Hey soldier! It's me, Sergeant Emu! I don't think that I have to remind you that we are at war! The first big battle of the TI midterm is coming closer at a rapid pace. But don't worry, we trained for this and I have the ultimate battle plan! Let's take a look at what awaits us!

Overview of the Exercise Types

1. construct Finite State Automaton
2. ✓ construct non-deterministic Finite State Automaton
3. prove lower bound of #states of FSA
4. ✓ determine (best) upper bound to Kolmogorov-Complexity of word-sequence
5. ✓ construct word-sequence that satisfies upper bound to Kolmogorov-Complexity
6. prove lower bound to Kolmogorov-Complexity of word-set
7. prove non-regularity of language
   - (i) ✓ by Kolmogorov-Complexity
   - (ii) by Lemma 3.3
   - (iii) by Pumping-Lemma
8. (un)countability
9. theory exercise

Erreichbarkeit von Zuständen ✓

You might be intimidated at first sight but I will show you now exactly how to fight each exercise type. In the end, you will see that the exercises should better fear you at the Midterm! Let's get into battle preparations!

1. construct Finite State Automaton

Given: language L

Sought: FSA A with $L(A) = L$

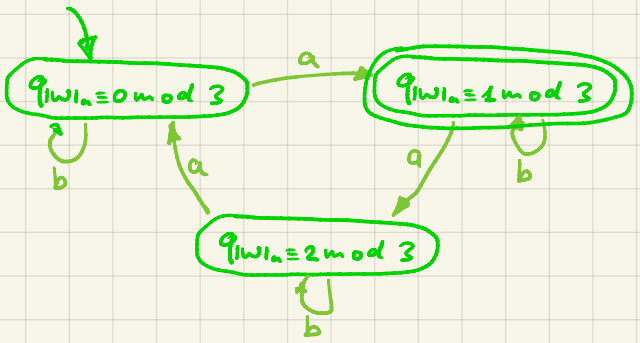
Let me first tell you the general rule of thumb of how to fight this exercise...

1. start with the states of the automaton and give each state a "meaning" (which makes it easier to determine the state-classes later). Determine the initial-state and the accepting states.
2. for each state, determine the next state for each letter.
3. write down the state-classes (with the help of the meanings of the states)

Let me also provide you with some helpful tools which are enough to solve most exercises.

Tool 1: modulo e.g. $L = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \mod 3\}$

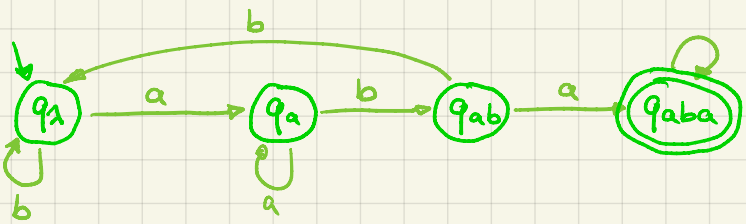
1. make a state for each Modulo-Class and determine the initial state and the accepting states
2. for each state, determine next state for each letter
3. the state-classes are equal to the modulo-classes



$$KI[q_{|w|_a \equiv i \mod 3}] = \{w \in \{a,b\}^* \mid |w|_a \equiv i \mod 3\}, \quad 1 \leq i \leq 3$$

Tool 2: substring e.g. $L = \{w \in \{a,b\}^* \mid w$ contains substring aba$\}$

1. make a state for each prefix of the substring, the empty prefix is the initial state, the whole substring is the accepting state
2. for each state, determine next state for each letter
3. to determine the state-classes, start with the accepting state, which is equal to the language itself. For all other prefixes $\neq \lambda$, say: "w ends on <prefix>" and subtract all previous state-classes. For the empty prefix, we get the state-class for free by taking $\{a,b\}^*$ and subtracting all other state-classes.



$KI[q_{aba}] = L$

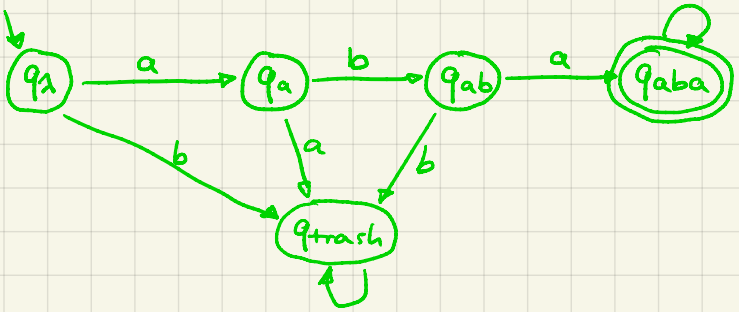
$KI[q_{ab}] = \{w \in \{a,b\}^* \mid w \text{ ends on } ab\} \setminus L$

$KI[q_a] = \{w \in \{a,b\}^* \mid w \text{ ends on } a\} \setminus (KI[q_{ab}] \cup L)$

$KI[q_\lambda] = \{a,b\}^* \setminus (KI[q_a] \cup KI[q_{ab}] \cup L)$

<u>Tool 3: prefix</u> e.g. $L = \{w \in \{a,b\}^* \mid w \text{ starts with prefix aba}\}$

1. exactly like the substring case but with an additional trash-state
2. the state-classe of the accepting state is the language L itself. The state-classes of every othe prefix-state is only one word (the prefix itself). The state-class of the trash-state is for free.



$KI[q_{aba}] = L$

$KI[q_\lambda] = \{\lambda\}$

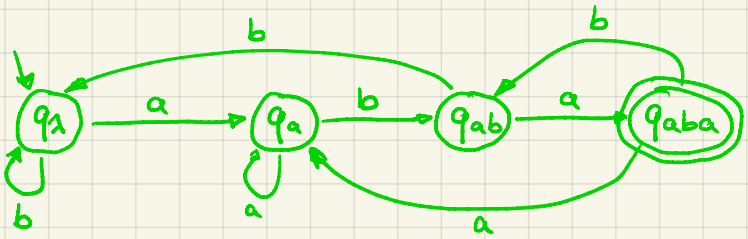
$KI[q_a] = \{a\}$

$KI[q_{ab}] = \{ab\}$

$KI[q_{trash}] = \{a,b\}^* \setminus (KI[q_{aba}] \cup KI[q_\lambda] \cup KI[q_a] \cup KI[q_{ab}])$

Tool 4: suffix e.g. $L = \{w \in \{a,b\}^* \mid w \text{ ends with suffix aba}\}$

1. exactly like the substring case but last state does not loop.
2. state-classes are exactly the same as for substring



$K\!\!\mid[q_{aba}] = L$

$K\!\!\mid[q_{ab}] = \{w \in \{a,b\}^* \mid w \text{ ends on } ab\} \setminus L$

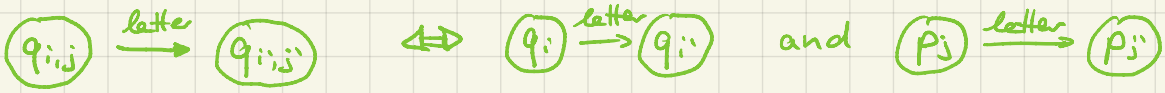
$K\!\!\mid[q_a] = \{w \in \{a,b\}^* \mid w \text{ ends on } a\} \setminus (K\!\!\mid[q_{ab}] \cup L)$

$K\!\!\mid[q_\lambda] = \{a,b\}^* \setminus (K\!\!\mid[q_a] \cup K\!\!\mid[q_{ab}] \cup L)$

Tool 5: "and", "or" (Modularer Entwurf)

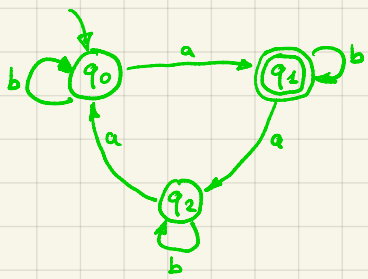
e.g. $L = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \bmod 3 \; \left[\begin{smallmatrix}\text{and}\\ \text{or}\end{smallmatrix}\right]$ w contains substring aba$\}$

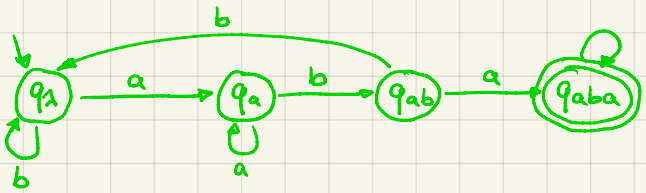
1. construct automatons for both parts. Let $q_1, \dots, q_n$ and $p_1, \dots, p_m$ be their states, where q_1 and p_1 are initial states.
2. make state joined state $q_{i,j}$ for all $1 \le i \le n$, $1 \le j \le m$. Initial state is $q_{1,1}$ and we have:

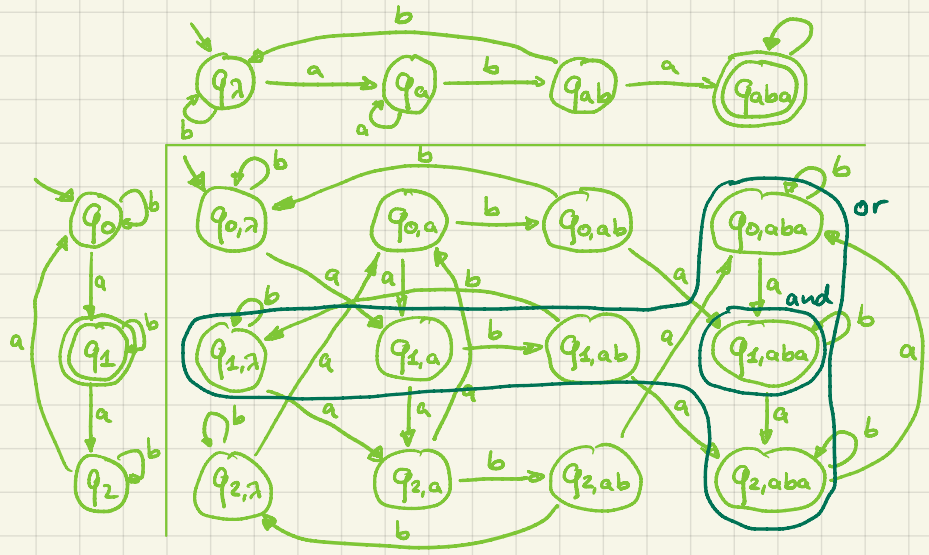


3. the accepting states are all states $q_{i,j}$ where $q_i \left[\begin{smallmatrix}\text{and}\\ \text{or}\end{smallmatrix}\right] p_j$ are accepting. For the state-classes, it holds:

$$K\!\!\mid[q_{i,j}] = K\!\!\mid[q_i] \cap K\!\!\mid[p_j]$$





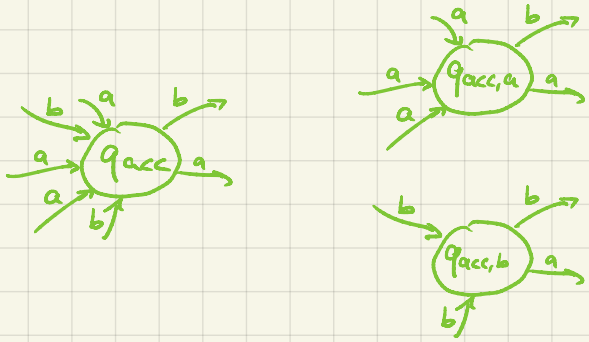


Tool 6: property + ends on letter

e.g. $L = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \bmod 3 \text{ and } w \text{ ends on } b\}$

1. construct automaton for property. Let q_{acc} be accepting state.

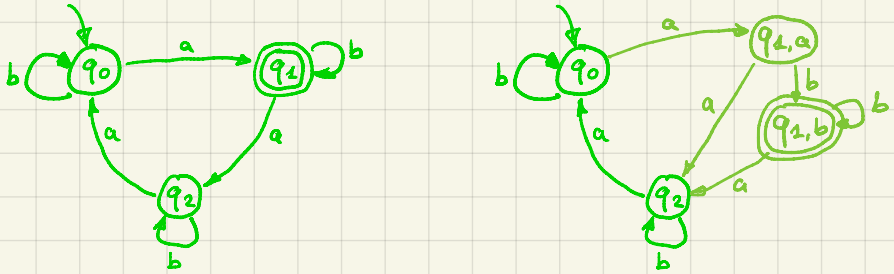
2. dublicate q_{acc} into $q_{acc,a}$ and $q_{acc,b}$. Make $q_{acc,b}$ the new accepting state. Reroute incoming arrows of q_{acc} with letter a into $q_{acc,a}$ and with letter b into $q_{acc,b}$. Outgoing arrows are copied.



3. For the state-classes, it holds:

$$KI[q_{acc,a}] = KI[q_{acc}] \cap \{w \in \{a,b\}^* \mid w \text{ ends on } a\}$$

$$KI[q_{acc,b}] = KI[q_{acc}] \cap \{w \in \{a,b\}^* \mid w \text{ ends on } b\}$$

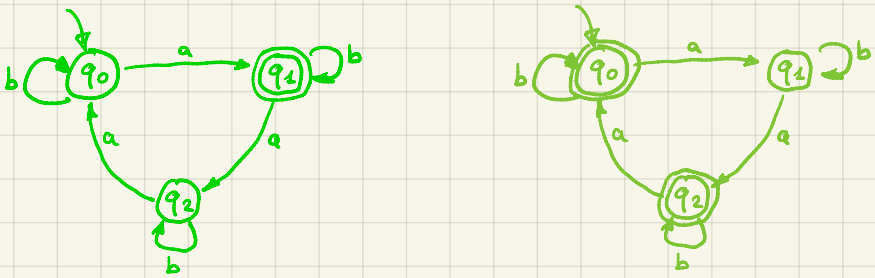


Note: If there are multiple accepting states, repeat this process for all accepting states.
If there are more than 2 letters in the alphabet, re-route them all into $q_{acc,a}$.

Tool 7: complement of property

e.g. $L = \{w \in \{a,b\}^* \mid |w|_a \neq 1 \bmod 3\}$

1. construct automaton for property.
2. take complement of accepting states
3. state-classes stay the same



How to get full points:

1. draw FSA
2. write down state-classes
3. (if wanted by exercise) give informal explanation

Tipps & Tricks:

1. don't forget to mark initial state
2. don't forget to do a transition for every letter
3. if there is only one accepting state, its state-class is for free (it is the language itself)
4. the last state-class is always for free
5. make sanity checks eg. by trying to find a word that is wrongly accepted by the NFSA.

Example Exercises

Sheet 3, 2021

Aufgabe 8

Entwerfen Sie endliche Automaten für die folgenden Sprachen in Diagrammdarstellung und geben Sie für Ihre Automaten jeweils die Klasse Kl[q] für jeden Zustand q an.

(a) $L_1 = \{w \in \{a,b\}^* \mid (|w|_a + 2 \cdot |w|_b + 1) \bmod 3 \neq 1\}$, 1+7

(b) $L_2 = \{w \in \{a,b\}^* \mid w$ enthält das Teilwort bab nicht$\}$. 2+7

10 Punkte

Aufgabe 9

(a) Entwerfen Sie einen endlichen Automaten für die Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

in Diagrammdarstellung. Erklären Sie kurz die Idee hinter ihrem Entwurf.

(b) Geben Sie für jeden Zustand q des entworfenen Automaten die Klasse Kl[q] an.

10 Punkte

Sheet 4, 2021

Aufgabe 10

Verwenden Sie die Methode des modularen Entwurfs (Konstruktion eines Produktautomaten), um einen endlichen Automaten für die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_a \bmod 3 = |w| \bmod 3 \text{ oder } (\ w \text{ enthält das Teilwort } ab \text{ und } w \text{ endet mit } b\ )\}$$ 1+5+2+6

zu entwerfen. Geben Sie für die Teilautomaten jeweils für jeden Zustand q die Klasse Kl[q] an.

10 Punkte

Sheet 5, 2021 (advanced exercise)

Aufgabe 15

Beweisen Sie die folgenden beiden Aussagen, indem Sie jeweils eine Automaten-Konstruktion angeben und informell ihre Korrektheit begründen.

(a) Seien L_1 und L_2 zwei reguläre Sprachen über dem Alphabet $\Sigma = \{a,b\}$. Dann ist auch die Sprache $L = L_1 \cdot \{c\} \cdot L_2$ über dem Alphabet $\{a,b,c\}$ regulär.

(b) Sei L eine reguläre Sprache über einem beliebigen Alphabet Σ. Dann ist auch die Sprache $L^R := \{w^R \mid w \in L\}$ regulär.

10 Punkte

Sheet 3, 2022

Aufgabe 7

Entwerfen Sie für die folgenden Sprachen jeweils einen endlichen Automaten (in Diagrammdarstellung) und geben Sie für jeden Zustand q Ihres Automaten die Klasse Kl[q] an.

(a) $L_1 = \{xbbya \in \{a,b\}^* \mid x,y \in \{a,b\}^*\}$,

(b) $L_2 = \{w \in \{a,b\}^* \mid (|w|_a)^{|w|_b} \equiv 0 \bmod 2\}$. 2 advanced

10 Punkte

Aufgabe 8

Verwenden Sie die Methode des modularen Entwurfs (Konstruktion eines Produktautomaten), um einen endlichen Automaten (in Diagrammdarstellung) für die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_a = 2 \text{ oder } w = ya\}$$ 5

zu entwerfen. Zeichnen Sie auch jeden der Teilautomaten und geben Sie für die Teilautomaten jeweils für jeden Zustand q die Klasse Kl$[q]$ an. **10 Punkte**

Aufgabe 9

Entwerfen Sie für die folgenden Sprachen jeweils einen endlichen Automaten (in Diagrammdarstellung) und begründen Sie kurz informell die Korrektheit Ihres Entwurfs.

(a) $L_1 = \{xa^k \in \{a,b\}^* \mid x \in \{b\}^* \wedge k := (|x|^2 \bmod (|x| - 2)) \wedge k \leq 3 \leq |x| \leq 10\}$,

(b) $L_2 = \{w \in \{a,b\}^* \mid |w|_a \cdot |w|_b \equiv 1 \bmod 3\}$. 2 advanced

10 Punkte

Sheet 3, 2023

Aufgabe 7

Entwerfen Sie endliche Automaten für die folgenden Sprachen in Diagrammdarstellung und geben Sie für Ihre Automaten jeweils die Klasse Kl$[q]$ für jeden Zustand q an.

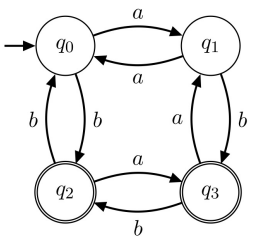
(a) $L_1 = \{w \in \{a,b\}^* \mid (2 \cdot |w|_a + |w|_b) \bmod 3 = 2\}$, 2

(b) $L_2 = \{w \in \{a,b\}^* \mid w$ enthält das Teilwort $baab$ nicht$\}$. 1,7

10 Punkte

Aufgabe 8 (reverse)

Bestimmen Sie für den folgenden endlichen Automaten M die akzeptierte Sprache $L(M)$, sowie die Klassen Kl$[q_i]$ für jeden Zustand q_i. Geben Sie eine kurze intuitive Begründung.



10 Punkte

Aufgabe 9

(a) Entwerfen Sie einen endlichen Automaten für die Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

in Diagrammdarstellung. Erklären Sie kurz die Idee hinter ihrem Entwurf.

(b) Geben Sie für jeden Zustand q des entworfenen Automaten die Klasse Kl$[q]$ an.

10 Punkte

Sheet 4, 2023

Aufgabe 10

Verwenden Sie die Methode des modularen Entwurfs (Konstruktion eines Produktautomaten), um einen endlichen Automaten für folgende Sprache zu entwerfen:

$$L = \{w \in \{a,b\}^* \mid |w|_a \bmod 3 = |w|_b \bmod 3 \text{ oder } (w \text{ enthält das Teilwort } ba \text{ und endet mit } a)\}$$

Geben Sie für die Teilautomaten jeweils für jeden Zustand q die Klasse Kl$[q]$ an. **10 Punkte**

Sheet 3, 2024

Aufgabe 8

Entwerfen Sie endliche Automaten für die folgenden Sprachen in Diagrammdarstellung und geben Sie für Ihre Automaten jeweils die Klasse Kl[q] für jeden Zustand q an.

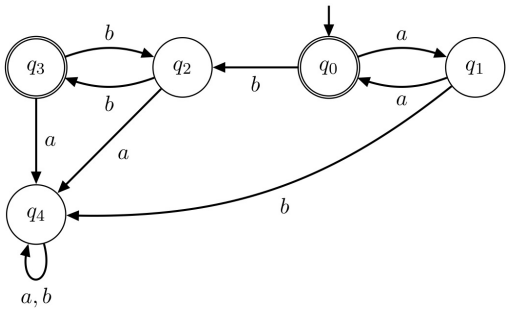
(a) $L_1 = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } abba\}$,

(b) $L_2 = \{w \in \{a,b\}^* \mid |w|_a \neq |w|_b \pmod 2\}$.

10 Punkte

Aufgabe 9

Bestimmen Sie für den folgenden endlichen Automaten M die akzeptierte Sprache $L(M)$, sowie die Klassen Kl[q_i] für jeden Zustand q_i. Geben Sie eine kurze intuitive Begründung.



10 Punkte

Sheet 4, 2024

Aufgabe 10

Entwerfen Sie einen endlichen Automaten für die folgende Sprache in Diagrammdarstellung und geben Sie für Ihren Automaten jeweils die Klasse Kl[q] für jeden Zustand q an.

$$L = \{xw \in \Sigma^*_{\text{bool}} \mid x \in \Sigma_{\text{bool}} \text{ und Nummer}(w) \equiv x \pmod 4\}$$

Hinweis: Für $w = w_1 w_2 \dots w_n \in \Sigma^*_{\text{bool}}$ ist $\text{Nummer}(w) := \sum_{i=1}^{n} w_i \cdot 2^{n-i}$. **10 Punkte**

Aufgabe 11

Verwenden Sie die Methode des modularen Entwurfs (Konstruktion eines Produktautomaten), um einen endlichen Automaten (in Diagrammdarstellung) für die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \pmod 3 \text{ oder } |w| = 2\}$$

zu entwerfen. Zeichnen Sie auch jeden der Teilautomaten und geben Sie für die Teilautomaten jeweils für jeden Zustand q die Klasse Kl[q] an.

10 Punkte

Exam Questions:

- 2023: 1, 2b)
- 2022: 1, 2
- 2021: 1a), b)
- 2019: 1
- 2018: 1b), 3a)
- 2017: 1
- 2016: 1
- 2015: 1a), b)
- 2014: 1a)
- 2013: 1
- 2012: 2
- 2011: 1
- 2010: 1
- 2008: 1
- 2007: 2
- 2004: 2

2. construct non-deterministic Finite State Automaton

Given: language L

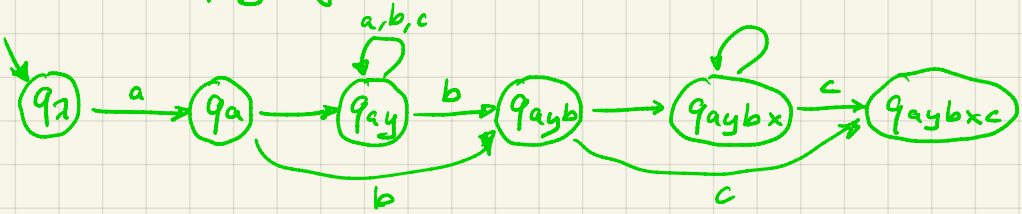
Sought: non-deterministic FSA A with $L(A) = L$

Listen up soldier! You know now how to construct a deterministic FSA for a given language. All the tools and tricks that you aquired for deterministic FSAs also transfer to non-deterministic ones. But there are two more tools you can use. In particular, we can exploit that NFSAs are good at "guessing". Let's go!

Tool 8: guess a placeholder

e.g. $L = \{aybxc \mid y, x \in \{a,b,c\}^*\}$

1. at the point, where the automaton reads in the "placeholders", it simply "guesses" when the word is done.

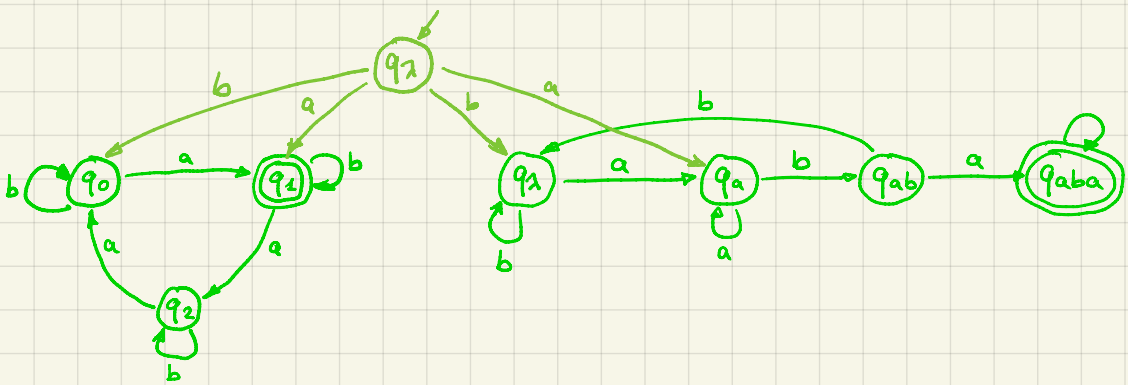


Note: there is no trash-state since NFSAs can have the empty set as the next states.

Tool 9: guess an "or"

e.g. $L = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \bmod 3 \text{ or } w \text{ contains substring } aba\}$

1. construct an NFSA for both parts.
2. combine both automatons by "guessing" in the initial-state which automaton to go into



How to get full points:

1. draw NFSA
2. (if wanted by exercise) give informal explanation

Tipps & Tricks:

1. dont forget to mark initial state
2. be aware that NFSAs can "guess" and have an integrated trash-state (by not making a transition for a letter)
3. make sanity checks eg. by trying to find a word that is wrongly accepted by the NFSA.
4. be careful with placeholders being λ
5. write letters on every arrow

Example Exercises

Sheet 5, 2021

Aufgabe 14

(a) Entwerfen Sie einen nichtdeterministischen endlichen Automaten mit höchstens 8 Zuständen für die Sprache

$$L = \{x \in \{0,1\}^* \mid |x|_0 \bmod 3 = 0 \text{ oder } x = 1y01 \text{ für } y \in \{0,1\}^+\} .$$

Geben Sie Ihren entworfenen Automaten in Diagrammdarstellung an und erläutern Sie die Idee Ihres Entwurfs.

Sheet 5, 2022

Aufgabe 14

Entwerfen Sie nichtdeterministische endliche Automaten für die folgenden Sprachen.

(a) $L_1 = \{x \in \{a,b\}^* \mid |x|_a \bmod 3 = 2 \text{ oder } (x = yaz \text{ mit } y, z \in \{a,b\}^* \text{ und } |z| = 3)\}$,

(b) $L_2 = \{w \in \{0,1\}^* \mid w = x0y0z \text{ mit } x, z \in \{0,1\}^* \text{ und } y \in \{0,1\}^2\}$.

Geben Sie Ihre entworfenen Automaten in Diagrammdarstellung an und erläutern Sie die Ideen Ihrer Entwürfe. **10 Punkte**

Sheet 5, 2023

Aufgabe 13

(a) Entwerfen Sie einen nichtdeterministischen endlichen Automaten mit höchstens neun Zuständen für die Sprache

$$L = \{x \in \{a,b\}^* \mid (|x|_a - |x|_b \bmod 3 = 0) \text{ oder } x = abaa \text{ oder } x = baaa\} .$$

Geben Sie Ihren entworfenen Automaten in Diagrammdarstellung an und erläutern Sie die Idee Ihres Entwurfs.

Sheet 6, 2023 (advanced exercise)

Aufgabe 18

Ein λ-NEA ist ein nichtdeterministischer endlicher Automat, bei dem wir auch Transitionen mit dem leeren Wort λ zulassen. Ein λ-NEA kann also entlang einer λ-Transition seinen Zustand wechseln, ohne dabei ein Symbol der Eingabe zu lesen. Formal ist ein λ-NEA M ein Quintupel $M = (Q, \Sigma, \delta, q_0, F)$, wobei Q, Σ, q_0 und F wie bei einem NEA definiert sind, und

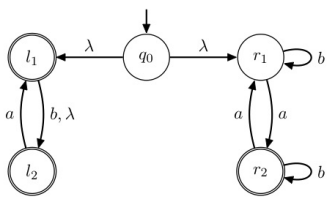
$$\delta \colon Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$$

die Übergangsfunktion ist, wobei jetzt neu $\delta(q, \lambda)$ die Menge von Zuständen ist, die von einem gegebenen Zustand $q \in Q$ aus mit einer λ-Transition erreichbar sind.

(a) Wir betrachten die folgende Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } bb \text{ nicht, oder } |w|_a \text{ ist ungerade}\} .$$

Geben Sie ein informelles Argument dafür, dass der folgende λ-NEA genau die Sprache L akzeptiert.



(b) Beschreiben Sie ein allgemeines Verfahren, mit dem man jeden λ-NEA in einen äquivalenten NEA umwandeln kann. Begründen Sie die Korrektheit des Verfahrens.

(c) Wenden Sie Ihr Verfahren auf den λ-NEA aus Teilaufgabe (a) an, um einen äquivalenten NEA für die Sprache L zu erhalten.

10 Punkte

Sheet 5, 2024

Aufgabe 14

(a) Entwerfen Sie einen nichtdeterministischen endlichen Automaten für die folgende Sprache in Diagrammdarstellung und erläutern Sie die Idee Ihres Entwurfs.

$$L_1 = \{w \in \{0,1\}^* \mid w = x0y0z \text{ mit } x, z \in \{0,1\}^* \text{ und } y \in \{0,1\}^2\}.$$

Exam Questions:

- 2023: 2 a)
- 2022: 3 a)
- 2019: 3 a)
- 2018: 1 a)
- 2014: 1 b)

3. prove lower bound of #states of FSA

Given: language L, $k \in \mathbb{N}$

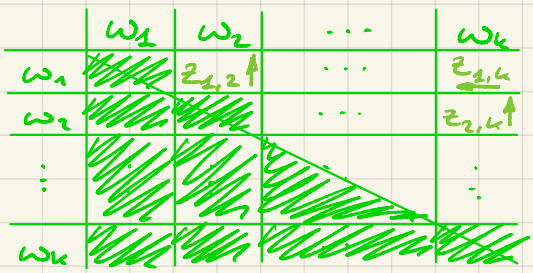
Sought: prove that every FSA A with $L(A) = L$ has at least k states

Hey, you are doing very good so far! It looks like you got a grasp now what FSAs and NFSAs are all about! This task is going to be a snack for you, I'm sure. Let me tell you what has to be done here. Remember Lemma 3.3 from the script? If not, this is the point, where you should quickly open the script and read again what Lemma 3.3 says. Done? Ok, let's roll!

1. find k words $w_1, \dots, w_k$ of which you think that they are in pairwise different states in every automaton. Write them into a table (like below).

2. for each pair $i < j$, find a suffix $z_{i,j}$ such that exactly one of the words $w_i z_{i,j}$, $w_j z_{i,j}$ is in L and the other is not. Best is if you also indicate in the table which is in L, e.g. with an arrow.

3. now, by Lemma 3.3, each of the words $w_1, \dots, w_k$ has an own state, which implies that every FSA A indeed has at least k states.



$w_1 z_{1,2} \in L$, $w_2 z_{1,2} \notin L$ (or other way around) $\xrightarrow{\text{Lemma 3.3}}$ $\hat{\delta}(q_0, w_1 z_{1,2}) \neq \hat{\delta}(q_0, w_2 z_{1,2})$

Note: The k can also be a number depending on L, e.g. if we are given a series of languages $L_1, L_2, \dots$ and have to prove that L_i needs at least i states. In this case, a table is not the way to go but instead just prove that for arbitrarily chosen w_i, w_j there exists a $z_{i,j}$ such that exactly one word of $w_i z_{i,j}$ and $w_j z_{i,j}$ is in L.

