Exercise 2.3

In the lecture you have seen the definition of the Myhill-Nerode relation $\sim_L$ for a given language $L \subseteq \Sigma^*$:

For two words $x, y \in \Sigma^*$ we write $x \sim_L y$ iff $xv \in L \Leftrightarrow yv \in L$ holds for all $v \in \Sigma^*$.

(a) Determine the equivalence classes of $\sim_L$ w.r.t. the following languages over $\Sigma = \{a, b\}$:

- $L_1 := \mathcal{L}((ab + ba)^*)$,
- $L_2 := \mathcal{L}((aa)^*)$,
- $L_3 := \{w \in \{a, b\}^* \mid$ the number of occurrences of ab and ba in w is the same $\}$.
- $L_4 := \{a^n b^n c^n \mid n \geq 0\}$.

(b) In the definition of $\sim_L$ we compare two given words by appending all possible words. Instead of appending, we may also prepend. Consider therefore the binary relation $\sim^L$ on Σ^* defined by

For two words $x, y \in \Sigma^*$ we write $x \sim^L y$ iff $ux \in L \Leftrightarrow uy \in L$ holds for all $u \in \Sigma^*$.

- Determine the equivalence classes of $\sim^{L_1}$ and $\sim^{L_2}$.
- Show that L is regular iff $\sim^L$ has only finitely many equivalence classes.
- Is the number of equivalence classes of $\sim_L$ equal to the one of $\sim^L$?

(c) Finally, we may compare two words x, y also by appending and prepending arbitrary words, i.e., define the relation $\equiv_L$ as follows:

For two words $x, y \in \Sigma^*$ we write $x \equiv_L y$ iff $uxv \in L \Leftrightarrow uzv \in L$ holds for all $u, v \in \Sigma^*$.

For $x \in \Sigma^*$ let $[x]_L$ denote the equivalence class of x w.r.t. $\equiv_L$, i.e., $[x]_L = \{t \in \Sigma^* \mid x \equiv_L y\}$. We write $\Sigma^*/\equiv_L$ for the set of equivalence classes.

- How does $\equiv_L$ relate to $\sim_L$, resp. $\sim^L$?
- Determine the equivalence classes of $\equiv_{L_2}$.
- Show that the following multiplication on $\Sigma^*/\equiv_L$ is well-defined, associative and has $[\varepsilon]_L$ as its neutral element:

$$[w]_L \cdot [w']_L := [ww']_L.$$

Remark: $\Sigma^*/\equiv_L$ is called the syntactic monoid of L.

Solution:

(a) *Reminder*:

- For $L \subseteq \Sigma^*$, the relation $\sim_L$ on Σ^* was defined by

$$x \sim_L y \text{ iff } \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L.$$

- Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be a DFA. Set $P(q) = \{x \in \Sigma^* \mid \delta(q_I, x) = q\}$ for $q \in Q$.

Obviously, for $x, y \in P(q)$ we have for any word $v \in \Sigma^*$ that

$$\delta(q_{\text{initial}}, xv) = \delta(\delta(q_{\text{initial}}, x), v) = \delta(q, v) = \delta(\delta(q_{\text{initial}}, y), v) = \delta(q_{\text{initial}}, yv).$$

as the automaton is deterministic, i.e.,

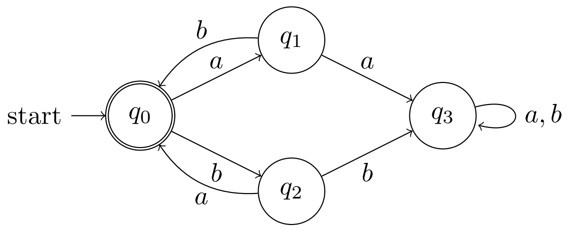
$$x, y \in P(q) \Rightarrow x \sim_L y.$$

Hence, the partition $\{P(q) \mid q \in Q\}$ is a refinement of $\Sigma^*/\sim_L$. So, if $\mathcal{A}$ is also minimal, then we even have

$$x, y \in P(q) \Leftrightarrow x \sim_L y.$$

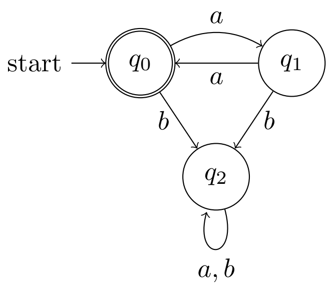
Hence, for calculating the equvialence classes of a regular language L it suffices to construct a minimal DFA for L and obtain from it the languages $P(q)$.

