ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Informatik

Theoretische Informatik

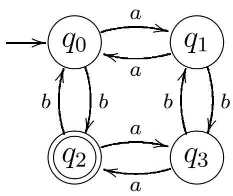
Prof. Dr. J. Hromkovič
Prof. Dr. M. Bläser

Musterlösungen zur 4. Übung

Zürich, 12. November 2004

Lösung zu Aufgabe 12

Wir bemerken zunächst, dass $|x|_b \bmod 4 \in \{1,3\}$ nichts anderes bedeutet, als dass $|x|_b$ ungerade ist. Es ergibt sich folgender Automat:

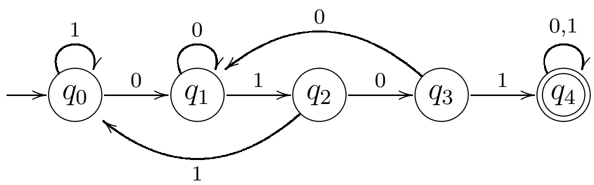


Es gilt:

$$\begin{aligned}
\mathrm{Kl}[q_0] &= \{x \in \{a,b\}^* \mid |x|_a \text{ und } |x|_b \text{ sind beide gerade}\}\\
\mathrm{Kl}[q_1] &= \{x \in \{a,b\}^* \mid |x|_a \text{ ist ungerade und } |x|_b \text{ ist gerade}\}\\
\mathrm{Kl}[q_2] &= \{x \in \{a,b\}^* \mid |x|_a \text{ ist gerade und } |x|_b \text{ ist ungerade}\}\\
\mathrm{Kl}[q_3] &= \{x \in \{a,b\}^* \mid |x|_a \text{ und } |x|_b \text{ sind beide ungerade}\}
\end{aligned}$$

Lösung zu Aufgabe 13

Idee: In Abhängigkeit von der Länge desjenigen Präfixes von 0101, das bereits erkannt wurde, befindet sich der Automat in verschiedenen Zuständen. Wird das «richtige» nächste Zeichen erkannt, gelangt man in den nächsten Zustand; falls nicht, verbleibt man im selben Zustand bzw. gelangt in einen vorherigen Zustand zurück.
Hieraus ergibt sich folgender Automat:



Zur kompakten Beschreibung der Zustandsklassen definieren wir $S := \{0,1\}^* \cdot \{01\} \cup L$, die Sprache aller Wörter, die entweder auf 01 enden oder 0101 enthalten. Nun gilt:

$$\begin{aligned}
\mathrm{Kl}[q_0] &= \{x \in \{0,1\}^* \mid x \text{ enthält 0101 nicht als Infix,} \\
&\qquad \text{und kein Präfix von 0101 ist Suffix von } x\} \\
\mathrm{Kl}[q_1] &= \{x0 \mid x \in \{0,1\}^* \setminus S\} \\
\mathrm{Kl}[q_2] &= \{x01 \mid x \in \{0,1\}^* \setminus S\} \\
\mathrm{Kl}[q_3] &= \{x010 \mid x \in \{0,1\}^* \setminus S\} \\
\mathrm{Kl}[q_4] &= \{x0101y \mid x, y \in \{0,1\}^*\} = L
\end{aligned}$$

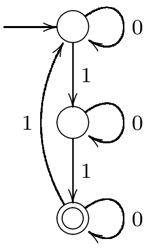
Lösung zu Aufgabe 14

Es seien

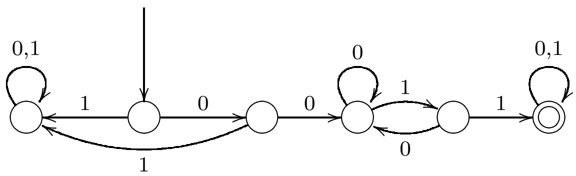
$$\begin{aligned}
L_1 &:= \{x \in \{0,1\}^* \mid |x|_1 \bmod 3 = 2\} \text{ und} \\
L_2 &:= \{00y11z \mid y, z \in \{0,1\}^*\}.
\end{aligned}$$

Damit ist offensichtlich $L = L_1 \cap L_2$. Ferner seien M_1 und M_2 gegeben durch:

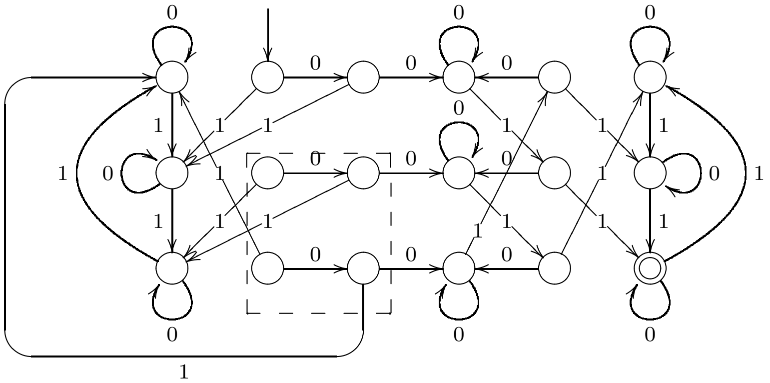
M_1 :



M_2 :



Der Übersichtlichkeit halber verzichten wir auf eine Beschriftung der Zustände. Aus M_1 und M_2 ergibt sich der Produktautomat M, dessen Zustände wir tabellarisch anordnen (die Zeile entspricht dem jeweiligen Zustand aus M_1, die Spalte demjenigen aus M_2):



Die durch das Rechteck markierten Zustände sind tatsächlich nicht erreichbar und können somit auch weggelassen werden.

Lösung zu Aufgabe 15

Die Bahuptung ist *nicht* korrekt. Wir beweisen $L \notin \mathcal{L}_{RE}$ mit Hilfe des Pumping-Lemmas (Lemma 3.4):
Angenommen also, L sei regulär. Dann existiert eine Konstante $n_0 \in \mathbb{N}$, so dass sich jedes Wort $w \in \Sigma^*$ mit $|w| \geq n_0$ in drei Teile y, x und z zerlegen lässt, d.h. $w = yxz$, wobei

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

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

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

Sei also n_0 diese Konstante und $w := 0^{n_0}1^{2n_0}2^{3n_0}$. Wegen

$$|w| = |0^{n_0}1^{2n_0}2^{3n_0}| = n_0 + 2 \cdot n_0 + 3 \cdot n_0 = 6 \cdot n_0$$

ist $|w| \geq n_0$ erfüllt, also lässt sich w wie oben beschrieben in y, x und z zerlegen. Wegen $|yx| \leq n_0$ gilt:

$$yx = 0^{|yx|} \quad (\text{also } w = \underbrace{0\ldots0}_{y}\underbrace{0\ldots0}_{x}\ldots1\ldots2). \qquad (*)$$

Mit $k = 1$ gilt ferner: $L \ni w = yxz = yx^1z \in \{yx^k z \mid k \in \mathbb{N}\}$. Also gilt $\{yx^k z \mid k \in \mathbb{N}\} \cap L \neq \emptyset$ und daher gemäss Eigenschaft (iii): $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L$. Folglich gilt:

$$yz = yx^0 z \in \{yx^k z \mid k \in \mathbb{N}\} \implies yz \in L$$

Aus Beobachtung $(*)$ folgt aber zusammen mit Eigenschaft (ii), dass yz aus $yxz = w$ lediglich durch Streichen einer positiven Anzahl von Nullen hervorgeht, weswegen $yz \notin L$ (denn $|z|_1 = |w|_1$ und $|z|_2 = |w|_2$).
Also ist die Annahme, L sei regulär, falsch.

Lösung zu Aufgabe 16

Wir beweisen sogar eine etwas stärkere Behauptung.
Behauptung: Für alle Zustände $p_1, q_1 \in Q_1$ und alle Zustände $p_2, q_2 \in Q_2$ und alle Wörter $x \in \Sigma^*$ gilt:

$$(q_1, x) \vdash^*_{M_1} (p_1, \lambda) \text{ und } (q_2, x) \vdash^*_{M_2} (p_2, \lambda) \iff ((q_1, q_2), x) \vdash^*_{M} ((p_1, p_2), \lambda)$$

Wir führen einen Induktionsbeweis über $|x|$.
Induktionsanfang: Sei $|x| = 0$, also $x = \lambda$.
Aus der Definition der Schrittrelation $\vdash_{\widetilde{M}}$ ($\widetilde{M} \in \{M, M_1, M_2\}$) wissen wir, dass es für keinen Zustand q eine Konfiguration C gibt mit $(q, \lambda) \vdash_{\widetilde{M}} C$. Daher ist $(q, \lambda) \vdash^*_{\widetilde{M}} (p, \lambda)$ genau dann erfüllt, wenn $p = q$. Damit folgt die Aussage für $x = \lambda$ unmittelbar.

