Prof. Javier Esparza
Philipp Czerner, Martin Helfrich

Technische Universität München
Lehrstuhl für Theoretische Informatik

Einführung in die Theoretische Informatik

Sommersemester 2021 – Hausaufgabenblatt 5

- Sei $\Phi := \{\{1\}, \{2,3\}, \{4\}, \{5\}, \{6\}\}$. Nach dem Abgabedatum werden wir für jede Menge $A \in \Phi$ eine zufällige Aufgabe $a \in A$ wählen und korrigieren.

  Update: Die Aufgaben $\{1, 2, 4, 5, 6\}$ werden korrigiert.
- Wenn Sie einen Beweis aufstellen, von dem Sie wissen, das einzelne Schritte problematisch oder unvollständig sind, merken Sie dies bitte in Ihrer Lösung an, damit wir das bei der Korrektur positiv berücksichtigen können.

Aufgabe H5.1. (*Automata Tutor*) 1+1 Punkte

Diese Hausaufgabe wird mit Automata Tutor bearbeitet und abgegeben. Falls Sie es noch nicht bereits gemacht haben, folgen Sie den Schritten in Ü1.2, um ein Konto zu erstellen. Achten Sie darauf, dass Sie sich, wie dort beschrieben, mit Ihrer TUM-Kennung anmelden. Ansonsten können wir Ihnen die Punkte nicht gutschreiben.

Bearbeiten Sie die Hausaufgaben H5.1 (a–b). **Achtung:** Während Sie für die Aufgaben aus dem Übungsblatt beliebig viele Versuche hatten, haben Sie für jede Hausaufgabe nur 5 Versuche. Sie bekommen nur dann einen Punkt, wenn Sie die Aufgabe nach 5 Versuchen vollständig (also mit 10/10 Punkten) gelöst haben.

Lösungsskizze.

(a) $S \to XS \mid \epsilon$
$X \to aXc \mid B$
$B \to bB \mid \epsilon$

(b) $S \to XY$
$X \to aXb \mid \epsilon$
$Y \to bYa \mid \epsilon$

Aufgabe H5.2. (*Gib mir ein a!*) 0.5+0.5+0.5+0.5 Punkte

Sei $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ eine Grammatik mit den Produktionen

$S \to AB \mid BC$ $\quad$ $A \to AA \mid a$
$B \to ABA \mid BCB \mid b$ $\quad$ $C \to ab \mid ba$

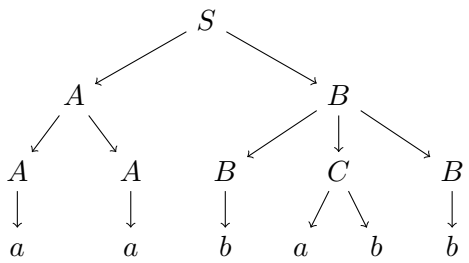
Sind die folgenden Wörter in $L(G)$ enthalten? Falls ja, geben sie einen Syntaxbaum und eine Linksableitung an, falls nein, begründen Sie dies *kurz*.

(a) ε (b) $aababb$ (c) $abaabbba$ (d) $abaabbbba$

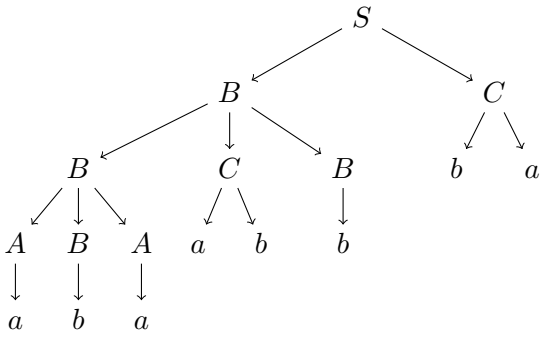
Lösungsskizze.

(a) Nein, da keine Produktion $\alpha \to \varepsilon$ enthalten ist.

(b) Ja, $S \to AB \to AAB \to aAB \to aaB \to aaBCB \to aabCB \to aababB \to aababb$.