- L_1:



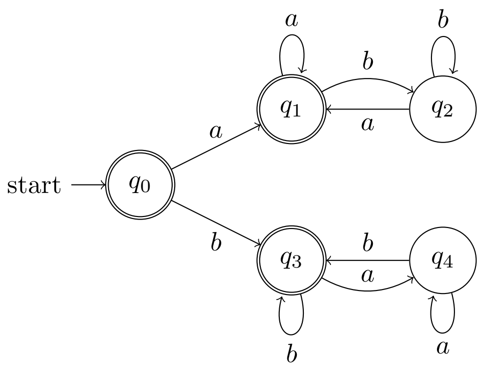
$P(q_0) = \mathcal{L}((ab + ba)^*)$, $P(q_1) = \mathcal{L}((ab + ba)^* a)$, $P(q_2) = \mathcal{L}((ab + ba)^* b)$ and $P(q_3) = \mathcal{L}((ab + ba)^*(aa + bb))$.

- L_2:



$P(q_0) = \mathcal{L}((aa)^*)$, $P(q_1) = \mathcal{L}((aa)^* a))$ and $P(q_2) = \mathcal{L}((aa)^*(b + ab))$.

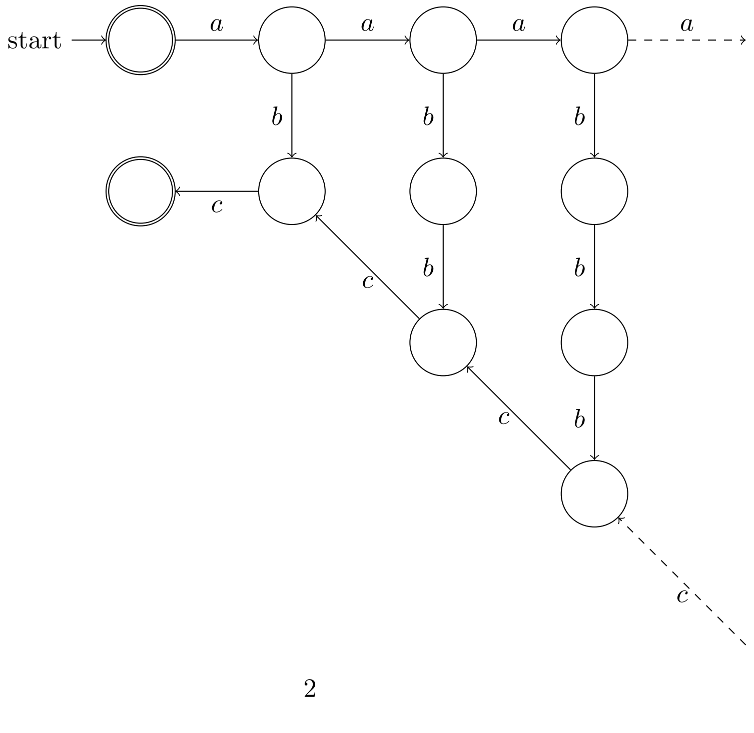
- L_3:



It's left to the reader to determine the equivalence classes from the DFA.

- L_4:

Although L_4 is context-sensitive, but neither context-free nor regular, the same idea as for the preceding languages can be used, albeit one has to consider an automaton with a infinite number of states (the rejecting state isn't shown here):



One easily checks that any two states are inequivalent. We obtain the equivalence classes:

$\{a^k\}$ for $k \geq 0$, $\{a^k b^l\}$ for $k > l \geq 1$, $\{a^{k+l}b^{k+l}c^l \mid l \geq 0\}$ for $k \geq 1$, and the complement of the union of these.

