Kontextfreie Sprachen und Kellerautomaten, Turing-Maschinen

Prof.Dr.Christian Wagenknecht

8. April 2008

Grenzen endlicher Automaten

Endliche Automaten können sich nichts merken: insbes. Anzahl und Reihenfolge von Zeichen.

- Anzahl:
  $L_1 = \{w \mid w = \{a, b\}^* \text{ und } w \text{ enthält genauso viele } a \text{ wie } b.\}$
- Anzahl: $L_2 = \{w \mid w = a^n b^n \text{ mit } n \geq 0\}$
- Reihenfolge:
  $L_3 = \{w \mid w = (0+1)^+!(0+1)^+ \text{ und } w \text{ ist ein Palindrom}\}$

Grenzen endlicher Automaten

Endliche Automaten können sich nichts merken: insbes. Anzahl und Reihenfolge von Zeichen.

- Anzahl:
  $L_1 = \{w \mid w = \{a, b\}^* \text{ und } w \text{ enthält genauso viele } a \text{ wie } b.\}$
- Anzahl: $L_2 = \{w \mid w = a^n b^n \text{ mit } n \geq 0\}$
- Reihenfolge:
  $L_3 = \{w \mid w = (0+1)^+!(0+1)^+ \text{ und } w \text{ ist ein Palindrom}\}$

Grenzen endlicher Automaten

Endliche Automaten können sich nichts merken: insbes. Anzahl und Reihenfolge von Zeichen.

- Anzahl:
  $L_1 = \{w \mid w = \{a, b\}^* \text{ und } w \text{ enthält genauso viele } a \text{ wie } b.\}$
- Anzahl: $L_2 = \{w \mid w = a^n b^n \text{ mit } n \geq 0\}$
- Reihenfolge:
  $L_3 = \{w \mid w = (0+1)^+ ! (0+1)^+ \text{ und } w \text{ ist ein Palindrom}\}$

Grenzen endlicher Automaten

Endliche Automaten können sich nichts merken: insbes. Anzahl und Reihenfolge von Zeichen.

- Anzahl:
  $L_1 = \{w \mid w = \{a, b\}^* \text{ und } w \text{ enthält genauso viele } a \text{ wie } b.\}$
- Anzahl: $L_2 = \{w \mid w = a^n b^n \text{ mit } n \geq 0\}$
- Reihenfolge:
  $L_3 = \{w \mid w = (0+1)^+ ! (0+1)^+ \text{ und } w \text{ ist ein Palindrom}\}$

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Definition

Ein **nichtdeterministischer Kellerautomat** NKA (engl.: NPDA = nondeterministic pushdown[1] automaton) ist durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

[1] *Pushdown* bedeutet so viel wie *nach unten drücken (winden)*, nicht etwa *umschubsen*.

Überführungsfunktion und Arbeitsweise

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp_{endlich}(Q \times \Gamma^*)$$

Aktueller Zustand von M sei q_i. $\delta(q_i, a, k) = (q_j, r_1 r_2 \ldots r_n)$ bedeutet:

- verbrauchendes Lesen des top of stack k
- Lesen des aktuellen Eingabezeichens a und Rechtsbewegung des Kopfes[2]
- Übergang in den Zustand q_j
- Schreiben (push) des Kellerwortes $r_1 r_2 \ldots r_n$ auf den Keller, so dass r_1 neues top of stack ist (leeres Kellerwort möglich)

[2] Ausnahme $\delta(q_i, \varepsilon, k)$: spontaner Übergang ($\varepsilon$), d.h. kein Zeichen wird gelesen, keine Kopfbewegung

Überführungsfunktion und Arbeitsweise

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp_{endlich}(Q \times \Gamma^*)$$

Aktueller Zustand von M sei q_i. $\delta(q_i, a, k) = (q_j, r_1 r_2 \ldots r_n)$ bedeutet:

- verbrauchendes Lesen des top of stack k
- Lesen des aktuellen Eingabezeichens a und Rechtsbewegung des Kopfes[2]
- Übergang in den Zustand q_j
- Schreiben (push) des Kellerwortes $r_1 r_2 \ldots r_n$ auf den Keller, so dass r_1 neues top of stack ist (leeres Kellerwort möglich)