Induktionsschritt: Es sei $n \in \mathbb{N} \setminus \{0\}$ eine natürliche Zahl, und wir nehmen an, die zu zeigende Aussage gelte für alle $x \in \Sigma^*$ mit $|x| < n$. Es sei nun $x \in \Sigma^n$ und $x = az$ für ein $a \in \Sigma$.

„$\Rightarrow$" Es gelte: $(q_1, az) \vdash^*_{M_1} (p_1, \lambda)$ und $(q_2, az) \vdash^*_{M_2} (p_2, \lambda)$. Dann gilt mit

$$\begin{aligned} s_1 &:= \delta_1(q_1, a) \text{ und} \\ s_2 &:= \delta_2(q_2, a) \end{aligned}$$

die Aussage:

$$\begin{aligned} &(q_1, az) \vdash_{M_1} (s_1, z) \vdash^*_{M_1} (p_1, \lambda) \text{ und} \\ &(q_2, az) \vdash_{M_2} \underbrace{(s_2, z) \vdash^*_{M_2} (p_2, \lambda)}_{(*)} \end{aligned}$$

Weil $|z| = |x| - 1 = n - 1 < n$ folgern wir aus Aussage $(*)$ und der Induktionsvoraussetzung, dass $((s_1, s_2), z) \vdash^*_M ((p_1, p_2), \lambda)$ gilt. Folglich:

$$\begin{aligned} &((q_1, q_2), az) \vdash_M ((s_1, s_2), z) \vdash^*_M ((p_1, p_2), \lambda) \\ &\implies ((q_1, q_2), w) \vdash^*_M ((p_1, p_2), \lambda) \end{aligned}$$

„$\Leftarrow$" Es gelte: $((q_1, q_2), az) \vdash^*_M ((p_1, p_2), \lambda)$. Dann gelten mit

$$(s_1, s_2) := \delta((q_1, q_2), a)$$

die Aussagen

$$\begin{aligned} s_1 &= \delta_1(q_1, a) \text{ und} \\ s_2 &= \delta_2(q_2, a) \end{aligned}$$

und daher:

$$((q_1, q_2), az) \vdash_M \underbrace{((s_1, s_2), z) \vdash^*_M ((p_1, p_2), \lambda)}_{(**)},$$

Nach Induktionsvoraussetzung sind die Aussagen $(**)$ und $(***)$ äquivalent:

$$\begin{aligned} &(q_1, az) \vdash_{M_1} \overbrace{(s_1, z) \vdash^*_{M_1} (p_1, \lambda)}^{(***)} \text{ und} \\ &(q_2, az) \vdash_{M_2} (s_2, z) \vdash^*_{M_2} (p_2, \lambda), \end{aligned}$$

und hieraus folgt offensichtlich:

$$(q_1, x) \vdash^*_{M_1} (p_1, \lambda) \text{ und } (q_2, x) \vdash^*_{M_2} (p_2, \lambda),$$

wie gewünscht.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Informatik

Theoretische Informatik

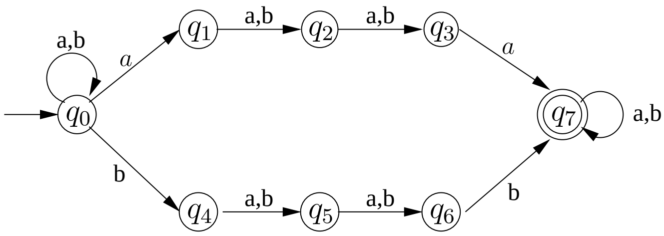
Prof. Dr. J. Hromkovič
Prof. Dr. M. Bläser

Musterlösungen zur 5. Übung

Zürich, 19. November 2004

Lösung zu Aufgabe 17

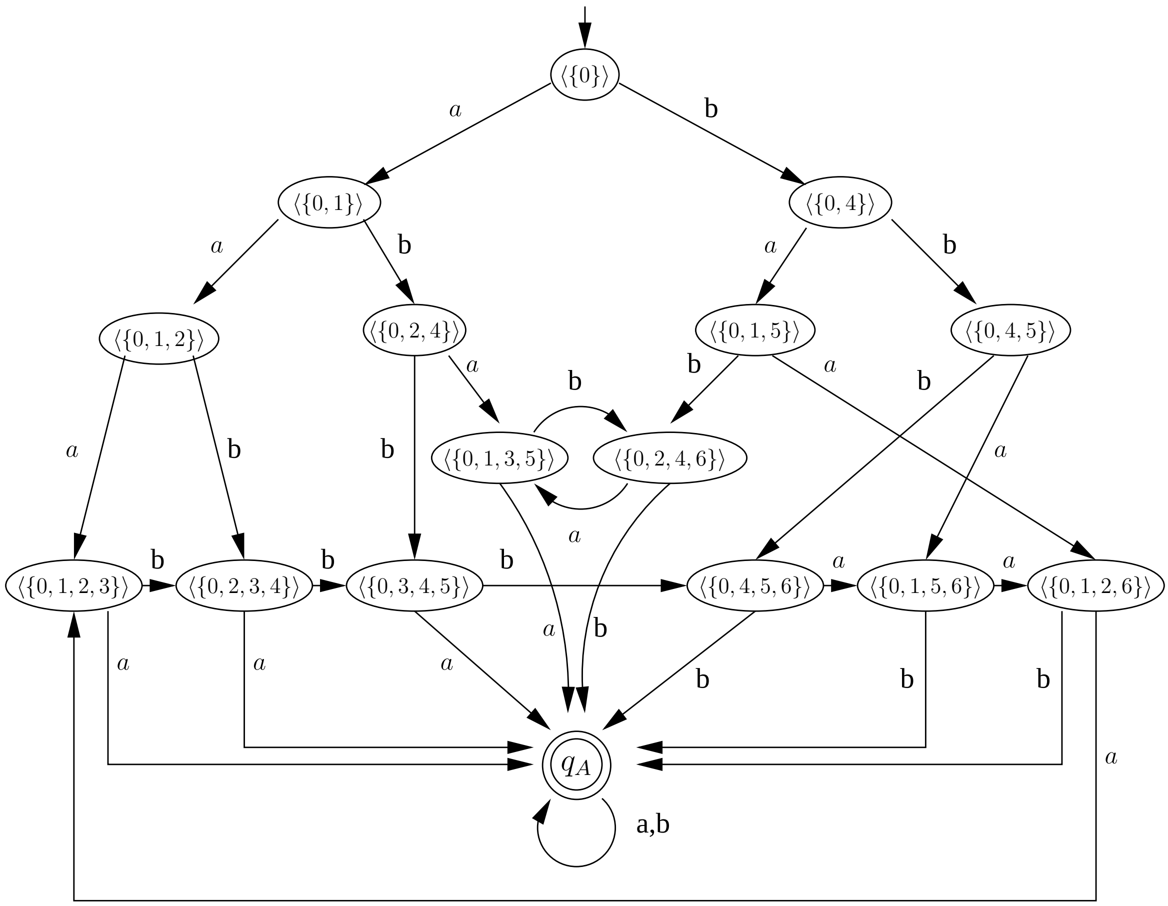
Idee: Ein nichtdeterministischer endlicher Automat kann die Position j erraten. Daher konstruieren wir zunächst einen NEA A, der L akzeptiert.



