following NBA:

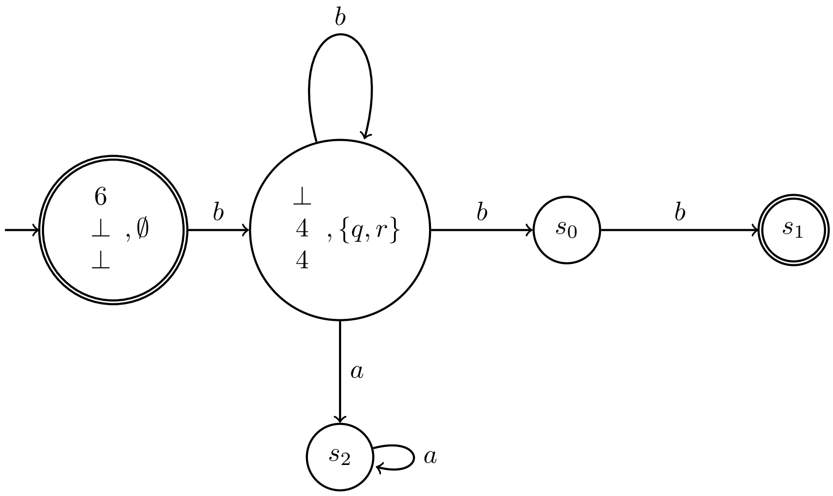


1. Draw dag(ab^ω) and dag($bbba^\omega$).

2. Does dag(ab^ω) admit an odd ranking? Give such a ranking if it exists. If it does not, argue why the conditions of an odd ranking cannot be fulfilled.

   Does dag($bbba^\omega$) admit an odd ranking? Give such a ranking if it exists. If it does not, argue why the conditions of an odd ranking cannot be fulfilled.

3. Below is part of the complement automaton $\overline{\mathcal{A}}$ constructed with the rank method seen in class. Give possible states for s_0, s_1, s_2. A reminder on notation: state $([\bot, 4, 4], \{q, r\})$ represents the level ranking $([\bot, 4, 4], \{q, r\})$ where q and r have rank 4 and there is no rank for p; and $\{q, r\}$ is the set of owing states.



Question 7 (6 points)
Let $AP = \{p, q, r\}$ and let $\Sigma = 2^{AP}$.

(a) Give an LTL formula that is satisfied by computations σ_1 and σ_2, and not satisfied by computations σ_3 and σ_4, where

$$\sigma_1 = \{p, q\}(\{r\}\{p, r\})^\omega, \quad \sigma_2 = \emptyset\emptyset\{p, q, r\}\{q\}^\omega, \quad \sigma_3 = \{r\}\{p, q, r\}\{p\}^\omega, \quad \sigma_4 = (\emptyset\{r\})^\omega.$$

Let C be the set of computations σ over Σ satisfying the following property: If there exists $i \geq 1$ such that $p, q \in \sigma(i)$ and $r \notin \sigma(i)$, then there also exists $j < i$ such that $r \in \sigma(j)$.

(b) Give a formula φ such that $L(\varphi) = C$

(c) Give an ω-regular expression s such that $L(s)$ is equal to the complement of $L(\varphi)$, that is, s represents the ω-language of all computations over Σ that do not belong to C.

Question 8 (2 points)
★ Let $\Sigma = \{a, b\}$. For every $n \geq 0$, let P_n the language of all palindromes over Σ of length $2n$.

(a) Show that every NFA recognizing P_n has at least 2^n states.

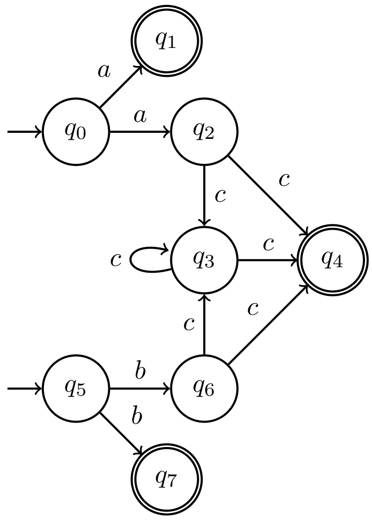
(b) For 2 **bonus** points: Show that every NFA recognizing P_n has at least $2^{n+1} - 1$ states.