[2] Ausnahme $\delta(q_i, \varepsilon, k)$: spontaner Übergang ($\varepsilon$), d.h. kein Zeichen wird gelesen, keine Kopfbewegung

Überführungsfunktion und Arbeitsweise

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp_{endlich}(Q \times \Gamma^*)$$

Aktueller Zustand von M sei q_i. $\delta(q_i, a, k) = (q_j, r_1 r_2 \ldots r_n)$ bedeutet:

- verbrauchendes Lesen des top of stack k
- Lesen des aktuellen Eingabezeichens a und Rechtsbewegung des Kopfes[2]
- Übergang in den Zustand q_j
- Schreiben (push) des Kellerwortes $r_1 r_2 \ldots r_n$ auf den Keller, so dass r_1 neues top of stack ist (leeres Kellerwort möglich)

[2] Ausnahme $\delta(q_i, \varepsilon, k)$: spontaner Übergang ($\varepsilon$), d.h. kein Zeichen wird gelesen, keine Kopfbewegung

Überführungsfunktion und Arbeitsweise

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp_{endlich}(Q \times \Gamma^*)$$

Aktueller Zustand von M sei q_i. $\delta(q_i, a, k) = (q_j, r_1 r_2 \ldots r_n)$ bedeutet:

- verbrauchendes Lesen des top of stack k
- Lesen des aktuellen Eingabezeichens a und Rechtsbewegung des Kopfes[2]
- Übergang in den Zustand q_j
- Schreiben (push) des Kellerwortes $r_1 r_2 \ldots r_n$ auf den Keller, so dass r_1 neues top of stack ist (leeres Kellerwort möglich)

[2] Ausnahme $\delta(q_i, \varepsilon, k)$: spontaner Übergang ($\varepsilon$), d.h. kein Zeichen wird gelesen, keine Kopfbewegung

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf Eingabeband und k_0 im Keller (top of stack)
Stopp, entweder wenn w vollständig gelesen oder durch Crash (wenn δ nicht anwendbar); Crash $\rightarrow w \notin L(M)$.

Die von einem 7-Tupel-NKA M akzeptierte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, k_0) \overset{*}{\vdash} (q_e, \varepsilon, K), \text{ mit } q_e \in E, K \in \Gamma^*\}.$$

Der aktuelle Kellerinhalt bei Stopp spielt *keine* Rolle.

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf Eingabeband und k_0 im Keller (top of stack)
Stopp, entweder wenn w vollständig gelesen oder durch Crash (wenn δ nicht anwendbar); Crash $\rightarrow w \notin L(M)$.

Die von einem 7-Tupel-NKA M akzeptierte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, k_0) \overset{*}{\vdash} (q_e, \varepsilon, K), \text{ mit } q_e \in E, K \in \Gamma^*\}.$$