Aus diesem NEA A generieren wir mit Hilfe der Potenzmengenkonstruktion einen äquivalenten EA B. Da der NEA A genau einen akzeptierenden Zustand q_7 hat, aus dem keine Transitionen herausführen, können wir bei der Durchführung der Potenzmengenkonstruktion einen gesonderten akzeptierenden Zustand q_A für alle Zustände verwenden, die q_7 beinhalten. Es ergibt sich die folgende Transitionstabelle für B:

Zustand	a	b
$\langle\{q_0\}\rangle$	$\langle\{q_0, q_1\}\rangle$	$\langle\{q_0, q_4\}\rangle$
$\langle\{q_0, q_1\}\rangle$	$\langle\{q_0, q_1, q_2\}\rangle$	$\langle\{q_0, q_2, q_4\}\rangle$
$\langle\{q_0, q_4\}\rangle$	$\langle\{q_0, q_1, q_5\}\rangle$	$\langle\{q_0, q_4, q_5\}\rangle$
$\langle\{q_0, q_1, q_2\}\rangle$	$\langle\{q_0, q_1, q_2, q_3\}\rangle$	$\langle\{q_0, q_2, q_3, q_4\}\rangle$
$\langle\{q_0, q_2, q_4\}\rangle$	$\langle\{q_0, q_1, q_3, q_5\}\rangle$	$\langle\{q_0, q_3, q_4, q_5\}\rangle$
$\langle\{q_0, q_1, q_5\}\rangle$	$\langle\{q_0, q_1, q_2, q_6\}\rangle$	$\langle\{q_0, q_2, q_4, q_6\}\rangle$
$\langle\{q_0, q_4, q_5\}\rangle$	$\langle\{q_0, q_1, q_5, q_6\}\rangle$	$\langle\{q_0, q_4, q_5, q_6\}\rangle$
$\langle\{q_0, q_1, q_2, q_3\}\rangle$	q_A	$\langle\{q_0, q_2, q_3, q_4\}\rangle$
$\langle\{q_0, q_2, q_3, q_4\}\rangle$	q_A	$\langle\{q_0, q_3, q_4, q_5\}\rangle$
$\langle\{q_0, q_1, q_3, q_5\}\rangle$	q_A	$\langle\{q_0, q_2, q_4, q_6\}\rangle$
$\langle\{q_0, q_3, q_4, q_5\}\rangle$	q_A	$\langle\{q_0, q_4, q_5, q_6\}\rangle$
$\langle\{q_0, q_1, q_2, q_6\}\rangle$	$\langle\{q_0, q_1, q_2, q_3\}\rangle$	q_A
$\langle\{q_0, q_2, q_4, q_6\}\rangle$	$\langle\{q_0, q_1, q_3, q_5\}\rangle$	q_A
$\langle\{q_0, q_1, q_5, q_6\}\rangle$	$\langle\{q_0, q_1, q_2, q_6\}\rangle$	q_A
$\langle\{q_0, q_4, q_5, q_6\}\rangle$	$\langle\{q_0, q_1, q_5, q_6\}\rangle$	q_A
q_A	q_A	q_A

Der EA B lässt sich folgendermaßen graphisch darstellen. Zur Vereinfachung schreiben wir in der Abbildung den Zustand $\langle\{q_{i_1}, \ldots, q_{i_k}\}\rangle$ als $\langle\{i_1, \ldots, i_k\}\rangle$.



Lösung zu Aufgabe 18

Wir zeigen, dass jeder EA, der die Sprache $L = \{x11001y \mid x, y \in \{0,1\}^*\}$ akzeptiert, mindestens 6 Zustände hat.

Sei $A = (Q, \Sigma_{\text{bool}}, \delta, q_0, F)$ ein Automat mit $L(A) = L$. Wenn $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$ für beliebige $w_i, w_j \in \Sigma_{\text{bool}}^*$ gilt, folgt:

$$w_i z \in L \iff w_j z \in L \quad \text{für alle } z \in \Sigma_{\text{bool}}^*.$$

Die Idee des Beweises ist es nun, eine Menge S von Wörtern zu finden, so dass für jeweils zwei Wörter $w_i, w_j \in S$ mit $w_i \neq w_j$ ein z existiert mit $w_i z \in L$ und $w_j z \notin L$ (oder vice versa). Daraus folgt, dass der Automat mindestens $|S|$ (paarweise verschiedene) Zustände benötigt.

Betrachte die Menge $S = \{w_0, \ldots, w_5\}$ mit $w_0 = \lambda, w_1 = 1, w_2 = 11, w_3 = 110, w_4 = 1100, w_5 = 11001$. Dann gilt

(1) $w_5\lambda \in L$, aber für alle anderen w_i gilt $w_i\lambda \notin L$.

(2) $w_4 1 \in L$, aber für alle $i \in \{0,\dots,3\}$ gilt $w_i 1 \notin L$.

(3) $w_3 01 \in L$, aber für alle $i \in \{0,\dots,2\}$ gilt $w_i 01 \notin L$.

(4) $w_2 001 \in L$, aber für alle $i \in \{0,1\}$ gilt $w_i 001 \notin L$.

(5) $w_1 1001 \in L$, aber für $i = 0$ gilt $w_i 001 \notin L$.

Daher benötigt jeder EA, der L akzeptiert, mindestens $|S| = 6$ Zustände.

Alternative Lösung:
Wir zeigen im Folgenden, dass jeder EA mit ≤ 5 Zuständen, der nicht $\emptyset$ akzeptiert, ein Wort mit ≤ 4 Symbolen akzeptieren muss. Daraus folgt dann unmittelbar, dass kein solcher EA für L existieren kann, da L nur Wörter der Länge ≥ 5 enthält.
Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA mit $L(M) \neq \emptyset$ und $|Q| \leq 5$. Sei $w = a_1 \dots a_k$ das kürzeste Wort in $L(M)$. Da $w \in L(M)$, existieren $q_1, \dots, q_k \in Q$ mit $q_k \in F$, so dass

$$(q_0, a_1 \dots a_k) \vdash_M (q_1, a_2 \dots a_k) \vdash_M \dots \vdash_M (q_k, \lambda).$$

Annahme: $k \geq 5$.
Da $|Q| \leq 5$, gibt es dann $i, j \in \{0, \dots, k\}$ mit $i < j$ und $q_i = q_j$. Damit gibt es auch eine akzeptierende Berechnung

$$(q_0, a_1 \dots a_i a_{j+1} \dots a_k) \vdash_M^* (q_k, \lambda),$$

vgl. den Beweis des Pumping-Lemmas (Lemma 3.4 der Vorlesung). Dies ist aber ein Widerspruch zu der Voraussetzung, dass w das kürzeste Wort in $L(M)$ ist.

Lösung zu Aufgabe 19

(a) Angenommen, L ist regulär. Dann sei n_0 die Konstante aus dem Pumping-Lemma. Es sei $w := a^{n_0} bba^{n_0}$. Offenbar ist $w \in L$ und $|w| \geq n_0$, also gibt es gemäss Pumping-Lemma eine Zerlegung $w = yxz$ mit $|yx| \leq n_0$. Also ist $y \in \{a\}^*$ und $x \in \{a\}^+$. Wegen $w \in L$ gilt in Eigenschaft (iii) des Pumping-Lemmas also:

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

folglich ist $yx^0 z = a^{n_0 - |x|} bba^{n_0} \in L$, was aber ein Widerspruch zu $|x| \geq 1$ ist.

Also ist L nicht regulär.

(b) Angenommen, L ist regulär. Dann existiert ein endlicher Automat $M = (Q, \{a, b\}, q_0, \delta, F)$, der L erkennt. Wir geben nun für die Wortfolge

$$(w_j)_{j=0}^{\infty} := (a^j bba^j)_{j=0}^{\infty}$$

die Programmfolge $(P_j)_{j=0}^{\infty}$ von Pseudo-Pascal-Programmen an:

P_j :

$q := \boxed{\hat{\delta}(q_0, a^j bb)}$; `(* Der Kasten ist eine Konstante im Programm. *)`

$e := false$;

$z := \lambda$;

while not e **do**

  if $\hat{\delta}(q, z) \in F$ **then** `(* Hier wird` M `mit konstant viel Code simuliert. *)`

    $e := true$

  else

    $z := z \cdot a$;

$write(z)$;

Offenbar führt jedes Programm P_j genau j Iterationen der Schleife aus, denn dann findet es mit $z = a^j$ dasjenige Wort über $\{a\}$, das, von rechts an $a^j bb$ multipliziert, ein zu L gehörendes Wort ergibt.

Nun gibt es eine Konstante c, so dass für alle j gilt: $K(w_j) \leq |P_j| < c$. Das ist aber ein Widerspruch zur Unendlichkeit von $\{w_j\}_{j=0}^{\infty}$.

