Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Werner Meixner, Dr. Alexander Krauss

Sommersemester 2010
Lösungsblatt 8
28. Juni 2010

Einführung in die Theoretische Informatik

Hausaufgabe 1 (5 Punkte)

Für Zwecke dieser Aufgabe nennen wir einen deterministischen Kellerautomaten $K = (Q, \Sigma, \Delta, \delta, q_0, Z_0, F)$, dessen Übergangsfunktion δ so beschaffen ist, dass der Kellerinhalt nie verändert wird, einen ϵ-DFA. Sei $L(K)$ die Sprache, die von einem ϵ-DFA K mit Endzuständen akzeptiert wird.

Geben Sie ein direktes (nicht über ϵ-NFA) Verfahren an, das zu einem beliebigen ϵ-DFA K einen deterministischen endlichen Automaten A definiert, der die Sprache $L(K)$ erkennt!

Lösungsvorschlag

Da der Keller bei Berechnungen nie verändert wird, besteht der Kellerinhalt stets genau aus dem Zeichen Z_0. Entsprechend können die überflüssigen Argumente aus der Funktionalität der Übergangsfunktion δ entfernt und so der Funktionalität der Übergangsfunktion bei endlichen Automaten angepasst werden. Wenn wir auch die überflüssigen Symbole entfernen, dann können wir zu K eindeutig einen Automaten $K_e = (Q, \Sigma, \delta_e, q_0, F)$ definieren für alle $a \in \Sigma \cup \{\epsilon\}$ und $q, p \in Q$ mit

$$\delta_e(q, a) = p, \quad \text{falls} \quad \delta(q, a, Z_0) = \{(p, Z_0)\}\,.$$

Würde K_e keine ϵ-Übergänge enthalten, dann wäre K_e bereits im Wesentlichen ein deterministischer endlicher Automat, der $L(K)$ erkennt.
Falls K_e ausgehend von einem Zustand q ein Eingabezeichen $a \in \Sigma$ verarbeitet, dann geschieht dies, indem er eine, möglicherweise leere, Folge ϵ^n liest, dabei eine Folge von Zuständen durchläuft, und schließlich in einem Zustand ankommt, indem die Eingabe von a definiert ist. Anschließend wird a gelesen und in einen Zustand p übergegangen.
Entsprechend definieren wir eine Übergangsfunktion δ' wie folgt. Es sei $a \in \Sigma$ und n die kleinste Zahl aus $\mathbb{N}_0$, so dass $\hat{\delta_e}(q, \epsilon^n a)$ definiert ist, dann definieren wir

$$\delta'(q, a) = \hat{\delta_e}(q, \epsilon^n a)\,.$$

Wir erinnern hier an die Bemerkung in Blatt 5, Tutoraufgabe 1 zu den ϵ-Folgen und Ausdrücken, wenn ϵ als Buchstabe behandelt wird. δ' ist eine partiell definierte Funktion. Das Einlesen geschieht mit δ' vollständig und ϵ-frei. Damit können wir bereits den gesuchten Automaten wie folgt ansetzen.

$$A' = (Q, \Sigma, \delta', q_0', F')\,,$$

wobei eventuell $q_0' = \epsilon^n p_0$ für geeignetes n gesetzt wird. Wir müssen nun die Menge F' der Endzustände bestimmen.

Dazu müssen wir berücksichtigen, dass der Kellerautomat, nachdem er ein Zeichen gelesen hat, mit ϵ-Übergängen zu einem Endzustand laufen und akzeptieren kann. Dies bedeutet, dass dann der Zustand p, der bei Einlesen eines Zeichens angenommen wird, ebenfalls ein akzeptierender Zustand ist, sozusagen modulo nachfolgende ϵ. Deshalb definieren wir

$$F' = \{p \in Q \mid p \in F \vee (\exists n \in \mathbb{N})[\hat{\delta}_e(p, \epsilon^n) \in F]\,.$$

Schließlich totalisieren wir noch die partiell definierte Funktion δ' und erhalten als Erweiterung die Funktion δ'', indem wir für Argumentpaare, für die δ' nicht definiert ist, als Wert einen neuen Zustand f_0 festsetzen mit $Q' = Q \cup \{f_0\}$. Wir fassen das Ergebnis zusammen:

$$A = (Q', \Sigma, \delta'', q_0', F')\,.$$

Hausaufgabe 2 (5 Punkte)