(c) Ja, $S \to BC \to BCBC \to ABACBC \to aBACBC \to abACBC \to abaCBC \to abaabBC \to abaabbC \to abaabbba$.



(d) Nein, da kein Wort in G vier konsekutive b enthält. (Aus A werden nur Wörter in a^+ erzeugt, aus C nur Wörter, die ein a beinhalten und mit höchstens einem b anfangen oder aufhören. Somit aus B nur Wörter, die ein a beinhalten und mit höchstens zwei b anfangen oder aufhören, und höchstens drei konsekutive b beinhalten; ebenso für S.)

Aufgabe H5.3. (*Do you even pump?*) 1+1 Punkte

Sei $\Sigma := \{a, b\}$. Beweisen oder widerlegen Sie, dass die folgenden Sprachen regulär sind. Bitte beachten Sie, dass es beim Beweisen von Regularität regelmäßig nicht genügt, einen entsprechenden Automaten anzugeben – zumindest eine Begründung ist erforderlich. Um zu widerlegen, dass eine Sprache regulär ist, steht Ihnen die Wahl der Beweistechnik (Pumping Lemma, unendlich viele Äquivalenzklassen / Residualsprachen) frei.

Wie üblich schreiben wir $|x|_v := |\{u : x = uvw, u, w \in \Sigma^*\}|$ für die Anzahl der Vorkommnisse des Wortes $v \in \Sigma^*$ in $x \in \Sigma^*$, z.B. gilt $|abbbabb|_{bb} = 3$.

(a) $L = \{w \in \Sigma^* : |w|_{aa} = |w|_{bb}\}$

(b) $L = \{w \in \Sigma^* : |w|_{ab} = |w|_{ba}\}$

Lösungsskizze. **(a)** Nein, die Sprache ist nicht regulär. Wie zeigen dies, indem wir unendlich viele Residualsprachen finden (siehe Übungsaufgabe Ü5.6). Wir wählen $u_i := (aab)^i$, $v_i := (abb)^i$ für $i \in \mathbb{N}$. Beachten Sie, dass $|u_i v_j|_{aa} = i$ und $|u_i v_j|_{bb} = j$ für alle $i, j \in \mathbb{N}$. In der Residualsprache L^{u_i} ist v_i enthalten, aber kein v_j für $j \neq i$. Also sind die Residualsprachen $L^{u_0}, L^{u_1}, L^{u_2}, ...$ paarweise verschieden, und L somit nicht regulär.
(b) Ja, die Sprache ist regulär, sie enthält genau die Wörter, deren erster und letzter Buchstabe gleich sind (und das leere Wort). Dies ist dadurch begründet, dass auf ein ab kein weiteres ab folgen kann, ohne dass ein ba dazwischen entsteht, und umgekehrt. Die Differenz $|w|_{ab} - |w|_{ba}$ kann also nicht betragsmäßig größer als 1 werden. Ein regulärer Ausdruck für die Sprache ist $a(a|b)^*a \,|\, b(a|b)^*b \,|\, a \,|\, b \,|\, \boldsymbol{\epsilon}$.

Aufgabe H5.4. (*no context*) 1+1+2+1 Punkte

Sei $G = (\{S\}, \{a, b\}, P, S)$ eine Grammatik mit den Produktionen $S \to aSbS \mid bSaS \mid \varepsilon$.

(a) Geben Sie eine Sprache L mit $L = L(G)$ an. Definieren Sie L auf möglichst leichte Art und Weise (insbesondere ohne G zu verwenden).

(b) Zeigen Sie, dass jedes Wort $w \in L \setminus \{\varepsilon\}$ sich in $aubv = w$ oder $buav = w$ zerlegen lässt, mit $u, v \in L$.

Achtung: L ist die Sprache, die Sie in (a) angegeben haben, nicht $L(G)$!

(c) Verwenden Sie (b) und beweisen Sie $L(G) = L$.

(d) Ist G mehrdeutig? Begründen Sie Ihre Antwort.

Lösungsskizze. **(a)** (Anmerkung: Zur Lösung dieser Aufgabe genügt eine Zeile. Wir wollen aber auch schildern, wie man auf diese Lösung kommt.)