(b) For $w \in \Sigma^*$ let w^R be its reverse, i.e., the word we obtain when reading w from right to left, e.g., $(abb)^R = bba$. For $L \subseteq \Sigma^*$ we then set $L^R := \{w^R \mid w \in L\}$.

It is well-known that L is regular if and only if L^R is regular (exercise!).

Consider the definition of $\sim^L$ now:

$$\begin{array}{lll}
x \sim^L y & \text{iff} & \forall u \in \Sigma^* ux \in L \Leftrightarrow uy \in L \\
& \text{iff} & \forall u \in \Sigma^* (ux)^R \in L^R \Leftrightarrow (uy)^R \in L^R \\
& \text{iff} & \forall u \in \Sigma^* x^R u^R \in L^R \Leftrightarrow y^R u^R \in L^R \\
& \text{iff} & \forall v \in \Sigma^* x^R v \in L^R \Leftrightarrow y^R v \in L^R \\
& \text{iff} & \forall v \in \Sigma^* x^R v \in L^R \Leftrightarrow y^R v \in L^R \\
& \text{iff} & x^R \sim_{L^R} y^R.
\end{array}$$

The crucial point here is that u^R is any possible word as we consider all $u \in \Sigma^*$ which allows us to replace u^R and u by v.

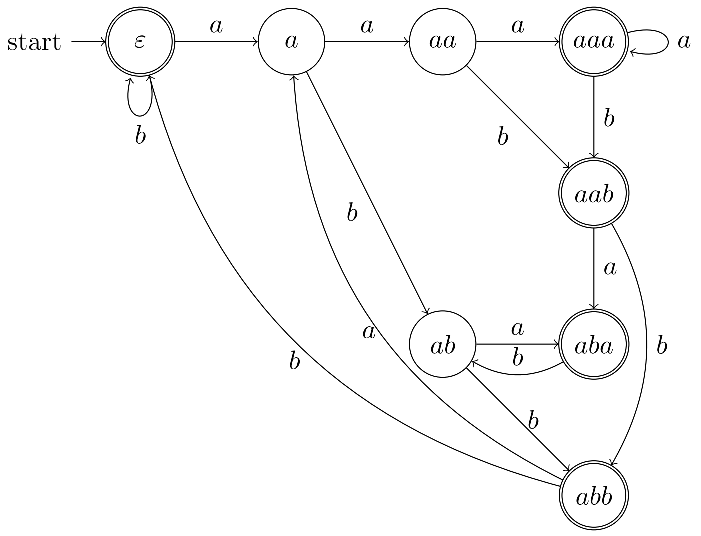
This means that we can obtain the equivalence classes of $\sim^L$ from those of $\sim_{L^R}$ by simply reversing the members of the equivalence classes, and we already know how to obtain the equivalence classes of $\sim_{L^R}$ (see (a)).

In both cases, L_1 and L_2, $L_i^R = L_i$, so we have already obtained the sought equivalence classes in (a).

It also immediately follows that $\sim^L$ is finite iff L is regular, as $|\sim^L| = |\sim_{L^R}|$ and we know that (i) L is regular iff L^R is regular, and (ii) L^R is regular iff $|\sim_{L^R}|$ is finite.

Finally, we show that in general $\sim_L$ and $\sim^L$ do not need to have the same number of classes: Let $L \subseteq \Sigma^*$ consist of those words whose third letter is an a. It is easy to see that a minimal DFA for L consists of $4+1$ states; four states are required to count how many letters have already been read, while the fifth state is the rejecting state.

On the other hand, L^R consists of all words whose third last letter is an a. A corresponding minimal DFA has to remember the last a seen within a window of three letters, leading to the equivalence classes $[\varepsilon]$, $[a]$, $[aa]$, $[aaa]$, $[aab]$, $[ab]$, $[aba]$, $[abb]$:



(c)
- Obviously, $x \equiv_L y$ implies both $x \sim_L y$ and $x \sim^L y$.

  But in general, $\equiv_L$ has more equivalence classes than $\sim_L$ or $\sim^L$. Consider for example L_1: We have both $\varepsilon \sim_{L_1} ab$ and $\varepsilon \sim^{L_1} ab$, but $\varepsilon \not\equiv_{L_1} ab$ as $a\varepsilon b \in L_1$, but $a(ab)b \notin L_1$. (See also exercise 3.2.)