Wir bezeichnen die Anzahl der Vorkommen eines Buchstaben x in einem Wort w mit $|w|_x$. Sei $\Sigma = \{a, b\}$ und

$$L = \{w \in \Sigma^* \mid w = uav, u, v \in \Sigma^*, |u|_a = |v|_b\}\,.$$

1. Geben Sie eine Grammatik $G = (V, \Sigma, P, S)$ an mit $L_G = L$.
2. Leiten Sie nach Satz der Vorlesung systematisch einen zu G äquivalenten Kellerautomaten K her.

Lösungsvorschlag

1. Seien $V = \{S, X, Y, A, A_0, B, B_0\}$ und P gegeben durch

$$\begin{array}{lcl} S & \to & XSY \mid B_0AA_0\,, \\ X & \to & B_0A\,, \\ Y & \to & BA_0\,, \\ A_0 & \to & aA_0 \mid \epsilon\,, \\ B_0 & \to & bB_0 \mid \epsilon\,, \\ A & \to & a\,, \\ B & \to & b\,. \end{array}$$

   Dann gilt $L_G = L$.

2. Wir konstruieren $K = (Q, \Sigma, \Gamma, q_0, S, \delta, \emptyset)$ mit $L_\epsilon(K) = L$ nach Satz 3.56 der Vorlesung. K wird als nichtdeterministischer Kellerautomat mit nur einem Zustand q_0 konstruiert.

   Seien $Q = \{q_0\}$ und $\Gamma = V$. Alle Produktionen von G haben bereits die in Satz 3.56 verlangte Form $C \to cD_1 \ldots D_k$ mit $c \in \Sigma \cup \{\epsilon\}$ und $D_i \in V$, wobei $k = 0$ zugelassen ist und dann für terminale bzw. ϵ-Produktionen steht.

   Die Übergangsfunktion δ wird für alle $c \in \Sigma \cup \{\epsilon\}$ und $D \in V$ definiert durch

$$\delta(q_0, c, D) := \{(q_0, \gamma) \mid (D \to c\gamma) \in P\}\,.$$

Wir listen die einzelnen Übergänge wie folgt auf.

$$\begin{aligned}
&\delta(q_0,\epsilon,S) \ni (q_0,XSY), && \delta(q_0,\epsilon,S) \ni (q_0,B_0AA_0),\\
&\delta(q_0,\epsilon,X) \ni (q_0,B_0A), && \delta(q_0,\epsilon,Y) \ni (q_0,BA_0),\\
&\delta(q_0,a,A_0) \ni (q_0,A_0), && \delta(q_0,\epsilon,A_0) \ni (q_0,\epsilon),\\
&\delta(q_0,b,B_0) \ni (q_0,B_0), && \delta(q_0,\epsilon,B_0) \ni (q_0,\epsilon),\\
&\delta(q_0,a,A) \ni (q_0,\epsilon), && \delta(q_0,b,B) \ni (q_0,\epsilon).
\end{aligned}$$

Hausaufgabe 3 (5 Punkte)

1. Geben Sie eine deterministische Turingmaschine A an, die für ein Eingabewort $w \in \{0,1\}^+$ eine Berechnung durchführt, so dass am Ende der Berechnung das Wort ww^Rw^R auf dem Band steht und der Kopf im Endzustand auf dem Anfang von ww^Rw^R steht. Kommentieren Sie Ihre Lösung durch eine informelle Beschreibung Ihrer Lösungsidee.

2. Geben Sie eine Turingmaschine B an, die die Sprache $L = \{ww^Rw^R \mid w \in \{0,1\}^*\}$ akzeptiert. Kommentieren Sie Ihre Lösung durch eine informelle Beschreibung Ihrer Lösungsidee.

Lösungsvorschlag

1. *Lösungsidee*: Wir realisieren 3 Phasen der Berechnung. Die ersten beiden Phasen bestehen darin, dass w in umgekehrter Reihenfolge jeweils an das rechte Ende des Bandinhalts kopiert wird. Dies geschieht durch je einen Aufruf einer Turingmaschine T. In der dritten und letzten Phase werden sämtliche Hilfszeichen in Zeichen des Alphabets Σ zurückverwandelt. Dies geschieht durch Aufruf einer Turingmaschine S. Die Maschine A realisiert die Kontrollstruktur der Aufrufe.

