René Maseli
Thomas Haas

Theoretical Computer Science 1
Exercise Sheet 5

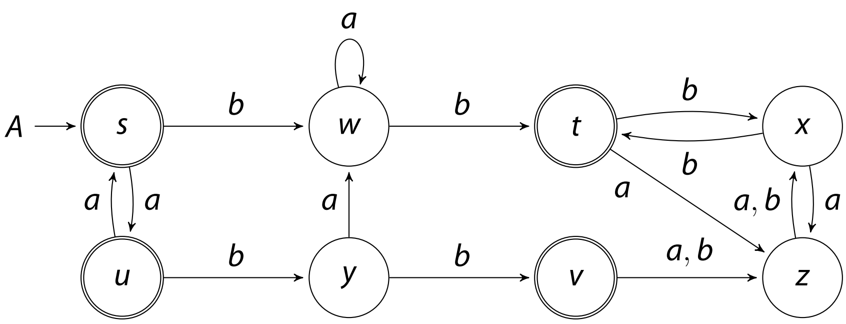
TU Braunschweig
Winter Semester 2023/24

Release: 2024-01-08 Due: 2024-01-18 23:59

Hand in your solutions to the Vips directory of the StudIP course until Thursday, January 18th 2024 23:59. You should provide your solutions either directly as .pdf file or as a readable scan/photo of your handwritten notes. Submit your results as a group of four and state **all** members of your group with **student id, name and course**.

Homework Exercise 1: Table-filling algorithm [12 points]

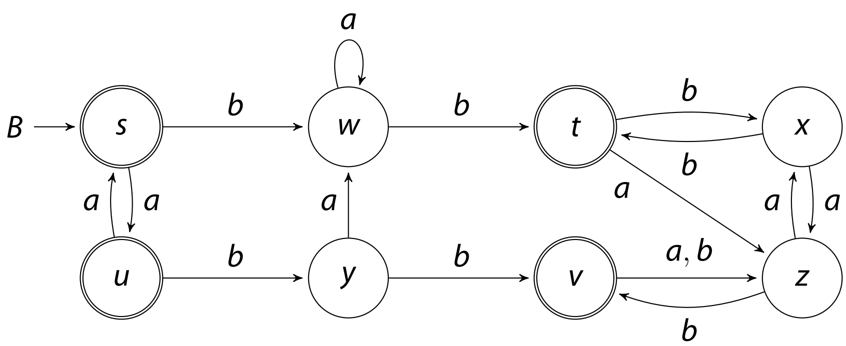
Consider the following DFA A.



a) [5 points] 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.

Now consider the DFA B, which differs from A only by one transition.



b) [5 points] Use the table-filling algorithm to find the minimal DFA $B_{\min}$ with $\mathcal{L}(B_{\min}) = \mathcal{L}(B)$. Draw the state chart of $B_{\min}$.

c) [2 points] List all equivalence classes of the Nerode-right-congruence of $\mathcal{L}(B)$ with at least one representant, each.

Homework Exercise 2: Pumping lemma for regular languages [9 points]

Consider $\Sigma = \{a, b\}$. For any word w let $|w|_a$ be the number of occurrences of symbol a in w. $|w|_b$ is defined analogously.

By using the Pumping Lemma, prove that the following languages are not regular.

a) [2 points] $L_1 = \{\, w \in \{a, b\}^* \mid |w|_b + 7 > |w|_a \,\}$

b) [3 points] $L_2 = \{\, x b^m y \in \{a, b\}^* \mid \exists n \in \mathbb{N} : |y| = n \text{ and } x \in (a^* b)^n \text{ and } m \geq 2 \,\}$

c) [2 points] $L_3 = \{\, a^n b^m \mid n < 42 \text{ or } m < n \,\}$

d) [2 points] $L_4 = \{\, w \in \{a, b\}^* \mid |w|_a \neq |w|_b \,\}$

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

Homework Exercise 3: Replacement systems [9 points]

Consider $\Sigma = \{a, b\}$. Give context free grammars G_1, G_2, G_3 and G_4, which produce the following languages:

a) [1 point] $L_1 = \{\, a^n b^m w \mid w \in \Sigma^* \text{ and } m > 2 \text{ and } |w|_a = n \,\}$.

b) [1 point] $L_2 = \{\, w \in \Sigma^* \mid |w|_a < |w|_b \,\}$.

c) [1 point] $L_3 = \{\, w \in \Sigma^* \mid \forall u, v\colon\ w = u.v \Rightarrow |v|_a \leq |v|_b \,\}$.

d) [2 points] $L_4 = \{\, b^m M^m \mid m \in \mathbb{N} \,\}$, where $M = \{\, a^n b^n \mid n \in \mathbb{N} \,\}$ is already known to be context-free (see Example 8.18 from the lecture notes). E.g. $bbabaabb \in L_4$.

