Theoretical Computer Science 1
Exercise Sheet 4

René Maseli — TU Braunschweig
Prof. Dr. Roland Meyer — Winter semester 2021/22

Release: 21.12.2021 — Due: 13.01.2022, 23:59

Hand in your solutions per e-mail or StudIP to your tutor until Thursday, 13.01.2022 23:59 pm. You should provide your solutions either directly as .pdf file or as a readable scan/photo of your handwritten notes.

Definition: Finite-state Transducer
A finite-state transducer is formally a 6-tuple $T = \langle Q, \Sigma, \Gamma, \to, q_0, Q_F \rangle$ consisting of

1. a finite set of states Q,
2. finite alphabets Σ and Γ,
3. a transition relation $\to \subseteq Q \times (\Sigma \cup \{\tau\}) \times (\Gamma \cup \{\tau\}) \times Q$,
4. inital state q_0 and set of final states Q_F.

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.

In the following we fix notation and important definitions:

1. $(q, a, x, q') \in \to$ is denoted by $q \xrightarrow{a/x} q'$. When reading an a in state q, the transducer transitions to state q' and outputs x. Intuitively, $a = \tau$ denotes a spontaneous transition, while $x = \tau$ denotes a transition without output.
2. $\to^* \subseteq Q \times (\Sigma \cup \{\tau\})^* \times (\Gamma \cup \{\tau\})^* \times Q$ denotes the reflexive, transitive closure of $\to$. It satisfies $q \xrightarrow{\varepsilon/\varepsilon}{}^* q$ and $q \xrightarrow{w/o}{}^* q_n \iff \exists q_1, \ldots, q_{n-1} : q \xrightarrow{w_1/o_1} q_1 \xrightarrow{w_2/o_2} \ldots \xrightarrow{w_{n-1}/o_{n-1}} q_{n-1} \xrightarrow{w_n/o_n} q_n$.
3. T induces a relation $[T] \subset \Sigma^* \times \Gamma^*$ as follows:

   $$w[T]o \iff \exists w' \in (\Sigma \cup \{\tau\})^*, o' \in (\Gamma \cup \{\tau\})^* : q_0 \xrightarrow{w'/o'}{}^* q_f \in Q_F \quad \text{and} \quad \pi_\Sigma(w') = w, \pi_\Gamma(o') = o$$

   Hereby, $\pi_\Sigma : (\Sigma \cup \{\tau\}) \to \Sigma^*$ with $\pi_\Sigma(\tau) = \varepsilon$ and $\forall a \in \Sigma : \pi_\Sigma(a) = a$ induces a homomorphism, which deletes τ from a word. τ denotes either spontaneous transitions or empty output and hence should not be visible.

   We say that $o \in \Gamma^*$ is an output of T on $w \in \Sigma^*$.
4. T does not only transduce single words, but whole languages. We define for any language $L \subseteq \Sigma^*$ the translation under T as $T(L) = \{o \in \Gamma^* \mid \exists w \in L : w[T]o\} \subseteq \Gamma^*$.

Exercise 1: Finite transducers [11 points]

a) [3 points] Construct a transducer T that for any given word $w \in \{a, b\}^*$ works a follows: it removes every second occurrence of a and appends c to every occurrence of b. Give a regular expression for $T((ab)^*)$.

   A proof of correctness is not needed.

b) [5 points] 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.

c) [3 points] We call a transducer deterministic if in any state and for any input, the transducer has **at most 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 $T(L) = T^{det}(L)$ for all languages $L \in \Sigma^*$.

Exercise 2: (Bonus) Operations expressible by transducers [14 points]

a) [3 points] 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 $L \in \Sigma^*$. Prove the correctness of your construction.

b) [3 points] 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) [2 points] Show that for any regular language M, there is a transducer T_M with $T_M(L) = L \cap M$.

   Remark: In this exercise you have shown that transducers are capable of representing many typical operations on languages. If a class of languages is closed under translations of transducers, then it follows directly that it is also closed under the above mentioned operations.

d) [6 points] Now show that the converse also holds true, i.e. a class of languages that is closed under those three mentioned operations is also closed under translations of transducers.

   Remark: You have to show that for any transducer T, you can express the translation of a language L under T in terms of those three operations.