- In the case of L_2 one easily checks $x \sim_{L_2} y$ iff $x \equiv_{L_2} y$

- It remains to show that the defined multiplication on $\Sigma^* / \equiv_L$ is independent of the representative taken from the equivalence class.

  Choose $x, y, x', y' \in \Sigma^*$ s.t. $x \equiv_L x'$ and $y \equiv_L y'$. We have to show that $xy \equiv_L x'y'$:

  – $x \equiv_L x'$ implies $xy \equiv_L x'y$. (Set $v = yv'$ in the definition with $v' \in \Sigma^*$).

  – $y \equiv_L y'$ implies $x'y \equiv_L x'y'$. (Set $u = u'x$ in the definition with $u' \in \Sigma^*$.)

  – As $\equiv_L$ is a equivalence relation, it is transitive, and we obtain $xy \equiv_L x'y'$.

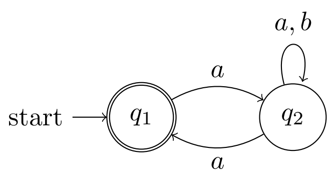
Exercise 3.1

Let $\mathcal{A} = (Q, \Sigma, \delta, q_{\text{initial}}, F)$ be some NFA. We assume that $Q = \{q_1, q_2, \ldots, q_n\}$. With every word $w \in \Sigma^*$ we then associate the boolean matrix $M_w \in \{0,1\}^{n \times n}$ with

$$(M_w)_{i,j} = 1 \text{ iff } \mathcal{A} \text{ can end up in state } q_j \text{ when reading the word } w \text{ starting from state } q_i.$$

The set $\mathcal{T}_\mathcal{A} := \{M_w \mid w \in \Sigma^*\}$ is then called the *transition monoid* of $\mathcal{A}$.

Example: Consider the following FA:



For this automaton, the matrices M_a, M_b, M_{aa} and M_{ab} are:

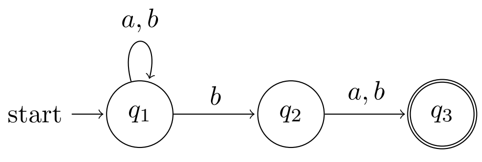
$$M_a = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \; M_b = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \; M_{aa} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \text{ and } M_{ab} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

(a) Assuming standard multiplication of boolean matrices, we have in our example that $M_{aa} = M_a \cdot M_a$ and $M_{ab} = M_a \cdot M_b$.

- Show that $M_u \cdot M_v = M_{uv}$ holds for all $u, v \in \Sigma^*$ for any given NFA $\mathcal{A}$.
- Check that $\mathcal{T}_\mathcal{A}$ w.r.t. this multiplication is indeed a monoid.

  Reminder:
  $\langle S, \cdot \rangle$ is a monoid if (i) $\forall a, b \in S : a \cdot b \in S$, (ii) $\forall a, b, c \in S : a \cdot (b \cdot c) = (a \cdot b) \cdot c$, and (iii) $\exists 1 \in S \forall a \in S : a \cdot 1 = a = 1 \cdot a$.

(b) Consider the following FA $\mathcal{A}$:



- Draw the labeled graph with states the elements of $\mathcal{T}_\mathcal{A}$ and edges $M_u \xrightarrow{a} M_{ua}$ for $a \in \Sigma, u \in \Sigma^*$.

  Note that $\mathcal{T}_\mathcal{A}$ has at most 2^9 elements; construct the graph on the fly starting from M_ε. You should end up with 7 elements/states.
- How can you obtain from this graph a deterministic finite automaton accepting the same language as the original nondeterministic automaton? How does this construction relate to the determinization procedure you have seen in the lecture?

Solution:

(a) By definition of $\mathcal{T}_\mathcal{A}$, for any $M \in \mathcal{T}_\mathcal{A}$ we find a u such that $M_u = M$. It therefore suffices to consider matrices M_u, M_v, M_w with $u, v, w \in \Sigma^*$.