A control path of a while-program can be seen as a walk in the controlflow graph, i.e. a block sequence, such that all consecutive pairs are connected with a flow edge. In program analysis, those walks are usually called paths (acyclic walks), because e.g. their index in the sequence are also considered.

Consider the following programs P_5 and P_6 with blocks $B = \{0,1,2,3,4,5,6,7\}$.

P_5:

```
[x := 0]^0
while [x^2 < y]^1 do
    [z := x + 1]^2
    [x := z^2 + z]^3
end while
if [x^2 = y]^4 then
    [z := 1]^5
else
    [z := y]^6
end if
[x := z]^7
```

P_6:

```
[x := 0]^0
while [x < 2^4]^1 do
    [y := 3x + 2]^2
    while [y < 5x]^3 do
        [y := y + 2]^4
        if [3x < y]^5 then
            [x := x + 1]^6
        end if
    end while
end while
[x := x - 14]^7
```

e) [2 points] Construct a **right-linear** grammar G_5 over the alphabet $\Sigma = B$, that produces exactly all control paths of P_5 starting in block 0 and ending in block 7. E.g. $01457 \in \mathcal{L}(G_a)$ and $01231467 \in \mathcal{L}(G_5)$, are valid, but e.g. $01234567 \notin \mathcal{L}(G_5)$ must not be producable.

f) [2 points] Construct a right-linear grammar G_6 over the alphabet $\Sigma = B$, that produces exactly all control paths of P_6 starting in block 0 and ending in block 7. E.g. $017 \in \mathcal{L}(G_6)$ is the shortest word of the language. Another element is e.g. $0123456317 \in \mathcal{L}(G_6)$, but e.g. $01234567 \notin \mathcal{L}(G_6)$ must not be produceable.

Homework Exercise 4: Linear grammars [8 points]

Prove that the regular languages exactly coincide with the languages that are produced by some right linear grammar G.

a) [4 points] Explain how to construct a right linear grammar G from a given NFA A such that $\mathcal{L}(G) = \mathcal{L}(A)$ holds.

b) [4 points] Explain how to construct an NFA A from a given right linear grammar G such that $\mathcal{L}(G) = \mathcal{L}(A)$ holds.

Remark: An analogous result holds for left linear grammars as well. That is why we speak of **regular** grammars in both cases.

Exercise 5:

Consider programs on boolean variables only. Expressions $e \in \mathrm{Exp}$ are non-deterministicly evaluated on variable states $\sigma \in \{0,1\}^V$: For $v \in \{0,1\}$, $S_v(e)$ marks the set of variable states, on which e may return v.

For a variable $x \in V$ is $S_v(x) = \{ \sigma \in \{0,1\}^V \mid \sigma(x) = v \}$. I.e. $\sigma \in S_v(y \text{ or } \text{not } z)$ holds, if and only if $v = \max(\sigma(y), 1 - \sigma(z)) \in \{0,1\}$. There is also the *havoc*-expression $*$, which non-deterministicly evaluates to both 0 and 1: $S_0(*) = S_1(*) = \{0,1\}^V$.

Let V be the set of variables in the boolean program. Some words of $\{s\} \cup (V \times \{0,1\})$ correspond to executions of the program. There is an NFA $\langle (B \times \{0,1\}^V), \langle b_0, 0^V \rangle, \rightarrow, \{f\} \times \{0,1\}^V \rangle$, whose language is exactly the set of words that correspond to halting executions. It starts on the initial block $i \in B$ with all variables set to 0 and accepts on the (sole) final block $f \in B$, regardless of the variables.

The transition $\langle b, \sigma \rangle \xrightarrow{\langle x,v \rangle} \langle c, \tau \rangle$ exists in the NFA, if and only if all $y \in V \setminus \{x\}$ satisfy $\sigma(y) = \tau(y)$, the value is $\tau(x) = v$, $b = [x := e]^\ell$ is an assignment, c is the successor of b and if $\sigma \in S_v(e)$.

The transition $\langle b, \sigma \rangle \xrightarrow{s} \langle c, \tau \rangle$ exists, if and only if $\sigma = \tau$, $b = [e]^\ell$ is the condition of a conditional or a loop, and either

- c is the first else-Block or if no such exists, the successor of b, and $\sigma \in S_0(e)$.
- c is the first then-Block or the first inner loop block and $\sigma \in S_1(e)$.

a) Consider the following program P with blocks $B = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$ and purely-boolean variables $V = \{x, y, z\}$.

