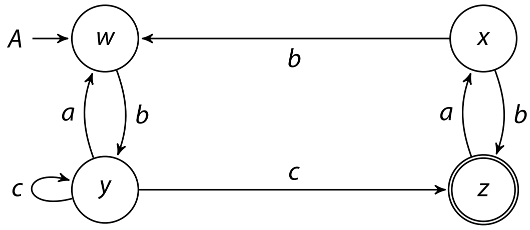
Theoretical Computer Science
Exercise Sheet 4

Prof. Dr. Roland Meyer
René Maseli
TU Braunschweig
Winter Semester 2025/26

Release: 2025-12-08 Due: 2025-12-18 23:45

Homework Exercise 4.1: Homomorphisms [3 points]

Examine the following NFA A over the alphabet $\Sigma = \{a, b, c\}$, and homomorphisms h, g with



$h : \Sigma \to \{0, 1\}$

$h(a) = \varepsilon \quad h(b) = 10 \quad h(c) = 01$

$g : \{d, e, f\} \to \Sigma$

$g(d) = bab \quad g(e) = cca \quad g(f) = \varepsilon$.

a) [1 point] Show $1001011010100110 \in h(\mathcal{L}(A))$ and $dfedf \in g^{-1}(\mathcal{L}(A))$ by giving corresponding runs through A.

b) [1 point] Construct the image-automaton $h(A)$ with $\mathcal{L}(h(A)) = h(\mathcal{L}(A))$.

c) [1 point] Construct the co-image-automaton $g^{-1}(A)$ with $\mathcal{L}(g^{-1}(A)) = g^{-1}(\mathcal{L}(A))$.

Homework Exercise 4.2: Theorem of Myhill & Nerode [6 points]

Let $L \subseteq \Sigma^*$ and $\equiv_L$ be the Nerode right-congruence, known from the lecture, with

$$u \equiv_L v \quad \text{iff} \quad \forall w \in \Sigma^*\colon u.w \in L \Leftrightarrow v.w \in L\,.$$

a) [2 points] Prove that $\equiv_L$ is indeed an equivalence relation and a right-congruence. The latter means, that for all u, v with $u \equiv_L v$ and all $x \in \Sigma^*$ it holds that: $u.x \equiv_L v.x$.

Let $L \subseteq \Sigma^*$ be a regular language with $\text{Index}(\equiv_L) = k \in \mathbb{N}$ and let $A = \langle Q, q_0, \to, Q_F\rangle$ be a DFA with $L = \mathcal{L}(A)$ and $|Q| = k$. Let further $A_L = \langle Q_L, q_{0L}, \to_L, Q_{FL}\rangle$ be the equivalence automaton for L with $\mathcal{L}(A_L) = L$ and $u_1, \ldots, u_k$ be the representants of the equivalence classes of $\equiv_L$.
Show Theorem 6.11 from the script: A and A_L are isomorphic. The isomorphism $\beta : Q_L \to Q$ is defined as: $\beta([u_i]_{\equiv_L}) = q$ with $q_0 \xrightarrow{u_i} q$ in A.

b) [1 point] Consider the equivalence relation $\equiv_A$. Show that $\text{Index}(\equiv_A) = \text{Index}(\equiv_L)$ holds. With the result $\equiv_A \subseteq \equiv_L$ from the lecture, this implies $\equiv_A = \equiv_L$.

c) [1 point] Show that β is well-defined.

Hint: The function β was defined on equivalence classes. You have to show, that β is independent of the choice of the representant $u_1, \ldots, u_k$. Let us assume $\hat{u}_i \equiv_L u_i$ and show that $\beta([\hat{u}_i]_{\equiv_L}) = \beta([u_i]_{\equiv_L})$ holds.

d) [1 point] Show that β is a bijection between Q_L and Q.

e) [1 point] Show that β is isomorphic. It remains to show, that $\beta(q_{0L}) = q_0$, $\beta(Q_{FL}) = Q_F$ and for all $p, p' \in Q_L$ and $a \in \Sigma$ the property $p \xrightarrow{a}_L p'$ iff $\beta(p) \xrightarrow{a} \beta(p')$ holds.