   Insgesamt realisieren wir die Möglichkeit, alle Eingabezeichen aus $a \in \Sigma$ als a_o und $\hat{a}$ zu markieren und fassen diese markierten Zeichen in den Bandalphabeten Γ_o und $\widehat{\Gamma}$ zusammen. Außerdem benötigen wir noch 3 Hilfszeichen $\{1,2,3\}$ zur Kennzeichnung der Aufrufphasen am (linken) Anfang des Eingabewortes w.

   Sei nun $A = (Q,\Sigma,\Gamma,\delta,q_0,\square,\{q_f\})$ mit $Q = \{q_0,q_1,\ldots,q_f\} \cup Q_T \cup Q_S$ und $\Gamma = \{\square\} \cup \Gamma_o \cup \widehat{\Gamma}$, wobei wir $t_0, t_f \in Q_T$ und $s_0, s_f \in Q_S$ für die entsprechenden Start und Endzustände annehmen.

   δ besteht aus den Übergangsfunktionen δ_T und δ_S der Maschinen T und S, sowie der folgenden Kontrollstruktur von A. Alle nicht definierten Übergänge können mit einem Fangzustand q_∞ abgefangen werden. Wir lassen δ aber als partiell definiert zu.

Übergang	Bereich	Kommentar
$\delta(q_0, x) \rightarrow (q_0, x, L)$	$x \in \Sigma \cup \widehat{\Gamma}$	Phasenüberprüfung
$\delta(q_0, \square) \rightarrow (t_0, 1, R)$		1. Aufruf Maschine T
$\delta(q_0, 1) \rightarrow (t_0, 2, R)$		2. Aufruf Maschine T
$\delta(q_0, 2) \rightarrow (s_0, 3, R)$		2. Aufruf Maschine S
$\delta(q_0, 3) \rightarrow (q_f, \square, R)$		Ende.
$\delta(t_f, x) \rightarrow (q_0, x, N)$	$x \in \Sigma$	Nächste Phase.
$\delta(s_f, x) \rightarrow (q_0, x, N)$	$x \in \Sigma$	Nächste Phase.

Beim Aufruf der Maschine T steht ein Wort wv_o mit dem Eingabewort w und einem als Ausgabe markierten Wort $v_o \in \Gamma_o^*$ auf dem Band. Der Kopf der Maschine steht auf dem ersten Buchstaben von w.

Die Maschine T erzeugt nun $wv_ow_o^R$ und endet mit Kopf wieder auf dem Anfang von w. Dabei ist w_o^R gleich dem markierten Wort w^R.

Im Kopiervorgang wird ein gelesenes Zeichen a in den Zustand t durch $t^a \in Q_T$ für alle $a \in \Sigma$ kodiert.

Übergang	Bereich	Kommentar
$\delta(t_0, a) \rightarrow (t_0, a, R)$	$a \in \Sigma$	zum rechten Ende der Eingabe.
$\delta(t_0, v) \rightarrow (t_1, v, L)$	$v \in \Gamma_o \cup \{\square\}$	nach links zum Eingabeende
$\delta(t_1, a) \rightarrow (t^a, \hat{a}, R)$	$a \in \Sigma$	a lesen und speichern
$\delta(t^a, v) \rightarrow (t^a, v, R)$	$v \in \Sigma \cup \Gamma_o$	zum rechten Wortende
$\delta(t^a, \square) \rightarrow (t_2, a_o, L)$	$a \in \Sigma$	a_0 neben Wortende schreiben
$\delta(t_2, v) \rightarrow (t_2, v, L)$	$v \in \Sigma \cup \Gamma_o$	nach links bis $\hat{a}$ laufen
$\delta(t_2, \hat{a}) \rightarrow (t_3, a, L)$	$a \in \Sigma$	Markierung entfernen
$\delta(t_3, a) \rightarrow (t_1, a, N)$	$a \in \Sigma$	Rücksprung
$\delta(t_3, z) \rightarrow (t_f, z, R)$	$\in \{1, 2, 3\}$	Ende.