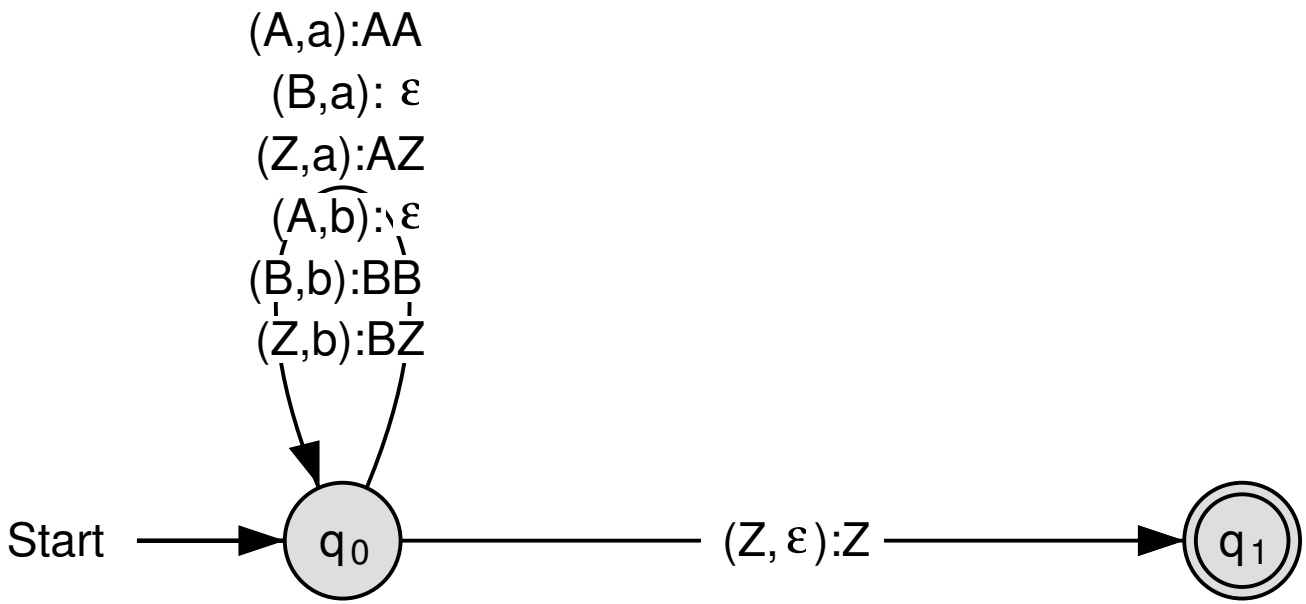
Der aktuelle Kellerinhalt bei Stopp spielt *keine* Rolle.

Beispiel

$M = (\{q_0, q_1\}, \{a, b\}, \{A, B, Z\}, \delta, q_0, Z, \{q_1\})$, mit

$$\begin{aligned}
\delta(q_0, a, A) &= \{(q_0, AA)\} \\
\delta(q_0, b, A) &= \{(q_0, \varepsilon)\} \\
\delta(q_0, a, B) &= \{(q_0, \varepsilon)\} \\
\delta(q_0, b, B) &= \{(q_0, BB)\} \\
\delta(q_0, a, Z) &= \{(q_0, AZ)\} \\
\delta(q_0, b, Z) &= \{(q_0, BZ)\} \\
\delta(q_0, \varepsilon, Z) &= \{(q_1, Z)\}.
\end{aligned}$$

Übergangsgraph



Konfigurationenfolge (Tripel aus $Q \times \Sigma^* \times \Gamma^*$)

Konfigurationenfolge für die Verarbeitung von a^2b^2 durch M

Zustand	Eingabewort	Kellerbelegung (links: top of stack)
q_0	$aabb$	Z
q_0	abb	AZ
q_0	bb	AAZ
q_0	b	AZ
q_0	ε	Z
q_1	ε	Z

$aabb \in L(M)$

Konfigurationenfolge (Tripel aus $Q \times \Sigma^* \times \Gamma^*$)

Konfigurationenfolge für die Verarbeitung von a^2b^2 durch M

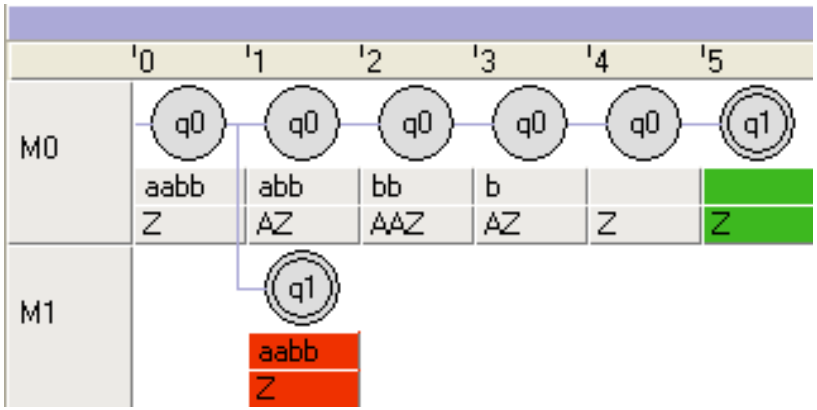
Zustand	Eingabewort	Kellerbelegung (links: top of stack)
q_0	$aabb$	Z
q_0	abb	AZ
q_0	bb	AAZ
q_0	b	AZ
q_0	ε	Z
q_1	ε	Z

$aabb \in L(M)$

Konfigurationenfolge (Tripel aus $Q \times \Sigma^* \times \Gamma^*$)

Konfigurationenfolge für die Verarbeitung von a^2b^2 durch M

Zustand	Eingabewort	Kellerbelegung (links: top of stack)
q_0	$aabb$	Z
q_0	abb	AZ
q_0	bb	AAZ
q_0	b	AZ
q_0	ε	Z
q_1	ε	Z



$aabb \in L(M)$

Definition

Ein deterministischer Kellerautomat DKA (engl.: DPDA = deterministic pushdown automaton) ist durch ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, k_0, E)$ definiert.