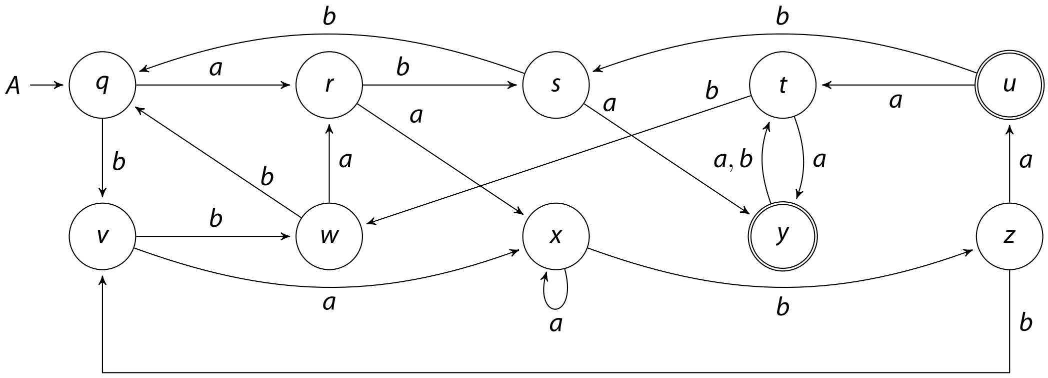
Homework Exercise 4.3: Nerode's right-congruence with non-regular languages [3 points]
Let $\Sigma = \{a, b\}$ be an alphabet. Consider $L = \{ a^n b a^m \mid n, m \in \mathbb{N}, n \geq m \}$. Prove that

$$[a^n]_{\equiv_L} = \{a^n\} \text{ for all } n \in \mathbb{N}$$
$$[a^n.b]_{\equiv_L} = \left\{ a^{n+\ell}.b.a^{\ell} \;\middle|\; \ell \in \mathbb{N} \right\} \text{ for all } n \in \mathbb{N}$$

holds. With infinite congruence classes, L is not regular by the Theorem of Myhill & Nerode.

Find all remaining equivalence classes with respect to $\equiv_L$.

Homework Exercise 4.4: Table-filling algorithm [3 points]
Consider the following DFA A.



a) [2 points] Use the table-filling algorithm to find the minimal DFA A_{min} with $\mathcal{L}(A_{\text{min}}) = \mathcal{L}(A)$. Draw the state chart of A_{min}.

Hint: Fill the cells with the number of iteration, where you first distinguished the state pair.

b) [1 point] List all equivalence classes of the Nerode-right-congruence of $\mathcal{L}(A)$ with at least one representant, each.

Homework Exercise 4.5: Pumping lemma for regular languages [3 points]
Consider $\Sigma = \{a, b\}$. By using the Pumping Lemma, prove that the following languages are not regular.

a) $L_1 = \{ w \in \{a, b\}^* \mid w \text{ enthält genau 3 a's mehr als b's} \}$

b) $L_2 = \{ a^n b^m \mid n < 42 \text{ or } m < n \}$

c) $L_3 = \{ w \in \{a, b\}^* \mid w \text{ enthält nicht genau so viele a's wie b's} \}$

Hint for c): Consider the following: For any given number $n \in \mathbb{N}$, which number is divisible by all numbers $\leq n$?

Definition: Finite-state Transducer: A finite-state transducer over a finite input alphabet Σ and a finite output alphabet Γ is formally a quadruple $T = \langle Q, q_0, \rightarrow, Q_F \rangle$ consisting of

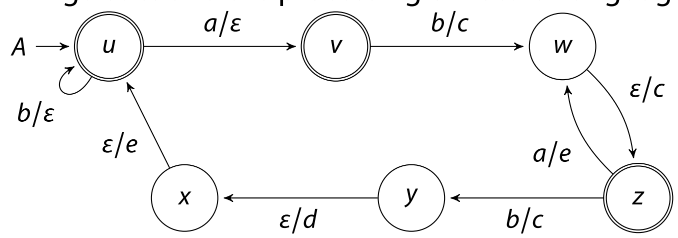
1. a finite set of states Q,
2. an initial state $q_0 \in Q$,
3. a transition relation $\rightarrow \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$,
4. and a set of accepting states $Q_F \subseteq Q$.