Btw, if $k = \infty$, we prove that L is not regular ☺.

How to get full points

1. state the words $w_1, \dots, w_k$
2. state the words $z_{i,j}$ $\forall i<j$. If k is a constant, make the table with the indicators. Else, take general w_i and w_j and show that exactly one of $w_i z_{i,j}$ and $w_j z_{i,j}$ is in L
3. cite Lemma 3.3 and state that therefore all words are in pairwise different states, which implies $\geq k$ states.

Tipps & Tricks

1. don't forget to state Lemma 3.3
2. if you have trouble finding the words $w_1, \dots, w_k$ it might help you to think of how you would construct a (minimal) FSA for the language L. If you take a word out of each state-class, they will be definitely in different states for all FSAs of L. However, the more exercises you solve, the better your intuition gets :-)
3. this exercise-type is prone to careless mistakes (Flüchtigkeitsfehler). So, be carefull and do not rush too much.
4. you only have to find suffixes $z_{i,j}$ for $i<j$ (see table, not all cells need to be filled out).

Example Exercises

Sheet 4, 2021

Aufgabe 11

Zeigen Sie, dass jeder endliche Automat, der die Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

akzeptiert, mindestens 5 Zustände hat. **5 Punkte**

Sheet 4, 2022

Aufgabe 10

Zeigen Sie, dass jeder (deterministische) endliche Automat, der die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_b \bmod 3 = 1 \text{ und } w \text{ beginnt mit } ab\}$$

akzeptiert, mindestens 6 Zustände benötigt.

Hinweis: Der Beweis von Lemma 3.6 im Buch und die daran anschliessende Diskussion liefern eine mögliche Beweismethode hierfür. **10 Punkte**

Sheet 5, 2023

Aufgabe 15

Zeigen Sie, dass jeder deterministische endliche Automat, der die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_b \bmod 3 = 2 \text{ und } w \text{ beginnt mit } bb\}$$

akzeptiert, mindestens sechs Zustände benötigt.

10 Punkte

Sheet 5, 2024

Aufgabe 14

(b) Zeigen Sie, dass jeder (deterministische) endliche Automat, der die Sprache

$$L_2 = \{w \in \{a,b\}^* \mid |w|_b \equiv 1 \pmod 3 \text{ und } w \text{ beginnt mit } ab\}$$

akzeptiert, mindestens 6 Zustände hat.

5+10 Punkte

Exam Questions:

- 2023: 2c)
- 2022: 3b)
- 2021: 1c)
- 2019: 3b)
- 2018: 3b)
- 2016: 2a)
- 2015: 1c)

4. determine (best) upper bound to Kolmogorov-Complexity of word-sequence

Given: sequence $x_1, x_2, x_3, \ldots$ of words

Sought: (best) upper bound $K(x_i) \leq f(|x_i|)$

Ok soldier, remember the concept of Kolmogorov-Complexity? No? Then go to the script again and refresh your memory. Done? Ok, then let me also loose a couple of summarizing words about Kolmogorov-Complexity. $K(x)$ is the binary length of the SHORTEST program that DOES NOT TAKE ANY INPUT and writes out the word x. Did you note that your inner voice read some words in a screaming voice and the rest normally? That is because these words are IMPORTANT. Let me tell you why. Since $K(x)$ is the shortest length of a program, we can give one program P that prints x and measure its length $len(P)$. This length is then an upper bound to $K(x)$ since there might of course still be shorter programs but our program is a "witness" that that $K(x)$ can not be highe than $len(P)$. Furthermore, since the program is not allowed to get any input, we somehow have to hardcode the information that we need into the program. This is exactly the part that changes between words of the sequence $x_1, x_2, \ldots$ The rest of the program stays the same and has constant length. The information that we hardcode for word x_i will (almost always) be i. Since we can hardcode i in binary, we get the length $\log i$ for the hardcoded part.
Let us now tackle the exercise!

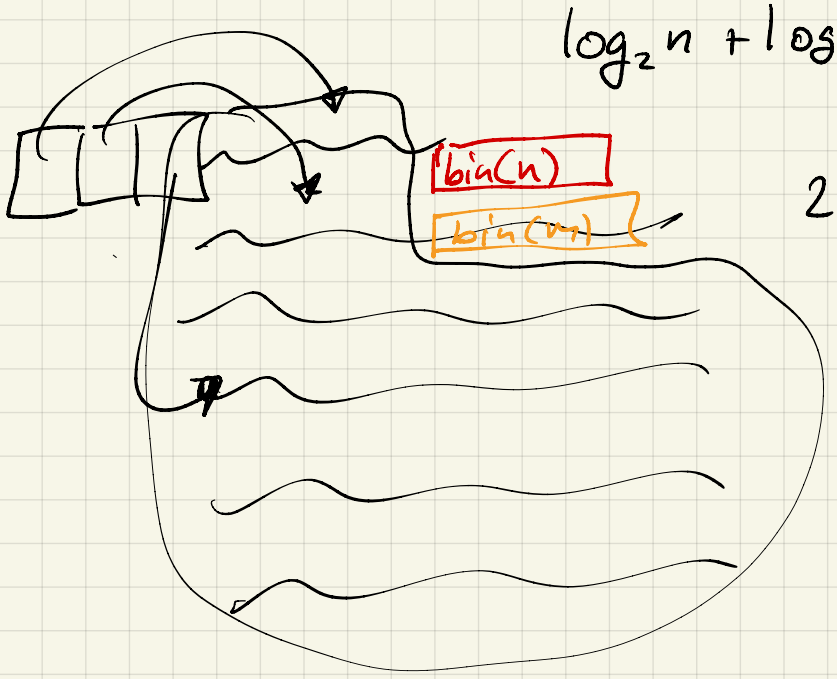
1. write a program P_i (in pseudo-code) that does not take any input and writes out x_i.
2. measure the length of P_i (this length will depend on i)
3. determine the length of P_i in relation to $|x_i|$ (that is, determine i in relation to $|x_i|$ and replace i in $len(P_i)$)

How to get full points:

1. write down the program P_i and state that it (obviously) writes out x_i.
2. write down $len(P_i)$
3. show how to determine i in relation to $|x_i|$
4. replace i in $len(P_i)$ by what you found in 3. and simplify (if needed).

Tipps & Tricks:

1. don't forget to write down the program P_i
2. don't forget that P_i does NOT take any input.
3. be careful with $|x_i|$, e.g. $|(010)^n| \neq 3^n$ it is $3n$
4. in P_i hardcode the information (almost always i) only once. This is done best by assigning i to a variable I in the first line and then only using I.



$$\log_2 n + \log_2 m + \boxed{\log_2 \log_2 n}$$

$$2 \cdot \log_2 n + C$$

$$\lceil \log_2 (n+d) \rceil$$

Example Exercises

Sheet 2, 2021

Aufgabe 6

Wir betrachten die Sprache

$$L = \{1^i 0^j 1^k \mid i, j, k \in \mathbb{N} - \{0\}\}.$$

Sei x_n das kanonisch n-te Wort in L. Zeigen Sie, dass es eine Konstante $c \in \mathbb{N}$ gibt, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq 3 \cdot \log_2(|x_n|) + c$$

gilt. **10 Punkte**

Sheet 2, 2022

Aufgabe 4

(a) Sei $w_n = (010)^{3^{2n^3}} \in \{0,1\}^*$ für alle $n \in \mathbb{N} - \{0\}$. Geben Sie eine möglichst gute obere Schranke für die Kolmogorov-Komplexität von w_n an, gemessen in der Länge von w_n.

Sheet 2, 2023

Aufgabe 6

Sei $\Sigma = \{0,1\}$. Ein *Palindrom* über Σ ist ein Wort $x \in \Sigma^*$, das vorwärts und rückwärts gelesen gleich ist, für das also $x^R = x$ gilt.

(a) Sei $k \in \mathbb{N}$. Wieviele Palindrome der Länge k gibt es? Wieviele Palindrome der Länge höchstens k gibt es? Berechnen Sie die exakten Werte.

(b) Sei x_n das n-te Palindrom in kanonischer Ordnung. Verwenden Sie Satz 2.2, um eine obere Schranke der Form $K(x_n) \leq \alpha \cdot |x_n| + d$ zu beweisen, wobei α eine möglichst kleine und d eine beliebige von n unabhängige Konstante ist.

Sheet 2, 2024

Aufgabe 4

(a) Sei $w_n = 0^{100\,000} \cdot (0101)^{4n^2} \in \{0,1\}^*$ für alle $n \in \mathbb{N} - \{0\}$. Geben Sie eine möglichst gute obere Schranke für die Kolmogorov-Komplexität von w_n an, gemessen in der Länge von w_n.

Exam Questions

- 2023: 6
- 2017: 3a)
- 2014: 4a)
- 2010: 2a)
- 2004: 1a)

5. construct word-sequence that satisfies upper bound to Kolmogorov-Complexity

Given: upper bound $f(l)$

Sought: sequence $x_1, x_2, x_3, \ldots$ of words such that $K(x_i) \leq f(|x_i|)$

You surely noticed that this exercise is simply the reverse of the previous one. This is exactly right and also tells us what to do to solve it. We have to walk backwards in our mind... This is best explained on an example.

e.g. $f(l) = \log_2 \log_3 \log_2 l + c$

1. consider $f(l)$ and go backwards in your mind: at the end we inserted i in relation to $|x_i|$ into $\log_2 i + c$. Thus, we can extract i in relation to $|x_i|$ from $f(l)$.

2. now, we want to know what $|x_i|$ is in relation to i, which is some simple calculation

$$i = \log_3 \log_2 |x_i| \quad \Rightarrow$$

$$3^i = \log_2 |x_i| \quad \Rightarrow$$

$$2^{3^i} = |x_i|$$

3. lastly, construct a sequence of words that satisfies this equation:

$$x_i := a^{2^{3^i}}$$

How to get full points:

1. give the sequence of words (that you found by the steps above)

   prove that your sequence indeed satisfies the wanted upper bound in the same way as in exercise type 4. That is:

2. write down the program P_i and state that it (obviously) writes out x_i.

3. write down $len(P_i)$

4. show how to determine i in relation to $|x_i|$

5. replace i in $len(P_i)$ by what you found in 3. and simplify (if needed). This should prove the sought upper bound.

Tipps & Tricks

The same Tipps and Tricks from exercise type 4. apply, that is:

1. don't forget to write down the program P_i

2. don't forget that P_i does NOT take any input.

3. be careful with $|x_i|$, e.g. $|(010)^n| \neq 3^n$ it is $3n$

4. in P_i hardcode the information (almost always i) only once. This is done best by assigning i to a variable I in the first line and then only using I.

Here some more Tipps & Tricks:

5. note that the outermost $\log_2$ corresponds to the hardcoding in binary (not the innermost)

6. best is that you take the time to really calculate $|x_i|$ like in the steps I showed you and do not guess the word x_i since this is where careless mistakes are made.

Example Exercises

Sheet 2, 2021

Aufgabe 4

Geben Sie jeweils eine unendliche Folge $(x_n)_{n=1}^{\infty}$ paarweise unterschiedlicher Wörter an, welche die Bedingung erfüllt, oder beweisen Sie, dass es keine solche Folge gibt.

(a) Es gibt eine Konstante $c \in \mathbb{N}$, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq \log_2 \log_2 |x_n| + c$$

gilt.

Sheet 2, 2022

Aufgabe 4

(b) Geben Sie eine unendliche Folge von Wörtern $y_1 < y_2 < y_3 < \cdots$ an, so dass eine Konstante $c \in \mathbb{N}$ existiert, so dass für alle $i \geq 1$ gilt, dass

$$K(y_i) \leq \log_2 \log_2 \log_3 \log_2(|y_i|) + c\,.$$

Sheet 2, 2023

Aufgabe 4

(b) Geben Sie eine unendliche Folge von Wörtern $y_0 < y_1 < y_2 < \cdots$ an, so dass eine Konstante $c \in \mathbb{N}$ existiert, so dass für alle $i \in \mathbb{N}$ gilt, dass

$$K(y_i) \leq \log_2 \log_3(|y_i|) + c\,.$$

Exam Questions

- 2023: 5
- 2017: 3b)
- 2012: 1c)

6. prove lower bound to Kolmogorov-Complexity of word-set

Given: set of words W, lower bound f

Sought: prove that at least some amount of words w satisfies $K(w) \geq f$

This exercise comes in many variations as you will see in the exercise-section later. However, all of those exercises are solved in the same way: counting the amount of programs of length f, counting the amount of words in W and then using the Pigeonhole-Principle (almost always). Let's go!

1. count the amount n of programs of length $\leq f$
2. count the amount m of words in W
3. $m-n$ words do not have a program and therefore satisfy the lower bound $K(w) \geq f$.

Note: This exercise-type can hide behind an indirect proof, e.g. "show there is no sequence $x_1, x_2, \ldots$ such that $\forall i : K(x_i) < f$". Then you will show that for every sequence $x_1, x_2, \ldots$ it holds $K(x_i) \geq f$ for at least one i.

How to get full points:

Exactly what I already wrote in the tutorial. That is:

1. count the amount n of programs of length $\leq f$
2. count the amount m of words in W
3. $m-n$ words do not have a program and therefore satisfy the lower bound $K(w) \geq f$.

Tipps & Tricks:

1. The amount of programs of length i is 2^i.
2. The amount of words over alphabet Σ with length i is $|\Sigma|^i$.
3. The amount of programs of length $\leq i$ is $2^{i+1}-1$.
4. The amount of words over alphabet Σ with length $\leq i$ is $\frac{|\Sigma|^{i+1}-1}{|\Sigma|-1}$.
5. In general, to calculate a geometric series $d^k + d^{k+1} + \cdots + d^n$, do the following trick:

$$D := d^k + d^{k+1} + \cdots + d^n \quad \Rightarrow$$

$$dD = d^{k+1} + d^{k+2} + \cdots + d^{n+1} \quad \Rightarrow$$

$$dD - D = d^{n+1} - d^k \quad \Rightarrow$$

$$D = \frac{d^{n+1} - d^k}{d-1}$$

Example Exercises

Sheet 2, 2021

Aufgabe 4

Geben Sie jeweils eine unendliche Folge $(x_n)_{n=1}^{\infty}$ paarweise unterschiedlicher Wörter an, welche die Bedingung erfüllt, oder beweisen Sie, dass es keine solche Folge gibt.

(b) Es gibt eine Konstante $c \in \mathbb{N}$, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq \log_2 \sqrt{n} + c$$

gilt.

1.

10 Punkte

1. zähle Prog. und Wörter

2. dir wird untere Schranke gegeben (durch Zufälligkeit) finde obere Schranke durch Programm

Aufgabe 5

Zeigen Sie, dass die Menge $\{n^2 \mid n \in \mathbb{N}\}$ endlich viele Zahlen enthält, die als zufällige Zahlen betrachtet werden können. 2. **10 Punkte**

Sheet 3, 2021

Aufgabe 7

Sei $n \in \mathbb{N}$. Zeigen Sie, dass mindestens die Hälfte der Wörter über $\{0,1\}$ der Länge höchstens n zufällig ist. 1. **10 Punkte**

Sheet 2, 2022

Aufgabe 5

Geben Sie eine Funktion $f\colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$ an, so dass für jedes $n \in \mathbb{N}$ mindestens ein Anteil von $7/8$ der natürlichen Zahlen kleiner als $f(n)$ eine Kolmogorov-Komplexität von mindestens $2n$ haben. 1. **10 Punkte**

Aufgabe 6

Zeigen Sie, dass es höchstens endlich viele Primzahlen geben kann, die als zufällige Zahlen (im Sinne von Definition 2.19 im Buch) betrachtet werden können. 2.

Hinweis: Sie dürfen für den Beweis den Primzahlsatz verwenden. **10 Punkte**

Sheet 2, 2023

Aufgabe 5

Zeigen Sie, dass es für jedes $n \in \mathbb{N}$ und jedes $i < n$ mindestens $2^n - 2^{n-i}$ natürliche Zahlen x in dem Intervall $[2^n, 2^{n+1} - 1]$ gibt mit $K(x) \geq n - i$. 1. **10 Punkte**

Sheet 2, 2024

Aufgabe 5

Sei $n \in \mathbb{N}$. Zeigen Sie, dass die Kolmogorov-Komplexität von mehr als 99% der Wörter in $\{0,1\}^n$ grösser als $n - 8$ ist. 1. **10 Punkte**

Aufgabe 6

Zeigen Sie, dass kein Pascal-Programm existiert, welches als Input eine beliebige Zahl $n \in \mathbb{N}$ entgegennimmt und dann ein Wort $w \in \{0,1\}^n$ mit $K(w) > n - 8$ ausgibt. **10 Punkte**

Sheet 3, 2024

Aufgabe 7

Zeigen Sie, dass es höchstens endlich viele Primzahlen geben kann, die als zufällige Zahlen (im Sinne von Definition 2.19 im Buch) betrachtet werden können. *Hinweis:* Sie dürfen für den Beweis den Primzahlsatz verwenden. **10 Punkte**

Exam Questions

- 2021: 3
- 2018: 4
- 2016: 4
- 2015: 3
- 2014: 4b)
- 2013: 3
- 2012: 1b)
- 2011: 3
- 2010: 2b)
- 2008: 4
- 2007: 1

7. prove non-regularity of language

Given: language L

Sought: prove that L is not regular.

Ok, this section is quite a big section. I can guarantee that there will be an exercise in the Midterm about this section. Sometimes, you can choose with what method you want to prove non-regularity. Sometimes you get forced to use a specific one. I will go over every single method in the next three subsections. For each subsection, there will be exercises and exam questions that force you to use the corresponding method. At the end of this section, there will be exercises and exam questions that let you choose the method yourself.
Let us begin!

7.1 prove non-regularity of language by Kolmogorov-Complexity

Given: language L

Sought: prove that L is not regular using the method of Kolmogorov-Complexity

The essence of this method is to make a prove by contradiction: we assume that L is regular and then show that under this assumption, we can find infinitely many (different) words with constant Kolmogorov-Complexity.
I am not going to repeat here how the method of the Kolmogorov-Complexity works in detail. If you need a refresher of that, go read the corresponding section in the script or read my notes from week 4. To solve the exercise, we also do not reprove the details of the method (this would eat away too much time in the exam); instead, we cite Theorem 3.1 (Satz 2.2) from the script. If you don't know exactly what the Theorem says, this is the time where you should read the script. Done? Then let's roll!

1. do a proof by contradiction: assume that L is regular
2. "smartly" find prefix-languages L_i
3. for each L_i, consider the canonically first word w_i
4. prove that the set $W := \{w_i \mid i \in \mathbb{N}\}$ is infinite (Satz 2.2)
5. use Theorem 3.1, which gives us that $\forall i \in \mathbb{N}: K(w_i) \leq c$
6. combine the fact that W is infinite and that all words in W have constant Kolmogorov-Complexity to get a contradiction $\Rightarrow$ L must not be regular.

Note: You do not have to take the first words of the prefix-languages. Sometimes, the first words are only finitely many but the second words are infinitely many. In general, you can take all k-th words $w_i^{(k)}$ of the prefix languages, as long as k is fix and constant. For general k, Theorem 3.1 (Satz 2.2) gives us that for all $i \in \mathbb{N}$: $K(w_i^{(k)}) \leq \lceil \log_2(k+1) \rceil + c$, which is constant since k is constant.

How to get full points:

1. state that you are doing a proof by contradiction and that you assume that L is regular
2. state the prefix languages L_i
3. state that you are considering the canonically k-th words of the prefix languages (for the constant fix k of your choice).
4. prove that there are infinitely many (different) words $w_i^{(k)}$. (Satz 2.2)
5. cite Theorem 3.1 and state that it holds:

$$K(w_i^{(k)}) \leq \lceil \log(k+1) \rceil + c = d$$

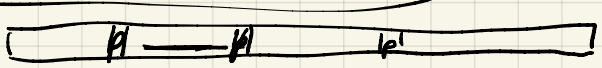
6. state that since there are infinitely many words with constant Kolmogorov-Complexity, we get a contradiction, which implies that L is non-regular.

Tipps & Tricks:

1. don't forget any of the above points (even the trivial statements e.g. stating that we do a proof by contradiction)
2. don't forget to cite Theorem 3.1 (Satz 2.2)
3. when you have trouble finding the right prefix-languages L_i, it might help to keep in mind that you want to take the canonically k-th word and that those words must be infinitely many.
4. make really 100% sure that $w_i^{(k)}$ is indeed what you think that it is.
   e.g. for the language $L := \{ww \mid w \in \{0,1\}^*\}$ and the prefix language $L_w := \{x \mid wx \in L\}$, $x = w$ might not be the first word of L_w.
5. when proving that W is infinite, it is enough to prove that the lengths of the words $w_i^{(k)}$ are strictly increasing. It is also enough to prove that there is no longest word $w_i^{(k)}$ in W.

$$\frac{n}{\log n} \overset{PS}{\geq} n \overset{\text{Anzahl}}{\geq} \frac{n}{k}$$

$$p_{i+1} - p_i \leq k \quad \forall i$$



Example Exercises: (that force you to use Kolmogorov-Complexity)

Sheet 4, 2021

Aufgabe 12

Zeigen Sie unter Vewendung der angegebenen Methode, dass die folgenden Sprachen nicht regulär sind.

(c) Verwendung der Methode der Kolmogorov-Komplexität:

$$L_3 = \{0^{\binom{2n}{n}} \mid n \in \mathbb{N}\}$$

15 Punkte

Sheet 4, 2022

Aufgabe 12

(a) Verwenden Sie die Methode der Kolmogorov-Komplexität, um zu zeigen, dass die Sprache

$$L_1 = \{0^{n^2 \cdot 2^n} \mid n \in \mathbb{N}\}$$

nicht regulär ist.

Sheet 4, 2023

Aufgabe 12

(a) Sei F_n die n-te Fibonacci-Zahl (rekursiv definiert durch $F_0 = 0$, $F_1 = 1$ und $F_{n+1} = F_n + F_{n-1}$ für alle $n \geq 1$). Verwenden Sie die Methode der Kolmogorov-Komplexität, um zu zeigen, dass folgende Sprache über dem Alphabet $\{0, 1\}$ nicht regulär ist:

$$L_1 = \{0^{F_n} \mid n \in \mathbb{N}\}$$

Sheet 5, 2023

Bonus-Aufgabe 1 (advanced!)

Wir wollen zeigen, dass das Pumping-Lemma manchmal mächtiger ist als die Methode der Kolmogorov-Komplexität, wenn es um den Nachweis der Nichtregularität gewisser Sprachen geht. Wir betrachten zu diesem Zweck die Sprache

$$L = \{0, 1\}^* - \{0^n 1^n \mid n \in \mathbb{N}\} .$$

(a) Verwenden Sie das Pumping-Lemma für reguläre Sprachen, um zu beweisen, dass die Sprache L nicht regulär ist.

(b) Beweisen Sie ohne weitere Annahmen die Aussage von Satz 3.1 für L: Es existiert eine Konstante c, so dass für alle Wörter $x, y \in \{0, 1\}^*$

$$K(y) \leq \lceil \log_2(n + 1) \rceil + c$$

gilt, falls y das n-te Wort in der Sprache L_x ist.

Sheet 5, 2024

Aufgabe 13

(a) Verwenden Sie die Methode der Kolmogorov-Komplexität, um die Nichtregularität der folgenden Sprache zu beweisen:

$$L_1 = \{ww \mid w \in \{0, 1\}^*\}$$

Exam Questions: (that force you to use Kolmogorov-Complexity)

- 2022: 5a)
- 2021: 5a)
- 2019: 2a)
- 2004: 1b)

7.2 prove non-regularity of language by Lemma 3.3.

Given: language L

Sought: prove that L is not regular using Lemma 3.3

We have already seen how to prove a lower bound on the amount of states of FSAs for a language with the help of Lemma 3.3. Do you remember the note in that chapter that if we prove a lower bound of $k=\infty$, that we actually prove non-regularity? Well, this is exactly what this task-type. Let's go!

1. do a proof by contradiction: assume that L is regular $\Rightarrow$ it exists an FSA A with $L(A)=L$. This FSA has $|Q|$ states (which is unknown but constant).
2. "smartly" find $|Q|+1$ words $w_1, \ldots, w_{|Q|+1}$
3. because of the Pigeonhole Principle, some words w_i and w_j with $i<j$ are end the same state.
4. find a suffix $z_{i,j}$ such that exactly one of the words $w_i z_{i,j}$ and $w_j z_{i,j}$ is in L (and the other isn't).
5. use Lemma 3.3 to get a contradiction $\Rightarrow$ L must not be regular.

How to get full points:

1. state that you are doing a proof by contradiction and that you assume that L is regular
2. state that since L is regular, there exists an FSA A with $L(A) = L$ (that has $|Q|$ many states).
3. write down $|Q|+1$ words $w_1, \dots, w_{|Q|+1}$ (by defining w_i)
4. state that because of the Pigeonhole Principle, there exist two words w_i and w_j with $i<j$ that end in the same state of A.
5. cite Lemma 3.3 and state that because of this Lemma, for every suffix z, the words w_iz and w_jz end in the same state of A. In particular, depending on whether the state is accepting or not, either both words are in L or none is.
6. write down the suffix $z_{i,j}$ such that exactly one of the words $w_iz_{i,j}$ and $w_jz_{i,j}$ is in L. Quickly state which one is in L and which one is not.
7. state that this is a contradiction, which implies that L is non-regular.

Tipps & Tricks:

1. don't forget any of the above points (even the trivial statements e.g. stating that we do a proof by contradiction)
2. don't forget to cite Lemma 3.3
3. if you have trouble finding the words $w_1, \dots, w_k$ it might help you to think of the following: (almost always) the reason for a language not to be regular is because there is some sort of counting to be done in order to recognize the language (e.g. $L = \{a^ib^i \mid i \in \mathbb{N}\}$, you have to count the amount of a's at the beginning and check whether the right amount of b's follow). This counting can not be done by FSAs since they have finite memory. So, usually, taking $|Q|+1$ words that have a different counter will do the job (e.g. $w_i = a^i$ for $L = \{a^ib^i \mid i \in \mathbb{N}\}$).
4. if you got the right words, usually a suffix $z_{i,j}$ for

which $w_i z_{i,j} \in L$ will also satisfy $w_j z_{i,j} \notin L$.

Example Exercises: (that force you to use Lemma 3.3)

Sheet 4, 2021

Aufgabe 12

Zeigen Sie unter Vewendung der angegebenen Methode, dass die folgenden Sprachen nicht regulär sind.

(a) Verwendung von Lemma 3.3:

$$L_1 = \{w \in \{a, b, c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

Sheet 4, 2022

Aufgabe 11

(a) Verwenden Sie eine direkte Argumentation über den Automaten (unter Verwendung von Lemma 3.3 aus dem Buch), um zu zeigen, dass die Sprache

$$L_1 = \{a^i b^j \mid i, j \in \mathbb{N} \text{ und es gibt ein } k \in \mathbb{N} \text{ so, dass } j = k \cdot i\}$$

nicht regulär ist.

Sheet 4, 2023

Aufgabe 11

(a) Verwenden Sie eine direkte Argumentation über den Automaten (unter Verwendung von Lemma 3.3), um zu zeigen, dass folgende Sprache nicht regulär ist:

$$L_1 = \{a^i b^j \mid i, j \in \mathbb{N} \text{ so dass } j < i\}$$

Sheet 4, 2024

Aufgabe 12

(a) Verwenden Sie eine direkte Argumentation über den Automaten (unter Verwendung von Lemma 3.3 aus dem Buch), um zu zeigen, dass die Sprache

$$L_1 = \{w \in \{a, b, c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

nicht regulär ist.

Exam Questions: (that force you to use Lemma 3.3)

there are none

7.3 prove non-regularity of language by Pumping-Lemma

Given: language L

Sought: prove that L is not regular using the Pumping-Lemma

Alright, we have reached the last subsection of non-regularity proofs. I hope you know what the Pumping-Lemma is all about. If not, please go and read the Script. Let's go!

1. do a proof by contradiction: assume that L is regular $\Rightarrow$ the Pumping-Lemma is applicable.

2. apply the Pumping-Lemma: it exists an $n_0 \in \mathbb{N}$ such that for all words w with $|w| \geq n_0$, it exists a decomposition $w = yxz$ such that:

   (i) $|yx| \leq n_0$

   (ii) $|x| \geq 1$

   (iii) the "pumped" words $w_i := yx^iz$ are either all in L or none is in L

3. "smartly" select a word w with $|w| \geq n_0$ such that for all decompositions $w = yxz$, there exist i and j with $i \neq j$ such that $w_i = yx^iz \in L$ and $w_j = yx^jz \notin L$. Use (i) and (ii) of the Pumping-Lemma to "get information" about how every decomposition looks like.

4. this a contradiction $\Rightarrow$ L must not be regular

How to get full points:

1. state that you are doing a proof by contradiction and that you assume that L is regular
2. state that because of the Pumping-Lemma, it exists an $n_0 \in \mathbb{N}$ such that for all words w with $|w| \geq n_0$, it exists a decomposition $w = yxz$ such that:
   (i) $|yx| \leq n_0$
   (ii) $|x| \geq 1$
   (iii) the "pumped" words $w_i := yx^iz$ are either all in L or none is in L
3. give a word w with $|w| \geq n_0$
4. give an i and a j such that (no matter how the decomposition $w = yxz$ looks like) $w_i = yx^iz \in L$ and $w_j = yx^jz \notin L$
5. state that this is a contradiction, which implies that L is non-regular.

Tipps & Tricks:

1. don't forget any of the above points (even the trivial statements e.g. stating that we do a proof by contradiction).
2. don't forget to cite (and state) the Pumping-Lemma
3. if you have trouble finding the word w it might help you to think of the following: you have to get a contradiction for every decomposition $w = yxz$ and you can only use (i) and (ii) of the Pumping-Lemma to get some information about how every decomposition looks like. So, you want that the first n_0 letters of w are in such a way that it is easy to say how x looks like.
4. remember that you can choose w but not the decomposition y, x and z.

Example Exercises: (that force you to use Pumping-Lemma)

Sheet 4, 2021

Aufgabe 12

Zeigen Sie unter Vewendung der angegebenen Methode, dass die folgenden Sprachen nicht regulär sind.

(b) Verwendung des Pumping-Lemmas:

$$L_2 = \{w \in \{0,1\}^* \mid |w|_0 \neq |w|_1\}$$

Sheet 4, 2022

Aufgabe 11

(b) Verwenden Sie das Pumping-Lemma für reguläre Sprachen, um zu zeigen, dass die Sprache

$$L_2 = \{wabw^{\mathsf{R}} \mid w \in \{a,b\}^*\}$$

nicht regulär ist. Dabei bezeichnet w^{R} die Umkehrung (Reversal) von w.

10 Punkte

Sheet 4, 2023

Aufgabe 11

(b) Verwenden Sie das Pumping-Lemma für reguläre Sprachen, um zu zeigen, dass folgende Sprache nicht regulär ist:

$$L_2 = \{wbbbv \mid w, v \in \{a,b\}^* \text{ mit } |w|_a = |v|_a\}$$

10 Punkte

Sheet 5, 2024

Aufgabe 12

(b) Verwenden Sie das Pumping-Lemma für reguläre Sprachen, um zu zeigen, dass die Sprache

$$L_2 = \{wabw^{\mathsf{R}} \mid w \in \{a,b\}^*\}$$

nicht regulär ist. Dabei bezeichnet w^{R} die Umkehrung (Reversal) von w.

10 Punkte

Exam Questions: (that force you to use Pumping-Lemma)

there are none 

The next 2 pages contain exercises in which you can choose the method yourself.

Example Exercises: (that let you choose the method yourself)

Sheet 5, 2021

Aufgabe 13

Zeigen Sie, dass die folgenden beiden Sprachen nicht regulär sind.