Es folgt die Maschine S, die alle o-Markierungen entfernt.

Übergang	Bereich	Kommentar
$\delta(s_0, x) \rightarrow (s_0, x, R)$	$x \in \Sigma$	Eingabe nach rechts überfahren
$\delta(s_0, x_0) \rightarrow (s_0, x_o, R)$	$x \in \Gamma_o$	Markierung löschen
$\delta(s_0, \square) \rightarrow (s_1, \square, L)$		Beginn Rücklauf
$\delta(s_1, x) \rightarrow (s_1, x, L)$	$x \in \Sigma$	Eingabe nach links überfahren
$\delta(s_1, z) \rightarrow (s_f, z, L)$	$z \in \{1, 2, 3\}$	Ende.

2. Sei $\Sigma = \{0, 1\}$. Wir konstruieren eine nichtdeterministische Turingmaschine $B = (Q, \Sigma, \Gamma, \delta, q_0, \square, \{q_f\})$, die L akzeptiert, d. h. $L(B) = L$.

   Lösungsidee: Wir beobachten, dass in $v = ww^Rw^R$ das erste w^R quasi die Mitte zwischen einem w und dem zweiten w^R darstellt. Diese Mitte markieren wir in der Eingabe in nichtdeterministischer Weise als $\widehat{w^R}$ und können $u = w\widehat{w^R}w^R$ erhalten, wenn die Eingabe aus L ist. Nun transformieren wir u wiederholt für einen Buchstaben $x \in \Sigma$, falls u mit x beginnt, mit x endet und $\widehat{w^R}$ ebenfalls mit x endet.

Die Transformation löscht dann das x am Ende und das x am Anfang von u, und markiert das x am Ende von $\widehat{w^R}$ als $\bar{x}$. Wenn es gelingt, das v in $\overline{w} = \overline{x_1}\,\overline{x_2}\ldots\overline{x_n}$ zu transformieren, dann wird die Eingabe akzeptiert.

Übergang		Bereich	Kommentar
$\delta(q_0, \square)$	$\to (q_f, \square, R)$		Das leere Wort wird akzeptiert.
$\delta(q_0, x)$	$\to (q_0, x, R)$	$x \in \Sigma$	Nichtdeterministischer Lauf zu $\widehat{w}$.
$\delta(q_0, x)$	$\to (\hat{q}, x, N)$	$x \in \Sigma$	Übergang in Markierungsphase.
$\delta(\hat{q}, x)$	$\to (\hat{q}, \hat{x}, R)$	$x \in \Sigma$	Markierung.
$\delta(\hat{q}, x)$	$\to (q_R, x, R)$	$x \in \Sigma$	Ende Markierung, Lauf nach rechts.
$\delta(q_R, y)$	$\to (q_R, y, R)$	$y \in \Sigma \cup \widehat{\Sigma} \cup \overline{\Sigma}$	Lauf an das Ende, rechts.
$\delta(q_R, \square)$	$\to (q_1, \square, L)$		Zur Verarbeitung nächstes Zeichen.
$\delta(q_1, x)$	$\to (q^x, \square, L)$	$x \in \Sigma$	x merken und löschen.
$\delta(q^x, u)$	$\to (q^x, u, L)$	$x \in \Sigma, u \in \Sigma \cup \overline{\Sigma}$	Lauf zu Endezeichen von $\widehat{w}$.
$\delta(q^x, \hat{x})$	$\to (\hat{q}^x, \bar{x}, L)$	$x \in \Sigma$	Endezeichen $\hat{x}$ von $\widehat{w}$ markieren.
$\delta(\hat{q}^x, z)$	$\to (\hat{q}^x, z, L)$	$x \in \Sigma, z \in \widehat{\Sigma} \cup \Sigma$	Lauf an linken Anfang
$\delta(\hat{q}^x, \square)$	$\to (q_R^x, \square, R)$	$x \in \Sigma$	Auf erstes Zeichen setzen.
$\delta(q_L^x, x)$	$\to (q_R, \square, R)$	$x \in \Sigma$	Erstes Zeichen x löschen.
$\delta(q_1, v)$	$\to (q_2, v, L)$	$v \in \overline{\Sigma}$	Lauf nach links über $\overline{w}$.
$\delta(q_2, v)$	$\to (q_2, v, L)$	$v \in \overline{\Sigma}$	Lauf nach links über $\overline{w}$.
$\delta(q_2, \square)$	$\to (q_f, \square, N)$		Ende, falls alle Zeichen aus $\overline{\Sigma}$.