(c) For 2 **bonus** points: Show that every NFA recognizing P_n has at least $2^{n+1} + 2^n - 2$ states.

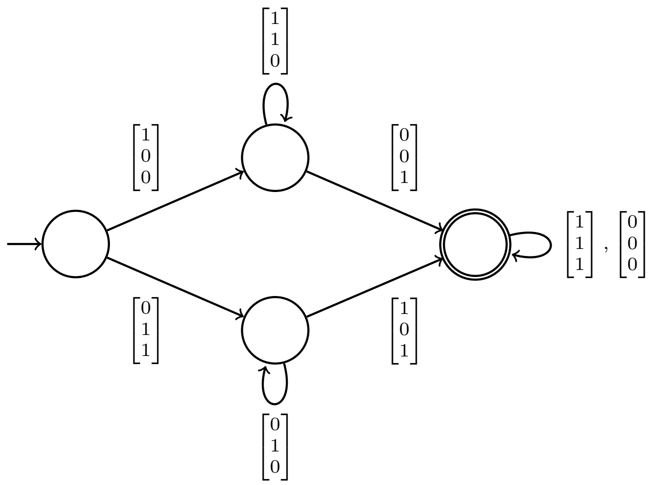
(A correct answer to (b) counts as a correct answer to both (a) and (b), and a correct answer to (c) counts as a correct answer to (a), (b), and (c).)

Solution 1 (2 + 3 + 2+ 2 + 3 = 12 points)

a. Here is a possible solution.



b. This is the minimal transducer:



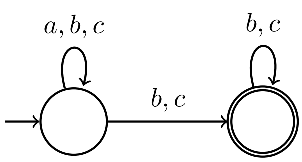
c. The exercise can be solved in many different ways, we just give one solution. There are other solutions with smaller ω-regular expressions and/or smaller NBAs.

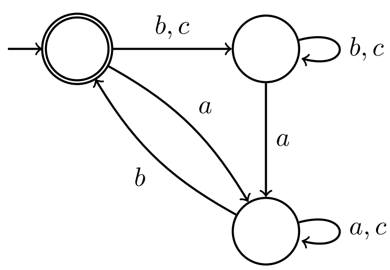
Observe that $L = L_1 \cup L_2$, where $L_1 = \{w \in \Sigma^\omega : a \notin \inf(w)\}$ is the language of all words over Σ containing finitely many as, and $L_2 = \{w \in \Sigma^\omega : \{a, b\} \subseteq \inf(w)\}$ is the language of all words over Σ containing infinitely many as and infinitely many bs. In the tutorials we have considered these two languages:

- ω-regular expressions: $(a+b+c)^*(b+c)^\omega$ for L_1, and $((b+c)^*a(a+c)^*b)^\omega$ for L_2 (there are others). So a possible ω-regular expression for L is

$$(a+b+c)^*(b+c)^\omega + ((b+c)^*a(a+c)^*b)^\omega$$

- NBAs for L_1 and L_2:





A possible NBA for L is just the result of putting the two automata above side by side (NBA with two initial states).

d. $\varphi(X, Y, Z) = \exists W \, \exists U \; Sum(X, Y, W) \wedge Sum(W, U, Z)$

e. Let $\mathcal{K} = (Q_K, \delta_K, \Sigma, q_0^K, F_K)$ and $\mathcal{L} = (Q_L, \delta_L, \Sigma, q_0^L, F_L)$ be NFAs recognizing K and L. We define an ε-NFA $\mathcal{M} = (Q, \delta, \Sigma, q_0, F)$ that recognizes $K \sqcup L$.