Exercise 3: Costs of Determinization [7 points]

In this exercise 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.

For a number $k \in \mathbb{N}, k > 0$ let $L_{a@k} = \Sigma^*.a.\Sigma^{k-1}$ be the language of words over $\Sigma = \{a, b\}$ that have an a at the k-th last position.

a) [2 points] Show how to construct for any $k \in \mathbb{N}, k > 0$ an NFA $A_k = \langle Q, q_0, \to, Q_F \rangle$ with $\mathcal{L}(A_k) = L_{a@k}$. Give the automaton formally as a tuple.

   You do not have to show correctness of your construction.

   How many states does A_k have?

b) [2 points] Now draw A_3 and its determinization A_3^{det} via Rabin-Scott-power set construction.

   Compare the number of states of A_3 and A_3^{det}.

c) [3 points] Let $k \in \mathbb{N}, k > 0$ be arbitrary. Prove that for $L_{a@k}$ there is no DFA B with less than 2^k many states such that $\mathcal{L}(B) = L_{a@k}$ holds.

   Hint: Proceed as follows:

   1. Assume there is a DFA $B = (Q', q_0', \to', Q_F')$ with $\mathcal{L}(B) = L_{a@k}$ and $|Q'| < 2^k$.
   2. Consider the set Σ^k of words of length k. How many such words are there?
   3. Now consider to each word $w \in \Sigma^k$ the (unique) state q_w in the DFA B after it read the word w.
   4. Now derive a contradiction.

Exercise 4: Equivalence relations [7 points]

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 $\equiv$.

a) [2 points] 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: $u \not\equiv v \implies [u]_\equiv \cap [v]_\equiv = \varnothing$.

b) [2 points] Let $L \subseteq \Sigma^*$ and $\equiv_L$ be the Nerode right-congruence, known from the lecture, with

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

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: $u.x \equiv_L v.x$.

c) [2 points] Let $A = (Q, q_0, \to, Q_F)$ be a DFA. The relation $\equiv_A \subseteq \Sigma^* \times \Sigma^*$ is defined by:

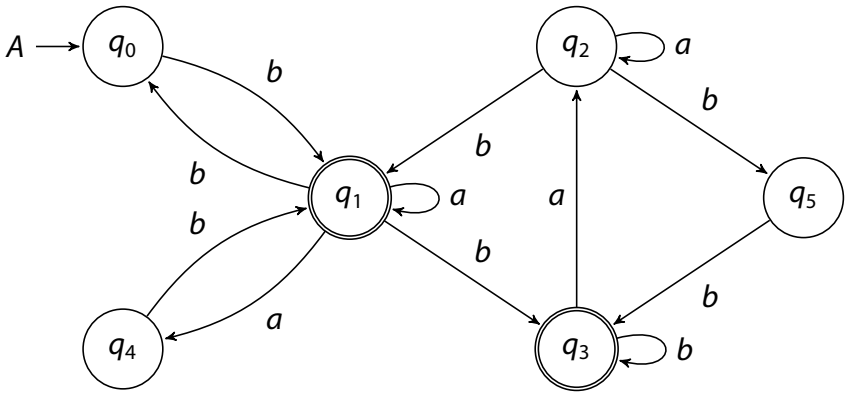
$$u \equiv_A v \quad \text{iff} \quad \exists q \in Q : q_0 \xrightarrow{u} q \text{ and } q_0 \xrightarrow{v} q.$$

Show that $\equiv_A$ is an equivalence relation.

d) [1 point] Is $\equiv_A$ from c) still an equivalence relation if A is a an NFA instead? Explain your answer!

Exercise 5: (Bonus) NFAs [11 points]

Minimalise the following automaton A.



a) [2 points] Use Arden's Rule to find a regular expression for $\mathcal{L}(A)$.

b) [3 points] Construct the powerset automaton $\mathcal{P}(A)$.

c) [5 points] Perform the *Table-Filling Algorithm* on $\mathcal{P}(A)$. Annote each marked cell of the table with the step, when that pair of states was marked. (starting with 0 for final/nonfinal states.)

d) [1 point] Draw the minimal DFA for $\mathcal{L}(A)$.