Lösung zu Aufgabe 20

L ist nicht regulär.
Angenommen, L ist regulär. Dann sei n_0 die Konstante aus dem Pumping-Lemma. Es sei $w := a^{n_0^3}$. Offenbar ist $w \in L$ und $|w| \geq n_0$, also gibt es gemäss Pumping-Lemma eine Zerlegung $w = yxz$ mit $1 \leq |x| \leq n_0$. Wegen $w \in L$ gilt in Eigenschaft (iii) des Pumping-Lemmas also:

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

folglich ist $yx^2z = a^{n_0^3+|x|} \in L$, also sind n_0^3 und $n_0^3 + |x| > n_0^3$ beides Kubikzahlen. Die kleinste Kubikzahl c mit $c > n_0^3$ ist aber offensichtlich

$$c = (n_0+1)^3 = n_0^3 + 3n_0^2 + 3n_0 + 1 = (n_0^3 + n_0) + \underbrace{3n_0^2 + 2n_0 + 1}_{>0} > n_0^3 + n_0 \geq n_0^3 + |x|,$$

was ein Widerspruch ist.
Also ist L nicht regulär.

Lösung zu Aufgabe 21

L ist nicht regulär.
Angenommen, L ist regulär. Dann existiert ein endlicher Automat $M = (Q, \{a, b\}, q_0, \delta, F)$, der L erkennt. Wir geben nun für die Wortfolge

$$(w_j)_{j=0}^{\infty} := ((ba^{2j+1}b)^2 a)_{j=0}^{\infty}$$

die Programmfolge $(P_j)_{j=0}^{\infty}$ von Pseudo-Pascal-Programmen an, wobei wir annehmen, dass im Alphabet $\{a, b\}$ die Ordnung $a < b$ gilt:

P_j :

$q := \boxed{\hat{\delta}(q_0, ba^{2j+1}b)}$; `(* Der Kasten ist eine Konstante im Programm. *)`

$e := false$;

$z := \lambda$;

while not e **do**

  if $\hat{\delta}(q, z) \in F$ **then** `(*` Hier wird M mit konstant viel Code simuliert. `*)`

    $e := true$

  else

    $z :=$ *kanonischer Nachfolger von* z;

$write(z)$;

Offenbar terminiert jedes Programm P_j, denn es findet mit

$$z = ba^{2j+1}ba \implies ba^{2j+1}b \cdot z = w_j \in L$$

auf jeden Fall einen rechten Faktor, so dass sich ein zu L gehörendes Wort ergibt.

Behauptung: Es ist z das kleinste und damit erste Wort in P_j, das $ba^{2j+1}b \cdot z \in L$ erfüllt. Folglich ist z die Ausgabe von P_j. $(*)$

Nun gibt es eine Konstante c, so dass für alle j gilt: $|P_j| < c$. Folglich gilt: $K(w_j) < c$ für alle j, was ein Widerspruch zur Unendlichkeit von $\{w_j\}_{j=0}^{\infty}$ ist.

Es verbleibt, die Behauptung $(*)$ zu zeigen.
Es sei v das kleinste Wort (bezüglich der kanonischen Ordnung) in $\{a, b\}^*$ mit $ba^{2j+1}bv \in L$. Also gibt es $x, y \in \{a, b\}^+$ mit $ba^{2j+1}bv = xx^Ry$. Also beginnt x mit b, also endet x^R mit b. Da xx^R gerade, $ba^{2j+1}b$ jedoch ungerade Länge hat, folgt, dass $ba^{2j+1}b$ Präfix von x ist. Daher gilt:

$$|x| \geq |ba^{2j+1}b| = 2j + 3 \implies 4j + 7 \leq |xx^Ry| = |ba^{2j+1}bv| = 2j + 3 + |v|,$$

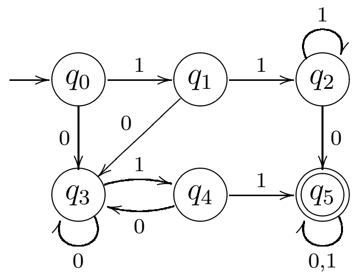
woraus folgt: $|v| \geq 2j + 4 = |z|$, also $|v| = 2j + 4$ (denn $v \leq z$).
Wegen $|xx^Ry| = |ba^{2j+1}bv| = 2j + 3 + |v| = 4j + 7$ und $y \neq \lambda$ folgt:

$$|xx^R| \leq 4j + 6 \implies |x| \leq 2j + 3;$$

da $|x| \geq 2j + 3$ bereits gezeigt wurde, folgt: $|x| = 2j + 3$, also $x = ba^{2j+1}b$. Also ist $x = x^R$ (echtes) Präfix von v, also ist $v \in \{xa, xb\}$ (denn $|v| = |x| + 1$). Mit $xb > xa = z$ folgt $v = z$, womit Behauptung $(*)$ bewiesen ist.

Lösung zu Aufgabe 22

Idee: Es gilt: $L = (\{11\} \cdot \{1\}^* \cdot \{0\} \uplus \{0, 10\}^+ \cdot \{11\}) \cdot \{0, 1\}^*$. (Das Zeichen «$\uplus$» steht für die disjunkte Vereinigung zweier Mengen.)
Mit dieser Idee konstruieren wir den Automaten M:



Wir beweisen nun $L(M) = L$.

$\subseteq$: Es sei $w \in L(M)$. Daher existiert eine akzeptierende Berechnung von M auf w. Diese Berechnung beginnt in q_0, endet in q_5 und führt entweder über q_2 oder q_3.

- Führt die Berechnung über q_2, so sei z das längste Wort, für das es Wörter $u, v \in \{0, 1\}^*$ gibt mit der Eigenschaft, dass:

$$(q_0, w) \vdash_M (q_1, u) \vdash_M (q_2, zv) \vdash_M^* (q_2, v) \vdash_M (q_5, \lambda).$$

  Weil man in einem Berechnungsschritt von (q_2, v) zu (q_5, λ) gelangt, ist offenbar $|v| = 1$, und wegen $\delta(q_2, 1) \neq q_5$ gilt: $v = 0$.
  Wegen $\delta(q_0, 0) \neq q_1$ und $\delta(q_1, 0) \neq q_2$ gilt ausserdem:

$$w = 1u = 11zv = 11z0.$$

  Wir zeigen noch: $z \in \{1\}^*$, denn damit ist 110 Infix von w und somit $w \in L$. Angenommen also, es gibt $x, y \in \{0, 1\}^*$, so dass $z = x0y$. Wegen $(q_2, zv) \vdash_M^* (q_2, v)$ gilt auch: $(q_2, z) \vdash_M^* (q_2, \lambda)$, also gibt es Zustände p_1, p_2, so dass gilt:

$$(q_2, x0y) \vdash_M^* (p_1, 0y) \vdash_M (p_2, y) \vdash_M^* (q_2, \lambda).$$

  Da $\{\hat{\delta}(q_2, t) \mid t \in \{0, 1\}^*\} = \{q_2, q_5\} \ni p_2$ und da $\delta(p_1, 0) = p_2$, aber es keinen Zustand r gibt mit $\delta(r, 0) = q_2$ folgt: $p_2 = q_5$. Also gilt:

$$(q_5, y) \vdash_M^* (q_2, \lambda) \implies q_2 \in \{\hat{\delta}(q_5, s) \mid s \in \{0, 1\}^*\} = \{q_5\},$$

  was ein Widerspruch ist.

- Führt die Berechnung über q_3, so sei z das kürzeste Wort, für das gilt:

$$(q_0, w) \vdash_M^* (q_3, z) \vdash_M^* (q_5, \lambda).$$

Es ist also $w = uz$ für ein Wort $u \in \{0,1\}^*$. Wegen $q_0 \neq q_3$ gilt: $u \neq \lambda$. Weil es keinen Zustand q gibt mit $\delta(q,1) = q_3$ endet u auf das Zeichen 0. Wegen $q_3 \neq q_5$ gilt: $z \neq \lambda$. Gleichzeitig beginnt z mit dem Zeichen 1, denn andernfalls wäre wegen $\delta(q_3, 0) = q_3$ das Suffix von z mit der Länge $|z| - 1$ ein kürzeres Wort als z mit der oben geforderten Eigenschaft. Es gilt aber auch: $|z| > 1$, denn ansonsten wäre $\delta(q_3, 1) = q_5$. Angenommen, das zweite Zeichen von z ist eine 0, also angenommen, es gibt ein Wort $x \in \{0,1\}^*$ mit $z = 10x$. Dann ist wegen $\delta(\delta(q_3,1),0) = q_3$ aber x wiederum ein gegenüber z kürzeres Wort mit der oben geforderten Eigenschaft. Also beginnt z mit 11, und es endet u auf 0, also hat $w = uz$ das Infix 011, also gilt: $w \in L$.