Falls B wegen Undefiniertheit von δ hält, dann wird die Eingabe nicht akzeptiert.

Vorbereitung 1

Die Auswertung von Polynomen $p(x) = \sum_{i=0}^{n} a_i x^i$ vom Grad $n \geq 0$ für bestimmte x nach dem Horner-Algorithmus $p_i := p_{i-1}x + a_{n-i}$ für $i = 0, \ldots, n$ mit $p_{-1} = 0$ liefert $p(x) = p_n$. Wir wollen den Horner-Algorithmus dazu benutzen, eine in Binärdarstellung ohne führende Nullen gegebene natürliche Zahl $y = \sum_{i=0}^{n} b_i 2^i$ mit $b_i \in \{0, 1\}$ in eine gleichwertige unäre Strichdarstellung $u(y) \in \{|\}^*$ zu transformieren.

Sei $\beta_k = b_{k-1}b_{k-2}\ldots b_0$. Die Berechnung nach Horner protokollieren wir als Folge von „Konfigurationen", d. h. Tripeln $(u(p_i), q, \beta_{n-i})$ mit

$$u(p_i) = u(2p_{i-1}+b_{n-i}) = u(p_{i-1})u(p_{i-1})u(b_{n-i})\,,$$

wobei wir $q = q_0 = 0$ setzen wollen.
Bestimmen Sie $u(10)$, indem Sie die Folge der genannten Konfigurationen für $y = 10$ berechnen.

Lösungsvorschlag

Die Zahl 10 (i.W. „zehn") hat die Binärdarstellung $b_3b_2b_1b_0 = 1010$, für die dann gilt $y = \sum_{i=0}^{n} b_i 2^i = 2^3 + 2^1 = 10$. Um nach Horner $u(10)$ zu berechnen, müssen wir p_i für alle $i \in \{0, 1, 2, 3\}$ berechnen. Wir erhalten die Folge von Konfigurationen (ohne die Komponente q wiederholt niederzuschreiben)

$$\begin{array}{lcll} u(p_0) & = & u(p_{-1})u(p_{-1})u(b_3) & = |\,, \\ u(p_1) & = & u(p_0)u(p_0)u(b_2) & = ||\,, \\ u(p_2) & = & u(p_1)u(p_1)u(b_1) & = |||||\,, \\ u(p_3) & = & u(p_2)u(p_2)u(b_0) & = ||||||||||\,, \end{array} \qquad \begin{array}{lcl} \beta_3 & = & b_2b_1b_0 = 010\,, \\ \beta_2 & = & b_1b_0 = 10\,, \\ \beta_1 & = & b_0 = 0\,, \\ \beta_0 & = & \epsilon\,. \end{array}$$

Vorbereitung 2

Beweisen oder widerlegen Sie:
Für jede Turingmaschine T gibt es eine Turingmaschine T' mit nur einem Zustand, die die Sprache von T akzeptiert.

Lösungsvorschlag

Die Aussage ist falsch. Die Sprache, die eine TM akzeptiert, ist über Endzustände definiert. Sei also T eine TM mit nur einem Zustand q. Dann gibt es zwei Fälle:

- $q \notin F$. Dann ist $L(T) = \emptyset$, da nie ein Endzustand erreicht werden kann.
- $q \in F$. Dann ist $L(T) = \Sigma^*$, da die TM gleich zu Beginn in einem Endzustand ist und es damit eine akzeptierende Konfigurationsfolge gibt.

Also braucht man für jede nicht-triviale Sprache mehrere Zustände.
Bemerkung: Auf ähnliche Weise (aber nicht genauso) kann man zeigen, dass die entsprechende Aussage für Kellerautomaten auch falsch ist, wenn man Akzeptieren mit Endzuständen betrachtet.