Notation: We also write δ for its extension to words (sometimes also explicitly denoted by $\hat{\delta}$). We then may write the definition of M_w also as follows:

$$(M_w)_{i,j} = \begin{cases} 1 & \text{if } q_j \in \delta(q_i, w) \\ 0 & \text{else} \end{cases}$$

One then easily checks then that for any two words $u, v \in \Sigma^*$ we have $M_u \cdot M_v = M_{uv}$ $(*)$:

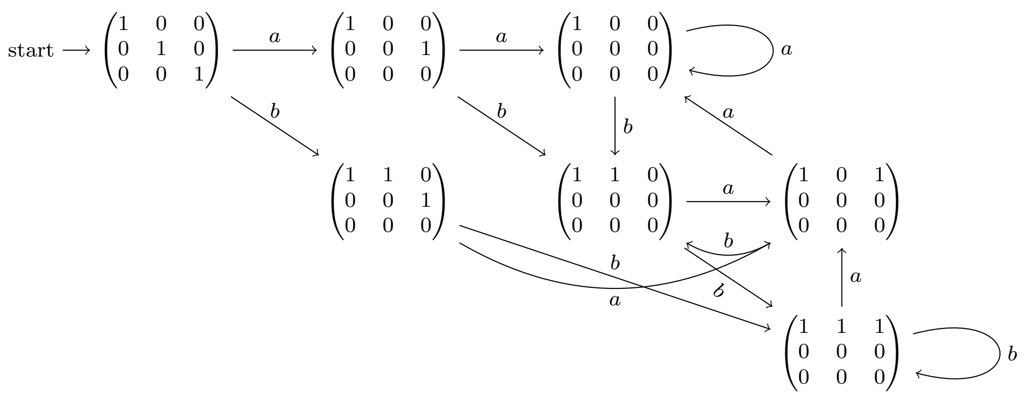
$$\begin{array}{rll} (M_u \cdot M_v)_{i,j} = 1 & \text{iff} & \exists k \in [n] \,:\, (M_u)_{i,k} = 1 \wedge (M_v)_{k,j} = 1 \\ & \text{iff} & \exists k \in [n] \,:\, \delta(q_i, u) \ni q_k \wedge \delta(q_k, v) \ni q_j \\ & \text{iff} & \delta(q_i, uv) \ni q_j \\ & \text{iff} & (M_{uv})_{i,j} = 1. \end{array}$$

By $(*)$ we therefore have immediately that

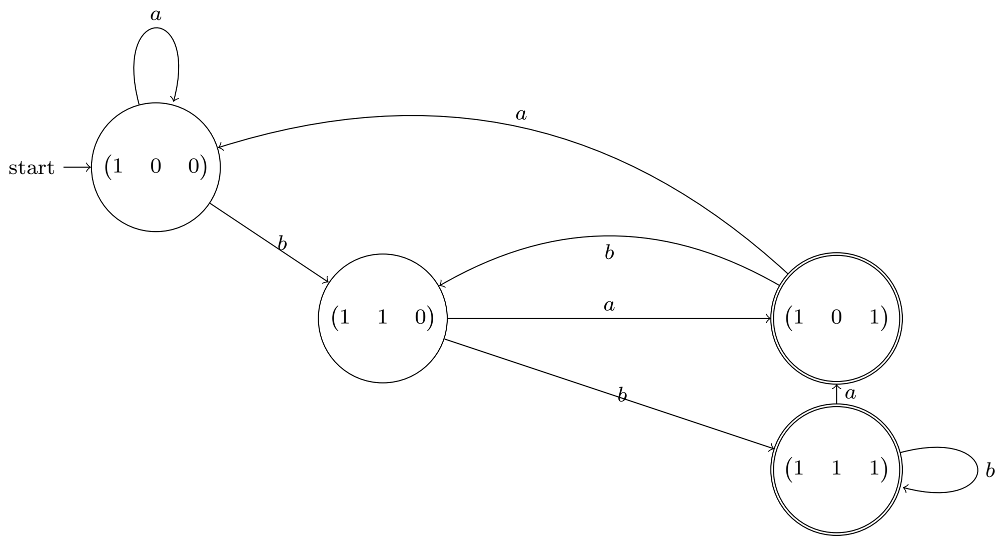
$$M_u \cdot M_v = M_{uv} \in \mathcal{T}_\mathcal{A} \text{ and, in particular, } M_\varepsilon \cdot M_u = M_u = M_u \cdot M_\varepsilon.$$

That is, $\mathcal{T}_\mathcal{A}$ is closed w.r.t. the multiplication of boolean matrices and M_ε is a neutral element. Associativity also follows easily, as matrix multiplication is associativ.

