Prof. Javier Esparza
Philipp Czerner, Kevin Kappelmann, Valentin Krasotin
Technische Universität München
Lehrstuhl für Theoretische Informatik

Einführung in die Theoretische Informatik

Sommersemester 2024 – Hausaufgabenblatt 4

- Die Hausaufgaben werden in folgender Reihenfolge korrigiert: H4.3, H4.4.
- Die Knobelaufgabe bitte separat auf Moodle abgeben. Sie wird korrigiert.

AT-Aufgabe H4.1. (*sic infit*) unkorrigiert (1+1+1 Punkte)

Bearbeiten Sie diese Aufgabe in Automata Tutor. Bestimmen Sie einen regulären Ausdruck für die Residualsprache $L(r)^a$ für jeden der folgenden REs r.

(a) $ab(aa \mid bb)^* \mid ba$ (b) $(abb \mid baa)^*$ (c) $(ab \mid a)^* ab$

Lösungsskizze.

(a) $b(aa \mid bb)^*$

(b) $bb(abb \mid baa)^*$

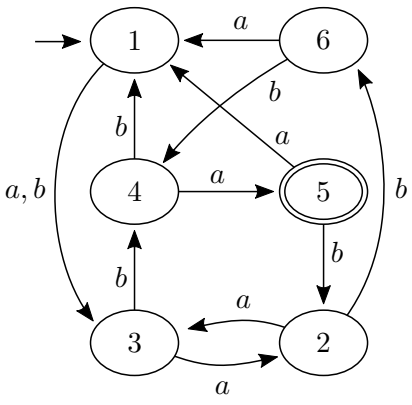
(c) $(b \mid \epsilon)(ab \mid a)^* ab \mid b$

AP-Aufgabe H4.2. (*in nuce*) unkorrigiert (2+2 Punkte)

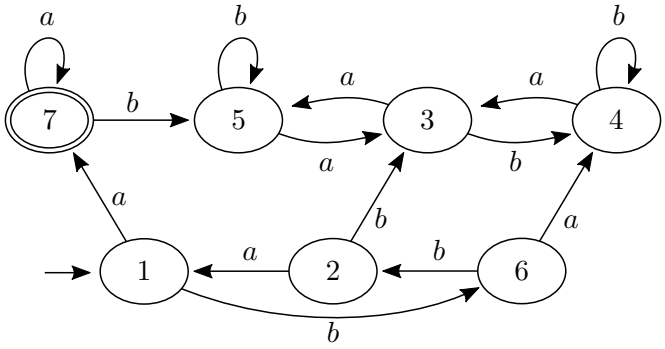
Doras Kindergarten hat beschlossen, namenstreu zu werden, und die minderjährigen Mitglieder dürfen sich nunmehr botanisch betätigen. Dora kümmert sich gerade begeistert um eine Amaryllis. Leider ist diese zu groß für den dafür vorgesehenen Topf. Können Sie Doras fabelhafte Amaryllis verkleinern?

Update: Bearbeiten Sie diese Aufgabe auf apotheosis: (a), (b).

Minimieren Sie folgende DFAs unter Verwendung des erweiterten Minimierungsalgorithmus aus Ü4.5(b).



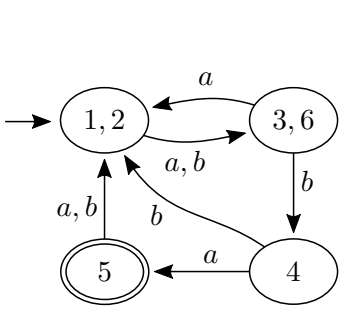
(a)



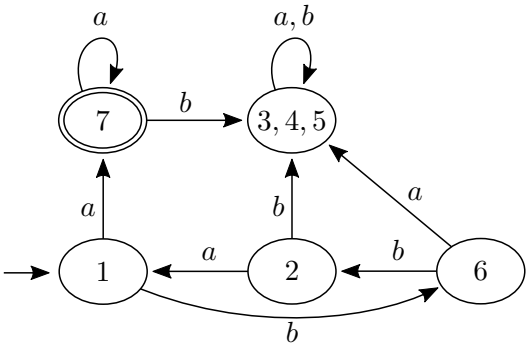
(b)

Anmerkung: Zusätzlich gibt es diese Aufgabe auch auf Automata Tutor. Allerdings überprüft AT nur den resultierenden DFA.