Die verwendeten Symbole haben folgende Bedeutungen:

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet
- Γ ... Kelleralphabet
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- k_0 ... Kellervorbelegungszeichen, $k_0 \in \Gamma$ und $k_0 \notin \Sigma$
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Überführungsfunktion und Arbeitsweise

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

Der DKA befinde sich im Zustand q_i.

- verbrauchendes Lesen des top of stack k
- Lesen des nächsten Eingabezeichens a und Rechtsbewegung des Kopfes – Ausnahme: spontaner Übergang (ε), d.h. kein Zeichen wird gelesen, keine Kopfbewegung; allerdings nur erlaubt, wenn nicht parallel zu verbrauchendem Übergang anwendbar: $\delta(q_i, a, k) = \ldots$ und $\delta(q_i, \varepsilon, k) = \ldots$ verboten!
- Übergang in den Zustand q_j, wenn $\delta(q_i, a, k) = (q_j, r_1 r_2 \ldots r_n)$
- Schreiben (push) des Kellerwortes $r_1 r_2 \ldots r_n$ auf den Keller, so dass r_1 neues top of stack ist

Akzeptierte Sprache

wie NKA

Start des Automaten in q_0 mit w auf Eingabeband und k_0 im Keller (top of stack)
Stopp, entweder wenn w vollständig gelesen oder durch Crash (wenn δ nicht anwendbar); Crash $\rightarrow w \notin L(M)$.

Die von einem 7-Tupel-NKA M akzeptierte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, k_0) \overset{*}{\vdash} (q_e, \varepsilon, K), \text{ mit } q_e \in E, K \in \Gamma^*\}.$$

Kellerinhalt bei Stopp spielt keine Rolle.

Palindrome

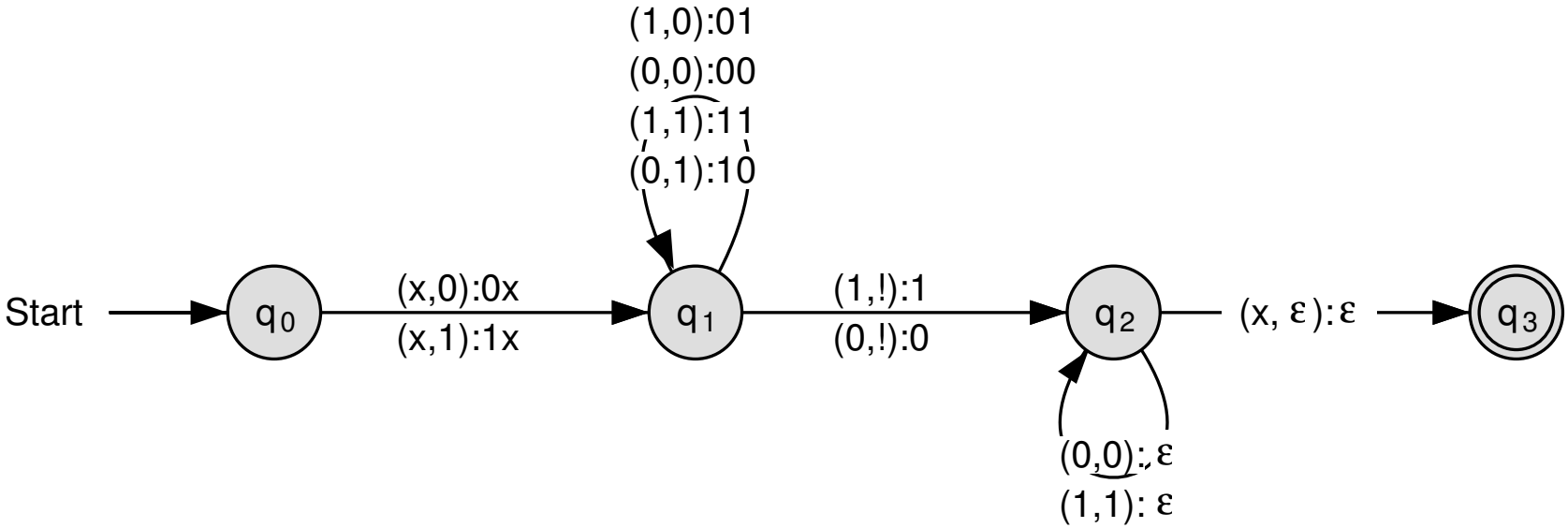
Palindrome sind Wörter, die vorwärts und rückwärts gelesen übereinstimmen. Beispiele sind: otto, reliefpfeiler, rentner und lagerregal. Die Symmetrieachse (Wortmitte) kann zwischen zwei Zeichen oder durch ein Zeichen hindurch verlaufen. (kfS $\rightarrow$ NKA)