(a) $L_1 = \{u\#v\#w \mid u, v, w \in \{0,1\}^+ \text{ und Nummer}(u) \cdot \text{Nummer}(v) = \text{Nummer}(w)\}$

(b) $L_2 = \{0^p \mid p \text{ ist eine Primzahl}\}$

10 Punkte

Sheet 5, 2022

Aufgabe 12

(b) Verwenden Sie eine beliebige der in der Vorlesung vorgestellten Methoden, um zu zeigen, dass die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid |u|_0 \leq |u|_1 \text{ für alle Präfixe } u \text{ von } w\}$$

nicht regulär ist.

10 Punkte

Sheet 6, 2022

Aufgabe 13

Sei $L = \{w_i \mid i \in \mathbb{N}\} \subseteq \{0\}^*$ eine unendliche Sprache mit

$$|w_{i+1}| - |w_i| \geq \log_2 \log_2 i$$

für alle $i \in \mathbb{N}$. Zeigen Sie, dass L nicht regulär ist. **10 Punkte**

Sheet 4, 2023

Aufgabe 12

(b) Verwenden Sie eine beliebige der in der Vorlesung vorgestellten Methoden, um zu zeigen, dass folgende Sprache nicht regulär ist:

$$L_2 = \{w \in \{a,b,c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

10 Punkte

Sheet 5, 2023

Aufgabe 14

Im Folgenden sei L immer eine beliebige reguläre Sprache über einem beliebigen Alphabet Σ, wobei $\# \notin \Sigma$ ein zusätzliches unbenutztes Symbol ist. Beweisen oder widerlegen Sie folgende Aussagen (indem Sie entweder einen möglicherweise nichtdeterministischen endlichen Automaten für L_1 oder L_2 angeben und informell dessen Korrektheit begründen, oder indem Sie ein konkretes Gegenbeispiel für L und Σ angeben und dann die Nichtregularität der zugehörigen Sprache L_1 oder L_2 beweisen).

(a) Die Sprache $L_1 = \{v\#w^R \mid v, w \in L\}$ ist regulär für jedes reguläre L und Σ.

(b) Die Sprache $L_2 = \{w\#w^R \mid w \in L\}$ ist regulär für jedes reguläre L und Σ.

10 Punkte

Sheet 5, 2024

Aufgabe 13

(b) Zeigen Sie, dass folgende Sprache nicht regulär ist:

$$L_2 = \{0^p \mid p \text{ ist eine Primzahl}\}$$

5+10 Punkte

Exam Questions: (that let you choose the method yourself)

- 2023: 4
- 2022: 5b)
- 2021: 5b)
- 2019: 2b)
- 2018: 2
- 2017: 2
- 2016: 3
- 2015: 4
- 2014: 2
- 2013: 2
- 2012: 3
- 2011: 2
- 2010: 4
- 2008: 2
- 2007: 3
- 2004: 3d)

8. (un)countability

Given: set S

Sought: prove that S is (un)countable

(Un)countability is all about finding injections from one set to another. When there is an injection from S_1 to S_2, then $|S_1| \leq |S_2|$. So, for proving that a set S is countable, one has to find an injection from S to $\mathbb{N}$. For proving that a set S is uncountable, one has to find an injection from an uncountable set U to S.

For proving countability:

1. find an injection from S to $\mathbb{N}$

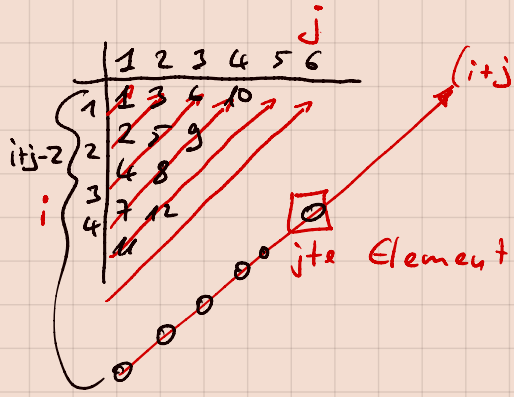
How to get full points:

1. give an injective function $f: S \to \mathbb{N}$

Tipps & Tricks:

1. if you have trouble finding the function f, it might help to think about a way to describe every element of S by two integers i and j (e.g. for the pralines, every praline can be described by being the ith praline of round j). Then, you can find a function $f: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ like we saw in the exercise session:

$\mathbb{N} \times \mathbb{N}$ abzählen



$$f(i,j) = \sum_{k=1}^{i+j-2} \#\text{Zahlen auf } k\text{ter Diag.} + j$$

$$= \sum_{k=1}^{i+j-2} k + j$$

$$= \frac{(i+j-2)(i+j-1)}{2} + j = \binom{i+j-1}{2} + j$$

recall the trick with counting the diagonals and the numbers on the diagonals.

For proving uncountability:

1. find an injection from an uncountable set U to S.

How to get full points:

1. give an uncountable set U and quickly state that it is uncountable
2. give an injective function $f: U \to S$

Tipps & Tricks:

1. the set U is (probably) going to be either $[0,1]$ or $\mathcal{P}(\Sigma^*)$, where Σ is an alphabet.
2. check out all exercises and solutions to see the few tricks that there are to define the function $f: U \to S$.

Example Exercises:

Sheet 6, 2021

Aufgabe 16

(a) Wir betrachten das unendliche Hotel Hilbert. Alle Zimmer des Hotels sind belegt und es kommen drei unendliche Busse mit neuen Gästen an. Einige der neuen Gäste haben unterschiedliche Reservierungswünsche: Die Gäste aus Bus 1 wollen unbedingt ungerade Zimmernummern haben, jeder Gast aus Bus 2 will unbedingt mindestens einen Zimmernachbarn haben, der auch mit Bus 2 angekommen ist. Beschreiben Sie eine Strategie, nach der der Portier alle Gäste unterbringen kann und dabei die Wünsche der Gäste berücksichtigt.

(b) Wir betrachten wieder das Hotel Hilbert und nehmen diesmal an, dass das Hotel ganz leer ist. Da das nächtliche Umziehen für die Gäste sehr unbequem ist, wollen wir im Folgenden ohne Umziehen auskommen. Wir nehmen an, dass zu einer beliebigen abzählbaren Anzahl von Zeitpunkten jeweils eine Gruppe mit einer endlichen oder abzählbar unendlichen Anzahl von Gästen ankommt. Innerhalb jeder Gruppe sind die ankommenden Gäste durchnummeriert. Beschreiben Sie eine Strategie, mit der der Portier die Gäste so auf die Zimmer verteilen kann, dass niemals ein Gast umziehen muss.

10 Punkte

Sheet 8, 2022

Aufgabe 24

Ein Frosch hüpft auf der Integer-Zahlengeraden entlang. Er startet am Punkt $u \in \mathbb{Z}$ und hüpft jede Nacht die gleiche Distanz $s \in \mathbb{Z} - \{0\}$ in die gleiche Richtung (negative Distanzen bedeuten Hüpfer nach links), tagsüber verbleibt er auf der erreichten Zahl und schläft.

Sie wollen den Frosch fangen, kennen aber weder Richtung noch Geschwindigkeit des Frosches, auch die Zahl, bei der er am ersten Abend beginnt, ist Ihnen unbekannt. Sie können sich dazu nicht einfach irgendwohin stellen und warten, bis der Frosch vorbeikommt – schliesslich können Sie ihn nachts nicht sehen, ausserdem könnte er ja auch in die andere Richtung hüpfen und nie bei Ihnen vorbeikommen. Stattdessen müssen Sie sich jeden Tag für eine Zahl entscheiden und können dann nachsehen, ob sich der Frosch dort befindet. Falls er dort schläft, können Sie ihn einfach fangen.

Mit welcher Strategie können Sie die besuchten Zahlen wählen, um den Frosch mit Sicherheit nach endlicher Zeit zu fangen? **10 Punkte**

Sheet 6, 2023

Aufgabe 16

(a) Wir betrachten das unendliche Hotel Hilbert. Alle Zimmer des Hotels sind belegt und es kommen drei unendliche Busse mit neuen Gästen an. Einige der neuen Gäste haben unterschiedliche Reservierungswünsche: Die Gäste aus Bus 1 wollen unbedingt ungerade Zimmernummern haben, die Gäste aus Bus 2 wollen mindestens eine Zimmernachbarin oder einen Zimmernachbarn haben, die oder der auch mit Bus 2 angekommen ist. Beschreiben Sie eine Strategie, nach der der Portier alle Gäste unterbringen kann und dabei die Wünsche der Gäste berücksichtigt.

(b) Wir betrachten wieder das Hotel Hilbert und nehmen diesmal an, dass das Hotel ganz leer ist. Da das nächtliche Umziehen für die Gäste sehr unbequem ist, wollen wir im Folgenden ohne Umziehen auskommen. Wir nehmen an, dass zu einer beliebigen abzählbaren Anzahl von Zeitpunkten jeweils eine Gruppe mit einer endlichen oder abzählbar unendlichen Anzahl von Gästen ankommt. Innerhalb jeder Gruppe sind die ankommenden Gäste durchnummeriert. Beschreiben Sie eine Strategie, mit der der Portier die Gäste so auf die Zimmer verteilen kann, dass niemals ein Gast umziehen muss.

10 Punkte

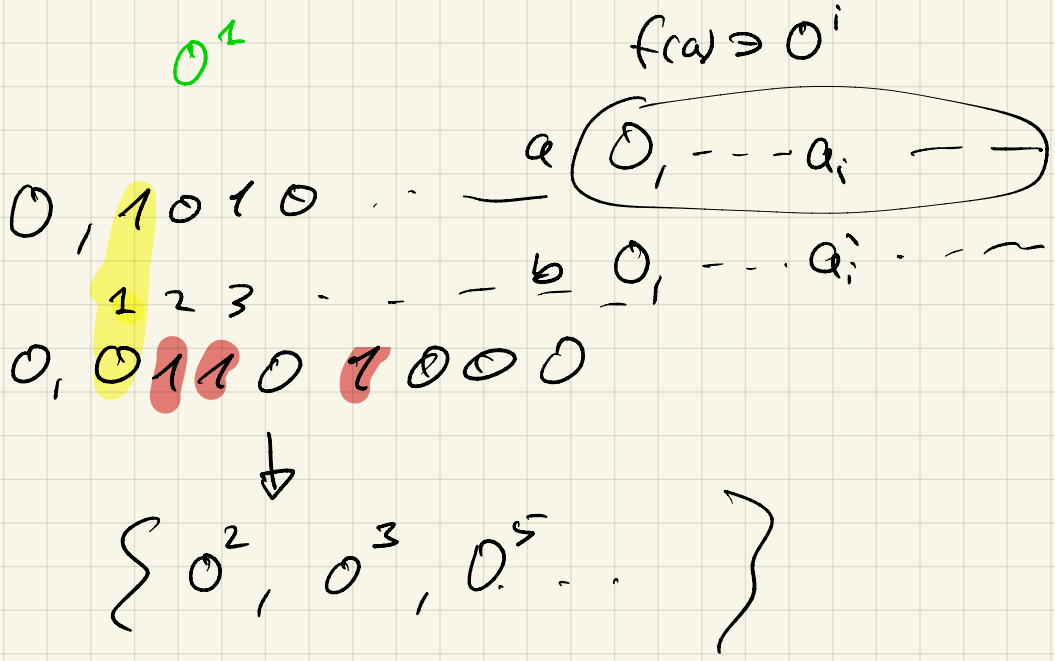
Aufgabe 17

Zeigen Sie, dass $|[0,1]| = |\mathcal{P}(\mathbb{Q}^+)|$ gilt. Dabei bezeichnet $\mathcal{P}(S) = \{S' \mid S' \subseteq S\}$ die Potenzmenge von einer gegebenen Menge S.

10 Punkte

Exam Questions

- 2021: 6
- 2015: 6



$$\{0^2, 0^3, 0^5 \ldots\}$$

$$f(0, a_1 \ldots a_i \ldots) = \{0^i \mid a_i = 1\} \in P(\{0\}^*)$$

9. theory exercise

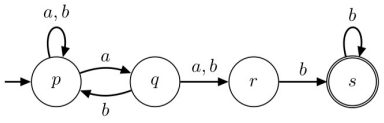
Well, as you can imagine there is not too much to say about this type of exercise other than read the Script lol. Therefore, I will only give you exercises and no tutorial... But I am sure that all of you of course read the Script and knows every single Theorem and corresponding Proof :)

Example Exercises

Sheet 5, 2021

Aufgabe 14

(b) Verwenden Sie die Potenzmengenkonstruktion, um den folgenden nichtdeterministischen endlichen Automaten in einen äquivalenten deterministischen Automaten umzuwandeln.



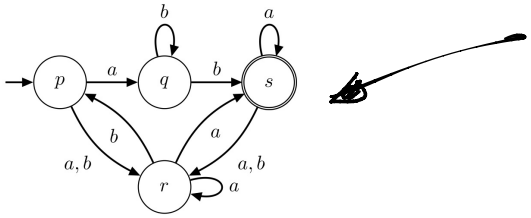
Geben Sie den von Ihnen konstruierten Automaten in Diagrammdarstellung an. Dabei können Sie alle nicht erreichbaren Zustände weglassen.

10 Punkte

Sheet 5, 2022

Aufgabe 15

Verwenden Sie die Potenzmengenkonstruktion, um den folgenden nichtdeterministischen endlichen Automaten in einen äquivalenten deterministischen endlichen Automaten umzuwandeln.

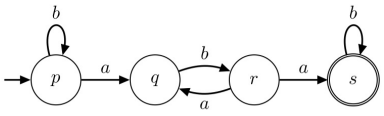


Geben Sie den von Ihnen konstruierten Automaten in Diagrammdarstellung an. Dabei können Sie alle nicht erreichbaren Zustände weglassen. **10 Punkte**

Sheet 5, 2022

Aufgabe 13

(b) Verwenden Sie die Potenzmengenkonstruktion, um den folgenden nichtdeterministischen endlichen Automaten in einen äquivalenten deterministischen endlichen Automaten umzuwandeln.



Geben Sie den von Ihnen konstruierten Automaten in Diagrammdarstellung an. Dabei können Sie alle nicht erreichbaren Zustände weglassen.

10 Punkte

Exam Questions

- 2023: 3
- 2022: 4
- 2021: 2, 4
- 2016: 5
- 2012: 1a)
- 2004: 3a), b), c)

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 2

Zürich, 8. Oktober 2021

Lösung zu Aufgabe 4

(a) Wir definieren die Folge $(x_n)_{n=1}^{\infty}$ als $x_n = 0^{2^n}$ für jedes $n \in \mathbb{N} - \{0\}$. Es ist offenbar, dass alle Wörter x_n paarweise unterschiedlich sind.

Wir geben jetzt für jedes $n \in \mathbb{N} - \{0\}$ ein Programm an, das x_n erzeugt:

```
begin
  s := n;
  j := 1;
  for i := 1 to s do
    j := j * 2;
  for i := 1 to j do
    write(0);
end.
```

Dieses Programm berechnet zuerst in einer Schleife $j = 2^n$. In einer weiteren Schleife gibt das Programm dann 2^n Mal das Zeichen 0 aus, was genau das Wort x_n ergibt.

Der einzige Teil des Maschinencodes des Programms, der von x_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Programmcode hat eine konstante Länge. Also ist die binäre Länge des Programms höchstens $\lceil \log_2(n+1) \rceil + c$ für eine Konstante c.

Damit lässt sich die Kolmogorov-Komplexität von x_n von oben abschätzen durch

$$K(x_n) \leq \lceil \log_2(n+1) \rceil + c \leq \log_2 n + (c+1)$$

für eine Konstante c.

Die Länge von x_n ist $|x_n| = |0^{2^n}| = 2^n$. Also gilt $\log_2 \log_2 |x_n| = \log_2 n$ und somit erhalten wir eine obere Schranke von

$$K(x_n) \leq \log_2 \log_2 |x_n| + (c+1)$$

für die Kolmogorov-Komplexität von x_n.

(b) Wir zeigen indirekt, dass es keine solche Folge $(x_n)_{n=1}^{\infty}$ von paarweise unterschiedlichen Wörtern gibt. Nehmen wir an, $(x_n)_{n=1}^{\infty}$ sei eine solche Folge. Dann gibt es eine Konstante $c \in \mathbb{N}$, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq \log_2 \sqrt{n} + c$$

gilt.

Für ein fixes $n \in \mathbb{N} - \{0\}$ und $t := \log_2 \sqrt{n} + c$ gibt es höchstens

$$\sum_{i=0}^{t} 2^i = 2^{t+1} - 1 = 2^{\log_2 \sqrt{n}+c+1} - 1 = \sqrt{n} \cdot 2^{c+1} - 1$$

Maschinencodes der Länge höchstens t, also können höchstens $\sqrt{n} \cdot 2^{c+1} - 1$ paarweise unterschiedliche Wörter durch Maschinencodes der Länge höchstens t erzeugt werden. Für ein genügend grosses n, das nur von der Konstante c abhängt, gilt $\sqrt{n} \cdot 2^{c+1} - 1 < n$, was im Widerspruch dazu steht, dass die n Wörter $x_1, \ldots, x_n$ paarweise unterschiedlich sind und $K(x_i) \leq t$ für alle $i \in \{1, \ldots, n\}$ gilt, d.h., $x_1, \ldots, x_n$ durch Maschinencodes der Länge höchstens t erzeugt werden.

Lösung zu Aufgabe 5

Wir beweisen die Aussage indirekt. Somit gebe es also unendlich viele Zahlen in der Menge $\{n^2 \mid n \in \mathbb{N}\}$, die als zufällig angesehen werden können. Nach Definition bedeutet dies, dass es unendlich viele $n \in \mathbb{N}$ gibt, sodass

$$K(n^2) \geq \lceil \log_2(n^2 + 1) \rceil - 1 \geq 2 \cdot \log_2 n - 1 \tag{1}$$

gilt.

Ferner lässt sich die Zahl n^2 für jedes $n \in \mathbb{N}$ mit einem Programm C_n berechnen, das die binäre Kodierung von n enthält, die Zahl n^2 berechnet und anschliessend ausgibt. Alle Bestandteile dieses Programms mit Ausnahme der Kodierung von n haben eine konstante Länge, also hat C_n die binäre Länge $\lceil \log_2(n+1) \rceil + c \leq \log_2 n + (c+1)$ für eine Konstante c. Es folgt, dass

$$K(n^2) \leq \log_2 n + (c + 1) \tag{2}$$

gilt.

Aus den beiden Schranken (1) und (2) für die Kolmogorov-Komplexität ergibt sich, dass

$$2 \cdot \log_2 n - 1 \leq \log_2 n + (c + 1)$$

für unendlich viele $n \in \mathbb{N}$ gelten muss. Hieraus folgt wiederum, dass für diese n auch

$$\log_2 n \leq c + 2$$

gelten muss, was unmöglich ist, da c eine Konstante ist und $\log_2 n$ mit n beliebig wächst. Somit ist unsere Annahme falsch und die zu beweisende Aussage folgt.

Lösung zu Aufgabe 6

Offenbar gibt es ein Programm A_L, das für ein gegebenes Wort $x \in \Sigma_{\text{bool}}^*$ entscheidet, ob $x \in L$ gilt. Nach Satz 2.2 aus dem Buch gilt damit für das kanonisch n-te Wort x_n aus L

$$K(x_n) \leq \lceil \log_2(n + 1) \rceil + c \leq \log_2 n + (c + 1)$$

für eine von n unabhängige Konstante c.

Wir schätzen nun n in Abhängigkeit von $|x_n|$ ab, indem wir die Anzahl Wörter der Form $1^i0^j1^k$ abschätzen, deren Länge $i+j+k$ höchstens $|x_n|$ beträgt. Die ersten n Wörter $x_1, \ldots, x_n$ aus L in der kanonischen Reihenfolge haben die Form $1^i0^j1^k$ mit $i+j+k \leq |x_n|$, aus $i+j+k \leq |x_n|$ folgt, dass $i, j, k \leq |x_n|$ gilt, und deshalb gilt $n \leq |x_n|^3$. Daraus folgt schliesslich, dass

$$K(x_n) \leq \log_2 n + (c+1) \leq \log_2 |x_n|^3 + (c+1) = 3 \cdot \log_2 |x_n| + (c+1)$$

gilt.

Bemerkung: Man könnte eine alternative Lösung vorschlagen, in der ein Programm einfach die drei Werte i, j und k als variable Teile enthält und daraus die Ausgabe berechnet. Leider ist dies nicht so einfach möglich, wie es zunächst scheint. Weil i, j und k beliebige natürliche Zahlen sein können, kann ihre binäre Darstellung in manchen Fällen mit der ASCII-Darstellung gültiger Programmfragmente übereinstimmen. Wir müssten also dem Compiler mitteilen können, wo genau die Werte i, j und k im Programm stehen, die nicht als Programmtext interpretiert werden sollen. Für eine einzelne Zahl i können wir dies tun, indem wir die Längen der konstanten Programmteile davor und dahinter mit konstantem Platz (z. B. in ASCII-Kodierung) angeben. Ab zwei Zahlen variabler Länge ist aber auch mit der Angabe von mehreren solchen konstanten Längen eine eindeutige Interpretation des Programms noch nicht gewährleistet. Es könnte passieren, dass es mehrere Stellen gibt, an denen man die Zahlen voneinander trennen kann.

Deshalb muss man die Zahlen i, j und k selbstbeschränkend kodieren (vgl. die Kodierung im Beweis des Pimzahlsatzes), was aber allgemein zu einer grösseren Länge des Maschinencodes führt. Die einfachste selbstbeschränkende Kodierung liefert einen Maschinencode der Länge $2 \cdot (\log_2 i + \log_2 j + \log_2 k) + c$ für eine Konstante c, was eine Länge bis zu $6 \cdot \log_2 |x_n| + c$ ergibt.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 3

Zürich, 15. Oktober 2021

Lösung zu Aufgabe 7

Nach Definition 2.19 aus dem Buch heisst ein Wort $w \in \{0,1\}^*$ zufällig, wenn $K(w) \geq |w|$ gilt. Wir zeigen im Folgenden mit einem einfachen Abzählargument, dass für mindestens die Hälfte aller Wörter $w \in \{0,1\}^*$ mit $0 \leq |w| \leq n$ sogar gilt, dass $K(w) \geq n$. Damit folgt unmittelbar die Behauptung.

Es gibt $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ Wörter über $\{0,1\}$ der Länge höchstens n.

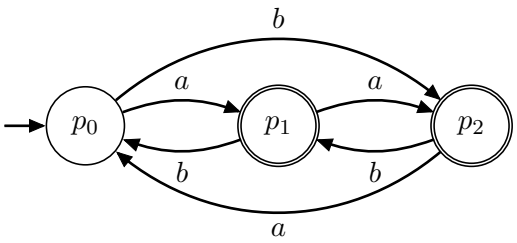
Auf der anderen Seite gibt es $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ Wörter über $\{0,1\}$ der Länge höchstens $n-1$. Damit gibt es auch höchstens $2^n - 1$ Binärkodierungen von Programmen der Länge höchstens $n-1$. Da zwei unterschiedliche Wörter von unterschiedlichen Programmen ausgegeben werden müssen, gibt es also höchstens $2^n - 1 \leq \frac{1}{2} \cdot (2^{n+1} - 1)$ Wörter mit einer Kolmogorov-Komplexität von höchstens $n-1$. Damit folgt sofort die Behauptung.

Lösung zu Aufgabe 8

(a) Es gilt

$$\begin{aligned} L_1 &= \{w \in \{a,b\}^* \mid (|w|_a + 2 \cdot |w|_b + 1) \bmod 3 \neq 1\} \\ &= \{w \in \{a,b\}^* \mid (|w|_a - |w|_b) \bmod 3 \neq 0\}. \end{aligned}$$

Der folgende endliche Automat akzeptiert die Sprache L_1:



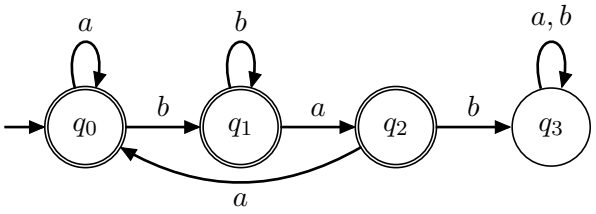
Dieser Automat zählt einfach die Buchstaben in der Eingabe modulo 3, wobei entsprechend der geforderten Bedingung jedes a positiv und jedes b negativ gezählt werden. Damit gilt für die Klassen, dass

$$\mathrm{Kl}[p_i] = \{w \in \{a,b\}^* \mid (|w|_a - |w|_b) \bmod 3 = i\}$$

für $i \in \{0,1,2\}$.

(b) Der folgende endliche Automat akzeptiert die Sprache

$$L_2 = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } bab \text{ nicht}\}\,.$$



Der Automat arbeitet mit denselben Transitionen wie ein Automat, der einfach das Muster bab in der Eingabe sucht. Im Zustand q_i wurde an der aktuellen Position der Eingabe bereits ein Präfix der Länge i des Musters gefunden. Weil der Automat alle Wörter akzeptieren soll, die das Muster nicht enthalten, sind alle Zustände bis auf q_3 akzeptierend.

Es ergeben sich die folgenden Klassen für die Zustände:

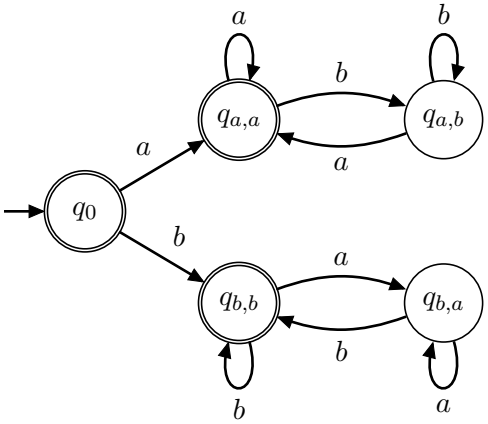
$$\begin{aligned}
\mathrm{Kl}[q_3] &= \{a,b\}^* - L_2\,,\\
\mathrm{Kl}[q_2] &= \{xba \mid x \in \{a,b\}^*\} \cap L_2\,,\\
\mathrm{Kl}[q_1] &= \{xb \mid x \in \{a,b\}^*\} \cap L_2\,,\\
\mathrm{Kl}[q_0] &= \{a,b\}^* - \bigcup_{i=1}^{3} \mathrm{Kl}[q_i]\,.
\end{aligned}$$

Man bemerke, dass es oftmals hilfreich ist, die Reihenfolge der Klassenbeschreibungen geschickt auszuwählen.

Lösung zu Aufgabe 9

(a) Der folgende endliche Automat akzeptiert die Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}\,.$$



Beim Lesen der Buchstaben in w von links nach rechts kommt ab genau für jeden Wechsel von a nach b einmal als Teilwort vor und analog ba genau einmal für jeden Wechsel von b nach a. Somit kommen ab und ba genau dann gleich oft in w als

Teilwort vor, wenn w entweder das leere Wort λ ist oder w mit demselben Buchstaben beginnt dem es endet. Letztere Bedingung ist insbesondere für $w = a$ und für $w = b$ erfüllt. Wenn w mit a beginnt, prüft der Automat in den beiden oberen Zuständen, ob der zuletzt gelesene Buchstabe mit dem zuerst gelesenen übereinstimmt. Wenn w mit b beginnt, tut er dasselbe in den beiden unteren Zuständen.

(b) Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned} \mathrm{Kl}[q_0] &= \{\lambda\} \text{ und} \\ \mathrm{Kl}[q_{x,y}] &= \{w \in \{a,b\}^* \mid w \text{ beginnt mit } x \text{ und endet mit } y\} \text{ für alle } x, y \in \{a,b\}. \end{aligned}$$

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 4

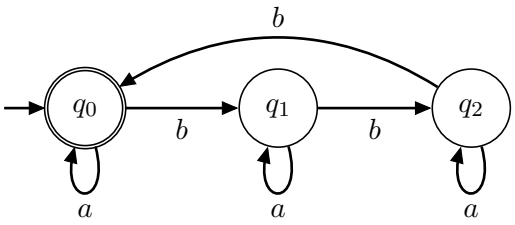
Zürich, 22. Oktober 2021

Lösung zu Aufgabe 10

Die in der Aufgabenstellung gegebene Sprache L lässt sich schreiben als $L = L_1 \cup L_2$ mit

$$\begin{aligned} L_1 &= \{w \in \{a,b\}^* \mid |w|_a \bmod 3 = |w| \bmod 3\} \\ &= \{w \in \{a,b\}^* \mid |w|_a \bmod 3 = (|w|_a + |w|_b) \bmod 3\} \\ &= \{w \in \{a,b\}^* \mid |w|_b \bmod 3 = 0\} \\ L_2 &= \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ und } w \text{ endet mit } b\}\,. \end{aligned}$$

Für die Sprache L_1 lässt sich der folgende Automat A_1 konstruieren:

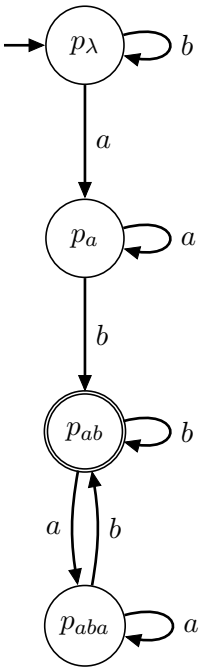


Dieser Automat zählt in seinen Zuständen den Wert von $|w|_b$ modulo 3. Es gilt für $i \in \{0,1,2\}$, dass

$$\mathrm{Kl}[q_i] = \{w \in \{a,b\}^* \mid |w|_b \bmod 3 = i\}\,.$$

(bitte wenden)

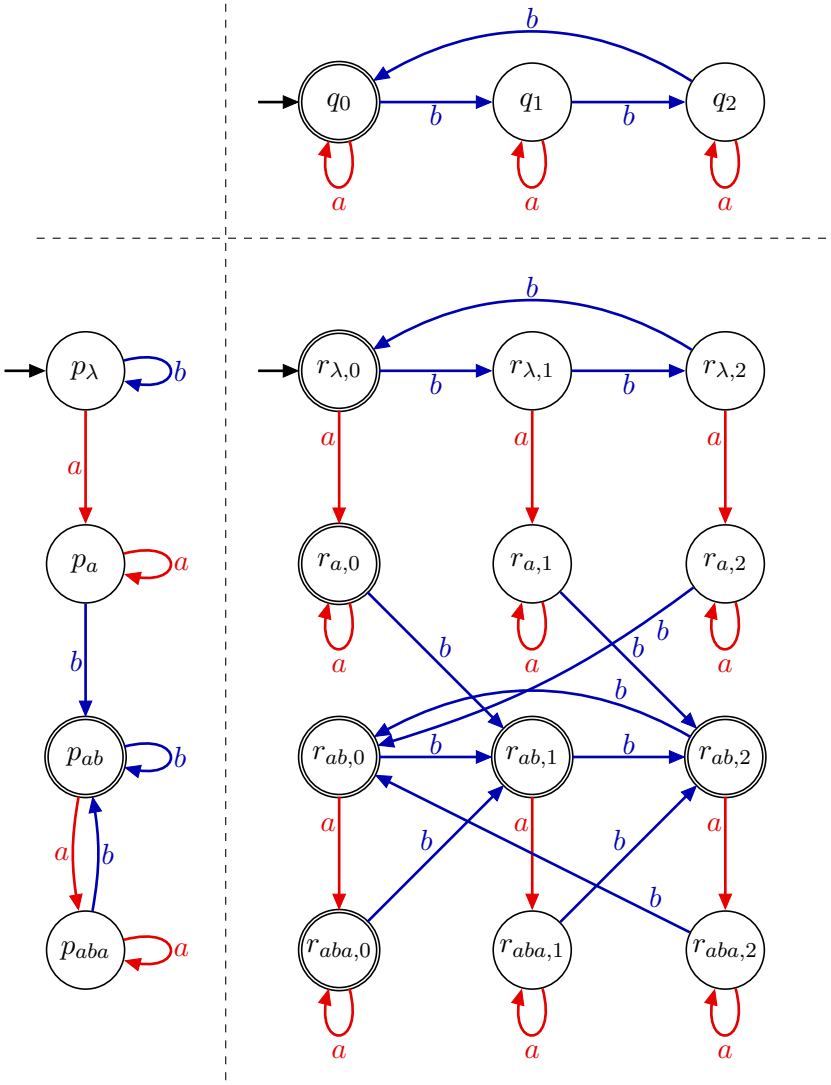
Für die Sprache L_2 lässt sich der folgende Automat A_2 konstruieren:



Die Zustände p_z für $z \in \{\lambda, a, ab\}$ geben an, welches längste Präfix des gesuchten Musters ab aktuell gelesen wurde, p_{ab} ist der akzeptierende Zustand. Der Zustand p_{aba} wird erreicht, wenn das Muster ab bereits gefunden wurde und das aktuell gelesene Präfix des Wortes mit a endet. Damit ergeben sich die folgenden Klassen:

$$\begin{aligned}
\mathrm{Kl}[p_\lambda] &= \{b\}^* \,, \\
\mathrm{Kl}[p_a] &= \{xay \mid x \in \{b\}^* \text{ und } y \in \{a\}^*\} \,, \\
\mathrm{Kl}[p_{ab}] &= L_2 \,, \\
\mathrm{Kl}[p_{aba}] &= \{a, b\}^* - \bigcup_{z \in \{\lambda, a, ab\}} \mathrm{Kl}[p_z] \,.
\end{aligned}$$

Mit der Methode des modularen Entwurfs kann man nun aus A_1 und A_2 den folgenden Produktautomaten A konstruieren, der die Sprache $L = L_1 \cup L_2$ akzeptiert. Zur einfacheren Darstellung verwenden wir die Notation $r_{x,y} = \langle p_x, q_y \rangle$. Weil dies ein Produktautomat für die Vereinigung von zwei Sprachen ist, sind alle Zustände akzeptierend, die einen akzeptierenden Zustand aus einem der beiden Teilautomaten enthalten, also die zu q_0 gehörige Spalte und die zu p_{ab} gehörige Zeile.