Tutoraufgabe 1

Sei L die zu dem regulären Ausdruck $0|(1(0|1)^*)$ gehörige Sprache. Für eine Zahl $n \in \mathbb{N}$ bezeichne $b(n) \in L$ bzw. $u(n) \in \{|\}^*$ deren binäre bzw. unäre Darstellung.
Es ist eine (deterministische) Ein-Band-Turing-Maschine zu konstruieren, die für $n \neq 0$ die Eingabe $b(n)$ in $u(n)$ umwandelt. Die Eingabe und die Ausgabe sollen sich auf dem

sonst leeren Band befinden, und der Lese-/Schreibkopf der Maschine bei Start und Ende der Berechnung soll links auf dem ersten beschriebenen Feld des Bandes stehen.

1. Entwerfen Sie eine Programmstruktur für die zu konstruierende Turingmaschine, indem Sie versuchen, die Gesamtlösung auf Teilprobleme zurückzuführen, für deren Lösung dann in der nachfolgenden Teilaufgabe jeweils spezielle Turingmaschinen implementiert werden können.

2. Implementieren Sie Ihre spezifizierten Turingmaschinen und fügen Sie die Programme zu einem Gesamtprogramm zusammen zur Umwandlung einer binären in eine unäre Darstellung einer natürlichen Zahl.

Lösungsvorschlag

1. L ist die Sprache der Binärdarstellungen von natürlichen Zahlen (einschließlich 0) ohne führende Nullen. Man könnte die gesuchte Lösung auf die Überlegung gründen, die Zahl n jeweils um 1 zu dekrementieren und gleichzeitig eine entsprechende Strichzahl links vor dem Binärwort für n additiv zu erzeugen.

   Wir gehen einen Weg für den Aufbau der Strichzahl, der einer Auswertung einer Binärzahl nach dem Hornerschema entspricht wie in der Vorbereitungsaufgabe dargestellt.

   Für $n \neq 0$ sei $l = \lfloor \log_2(n) \rfloor$, $\beta = b_l b_{l-1} \dots b_0$ mit $b_i \in \{0,1\}$ und $n = n(\beta) = \sum_{i=0}^{l} b_i 2^i$. Die Strichzahldarstellung einer Zahl n sei $u(n) = u_n u_{n-1} \dots u_1$ mit $u_i \in \{|\}$. Nach dem Hornerschema gilt $p_l = n$, falls $p_i = p_{i-1} \cdot 2 + b_{l-i}$ für $i = 0, \dots, l$ mit $p_{-1} = 0$ gilt.

   Wir konstruieren eine Turingmaschine $T = (Q, \Sigma, \Gamma, \delta, q_0, \square, \{q_f\})$, die sukzessive die p_i als Strichzahlen $u(p_i)$ erzeugt, wobei zur Berechnung von $u(p_i)$ eine Turingmaschine T_H aufgerufen wird, die die Binärziffer b_{l-i} und die Strichzahl $u(p_{i-1})$ verarbeitet.

   (a) T findet im Zustand q_0 die Turing-Konfiguration $(u(p), q_0, b'\beta')$ mit $b' \in \{0, 1, \square\}$ und $\beta' \in \{0,1\}^*$ vor.

   (b) Falls $b' = \square$, dann geht T in den Endzustand q_f.

   (c) Andernfalls wird T_H aufgerufen mit Anfangszustand $q_{H,0}$. Die Berechnung von T_H endet mit der Konfiguration $(u(p \cdot 2 + b'), q_{H,f}, \beta')$ im Endzustand $q_{H,f}$.

   (d) Falls der Endzustand $q_{H,f}$ von T_H vorliegt, dann geht T in den Anfangszustand q_0 und fährt mit Schritt (a) fort.

   T_H führt einen Hornerschritt aus und berechnet dabei aus der Konfiguration $(u(p), q_{H,0}, b'\beta')$ die Konfiguration $(u(p \cdot 2 + b'), q_{H,f}, \beta')$ mit $b' \in \{0,1\}$. Die Verdopplung der Strichzahl wird nach links über den linken Anfang von $u(p)$ hinaus notiert. Dies könnte durch eine Turingmaschine T_V erledigt werden. Anschließend wird im Fall $b' = 0$ ein Shift von $u(p)$ nach rechts gemacht und im Fall $b' = 1$ einfach die Ziffer b' in einen Strich | umgewandelt. Diese beiden Berechnungen können weitere Turingmaschinen $T_{\rightharpoonup}, T_{|}$ erledigen.