Intuitively, $\mathcal{M}$ runs in four phases, and decides nondeterministically when to move to the next phase. In phases 1 and 2 the automaton initiates a simulation of $\mathcal{K}$ and $\mathcal{L}$, respectively. In phase 3 it continues the simulation of $\mathcal{K}$ from the state reached at the end of phase 1. In phase 4 it continues the simulation of $\mathcal{L}$ from the state reached at the end of phase 2.

Formally, $\mathcal{M} = (Q, \delta, \Sigma, q_0, F)$ is defined as follows:

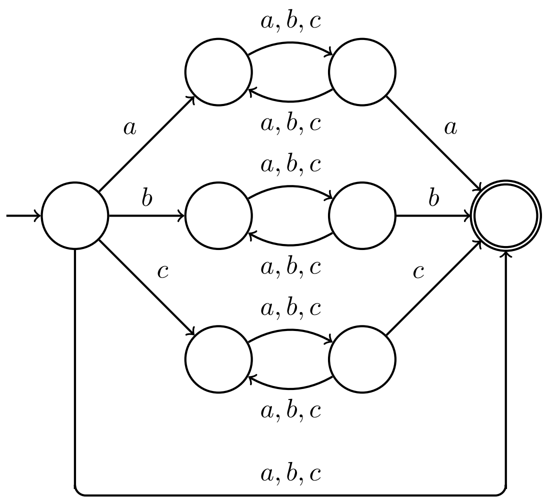
- $Q := Q_K \times Q_L \times \{1, 2, 3, 4\}$.
- A transition $[p, q, c] \xrightarrow{a} [p', q', c']$ belongs to δ if and only if

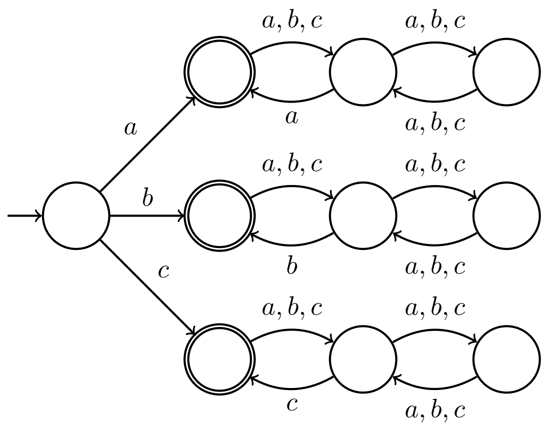
$$\begin{array}{llllllllll} a \in \Sigma & \text{and} & p \xrightarrow{a} p' \in \delta_K & \text{and} & q = q' & \text{and} & c' = c & \text{and} & c \in \{1, 3\}\,, & \text{or} \\ a \in \Sigma & \text{and} & p = p' & \text{and} & q \xrightarrow{a} q' \in \delta_L & \text{and} & c' = c & \text{and} & c \in \{2, 4\}\,, & \text{or} \\ a = \varepsilon & \text{and} & p = p' & \text{and} & q = q' & \text{and} & c' = c + 1 & \text{and} & c < 4\,. & \end{array}$$

- The initial state is $[q_0^K, q_0^L, 1]$.
- The set of final states is $F_K \times F_L \times \{4\}$.

Solution 2 (4 points)

(a) R is a regular language. Here are two NFAs recognizing R:





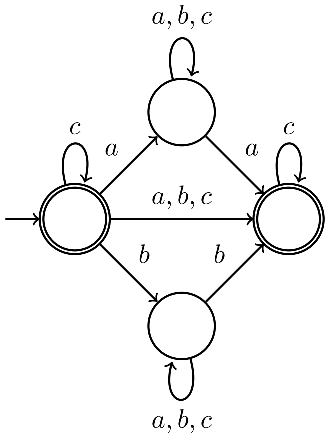
(b) S is non-regular. We prove that $S^{a^i} \neq S^{a^j}$ for every $0 \leq i < j$, $i \neq j$, which shows that S has infinitely many residuals. For this, we observe that $b^i \in S^{a^i}$, becaue $a^i b^i \in S$, but $b^i \notin S^{a^j}$, because $a^j b^i \notin S$.