Lösung zu Aufgabe 11

Um zu zeigen, dass jeder endliche Automat, der die Sprache L akzeptiert, mindestens 5 Zustände benötigt, bestimmen wir 5 Wörter $w_1, \ldots, w_5$ und zeigen, dass diese in einem solchen Automaten in 5 paarweise verschiedene Zustände führen müssen. Falls es einen endlichen Automaten gibt, den das Lesen von zwei verschiedenen Wörtern w_i und w_j in denselben Zustand führt, dann gilt nach Lemma 3.3 aus dem Buch für jedes $z \in \Sigma^*$, dass dann auch $w_i z$ und $w_j z$ den Automaten in denselben Zustand führen. Wir zeigen, dass dies für einen Automaten mit weniger als 5 Zuständen nicht möglich ist, indem wir für je zwei verschiedene Wörter w_i und w_j ein Wort $z_{i,j}$ angeben, so dass

$$w_i z_{i,j} \in L(A) \iff w_j z_{i,j} \notin L(A)\,. \tag{1}$$

Wir wählen die fünf Wörter $w_1 = \lambda$, $w_2 = a$, $w_3 = b$, $w_4 = ab$ und $w_5 = ba$.

(bitte wenden)

Die folgende Tabelle gibt für alle Paare (w_i, w_j) mit $i < j$ jeweils ein Wort $z_{i,j}$ an.

$z_{i,j}$	$w_2 = a$	$w_3 = b$	$w_4 = ab$	$w_5 = ba$
$w_1 = \lambda$	b	a	λ	λ
$w_2 = a$	$-$	a	λ	λ
$w_3 = b$	$-$	$-$	λ	λ
$w_4 = ab$	$-$	$-$	$-$	a

Es ist einfach zu sehen, dass diese Wörter die Bedingung (1) erfüllen, zum Beispiel ist $w_4 z_{4,5} = aba \in L(A)$, aber $w_5 z_{4,5} = baa \notin L(A)$.

Lösung zu Aufgabe 12

(a) Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache

$$L_1 = \{w \in \{a, b, c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

nicht regulär ist. Angenommen, L_1 sei regulär. Dann gibt es einen Automaten $A = (Q, \{a, b, c\}, \delta, q_0, F)$ mit $L(A) = L_1$. Sei $m = |Q|$. Wir betrachten die Wörter

$$\lambda, abc, (abc)^2, \ldots, (abc)^m \,.$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A Zustände hat, gibt es $i, j \in \{0, \ldots, m\}$ mit $i \neq j$, so dass

$$\hat{\delta}(q_0, (abc)^i) = \hat{\delta}(q_0, (abc)^j) \,.$$

Nach Lemma 3.3 gilt nun für alle $z \in \{a, b, c\}^*$, dass

$$(abc)^i z \in L_1 \iff (abc)^j z \in L_1.$$

Die Wahl von $z = (bac)^i$ führt aber zum Widerspruch, weil

$$(abc)^i z = (abc)^i (bac)^i \in L_1$$

und $(abc)^j z = (abc)^j (bac)^i \notin L_1$ gilt. Also ist die Annahme falsch und L_1 ist nicht regulär.

(bitte wenden)

(b) Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid |w|_0 \neq |w|_1\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0,1\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

1. $|yx| \leq n_0$,
2. $|x| \geq 1$ und
3. entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = 0^{n_0}1^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = 0^l$ und $x = 0^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$, also insbesondere $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \notin L_2$, gilt nach (iii), dass auch

$$\{yx^kz \mid k \in \mathbb{N}\} = \{0^{n_0+(k-1)m}1^{n_0} \mid k \in \mathbb{N}\} \cap L_3 = \emptyset\,.$$

Dies ist aber ein Widerspruch, da $yx^2z = 0^{n_0+m}1^{n_0} \in L_2$. Also ist die Annahme falsch und L_2 ist nicht regulär.

(c) Wir verwenden die Methode der Kolmogorov-Komplexität, um zu zeigen, dass die Sprache

$$L = L_3 = \{0^{\binom{2n}{n}} \mid n \in \mathbb{N}\}$$

nicht regulär ist. Angenommen, L_3 sei regulär. Für alle $n \geq 1$ definieren wir

$$\Delta_n := \binom{2(n+1)}{n+1} - \binom{2n}{n} = \binom{2n}{n} \cdot \left(\frac{(2n+1)(2n+2)}{(n+1)^2} - 1 \right) = \binom{2n}{n} \cdot \left(3 - \frac{2}{n+1}\right).$$

Für jedes $m \in \mathbb{N}$ ist also $0^{\Delta_m - 1}$ das erste Wort in der Sprache

$$L_{0^{\binom{2m}{m}+1}} = \{y \mid 0^{\binom{2m}{m}+1}y \in L\}\,.$$

Nach Satz 3.1 aus dem Buch existiert eine Konstante c, unabhängig von m, so dass

$$K(0^{\Delta_m - 1}) \leq \lceil \log_2(1+1) \rceil + c = 1 + c\,.$$

Da es nur endlich viele Programme der konstanten Länge höchstens $1 + c$ gibt, aber unendlich viele Wörter der Form $0^{\Delta_m - 1}$, ist dies ein Widerspruch. Also ist die Annahme falsch und L_3 ist nicht regulär.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 5

Zürich, 29. Oktober 2021

Lösung zu Aufgabe 13

(a) Sei $L_1 = \{u\#v\#w \mid u, v, w \in \{0,1\}^+ \text{ und } \text{Nummer}(u) \cdot \text{Nummer}(v) = \text{Nummer}(w)\}$. Wir zeigen die Nichtregularität von L_1 mit Hilfe von Lemma 3.3 sowie mit dem Pumping-Lemma. Da L_1 über einem Alphabet der Grösse 3 definiert ist, ist die Methode der Kolmogorov-Komplexität nicht unmittelbar anwendbar.

Beweis mit Hilfe von Lemma 3.3. Angenommen, L_1 sei regulär. Dann gibt es einen Automaten $A_1 = (Q, \{0,1,\#\}, \delta, q_0, F)$ mit $L(A_1) = L_1$. Sei $m = |Q|$. Wir betrachten die Wörter

$$1\#1\#, 1^2\#1\#, 1^3\#1\#, \ldots, 1^{m+1}\#1\#\,.$$

Weil dies $m+1$ Wörter sind, also mehr Wörter als A_1 Zustände hat, gibt es $i, j \in \{1, \ldots, m+1\}$ mit $i < j$, so dass

$$\hat{\delta}(q_0, 1^i\#1\#) = \hat{\delta}(q_0, 1^j\#1\#).$$

Nach Lemma 3.3 gilt nun für alle $z \in \{0,1,\#\}^*$, dass

$$1^i\#1\#z \in L_1 \iff 1^j\#1\#z \in L_1.$$

Die Wahl von $z = 1^i$ führt aber zu einem Widerspruch, weil $1^i\#1\#z = 1^i\#1\#1^i \in L_1$ und $1^j\#1\#z = 1^j\#1\#1^i \notin L_1$ gilt. Also ist die Annahme falsch und L_1 ist nicht regulär.

Beweis mit Hilfe des Pumping-Lemmas Angenommen, L_1 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0,1,\#\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

(i) $|yx| \leq n_0$,
(ii) $|x| \geq 1$ und
(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_1$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L_1 = \emptyset$.

Wir wählen das Wort $w = 1^{n_0}\#1\#1^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = 1^l$ und $x = 1^m$ für $l, m \in \mathbb{N}$ mit $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_1$, gilt nach (iii), dass auch

$$\{yx^kz \mid k \in \mathbb{N}\} = \{1^{n_0+(k-1)\cdot m}\#1\#1^{n_0} \mid k \in \mathbb{N}\} \subseteq L_1\,.$$

Dies ist aber ein Widerspruch, da $yx^2z = 1^{n_0+m}\#1\#1^{n_0} \notin L_1$. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Sei $L_2 = \{0^p \mid p \in \mathbb{N} \text{ ist eine Primzahl}\}$. Wir zeigen die Nichtregularität von L_2 zunächst mit Hilfe des Pumping-Lemmas.

Beweis mit Hilfe des Pumping-Lemmas Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

(i) $|yx| \leq n_0$,
(ii) $|x| \geq 1$ und
(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = 0^p$ für eine Primzahl $p \geq n_0$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^kz \mid k \in \mathbb{N}\} = \{0^{p+(k-1)\cdot m} \mid k \in \mathbb{N}\} \subseteq L_2\,.$$

Wir wählen $k = p + 1$. Damit ist $yx^kz = 0^{p+(k-1)m} = 0^{p+pm} = 0^{p\cdot(m+1)}$. Damit erhalten wir einen Widerspruch, weil $p \cdot (m+1)$ keine Primzahl ist; man beachte, dass $m > 0$. Also ist die Annahme falsch und L_2 ist nicht regulär.

Eine Konsequenz aus dem Primzahlsatz. Wir könnten die Nichtregularität von L_2 auch mit Hilfe der Kolmogorov-Komplexität oder mit Hilfe von Lemma 3.3 zeigen. Dafür benötigen wir allerdings den Primzahlsatz (Satz 2.3 im Buch), den wir in dieser Vorlesung nicht bewiesen haben. Aus dem Primzahlsatz folgt, dass der Abstand zwischen zwei aufeinanderfolgenden Primzahlen beliebig gross werden kann: Wäre dieser Abstand durch eine Konstante $k \in \mathbb{N}$ nach oben beschränkt, dann gäbe es mindestens n/k Primzahlen unter den ersten n natürlichen Zahlen im Widerspruch zum Primzahlsatz, der besagt, dass es ungefähr $n/\ln n$ solche Primzahlen gibt.

Wir verwenden diese Beobachtung nun, um mit Hilfe der Kolmogorov-Komplexität zu zeigen, dass L_2 nicht regulär ist.

Beweis mit Hilfe der Methode der Kolmogorovkomplexität. Angenommen, $L = L_2$ sei regulär. Sei p_m die m-te Primzahl. Dann ist $0^{p_{m+1}-p_m-1}$ das erste Wort in der Sprache

$$L_{0^{(p_m)+1}} = \{y \mid 0^{(p_m)+1}y \in L\}.$$

Nach Satz 3.1 aus dem Buch existiert eine Konstante c, unabhängig von m, so dass

$$K(0^{p_{m+1}-p_m-1}) \leq \lceil \log_2(1+1) \rceil + c = 1 + c.$$

Da es nur endlich viele Programme der konstanten Länge höchstens $1 + c$ gibt, aber nach der obigen Folgerung aus dem Primzahlsatz unendlich viele Wörter der Form $0^{p_{m+1}-p_m-1}$, ist dies ein Widerspruch. Also ist die Annahme falsch und L_2 ist nicht regulär.

Wir können diese Folgerung aus dem Primzahlsatz ebenfalls verwenden, um mit Hilfe von Lemma 3.3 zu zeigen, dass L_2 nicht regulär ist.

Beweis mit Hilfe von Lemma 3.3. Angenommen, L_2 sei regulär. Dann gibt es einen Automaten $A_2 = (Q, \{0\}, \delta, q_0, F)$ mit $L(A_2) = L_2$. Sei $m = |Q|$. Wir wählen $m+1$ verschiedene Primzahlen $p_{l_0}, \ldots, p_{l_m}$ so, dass die Differenzen $p_{l_{j+1}} - p_{l_j}$, also die Abstände zur nächstgrösseren Primzahl, paarweise verschieden und – ohne Beschänkung der Allgemeinheit – monoton aufsteigend sind. Solche Primzahlen existieren nach der oben beschriebenen Folgerung aus dem Primzahlsatz. Dann betrachten wir die Wörter

$$0^{p_{l_0}}, 0^{p_{l_1}}, \ldots, 0^{p_{l_m}} .$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A_2 Zustände hat, gibt es $i, j \in \{0, \ldots, m\}$ mit $i < j$, so dass

$$\hat{\delta}_{A_2}(q_0, 0^{p_{l_i}}) = \hat{\delta}_{A_2}(q_0, 0^{p_{l_j}}).$$

Nach Lemma 3.3 gilt nun für alle $z \in \{0\}^*$, dass

$$0^{p_{l_i}} z \in L_2 \iff 0^{p_{l_j}} z \in L_2.$$

Die Wahl von $z = 0^{p_{l_{i+1}} - p_{l_i}}$ führt aber zum Widerspruch, weil

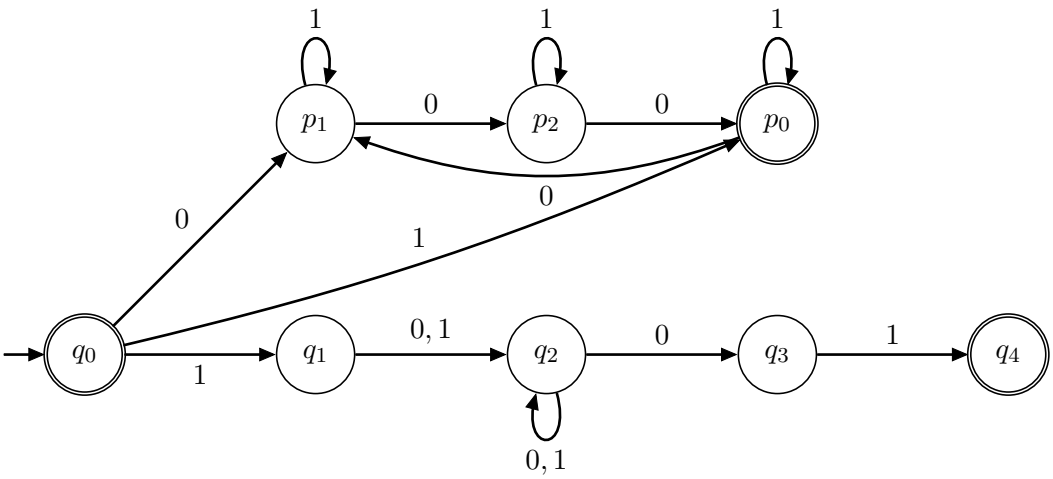
$$0^{p_{l_i}} z = 0^{p_{l_{i+1}}} \in L_2$$

und $0^{p_{l_j}} z = 0^{p_{l_j} + (p_{l_{i+1}} - p_{l_i})} \notin L_2$ gemäss der Voraussetzung über die gewählten Primzahlen gilt. Also ist die Annahme falsch und L_2 ist nicht regulär.

Lösung zu Aufgabe 14

(a) Der folgende nichtdeterministische endliche Automat M akzeptiert die Sprache

$$L = \{x \in \{0,1\}^* \mid |x|_0 \bmod 3 = 0 \text{ oder } x = 1y01 \text{ für } y \in \{0,1\}^+\} .$$



Dieser Automat besteht aus zwei Teilautomaten für die beiden Sprachen

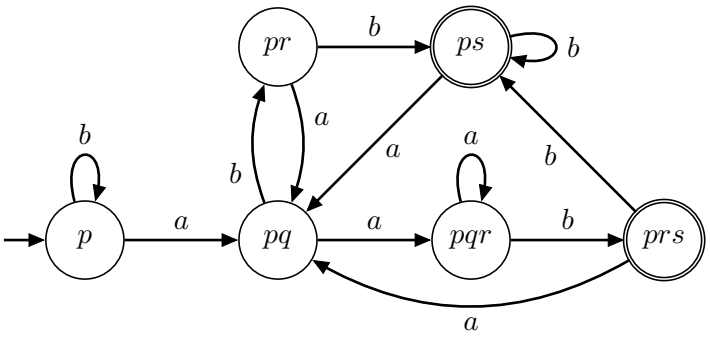
$$L_1 = \{x \in \{0,1\}^* \mid |x|_0 \bmod 3 = 0\}$$

und

$$L_2 = \{x \in \{0,1\}^* \mid x = 1y01 \text{ für } y \in \{0,1\}^+\}$$

mit $L_1 \cup L_2 = L$. Vom Startzustand q_0 aus verzweigt M nichtdeterministisch in einen der beiden Teilautomaten. In den Zuständen p_0, p_1 und p_2 zählt M die Anzahl der Nullen im Eingabewort modulo drei. Falls diese Zählung 0 ergibt, ist die erste der zwei Bedingungen von L erfüllt und M akzeptiert die Eingabe im Zustand p_0. In den Zuständen q_1 bis q_4 sucht M das Muster $y01$ für die zweite Bedingung von L. Dabei entscheidet M im Zustand q_2 nichtdeterministisch, wann das Suffix 01 beginnt. Das Präfix 1 des Gesamtmusters $1y01$ wird bereits beim Übergang von q_0 nach q_1 gelesen. Der Zustand q_0 muss akzeptierend sein, weil das leere Wort λ in $L_1 \subseteq L$ liegt.

(b) Die Anwendung der Potenzmengenkonstruktion auf den auf dem Aufgabenblatt gezeigten nichtderministischen endlichen Automaten ergibt den folgenden deterministischen endlichen Automaten A. Dabei wurden alle nicht erreichbaren Zustände weggelassen. Zur einfacheren graphischen Darstellung wurde die Beschriftung der Zustände verkürzt, pqr steht zum Beispiel für den Zustand $\langle\{p,q,r\}\rangle$.



(bitte wenden)

Lösung zu Aufgabe 15

(a) Da L_1 und L_2 reguläre Sprachen sind, existieren endliche Automaten

$$A_1 = (Q_1, \{a, b\}, \delta_1, q_{0,1}, F_1) \text{ und } A_2 = (Q_2, \{a, b\}, \delta_2, q_{0,2}, F_2)$$

mit $L(A_1) = L_1$ und $L(A_2) = L_2$. Wir geben einen endlichen Automaten A mit $L(A) = L$ an, woraus die Regularität von $L = L_1\{c\}L_2$ folgt. Ohne Beschränkung der Allgemeinheit seien die Mengen Q_1, Q_2 und $\{q_s\}$ paarweise disjunkt, wobei q_s einen zusätzlichen Zustand bezeichnet. Sei $A = (Q_1 \cup Q_2 \cup \{q_s\}, \{a, b, c\}, \delta, q_{0,1}, F_2)$ mit

$$\begin{aligned}
&\delta(q, s) = q' \text{ für alle } s \in \{a, b\} \text{ und } q \in Q_1 \text{ mit } \delta_1(q, s) = q',\\
&\delta(q, s) = q' \text{ für alle } s \in \{a, b\} \text{ und } q \in Q_2 \text{ mit } \delta_2(q, s) = q',\\
&\delta(q, c) = q_{0,2} \text{ für alle } q \in F_1,\\
&\delta(q, c) = q_s \text{ für alle } q \in (Q_1 - F_1) \cup Q_2 \text{ und}\\
&\delta(q_s, s) = q_s \text{ für alle } s \in \{a, b, c\}.
\end{aligned}$$

Der Automat A funktioniert also beim Einlesen von Buchstaben aus $\{a, b\}$ zunächst so wie der Automat A_1, mit dem Unterschied, dass in A die Zustände $F_1 \subseteq Q_1$ nicht mehr akzeptierend sind. Wird in einem der Zustände von F_1 ein c gelesen, geht der Automat A zum Startzustand $q_{0,2}$ von Automat A_2 über und funktioniert dann genau so wie A_2, solange nur Buchstaben aus $\{a, b\}$ gelesen werden. Wird in irgendeinem Zustand ausserhalb von F_1 ein c gelesen, geht der Automat A in einen nicht akzeptierenden Senkenzustand q_s über und bleibt dort, egal was sont noch gelesen wird. Somit werden genau die Wörter in $L_1\{c\}L_2 = \{w_1 c w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ akzeptiert.

(b) Da L regulär ist, existiert ein endlicher Automat $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Wir beschreiben einen endlichen Automaten A^R mit $L(A^R) = L^R$, was die Aussage beweist.

Wir nehmen zunächst an, dass A genau einen akzeptierenden Zustand hat, also $F = \{f\}$ für ein $f \in Q$. In diesem Fall reicht es aus, einfach alle Transitionen umzudrehen. Formal entsteht dabei im Allgemeinen ein nichtdeterministischer endlicher Automat $A^R = (Q, \Sigma, \delta^R, f, \{q_0\})$ mit

$$\delta^R(q, s) = \{q' \in Q \mid \delta(q', s) = q\} \text{ für alle } s \in \Sigma \text{ und } q \in Q.$$

Dieser Automat akzeptiert offenbar L^R und wir können ihn mit der Potenzmengenkonstruktion (Satz 3.2) auch in einen äquivalenten deterministischen endlichen Automaten umwandeln.

Wenn der Automat A nicht genau einen akzeptierenden Zustand hat, dann definieren wir $A_f = (\Sigma, Q, q_0, \delta, \{f\})$ für jeden Zustand $f \in F$ und beobachten, dass $L(A) = \bigcup_{f \in F} L(A_f)$. Somit können wir die obige Konstruktion $|F|$ Mal anwenden, um für jedes $f \in F$ einen deterministischen endlichen Automaten A_f^R mit $L(A_f^R) = L(A_f)^R$ zu erhalten, und die gewonnenen Automaten dann zu einem grossen Produktautomaten A^R kombinieren, der die Sprache $L(A^R) = \bigcup_{f \in F} L(A_f^R) = \left(\bigcup_{f \in F} L(A)\right)^R$ erkennt.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm
https://courses.ite.inf.ethz.ch/theo_inf_22

Lösungsvorschläge – Blatt 2

Zürich, 7. Oktober 2022

Lösung zu Aufgabe 4

(a) Wir geben zunächst für jedes $n \in \mathbb{N} - \{0\}$ ein Pascal-Programm an, das w_n erzeugt:

```
begin
  s:= n;
  s:= 2*s*s*s;
  j:= 1;
  for i:=1 to s do
    j:=j*3;
  for i:=1 to j do
    write(010);
end.
```

Dieses Programm berechnet zuerst $s = 2n^3$, womit dann in einer Schleife $j = 3^{2n^3}$ berechnet wird. In einer weiteren Schleife gibt das Programm dann 3^{2n^3} Mal die Zeichenfolge 010 aus, was genau das Wort w_n ergibt.

Wenn wir die Operation ^ für die Exponentiation zulassen (dies ist nicht ursprüngliche Pascal-Syntax, wir wollen diese Notation aber im Folgenden verwenden, um die Programme lesbarer darzustellen), erzeugt auch das folgende Programm das Wort w_n:

```
begin
  s:= n;
  s:= 3^(2*s*s*s);
  for i:=1 to s do
    write(010);
end.
```

Der einzige Teil des Maschinencodes dieser beiden Programme, der von w_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Programmcode hat eine konstante Länge. Also ist die binäre Länge dieser Programme höchstens $\lceil \log_2(n+1) \rceil + c$ für eine Konstante c.

Damit lässt sich die Kolmogorov-Komplexität von w_n von oben abschätzen durch

$$K(w_n) \leq \lceil \log_2(n+1) \rceil + c$$

für eine Konstante c.

Die Länge von w_n ist $|w_n| = |010| \cdot 3^{2n^3} = 3^{2n^3+1}$. Also gilt $\log_3 |w_n| = 2n^3 + 1$ und somit $n = \sqrt[3]{(\log_3 |w_n| - 1)/2}$ und wir erhalten eine obere Schranke von

$$K(w_n) \leq \left\lceil \log_2 \left(1 + \sqrt[3]{(\log_3 |w_n| - 1)/2}\right)\right\rceil + c \leq \lceil \log_2 \log_3 |w_n| \rceil + c'$$

für die Kolmogorov-Komplexität von w_n für eine Konstante c'.

(b) Wir definieren die Folge von Wörtern $y_1, y_2, y_3, \ldots$ durch $y_i = 0^{2^{3^{2^i}}}$ für alle $i \in \mathbb{N}$. Offensichtlich gilt $y_1 < y_2 < y_3 < \ldots$ und wir haben $i = \log_2 \log_3 \log_2 |y_i|$ für alle $i \geq 1$.

Nun konstruieren wir für jedes y_i ein Pascal-Programm C_i, das y_i erzeugt:

```
begin
  k:= 2^i;
  k:= 2^(3^k);
  for j:=1 to k do
    write(0);
end.
```

In diesem Programm wird zuerst in zwei Schritten der Wert $k = 2^{3^{2^i}}$ berechnet und dann werden in der `for`-Schleife k Nullen geschrieben. Der Teil des Maschinencodes von C_i, der von y_i abhängt, ist nur die Darstellung von i. Der Maschinencode für die Darstellung des restlichen Programms hat also nur eine konstante Länge, während die binäre Kodierung von i die Länge $\lceil \log_2(i+1) \rceil \leq 2 + \log_2 i$ hat. Damit folgt

$$\begin{aligned} K(y_i) &\leq \log_2 i + c \\ &= \log_2 \log_2 \log_3 \log_2(|y_i|) + c \end{aligned}$$

für eine Konstante c.

Lösung zu Aufgabe 5

Wir wählen $f(n) = 2^{2n+3}$. Es gibt 2^{2n+3} natürliche Zahlen, die kleiner als $f(n)$ sind. Es ist zu zeigen, dass maximal ein Achtel dieser Zahlen – also maximal 2^{2n} davon – eine Kolmogorov-Komplexität kleiner als $2n$ haben.

Die Anzahl der unterschiedlichen Programme der Länge kleiner als $2n$ ist höchstens

$$\sum_{j=1}^{2n-1} 2^j = 2^{2n} - 2\,,$$

da es für jede Länge nicht mehr Maschinencodes als nichtleere Binärstrings geben kann. Also gibt es maximal $2^{2n} - 2$ Wörter w mit $K(w) < 2n$. Da alle natürlichen Zahlen paarweise verschiedene Binärdarstellungen haben, gibt es also höchstens $2^{2n} - 2$ Zahlen mit einer Kolmogorov-Komplexität kleiner als $2n$.

Lösung zu Aufgabe 6

Wir beweisen die Aussage indirekt. Somit gebe es also eine unendlich grosse Anzahl von Primzahlen, die als zufällig angesehen werden können. Nach Definition bedeutet dies, dass es unendlich viele $k \in \mathbb{N} - \{0\}$ gibt, sodass

$$K(p_k) \geq \lceil \log_2(p_k + 1) \rceil - 1 \geq \log_2(p_k) - 1 \qquad (1)$$

gilt, wobei p_k die k-te Primzahl ist.

Ferner lässt sich die m-te Primzahl p_m für jedes $m \in \mathbb{N} - \{0\}$ mit einem Pascal-Programm C_m berechnen, das die binäre Kodierung von m enthält, alle natürlichen Zahlen in aufsteigender Reihenfolge darauf testet, ob sie prim sind (Sieb des Eratosthenes) und die m-te gefundene Primzahl ausgibt. Alle Bestandteile dieses Programms mit Ausnahme der Kodierung von m haben eine konstante Länge, also hat C_m die Länge $\lceil \log_2(m+1) \rceil + c \leq \log_2 m + c'$ für Konstanten c und $c' = c + 1$.

Sei $n \in \mathbb{N}$ und sei $\mathrm{Prim}(n)$ die Anzahl der Primzahlen kleiner gleich n. Nach dem Primzahlsatz gilt für alle $n > 67$, dass

$$\mathrm{Prim}(n) < \frac{n}{\ln n - \frac{3}{2}} .$$

Da nach Definition wiederum $m = \mathrm{Prim}(p_m)$ ist, erhalten wir

$$\mathrm{Prim}(p_m) = m < \frac{p_m}{\ln(p_m) - \frac{3}{2}}$$

und schliesslich

$$K(p_m) \leq \log_2\left(\frac{p_m}{\ln(p_m) - \frac{3}{2}}\right) + c' = \log_2(p_m) - \log_2\left(\ln(p_m) - \frac{3}{2}\right) + c' . \tag{2}$$

Aus den beiden Schranken (1) und (2) für die Kolmogorov-Komplexität ergibt sich, dass

$$\log_2(p_l) - 1 \leq \log_2(p_l) - \log_2\left(\ln(p_l) - \frac{3}{2}\right) + c'$$

für unendlich viele $l \in \mathbb{N}$ gelten muss. Hieraus folgt wiederum, dass für diese l auch

$$\log_2\left(\ln(p_l) - \frac{3}{2}\right) \leq c' + 1$$

sein muss, was unmöglich ist, da c' eine Konstante ist und $\log_2(\ln(p_l) - 3/2)$ mit p_l wächst. Somit ist unsere Annahme falsch und die zu beweisende Aussage folgt.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm
https://courses.ite.inf.ethz.ch/theo_inf_22/

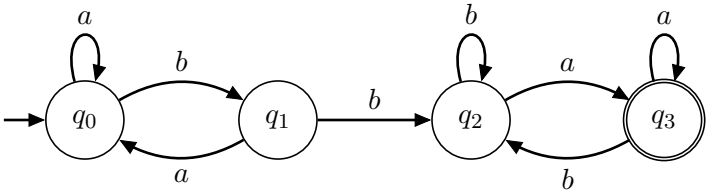
Lösungsvorschläge – Blatt 3

Zürich, 14. Oktober 2022

Lösung zu Aufgabe 7

(a) Der folgende endliche Automat akzeptiert die Sprache

$$L_1 = \{xbbya \in \{a,b\}^* \mid x, y \in \{a,b\}^*\}.$$



Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned}
&\mathrm{Kl}[q_0] = \{wa \in \{a,b\}^* \mid w \text{ enthält nicht } bb\} \cup \{\lambda\}\,,\\
&\mathrm{Kl}[q_1] = \mathrm{Kl}[q_0] \cdot \{b\}\,,\\
&\mathrm{Kl}[q_3] = L_1\,,\\
&\mathrm{Kl}[q_2] = \{a,b\}^* - (\mathrm{Kl}[q_0] \cup \mathrm{Kl}[q_1] \cup \mathrm{Kl}[q_3])\,.
\end{aligned}$$

Man beachte, dass es hilfreich sein kann, die Reihenfolge der Klassenbeschreibungen geschickt zu wählen.