Zunächst ist es immer eine gute Idee, sich zu überlegen, welche Arten von Wörtern in der Sprache enthalten sind (untere Schranke) und welche Arten definitiv nicht enthalten sind (obere Schranke).

Obere Schranke: Für eine obere Schranke suchen wir nach Invarianten, wie etwa „Jede Produktion aus dem Nichtterminal ... erzeugt gleich viele ... wie ...“. Hier ist leicht zu sehen, dass jede Produktion (aus dem einzigen Nichtterminal S) gleich viele a und b erzeugt. Es gilt also $L(G) \subseteq \{w \in \Sigma^* : |w|_a = |w|_b\}$.

Welche Wörter liegen sicher in der Sprache? Hierfür versuchen wir immer, auf einige Nichtterminale sofort die ε-Produktion zu verwenden. Zum Beispiel hier $S \to aSbS \to aSb$. Damit können wir nun $a^n b^n$ für alle $n \in \mathbb{N}$ erzeugen. Ähnlich (mit der zweiten Produktion) enthält die Sprache auch $b^n a^n$ für alle $n \in \mathbb{N}$. Andererseits ist diese Grammatik (wenn wir bei beiden Produktionen wieder das rechte S durch ε ersetzen) ähnlich zu der Grammatik für Palindrome. Aus dieser Ähnlichkeit kann man erkennen, dass diese Grammatik alle Wörter gerader Länge enthält, die von rechts und links gelesen genau entgegengesetzt sind. Immer wenn das i-te Zeichen von links ein a ist, ist das i-te Zeichen von rechts ein b, zum Beispiel bei $abbaab$. Als Erklärung, wie wir diese Wörter produzieren: Wir produzieren immer das erste und letzte Zeichen gleichzeitig, dann das zweite und vorletzte usw. Bis jetzt haben wir das zweite S noch nicht benutzt – die Grammatik erzeugt also wahrscheinlich wesentlich mehr Wörter.

Zusammengefasst: Die untere Schranke wirkt kompliziert, als obere Schranke finden wir auf die Schnelle eine Invariante, nämlich dass es gleich viele a wie b gibt. In der Tat stellt sich nachher in Aufgabenteil (c) heraus, dass diese simple obere Schranke $L(G) = \{w \in \Sigma^* : |w|_a = |w|_b\}$ bereits genau unsere Sprache ist. Wichtig: Soweit steht hier noch kein formaler Beweis für diese Aussage!

(b) Unsere Idee ist, dass wir uns den ersten Buchstaben des Wortes anschauen, und dann einen „passenden“ Buchstaben im Wort suchen. Für $aababbba$ hätten wir beispielsweise folgende Optionen:

(1) $a\,a\,b\,abbba$ (2) $a\,aba\,b\,bba$ (3) $a\,abab\,b\,ba$ (4) $a\,ababb\,b\,a$

Option (1) funktioniert nicht, da $a \notin L$. Es gibt vor dem ersten b also zu viele a, ebenso wie Option (2). Option (4) funktioniert ebenfalls nicht, in $ababb \notin L$ sind zu viele b enthalten. Aber Option (3) passt, $abab \in L$ und $ba \in L$ gelten. Wir können also nicht immer das erste oder das letzte b nehmen, sondern wir müssen die Möglichkeiten so lange durchgehen, bis eine „passt“. Nun der eigentliche Beweis:

Sei $w \in L$, $w \neq \varepsilon$ ein nichtleeres Wort in L und $n := |w|$ seine Länge. Wir betrachten den Fall, dass das erste Zeichen von w ein a ist, der Fall für b ist analog.

Wir gehen ähnlich zum Beweis von Satz 4.16 vor und betrachten die Funktion

$$h : \{1, ..., n\} \to \mathbb{Z},\ h(i) := |w_1 w_2 ... w_i|_a - |w_1 w_2 ... w_i|_b$$

Um $h(i)$ auszurechnen lesen wir also die Zeichen $w_1, w_2, ..., w_i$ und rechnen $+1$ für jedes a, und -1 für jedes b. Da $w \in L$ muss $h(n) = 0$ gelten, wir können also das auch kleinste i wählen, sodass $h(i) = 0$ gilt. In jedem Schritt kann sich h nur um maximal 1 verändern, $h(i-1) = 1$ muss also gelten (wenn $h(i-1) = -1$ wäre, würde es ein

kleineres i mit $h(i) = 0$ geben). Daraus folgt $w_i = b$, und wir definieren nun unsere Zerlegung als $u := w_2w_3...w_{i-1}$, $v =: w_{i+1}w_{i+2}...w_n$. (Beachten Sie, dass nach Konvention $w_1w_2...w_i = \varepsilon$ für $i = 0$.)