$\supseteq$: Es sei $w \in L$. Also hat w als Infix 011 oder 110 (oder beides). Es seien nun $x, y, z \in \{0,1\}^*$ so gewählt, dass $w = xyz$, $y \in \{011, 110\}$ und x minimale Länge hat.

- Wenn $y = 110$, so gilt: $x \in \{1\}^*$, denn angenommen, x enthält eine 0, dann gibt es Wörter $u \in \{0,1\}^*, v \in \{1\}^*$ mit $x = u0v$, also:

  $$w = xyz = x110z = u0v110z = u011v0z.$$

  Nun ist aber
  $$(x', y', z') := (u, 011, v0z)$$
  eine Zerlegung von $w = x'y'z'$ mit den geforderten Eigenschaften und $|x'| < |x|$, ein Widerspruch zur Minimalität von $|x|$. Also ist $x \in \{1\}^*$. Also gilt:

  $$w = xyz = x110z = 1^{|x|+2}0z = 11x0z.$$

  Wegen $\delta(\delta(q_0, 1), 1) = q_2$ und $\delta(q_2, 1) = q_2$ folgt:

  $$(q_0, w) \vdash_M^* (q_2, 0z) \vdash_M (q_5, z).$$

  Wegen $\delta(q_5, a) = q_5$ für alle $a \in \{0,1\}$ folgt: $(q_0, w) \vdash_M^* (q_5, \lambda)$, also: $w \in L(M)$.
- Wenn $y = 011$, so hat x *nicht* 11 als Infix, denn ansonsten gibt es Wörter $u, v \in \{0,1\}^*$ mit $x = u11v$ und Wörter $v_1 \in \{1\}^*$, $v_2 \in \{0,1\}^* \setminus \{1\}\{0,1\}^*$ mit $v = v_1v_2$, also gilt:

  $$w = xyz = x011z = u11v011z = u11v_1v_2011z = uv_111v_2011z.$$

  Da v_2 nicht mit einer 1 beginnt, beginnt v_20 mit einer 0, es gibt also ein Wort $\tilde{v} \in \{0,1\}^*$ mit $v_20 = 0\tilde{v}$. Also ist

  $$(x', y', z') := (uv_1, 110, \tilde{v}11z)$$

  eine Zerlegung von $w = x'y'z'$ mit den geforderten Eigenschaften und $|x'| < x$, ein Widerspruch zur Minimalität von $|x|$. Also hat x als Infix nicht 11.

Es gilt: $\hat{\delta}(q_0, x) \in \{q_0, q_1, q_3, q_4\}$, denn wäre $\hat{\delta}(q_0, x) \in \{q_2, q_5\}$, so folgte:

$$(q_0, w) \vdash_M^0 (q_0, 11s) \vdash_M (q_1, 1s) \vdash_M (q_2, s) \vdash_M^* (q_2, yz)$$

oder

$$(q_0, w) \vdash_M^* (q_3, 11s) \vdash_M (q_4, 1s) \vdash_M (q_5, s) \vdash_M^* (q_5, yz)$$

für ein $s \in \{0,1\}^*$, woraus folgte, dass 11 ein Infix von x wäre.
Wegen $\delta(q, 0) = q_3$ für alle $q \in \{q_0, q_1, q_3, q_4\}$ folgt:

$$\delta(\delta(\delta(q, 0), 1), 1) = \delta(\delta(q_3, 1), 1) = \delta(q_4, 1) = q_5,$$

mithin:

$$(q_0, w) \vdash_M^* (q_3, 11z) \vdash_M (q_4, 1z) \vdash_M (q_5, z).$$

Wegen $\delta(q_5, a) = q_5$ für alle $a \in \{0, 1\}$ folgt: $(q_0, w) \vdash_M^* (q_5, \lambda)$, also: $w \in L(M)$.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Informatik

Theoretische Informatik

Prof. Dr. J. Hromkovič
Prof. Dr. M. Bläser

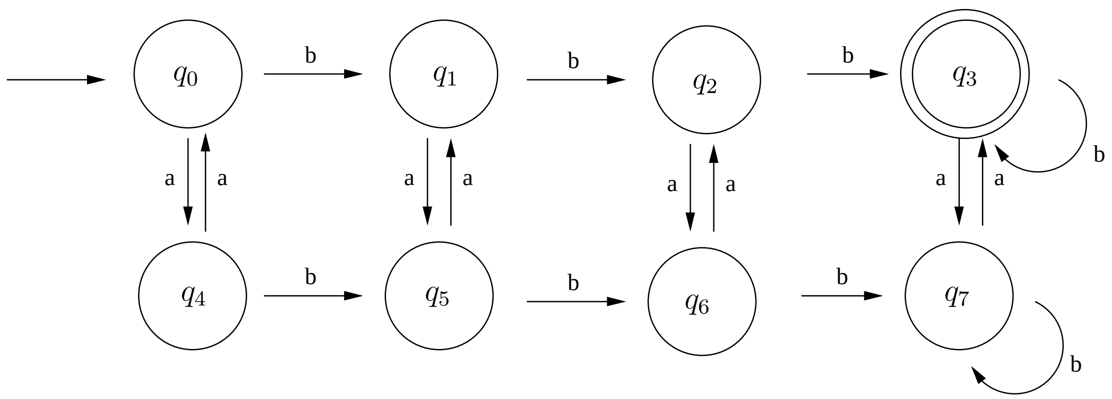
Musterlösungen zur 6. Übung

Zürich, 26. November 2004

Lösung zu Aufgabe 23

Der folgende Automat M akzeptiert die Sprache

$$L = \{x \in \{a,b\}^* \mid |x|_a \text{ ist gerade und } |x|_b \geq 3\}:$$



Um zu beweisen, dass der Automat M minimal ist, zeigen wir nun, dass jeder Automat, der L akzeptiert, mindestens 8 Zustände benötigt.
Sei $A = (Q, \{a,b\}, \delta, q_0, F)$ ein Automat mit $L(A) = L$.
Annahme: $|Q| < 8$.
Betrachte die Menge $S := \{\lambda, a, b, ab, bb, bba, bbb, bbba\}$. Da $|S| = 8 > |Q|$, existieren $w_i, w_j \in S$ mit $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$. Es gilt aber für jeweils zwei Wörter $w_i, w_j \in S$ mit $w_i \neq w_j$ auch $|w_i|_a \neq |w_j|_a$ oder $|w_i|_b \neq |w_j|_b$, was wir wie folgt nutzen werden.

1.Fall ($|w_i|_a \neq |w_j|_a$): Sei ohne Einschränkung der Allgemeinheit $|w_i|_a = 0$. Dann ist $w_i bbb \in L$ aber $w_j bbb \notin L$, da die Anzahl der as in w_j und daher auch in $w_j bbb$ ungerade ist.

2.Fall ($|w_i|_b \neq |w_j|_b$): Sei ohne Einschränkung der Allgemeinheit $\ell := |w_i|_b > |w_j|_b$. Falls $|w_i|_a = 0$, ist $w_i b^{3-\ell} = bbb \in L$, aber $w_j b^{3-\ell} \notin L$, da weniger als drei bs in $w_j b^{3-\ell}$ vorkommen. Falls $|w_i|_a = 1$, betrachte analog $w_i b^{3-\ell} a$ und $w_j b^{3-\ell} a$.

Dies ergibt einen Widerspruch, und daher folgt $|Q| \geq 8$.