(b) Wir betrachten die Sprache

$$L_2 = \{w \in \{a,b\}^* \mid (|w|_a)^{|w|_b} \equiv 0 \bmod 2\}.$$

Sei $k = |w|_a$ und $l = |w|_b$. Wir betrachten zwei Fälle und vereinfachen so die Beschreibung der Sprache:

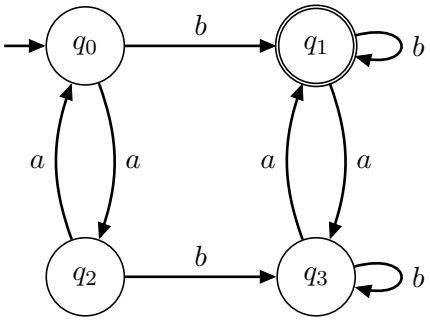
1. Falls $l = 0$, dann $k^l = k^0 = 1$.
2. Falls $l \geq 1$, dann $k^l \equiv k \bmod 2$.
   (a) Falls $l \geq 1$ und $k = 2n$, dann $k^l = (2n)^l = 2^l n^l$, also $k^l \equiv 0 \bmod 2$.

(b) Falls $l \geq 1$ und $k = 2n+1$, dann $k^l = (2n+1)^l = \sum_{i=0}^{l} \binom{l}{i}(2n)^i = 1 + 2\sum_{i=1}^{l} \binom{l}{i} 2^{i-1} n^i$, also $k^l \equiv 1 \bmod 2$.

Wir können die Sprache L_2 umformulieren zu

$$L_2 = \{w \in \{a,b\}^* \mid |w|_b \geq 1 \text{ und } |w|_a \equiv 0 \bmod 2\}\,.$$

Der folgende Automat akzeptiert L_2.



Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned}
&\mathrm{Kl}[q_0] = \{a^{2k} \in \{a,b\}^* \mid k \in \mathbb{N}_0\}\,,\\
&\mathrm{Kl}[q_2] = \{a^{2k+1} \in \{a,b\}^* \mid k \in \mathbb{N}_0\} = \mathrm{Kl}[q_0] \cdot \{a\}\,,\\
&\mathrm{Kl}[q_1] = L_1\,,\\
&\mathrm{Kl}[q_3] = \{a,b\}^* - (\mathrm{Kl}[q_0] \cup \mathrm{Kl}[q_1] \cup \mathrm{Kl}[q_2])\,.
\end{aligned}$$

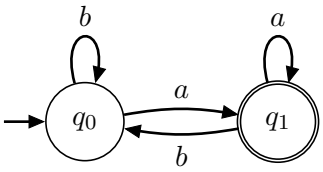
Die beiden Zustände q_0 und q_2 akzeptieren nicht, weil $l = 0$. In q_1 ist die Anzahl der a gerade und mindestens ein b enthalten, also $k^l \equiv k \equiv 0 \bmod 2$. In q_1 ist die Anzahl der a ungerade und mindestens ein b enthalten, also $k^l \equiv k \equiv 1 \bmod 2$.

Lösung zu Aufgabe 8

Die in der Aufgabenstellung gegebene Sprache L lässt sich schreiben als $L = L_1 \cup L_2$ mit

$$\begin{aligned}
&L_1 = \{w \in \{a,b\}^* \mid w = ya\}\,,\\
&L_2 = \{w \in \{a,b\}^* \mid |w| = 2\}\,.
\end{aligned}$$

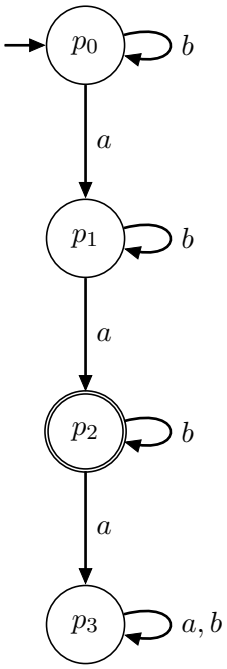
Für die Sprache L_1 lässt sich der folgende Automat A_1 konstruieren:



Es gilt

$$\begin{aligned}
&\mathrm{Kl}[q_1] = L_1\,,\\
&\mathrm{Kl}[q_0] = \{a,b\}^* - L_1\,.
\end{aligned}$$

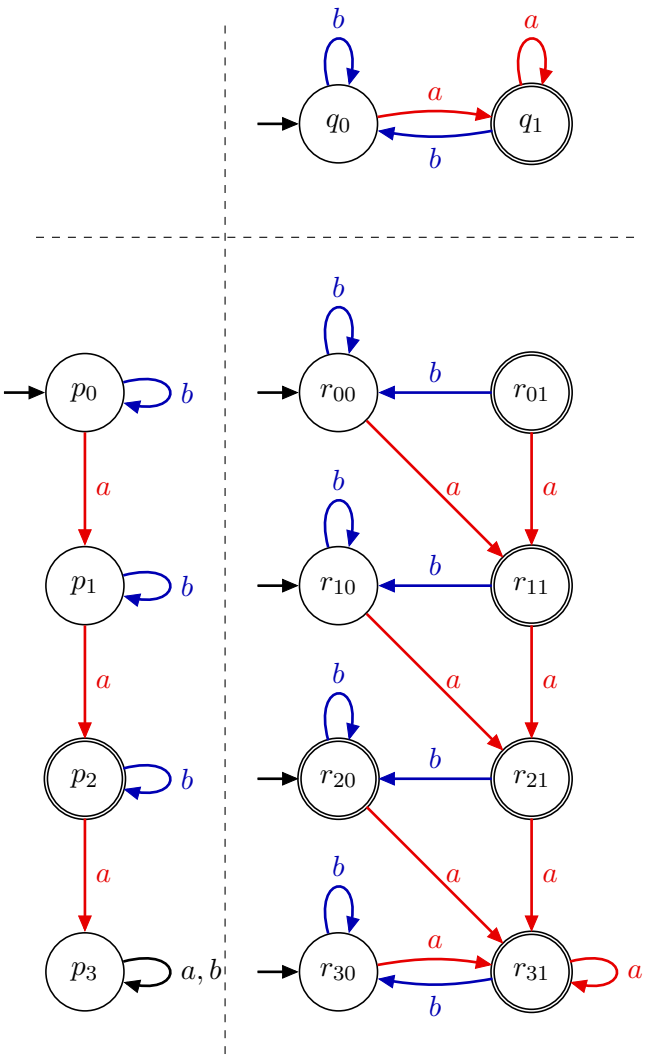
Für die Sprache L_2 lässt sich der folgende Automat A_2 konstruieren:



Es gilt

$$\begin{aligned}
&\mathrm{Kl}[p_0] = \{b\}^* \,, \\
&\mathrm{Kl}[p_1] = \mathrm{Kl}[p_0] \cdot \{a\} \cdot \{b\}^* \,, \\
&\mathrm{Kl}[p_2] = L_2 \,, \\
&\mathrm{Kl}[p_3] = \{a, b\}^* - (\mathrm{Kl}[p_0] \cup \mathrm{Kl}[p_1] \cup \mathrm{Kl}[p_2]) \,.
\end{aligned}$$

Mit der Methode des modularen Entwurfs kann man nun aus A_1 und A_2 den folgenden Produktautomaten A konstruieren, der die Sprache $L = L_1 \cup L_2$ akzeptiert. Zur einfacheren Darstellung verwenden wir die Notation $r_{x,y} = \langle p_x, q_y \rangle$. Weil dies ein Produktautomat für die Vereinigung von zwei Sprachen ist, sind alle Zustände akzeptierend, die einen akzeptierenden Zustand aus einem der beiden Teilautomaten enthalten.



Der Zustand r_{01} ist vom Startzustand aus nicht erreichbar, kann also auch weggelassen werden.

Lösung zu Aufgabe 9

(a) Wir betrachten die Sprache

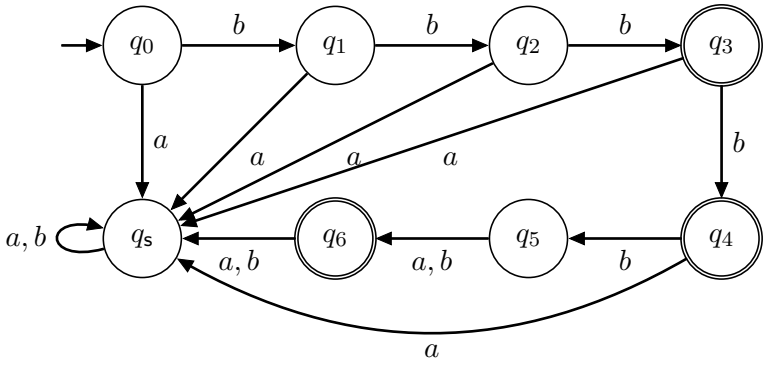
$$L_1 = \{xa^k \in \{a,b\}^* \mid x \in \{b\}^* \wedge k \coloneqq (|x|^2 \bmod (|x|-2)) \wedge k \leq 3 \leq |x| \leq 10\}.$$

Durch einfaches Ausrechnen erhalten wir für $3 \leq |x| \leq 10$, dass

$$k = \begin{cases} 1 & \text{falls } |x| = 5 \\ 4 & \text{falls } 7 \leq |x| \leq 10 \\ 0 & \text{sonst} \end{cases}.$$

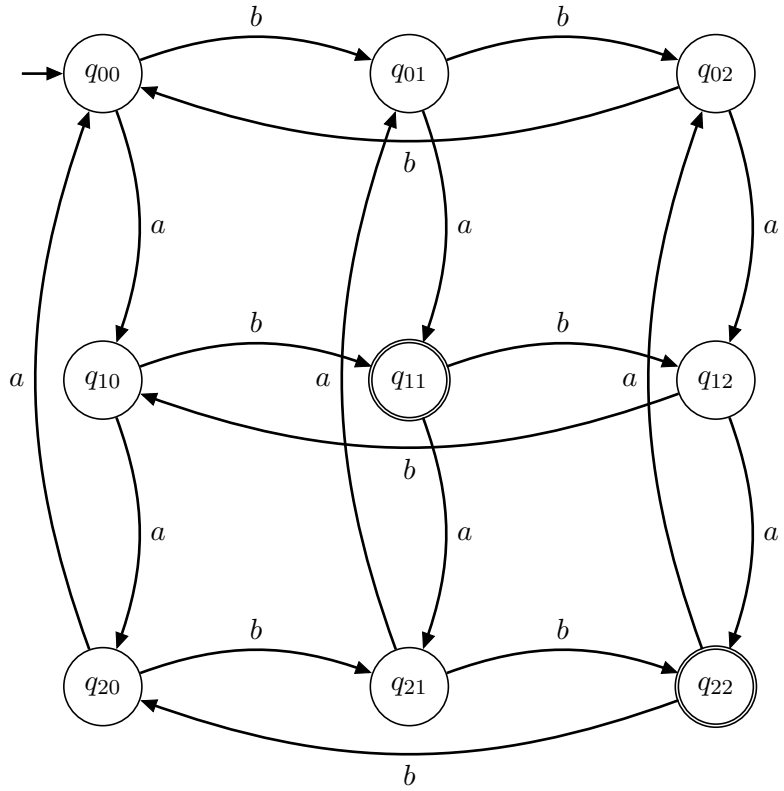
(Man beachte hierfür, dass $l \bmod 1 = 0$ für alle l gilt.) Also müssen wir gemäss der Definition von L_1 nur die Werte $3 \leq |x| \leq 6$ betrachten und es ergibt sich $L_1 = \{b^3, b^4, b^5a, b^6\}$.

Der folgende endliche Automat akzeptiert L_1.



(b) Der folgende endliche Automat akzeptiert die Sprache

$$L_2 = \{w \in \{a, b\}^* \mid |w|_a \cdot |w|_b \equiv 1 \bmod 3\}$$



Der Zustand r_{kl} zählt die Anzahl der bisherigen as und bs, d.h., $k = |w|_a \bmod 3$ und $l = |w|_b \bmod 3$.

Der Automat akzeptiert in jedem Zustand, in dem $k \cdot l \equiv 1 \bmod 3$ gilt, also in q_{11} und q_{22}.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm
https://courses.ite.inf.ethz.ch/theo_inf_22/

Lösungsvorschläge – Blatt 4

Zürich, 14. Oktober 2022

Lösung zu Aufgabe 10

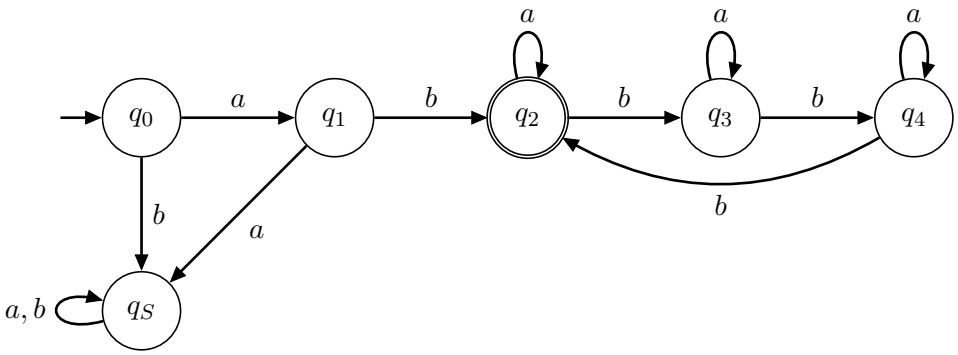
Um zu zeigen, dass jeder deterministische endliche Automat A, der die Sprache

$$L = \{w \in \{a,b\}^* \mid |w|_b = 1 \bmod 3 \text{ und } w \text{ beginnt mit } ab\}$$

akzeptiert, mindestens 6 Zustände benötigt, bestimmen wir 6 Wörter $w_1, \ldots, w_6$ und zeigen, dass diese den Automaten in 6 paarweise verschiedene Zustände führen müssen. Falls es einen EA gibt, der mit dem Lesen von zwei verschiedenen Wörtern w_i und w_j in demselben Zustand endet, dann gilt nach Lemma 3.3 aus dem Buch für jedes $z \in \{a,b\}^*$, dass dann auch $w_i z$ und $w_j z$ den Automaten in denselben Zustand führen. Wir wollen zeigen, dass dies für einen Automaten mit weniger als 6 Zuständen nicht möglich ist, indem wir für je zwei verschiedene Wörter w_i und w_j ein Wort $z_{i,j}$ angeben, so dass

$$w_i z_{i,j} \in L(A) \iff w_j z_{i,j} \notin L(A)\,. \tag{1}$$

Eine Strategie, um geeignete Wörter $w_1, \ldots, w_6$ zu bestimmen, besteht darin, zunächst einen EA A mit 6 Zuständen für die Sprache L zu entwerfen, und dann für jeden Zustand q das kürzeste Wort aus der Klasse Kl$[q]$ zu wählen. Der folgende Automat A mit 6 Zuständen akzeptiert die Sprache L.



Die kürzesten Wörter in den Zustandsklassen von A sind $w_1 = \lambda$, $w_2 = a$, $w_3 = b$, $w_4 = ab$, $w_5 = abb$ und $w_6 = abbb$.

Die folgende Tabelle gibt für alle Paare (w_i, w_j) mit $i < j$ jeweils ein Wort $z_{i,j}$ an.

$z_{i,j}$	$w_2 = a$	$w_3 = b$	$w_4 = ab$	$w_5 = abb$	$w_6 = abbb$
$w_1 = \lambda$	ab	ab	ab	ab	b
$w_2 = a$	$-$	b	λ	bb	ab
$w_3 = b$	$-$	$-$	λ	bb	b
$w_4 = ab$	$-$	$-$	$-$	λ	λ
$w_5 = abb$	$-$	$-$	$-$	$-$	b

Es ist einfach zu sehen, dass diese Wörter die Bedingung (1) erfüllen, zum Beispiel ist $w_2 z_{2,6} = aab \notin L(A)$, aber $w_6 z_{2,6} = abbbab \in L(A)$.

Lösung zu Aufgabe 11

(a) Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache

$$L_1 = \{a^i b^j \mid i, j \in \mathbb{N} \text{ und es gibt ein } k \in \mathbb{N} \text{ so dass } j = k \cdot i\}$$

nicht regulär ist. Angenommen, L_1 sei regulär. Dann gibt es einen Automaten $A_1 = (Q, \{a, b\}, \delta, q_0, F)$ mit $L(A_1) = L_1$. Sei $m = |Q|$. Wir betrachten die Wörter

$$a^l b \quad \text{für } l \in \{1, \ldots, m+1\}\,.$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A_1 Zustände hat, gibt es $i, j \in \{1, \ldots, m+1\}$ mit $i < j$, so dass

$$\hat{\delta}(q_0, a^i b) = \hat{\delta}(q_0, a^j b)\,.$$

Nach Lemma 3.3 gilt nun für alle $z \in \{a, b\}^*$, dass

$$a^i bz \in L_1 \iff a^j bz \in L_1.$$

Die Wahl von $z = b^{i-1}$ führt aber zum Widerspruch, weil

$$a^i bz = a^i b b^{i-1} = a^i b^i \in L_1\,,$$
$$a^j bz = a^j b b^{i-1} = a^j b^i \notin L_1$$

gilt. Dabei folgt letztere Aussage daraus, dass $j > i > 0$, also i kein Vielfaches von j sein kann. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache

$$L_2 = \{wabw^{\mathsf{R}} \mid w \in \{a, b\}^*\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{a, b\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

1. $|yx| \leq n_0$,
2. $|x| \geq 1$ und
3. entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = a^{n_0}aba^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = a^l$ und $x = a^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$, also insbesondere $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^kz \mid k \in \mathbb{N}\} = \{a^{n_0+(k-1)m}aba^{n_0} \mid k \in \mathbb{N}\} \subseteq L_2 \,.$$

Dies ist aber ein Widerspruch, da $yx^2z = a^{n_0+m}aba^{n_0} \notin L_2$. Also ist die Annahme falsch und L_2 ist nicht regulär.

Lösung zu Aufgabe 12

(a) Sei $L_1 = \{0^{n^2 \cdot 2^n} \mid n \in \mathbb{N}\}$. Wir beweisen mit Hilfe der Kolmogorov-Komplexität, dass L_1 nicht regulär ist.

Angenommen, $L = L_1$ sei regulär. Für jedes $m \in \mathbb{N}$ ist $0^{m^2 \cdot 2^m + (2m+1) \cdot 2^{m+1} - 1}$ das erste Wort in der Sprache

$$L_{0^{m^2 \cdot 2^m + 1}} = \{y \mid 0^{m^2 \cdot 2^m + 1}y \in L\} \,,$$

weil $(m+1)^2 \cdot 2^{m+1} = m^2 \cdot 2^m + m^2 \cdot 2^m + (2m+1) \cdot 2^{m+1}$ gilt. Nach Satz 3.1 aus dem Buch existiert eine Konstante c, unabhängig von m, so dass

$$K(0^{m^2 \cdot 2^m + (2m+1) \cdot 2^{m+1} - 1}) \leq \lceil \log_2(1+1) \rceil + c = 1 + c \,.$$

Da es nur endlich viele Programme der konstanten Länge $1 + c$ gibt, aber unendlich viele Wörter der Form $0^{m^2 \cdot 2^m + (2m+1) \cdot 2^{m+1} - 1}$, ist dies ein Widerspruch. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Wir zeigen im Folgenden, dass man Lemma 3.3 sowie das Pumping-Lemma verwenden kann, um zu zeigen, dass L_2 nicht regulär ist. Die Methode der Kolmogorov-Komplexität ist in diesem Fall nicht direkt anwendbar.

(i) Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid |u|_0 \leq |u|_1 \text{ für alle Präfixe } u \text{ von } w\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann gibt es einen Automaten $A_2 = (Q, \{0,1\}, \delta, q_0, F)$ mit $L(A_2) = L_2$. Sei $m = |Q|$. Wir betrachten die Wörter

$$1, 1^2, \ldots, 1^{m+1} \,.$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A_2 Zustände hat, gibt es $i, j \in \{1, \ldots, m+1\}$ mit $i < j$, so dass

$$\hat{\delta}(q_0, 1^i) = \hat{\delta}(q_0, 1^j) \,.$$

Nach Lemma 3.3 gilt nun für alle $z \in \{0,1\}^*$, dass

$$1^iz \in L_2 \iff 1^jz \in L_2.$$

Die Wahl von $z = 0^j$ führt aber zum Widerspruch, weil

$$1^i z = 1^i 0^j \notin L_2$$

und

$$1^j z = 1^j 0^j \in L_2$$

gilt. Letzteres folgt, weil $i < j$ und weil jedes Wort auch ein Präfix von sich selbst ist. Also ist die Annahme falsch und L_2 ist nicht regulär.

(ii) Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid |u|_0 \leq |u|_1 \text{ für alle Präfixe } u \text{ von } w\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0,1\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

1. $|yx| \leq n_0$,
2. $|x| \geq 1$ und
3. entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = 1^{n_0} 0^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = 1^l$ und $x = 1^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$, also insbesondere $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^k z \mid k \in \mathbb{N}\} = \{1^{n_0+(k-1)m} 0^{n_0} \mid k \in \mathbb{N}\} \subseteq L_2 \,.$$

Dies ist aber ein Widerspruch, da $yx^0 z = 1^{n_0-m} 0^{n_0} \notin L_2$. Letzteres folgt, weil $n_0 - m < n_0$ und weil jedes Wort auch ein Präfix von sich selbst ist. Also ist die Annahme falsch und L_2 ist nicht regulär.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm
https://courses.ite.inf.ethz.ch/theo_inf_22/

Lösungsvorschläge – Blatt 5

Zürich, 21. Oktober 2022

Lösung zu Aufgabe 13

Sei $L = \{w_i \mid i \in \mathbb{N}\} \subseteq \{0\}^*$ eine unendliche Sprache mit

$$|w_{i+1}| - |w_i| \geq \log_2 \log_2 i$$

für alle $i \in \mathbb{N}$. Wir verwenden das Pumping-Lemma, um zu zeigen, dass L nicht regulär ist. Angenommen, L sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0,1\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

(i) $|yx| \leq n_0$,

(ii) $|x| \geq 1$ und

(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L = \emptyset$.

Sei $w_i \in L$ mit $i > 2^{2^{n_0}}$ beliebig gewählt. Dann muss es eine Zerlegung $w_i = yxz$ von w_i geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Nach (i) und (ii) gilt $1 \leq |x| \leq n_0$. Da $w_i \in L$, gilt nach (iii), dass auch

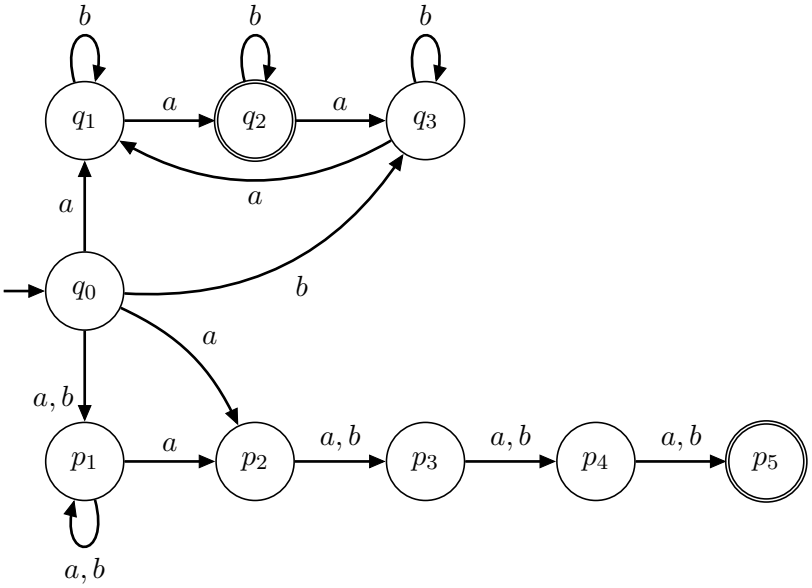
$$\{yx^kz \mid k \in \mathbb{N}\} \subseteq L\,.$$

Dies ist aber ein Widerspruch, da $|yx^2z| > |w_i|$ gilt, aber $|yx^2z| \leq |w_i| + n_0 < |w_i| + \log_2 \log_2 i \leq |w_{i+1}|$ und somit $yx^2z \notin L$. Also ist die Annahme falsch und L ist nicht regulär.

Lösung zu Aufgabe 14

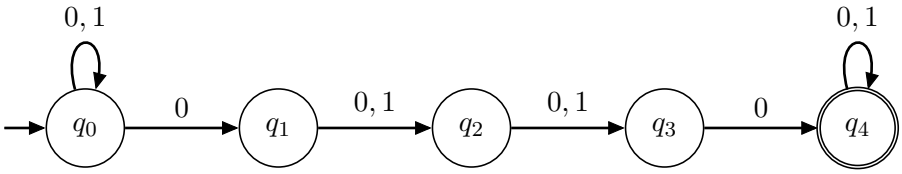
(a) Der folgende nichtdeterministische endliche Automat M_1 akzeptiert die Sprache

$$L_1 = \{x \in \{a,b\}^* \mid |x|_a \bmod 3 = 2 \text{ oder } (x = yaz \text{ mit } y,z \in \{a,b\}^* \text{ und } |z| = 3)\}\,.$$



Der NEA besteht aus zwei Teilautomaten für die zwei Bedingungen der Sprache L_1. Vom Startzustand q_0 aus verzweigt M_1 nichtdeterministisch in einen der beiden Teilautomaten. In den Zuständen q_1, q_2 und q_3 zählt M_1 die Anzahl der a's im Eingabewort modulo drei. Falls diese Zählung 2 ergibt, ist die erste der beiden Bedingungen erfüllt und M_1 akzeptiert die Eingabe im Zustand q_2. In den Zuständen p_1 bis p_5 überprüft M_1, ob das viertletzte Zeichen ein a ist, ob also die zweite Bedingung von L_1 gilt. Dabei entscheidet M_1 im Zustand p_1 nichtdeterministisch, wann das viertletzte Zeichen erreicht ist. Mit Hilfe der Zustände p_2 bis p_5 liest M_1 auf dem Weg von p_2 zum Zustand p_5 genau drei beliebige Symbole. Falls dann die komplette Eingabe gelesen wurde, akzeptiert M_1 im Zustand p_5.

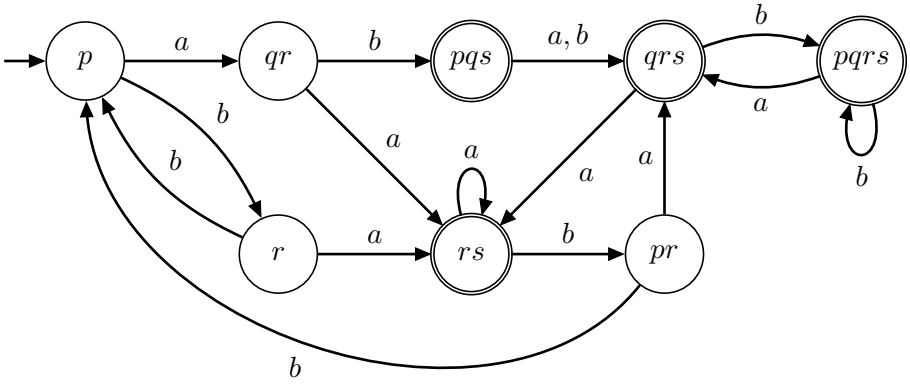
(b) Der folgende NEA akzeptiert die Sprache L_2.



Dieser NEA entscheidet nichtdeterministisch, wann das gesuchte Muster in der Eingabe beginnt und wechselt dann vom Zustand q_0 in den Zustand q_1. Wenn diese nichtdeterministische Entscheidung richtig war, dann kann der Automat nun das Muster lesen und endet im akzeptierenden Zustand q_4. Wenn die Eingabe das gesuchte Muster nicht enthält, dann hat der NEA keine Möglichkeit, sie zu akzeptieren.

Lösung zu Aufgabe 15

Die Anwendung der Potenzmengenkonstruktion auf den NEA vom Aufgabenblatt ergibt den folgenden deterministischen endlichen Automaten A. Dabei wurden alle nicht erreichbaren Zustände weggelassen. Zur einfacheren graphischen Darstellung wurde die Beschriftung der Zustände verkürzt, pqs steht zum Beispiel für den Zustand $\langle\{p, q, s\}\rangle$.



ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm

Lösungsvorschläge – Blatt 2

Zürich, 6. Oktober 2023

Lösung zu Aufgabe 4

(a) Wir definieren die Folge $(x_n)_{n=1}^{\infty}$ mit $x_n = 0^{2^n}$ für jedes $n \in \mathbb{N} - \{0\}$. Es ist offenbar, dass alle Wörter x_n paarweise unterschiedlich sind.

Wir geben jetzt für jedes $n \in \mathbb{N} - \{0\}$ ein Programm an, das x_n erzeugt:

```
begin
  s := n;
  j := 1;
  for i := 1 to s do
    j := j * 2;
  for i := 1 to j do
    write(0);
end.
```

Dieses Programm berechnet zuerst in einer Schleife $j = 2^n$. In einer weiteren Schleife gibt das Programm dann 2^n Mal das Zeichen 0 aus, was genau das Wort x_n ergibt.

Der einzige Teil des Quellcodes, der von x_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Quellcode hat eine konstante Länge. Also ist die binäre Länge des Quellcodes höchstens $\lceil \log_2(n+1) \rceil + c$ für eine Konstante c.

Damit lässt sich die Kolmogorov-Komplexität von x_n von oben abschätzen durch

$$K(x_n) \leq \lceil \log_2(n+1) \rceil + c \leq \log_2 n + (c+1)$$

für eine Konstante c.

Die Länge von x_n ist $|x_n| = |0^{2^n}| = 2^n$. Also gilt $\log_2 \log_2 |x_n| = \log_2 n$ und somit erhalten wir eine obere Schranke von

$$K(x_n) \leq \log_2 \log_2 |x_n| + (c+1)$$

für die Kolmogorov-Komplexität von x_n.

(b) Wir zeigen indirekt, dass es keine solche Folge $(x_n)_{n=1}^{\infty}$ paarweise unterschiedlicher Wörter gibt. Nehmen wir an, $(x_n)_{n=1}^{\infty}$ sei eine solche Folge. Dann gibt es eine Konstante $c \in \mathbb{N}$, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq \log_2 \sqrt{n} + c$$

gilt.

Für ein fixes $n \in \mathbb{N} - \{0\}$ und $t := \log_2 \sqrt{n} + c$ gibt es höchstens

$$\sum_{i=0}^{t} 2^i = 2^{t+1} - 1 = 2^{\log_2 \sqrt{n}+c+1} - 1 = \sqrt{n} \cdot 2^{c+1} - 1$$

Quellcodes der binären Länge höchstens t, also können höchstens $\sqrt{n} \cdot 2^{c+1} - 1$ paarweise unterschiedliche Wörter durch Programme mit Quellcodes der binären Länge höchstens t erzeugt werden. Für ein genügend grosses n, das nur von der Konstante c abhängt, gilt $\sqrt{n} \cdot 2^{c+1} - 1 < n$, was im Widerspruch dazu steht, dass die n Wörter $x_1, \ldots, x_n$ paarweise unterschiedlich sind und $K(x_i) \leq t$ für alle $i \in \{1, \ldots, n\}$ gilt, d.h., $x_1, \ldots, x_n$ durch Programme mit Quellcodes der binären Länge höchstens t erzeugt werden.

Lösung zu Aufgabe 5

Das Intervall $[2^n, 2^{n+1} - 1]$ enthält 2^n natürliche Zahlen der binären Länge $n + 1$. Weil die binären Darstellungen dieser Zahlen alle mit einer führenden 1 beginnen, kann man sie identifizieren mit den Wörtern der Länge n über dem binären Alphabet Σ_{bool}. Es bleibt also zu zeigen, dass es unter den 2^n Wörtern der Länge n mindestens $2^n - 2^{n-i}$ Wörter w gibt mit $K(w) \geq n - i$.