2. Zur Verdopplung der Strichzahlen bauen wir die Maschine T_V der Vorbereitungsaufgabe ein, d. h. wir betrachten die folgende Tabelle als Erweiterung der Tabelle für T_V.

Übergang			Kommentar
$\delta(q_0, \square)$	$\rightarrow$	$(q_f, \square, N)$	das leere Eingabewort wird akzeptiert bzw. „Stopp“!
$\delta(q_0, b)$	$\rightarrow$	$(q_{H,0}, b, N)$	b ist eine Ziffer 0 oder 1, Aufruf der Hornermaschine
$\delta(q_{H,0}, b)$	$\rightarrow$	$(q_{V,0}, b, L)$	b ist eine Ziffer, Beginn der Verdopplung
$\delta(q_{V,f}, 1)$	$\rightarrow$	$(q_{H,f}, \vert, R)$	Horner ist fertig
$\delta(q_{V,f}, 0)$	$\rightarrow$	$(q_5, \vert, L)$	Beginn Rechtsshift der Strichzahl
$\delta(q_5, \vert)$	$\rightarrow$	$(q_5, \vert, L)$	nach links an den Anfang
$\delta(q_5, \square)$	$\rightarrow$	$(q_6, \square, R)$	Kopf auf ersten Strich setzen
$\delta(q_6, \vert)$	$\rightarrow$	$(q_7, \square, R)$	ersten Strich löschen
$\delta(q_7, \vert)$	$\rightarrow$	$(q_7, \vert, R)$	nach rechts zur Mitte laufen
$\delta(q_7, b)$	$\rightarrow$	$(q_{H,f}, b, N)$	Kopf auf nächster Ziffer und Ende von Horner
$\delta(q_{H,f}, *)$	$\rightarrow$	$(q_0, *, N)$	Rückkehr aus Horner auf nächstem Feld
$\delta(q_7, \square)$	$\rightarrow$	$(q_f, \square, N)$	Ende

Tutoraufgabe 2

Ein *Queue-Automat* funktioniert ähnlich wie ein Kellerautomat, nur dass der Keller durch eine Queue („Warteschlange“) ersetzt ist.
Formal ist ein Queue-Automat ein Tupel $A = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$. Dabei sind Q, Σ, Γ endliche Mengen, $q_0 \in Q$, $Z_0 \in \Gamma$, und die Übergangsfunktion δ hat den Typ $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$, wobei $|\delta(q, b, Z)| < \infty$ für alle $q \in Q, b \in \Sigma \cup \{\epsilon\}, Z \in \Gamma$.
Eine *Konfiguration* eines Queue-Automaten ist ein Tripel $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$. Dabei ist q der aktuelle Zustand, w das noch zu lesende Wort und γ der aktuelle Inhalt der Queue. Daraus ergibt sich die Übergangsrelation $\rightarrow_A$ zwischen Konfigurationen:

$$(q, bw, Z\gamma) \rightarrow_A (q', w, \gamma\gamma') \,, \qquad \text{wenn} \qquad (q', \gamma') \in \delta(q, b, Z) \,.$$

Beachten Sie, dass der einzige Unterschied zum Kellerautomaten in der Definition von $\rightarrow_A$ liegt. Hier wird γ' hinter das γ geschrieben und nicht davor.
Die (mit leerer Queue) akzeptierte Sprache eines Queue-Automaten ist

$$L_\epsilon(A) = \{w \in \Sigma^* \mid \exists q' \in Q.\ (q_0, w, Z_0) \rightarrow_A^* (q', \epsilon, \epsilon)\} \,.$$