Lösung zu Aufgabe 24

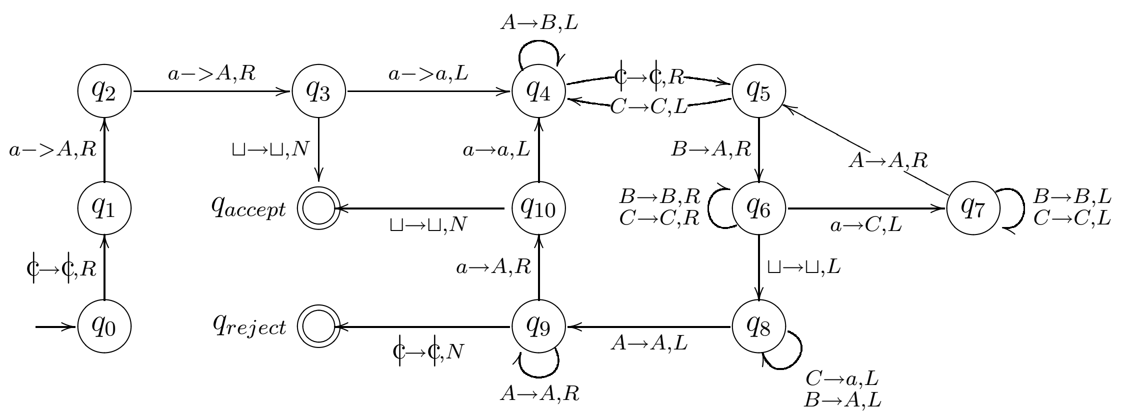
Idee: Bei einem Eingabewort a^p genügt es, für alle Zahlen $2 \leq j < p$ zu prüfen, ob j ein Teiler von p ist. Wird eine solche Zahl gefunden, ist p keine Primzahl, ansonsten ist p Primzahl.

Um die Zahl j darzustellen, werden die ersten j Zeichen der Eingabe a^p durch As dargestellt, es steht dann also $A^j a^{p-j}$ auf dem Band. Um die Teilbarkeit durch j zu prüfen, werden die nachfolgenden as jeweils durch j viele Cs ersetzt, auf dem Band werden also nacheinander die Wörter $A^j C^{jk} a^{p-j(k+1)}$ für $1 \leq k \leq \lfloor p/j \rfloor - 1$ generiert. Wird hierbei der rechte Rand der Eingabe überschritten (also ein $\sqcup$ gelesen), während die nächsten j vielen Cs generiert werden, ist p nicht durch j teilbar. Wird der rechte Rand hingegen überschritten, nachdem (k-mal) j viele Cs generiert wurden, ist p durch j teilbar, also keine Primzahl. Ist p nicht durch j teilbar, müssen alle Cs wieder durch as ausgetauscht werden, und das nächste a (ganz links) wird durch ein A ausgetauscht, so dass nunmehr $A^{j+1}a^{p-j-1}$ auf dem Band steht. Stehen nur noch As auf dem Band, ist also $j = p - 1$, und p war durch keine Zahl j mit $2 \leq j \leq p - 1$ teilbar, also ist p Primzahl.

Es sei $M := (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ mit

$$\Sigma := \{a\} \text{ und}$$
$$\Gamma := \{a, ¢, \sqcup, A, B, C\}$$

die durch folgendes Diagramm beschriebene Turing-Maschine:



(Wie auch ansonsten bei Turing-Maschinen üblich verstehen sich alle nicht eingezeichneten Transitionen als zu q_{reject} übergehend. Die einzige hier eingezeichnete zu q_{reject} übergehende Kante wird durchlaufen, wenn ein Teiler von p gefunden wurde.)

Wegen $\Sigma = \{a\}$ ist offensichtlicherweise $L(M) \subseteq \{a\}^*$. Dass tatsächlich $L(M) = \{a^p \mid p \text{ ist Primzahl}\}$ gilt, begründen wir mit folgenden Invarianten (die Eingabe sei stets a^p):

- Wann immer q_4 verlassen wird, so geschieht dies am Bandanfang, und es wird in eine Konfiguration $¢q_5 B^j a^{p-j}$ übergegangen.

- Wann immer q_5 in der Konfiguration x betreten wird, so folgt aus $|x|_A + |x|_B = j$ bereits: $|x|_C = jk$ für ein $k \in \mathbb{N}$ mit $k \leq \lfloor p/j \rfloor - 1$, und es gilt sogar:

$$x = \cent A^\ell q_5 B^{j-\ell} C^{jk} a^{p-j(k+1)}.$$

- Wann immer q_8 (von q_6 aus) betreten wird, so hat die Bandinschrift die Form $A^\ell B^{j-\ell} C^{p-j}$ für $\ell = (p \bmod j) + 1$. Wird q_8 (hin zu q_9) verlassen, so hat die Bandinschrift also die Form $A^j a^{p-j}$, und der Lese-/Schreibkopf befindet sich auf dem rechtesten A, bevor er um eine Position nach links verrückt wird.

- Wird q_9 betreten, nachdem in q_8 noch $\ell = 1$ gegolten hatte, stand also zuvor in Zustand q_8 der Lese-/Schreibkopf mit dem rechtesten A auf dem ersten Bandzeichen, so wird von q_9 zu q_{reject} übergegangen und die Eingabe somit verworfen. Dies ist korrekt, weil $\ell = 1$ bedeutet, dass p durch j teilbar, also keine Primzahl ist. Andernfalls wird das rechteste A der Bandinschrift $A^j a^{p-j}$ gesucht und noch von dort aus der Lese-/Schreibkopf um eine Position nach rechts verrückt. Hier steht auf jeden Fall ein a, welches die Turing-Maschine zu einem A umschreibt, um den Schritt $\ll j \leftarrow j + 1 \gg$ durchzuführen.

- In q_{10} ist nach diesem Schritt der Lese-/Schreibkopf um eine weitere Position nach rechts verrückt worden, um zu überprüfen, ob hier ein $\sqcup$ steht. Ist dies der Fall, so gilt für das neue j nun: $j = p$, somit war p durch kein $2 \leq j < p$ teilbar, also ist p Primzahl. Steht hier noch ein weiteres a, so wird q_4 betreten und (für das neue j) die Konfiguration

$$\cent A^{j-1} q_4 A a^{p-j}$$

erreicht.

Lösung zu Aufgabe 25

(a) Es seien K, L, M, N und P fünf weitere (auf 0 initialisierte) Register. Das Programm

$$
\begin{array}{rl}
 & K := I; \\
 & M := I; \\
 & L := J; \\
 & N := J; \\
3: & \texttt{if } L = 0 \texttt{ then goto } 5; \\
 & \texttt{if } K = 0 \texttt{ then goto } 4; \\
 & \texttt{if } M = 0 \texttt{ then goto } 6; \\
 & \texttt{if } N = 0 \texttt{ then goto } 8; \\
 & K := K - 1; \\
 & L := L - 1; \\
 & M := M + 1; \\
 & N := N + 1; \\
 & \texttt{if } P = 0 \texttt{ then goto } 3; \\
4: & K := K - 1; \\
 & \texttt{if } P = 0 \texttt{ then goto } 7; \\
5: & L := L - 1; \\
 & \texttt{if } P = 0 \texttt{ then goto } 7; \\
6: & M := M + 1; \\
7: & \texttt{if } K = 0 \texttt{ then goto } 1; \\
 & \texttt{if } L = 0 \texttt{ then goto } 2; \\
 & \texttt{if } M = 0 \texttt{ then goto } 2; \\
 & \texttt{if } N = 0 \texttt{ then goto } 1; \\
 & K := K - 1; \\
 & L := L - 1; \\
 & M := M + 1; \\
8: & N := N + 1; \\
 & \texttt{if } P = 0 \texttt{ then goto } 7;
\end{array}
$$

leistet das Gewünschte: Man beachte, dass I und/oder J auch negative Zahlen enthalten können; daher wird zunächst in der über Sprungmarke 3 realisierten Schleife im ℓ-ten Schritt in (K, L, M, N) das Quadrupel $(I-\ell, J-\ell, I+\ell, J+\ell)$ generiert, für das offensichtlich schliesslich entweder eine ($|I| \neq |J|$) oder zwei ($|I| = |J|$) Komponenten gleich Null werden. Wir führen eine Fallunterscheidung durch:

$|I| \neq |J|$: Nach $\ell_0 := \min\{|I|, |J|\}$ Schritten nimmt das erste Mal eine Komponente den Wert Null an. An der entsprechenden Marke (d.h. 4, 5, 6 oder 8) wird diese Komponente um eine Position weitergezählt, und sie wird somit im weiteren Verlauf des Programms nie wieder den Wert Null annehmen. Nach

$$\ell_1 := \max\{|I|, |J|\} - \ell_0$$

Schritten der über Sprungmarke 7 realisierten Schleife nimmt nun eine zweite Komponente den Wert Null an. Ist dies die

- Komponente K, so ist $I > 0$, und entweder ($-I < J < 0$) hat N oder ($0 \leq J < I$) es hat L als erste Komponente den Wert Null angenommen. Also ist $I \geq J$, und es wird zu Marke 1 gesprungen.
- Komponente L, so ist $J > 0$, und entweder ($0 \leq I < J$) hat K oder ($-J < I < 0$) es hat M als erste Komponente den Wert Null angenommen. Also ist $I < J$, und es wird zu Marke 2 gesprungen. *Beachte:* Bei Marke 3 ist es erforderlich, dass zunächst L, erst dann K mit Null verglichen wird.
- Komponente M, so ist $I < 0$, und entweder ($0 \leq J < -I$) hat L oder ($I < J < 0$) es hat N als erste Komponente den Wert Null angenommen. Also ist $I < J$, und es wird zu Marke 2 gesprungen.
- Komponente N, so ist $J < 0$, und entweder ($0 \leq I < -J$) hat K oder ($J < I < 0$) es hat M als erste Komponente den Wert Null angenommen. Also ist $I \geq J$, und es wird zu Marke 1 gesprungen.