Lösungsskizze.



(a)



(b)

Aufgabe H4.3. (*nil satis nisi optimum*) 2+4 Punkte

Dora hat in ihrer Kindergartenvorlesung gelernt, wie die Produkt-Konstruktion funktioniert. Jetzt ist ihr aber aufgefallen, dass der entstandene Produktautomat nicht minimal sein muss! Entrüstet möchte sie zu ihrem Kindergartenprofessor stampfen, ihn mit Sand bewerfen, und sich darüber beschweren, dass er ihr ein offensichtlich minderwertiges Verfahren beigebracht hat. Können Sie Dora beschwichtigen, indem Sie demonstrieren, dass das Verfahren, zumindest manchmal ein bestmögliches Ergebnis liefert?

Sei $\Sigma := \{a, b\}$, $A_k := \{w \in \Sigma^* : |w|_a \geq k\}$ die Sprache aller Wörter, die mindestens k-mal a enthalten, und $B_k := \{w \in \Sigma^* : |w|_b \geq k\}$ entsprechend für b.

(a) Seien $k, n \in \mathbb{N}$ und $L := A_k \cap B_n$. Bestimmen Sie alle Residualsprachen von L. Begründen Sie Ihre Antwort.

Hinweis: L hat genau $(k+1)(n+1)$ Residualsprachen.

(b) Beweisen Sie, dass Ihre Residualsprachen aus (a) paarweise unterschiedlich sind.

Dora versteht nun, worauf Sie hinauswollen: Die Anzahl der Zustände im minimalen DFA ist gleich der Anzahl der Residualsprachen. Der minimale DFA für A_k hat $k+1$ Zustände (da $A_k = A_k \cap B_0$) und der minimale DFA für B_n hat $n+1$ Zustände. Durch Anwenden der Produktkonstruktion auf diese beiden DFAs erhält man einen DFA mit höchstens $(k+1)(n+1)$ Zuständen – der muss aber minimal sein, da $A_k \cap B_n$ genau so viele Residualsprachen besitzt.

Lösungsskizze. **(a)** Wenn wir das Präfix $w \in \Sigma^*$ vorgegeben haben, dann haben wir bereits $|w|_a$ mal a gelesen. Wir müssen also mindestens $k - |w|_a$ weitere a einlesen, oder 0, falls $k \leq |w|_a$. Äquivalent kann man sagen, dass das Restwort in $A_{k-|w|_a}$ enthalten ist. Dies gilt analog für b, das Restwort muss also in $R := A_{k-|w|_a} \cap B_{n-|w|_b}$ sein.

Umgekehrt gilt $wv \in L$ für jedes Restwort $v \in R$, da $|wv|_a = |w|_a + |v|_a \geq k$ und analog $|wv|_b \geq n$. Das Residuum von L bezüglich w ist also genau R. Insgesamt erhalten wir also die Residuen $(A_i \cap B_j)_{0 \leq i \leq k, 0 \leq j \leq n}$.
(b) Seien $R = A_i \cap B_j$ und $R' = A_{i'} \cap B_{j'}$ zwei beliebige Residuen, mit $(i, j) \neq (i', j')$. Wir machen nun eine Fallunterscheidung.

1. $i < i'$: Das Wort $w := a^i b^{j+j'}$ ist in B_j und $B_{j'}$ enthalten, da $j + j' \geq j$ und $j + j' \geq j'$. Allerdings ist w in A_i enthalten (und somit R) aber nicht in $A_{i'}$ (und

somit nicht in R'). Wir haben ein Wort gefunden, dass in R ist und in R' nicht, damit sind die Sprachen unterschiedlich.