In the following we fix notation and important definitions:

1. $\langle p, a, x, q \rangle \in \rightarrow$ is denoted by $p \xrightarrow{a/x} q$. When reading an a in state p, the transducer transitions to state q and outputs x. Intuitively, a=ε denotes a spontaneous transition, while x=ε denotes a transition without output.
2. A computation $q_0 \xrightarrow{w_1/o_1} q_1 \xrightarrow{w_2/o_2} \cdots \xrightarrow{w_{n-1}/o_{n-1}} q_{n-1} \xrightarrow{w_n/o_n} q_n$ can also be denoted by $q_0 \xrightarrow{w/o} q_n$, where $w \in \Sigma^*$ and $o \in \Gamma^*$ are the respective concatenations without ε.
3. We define for any language $L \subseteq \Sigma^*$ the translation under T as $T(L) = \{\, o \in \Gamma^* \mid \exists w \in L, q_f \in Q_F : q_0 \xrightarrow{w/o} q_f \,\}$.

Tutorial Exercise 4.6:

A transducer can be thought of as an NFA with spontaneous transitions, which not only accepts input words but also outputs new words. It translates input words from Σ^* to output words in Γ^*. Transducers are used in linguistics and the processing of natural languages.



a) Consider the above transducer A over input alphabet $\Sigma = \{a, b\}$ and output alphabet $\Gamma = \{c, d, e\}$. Give regular expressions for $A(\Sigma^*)$, $A((ab)^*)$, $A(a^*b^*)$ and $A((abbab^*)^*)$.

b) Construct a transducer T that for any given word $w \in \{a, b, c\}^*$ works a follows: it prepends a b to every occurence of a and removes every second occurrence of c. Give a regular expression for $T((ac)^*)$. A proof of correctness is not needed.

c) Construct a transducer U that for any given word $w \in \{a, b\}^*$ works as follows: it removes every occurrence of the subsequence aba and for any b that is not part of such sequence, it can add an arbitrary amount of additional c's. Give a regular expression for $U(a^+(ba)^*)$. A proof of correctness is not needed.

d) We call a transducer deterministic if in any state and for any input, the transducer has exactly one possible, and hence unique, transition; this transition may be spontaneous. For example, a state with an a-labeled transition may not have another a-labeled transition nor another spontaneous transition, because in either case there would be two possible transitions on a. Show that it is **not** possible to determinize transducers in general. That means, there are transducers T which do not have any equivalent deterministic transducer T^{det} such that $T(L) = T^{\text{det}}(L)$ for all languages $L \subseteq \Sigma^*$.

Tutorial Exercise 4.7:

Prove that any class of languages is closed under translations of transducers, if and only if it is closed under intersection with regular languages, images and co-images of homomorphisms.

a) Let $h : \Sigma^* \to \Gamma^*$ be an arbitrary homomorphism between words. Construct a transducer T_h such that $T_h(L) = h(L)$ holds for all languages $L \subseteq \Sigma^*$. Prove the correctness of your construction.

b) Now prove that there is also a transducer $T_{h^{-1}}$ such that $T_{h^{-1}}(L) = h^{-1}(L)$ holds for all $L \subseteq \Gamma^*$. Prove the correctness of your construction.

c) Show that for any regular language M, there is a transducer T_M with $T_M(L) = L \cap M$.

d) Now show that the translation under any transducer T can be expressed as a combination of the three operations mentioned above.

Tutorial Exercise 4.8:

Let $\equiv \subseteq \Sigma^* \times \Sigma^*$ be an equivalence relation on words. As usual, we write $u \equiv v$ (instead of $\langle u, v\rangle \in \equiv$) to express that u and v are equivalent with respect to $\equiv$. Prove formally the following basic properties about equivalence relations:

- Every word is contained in its own equivalence class: $u \in [u]_\equiv$.
- The equivalence classes of equivalent words are equal: $u \equiv v \implies [u]_\equiv = [v]_\equiv$.
- The equivalence classes of non-equivalent words are disjoint: $u \not\equiv v \implies [u]_\equiv \cap [v]_\equiv = \emptyset$.

Tutorial Exercise 4.9:

Let $\Sigma = \{a, b\}$ be an alphabet. Consider the language $L = \Sigma^*.\{aab, abb\}.\Sigma^*$. Find all equivalence classes of $\equiv_L$ and construct the equivalence class automaton A_L .