Die Anzahl der unterschiedlichen Programme der Länge kleiner als $n - i$ ist höchstens

$$\sum_{j=0}^{n-i-1} 2^j = 2^{n-i} - 1\,,$$

da es für jede Länge nicht mehr Maschinencodes als Binärstrings geben kann. Also gibt es maximal $2^{n-i} - 1$ Wörter w mit $K(w) < n - i$. Für die Anzahl m der Wörter w in $(\Sigma_{\text{bool}})^n$ mit $K(w) \geq n - i$ gilt also

$$m \geq 2^n - (2^{n-i} - 1) > 2^n - 2^{n-i}\,.$$

Lösung zu Aufgabe 6

(a) Ein Palindrom mit gerader Länge $k = 2m$ hat die Form $x = yy^{\mathrm{R}}$, wobei y ein beliebiges Wort der Länge m ist. Ein Palindrom mit ungerader Länge $k = 2m + 1$ hat die Form $x = yzy^{\mathrm{R}}$, wobei y ein beliebiges Wort der Länge m und wobei z ein beliebiges Symbol ist. Es folgt:

- Für gerades $k = 2m$ gibt es $2^m = 2^{\frac{k}{2}}$ Palindrome der Länge k.
- Für ungerades $k = 2m + 1$ gibt es $2^m \cdot 2 = 2^{\frac{k+1}{2}}$ Palindrome der Länge k.

Sei k gerade. Mit Hilfe der geometrischen Summenformel lässt sich wie folgt die Anzahl aller Palindrome mit gerader Länge bis und mit Länge k bestimmen (der letzte Summand 2^0 entspricht dem leeren Wort, das auch ein Palindrom ist):

$$2^{\frac{k}{2}} + 2^{\frac{k}{2}-1} + 2^{\frac{k}{2}-2} + \cdots + 2^1 + 2^0 = 2^{\frac{k}{2}+1} - 1$$

Sei k ungerade. Auf analoge Weise lässt sich auch die Anzahl aller Palindrome mit ungerader Länge bis und mit Länge k bestimmen (der letzte Summand 2^1 hier entspricht den zwei möglichen Wörtern der Länge 1, welche beide Palindrome sind):

$$2^{\frac{k+1}{2}} + 2^{\frac{k+1}{2}-1} + 2^{\frac{k+1}{2}-2} + \cdots + 2^2 + 2^1 = 2^{\frac{k+1}{2}+1} - 2$$

Zusammengefasst, wenn wir die beiden gerade berechneten Terme entsprechend aufaddieren und vereinfachen, dann erhalten wir:

- Für gerades k gibt es $4 \cdot 2^{\frac{k}{2}} - 3$ Palindrome der Länge höchstens k.
- Für ungerades k gibt es $3 \cdot 2^{\frac{k+1}{2}} - 3$ Palindrome der Länge höchstens k.

(b) Sei $n \in \mathbb{N} - \{0\}$ beliebig und betrachte das n-te Palindrom x_n in kanonischer Ordnung. Wir wählen jetzt die eindeutige kleinste gerade Zahl k, so dass

$$n \leq 4 \cdot 2^{\frac{k}{2}} - 3$$

gilt. Der Term auf der rechten Seite dieser Ungleichung ist genau die in (a) berechnete Anzahl Palindrome der Länge höchstens k. Also muss das Palindrom x_n auch Länge höchstens k haben. Weil wir aber das k so klein wie nur möglich gewählt haben, muss tatsächlich $|x_n| = k$ (falls x_n gerade Länge hat) oder $|x_n| = k - 1$ (falls x_n ungerade Länge hat) gelten. In jedem Fall gilt also $k \leq |x_n| + 1$ für das von uns gewählte k.

Es ist sicher möglich, ein Programm zu schreiben, welches das Entscheidungsproblem für Palindrome löst (d.h. ein Programm, welches für ein gegebenes Wort $x \in \Sigma^*$ zuverlässig entscheidet, ob es sich bei x um ein Palindrom handelt oder nicht). Deshalb können wir tatsächlich Satz 2.2 anwenden und erhalten

$$\begin{aligned} K(x_n) &\leq \lceil \log_2(n+1) \rceil + c \\ &\leq \lceil \log_2(4 \cdot 2^{\frac{k}{2}} - 3 + 1) \rceil + c \\ &\leq \frac{k}{2} + 2 + c \\ &\leq \frac{|x_n| + 1}{2} + 2 + c \\ &= \frac{1}{2} \cdot |x_n| + 2.5 + c. \end{aligned}$$

Für die beiden gewünschten Konstanten wählen wir also $\alpha = \frac{1}{2}$ und $d = 2.5 + c$, wobei die Konstante c aus der Anwendung von Satz 2.2 entstammt.

Bemerkung: Anstatt die etwas ungenaue Ungleichung $k \leq |x_n| + 1$ zu verwenden, hätte man die beiden Fälle für "$|x_n|$ gerade" und "$|x_n|$ ungerade" auch getrennt betrachten können. Dies hätte aber nur zu einer kleinen Verbesserung der Konstanten d geführt, weshalb wir darauf verzichtet haben.

(c) Da die Länge der Palindrome x_n mit wachsendem n monoton ansteigt und auch beliebig gross werden kann, gibt es eine hinreichend grosse Zahl n_0, so dass für alle $n \geq n_0$ die Ungleichung $d < \alpha \cdot |x_n|$ gilt und somit auch

$$K(x_n) \leq \alpha \cdot |x_n| + d < 2\alpha \cdot |x_n| = |x_n|.$$

Folglich gibt es höchstens $n_0 - 1$ (also endlich viele) Palindrome, die als zufällig betrachtet werden können.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm

Lösungsvorschläge – Blatt 3

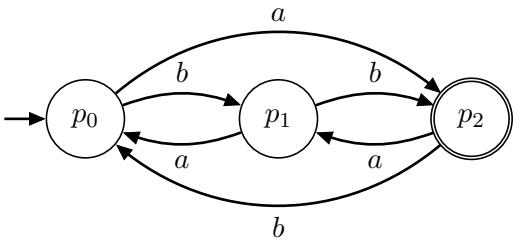
Zürich, 13. Oktober 2023

Lösung zu Aufgabe 7

(a) Es gilt

$$\begin{aligned} L_1 &= \{w \in \{a,b\}^* \mid (2 \cdot |w|_a + |w|_b) \bmod 3 = 2\} \\ &= \{w \in \{a,b\}^* \mid (-|w|_a + |w|_b) \bmod 3 = 2\}. \end{aligned}$$

Der folgende endliche Automat akzeptiert die Sprache L_1:



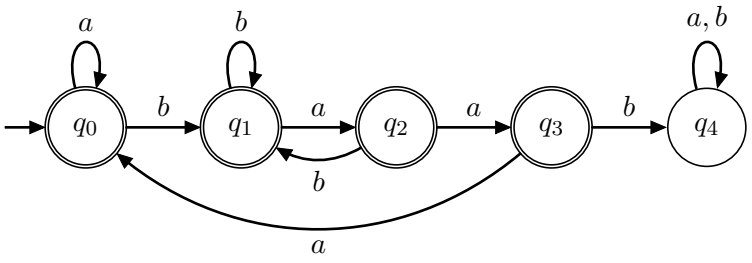
Dieser Automat zählt einfach die Buchstaben in der Eingabe modulo 3, wobei entsprechend der geforderten Bedingung jedes a negativ und jedes b positiv gezählt wird. Damit gilt für die Klassen, dass

$$\mathrm{Kl}[p_i] = \{w \in \{a,b\}^* \mid (-|w|_a + |w|_b) \bmod 3 = i\}$$

für $i \in \{0,1,2\}$.

(b) Der folgende endliche Automat akzeptiert die Sprache

$$L_2 = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } baab \text{ nicht}\}.$$



Der Automat arbeitet mit denselben Transitionen wie ein Automat, der einfach das Muster *baab* in der Eingabe sucht. Im Zustand q_i wurde an der aktuellen Position der Eingabe bereits ein Präfix der Länge i des Musters gefunden. Weil der Automat alle Wörter akzeptieren soll, die das Muster nicht enthalten, sind alle Zustände bis auf q_4 akzeptierend.

Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned}
\mathrm{Kl}[q_4] &= \{a,b\}^* - L_2\,,\\
\mathrm{Kl}[q_3] &= \{xbaa \mid x \in \{a,b\}^*\} \cap L_2\,,\\
\mathrm{Kl}[q_2] &= \{xba \mid x \in \{a,b\}^*\} \cap L_2\,,\\
\mathrm{Kl}[q_1] &= \{xb \mid x \in \{a,b\}^*\} \cap L_2\,,\\
\mathrm{Kl}[q_0] &= \{a,b\}^* - \bigcup_{i=1}^{4} \mathrm{Kl}[q_i]\,.
\end{aligned}$$

Man bemerke, dass es oftmals hilfreich ist, die Reihenfolge der Klassenbeschreibungen geschickt auszuwählen.

Lösung zu Aufgabe 8

Sei $\Sigma = \{a,b\}$ und sei $G \subseteq \Sigma^*$ die Menge aller Wörter über Σ mit gerader Länge. Die vier Klassen und die akzeptierte Sprache lassen sich nun wie folgt beschreiben:

$$\begin{aligned}
\mathrm{Kl}[q_0] &= \{xaa, xbb \mid x \in G\} \cup \{\lambda\}\\
\mathrm{Kl}[q_1] &= \{xa \mid x \in G\}\\
\mathrm{Kl}[q_2] &= \{xb \mid x \in G\}\\
\mathrm{Kl}[q_3] &= \{xab, xba \mid x \in G\}\\
L(M) &= \mathrm{Kl}[q_2] \cup \mathrm{Kl}[q_3] = \{xb, xab, xba \mid x \in G\}
\end{aligned}$$

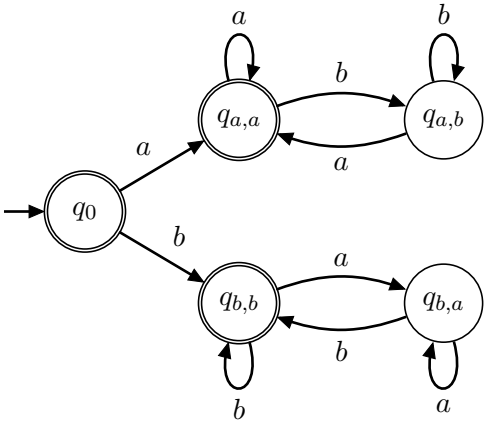
Begründung: Als Erstes beobachtet man, dass alle Wörter mit gerader Länge in einem der Zustände q_0 oder q_3 landen, während alle Wörter mit ungerader Länge in einem der Zustände q_1 oder q_2 landen. Dies stimmt sicher für das leere Wort λ, welches gerade Länge hat und einfach im Startzustand q_0 verbleibt. Für alle anderen Wörter reicht es dann aus, zu beobachten, dass M bei jedem weiteren gelesenen Symbol immer zwischen den beiden Zuständen q_0, q_3 und den beiden Zuständen q_1, q_2 hin- und herspringt.

Als Zweites betrachtet man ein beliebiges nichtleeres Wort $w \in \Sigma^* - \{\lambda\}$. Dieses Wort w lässt sich eindeutig schreiben in einer der Formen $xa, xb, xaa, xab, xba, xbb$, wobei $x \in G$ ein Wort gerader Länge ist. Nach dem Abarbeiten des geraden Präfixes x wird sich der Automat M, wie oben erwähnt, entweder im Zustand q_0 oder q_3 befinden. Man kann nun also einfach für alle möglichen Suffixes a, b, aa, ab, ba, bb systematisch überprüfen, dass M ausgehend von den Zuständen q_0 und q_3 immer im gewünschten Zustand endet.

Lösung zu Aufgabe 9

(a) Der folgende endliche Automat akzeptiert die Sprache

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}\,.$$



Beim Lesen der Buchstaben in w von links nach rechts kommt ab genau für jeden Wechsel von a nach b einmal als Teilwort vor und analog ba genau einmal für jeden Wechsel von b nach a. Somit kommen ab und ba genau dann gleich oft in w als Teilwort vor, wenn w entweder das leere Wort λ ist oder w mit demselben Buchstaben beginnt mit dem es endet. Letztere Bedingung ist insbesondere für $w = a$ und für $w = b$ erfüllt. Wenn w mit a beginnt, prüft der Automat in den beiden oberen Zuständen, ob der zuletzt gelesene Buchstabe mit dem zuerst gelesenen übereinstimmt. Wenn w mit b beginnt, tut er dasselbe in den beiden unteren Zuständen.

(b) Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned}\mathrm{Kl}[q_0] &= \{\lambda\} \text{ und}\\ \mathrm{Kl}[q_{x,y}] &= \{w \in \{a,b\}^* \mid w \text{ beginnt mit } x \text{ und endet mit } y\} \text{ für alle } x, y \in \{a,b\}.\end{aligned}$$

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Denni Komm

Lösungsvorschläge – Blatt 4

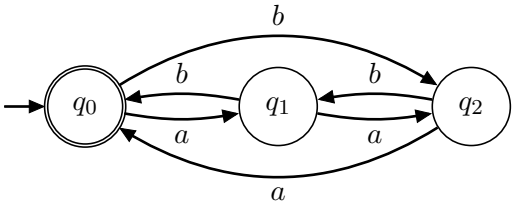
Zürich, 20. Oktober 2023

Lösung zu Aufgabe 10

Die in der Aufgabenstellung gegebene Sprache L lässt sich schreiben als $L = L_1 \cup L_2$ mit

$$\begin{aligned} L_1 &= \{w \in \{a,b\}^* \mid |w|_a \bmod 3 = |w|_b \bmod 3\} \\ &= \{w \in \{a,b\}^* \mid (|w|_a - |w|_b) \bmod 3 = 0\}\,, \\ L_2 &= \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } ba \text{ und endet mit } a\}\,. \end{aligned}$$

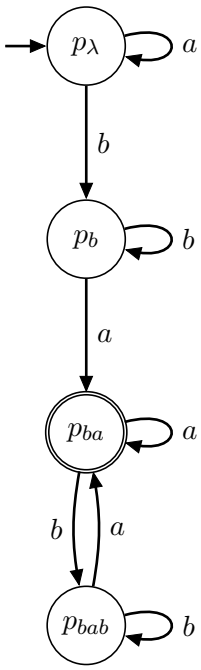
Für die Sprache L_1 lässt sich der folgende Automat A_1 konstruieren:



Dieser Automat zählt in seinen Zuständen die Anzahl von as und bs modulo 3 gemäss der Formel aus der Definition von L_1. Es gilt für $i \in \{0,1,2\}$, dass

$$\mathrm{Kl}[q_i] = \{w \in \{a,b\}^* \mid (|w|_a - |w|_b) \bmod 3 = i\}\,.$$

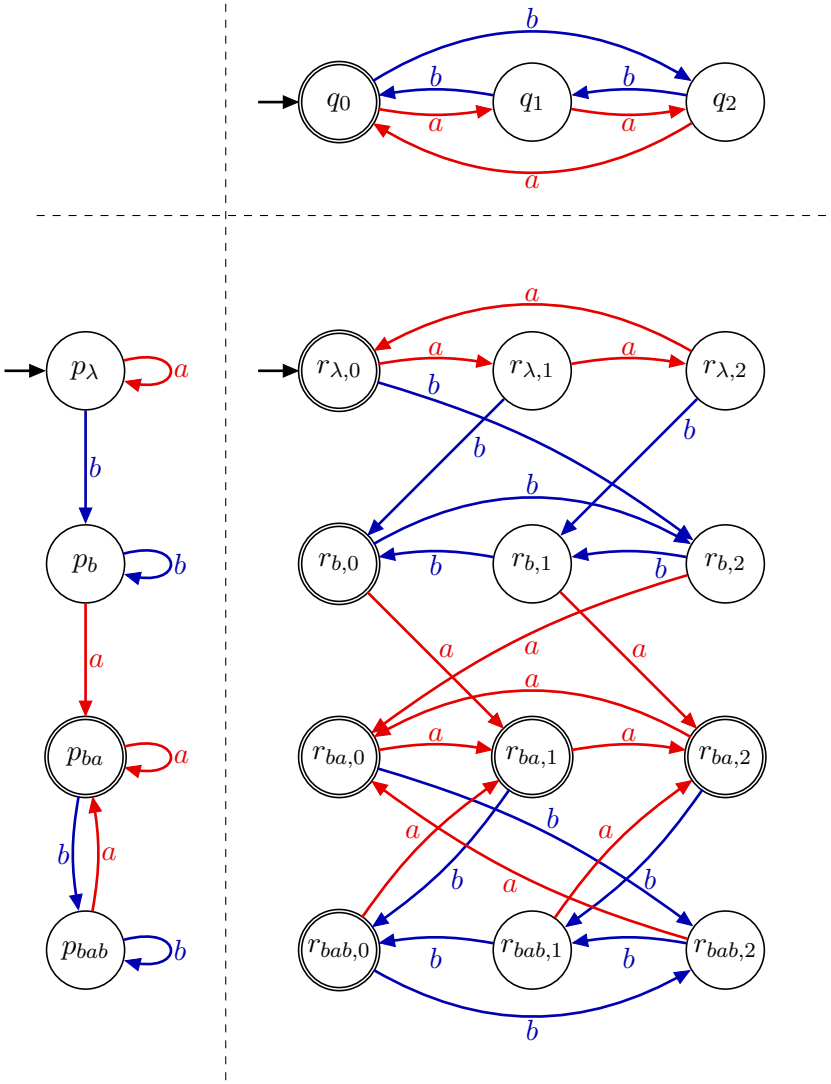
Für die Sprache L_2 lässt sich der folgende Automat A_2 konstruieren:



Die Zustände p_z für $z \in \{\lambda, b, ba\}$ geben an, welches längste Präfix des gesuchten Musters ba aktuell gelesen wurde, p_{ba} ist der akzeptierende Zustand. Der Zustand p_{bab} wird erreicht, wenn das Muster ba bereits gefunden wurde und das aktuell gelesene Präfix des Wortes mit b endet. Damit ergeben sich die folgenden Klassen:

$$\begin{aligned}
\mathrm{Kl}[p_\lambda] &= \{a\}^* \,, \\
\mathrm{Kl}[p_b] &= \{xby \mid x \in \{a\}^* \text{ und } y \in \{b\}^*\} \,, \\
\mathrm{Kl}[p_{ba}] &= L_2 \,, \\
\mathrm{Kl}[p_{bab}] &= \{a, b\}^* - \bigcup_{z \in \{\lambda, b, ba\}} \mathrm{Kl}[p_z] \,.
\end{aligned}$$

Mit der Methode des modularen Entwurfs kann man nun aus A_1 und A_2 den folgenden Produktautomaten A konstruieren, der die Sprache $L = L_1 \cup L_2$ akzeptiert. Zur einfacheren Darstellung verwenden wir die Notation $r_{x,y} = \langle p_x, q_y \rangle$. Weil dies ein Produktautomat für die Vereinigung von zwei Sprachen ist, sind alle Zustände akzeptierend, die einen akzeptierenden Zustand aus einem der beiden Teilautomaten enthalten, also die zu q_0 gehörige Spalte und die zu p_{ba} gehörige Zeile.



Lösung zu Aufgabe 11

(a) Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache

$$L_1 = \{a^i b^j \mid i, j \in \mathbb{N} \text{ so dass } j < i\}$$

nicht regulär ist. Angenommen, L_1 sei regulär. Dann gibt es einen Automaten $A_1 = (Q, \{a, b\}, \delta, q_0, F)$ mit $L(A_1) = L_1$. Sei $m = |Q|$. Wir betrachten die Wörter

$$a^k \quad \text{für } k \in \{1, \ldots, m+1\}\,.$$

Weil dies $m+1$ Wörter sind, also mehr Wörter als A_1 Zustände hat, gibt es $k, l \in \{1, \ldots, m+1\}$ mit $k < l$, so dass

$$\hat{\delta}(q_0, a^k) = \hat{\delta}(q_0, a^l)\,.$$

Nach Lemma 3.3 gilt nun für alle $z \in \{a, b\}^*$, dass

$$a^k z \in L_1 \iff a^l z \in L_1.$$

Die Wahl von $z = b^{l-1}$ führt aber zum Widerspruch, weil

$$a^k z = a^k b^{l-1} \notin L_1\,,$$

$$a^l z = a^l b^{l-1} \in L_1$$

gilt. Dabei folgt erstere Aussage daraus, dass $k < l$ die Ungleichung $k \leq l - 1$ für die beiden Exponenten impliziert. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache

$$L_2 = \{wbbbv \mid w, v \in \{a, b\}^* \text{ mit } |w|_a = |v|_a\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{a, b\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

(i) $|yx| \leq n_0$,

(ii) $|x| \geq 1$ und

(iii) entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = a^{n_0} bbba^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = a^l$ und $x = a^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$, also insbesondere $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^k z \mid k \in \mathbb{N}\} = \{a^{n_0+(k-1)m} bbba^{n_0} \mid k \in \mathbb{N}\} \subseteq L_2 \,.$$

Dies ist aber ein Widerspruch, da $yx^2 z = a^{n_0+m} bbba^{n_0} \notin L_2$. Also ist die Annahme falsch und L_2 ist nicht regulär.

Lösung zu Aufgabe 12

(a) Wir wollen mit Hilfe der Kolmogorov-Komplexität beweisen, dass L_1 nicht regulär ist. Wir nehmen deshalb widerspruchshalber an, dass L_1 regulär sei.

Für jedes $n \in \mathbb{N}$, $n \geq 2$, betrachten wir nun das Wort $z = 0^{F_{n+1}-F_n-1}$. Mit Hilfe der Fibonacci-Rekursion lässt sich dieses Wort auch einfacher als $z = 0^{F_{n-1}-1}$ schreiben (man beachte, dass die Annahme $n \geq 2$ notwendig war, weil sonst jetzt eine negative Zahl im Exponenten stehen würde).

Wir behaupten nun, dass dieses betrachtete Wort z das erste Wort (in kanonischer Ordnung) in der Sprache

$$L' = \{y \in \{0, 1\}^* \mid 0^{F_n+1} y \in L_1\} \,,$$

ist. Tatsächlich, es gilt offenbar $0^{F_n+1} z = 0^{F_n+1} 0^{F_{n+1}-F_n-1} = 0^{F_{n+1}} \in L_1$ und es gibt auch kein kürzeres Wort in der Sprache L', weil die Fibonacci-Folge monoton wachsend ist. Nach Satz 3.1 aus dem Buch existiert also eine Konstante c, die nur von der Sprache L_1 abhängt aber nicht von der Zahl n, so dass

$$K(z) = K(0^{F_{n-1}-1}) \leq \lceil \log_2(1 + 1) \rceil + c = 1 + c \,.$$

Da es nur endlich viele Programme der konstanten Länge $1 + c$ (oder kürzer) gibt, aber unendlich viele Wörter der Form $z = 0^{F_{n-1}-1}$, ist dies ein Widerspruch. Also war die Annahme falsch und L_1 ist nicht regulär.

(b) Wir zeigen im Folgenden, dass man Lemma 3.3 sowie das Pumping-Lemma verwenden kann, um zu zeigen, dass L_2 nicht regulär ist. Die Methode der Kolmogorov-Komplexität ist in diesem Fall nicht direkt anwendbar.

- Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache L_2 nicht regulär ist. Angenommen, L_2 sei regulär. Dann gibt es einen Automaten $A = (Q, \{a, b, c\}, \delta, q_0, F)$ mit $L(A) = L_2$. Sei $m = |Q|$. Wir betrachten die Wörter

  $$\lambda, abc, (abc)^2, \ldots, (abc)^m .$$

  Weil dies $m + 1$ Wörter sind, also mehr Wörter als A Zustände hat, gibt es $i, j \in \{0, \ldots, m\}$ mit $i \neq j$, so dass

  $$\hat{\delta}(q_0, (abc)^i) = \hat{\delta}(q_0, (abc)^j) .$$

  Nach Lemma 3.3 gilt nun für alle $z \in \{a, b, c\}^*$, dass

  $$(abc)^i z \in L_2 \iff (abc)^j z \in L_2.$$

  Die Wahl von $z = (bac)^i$ führt aber zum Widerspruch, weil

  $$(abc)^i z = (abc)^i (bac)^i \in L_2$$
  $$(abc)^j z = (abc)^j (bac)^i \notin L_2$$

  gilt. Also war die Annahme falsch und L_2 ist nicht regulär.

- Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache L_2 nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{a, b\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

  (i) $|yx| \leq n_0$,
  (ii) $|x| \geq 1$ und
  (iii) entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

  Wir wählen das Wort $w = (abc)^{n_0}(bac)^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist der Teil x gänzlich in der ersten Hälfte $(abc)^{n_0}$ von w enthalten. Wegen (ii) gilt weiter, dass x mindestens ein Symbol enthält. Im Folgenden führen wir eine Fallunterscheidung über den möglichen Inhalt von x durch, und gelangen mittels (iii) und der Tatsache, dass $w = yxz \in L_2$ gilt, dann jeweils zu einem Widerspruch.

  - Fall 1: Das Wort x hat Länge 1 und besteht nur aus dem Symbol c. In diesem Fall kann man beobachten, dass das Wort $yx^0z = yz$ das Teilwort ba einmal mehr enthält als ab, weil das Symbol c in einer Folge der Form $\ldots abcab \ldots$ gelöscht wird. Somit gilt $yx^0z \notin L_2$, im Widerspruch zu (iii).
  - Fall 2: Das Wort x enthält mindestens ein Symbol das entweder gleich a oder b ist. In diesem Fall betrachten wir das gleiche Wort $yx^0z = yz$. Wir beobachten, dass die Anzahl Teilwörter ba in yz entweder gleich bleibt oder um 1 erhöht wird (aus einem ähnlichen Grund wie bei Fall 1). Desweiteren beobachten wir, dass die Anzahl Teilwörter ab in yz mindestens um 1 kleiner wird. Also gilt auch hier $yx^0z \notin L_2$, im Widerspruch zu (iii).

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm

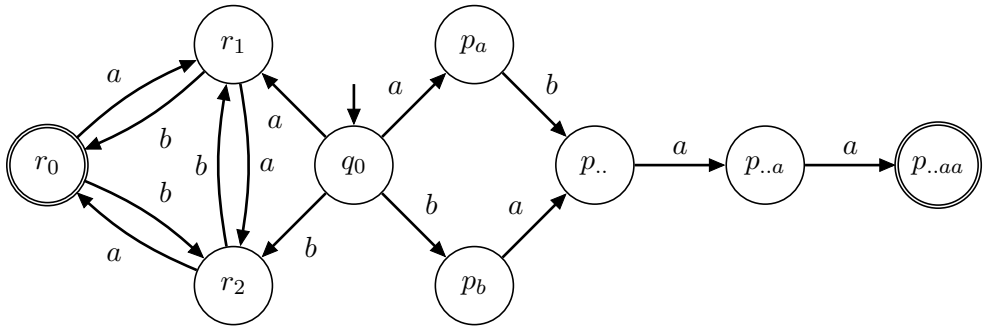
Lösungsvorschläge – Blatt 5

Zürich, 27. Oktober 2023

Lösung zu Aufgabe 13

(a) Der folgende nichtdeterministische endliche Automat akzeptiert die Sprache

$$L = \{w \in \{a,b\}^* \mid (|w|_a - |w|_b \bmod 3 = 0) \text{ oder } w = abaa \text{ oder } w = baaa\} .$$



Dieser Automat besteht aus zwei Teilautomaten für die beiden Sprachen

$$L_1 = \{w \in \{a,b\}^* \mid |w|_a - |w|_b \bmod 3 = 0\}$$

und

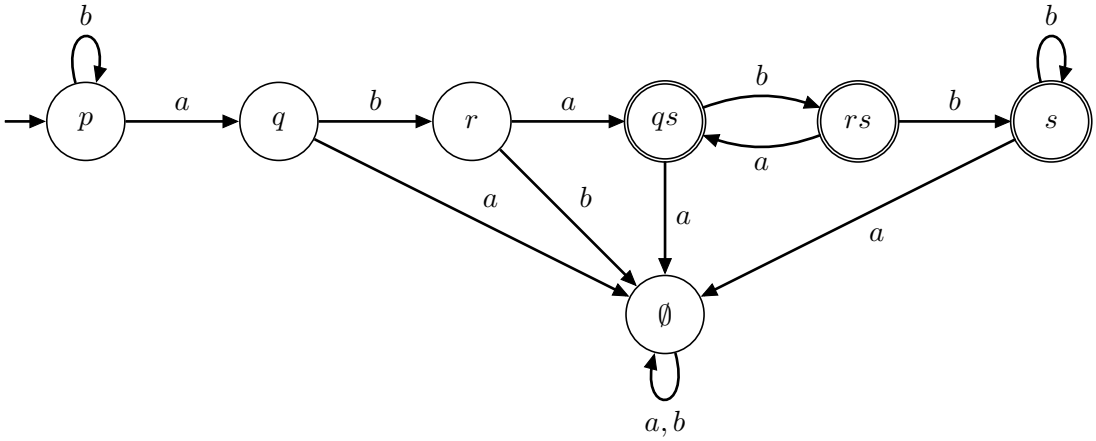
$$L_2 = \{abaa, baaa\}$$

mit $L = L_1 \cup L_2$.

Vom Startzustand q_0 aus verzweigt der Automat einerseits nichtdeterministisch in den linken Teilautomaten mit den Zuständen r_0, r_1, r_2 für die Sprache L_1. In diesen drei Zuständen merkt sich der Automat auf die übliche Art und Weise den Wert des Ausdrucks $|w|_a - |w|_b \bmod 3$ für das bisher gelesene Präfix w. Falls dieser Wert am Schluss 0 ergibt (also falls sich der Automat am Schluss im Zustand r_0 befindet), wird die Eingabe akzeptiert.

Andererseits verzweigt der Automat von q_0 aus auch in den rechten Teilautomaten mit den Zuständen $p_a, p_b, p_{..}, p_{..a}, p_{..aa}$ für die Sprache L_2. In den Zuständen p_a und p_b wird zuerst unterschieden, ob das Wort mit a oder b beginnt. Die restlichen Zustände $p_{..}, p_{..a}, p_{..aa}$ dienen dann dazu den Rest der beiden erlaubten Wörter $abaa$ und $baaa$ abzugleichen und genau diese im Zustand $p_{..aa}$ zu akzeptieren.

(b) Die Anwendung der Potenzmengenkonstruktion auf den auf dem Aufgabenblatt gezeigten nichtdeterministischen endlichen Automaten ergibt den folgenden deterministischen endlichen Automaten. Dabei wurden alle nicht erreichbaren Zustände weggelassen. Zur einfacheren graphischen Darstellung wurde die Beschriftung der Zustände verkürzt, qs steht zum Beispiel für den Zustand $\langle\{q, s\}\rangle$.



Speziell anzumerken ist, dass in vier unterschiedlichen Situationen ein gelesenes Symbol im nichtdeterministischen Automaten keine einzige mögliche Transition zulässt. Dies führt beim konstruierten deterministischen Automaten dann jeweils zu einer Transition in die leere Zustandsmenge $\langle\emptyset\rangle$. Hätten wir auf direktem Wege einen deterministischen Automaten für die erkannte Sprache

$$L = \{b^i aba(ba)^j b^k \mid i, j, k \in \mathbb{N}\}$$

konstruiert, dann wäre dieser spezielle Zustand der sogenannte "Abfallzustand".

Lösung zu Aufgabe 14

(a) Diese Aussage ist korrekt. Da L eine reguläre Sprache ist, existiert ein endlicher Automat $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Daraus konstruieren wir im Folgenden einen nichtdeterministischen endlichen Automaten A_1 mit $L(A_1) = L_1 = \{v\#w^{\mathrm{R}} \mid v, w \in L\}$, woraus dann die Regularität von L_1 folgt.

Sei $A^{\mathrm{R}} = (Q^{\mathrm{R}}, \Sigma, \delta^{\mathrm{R}}, Q_0^{\mathrm{R}}, F^{\mathrm{R}})$ eine Kopie von A mit den folgenden Unterschieden.

- Die Zustandsmenge Q^{R} von A^{R} ist im Prinzip die gleiche wie Q, ausser dass jeder Zustand $q^{\mathrm{R}} \in Q^{\mathrm{R}}$ mit einer speziellen Markierung versehen ist (und daher formal gesehen die Ungleichung $q \neq q^{\mathrm{R}}$ für den entsprechenden Ursprungszustand $q \in Q$ gilt), so dass wir ganz am Schluss dieser Teilaufgabe auf sinnvolle Art und Weise die disjunkte Vereinigung $Q \cup Q^{\mathrm{R}}$ bilden können.
- Die Transitionsfunktion δ^{R} von A^{R} ist die Umkehrung von δ in dem Sinne, dass für jeden Transitionspfeil in A dessen Richtung umgedreht wird, aber das gelesene Symbol beibehalten wird. Durch diese Umkehrung der Pfeile wird der Automat A^{R} typischerweise nichtdeterministisch sein, sogar dann wenn A ein deterministischer Automat ist.