2. $j < j'$: Dieser Fall funktioniert analog, mit $w := a^{i+i'}b^j$.
3. $i > i'$ oder $j > j'$: OBdA können wir R und R' vertauschen, und befinden uns dann in einem der ersten beiden Fälle.

Die Fallunterscheidung ist vollständig; nur $i = i' \wedge j = j'$ wird nicht abgedeckt, aber wir haben $(i, j) \neq (i', j')$ vorausgesetzt.

(Quiz)Aufgabe H4.4. (*vero nihil verius*) 5 Punkte

Eigentlich sollte der kleine Theodor seine Hausaufgaben machen. Er würde aber viel lieber seiner Lieblingspfütze beim Trocknen zuschauen, und hat deshalb die Hausaufgaben von seinem Hund Trodo machen lassen. Nun ist sich Theo aber nicht ganz sicher, ob Trodo auch alles richtig gemacht hat.

Bestimmen Sie für folgende Aussagen, ob sie wahr oder falsch sind. Geben sie eine kurze Begründung für wahre Aussagen an und ein Gegenbeispiel für falsche.

Sei $\Sigma \neq \emptyset$ ein Alphabet. Für alle regulären Sprachen $L_1, L_2 \subseteq \Sigma^*$, und nicht-regulären Sprachen $L_3, L_4 \subseteq \Sigma^*$ gilt:

(a) $(L_1 \cap L_2)^2$ ist regulär.

(b) $L_3 \cup L_4$ ist nicht regulär.

(c) L_4 erfüllt die Eigenschaft des Pumping-Lemmas nicht.

(d) Jede Teilmenge $L \subseteq L_1$ ist regulär.

(e) $(L_3)^*$ ist regulär.

Lösungsskizze. Anmerkung: Diese Aufgabe zielt darauf ab, die Abschlusseigenschaften regulärer Sprachen zu wiederholen. Durch Zitat des entsprechenden Satzes sind sie daher in einer Zeile gelöst. In dieser Lösung geben wir auch die Begründungen/Konstruktionen an, woher die Abschlusseigenschaften kommen.

(a) Wahr. Da L_1 und L_2 regulär sind, gibt es DFAs N_1 und N_2 mit $L(N_1) = L_1$ und $L(N_2) = L_2$. Die Produktkonstruktion liefert einen Automaten $N_1 \times N_2$ für $L_1 \cap L_2$, somit ist $L_1 \cap L_2$ regulär. Damit gibt es einen regulären Ausdruck r mit $L(r) = L_1 \cap L_2$. Dann gilt $L(rr) = (L_1 \cap L_2)^2$, somit ist die geforderte Sprache regulär.

(b) Falsch. Sei L_3 eine beliebige nicht-reguläre Sprache. Wähle $L_4 = \bar{L}_3$. Wir behaupten L_4 ist nicht-regulär. Angenommen L_4 wäre regulär. Dann gibt es einen DFA für L_4 und das Komplement dieses DFAs berechnet daher die Sprache L_3, die nicht-regulär ist, Widerspruch. Also ist L_4 nicht-regulär. Aber $L_3 \cup L_4 = \Sigma^*$ ist regulär.

(c) Falsch. In der Vorlesung wurde erwähnt, dass auch nicht-reguläre Sprachen das Pumping-Lemma erfüllen können. In Aufgabe H3.4 wurde außerdem ein konkretes Beispiel angegeben.

(d) Falsch. Gegenbeispiel: Wähle $\Sigma = \{a, b\}$, $L_1 := \Sigma^*$ und wähle für L eine beliebige nicht-reguläre Sprache, zum Beispiel $\{a^n b^n : n \in \mathbb{N}\}$. Dann ist L_1 regulär und $L \subseteq L_1$, aber L ist nicht regulär.