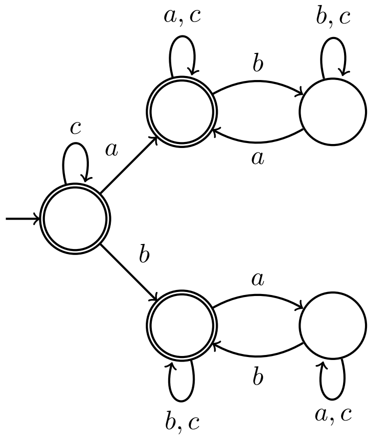
Common mistakes:

$b^i \notin S^{a^j}$ and $b^j \in S^{a^j}$ is not a proof that $S^{a^i} \neq S^{a^j}$. For example, it is compatible with $S^{a^j} = \{b^j\} = S^{a^j}$

Showing that a language $L \subseteq S$ is not regular is not a proof that S is not regular. Recall that Σ^* is regular and $L \subseteq \Sigma^*$. Thus, a superset of a non-regular language can be regular.

(c) T is a regular language. Intuitively, since a switch from a to b can only be followed by a switch from b to a, the language consists of all words over $\{a, b, c\}$ whose projection onto a, b is a word of $\varepsilon + a(a+b)^*a + b(a+b)^*b$. Here are two automata recognizing T:

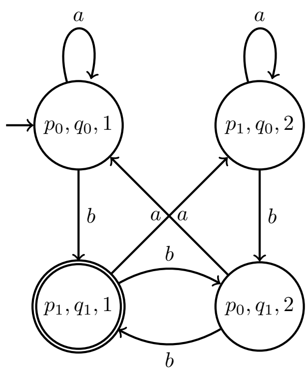




Solution 3 (3 points)
$\mathcal{B}_1$ recognizes the ω-language with an odd number of bs: this can be written $a^*b(a^*ba^*b+a^*)^\omega$, or $a^*b(ba^*b+a)^\omega$, or $a^*b(a^*(ba^*b)^*)^\omega$.

$\mathcal{B}_2$ recognizes the ω-language with an infinite number of bs: this can be written $(a^*b)^\omega$, or $a^*b(aa^*b+b)^\omega$, or $a^*b(b^*(aa^*b)^*)^\omega$.

The NBA $\mathcal{B}_1 \cap \mathcal{B}_2$ produced using the algorithm *IntersNBA* is:



Solution 4 (4 points)
We note "state[discovery time/finishing time]" for dfs_1's exploration:
$\alpha[1/14], \gamma[2/13], \zeta[3/6], \eta[4/5], \varepsilon[7/12], \delta[8/11], \beta[9/10]$.

Procedure dfs_2 is called on ζ at time 6 of dfs_1's exploration, and on β at time 10. Exploration of dfs_2 on ζ: $\zeta[1/4], \eta[2/3]$; exploration of dfs_2 on β: $\beta[1/], \alpha[2/], \gamma[3/], \varepsilon[4/], \delta[5/]$ and then the algorithm answers NEMP.

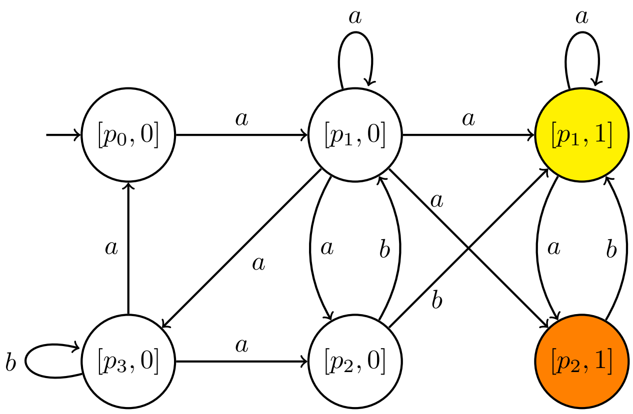
Solution 5 (4 points)

(a) F_1 gives an empty Muller automaton. First, note that there is no direct path from p_3 to p_1, that is, every path from p_3 to p_1 visits either p_0 or p_2. Therefore, every run that visits both states p_1 and p_3 infinitely often, also has to visit p_0 or p_2 infinitely often (or both). Since the Muller acceptance condition requires that the states visited infinitely often are exactly p_1 and p_3, we conclude that there is no accepting run, the NMA is empty.