Problem

Entwickeln Sie einen DKA für die Sprache der Palindrome als echte Teilmenge von $(0+1)^+!(0+1)^+$.
(Das Zeichen ! *markiert quasi die jeweilige Wortmitte.)*

Lösung

$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1, !\}, \{0, 1, x\}, \delta, q_0, x, \{q_3\})$, *mit*



Zusammenhang NKA, DKA, kfG

Satz

KfS werden sowohl mit NKA als auch mit kfG vollständig beschrieben. DKA beschreiben die dkfS (deterministisch kontextfreie Sprachen).
Es gilt: $\mathcal{L}_{dkfS} \subset \mathcal{L}_{kfS}$. $(\mathcal{L}_{dkfS} = \mathcal{L}_{LR(k)})$

Beweis.

s. Literatur □

DkfS und Praxis

DKA beschreiben die (für die Praxis im Allg. ausreichenden) dkfS (deterministisch kontextfreie Sprachen).

Effiziente Analyse-Verfahren gibt es für LL(1)-Sprachen (Methode des rekursiven Abstiegs) und für Teilmengen der LR(k)-Sprachen.

Grundprinzip: Scanner assistiert einem Parser (2 Durchläufe; in der Praxis: on demand)
Dadurch verändert sich die Grammatik: z.B. Zahl (= Nicht-terminal) wird zu Terminal der Parsergrammatik.

Automatisierte Compilergenerierung (Compiler = Sprachübersetzer, d.h. 1. parsing, 2. Zielcode-Generierung)

Computerübung

Aufgabe 1: Bearbeiten Sie sämtliche NKA/DKA-Beispiele aus der Vorlesung mit AutoEdit.

Aufgabe 2: Konstruieren Sie einen NKA für die Sprache
$L = \{w \mid w = 0^n 1^{2n} \text{ mit } n > 0\}$.

Aufgabe 3: Entwickeln Sie einen DKA für die durch folgende kfG gegebene Sprache $L(G)$.
$G = (\{S\}, \{x, (,), [,]\}, \{S \rightarrow x \mid SS \mid (S) \mid [S] \mid \varepsilon\}, S)$.
Beispielwörter: $(x\,(x))$, $[x\,(x)\,[x\,(x)]]$.

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- \$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Definition

Eine **Turingmaschine** (TM) ist ein 7-Tupel
$M = (Q, \Sigma, \Gamma, \delta, q_0, \$, E)$.

- Q ... endliche Menge von Zuständen
- Σ ... Eingabealphabet
- Γ ... Bandalphabet, wobei $\Sigma \subseteq \Gamma \backslash \{\$\}$ erlaubt ist
- δ ... partielle Überführungsfunktion
- q_0 ... Anfangszustand, $q_0 \in Q$
- $\$$ ... Bandvorbelegungszeichen, kurz: Blankzeichen, mit $\$ \in \Gamma$ und $\$ \notin \Sigma$
- E ... endliche Menge von Endzuständen $E \subseteq Q$

Überführungsfunktion und Arbeitsweise

DTM: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
NTM: $Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R, N\})$

Die DTM befinde sich im Zustand q_i. Wir betrachten:
$\delta(q_i, a) = (q_j, b, L)$.