Schließlich erhalten wir $u, v \in L$ aufgrund von

$$|u|_a - |u|_b = h(i-1) - h(1) = 1 - 1 = 0$$

$$|v|_a - |v|_b = h(n) - h(i) = 0 - 0 = 0$$

(c) Wir teilen den Beweis von $L(G) = L$ in zwei Teile auf.

„$\subseteq$“: Wir müssen also zeigen, dass jedes von G erzeugte Wort gleich viele a und b hat.

Möglichkeit 1: Wir zeigen die stärkere Aussage, dass jede von G ableitbare Satzform w (also ein aus Terminalen und Nichtterminalen bestehendes Wort) $|w|_a - |w|_b = 0$ erfüllt. Dies beweisen wir mit vollständiger Induktion über die Länge der Produktionskette.

Induktionsbasis: $|S|_a = 0 = |S|_b$

Induktionsschritt: Seien also $w, w' \in \{a, b, S\}$ mit $w \to w'$. Wenn Regel $S \to \varepsilon$ angewandt wurde, gilt $|w'|_a = |w|_a$ und $|w'|_b = |w|_b$. Ansonsten wurde $S \to aSbS \mid bSaS$ angewandt und wir erhalten $|w'|_a = |w|_a + 1$ und $|w'|_b = |w|_b + 1$. In allen Fällen gilt $|w'|_a - |w'|_b = |w|_a - |w|_b$, was nach Induktionsannahme 0 ist.

Möglichkeit 2: Wir verwenden wie in der Vorlesung die induktive Definition einer Sprache (Folie mit Beispiel 4.10, Seite 137 im Handout). Für unsere Grammatik erhalten wir (steht für allgemeine Grammatiken 8 Seiten weiter auf Seite 145):

- $\varepsilon \in L_G(S)$.
- $u \in L_G(S), v \in L_G(S) \Rightarrow aubv \in L_G(S)$.
- $u \in L_G(S), v \in L_G(S) \Rightarrow buav \in L_G(S)$.

Um also das Prädikat $P(w) \Leftrightarrow |w|_a = |w|_b$ per Induktion über die Erzeugung zu beweisen, müssen wir folgende Aussagen beweisen:

1.: $P(\varepsilon)$: Trivial.

2.: $P(u) \wedge P(v) \Rightarrow P(aubv)$: Es gilt

$$|aubv|_a = 1 + |u|_a + |v|_a =_{[P(u)\wedge P(v)]} 1 + |u|_b + |v|_b = |aubv|_b.$$

3.: $P(u) \wedge P(v) \Rightarrow P(buav)$: Es gilt

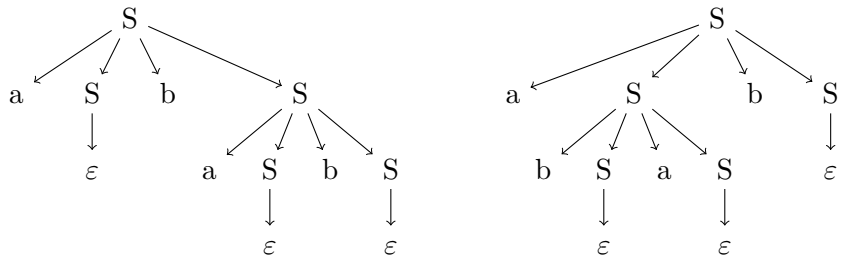
$$|buav|_a = |u|_a + 1 + |v|_a =_{[P(u)\wedge P(v)]} |u|_b + 1 + |v|_b = |buav|_b.$$

„$\supseteq$“: Jetzt zeigen wir, dass jedes Wort w mit gleich vielen a und b sich aus G erzeugen lässt. Dies geschieht per Induktion über die Wortlänge.