(e) Falsch. Gegenbeispiel: Wähle $\Sigma = \{a, b, c\}$, $L_3 := \{a^n b^n c : n \in \mathbb{N}\}$. Wenn $(L_3)^*$ regulär wäre, dann wäre nach Abschlusseigenschaften auch $(L_3)^* \cap L(a^*b^*c) = L_3$ regulär, ein Widerspruch.

Anmerkung: Bei der letzten Teilaufgabe gibt es gute und schlechte Wahlen von L_3. Die Sprache $L = \{w = w^R : w \in \Sigma^*\}$ der Palindrome zu wählen, würde zu $L^* = \Sigma^*$ führen, da jedes Zeichen einzeln ein Palindrom ist und daher jedes Wort trivial eine Konkatenation von Palindromen. Das Problem ist, dass Palindrome mit allen möglichen Buchstaben anfangen und aufhören können. Wenn wir also nur ein Wort wie *aabbbaaabb* haben, gibt es viele Möglichkeiten dies in Teil-Palindrome zu zerlegen. Wir könnten z.B. jedes Zeichen für sich lesen, oder $a(abbba)(aa)bb$ als Teil-Palindrome nehmen. Unsere Wahl von L_3 umgeht dies: Das Wortende ist in L_3 mit einem c markiert, sodass ein Wort aus L_3^* sich leicht in seine Teilwörter zerlegen lässt.

Quizaufgabe H4.5. (*divide et impera*) unkorrigiert (1+1+1 Punkte)

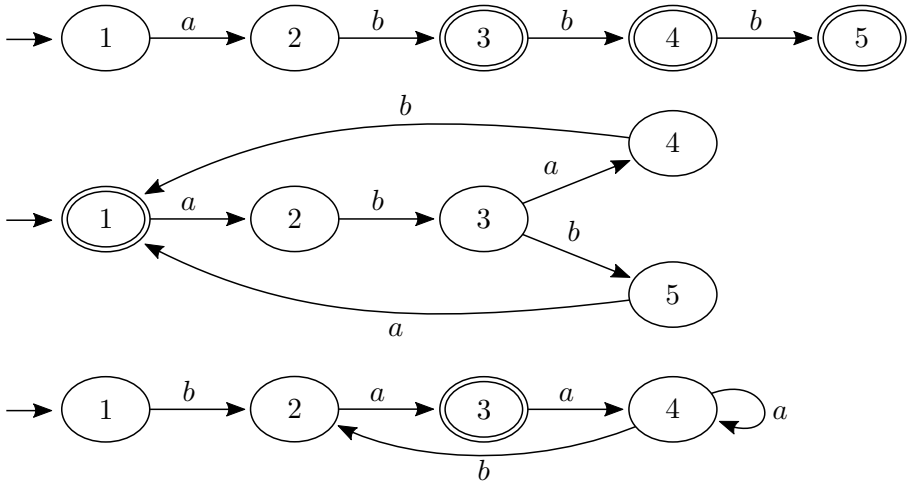
Sei $\Sigma := \{a, b\}$. Bestimmen Sie die Anzahl der Zustände des minimalen DFAs für die folgenden Sprachen.

(a) $L(ab \,|\, abb \,|\, abbb)$

(b) $L((abab \,|\, abba)^*)$

(c) $L(b(aaa^*b)^*a)$

Lösungsskizze.



Nicht abgebildete Transitionen führen zu einem impliziten Fangzustand, wir haben also (a) 6, (b) 6, und (c) 5 Zustände.

Knobelaufgabe H4.6. (*pauca sed bona*)