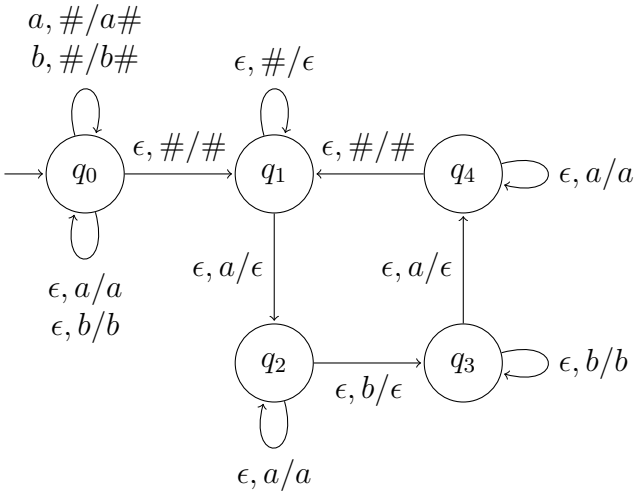
Die graphische Darstellung von Queue-Automaten ist die gleiche wie bei Kellerautomaten.

1. Geben Sie einen Queue-Automaten A an mit $L(A) = \{a^n b^n a^n \mid n \in \mathbb{N}\}$.

2. Zeigen Sie, dass jede (deterministische) Turingmaschine durch einen Queue-Automaten simuliert werden kann.

Lösungsvorschlag

1. Der Queue-Automat $A = (\{q_0, \ldots, q_4\}, \{a, b\}, \{a, b, \#\}, q_0, \#, \delta)$ sei wie folgt graphisch dargestellt.



2. Sei $M = (Q, \Sigma, \Gamma, q_0, \delta, \square, F)$ die Turingmaschine, die wir durch einen Queue-Automat A simulieren wollen. Wir nehmen an, dass $Q \cap \Gamma = \emptyset$.

   Die Konstruktionsidee ist nun, dass wir eine Konfiguration (v, q, w) der Turingmaschine als Wort $vqw\#$ in der Queue ablegen. Der QA kann die Queue rotieren, indem er ein Queue-Zeichen liest, und danach wieder ans Ende der Queue anhängt. So lässt sich die Queue flexibel manipulieren. Für jeden Schritt der Turingmaschine muss allerdings die Queue einmal komplett durchlaufen werden.

   Unser Queue-Automat A hat die Zustandsmenge

$$Q' = \{ff, start\} \cup \{find_x \mid x \in \Gamma\} \cup \{step_x^q \mid x \in \Gamma, q \in Q\}$$

   und die Übergänge

$\delta'(start, \epsilon, x)$	$\to$	$(find_x, \epsilon)$	$\forall x \in \Gamma$	Zum Kopf vorlaufen, dabei
$\delta'(find_x, \epsilon, y)$	$\to$	$(find_y, x)$	$\forall x, y \in \Gamma$	jeweils letztes Zeichen merken.
$\delta'(find_x, \epsilon, q)$	$\to$	$(step_x^q, \epsilon)$	$\forall x \in \Gamma, q \in Q$	Zustand auch merken.
$\delta'(step_x^q, \epsilon, y)$	$\to$	$\begin{cases} (ff, xq'z) & \text{falls } \delta(q,y) = (q', z, N) \\ (ff, xzq') & \text{falls } \delta(q,y) = (q', z, R) \\ (ff, q'xz) & \text{falls } \delta(q,y) = (q', z, L) \end{cases}$	$\forall x, y \in \Gamma, q \in Q$	Schritt.
$\delta'(ff, \epsilon, x)$	$\to$	(ff, x)	$\forall x \in \Gamma$	Vorspulen bis zum #...
$\delta'(ff, \epsilon, \#)$	$\to$	$(start, \#)$		... und von vorne.
$\delta'(start, \epsilon, q)$	$\to$	$(step_\square^q, \epsilon)$	$\forall q \in Q$	Sonderfälle für Rand
$\delta'(step_x^q, \epsilon, \#)$	$\to$	$(start, xq\square\#)$	$\forall x \in \Gamma, q \in Q$	

   Nun gilt (hier ohne Beweis):

$$(start, \epsilon, vqw\#) \to_A^* (start, \epsilon, v'q'w'\#) \iff (v, q, w) \to_M^* (v', q', w')$$

Um dafür zu sorgen dass auch wirklich $L(A) = L(M)$, muss man den Queue-Automaten noch so erweitern, dass er am Anfang das Eingabewort in die Queue schreibt, und bei erreichen eines Endzustands der TM die Queue komplett entleert.