Induktionsbasis: $S \to \varepsilon$ ist trivial.

Induktionsschritt: Sei $w \in L$ beliebig, mit $w \neq \varepsilon$. Nach (b) gilt $w = aubv$ oder $w = buav$ für $u, v \in L$. Im ersten Fall können wir $S \to aSbS$ ableiten, und aus $u, v \in L$ folgt nach Induktionsannahme $S \to^* u$ und $S \to^* v$, da $|u|, |v| < |w|$. Insgesamt erhalten wir $S \to aSbS \to^* aubS \to^* aubv = w$. Der zweite Fall $w = buav$ ist analog.

(d) Die Grammatik ist mehrdeutig, denn das Wort $w = abab$ hat zwei Syntaxbäume.



Aufgabe H5.5. (*dada*) 0.5+2+0.5 Punkte

Der kleine Theo ist schon wieder traurig. Er würde gerne Kekse backen, muss aber zuerst herausfinden, ob es ein Wort gibt, das er seinem noch kleineren Neffen beibringen kann. Theo hat sich schon aufgeschrieben, welche Arten von Wörtern sein Neffe bereits gesagt hat. Nun muss er aber ein möglichst kurzes Wort finden, das anders ist. Dazu würde er gerne den kanonischen Minimalautomaten konstruieren, hat aber Riesenprobleme, die Äquivalenzklassen aufzustellen. Sie haben neulich ein Gerücht gehört, dass sich der minimale DFA auch über Residualsprachen konstruieren lässt, und wollen Theo helfen. Zuerst müssen Sie ihm das Konzept einer Residualsprache erklären.

Sei $L \subseteq \Sigma^*$ eine Sprache über Σ. Für ein Wort $w \in \Sigma^*$ definieren wir die *Residualsprache* $L^w := \{u : wu \in L\}$. Die Residualsprache L^w enthält also genau die Wörter in L, die mit w beginnen, wobei das führende w entfernt wurde. Beispielsweise gilt $L(ad^* \mid da^*)^a = L(d^*)$ und $L(ad^* \mid da^*)^{ada} = \emptyset$.

(a) Sei $s := a^*da^* \mid d^*a(ad \mid dd)$. Geben Sie für $L(s)^d$, $L(s)^{da}$, $L(s)^{dad}$ und $L(s)^{dada}$ jeweils einen regulären Ausdruck an.

Hinweis: Es gilt $L^{uv} = (L^u)^v$ für eine beliebige Sprache L und Wörter u, v.

Wie in Übungsaufgabe Ü5.6 (a) gezeigt hängt die Residualsprache L^w nicht davon ab, welchen Repräsentanten w wir aus $[w]_{\equiv_L}$ wählen. Um den kanonischen Minimalautomaten einer Sprache L aufzuschreiben, können wir als Zustände anstatt der Äquivalenzklassen von L also auch die Residualsprachen von L verwenden.