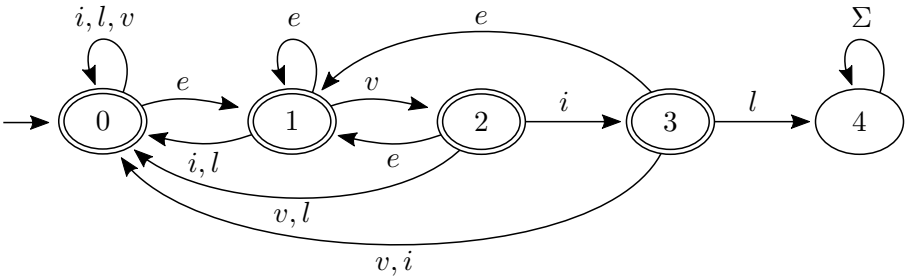
Der Superschurke Dr. Evilsparza hat jüngst ein Pharmaunternehmen (Evillest Evilness *Inc.*) gegründet, und behauptet nun, einen Impfstoff gegen schlechtes Wetter entwickelt zu haben. Die Öffentlichkeit ist begeistert – Sie trauen dem angeblich reformierten Sünder allerdings nicht und versuchen, in der DNA-Sequenz nach Hinweisen zu suchen.

Bekanntermaßen werden in der DNA vier verschiedene Basen kodiert: Evalcyclohexandifluorit, Isophoronditrigocyanat, Vinylcyclohexenquadroxid, und Levomethagameorphan. Diese werden mit ihrem Initialbuchstaben abgekürzt, sodass letztendlich ein Wort über dem Alphabet $\Sigma := \{e, i, l, v\}$ entsteht.

Ihnen ist bereits aufgefallen, dass die Sequenz ein merkwürdiges Muster aufweist: Auf die Zeichenfolge *evi* folgt nie ein l. Dr. Evilsparza behauptet, dass das reiner Zufall ist, aber Sie sind nicht überzeugt. Berechnen Sie die Wahrscheinlichkeit, dass ein (gleichmäßig) zufälliges Wort in Σ^{2024} obige Eigenschaft erfüllt.

Hinweise: Sie dürfen einen Computer verwenden. Es genügt, die Wahrscheinlichkeit auf 10 signifikante Nachkommastellen genau auszurechnen. Ein zufälliges Wort in Σ^{703} erfüllt die Eigenschaft mit Wahrscheinlichkeit ca. 0.06250879569.

Lösungsskizze. Wir konstruieren zuerst einen DFA $M = (Q, \Sigma, \delta, q_0, F)$ für die entsprechende Sprache:



Sei $f_n : Q \to [0, 1]$ die Wahrscheinlichkeit, dass ein Zustand ein zufälliges Wort der Länge n akzeptiert, für $n \in \mathbb{N}$. Es gilt also $f_n(q) := 4^{-n} \cdot |L_M(q) \cap \Sigma^n|$, für $q \in Q$. Wir sind also an $f_{2024}(q_0)$ interessiert.

Es ist leicht f_0 zu bestimmen: $f_0(q) = 1$ für $q \in F$, und $f_q(q) = 0$ sonst. Für $n \geq 0$ gilt

$$4^{n+1} f_{n+1}(q) = |L_M(q) \cap \Sigma^{n+1}| = \sum_{x \in \Sigma} |L_M(q) \cap \{x\}\Sigma^n|$$

Hier haben wir verwendet, dass $(\{x\}\Sigma^n)_{x \in \Sigma}$ eine Partition von Σ^{n+1} bildet. Weiter gilt

$$|L_M(q) \cap \{x\}\Sigma^n| = |L_M(\delta(q, x)) \cap \Sigma^n| = 4^n f_n(\delta(q, x))$$

Insgesamt erhalten wir also die Rekurrenz $f_{n+1}(q) = \frac{1}{4} \sum_{x \in \Sigma} f_n(\delta(q, x))$. Diese lässt sich leicht berechnen, z.B. mit folgendem Python3 Code.

```
Q = list(range(5))
d = [[1,0,0,0],[1,0,0,2],[1,3,0,0],[1,0,4,0],[4,4,4,4]]
f = [1,1,1,1,0]
for it in range(2024):
    f = [sum(f[p] for p in d[q])/4 for q in Q]
print(f[0])
```

Die ersten 40 Nachkommastellen sind wie folgt (dies kann man allerdings mit obigem Code nicht so genau berechnen).

$$f_{2024}(q_0) = 0.0003339121138034895068122731134195098853...$$