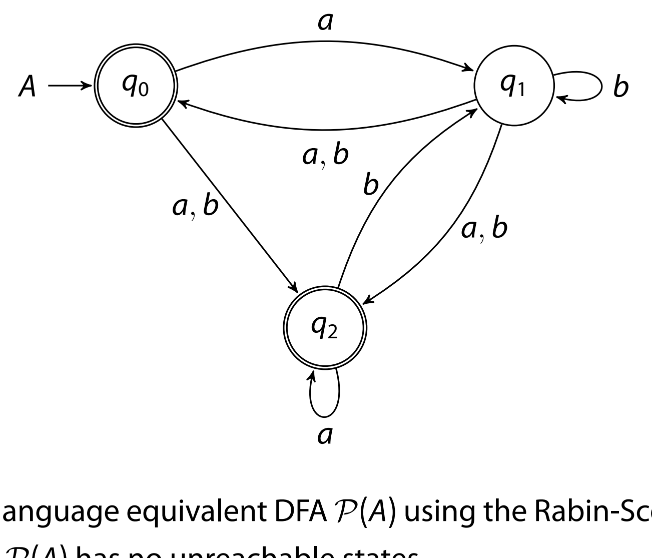
$[x := *]^0$
while $[\text{ not } x \text{ or } \text{ not } z]^1$ **do**
  $[y := \text{ not } x \text{ and } \text{ not } z]^2$
  $[x := *]^3$
  if $[y]^4$ **then**
    $[x := \text{ not } x]^5$
  end if
  $[z := x]^6$
end while
$[\text{skip}]^7$

Construct the finite automaton A_P that accepts words of $\{s\} \cup V \times \{0, 1\}$: $\Sigma = \{s, x0, x1, y0, y1, z0, z1\}$.
$\varepsilon \notin \mathcal{L}(A_P)$, since every execution must pass through Block b_0.
$x1.s \notin \mathcal{L}(A_P)$, since executions always start with $z = 0$ and therfore have to iterate at least once.
$x1.s.y0.x1.s.z1.s \in \mathcal{L}(A_P)$, because there is an execution that first reads 1 and then 0, breaking the loop.

b) Is your automaton A_P *partially* deterministic (missing transitions just have to lead into a new state $\varnothing$)?

Exercise 6:

Consider the following NFA A over $\{a, b\}$.



a) From A, construct a language equivalent DFA $\mathcal{P}(A)$ using the Rabin-Scott power set construction. Make sure that $\mathcal{P}(A)$ has no unreachable states.

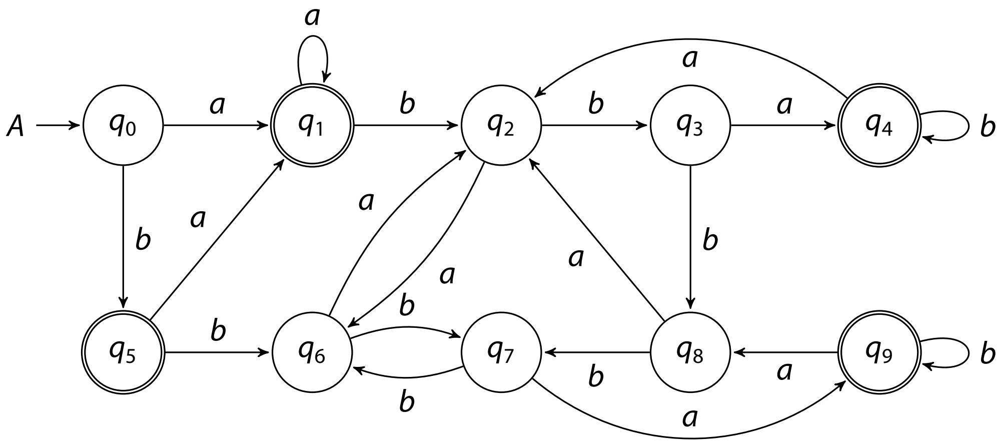
b) Determine the $\sim$-equivalence classes on the states of $\mathcal{P}(A)$ by using the Table-Filling-Algorithm from the lecture. Make clear in which order the cells of the table were marked.

c) Give the minimal DFA B for $\mathcal{L}(A)$. Make use of the $\sim$-equivalence classes.

d) Find all equivalence classes of the Nerode right-congruence $\equiv_{\mathcal{L}(A)}$.

Exercise 7:

Consider the following NFA A over $\{a, b\}$.



a) Determine the ~-equivalence classes on the states of A by using the Table-Filling-Algorithm from the lecture. Make clear in which order the cells of the table were marked.

b) Give the minimal DFA B for $\mathcal{L}(A)$. Make use of the ~-equivalence classes.

c) Find all equivalence classes of the Nerode right-congruence $\equiv_{\mathcal{L}(A)}$. Find an expression for $\mathcal{L}(A)$ as a union of a certain subset of those classes.