$I = J \geq 0$: Nach $I = J$ Sprüngen zu Marke 3 enthalten K und L den Wert Null, somit wird zu Marke 5 gesprungen, dort noch L dekrementiert und bei Marke 7 nun zu Marke 1 gesprungen, was korrekt ist, denn $I \geq J$.

$I = J < 0$: Nach $-I = -J$ Sprüngen zu Marke 3 enthalten M und N den Wert Null, somit wird zu Marke 6 gesprungen, dort noch M inkrementiert und bei Marke 7 (aber erst beim Vergleich von N) zu Marke 1 gesprungen, was korrekt ist, denn $I \geq J$.

$0 < I = -J$: Nach $I = -J$ Sprüngen zu Marke 3 gilt: $(K, L, M, N) = (0, -2I, 2I, 0)$. Also wird zu Marke 4 gesprungen, nun gilt: $(K, L, M, N) = (-1, -2I, 2I, 0)$, also wird bei Marke 7 durch den Vergleich $N = 0$ zu Marke 1 gesprungen.

$0 < J = -I$: Nach $J = -I$ Sprüngen zu Marke 3 gilt: $(K, L, M, N) = (-2J, 0, 0, 2J)$. Also wird zu Marke 5 gesprungen, nun gilt: $(K, L, M, N) = (-2J, -1, 0, 2J)$, also wird bei Marke 7 durch den Vergleich $M = 0$ zu Marke 2 gesprungen.

Bemerkung: Lösungen, die die Nichtnegativität von I und J vorausgesetzt haben, werden mit voller Punktzahl bewertet. Die Register I und J dürfen jedoch nicht zerstört werden.

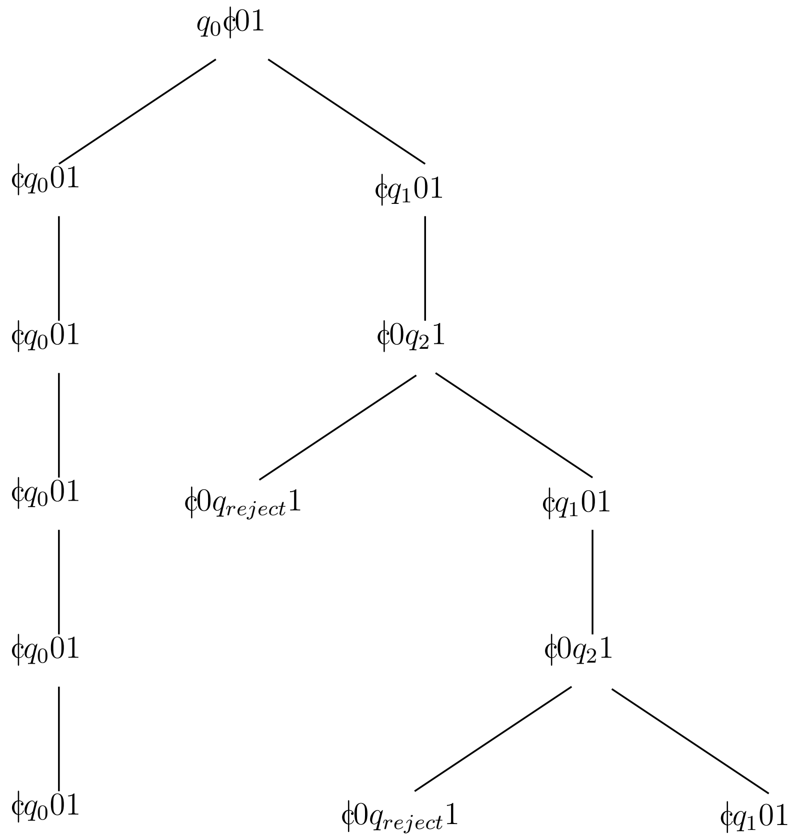
(b) Es seien c und d zwei weitere Register. Die beiden Programme

$c := a;$	$c := a;$
$d := b;$	$d := b;$
$1:$ `if` $d = 0$ `then STOP`	$1:$ `if` $d = 0$ `then STOP`
$d := d - 1;$	$d := d - 1;$
$c := c + 1;$	$c := c - 1;$
`goto` $1;$	`goto` $1;$

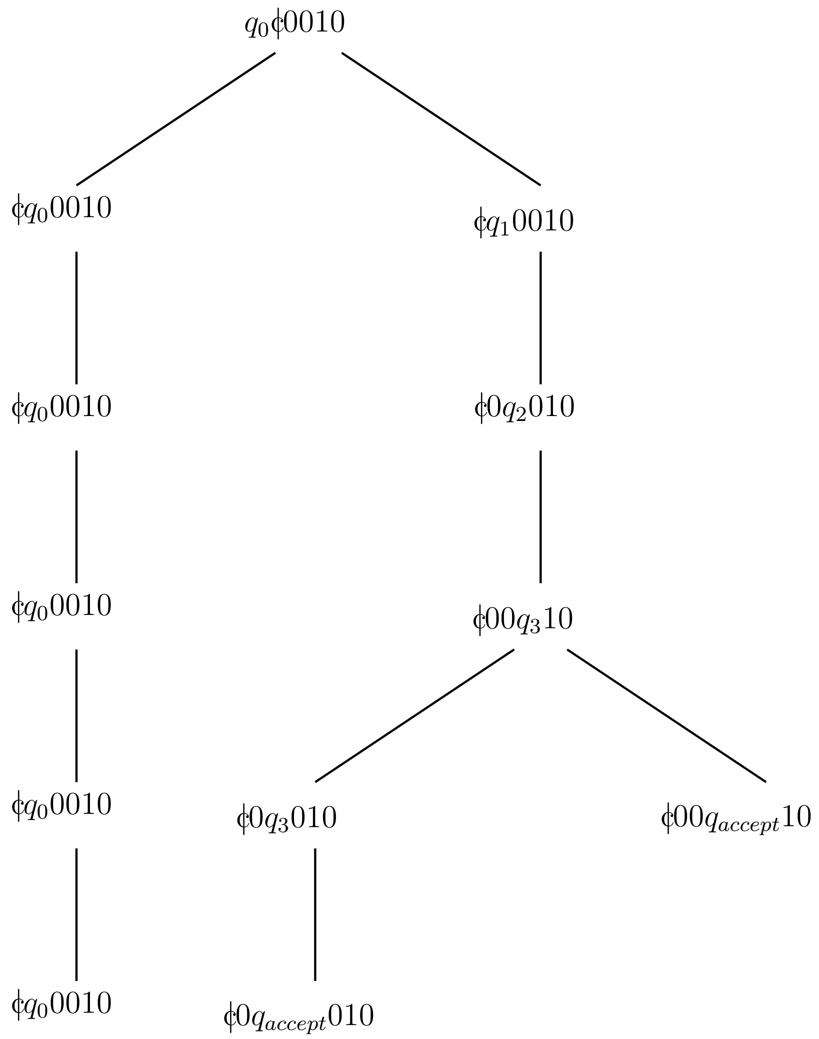
leisten offensichtlich das (jeweils) Gewünschte.

Lösung zu Aufgabe 26

(a) Für $x = 01$ erhalten wir folgenden Berechnungsbaum $T_A(01)$:



Für $x = 0010$ erhalten wir folgenden Berechnungsbaum $T_A(0010)$:



(b) Führt M im Zustand q_0 (nichtdeterministisch) irgendeine andere Transition als den Übergang von q_0 nach q_1 (durch Lesen von ¢ und Verschieben des Lese-/Schreibkopfs um eine Position nach rechts) aus, so verbleibt sie dauerhaft in q_0, denn der Kopf wird nie wieder nach links bewegt werden, und die einzige Transition aus q_0 heraus erfordert das Lesen des ¢-Symbols, das nur am Bandanfang steht.