1. Lesen des Inhalts des aktuellen Feldes a
2. Schreiben von b ins aktuelle Feld
3. Zustandswechsel in q_j
4. Kopfbewegung um ein Feld nach links

Überführungsfunktion und Arbeitsweise

DTM: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
NTM: $Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R, N\})$

Die DTM befinde sich im Zustand q_i. Wir betrachten:
$\delta(q_i, a) = (q_j, b, L)$.

1. Lesen des Inhalts des aktuellen Feldes a
2. Schreiben von b ins aktuelle Feld
3. Zustandswechsel in q_j
4. Kopfbewegung um ein Feld nach links

Überführungsfunktion und Arbeitsweise

DTM: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
NTM: $Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R, N\})$

Die DTM befinde sich im Zustand q_i. Wir betrachten:
$\delta(q_i, a) = (q_j, b, L)$.

1. Lesen des Inhalts des aktuellen Feldes a
2. Schreiben von b ins aktuelle Feld
3. Zustandswechsel in q_j
4. Kopfbewegung um ein Feld nach links

Überführungsfunktion und Arbeitsweise

DTM: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
NTM: $Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R, N\})$

Die DTM befinde sich im Zustand q_i. Wir betrachten:
$\delta(q_i, a) = (q_j, b, L)$.

1. Lesen des Inhalts des aktuellen Feldes a
2. Schreiben von b ins aktuelle Feld
3. Zustandswechsel in q_j
4. Kopfbewegung um ein Feld nach links

TM - stoppt durch Crash

Wenn wir eine TM M ansetzen auf ein Wort w, so kann M ...

- ... in einem Endzustand stoppen: $w \in L(M)$
- ... in einem Nicht-Endzustand stoppen: $w \notin L(M)$
- ... nicht stoppen (ewiger Links/Rechtslauf oder loop-Verhalten)

Die von M akzeptierten Wörter $w \in T^*$ bilden die von M akzeptierte Sprache. Sprachen dieses Typs heißen *entscheidbare* oder *rekursiv aufzählbare Sprachen*.

Es gibt TM, die für sämtliche Eingabewörter (in irgendeinem Zustand) stoppen. Die von diesen TM beschriebenen Sprachen gehören zur Menge der *rekursiven Sprachen*.

TM - stoppt durch Crash

Wenn wir eine TM M ansetzen auf ein Wort w, so kann M ...

- ... in einem Endzustand stoppen: $w \in L(M)$
- ... in einem Nicht-Endzustand stoppen: $w \notin L(M)$
- ... nicht stoppen (ewiger Links/Rechtslauf oder loop-Verhalten)

Die von M akzeptierten Wörter $w \in T^*$ bilden die von M akzeptierte Sprache. Sprachen dieses Typs heißen *entscheidbare* oder *rekursiv aufzählbare Sprachen*.

Es gibt TM, die für sämtliche Eingabewörter (in irgendeinem Zustand) stoppen. Die von diesen TM beschriebenen Sprachen gehören zur Menge der *rekursiven Sprachen*.

TM - stoppt durch Crash

Wenn wir eine TM M ansetzen auf ein Wort w, so kann M ...

- ... in einem Endzustand stoppen: $w \in L(M)$
- ... in einem Nicht-Endzustand stoppen: $w \notin L(M)$
- ... nicht stoppen (ewiger Links/Rechtslauf oder loop-Verhalten)

Die von M akzeptierten Wörter $w \in T^*$ bilden die von M akzeptierte Sprache. Sprachen dieses Typs heißen *entscheidbare* oder *rekursiv aufzählbare Sprachen*.

Es gibt TM, die für sämtliche Eingabewörter (in irgendeinem Zustand) stoppen. Die von diesen TM beschriebenen Sprachen gehören zur Menge der *rekursiven Sprachen*.

TM - stoppt durch Crash

Wenn wir eine TM M ansetzen auf ein Wort w, so kann M ...