- Anstatt nur einen einzigen Startzustand zu haben wird A^{R} gleich eine ganze Menge $Q_0^{\mathrm{R}} \subseteq Q^{\mathrm{R}}$ von Startzuständen haben. Und zwar definieren wir diese Menge als $Q_0^{\mathrm{R}} = \{q^{\mathrm{R}} \mid q \in F\}$, das heisst, jeder akzeptierende Zustand in A wird zu einem Startzustand in A^{R}. (Eigentlich wäre dies gar nicht zulässig, weil Automaten gemäss Definition aus der Vorlesung immer nur einen Startzustand haben dürfen. Trotzdem werden wir am Schluss dieser Teilaufgabe sehen, dass alles seine Richtigkeit hat, weil wir die Konstruktion A^{R} gar nie als eigenständigen Automaten einsetzen, sondern nur als Teil eines grösseren Automaten, der selber korrekterweise einen einzigen Startzustand hat.)
- Die Menge der akzeptierenden Zustände F^{R} von A^{R} ist die einelementige Menge $F^{\mathrm{R}} = \{q_0^{\mathrm{R}} = (q_0)^{\mathrm{R}}\}$, das heisst, der Startzustand q_0 von A wird zum einzigen akzeptierenden Zustand in A^{R}.

Die Idee hinter obiger Konstruktion ist, dass die Berechnungen in A^{R} (das heisst alle Pfade von einem Startzustand in Q_0^{R} zum akzeptierenden Zustand q_0^{R}) auf natürliche Weise genau den Wörtern w^{R} mit $w \in L$ entsprechen.

Den (nichtdeterministischen) endlichen Automaten $A_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, F_1)$ für die Sprache L_1 konstruieren wir jetzt wie folgt aus A und A^{R}.

- Es sei $Q_1 = Q \cup Q^{\mathrm{R}}$ die disjunkte Vereinigung der beiden Zustandsmengen.
- Es sei $\Sigma_1 = \Sigma \cup \{\#\}$ das um das zusätzliche Symbol # erweiterte Alphabet.
- Es sei δ_1 die Transitionsfunktion, die alle Transitionen von δ und δ^{R} enthält, aber zusätzlich auch alle möglichen Transitionen von jedem Zustand in der Menge F in jeden Zustand in der Menge Q_0^{R}, jeweils mit Symbol #.
- Es sei $q_{0,1} = q_0$ der gleiche Startzustand wie in A.
- Es sei $F_1 = F^{\mathrm{R}} = \{q_0^{\mathrm{R}}\}$ die gleiche einelementige akzeptierende Zustandsmenge wie in A^{R}.

Als Abschluss dieser Teilaufgabe fehlt jetzt noch ein informelles Argument für die Tatsache, dass $L(A_1) = L_1 = \{v\#w^{\mathrm{R}} \mid v, w \in L\}$ gilt.

Dazu beobachten wir zuerst, dass jede akzeptierende Berechnung in A_1 einem Wort der Form $x\#y$ entspricht, weil alle Transitionen von dem Teilautomaten A (der den Startzustand enthält) zum Teilautomaten A^{R} (der den einzigen akzeptierenden Zustand enthält) das Symbol # voraussetzen. Desweiteren gilt $x \in L$ und $y^{\mathrm{R}} \in L$, weil das Wort x vor dem Symbol # einer akzeptierenden Berechnung in A entspricht, während das Wort y nach dem Symbol # einer akzeptierenden Berechnung in A^{R} entspricht. Daraus folgt bereits $L(A_1) \subseteq L_1$.

Umgekehrt lässt sich für jedes Wort $v\#w^{\mathrm{R}} \in L_1$ eine entsprechende akzeptierende Berechnung in A_1 finden. Dazu nehmen wir zwei akzeptierende Berechnungen jeweils für v in A und für w^{R} in A^{R}, die wegen der Voraussetzung $v \in L$ und $w \in L$ existieren müssen. Diese beiden Berechnungen kombinieren wir zu einer akzeptierenden Berechnung in A_1 für das Wort $v\#w^{\mathrm{R}}$, indem wir einfach den passenden Berechnungsschritt mit Symbol # dazwischen einsetzen. Hieraus folgt $L_1 \subseteq L(A_1)$.

(b) Diese Aussage ist falsch. Bevor wir dies beweisen, wollen wir aber noch kurz erkären, wieso die Konstruktion des Automaten A_1 von Teilaufgabe (a) nicht einfach an die Sprache L_2 angepasst werden kann. Man könnte zum Beispiel die Idee haben, dass man nicht alle möglichen Transitionen von der Menge F zu der Menge Q_0^{R} mit

Symbol # einfügt, sondern nur diejenigen Transitionen, die von einem akzeptierenden Zustand $q \in F$ in A zum "gleichen" Startzustand $q^{\mathrm{R}} \in Q_0^{\mathrm{R}}$ in A^{R} gehen. Auf diese Weise könnte man tatsächlich den "richtigen" Startzustand in A^{R} erzwingen mit der Hoffnung, dass der Automat nach dem Lesen eines Wortes w gefolgt vom Symbol # gezwungen wäre, das gleiche Wort rückwärts (also w^{R}) nochmals zu lesen. Weil ein endlicher Automat aber typischerweise verschiedene Berechnungen hat, die im gleichen Zustand enden, reicht es leider nicht aus, wenn man nur den richtigen Startzustand erzwingt.

Wir beweisen jetzt die Nichtregularität der Sprache $L_2 = \{w\#w^{\mathrm{R}} \mid w \in L\}$, wenn wir zum Beispiel die (offensichtlich reguläre) Sprache $L = \{a, b\}^*$ über dem Alphabet $\Sigma = \{a, b\}$ wählen. Damit ist dann auch gezeigt, dass die Aussage von Teilaufgabe (b) nicht im Allgemeinen korrekt ist. Wir nehmen dazu widerspruchshalber an, dass L_2 regulär ist und wählen einen endlichen Automaten A_2 mit $L(A_2) = L_2$, der insgesamt n Zustände hat. Wir betrachten nun die $n+1$ Wörter $\lambda, a, a^2, \ldots, a^n \in L$, von denen mindestens zwei, sagen wir a^i und a^j mit $0 \leq i < j \leq n$, im gleichen Zustand von A_2 landen. Wegen $i \neq j$ gilt jetzt aber offenbar $a^i z \in L_2$ und $a^j z \notin L_2$ für das Wort $z = \#a^i$, im Widerspruch zu Lemma 3.3.

Lösung zu Aufgabe 15

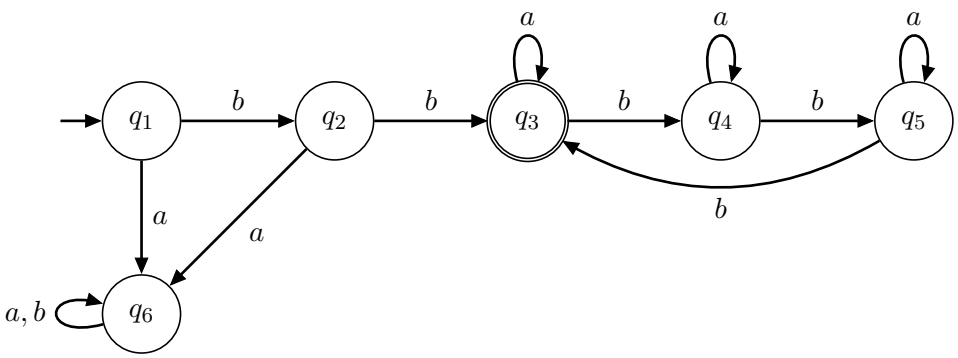
Um zu zeigen, dass jeder deterministische endliche Automat, der die Sprache

$$L = \{w \in \{a, b\}^* \mid |w|_b \bmod 3 = 2 \text{ und } w \text{ beginnt mit } bb\}$$

akzeptiert, mindestens 6 Zustände benötigt, bestimmen wir zuerst 6 Wörter $w_1, \ldots, w_6$ und zeigen dann, dass diese Wörter den Automaten in 6 paarweise verschiedene Zustände führen müssen. Dazu verwenden wir Lemma 3.3 aus dem Buch, welches besagt, dass ein Automat, der nach dem Lesen von zwei verschiedenen Wörtern w_i und w_j im gleichen Zustand endet, auch für die Wörter $w_i z$ und $w_j z$ im gleichen Zustand enden muss, wobei $z \in \{a, b\}^*$ ein beliebiges Suffix ist. Es reicht deshalb für je zwei unterschiedliche Wörter w_i und w_j ein Suffix $z_{i,j}$ zu finden, so dass

$$(w_i z_{i,j} \in L \text{ und } w_j z_{i,j} \notin L) \text{ oder } (w_i z_{i,j} \notin L \text{ und } w_j z_{i,j} \in L)\,. \tag{1}$$

Eine allgemeine Strategie, um geeignete Wörter $w_1, \ldots, w_6$ zu bestimmen, besteht darin, zunächst einen Automaten A mit 6 Zuständen $q_1, \ldots, q_6$ für die Sprache L zu entwerfen, und dann für jeden Zustand q_i von A das Wort w_i als das kürzeste Wort aus der Klasse $\mathrm{Kl}[q_i]$ zu wählen. Der folgende Automat A mit 6 Zuständen akzeptiert die Sprache L.



Die kürzesten Wörter in den Zustandsklassen von A sind $w_1 = \lambda$, $w_2 = b$, $w_3 = bb$, $w_4 = bbb$, $w_5 = bbbb$ und $w_6 = a$. Die folgende Tabelle gibt für alle Paare (w_i, w_j) mit $i < j$ jeweils ein Suffix $z_{i,j}$ an, das die Bedingung (1) erfüllt.

	$w_2 = b$	$w_3 = bb$	$w_4 = bbb$	$w_5 = bbbb$	$w_6 = a$
$w_1 = \lambda$	bb	bb	abb	bb	bb
$w_2 = b$	$-$	b	b	ab	b
$w_3 = bb$	$-$	$-$	λ	λ	λ
$w_4 = bbb$	$-$	$-$	$-$	bb	bb
$w_5 = bbbb$	$-$	$-$	$-$	$-$	b

Es ist leicht zu überprüfen, dass diese Suffixe die Bedingung (1) tatsächlich erfüllen. Zum Beispiel ist $w_2 z_{2,5} = bab \notin L$, aber $w_5 z_{2,5} = bbbbab \in L$, woraus mit Lemma 3.3 dann folgt, dass die Wörter $w_2 = b$ und $w_5 = bbbb$ gezwungenermassen in unterschiedlichen Zuständen landen müssen.

Lösung zu Bonus-Aufgabe 1

(a) Diese Teilaufgabe sollte mittlerweile Routine sein. Wir nehmen wie immer widerspruchshalber an, dass die Sprache

$$L = \{0,1\}^* - \{0^n 1^n \mid n \in \mathbb{N}\}$$

regulär sei. Laut dem Pumping-Lemma für reguläre Sprachen existiert dann eine Konstante $n_0 \in \mathbb{N}$, so dass insbesondere für das Wort $w = 0^{n_0}1^{n_0}$ mit $|w| \geq n_0$ eine Zerlegung $w = yxz$ exisiert, wobei

(i) $|yx| \leq n_0$,

(ii) $|x| \geq 1$ und

(iii) entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L = \emptyset$.

Aus (i) folgt $y = 0^i$ und $x = 0^j$ für $i, j \in \mathbb{N}$ mit $i + j \leq n_0$. Aus (ii) folgt weiter, dass $j \geq 1$. Aus (iii) und der Tatsache, dass $w \notin L$, folgt

$$yx^k z = 0^i 0^{kj} 0^{n_0 - i - j} 1^{n_0} = 0^{n_0 + (k-1)j} 1^{n_0} \notin L \text{ für alle } k \in \mathbb{N} .$$

Jedoch führt zum Beispiel die Wahl $k = 2$ zu der Aussage $0^{n_0+j}1^{n_0} \notin L$, was wegen $j \geq 1$ natürlich nicht stimmen kann.

(b) Die zündende Idee bei dieser Teilaufgabe ist, dass die Sprache $\{0^n 1^n \mid n \in \mathbb{N}\}$ nur *sehr wenige* Wörter enthält (nämlich pro möglicher Wortlänge höchstens ein Wort) und dass deshalb das Komplement

$$L = \{0,1\}^* - \{0^n 1^n \mid n \in \mathbb{N}\}$$

als Sprache über dem binären Alphabet *fast alle* möglichen Wörter enthält. Die Konsequenz daraus wird sein, dass das n-te Wort y in L (oder in L_x für ein beliebiges $x \in \{0,1\}^*$) ungefähr die Länge $\log_2(n)$ hat, was bedeutet, dass wir ein Programm der ungefähren Länge $\log_2(n)$ schreiben können, welches das Wort y fest im Programmcode integriert und ganz einfach dieses y liest und ausgibt.

Seien also $x, y \in \{0,1\}^*$ so, dass y das n-te Wort in L_x ist, aber ansonsten völlig beliebig. Folgendes Programm gibt y aus und hat die binäre Länge $|y| + c$, wobei c eine von x, y und n unabhängige Konstante ist.

```
begin
  w := y;
  write(w);
end.
```

Somit gilt $K(y) \leq |y| + c$. Wenn wir zusätzlich zeigen können, dass die Ungleichung

$$|y| \leq \lceil \log_2(n+1) \rceil + d$$

gilt für eine weitere von x, y und n unabhängige Konstante d, dann ist die gewünschte Aussage von dieser Teilaufgabe bewiesen.

Wir betrachten nun diejenigen Wörter $w \in L_x$, deren Länge kleiner als die Länge von y ist, für die also $|w| < |y|$ gilt. All diese Wörter w kommen wegen der kürzeren Länge vor y in kanonischer Ordnung. Zudem gibt es mindestens

$$2^0 + 2^1 + \cdots + 2^{|y|-1} - |y| = 2^{|y|} - 1 - |y|$$

viele solche Wörter w (der Term $-|y|$ kommt daher, dass wir für jede mögliche Wortlänge $0, 1, \ldots, |y| - 1$ maximal ein Wort verlieren). Für den Index des n-ten Worts y in L_x gilt deshalb

$$2^{|y|} - |y| \leq n \; .$$

Man kann sich leicht überlegen, dass auch $|y| \leq n$ gelten muss, und wir erhalten somit die etwas handlichere Ungleichung

$$2^{|y|} \leq 2n \; .$$

Durch Nehmen des Logarithmus auf beiden Seiten schliessen wir den Beweis ab, denn es gilt wie gewünscht

$$|y| \leq \log_2(2n) = \log_2(n) + 1 \leq \lceil \log_2(n+1) \rceil + 1 \; .$$

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm

Lösungsvorschläge – Blatt 6

Zürich, 3. November 2023

Lösung zu Aufgabe 16

(a) Wir nehmen an, dass die Gäste wie folgt durchnummeriert sind: Sei $G_{0,i}$ der bereits im Hotel wohnende Gast im Zimmer Z_i für $i \in \mathbb{N}$, sei $G_{j,i}$ der i-te Gast im Bus j für $j \in \{1,2,3\}$ und $i \in \mathbb{N}$.

Um alle Wünsche der Gäste zu berücksichtigen, kann der Portier zum Beispiel die folgende Aufteilung wählen. Er teilt die Folge der Zimmer in Blöcke der Grösse 6 auf und vergibt die Zimmer mit Nummern 0 mod 6 an die bereits anwesenden Gäste, die Zimmer mit Nummern 1 mod 6 an die Gäste aus dem ersten Bus, die Zimmer mit Nummern 2 mod 6 und 3 mod 6 an die Gäste aus dem zweiten Bus und die Zimmer mit Nummern 4 mod 6 und 5 mod 6 an die Gäste aus dem dritten Bus.

Formal lässt sich diese Zimmerbelegung für alle $i \in \mathbb{N}$ durch eine Funktion f angeben mit

$$
\begin{aligned}
&f(G_{0,i}) = Z_{6i}, \\
&f(G_{1,i}) = Z_{6i+1}, \\
&f(G_{2,i}) = Z_{6\cdot\frac{i}{2}+2} = Z_{3i+2}, && \text{falls } i \text{ gerade}, \\
&f(G_{2,i}) = Z_{6\cdot\frac{i-1}{2}+3} = Z_{3i}, && \text{falls } i \text{ ungerade}, \\
&f(G_{3,i}) = Z_{6\cdot\frac{i}{2}+4} = Z_{3i+4}, && \text{falls } i \text{ gerade}, \\
&f(G_{3,i}) = Z_{6\cdot\frac{i-1}{2}+5} = Z_{3i+2}, && \text{falls } i \text{ ungerade}.
\end{aligned}
$$

(b) Der Portier teilt zunächst die Zimmer in unendlich viele Gruppen unendlicher Grösse ein. Dann kann er für jede eintreffende Gruppe von Gästen (unabhängig davon, ob diese endlich oder unendlich ist) die nächste noch nicht belegte Zimmergruppe verwenden.

Um die Zimmergruppen zu bestimmen, verwenden wir die folgende Idee: Wir wissen, dass es unendlich viele Primzahlen gibt. Sei p_i die i-te Primzahl in aufsteigender Reihenfolge. Dann können wir die i-te Gruppe von Zimmernummern definieren als $\text{Gruppe}(i) = \{p_i^j \mid j \in \mathbb{N}^+\}$. Hierbei wird kein Zimmer mehreren Gruppen zugeordnet, weil jede Zahl eine eindeutige Primfaktorzerlegung besitzt.

Lösung zu Aufgabe 17

Ungleichung $\leq$: Für den Beweis, dass $\mathcal{P}(\Sigma^*_{\text{bool}})$ überabzählbar ist, haben wir in der Vorlesung bereits gezeigt, dass

$$|[0,1]| \leq |\mathcal{P}(\Sigma^*_{\text{bool}})|.$$

Um die gewünschte Ungleichung zu erhalten, reicht es also aus, eine injektive Abbildung $f\colon \mathcal{P}(\Sigma^*_{\text{bool}}) \to \mathcal{P}(\mathbb{Q}^+)$ anzugeben. Tatsächlich wird diese Abbildung f aber sogar bijektiv sein. Dazu identifizieren wir ganz einfach das i-te Wort w_i in Σ^*_{bool} mit der i-ten positiven rationalen Zahl q_i in $\mathbb{Q}^+$. Hierbei verwenden wir, dass sowohl Σ^*_{bool} als auch $\mathbb{Q}^+$ abzählbar sind (siehe Lemma 5.1 und Satz 5.2 aus dem Buch) und wir deshalb von den i-ten Elementen in diesen Mengen überhaupt sprechen dürfen. Ein bisschen formaler definieren wir für alle Teilmengen $S \subseteq \Sigma^*_{\text{bool}}$ das Bild $f(S) := Q \subseteq \mathbb{Q}^+$ so, dass

$$q_i \in Q \iff w_i \in S \text{ für alle } i \in \mathbb{N}.$$

Ungleichung $\geq$: Eine Teilmenge $Q \subseteq \mathbb{Q}^+$ kann als unendlicher Bitvektor $(b_1, b_2, \ldots)$ dargestellt werden, wobei $b_i = 1$ genau dann, wenn die i-te positive rationale Zahl aus $\mathbb{Q}^+$ in Q enthalten ist (wir verwenden hier wieder Satz 5.2 aus dem Buch). Es reicht deshalb aus, zu zeigen, dass für die Menge $\mathcal{B}$ solcher unendlichen Bitvektoren $|\mathcal{B}| \leq |[0,1]|$ gilt. Hierzu konstruieren wir eine injektive Abbildung $f\colon \mathcal{B} \to [0,1]$, die wie folgt definiert ist:

$$f((b_1, b_2, \ldots)) := \sum_{i=1}^{\infty} b_i \cdot 10^{-i}\,.$$

Hier ist es wichtig, dass wir die Folge $(b_1, b_2, \ldots)$ als *Dezimalbruch*, und nicht als *Dualbruch* betrachten. Deshalb ist zum Beispiel $0.0\overline{1}$ nicht dieselbe Zahl wie $0.1\overline{0}$ (hingegen wäre tatsächlich $0.0\overline{9} = 0.1\overline{0}$). Bei Dualdarstellung (also mit $b_i \cdot 2^{-i}$ in der Formel) gölte die Injektivität deshalb nicht, aber bei Dezimaldarstellung kann diese Uneindeutigkeit nicht auftreten, weil wir die Ziffer 9 gar nicht verwenden.

Lösung zu Aufgabe 18

(a) Zuerst entscheidet sich der λ-NEA nichtdeterministisch und ohne ein Symbol des Eingabewortes zu lesen, ob er in den linken Teilautomaten mit Zuständen l_1 und l_2 (der alle Wörter ohne das Teilwort bb akzeptiert) oder in den rechten Teilautomaten mit Zuständen r_1 und r_2 (der alle Wörter mit ungerader Anzahl a's akzeptiert) wechseln möchte. Damit ist bereits das "oder" in der Definition der Sprache L erklärt.

Der rechte Teilautomat ist deterministisch und es ist leicht zu sehen, dass der verwerfende Zustand r_1 den Wörtern mit gerader Anzahl a's entspricht, während der akzeptierende Zustand r_2 gerade den Wörtern mit ungerader Anzahl a's entspricht. Damit ist auch der Teil nach dem "oder" in der Definition von L erklärt.

Es bleibt der linke Teilautomat, der wegen der zusätzlichen λ-Transition ein bisschen schwieriger zu verstehen ist. Wir erkennen aber, dass wegen der Absenz weiterer Transitionen immer zwischen den beiden Zuständen l_1 und l_2 hin- und hergesprungen wird. Bei jedem Sprung hin zu l_2 wird entweder kein Symbol oder ein Symbol b gelesen, während bei jedem Sprung zurück zu l_1 immer genau ein Symbol a gelesen wird. Damit ist klar, dass das Symbol b niemals doppelt hintereinander (also das Teilwort

bb) gelesen werden kann. Umgekehrt ist es für jedes Wort, das nicht das Teilwort bb enthält, einfach zu sehen, dass es eine akzeptierende Berechnung ausgehend vom Zustand l_1 im linken Teilautomaten gibt. Damit ist auch der Teil vor dem "oder" in der Definition von L erklärt, und das Argument ist vollständig.

(b) Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein λ-NEA. Sei für alle Zustände $q \in Q$ die λ-Hülle $H(q)$ definiert als die Menge aller Zustände, die sich von q aus durch eine Folge von λ-Transitionen erreichen lassen, also

$$H(q) := \{p \in Q \mid p \in \hat{\delta}(q, \lambda)\}\,.$$

Diese Definition lässt sich kanonisch erweitern auf Mengen von Zuständen: Für $P \subseteq Q$ ist $H(P) := \bigcup_{p \in P} H(p)$.

Dann können wir einen äquivalenten NEA $A_N = (Q, \Sigma, \delta_N, q_0, F_N)$ wie folgt definieren:

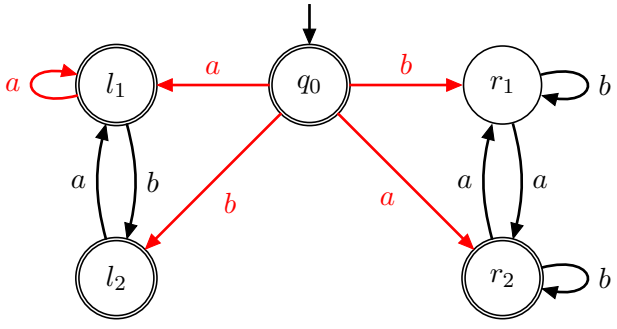
$$\delta_N(q, a) := \bigcup_{p \in H(q)} \delta(p, a)$$

für alle $q \in Q$ und alle $a \in \Sigma$ und

$$F_N := \{p \in Q \mid \hat{\delta}(p, \lambda) \cap F \neq \emptyset\}\,.$$

Man beachte, dass $q \in H(q)$ gilt für alle $q \in Q$. Damit gilt insbesondere $\delta(q, a) \subseteq \delta_N(q, a)$ für alle $q \in Q$ und alle $a \in \Sigma$. Dadurch, dass alle Zustände aus $H(q)$ berücksichtigt werden, sind alle Folgen von λ-Transitionen abgedeckt, die von q aus zu dem Startpunkt einer a-Transition führen. Weil alle Zustände, von denen aus ein akzeptierender Zustand durch eine Folge von λ-Transitionen erreichbar ist, in dem neuen Automaten selbst zu akzeptierenden Zuständen werden, ist garantiert, dass jedes von A akzeptierte Wort auch in A_N akzeptiert wird.

(c) Für den λ-NEA von Teilaufgabe (a) gilt $H(q_0) = \{q_0, l_1, l_2, r_1\}$, $H(l_1) = \{l_1, l_2\}$, $H(l_2) = \{l_2\}$, $H(r_1) = \{r_1\}$ und $H(r_2) = \{r_2\}$. Also ergibt sich mit dem in Teilaufgabe (b) beschriebenen Verfahren der folgende äquivalente NEA:



In der Abbildung sind die neu eingefügten Transitionen rot gezeichnet. Man beachte, dass der Startzustand q_0 neu auch akzeptierend ist, weil er im λ-NEA über λ-Transitionen mit den akzeptierenden Zuständen l_1 und l_2 verbunden war.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
Prof. Dr. Dennis Komm

Lösungsvorschläge – Blatt 2

Zürich, 4. Oktober 2024

Lösung zu Aufgabe 4

(a) Wir geben zunächst für jedes $n \in \mathbb{N} - \{0\}$ ein Pascal-Programm an, das w_n erzeugt:

```
begin
  s:= n;
  s:= 4*s*s;
  for i:=1 to s do
    write(0101);
end.
```

Dieses Programm berechnet zuerst $s = 4n^2$. In der `for`-Schleife gibt das Programm dann $4n^2$-mal die Zeichenfolge 0101 aus, was genau das Wort w_n ergibt.

Der einzige Teil des Maschinencodes dieses Programms, der von w_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Programmcode hat eine konstante Länge. Also ist die binäre Länge dieser Programme höchstens $\lceil \log_2(n+1) \rceil + c$ für eine Konstante c. Damit lässt sich die Kolmogorov-Komplexität von w_n von oben abschätzen durch

$$K(w_n) \leq \lceil \log_2(n+1) \rceil + c$$

für eine Konstante c. Die Länge von w_n ist $|w_n| = |0101| \cdot 4n^2 = 16n^2$. Also gilt $n = \sqrt{|w_n|}/4$ und wir erhalten eine obere Schranke von

$$K(w_n) \leq \left\lceil \log_2\left(1 + \sqrt{|w_n|}/4\right) \right\rceil + c \leq \log_2(|w_n|)/2 + c'$$

für die Kolmogorov-Komplexität von w_n für eine Konstante c'.

(b) Wir definieren die Folge von Wörtern $y_0, y_1, y_2, \ldots$ durch $y_i = 0^{3^{i+1}}$ für alle $i \in \mathbb{N}$. Offensichtlich gilt $y_0 < y_1 < y_2 < \ldots$ und wir haben $i + 1 = \log_3 |y_i|$ für alle $i \in \mathbb{N}$. Nun konstruieren wir für jedes y_i ein Pascal-Programm P_i, das y_i erzeugt:

```
begin
  k:= 3*3^i;
  for j:=1 to k do
    write(0);
end.
```

In diesem Programm wird zuerst der Wert $k = 3^{i+1}$ berechnet und dann werden in der `for`-Schleife k Nullen geschrieben. Der Teil des Maschinencodes von P_i, der von y_i abhängt, ist nur die Darstellung von i. Der Maschinencode für die Darstellung des restlichen Programms hat also nur eine konstante Länge, während die binäre Kodierung von i die Länge $\lceil \log_2(i+1) \rceil \leq 1 + \log_2(i+1)$ hat. Damit folgt

$$K(y_i) \leq \log_2(i+1) + c = \log_2 \log_3(|y_i|) + c$$

für eine Konstante c.

Lösung zu Aufgabe 5

Für $n < 8$ ist die Aussage in der Aufgabenstellung trivial, wir können also annehmen, dass $n \geq 8$ ist. Wir zeigen, dass weniger als 1% der Wörter in $\{0,1\}^n$ Kolmogorov-Komplexität $\leq n - 8$ haben. Es gibt

$$\sum_{j=1}^{n-8} 2^j = 2^{n-7} - 2 < 2^{n-7}$$

nichtleere binäre Wörter mit Länge höchstens $n - 8$. Also existieren höchstens 2^{n-7} unterschiedliche Pascal-Programme mit Maschinencodes der Länge höchstens $n-8$. Daraus folgt, dass auch höchstens 2^{n-7} viele Wörter existieren, die Kolmogorov-Komplexität $\leq n-8$ haben, da wir jedem solchen Wort w den Maschinencode eines Pascal-Programms zuordnen können, das w erzeugt und dessen Kodierung Länge $K(w)$ hat. Da $|\{0,1\}^n| = 2^n$, haben also höchstens $2^{n-7}/2^n = 2^{-7} < 1\%$ der Wörter in $\{0,1\}^n$ eine Kolmogorov-Komplexität von höchstens $n - 8$.

Lösung zu Aufgabe 6

Wir nehmen per Widerspruch an, $\mathbf{A}$ sei ein Pascal-Programm, welches einen Input $n \in \mathbb{N}$ entgegennimmt und ein Wort $\mathbf{A}(n) \in \{0,1\}^n$ ausgibt mit $K(\mathbf{A}(n)) > n - 8$. Wir können mithilfe von $\mathbf{A}$ nun für jedes $n \in \mathbb{N}$ folgendes Pascal-Programm konstruieren:

```
begin
  m := n;
  s := A(m);
  write(s);
end.
```

Dies ist also einfach das Programm $\mathbf{A}$ mit fixiertem Input n, d.h., dieses Programm schreibt das Wort $\mathbf{A}(n)$. Der einzige Teil des Maschinencodes, der von $\mathbf{A}(n)$ abhängt, ist die Darstellung von n in der zweiten Zeile. Die Darstellung des restlichen Maschinencodes hat eine konstante Länge c. Also gilt für alle Wörter $\mathbf{A}(n)$, $n \in \mathbb{N}$:

$$K(\mathbf{A}(n)) \leq \lceil \log_2(n+1) \rceil + c \leq \log_2(n) + c',$$

für eine Konstante c'. Andererseits gilt per Annahme, dass $K(\mathbf{A}(n)) > n - 8$ für alle $n \in \mathbb{N}$. Es folgt, für alle $n \in \mathbb{N}$,

$$n - 8 < \log_2(n) + c',$$

ein Widerspruch für genug grosse n.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Dennis Komm
Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer

Lösungsvorschläge – Blatt 3

Zürich, 11. Oktober 2024

Lösung zu Aufgabe 7

Wir beweisen die Aussage indirekt. Somit gebe es also eine unendlich grosse Anzahl von Primzahlen, die als zufällig angesehen werden können. Nach Definition bedeutet dies, dass es unendlich viele $k \in \mathbb{N} - \{0\}$ gibt, sodass

$$K(p_k) \geq \lceil \log_2(p_k + 1) \rceil - 1 \geq \log_2(p_k) - 1 \tag{1}$$

gilt, wobei p_k die k-te Primzahl ist. Ferner lässt sich die m-te Primzahl p_m für jedes $m \in \mathbb{N} - \{0\}$ mit einem Pascal-Programm C_m berechnen, das die binäre Kodierung von m enthält, alle natürlichen Zahlen in aufsteigender Reihenfolge darauf testet, ob sie prim sind (Sieb des Eratosthenes) und die m-te gefundene Primzahl ausgibt. Alle Bestandteile dieses Programms mit Ausnahme der Kodierung von m haben eine konstante Länge, also hat C_m die Länge $\lceil \log_2(m+1) \rceil + c \leq \log_2 m + c'$ für Konstanten c und $c' = c + 1$. Sei $n \in \mathbb{N}$ und sei $\text{Prim}(n)$ die Anzahl der Primzahlen kleiner gleich n. Nach dem Primzahlsatz gilt für alle $n > 67$, dass

$$\text{Prim}(n) < \frac{n}{\ln n - \frac{3}{2}}\,.$$

Da nach Definition wiederum $m = \text{Prim}(p_m)$ ist, erhalten wir

$$\text{Prim}(p_m) = m < \frac{p_m}{\ln(p_m) - \frac{3}{2}}$$

und schliesslich

$$K(p_m) \leq \log_2\left(\frac{p_m}{\ln(p_m) - \frac{3}{2}}\right) + c' = \log_2(p_m) - \log_2\left(\ln(p_m) - \frac{3}{2}\right) + c'\,. \tag{2}$$

Aus den beiden Schranken (1) und (2) für die Kolmogorov-Komplexität ergibt sich, dass

$$\log_2(p_l) - 1 \leq \log_2(p_l) - \log_2\left(\ln(p_l) - \frac{3}{2}\right) + c'$$

für unendlich viele $l \in \mathbb{N}$ gelten muss. Hieraus folgt wiederum, dass für diese l auch

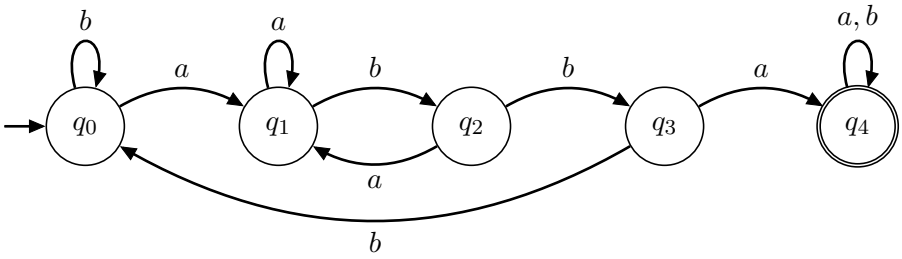
$$\log_2\left(\ln(p_l) - \frac{3}{2}\right) \leq c' + 1$$

sein muss, was unmöglich ist, da c' eine Konstante ist und $\log_2(\ln(p_l) - 3/2)$ mit p_l wächst. Somit ist unsere Annahme falsch und die zu beweisende Aussage folgt.