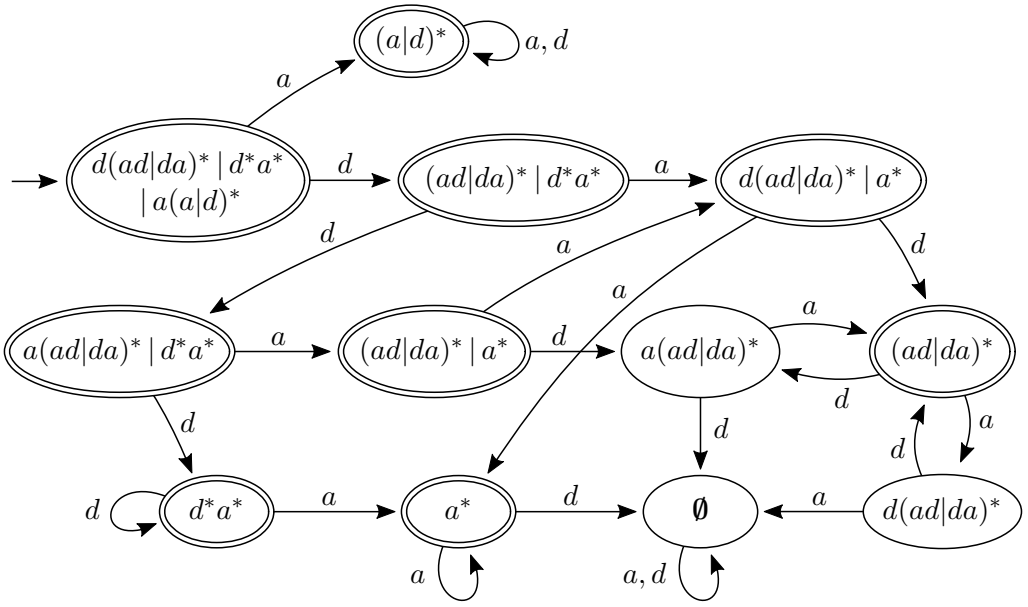
Sei nun $\Sigma := \{a, d\}$ und $r := d(ad \mid da)^* \mid d^*a^* \mid a(a \mid d)^*$. Wir suchen das kürzeste Wort $w \notin L(r)$. Dazu wollen wir den minimalen DFA für $L(r)$ erstellen.

(b) Konstruieren Sie den kanonischen Minimalautomat für $L(r)$. Beschriften Sie hierbei einen Zustand $[w]_{\equiv_{L(r)}}$ nicht mit dessen Äquivalenzklasse, sondern mit einem regulären Ausdruck für die Residualsprache $L(r)^w$.

(c) Finden Sie alle kürzesten Wörter $w \notin L(r)$.

Lösungsskizze. **(a)** $a^* \mid d^*a(ad \mid dd)$, $a^* \mid (ad \mid dd)$, d und $\emptyset$.

(b) Der minimale DFA sieht wie folgt aus:



(c) Die möglichen Wörter sind *daad*, *dada*, *dadd*, und *ddad*.

Bonusaufgabe H5.6. (*Irrational irregulär*) 1+3 Bonuspunkte

Sei $\Sigma := \{0, ..., 9\}$. Für Wörter $u, v \in \Sigma^*$ bezeichnen wir mit $(u.v)_{10} := (u)_{10} + 10^{-|v|} \cdot (v)_{10}$ den Wert eines Dezimalbruchs, es gilt also z.B. $(1.75)_{10} = \frac{7}{4}$.

(a) Zeigen Sie, dass $L_1 := \{w \in \Sigma^* : (0.w)_{10} < \frac{1}{7}\}$ regulär ist.

(b) Beweisen Sie, dass $L_2 := \{w \in \Sigma^* : (0.w)_{10} < \frac{1}{\sqrt{2}}\}$ nicht regulär ist, indem sie unendlich viele Äquivalenzklassen (oder Residualsprachen, s. Übungsaufgabe Ü5.6) von $\equiv_{L_2}$ identifizieren. (Sie müssen nicht angeben, welche Elemente in den Äquivalenzklassen / Residualsprachen enthalten sind.)

Lösungsskizze. **(a)** In Dezimalschreibweise erhalten wir $\frac{1}{7} = 0.\overline{142857}$. Wir sind nun an den endlichen Dezimalbrüchen interessiert, die kleiner sind, wobei wir die Ziffern von links nach rechts einlesen. Zur Veranschaulichung ein paar Beispiele: Das Wort 27384 würden wir nicht akzeptieren, da $0.27384 > 0.\overline{142857}$. Dies lässt sich bereits an der ersten Ziffer erkennen, da $0.2 > 0.\overline{142857}$. Es ist nicht möglich, die Zahl zu verkleinern, indem weitere Ziffern angehängt werden. Umgekehrt würden wir 1234 akzeptieren, da $0.1234 < 0.\overline{142857}$. Hier lässt sich dies bereits nach der zweiten Ziffer bestimmen, denn $0.12\overline{9} = 0.13 < 0.\overline{142857}$ – auch wenn nach Lesen von 12 stets die größtmögliche Ziffer käme, können wir nur Zahlen erreichen, die kleiner als $\frac{1}{7}$ sind.