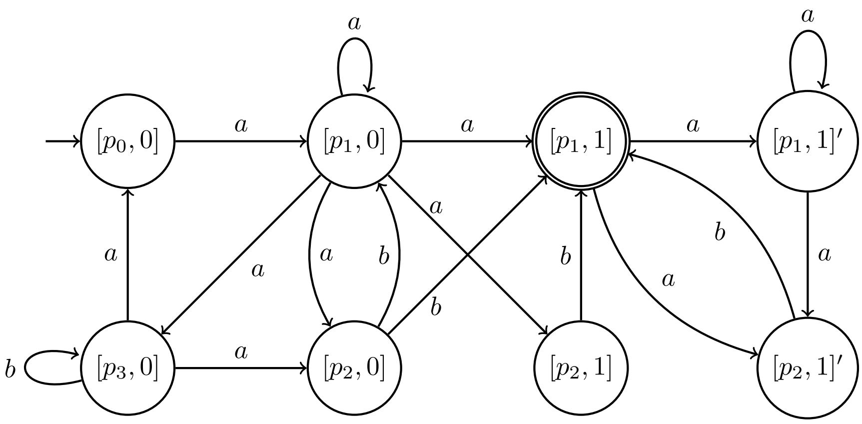
(Note that the only way to avoid visiting p_0 and p_2 infinitely often is that a run stays either in p_1 or in p_3, which is also not allowed, as we have to visit both p_1 and p_3 infinitely often.)

(b) F_1 gives a non-empty Muller automaton. Namely, a run of the ω-word $a(ab)^\omega$ is accepting as it visits infinitely often exactly p_1 and p_2.

First we transform the NMA into an equivalent NGA with acceptance condition $\{\{[p_1, 1]\}, \{[p_2, 1]\}\}$:



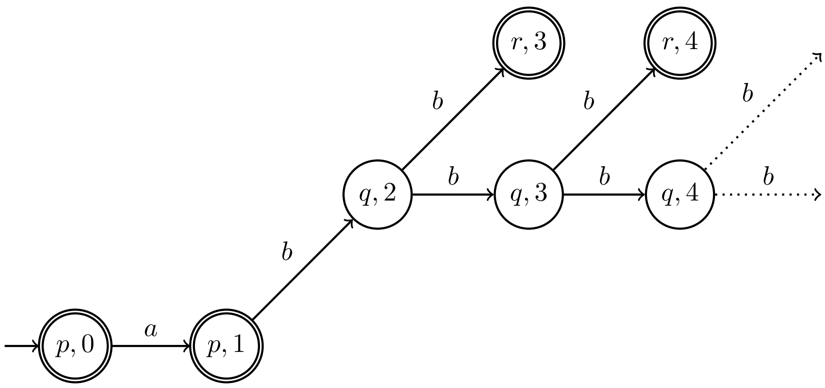
Now, we transform the NGA into an equivalent NBA:



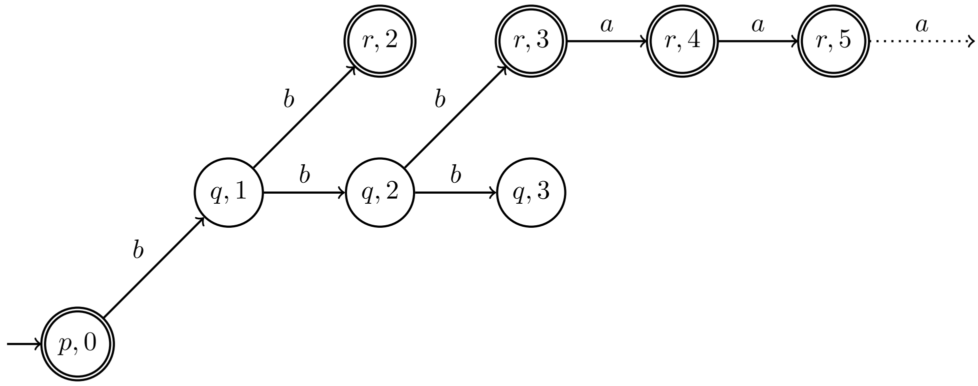
Note that some transitions are omitted for simplicity, and some more could be omitted without changing the recogniized language.

Solution 6 (5 points)

1. dag(ab^ω):



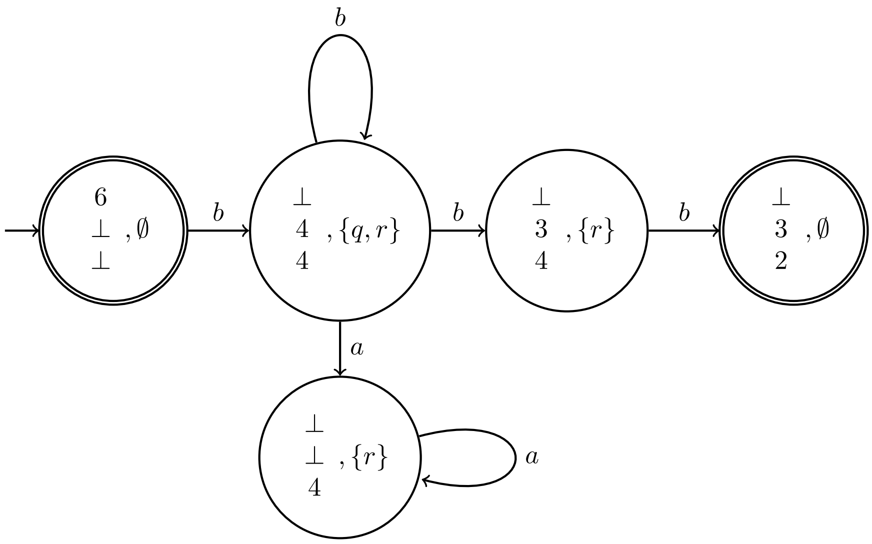
dag($bbba^\omega$):



2. Yes it does, for example 2 on the first two p-nodes, then 1 on the q nodes and 0 on the r-nodes.

   No it does not: the only infinite path is made of r-nodes which are final and thus must have even rank by definition of a rank.

3. An answer for the complement automaton:



Solution 7 (6 points)

(a) There are many solutions, for example (1) $\mathbf{F}(p \wedge q) \wedge \neg r$ or (2) $\mathbf{XX}r$ or (3) $p \vee \mathbf{FG}q$...

(b) $\mathbf{XF}(p \wedge q \wedge \neg r) \rightarrow (\mathbf{X}\neg(p \wedge q \wedge \neg r)\ \mathbf{U}\ r)$ or equivalently $\mathbf{XG}\neg(p \wedge q \wedge \neg r) \vee (\mathbf{X}\neg(p \wedge q \wedge \neg r)\ \mathbf{U}\ r)$

Frequent mistakes:

$\mathbf{F}(p \wedge q \wedge \neg r) \rightarrow (\neg(p \wedge q \wedge \neg r)\ \mathbf{U}\ r)$ is not correct since we have the requirement that $i \geq 1$, and we have to take care of the point 0 using the operator $\mathbf{X}$.

$\mathbf{F}(p \wedge q \wedge \neg r) \rightarrow (\mathbf{F}r\ \mathbf{U}\ (p \wedge q \wedge \neg r))$ is not correct since this formula just claims that r will appear eventually, but we have no guarantee that it will happen strictly before $p \wedge q \wedge \neg r$. Also, we have to take care of the $i = 0$ point as above.