Offensichtlich kann q_1 nur durch Lesen einer 0 am Wortanfang verlassen werden, also haben alle $w \in L(M)$ den Anfangsbuchstaben 0. Folgt nun eine 1, so darf M nach q_{reject} übergehen; folgt kein Zeichen, so gibt es keine passende Transition. Also muss zum Akzeptieren eines Wortes auch das zweite Zeichen eine 0 sein. Also beginnen alle $w \in L(M)$ mit dem Präfix 00.

Nach Lesen des Präfixes 00 gerät M nach q_3. Von q_3 aus gelangt man durch das Lesen mindestens eines weiteren Symbols (0 oder 1) in den Zustand q_{accept}. Also erkennt M die Sprache
$$L(M) = \{w \in \{0,1\}^* \mid w = 00x, |x| \geq 1\}.$$

Lösung zu Aufgabe 27

Wir führen die Behauptung zurück auf das (nicht-modifizierte) Pumping-Lemma und behaupten:

$$\text{Ist } L \text{ regulär, so auch } L^R := \{w^R \mid w \in L\}. \tag{$*$}$$

Also gilt das (nicht-modifizierte) Pumping-Lemma für L^R, und es gibt eine Konstante $n_0' \in \mathbb{N}$, so dass sich jedes Wort $w \in \Sigma^*$ mit $|w| \geq n_0'$ in drei Teile y', x' und z' zerlegen lässt, d.h. $w = y'x'z'$, wobei

1. $|y'x'| \leq n_0$,
2. $|x'| \geq 1$ und
3. entweder $\{y'(x')^k z' \mid k \in \mathbb{N}\} \subseteq L^R$ oder $\{y'(x')^k z' \mid k \in \mathbb{N}\} \cap L^R = \emptyset$.

Wir wählen nun $n_0 := n_0'$ und ermitteln für alle Worte $w \in \Sigma^*$ mit $|w| = |w^R| \geq n_0 = n_0'$ die Aufteilung von
$$w^R = y'x'z'$$
und wählen
$$\begin{aligned} y &:= (z')^R \\ x &:= (x')^R \\ z &:= (y')^R \end{aligned}$$

Offensichtlich folgen die Bedingungen (1) und (2) des modifizierten Pumping-Lemmas direkt aus den Bedingungen (1) und (2) des nicht-modifizierten Pumpung-Lemmas. Bedingung (3) folgt mit
$$y'(x')^k z' = z^R (x^R)^k y^R = z^R (x^k)^R y^R = (yx^k z)^R \quad \text{für alle } k \in \mathbb{N}$$
aus der Definition von L^R bzw. folgender Äquivalenz:
$$y'(x')^k z' \in L^R \iff yx^k z \in L \quad \text{für alle } k \in \mathbb{N}$$

Es verbleibt, Behauptung $(*)$ zu zeigen: Dazu sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA mit $L(M) = L$. Wir konstruieren den nichtdeterministischen Automaten $M^R = (Q \uplus \{q_I\}, \Sigma, \delta^R, q_i, F^R)$ mit

$$\begin{aligned}
\delta^R(q,a) &:= \{p \in Q \mid \delta(p,a) = q\} && \text{für alle } q \in Q,\ a \in \Sigma \\
\delta^R(q_I,a) &:= \{p \in Q \mid \delta(p,a) \in F\} && \text{für alle } a \in \Sigma \\
F^R &:= \begin{cases} \{q_0\} & ;\ \lambda \notin L \\ \{q_0, q_I\} & ;\ \lambda \in L \end{cases}
\end{aligned}$$

Weil für alle (deterministischen wie nichtdeterministischen) endlichen Automaten $M = (Q, \Sigma, q_0, \delta, F)$ gilt, dass $\lambda \in L(M) \iff q_0 \in F$, genügt es, für alle $w \in \Sigma^+$ zu zeigen, dass $w \in L(M) \iff w^R \in L(M^R)$ gilt. Für ein $n \geq 1$ Zeichen langes Wort $w = w_1 w_2 \ldots w_n$ gilt aber:

$$\begin{aligned}
& w \in L(M) \\
\iff & \text{es gibt } q_1, \ldots, q_n \in Q \text{ so dass} \\
& \qquad \text{für alle } 0 < i \leq n \text{ gilt: } \delta(q_{i-1}, w_i) = q_i \text{ und } q_n \in F \\
\iff & \text{es gibt } q_1, \ldots, q_n \in Q \text{ so dass} \\
& \qquad \text{für alle } 0 < i < n \text{ gilt: } q_{i-1} \in \delta^R(q_i, w_i) \text{ und } \delta(q_{n-1}, w_n) \in F \\
\iff & \text{es gibt } q_1, \ldots, q_n \in Q \text{ so dass} \\
& \qquad \text{für alle } 0 < i < n \text{ gilt: } q_{i-1} \in \delta^R(q_i, w_i) \text{ und } q_{n-1} \in \delta^R(q_I, w_n) \\
\iff & \text{es gibt } q_1, \ldots, q_{n-1} \in Q \text{ so dass} \\
& \qquad \text{für alle } n > i > 0 \text{ gilt: } (q_i, w_1 \ldots w_{i-1}) \vdash_{M^R} (q_{i-1}, w_1 \ldots w_{i-1}) \\
& \qquad \text{und } (q_I, w_n \ldots w_1) \vdash_{M^R} (q_{n-1}, w_{n-1} \ldots w_1) \\
\iff & (q_I, w_n \ldots w_1) \vdash_{M^R} (q_0, \lambda) \\
\iff & w^R = w_n \ldots w_1 \in L(M^R),
\end{aligned}$$

denn q_0 ist für nichtleere Wörter der einzige Endzustand in M^R.

Lösung zu Aufgabe 28

Es sei $L := \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$. Für diese Sprache gilt das vereinfachte Pumping-Lemma mit $n_0 := 2$, denn sei $w \in \{a,b\}^*$ mit $|w| \geq 2$. Besitzt w weder das Infix ab noch das Infix ba, so gilt:

$$w \in \{a\}^+ \cup \{b\}^+,$$

und wir wählen

$$\begin{aligned}
x &:= \alpha \\
y &:= \lambda \\
z &:= \alpha^{|w|-1},
\end{aligned}$$

wobei $\alpha \in \{a, b\}$ dasjenige Zeichen ist, aus dem w besteht. Offensichtlich ist $x \neq \lambda$ und $yx^kz = \alpha^{|w|+k-1}$ für kein k in L enthalten, also gilt das vereinfachte Pumping-Lemma in diesem Fall.

Besitzt w hingegen ab oder ba als Infix, hat w also die Form yxz, wobei $x \in \{ab, ba\}$ dieses Infix sei, so gilt: $x \neq \lambda$, und es gilt auch:

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

denn:

$$\begin{aligned}
&|yx^kz|_a - |yx^kz|_b \\
=&(|y|_a + |x^k|_a + |z|_a) - (|y|_b + |x^k|_b + |z|_b) \\
=&|yz|_a - |yz|_b + k \cdot \underbrace{|x|_a}_{=1} - k \cdot \underbrace{|x|_b}_{=1} \\
=&(|w|_a + (k-1)) - (|w|_b + (k-1)) \\
=&|w|_a - |w|_b,
\end{aligned}$$

und offensichtlich sind w und alle Wörter yx^kz genau dann in L enthalten, wenn diese Zahl 0 ist.

Kann man dennoch beweisen, dass $L \notin \mathcal{L}(EA)$ gilt, *ohne* die Verschärfung des aus dem Buch bekannten Pumping-Lemmas zu nutzen? Die Antwortet lautet ja, denn bekanntlich ist der Schnitt zweier regulärer Sprachen wieder regulär. Es ist aber:

$$L \cap \underbrace{\{a\}^*\{b\}^*}_{\text{offensichtlich regulär}} = \{a^nb^n \mid n \in \mathbb{N}\}$$

bekanntlich irregulär, also ist L auch nicht regulär.

Theoretische Informatik
Formale Sprachen, Berechenbarkeit,
Komplexitätstheorie, Algorithmik, Kommunikation und
Kryptographie
Hromkovič, J.
2011, II, 415 S. 87 Abb., Softcover
ISBN: 978-3-8348-0650-5