(b) •



• In the determinization procedure we only remember which states have been reached starting from the initial state. We therefore simply have to multiply (from the left) the states of above graph by the (row) vector encoding the initial state(s) (this also generalizes to FAs with multiple initial states). In our example, this vector is $(1,0,0)$, yielding (after unifying states):



In order to determine the final states, we similarly have to multiply (from the right) with the (column) vector encoding the final states (here: $(0,0,1)^\top$).

Exercise 3.2

In the preceding exercise, the transition monoid $\mathcal{T}_\mathcal{A}$ of a finite automaton $\mathcal{A}$ has been introduced. Recall also the syntactic monoid $\mathcal{S}_L := \Sigma^*/\equiv_L$ of a language $L \subseteq \Sigma^*$ which was the set of equivalence classes w.r.t. the binary relation $\equiv_L$ on Σ^* defined by

$$x \equiv_L y \text{ iff } \forall u, v \in \Sigma^* \,:\, uxv \in L \Leftrightarrow uyv \in L.$$

(a) Let $\mathcal{A} = (Q, \Sigma, \delta, q_i, F)$ be an NFA. Show that for $x, y \in \Sigma^*$ we have

$$M_x = M_y \Rightarrow x \equiv_L y.$$

(b) Let L be a regular language over Σ. Let $\mathcal{A}_L$ be a minimal DFA with $L = \mathcal{L}(\mathcal{A})$. Show that for $x, y \in \Sigma^*$ it holds that

$$x \equiv_L y \Rightarrow M_x = M_y.$$

Is it necessary for $\mathcal{A}$ to be deterministic or minimal?

(c) Calculate the size of the syntatic monoid of the following languages:

- $\mathcal{L}((aa)^*)$ for $\Sigma = \{a\}$.
- $\mathcal{L}((ab + ba)^*)$ for $\Sigma = \{a, b\}$.

You can find some incomplete C++-code on the webpage for an easier calculation of the transition monoid.

(d) A famous result by Schützenberger says that a regular language L is representable by a star-free expression if and only if its syntactic monoid $\mathcal{S}_L$ is aperiodic. (A monoid $\langle M, \cdot, 1\rangle$ is aperiodic if for any $a \in M$ there is a natural number n such that $a^n = a^{n+1}$.)

- Show that syntactic monoid of a regular language L is isomorphic to the transition monoid of a minimal DFA for L, i.e., show:
$$\forall x, y, z \in \Sigma^* \ : \ [x]_L \cdot [y]_L = [z]_L \text{ iff } M_x \cdot M_y = M_z.$$
- Decide for the two languages considered in (c) whether they are representable by star-free expressions. Use a computer.

Solution:

(a) Assume $M_x = M_y$. Let $u, v \in \Sigma^*$ be arbitrary words. Further, let v_I be the row vector which encodes the initial states of $\mathcal{A}$. Similarly, let v_F be the column vector encoding the final states. It then holds:

$$\begin{aligned} uxv \in L \quad & \text{iff} \quad v_I M_{uxv} v_F = 1 \\ & \text{iff} \quad v_I M_u M_x M_v v_F = 1 \\ & \text{iff} \quad v_I M_u M_y M_v v_F = 1 \qquad \text{(as } M_x = M_y) \\ & \text{iff} \quad v_I M_{uyv} v_F = 1 \\ & \text{iff} \quad uyv \in F. \end{aligned}$$

So, $M_x = M_y \Rightarrow x \equiv_L y$ indeed holds.