Lösung zu Aufgabe 8

(a) Der folgende endliche Automat akzeptiert die Sprache

$$L_1 = \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } abba\}.$$



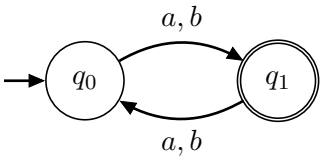
Es ergeben sich die folgenden Klassen für die Zustände:

$$\begin{aligned}
&\mathrm{Kl}[q_0] = \{wbbb \in \{a,b\}^* \mid w \text{ enthält nicht } abba\} \cup \{\lambda, b, bb\}\,,\\
&\mathrm{Kl}[q_1] = \{wa \in \{a,b\}^* \mid wa \text{ enthält nicht } abba\}\,,\\
&\mathrm{Kl}[q_2] = \mathrm{Kl}[q_1] \cdot \{b\}\,,\\
&\mathrm{Kl}[q_3] = \mathrm{Kl}[q_2] \cdot \{b\} = \{a,b\}^* - (\mathrm{Kl}[q_0] \cup \mathrm{Kl}[q_1] \cup \mathrm{Kl}[q_2] \cup L_1)\,,\\
&\mathrm{Kl}[q_4] = L_1.
\end{aligned}$$

(b) Wir bemerken, dass L_2 folgende alternative Charakterisierung hat

$$L_2 = \{w \in \{a,b\}^* \mid |w|_a \neq |w|_b \pmod 2\} = \{w \in \{a,b\}^* \mid |w| \equiv 1 \pmod 2\}.$$

Daher wird L_2 vom endlichen Automaten



erkannt. Dieser hat die folgenden Zustandsklassen

$$\begin{aligned}
&\mathrm{Kl}[q_0] = \{w \in \{a,b\}^* \mid |w| \equiv 0 \pmod 2\}\,,\\
&\mathrm{Kl}[q_1] = \{w \in \{a,b\}^* \mid |w| \equiv 1 \pmod 2\} = L_2.
\end{aligned}$$

Lösung zu Aufgabe 9

Wir bemerken zuerst, dass der Zustand q_4 ein Senkenzustand ist, d.h., landet der endliche Automat M beim Lesen eines Wortes einmal in q_4, bleibt er dort und akzeptiert die Eingabe nicht. Weiter bemerken wir, dass M nur auf Wörtern der Form a^k für $k \in \mathbb{N}$ im Zustand q_0 oder q_1 landet. Genauergesagt gilt

$$\begin{aligned}
&\mathrm{Kl}[q_0] = \{a^{2k} \mid k \in \mathbb{N}\}\,,\\
&\mathrm{Kl}[q_1] = \{a^{2k+1} \mid k \in \mathbb{N}\}.
\end{aligned}$$

Liest M in q_1 ein b, geht er in den Senkenzustand q_4. Liest M in q_0 ein b, geht er nach q_2 und 'misst' dann durch Wechseln zwischen q_2 und q_3 die Parität der Anzahl b's im Suffix des gelesenen Wortes. Sobald M in q_2 oder q_3 ein a liest, geht er in den Senkenzustand. Wir haben also

$$\begin{aligned}
\mathrm{Kl}[q_2] &= \mathrm{Kl}[q_0] \cdot \{b^{2l+1} \mid l \in \mathbb{N}\} = \{a^{2k}b^{2l+1} \mid k, l \in \mathbb{N}\}, \\
\mathrm{Kl}[q_3] &= \mathrm{Kl}[q_0] \cdot \{b^{2l+2} \mid l \in \mathbb{N}\} = \{a^{2k}b^{2l+2} \mid k, l \in \mathbb{N}\}, \\
\mathrm{Kl}[q_4] &= \{a, b\}^* - (\mathrm{Kl}[q_0] \cup \mathrm{Kl}[q_1] \cup \mathrm{Kl}[q_2] \cup \mathrm{Kl}[q_3]).
\end{aligned}$$

Also gilt

$$L(M) = \mathrm{Kl}[q_0] \cup \mathrm{Kl}[q_3] = \{a^{2k}b^{2l} \mid k, l \in \mathbb{N}\}.$$

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Dennis Komm
Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer

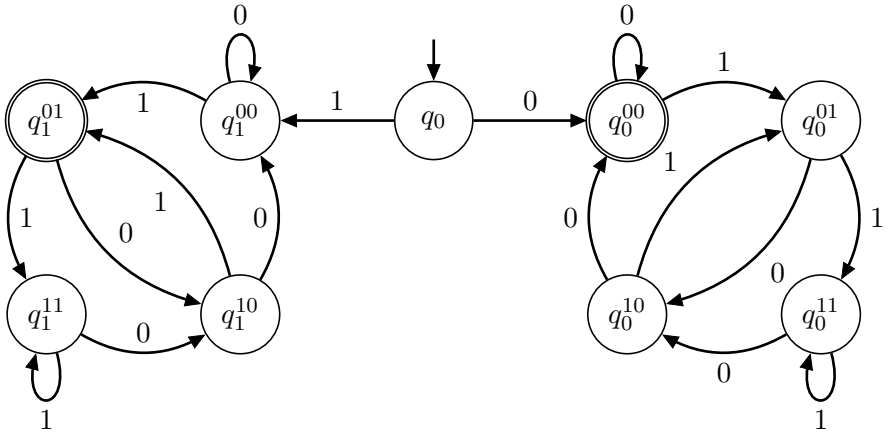
Lösungsvorschläge – Blatt 4

Zürich, 18. Oktober 2024

Lösung zu Aufgabe 10

Der folgende endliche Automat erkennt die Sprache

$$L = \{xw \in \Sigma^*_{\text{bool}} \mid x \in \Sigma_{\text{bool}} \text{ und } \text{Nummer}(w) \equiv x \pmod 4\}.$$



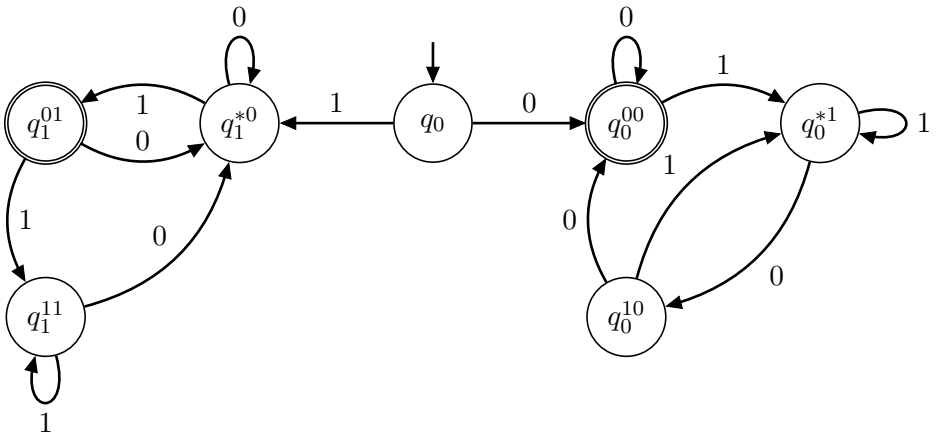
Der Automat liest den Wert $x \in \Sigma_{\text{bool}}$ und geht entsprechend in einen von zwei Teilautomaten – nach rechts, falls $x = 0$ und nach links, falls $x = 1$. Da für ein Wort $w \in \Sigma^*_{\text{bool}}$ der Wert $\text{Nummer}(w) \pmod 4$ nur von den letzten beiden Symbolen von w abhängt, speichern die beiden Teilautomaten jeweils die letzten beiden Symbole des gelesenen Präfixes in den Zuständen und akzeptieren entsprechend. Präziser formuliert induziert der Automat folgende Zustandsklassen:

$$\begin{aligned}
\text{Kl}[q_0] &= \{\lambda\}\,, \\
\forall x \in \Sigma_{\text{bool}} : \text{Kl}[q_x^{00}] &= \{xv00 \mid v \in \Sigma^*_{\text{bool}}\} \cup \{x, x0\}\,, \\
\forall x \in \Sigma_{\text{bool}} : \text{Kl}[q_x^{01}] &= \{xv01 \mid v \in \Sigma^*_{\text{bool}}\} \cup \{x1\}\,, \\
\forall x \in \Sigma_{\text{bool}}\ \forall s \in \{10, 11\} : \text{Kl}[q_x^{s}] &= \{xvs \in \Sigma^*_{\text{bool}} \mid v \in \Sigma^*_{\text{bool}}\}\,.
\end{aligned}$$

Alternative Lösung: Wir bemerken, dass wir im rechten Teilautomaten oben die Zustände q_0^{01} und q_0^{11} zu einem Zustand q_0^{*1} zusammenfassen können. Dies sieht man folgendermassen:

- Beim Lesen einer 0 geht der Automat sowohl von q_0^{01} wie auch von q_0^{11} nach q_0^{10}.
- Beim Lesen einer 1 bleibt der Automat, falls er in einem der Zustände q_0^{01} oder q_0^{11} ist, in einem dieser beiden Zustände.
- Unter den Zuständen q_0^{01} und q_0^{11} ist nicht einer akzeptierend und der andere nicht.

Dass im rechten Teilautomaten nur drei Zustände nötig sind, ist auch intuitiv klar: Falls das bisher gelesene Wort auf 1 endet, muss der Automat das zweitletzt gelesene Symbol nicht kennen, um entscheiden zu können, ob die gesamte Eingabe Suffix 00 hat.
Ganz analog bemerkt man, dass im linken Teilautomaten oben die Zustände q_1^{00} und q_1^{10} zu q_1^{*0} zusammengefasst werden können. Dies ergibt also den Automaten



wobei dann entsprechend gilt

$$\mathrm{Kl}[q_0^{*1}] = \mathrm{Kl}[q_0^{01}] \cup \mathrm{Kl}[q_0^{11}],$$

sowie

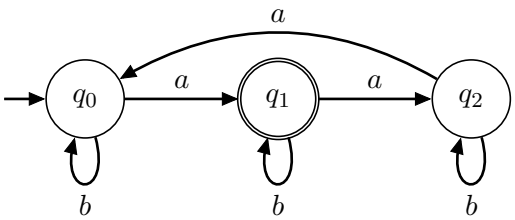
$$\mathrm{Kl}[q_1^{*0}] = \mathrm{Kl}[q_1^{00}] \cup \mathrm{Kl}[q_1^{10}].$$

Lösung zu Aufgabe 11

Die in der Aufgabenstellung gegebene Sprache L lässt sich schreiben als $L = L_1 \cup L_2$ mit

$$L_1 = \{w \in \{a,b\}^* \mid |w|_a \equiv 1 \pmod 3\},$$
$$L_2 = \{w \in \{a,b\}^* \mid |w| = 2\}.$$

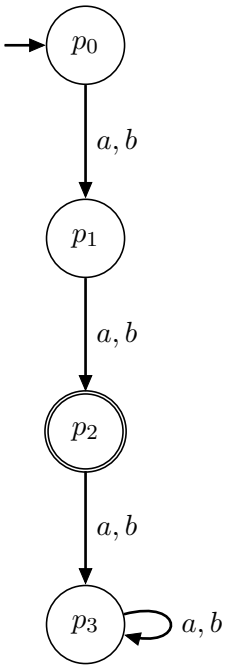
Für die Sprache L_1 lässt sich der folgende Automat A_1 konstruieren:



Es gilt

$$\forall i \in \{0,1,2\} : \mathrm{Kl}[q_i] = \{w \in \{a,b\}^* \mid |w|_a \equiv i \pmod 3\}$$

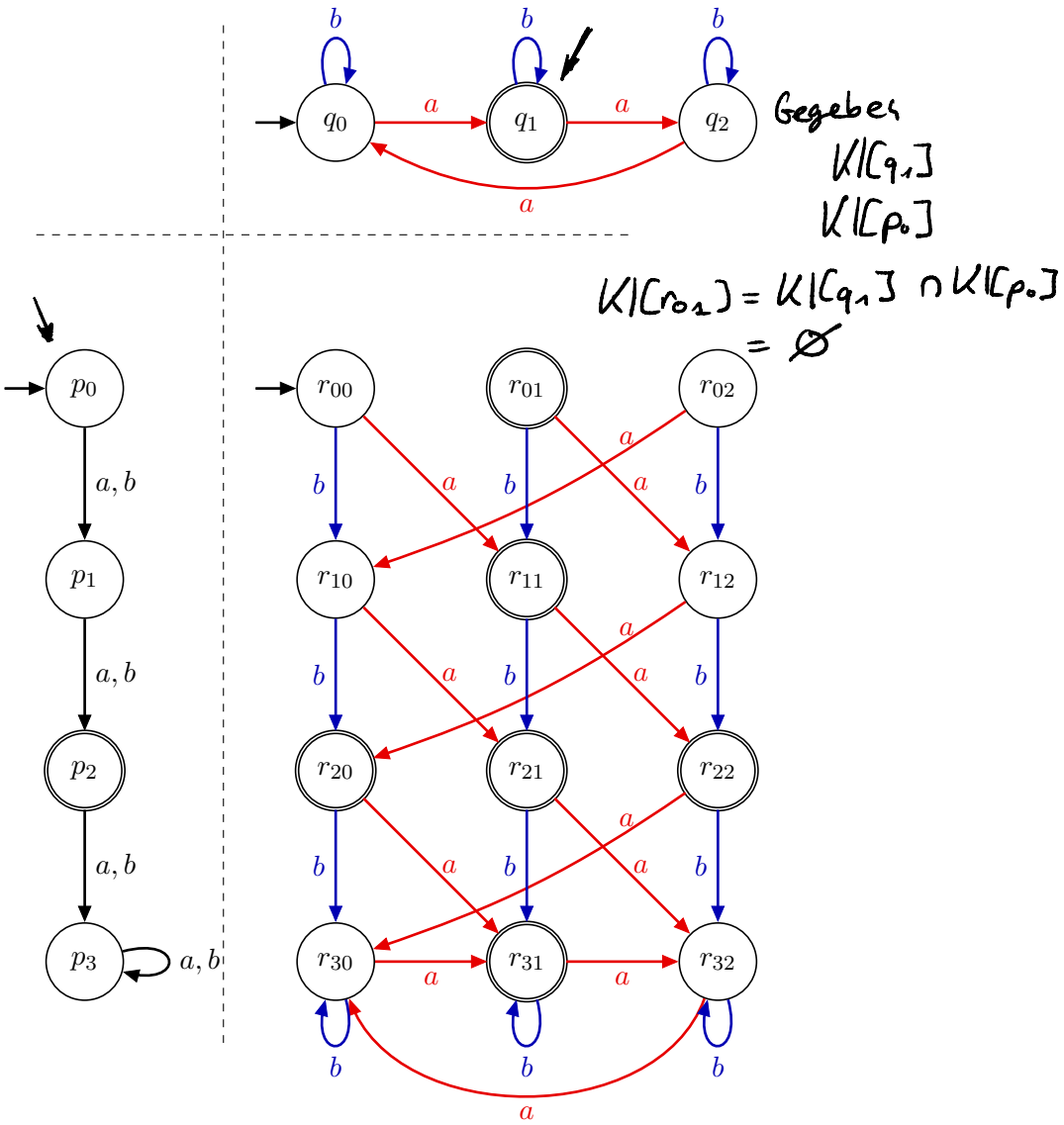
Für die Sprache L_2 lässt sich der folgende Automat A_2 konstruieren:



Es gilt

$$\begin{aligned} \forall i \in \{0,1,2\} : \mathrm{Kl}[p_i] &= \{a,b\}^i\,, \\ \mathrm{Kl}[p_3] &= \{a,b\}^* - (\mathrm{Kl}[p_0] \cup \mathrm{Kl}[p_1] \cup \mathrm{Kl}[p_2])\,. \end{aligned}$$

Mit der Methode des modularen Entwurfs kann man nun aus A_1 und A_2 den folgenden Produktautomaten A konstruieren, der die Sprache $L = L_1 \cup L_2$ akzeptiert. Zur einfacheren Darstellung verwenden wir die Notation $r_{x,y} = \langle p_x, q_y \rangle$. Weil dies ein Produktautomat für die Vereinigung von zwei Sprachen ist, sind alle Zustände akzeptierend, die einen akzeptierenden Zustand aus einem der beiden Teilautomaten enthalten.



Die Zustände r_{01}, r_{02} und r_{12} sind vom Startzustand aus nicht erreichbar, können also auch weggelassen werden.

Lösung zu Aufgabe 12

(a) Wir zeigen mit Hilfe von Lemma 3.3 aus dem Buch, dass die Sprache

$$L_1 = \{w \in \{a, b, c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

nicht regulär ist. Angenommen, L_1 sei regulär. Dann gibt es einen Automaten $A = (Q, \{a, b, c\}, \delta, q_0, F)$ mit $L(A) = L_1$. Sei $m = |Q|$. Wir betrachten die Wörter

$$\lambda, abc, (abc)^2, \ldots, (abc)^m\,.$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A Zustände hat, gibt es $i, j \in \{0, \ldots, m\}$ mit $i \neq j$, so dass

$$\hat{\delta}(q_0, (abc)^i) = \hat{\delta}(q_0, (abc)^j)\,.$$

Nach Lemma 3.3 gilt nun für alle $z \in \{a, b, c\}^*$, dass

$$(abc)^i z \in L_1 \iff (abc)^j z \in L_1.$$

Die Wahl von $z = (bac)^i$ führt aber zum Widerspruch, weil

$$(abc)^i z = (abc)^i (bac)^i \in L_1$$

und $(abc)^j z = (abc)^j (bac)^i \notin L_1$ gilt. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Wir verwenden das Pumping-Lemma, um zu zeigen, dass die Sprache

$$L_2 = \{wabw^{\mathsf{R}} \mid w \in \{a, b\}^*\}$$

nicht regulär ist. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{a, b\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

1. $|yx| \leq n_0$,
2. $|x| \geq 1$ und
3. entweder $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = a^{n_0} aba^{n_0}$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (i) gilt $|yx| \leq n_0$, also ist $y = a^l$ und $x = a^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$, also insbesondere $m \leq n_0$. Wegen (ii) gilt weiter $m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^k z \mid k \in \mathbb{N}\} = \{a^{n_0+(k-1)m} aba^{n_0} \mid k \in \mathbb{N}\} \subseteq L_2\,.$$

Dies ist aber ein Widerspruch, da $yx^2 z = a^{n_0+m} aba^{n_0} \notin L_2$. Also ist die Annahme falsch und L_2 ist nicht regulär.

ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Dennis Komm
Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer

Lösungsvorschläge – Blatt 5

Zürich, 25. Oktober 2024

Lösung zu Aufgabe 13

(a) Wir beweisen mit Hilfe der Methode der Kolmogorov-Komplexität, dass die Sprache $L_1 = \{ww \mid w \in \{0,1\}^*\}$ nicht regulär ist.

Angenommen, $L = L_1$ sei regulär. Für jedes $m \in \mathbb{N}$ ist 0^m1 das erste Wort in der Sprache

$$L_{0^m1} = \{y \mid 0^m1y \in L\}.$$

Nach Satz 3.1 aus dem Buch existiert eine Konstante c, unabhängig von m, so dass

$$K(0^m1) \leq \lceil \log_2(1+1) \rceil + c = 1 + c\,.$$

Da es nur endlich viele Programme der konstanten Länge $1 + c$ gibt, aber unendlich viele Wörter der Form 0^m1, ist dies ein Widerspruch. Also ist die Annahme falsch und L_1 ist nicht regulär.

(b) Sei $L_2 = \{0^p \mid p \in \mathbb{N} \text{ ist eine Primzahl}\}$. Wir zeigen die Nichtregularität von L_2 zunächst mit Hilfe des Pumping-Lemmas.

Beweis mit Hilfe des Pumping-Lemmas. Angenommen, L_2 sei regulär. Dann existiert nach dem Pumping-Lemma (Lemma 3.4) eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \{0\}^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, so dass

(i) $|yx| \leq n_0$,
(ii) $|x| \geq 1$ und
(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_2$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$.

Wir wählen das Wort $w = 0^p$ für eine Primzahl $p \geq n_0$. Offenbar gilt $|w| \geq n_0$. Also muss es eine Zerlegung $w = yxz$ von w geben, die die Bedingungen (i), (ii) und (iii) erfüllt. Wegen (ii) gilt weiter $|x| =: m > 0$. Da $w \in L_2$, gilt nach (iii), dass auch

$$\{yx^kz \mid k \in \mathbb{N}\} = \{0^{p+(k-1)\cdot m} \mid k \in \mathbb{N}\} \subseteq L_2\,.$$

Wir wählen $k = p + 1$. Damit ist $yx^kz = 0^{p+(k-1)m} = 0^{p+pm} = 0^{p\cdot(m+1)}$. Damit erhalten wir einen Widerspruch, weil $p \cdot (m+1)$ keine Primzahl ist; man beachte, dass $m > 0$. Also ist die Annahme falsch und L_2 ist nicht regulär.

Eine Konsequenz aus dem Primzahlsatz. Wir könnten die Nichtregularität von L_2 auch mit Hilfe der Kolmogorov-Komplexität oder mit Hilfe von Lemma 3.3 zeigen. Dafür benötigen wir allerdings den Primzahlsatz (Satz 2.3 im Buch), den wir in dieser Vorlesung nicht bewiesen haben. Aus dem Primzahlsatz folgt, dass der Abstand zwischen zwei aufeinanderfolgenden Primzahlen beliebig gross werden kann: Wäre dieser Abstand durch eine Konstante $k \in \mathbb{N}$ nach oben beschränkt, dann gäbe es mindestens n/k Primzahlen unter den ersten n natürlichen Zahlen im Widerspruch zum Primzahlsatz, der besagt, dass es ungefähr $n/\ln n$ solche Primzahlen gibt. Wir verwenden diese Beobachtung nun, um mit Hilfe der Kolmogorov-Komplexität zu zeigen, dass L_2 nicht regulär ist.

Beweis mit Hilfe der Methode der Kolmogorovkomplexität. Angenommen, $L = L_2$ sei regulär. Sei p_m die m-te Primzahl. Dann ist $0^{p_{m+1}-p_m-1}$ das erste Wort in der Sprache

$$L_{0^{(p_m)+1}} = \{y \mid 0^{(p_m)+1}y \in L\}.$$

Nach Satz 3.1 aus dem Buch existiert eine Konstante c, unabhängig von m, so dass

$$K(0^{p_{m+1}-p_m-1}) \leq \lceil \log_2(1+1) \rceil + c = 1 + c.$$

Da es nur endlich viele Programme der konstanten Länge höchstens $1 + c$ gibt, aber nach der obigen Folgerung aus dem Primzahlsatz unendlich viele Wörter der Form $0^{p_{m+1}-p_m-1}$, ist dies ein Widerspruch. Also ist die Annahme falsch und L_2 ist nicht regulär. Wir können diese Folgerung aus dem Primzahlsatz ebenfalls verwenden, um mit Hilfe von Lemma 3.3 zu zeigen, dass L_2 nicht regulär ist.

Beweis mit Hilfe von Lemma 3.3. Angenommen, L_2 sei regulär. Dann gibt es einen Automaten $A_2 = (Q, \{0\}, \delta, q_0, F)$ mit $L(A_2) = L_2$. Sei $m = |Q|$. Wir wählen $m+1$ verschiedene Primzahlen $p_{l_0}, \ldots, p_{l_m}$ so, dass die Differenzen $p_{l_j+1} - p_{l_j}$, also die Abstände zur nächstgrösseren Primzahl, paarweise verschieden und – ohne Beschränkung der Allgemeinheit – monoton aufsteigend sind. Solche Primzahlen existieren nach der oben beschriebenen Folgerung aus dem Primzahlsatz. Dann betrachten wir die Wörter

$$0^{p_{l_0}}, 0^{p_{l_1}}, \ldots, 0^{p_{l_m}}.$$

Weil dies $m + 1$ Wörter sind, also mehr Wörter als A_2 Zustände hat, gibt es $i, j \in \{0, \ldots, m\}$ mit $i < j$, so dass

$$\hat{\delta}_{A_2}(q_0, 0^{p_{l_i}}) = \hat{\delta}_{A_2}(q_0, 0^{p_{l_j}}).$$

Nach Lemma 3.3 gilt nun für alle $z \in \{0\}^*$, dass

$$0^{p_{l_i}} z \in L_2 \iff 0^{p_{l_j}} z \in L_2.$$

Die Wahl von $z = 0^{p_{l_i+1}-p_{l_i}}$ führt aber zum Widerspruch, weil

$$0^{p_{l_i}} z = 0^{p_{l_i+1}} \in L_2$$

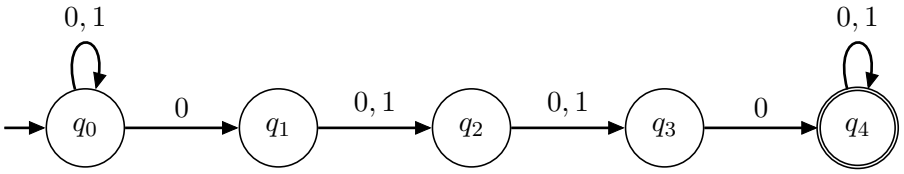
und $0^{p_{l_j}} z = 0^{p_{l_j}+(p_{l_i+1}-p_{l_i})} \notin L_2$ gemäss der Voraussetzung über die gewählten Primzahlen gilt. Also ist die Annahme falsch und L_2 ist nicht regulär.

Lösung zu Aufgabe 14

(a) Der folgende NEA akzeptiert die Sprache

$$L_1 = \{w \in \{0,1\}^* \mid w = x0y0z \text{ mit } x, z \in \{0,1\}^* \text{ und } y \in \{0,1\}^2\}$$

.



Dieser NEA entscheidet nichtdeterministisch, wann das gesuchte Muster in der Eingabe beginnt und wechselt dann vom Zustand q_0 in den Zustand q_1. Wenn diese nichtdeterministische Entscheidung richtig war, dann kann der Automat nun das Muster lesen und endet im akzeptierenden Zustand q_4. Wenn die Eingabe das gesuchte Muster nicht enthält, dann hat der NEA keine Möglichkeit, sie zu akzeptieren.

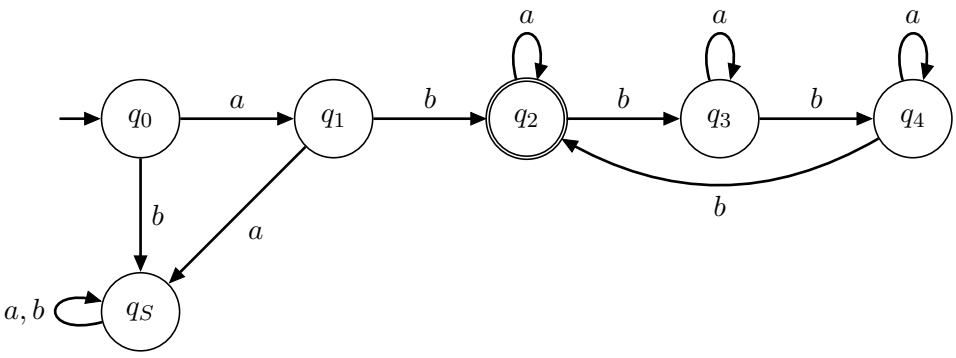
(b) Um zu zeigen, dass jeder deterministische endliche Automat A, der die Sprache

$$L_2 = \{w \in \{a,b\}^* \mid |w|_b \equiv 1 \pmod 3 \text{ und } w \text{ beginnt mit } ab\}$$

akzeptiert, mindestens 6 Zustände benötigt, bestimmen wir 6 Wörter $w_1, \ldots, w_6$ und zeigen, dass diese den Automaten in 6 paarweise verschiedene Zustände führen müssen. Falls es einen EA gibt, der mit dem Lesen von zwei verschiedenen Wörtern w_i und w_j in demselben Zustand endet, dann gilt nach Lemma 3.3 aus dem Buch für jedes $z \in \{a,b\}^*$, dass dann auch $w_i z$ und $w_j z$ den Automaten in denselben Zustand führen. Wir wollen zeigen, dass dies für einen Automaten mit weniger als 6 Zuständen nicht möglich ist, indem wir für je zwei verschiedene Wörter w_i und w_j ein Wort $z_{i,j}$ angeben, so dass

$$w_i z_{i,j} \in L(A) \iff w_j z_{i,j} \notin L(A)\,. \tag{1}$$

Eine Strategie, um geeignete Wörter $w_1, \ldots, w_6$ zu bestimmen, besteht darin, zunächst einen EA A mit 6 Zuständen für die Sprache L_2 zu entwerfen, und dann für jeden Zustand q das kürzeste Wort aus der Klasse Kl$[q]$ zu wählen. Der folgende Automat A mit 6 Zuständen akzeptiert die Sprache L_2.



Die kürzesten Wörter in den Zustandsklassen von A sind $w_1 = \lambda$, $w_2 = a$, $w_3 = b$, $w_4 = ab$, $w_5 = abb$ und $w_6 = abbb$. Die folgende Tabelle gibt für alle Paare (w_i, w_j) mit $i < j$ jeweils ein Wort $z_{i,j}$ an.

$z_{i,j}$	$w_2 = a$	$w_3 = b$	$w_4 = ab$	$w_5 = abb$	$w_6 = abbb$
$w_1 = \lambda$	ab	ab	ab	ab	b
$w_2 = a$	$-$	b	λ	bb	ab
$w_3 = b$	$-$	$-$	λ	bb	b
$w_4 = ab$	$-$	$-$	$-$	λ	λ
$w_5 = abb$	$-$	$-$	$-$	$-$	b

Es ist einfach zu sehen, dass diese Wörter die Bedingung (1) erfüllen, zum Beispiel ist $w_2 z_{2,6} = aab \notin L(A)$, aber $w_6 z_{2,6} = abbbab \in L(A)$.