Nun machen wir diese Überlegung formal. Sei $f_1, f_2, ... \in \Sigma$ eine Folge, sodass $f_1 f_2 ... f_6 =$ 142857 und $f_{i+6} = f_i$ für $i \in \mathbb{N}_+$. Die Folge $(f_i)_{i \in \mathbb{N}_+}$ entspricht also den Ziffern der Dezimaldarstellung von $\frac{1}{7}$. Es gilt

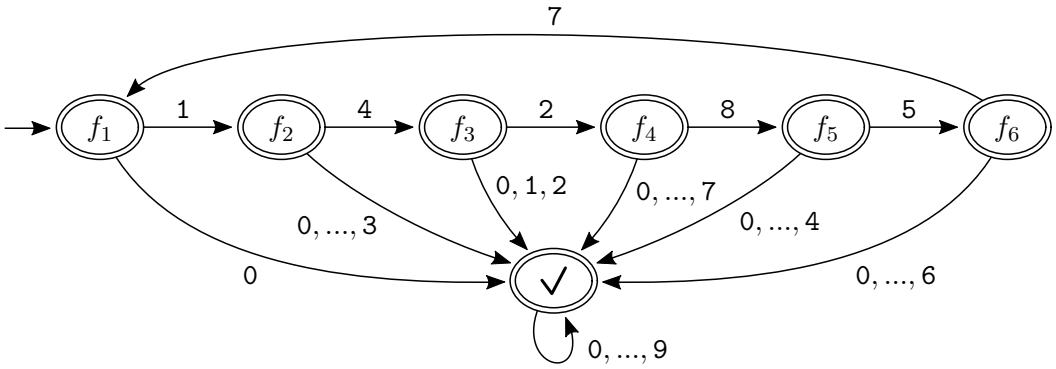
$$
\begin{aligned}
& (0.w)_{10} < \tfrac{1}{7} \\
\Leftrightarrow\ & 10^{-|w|}(w)_{10} < 0.\overline{142857} \\
\Leftrightarrow\ & (w)_{10} < (f_1 f_2 ... f_{|w|})_{10} + \alpha
\end{aligned}
$$

wobei $0 < \alpha < 1$ hier eine geeignete Konstante ist. Mit $f := f_1 f_2 ... f_{|w|}$ erhalten wir

$$\Leftrightarrow (w)_{10} \leq (f_1 f_2 ... f_{|w|})_{10} = (f)_{10}$$
$$\Leftrightarrow w \leq_{\text{lex}} f$$

Hier bezeichnet $\leq_{\text{lex}}$ die lexikographische Sortierung: wir vergleichen zunächst w_1 und f_1. Wenn $w_1 < f_1$, dann ist $w <_{\text{lex}} f$, wenn $w_1 > f_1$, dann ist $w >_{\text{lex}} f$. Ansonsten gilt $w_1 = f_1$, und wir wiederholen diese Prozedur mit w_2 und f_2. (Nach diesem Schema werden auch Wörter in einem Lexikon sortiert, daher der Name.)

Da sich die Zeichen von f wiederholen, lässt sich leicht ein DFA zeichnen, der diese Vergleiche ausführt.



Nicht gezeichnete Kanten führen zu einem impliziten Fangzustand. In Zustand $f_i \in \{f_1, f_2, ..., f_6\}$ wird das eingelesene Zeichen c mit f_i verglichen. Falls $c < f_i$, gehen wir zu einem akzeptieren Zustand, der nicht wieder verlassen werden kann. Für $c > f_i$ gehen wir in einen Fangzustand, und wenn $c = f_i$ machen wir weiter mit $f_{(i \bmod 6)+1}$.

(b) Die Idee des Beweises ist, dass nach Einlesen der ersten i Nachkommastellen von $\frac{1}{\sqrt{2}}$ die Residualsprachen alle unterschiedlich sein müssen, da $\frac{1}{\sqrt{2}}$ irrational ist. Falls der DFA nach Einlesen von i und von $j \neq i$ Nachkommastellen im gleichen Zustand wäre, können wir folgern, dass die Dezimaldarstellung von $\frac{1}{\sqrt{2}}$ periodisch wäre, was aber nicht der Fall ist. Da der Automat aber nur endlich viele Zustände hat, muss dies irgendwann geschehen. Es folgt der formale Beweis.