$\mathbf{F}(p \wedge q \wedge \neg r) \rightarrow (r\ \mathbf{U}\ (p \wedge q \wedge \neg r))$ is not correct since it forces r to appear *everywhere* before $p \wedge q \wedge \neg r$, and we only need one point where r will appear. Also, we have to take care of the $i = 0$ point as above.

(c) $s = (\emptyset + \{p\} + \{q\} + \{p,q\})(\emptyset + \{p\} + \{q\} + \{p,q\})^*\{p,q\}\Sigma^\omega$, that is, $(\emptyset + \{p\} + \{q\} + \{p,q\})^+\{p,q\}\Sigma^\omega$.

Frequent mistakes:

$s = (\emptyset + \{p\} + \{q\} + \{p,q\})^*\{p,q\}\Sigma^\omega$. This ω-expression is not correct. For example, consider $\sigma = \{p,q\}\emptyset^\omega$. We have that $\sigma \in C$ so this computation should not be captured by the solution expression. Still, $\sigma \in L(s)$.

$s = (\emptyset + \{p\} + \{q\})^*\{p,q\}\Sigma^\omega$. This ω-expression is not correct. For example, consider $\sigma = \{p,q\}\{r\}\{p,q\}\emptyset^\omega$. We have that $\sigma \in C$ so this computation should not be captured by the solution expression. Still, $\sigma \in L(s)$.

Solution 8 (2 points)
Let A_n be an arbitrary NFA recognizing P_n. Let $w_1w_1^R, w_2w_2^R$ be two different palindromes of length $2n$. Since they are both accepted by A_n, there exist initial states q_{01}, q_{02}, states q_1, q_2, and final states q_{1f}, q_{2f} such that $q_{01} \xrightarrow{w_1} q_1 \xrightarrow{w_1^R} q_{f1}$ and $q_{02} \xrightarrow{w_2} q_2 \xrightarrow{w_2^R} q_{f2}$. We have $q_1 \neq q_2$, since otherwise A would accept $w_1w_2^R$, which is not a palindrome. Since there are exactly 2^n palindromes of length $2n$ (one for each word $w \in \{a,b\}^n$), and A has a different state for each of them, the automaton A has at least 2^n states.

For the first bonus points: Let w_1, w_2 be two different words of length $0 \leq \ell_1, \ell_2 \leq n-1$. There exist different words w_1', w_2' such that w_1w_1' and w_2w_2' are palindromes. Assume $q_{01} \xrightarrow{w_1} q_1 \xrightarrow{w_1'} q_{f1}$ and $q_{02} \xrightarrow{w_2} q_1 \xrightarrow{w_2'} q_{f2}$. We prove $q_1 \neq q_2$. Assume $q_1 = q_2$. Then the NFA accepts w_1w_2'. If $\ell_1 \neq \ell_2$ then w_1w_2' does not have length $2n$, and so it does not belong to P_n, contradiction. If $\ell_1 = \ell_2$ then w_1w_2' is not a palindrome, contradiction. Since there are $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ words of length up to $n-1$, the NFA has at least $2^n - 1$ additional states, on top of the 2^n above. So the NFA has at least $2^{n+1} - 1$ states.

For the second bonus points: Let w_1, w_2 be two different words of length $1 \leq \ell_1, \ell_2 < n$. There exist different words w_1', w_2' such that $w_1'w_1$ and $w_2'w_2$ (observe the order!) are palindromes. Using the same argument as above, we get $q_1 \neq q_2$. So we get at least $2^n - 1$ states for the words of length $0 \leq \ell \leq n-1$, at least 2^n states for the words of length n, and at least $2^n - 1$ state for the words of length $n+1 \leq \ell \leq 2n$. In total: $2(2^n - 1) + 2^n = 2^{n+1} + 2^n - 2$ states.