Tutorial Exercise 4.10:

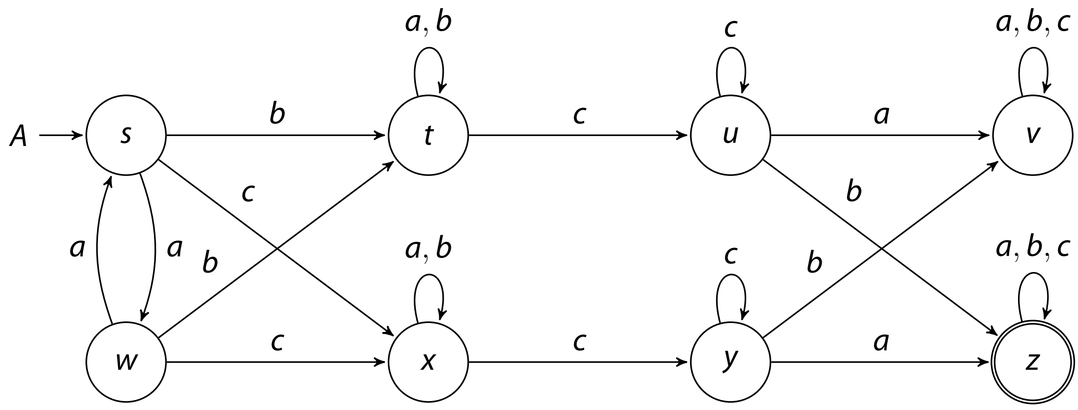
Here we want to show that some languages that admit a description by small NFAs do not admit a description by small DFAs; every DFA for that language is necessarily large.
Given $\Sigma = \{a, b\}$. Consider for all numbers $k \in \mathbb{N}, k > 0$ the language $L_k = \Sigma^*.a.\Sigma^{k-1}$ of words, that have an a at the k-th last position.

a) Show how to construct for any $k \in \mathbb{N}, k > 0$ an NFA $A_k = \langle Q_k, q_0, \to, F_k\rangle$ with $\mathcal{L}(A_k) = L_k$ and $|Q_k| = k + 1$. Give the automaton formally as a tuple.
You do not have to show correctness of your construction.

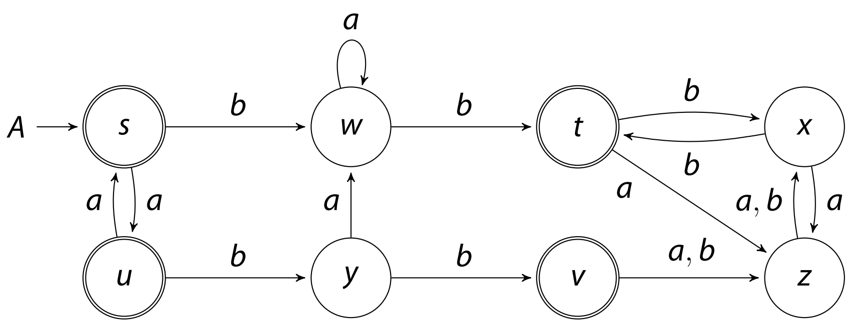
b) Now draw A_3 and its determinization $\mathcal{P}(A_3)$ via Rabin-Scott-power set construction.

c) Show for all $k \in \mathbb{N}, k > 0$ and $u, v \in \Sigma^k$ with $u \neq v$ the proposition $u \not\equiv_{L_k} v$.
What can you derive for the size of all DFAs for L_k?

Tutorial Exercise 4.11:
Consider the following DFA A. Find its Equivalence-Class-Automaton $A_{\mathcal{L}(A)}$ by Myhill & Nerode by using the Table-Filling algorithm and state all equivalence classes of Nerode's right-congruence.



Tutorial Exercise 4.12:
Consider the following DFA A.



Show that A is minimal, by using the table-filling algorithm. Fill cells with 0, if the respective state pair is initially separated, and with the number of the iteration, where that pair is separated for the first time.

Hint: While filling your table, note down in which order you separated a state class, e.g. initially, we separate accepting states from the rest: $\{s, t, u, v\} \not\sim_A \{w, x, y, z\}$, which allows us to separate $\{s, u\} \not\sim_A \{t, v\}$ in iteration 1, etc.

Tutorial Exercise 4.13:
Show, that the following languages are not regular, by using the Pumping Lemma.

a) $L_0 := \{ w \in \{a, b\}^* \mid |w|_a \geq |w|_b \}$

b) $L_1 := \{ w \in \{a, b\}^* \mid |w|_a \leq |w|_b \text{ oder } 2|w|_b \leq |w|_a \}$

c) $L_2 := \{ a^n b^m \mid n, m \in \mathbb{N} \text{ und } (n \neq 1 \text{ oder } \exists \ell \in \mathbb{N}\colon m = \ell^2) \}$.