Sei $0.f_1 f_2 ... := 0.707106781186...$ die Ziffernfolge der Dezimaldarstellung von $\frac{1}{\sqrt{2}}$. Sei nun $i \in \mathbb{N}$ beliebig, und sei $f := f_1 f_2 ... f_i$. Wir betrachten die Residualsprache L_2^f; insbesondere sind wir an dem Supremum von $M := \{(0.w)_{10} : w \in L_2^f\}$ interessiert. Wir behaupten nun, dass $\sup M = 10^i \cdot \frac{1}{\sqrt{2}} - (f)_{10} =: x$ gilt. Wie üblich, erfolgt dieser Beweis in zwei Schritten: zunächst zeigen wir, dass x eine obere Schranke ist, und dann, dass es keine kleinere obere Schranke gibt.

- „x ist obere Schranke“: Angenommen, dies wäre nicht der Fall. Dann gibt es ein $w \in L_2^f$ mit $(0.w)_{10} > x$. Aus $w \in L_2^f$ folgt nach Definition von Residualsprachen $fw \in L_2$, also gilt

  $$10^i (0.fw)_{10} \overset{(1)}{=} (f)_{10} + (0.w)_{10} \overset{(2)}{>} (f)_{10} + 10^i \cdot \tfrac{1}{\sqrt{2}} - (f)_{10}$$

  Für (1) verwenden wir $|f| = i$, für (2) $(0.w)_{10} > x$. Insgesamt erhalten wir $(0.fw)_{10} > \frac{1}{\sqrt{2}}$, was im Widerspruch zu $fw \in L_2$ steht.

- „keine kleinere obere Schranke“: Sei nun $y < x$ beliebig. Wir wählen nun ein $w \in \Sigma^*$ mit $y < (\mathtt{0}.w)_{10} < x$, d.h. w stellt einen Dezimalbruch zwischen x und y dar. Nun gilt

  $$10^i(\mathtt{0}.fw)_{10} = (f)_{10} + (\mathtt{0}.w)_{10} < (f)_{10} + x = \tfrac{1}{\sqrt{2}}$$

  Also ist $fw \in L_2$ und somit $w \in L_2^f$ und schließlich $(\mathtt{0}.w)_{10} \in M$. Da $y < (\mathtt{0}.w)_{10}$, ist y keine obere Schranke vom M.

Die Behauptung ist somit bewiesen. Wenn L_2 regulär wäre, kann es – wie in Übungsaufgabe Ü5.6 gezeigt – nur endliche viele Residualsprachen geben. Obige Ausführungen gelten für ein beliebiges i, wir wählen also i, i' mit $L_2^f = L_2^{f'}$, wobei wir $f := f_1 f_2 ... f_i$ und $f' := f_1 f_2 ... f_{i'}$ definieren. Weiterhin definieren wir M, M' und x, x' entsprechend. Da $L_2^f = L_2^{f'}$ muss auch $M = M'$ und somit $x = \sup M = \sup M' = x'$ gelten. Die Aussage $x = x'$ ist nach Definition von x äquivalent zu

$$\begin{aligned}
& 10^i \cdot \tfrac{1}{\sqrt{2}} - (f)_{10} = 10^{i'} \cdot \tfrac{1}{\sqrt{2}} - (f')_{10} \\
\Rightarrow\ & (10^i - 10^{i'}) \cdot \tfrac{1}{\sqrt{2}} = (f)_{10} - (f')_{10} \\
\Rightarrow\ & \sqrt{2} = \frac{10^i - 10^{i'}}{(f)_{10} - (f')_{10}} \in \mathbb{Q}
\end{aligned}$$

Da $10^i, 10^{i'}, (f)_{10} - (f')_{10}$ rationale Zahlen sind, müsste $\sqrt{2}$ es demnach auch sein. Dies ist aber bekanntermaßen nicht der Fall und L_2 kann somit nicht regulär sein sein.