- ... in einem Endzustand stoppen: $w \in L(M)$
- ... in einem Nicht-Endzustand stoppen: $w \notin L(M)$
- ... nicht stoppen (ewiger Links/Rechtslauf oder loop-Verhalten)

Die von M akzeptierten Wörter $w \in T^*$ bilden die von M akzeptierte Sprache. Sprachen dieses Typs heißen *entscheidbare* oder *rekursiv aufzählbare Sprachen*.

Es gibt TM, die für sämtliche Eingabewörter (in irgendeinem Zustand) stoppen. Die von diesen TM beschriebenen Sprachen gehören zur Menge der *rekursiven Sprachen*.

TM - stoppt durch Crash

Wenn wir eine TM M ansetzen auf ein Wort w, so kann M ...

- ... in einem Endzustand stoppen: $w \in L(M)$
- ... in einem Nicht-Endzustand stoppen: $w \notin L(M)$
- ... nicht stoppen (ewiger Links/Rechtslauf oder loop-Verhalten)

Die von M akzeptierten Wörter $w \in T^*$ bilden die von M akzeptierte Sprache. Sprachen dieses Typs heißen *entscheidbare* oder *rekursiv aufzählbare Sprachen*.

Es gibt TM, die für sämtliche Eingabewörter (in irgendeinem Zustand) stoppen. Die von diesen TM beschriebenen Sprachen gehören zur Menge der *rekursiven Sprachen*.

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf dem mit \$ vorbelegten Eingabeband, Kopf auf erstem Zeichen von w

Stopp in Zustand q_i durch Crash, d.h. wenn $\delta(q_i, a)$ n.d. für das aktuell gelesene Zeichen a

Die von einer TM M *akzeptierte Sprache* ist

$$\begin{aligned} L(M) \quad = \quad & \{w \in \Sigma^* \mid (\alpha, q_0, w) \overset{*}{\vdash} (\beta, q_e, \gamma), \\ & \text{mit } \alpha = \{\$\}^+,\ \beta, \gamma \in \Gamma^* \text{ und } q_e \in E\}. \end{aligned}$$

Die Kopfposition beim Stopp der TM spielt also keine Rolle.

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf dem mit \$ vorbelegten Eingabeband, Kopf auf erstem Zeichen von w

Stopp in Zustand q_i durch Crash, d.h. wenn $\delta(q_i, a)$ n.d. für das aktuell gelesene Zeichen a

Die von einer TM M *akzeptierte Sprache* ist

$$L(M) = \{w \in \Sigma^* \mid (\alpha, q_0, w) \overset{*}{\vdash} (\beta, q_e, \gamma), \text{ mit } \alpha = \{\$\}^+,\ \beta, \gamma \in \Gamma^* \text{ und } q_e \in E\}.$$

Die Kopfposition beim Stopp der TM spielt also keine Rolle.

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf dem mit \$ vorbelegten Eingabeband, Kopf auf erstem Zeichen von w

Stopp in Zustand q_i durch Crash, d.h. wenn $\delta(q_i, a)$ n.d. für das aktuell gelesene Zeichen a

Die von einer TM M *akzeptierte Sprache* ist

$$L(M) = \{w \in \Sigma^* \mid (\alpha, q_0, w) \overset{*}{\vdash} (\beta, q_e, \gamma), \text{ mit } \alpha = \{\$\}^+,\ \beta, \gamma \in \Gamma^* \text{ und } q_e \in E\}.$$

Die Kopfposition beim Stopp der TM spielt also keine Rolle.

Akzeptierte Sprache

Start des Automaten in q_0 mit w auf dem mit \$ vorbelegten Eingabeband, Kopf auf erstem Zeichen von w

Stopp in Zustand q_i durch Crash, d.h. wenn $\delta(q_i, a)$ n.d. für das aktuell gelesene Zeichen a

Die von einer TM M *akzeptierte Sprache* ist

$$\begin{aligned} L(M) &= \{w \in \Sigma^* \mid (\alpha, q_0, w) \overset{*}{\vdash} (\beta, q_e, \gamma), \\ &\quad \text{mit } \alpha = \{\$\}^+,\ \beta, \gamma \in \Gamma^* \text{ und } q_e \in E\}. \end{aligned}$$