(b) Assume $x \equiv_L y$ and $M_x \neq M_y$. We then find i, j such that $(M_x)_{i,j} \neq (M_y)_{i,j}$. W.l.o.g. we may assume that $(M_x)_{i,j} = 1$ (otherwise swap x and y), i.e., $\delta(q_i, x) = q_j \neq \delta(q_i, y)$. As $\mathcal{A}$ is deterministic, $\delta(q_i, y)$ is defined, hence, set $q_k := \delta(q_i, y)$.

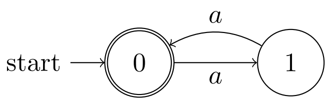
By minimality of $\mathcal{A}$, any state is reachable from the initial state, so there is a word $u \in \Sigma^*$ s.t. $\delta(q_{\text{initial}}, u) = q_i$. We now have $ux \in P(q_i)$ and $uy \in P(q_k)$, and, again by minimality of $\mathcal{A}$, we conclude $ux \not\sim_L uy$ (see ex.2.3), and subsequently $ux \not\equiv_L uy$ which contradicts our assumption.

(c) From (a) and (b) it follows:

Let $\mathcal{A}$ be a minimal DFA for $L \subseteq \Sigma^*$. Then $x \equiv_L y$ iff $M_x = M_y$.

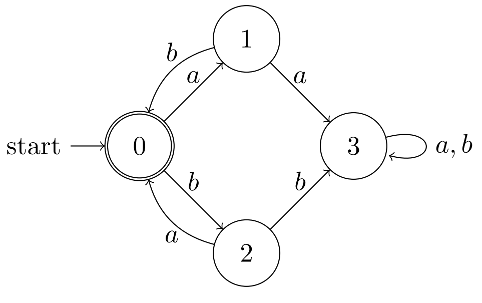
This means that the transition monoid of a minimal DFA for L has the same number of elements as $\Sigma^*/\equiv_L$. It therefore suffices to determine the size of $\mathcal{T}_\mathcal{A}$ in order to determine $|\Sigma^*/\equiv_L|$:

- The minimal DFA for $(aa)^*$ is drawn below:



This yields the transition matrix
$$M_a = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$
One easily checks that $M_a^2 = \text{Id} = M_\varepsilon$. So, the syntactic monoid consists of the equivalence classes $[\varepsilon] = \mathcal{L}((aa)^*)$ and $[a] = \mathcal{L}(a(aa)^*)$.

- The minimal DFA for $(ab + ba)^*$:



This yields the matrices:
$$M_a = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } M_b = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
Using the supplied code, one can check that the transition monoid consists of 15 elements. By (a) and (b) the transition monoid of the minimal DFA has the same size as the syntactic monoid.

(d)

- The multiplication on $\mathcal{S}_L$ was defined by
$$[x]_L \cdot [y]_L := [xy]_L.$$
In exercise 2.3 it was shown that this multiplication is well-defined. So
$$\begin{aligned} [x]_L \cdot [y]_L = [z]_L \quad &\text{iff} \quad xy \equiv_L z \\ &\text{iff} \quad M_{xy} = M_z \qquad \text{(cf. (a) and (b))} \\ &\text{iff} \quad M_x \cdot M_y = M_z \end{aligned}$$
Therefore we have
$$\begin{aligned} [x]_L^n = [x]_L^{n+1} \quad &\text{iff} \quad [x^n]_L = [x^{n+1}]_L \\ &\text{iff} \quad M_{x^n} = M_{x^{n+1}} \\ &\text{iff} \quad (M_x)^n = (M_x)^{n+1}. \end{aligned}$$
It therefore suffices to check that the transition monoid of a minimal DFA for L is aperiodic.

- For $(aa)^*$ we have already seen that the transition monoid of the minimal DFA is periodic as $M_a^{2k} = M_\varepsilon, M_a^{2k+1} = M_a$ for all $k \in \mathbb{N}_0$. As the syntactic monoid is isomorphic to this transition monoid, $(aa)^*$ is not representable by a star-free expression.

  For $(ab + ba)^*)$ one easily extends the supplied code to also check that the transition, and therefore syntactic monoid are aperiodic. Thus a star-free expression exists for $\mathcal{L}((ab + ba)^*)$.