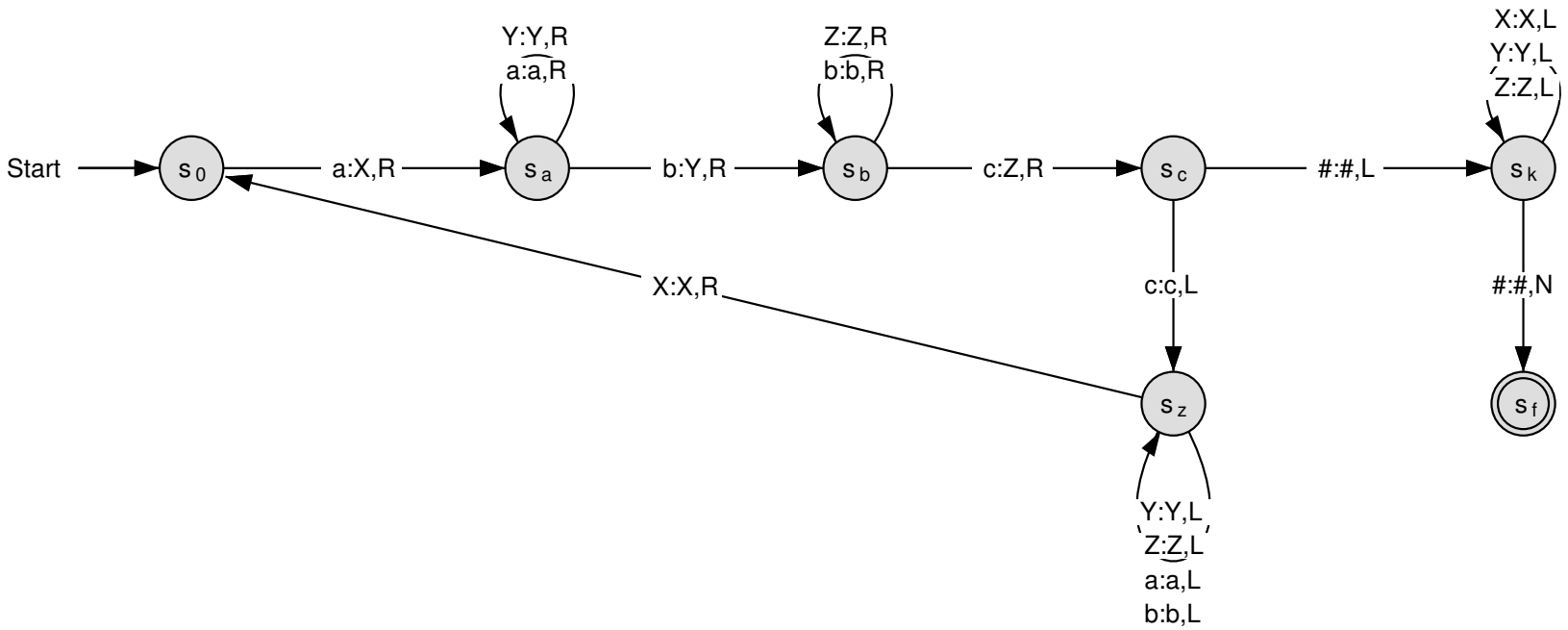
Die Kopfposition beim Stopp der TM spielt also keine Rolle.

Beispiel

TM M für die Sprache $L = \{w \mid w = a^n b^n c^n \text{ und } n \geq 1\}$. $M = (\{s_0, s_a, s_b, s_c, s_k, s_f, s_z\}, \{a, b, c\}, \{a, b, c, \#, X, Y, Z\}, \delta, s_0, \#, \{s_f\})$, mit

δ	$\#$	Z	a	b	c	X	Y
s_0	-	-	(s_a, X, R)	-	-	-	-
s_a	-	-	(s_a, a, R)	(s_b, Y, R)	-	-	(s_a, Y, R)
s_b	-	(s_b, Z, R)	-	(s_b, b, R)	(s_c, Z, R)	-	-
s_c	$(s_k, \#, L)$	-	-	-	(s_z, c, L)	-	-
s_k	$(s_f, \#, N)$	(s_k, Z, L)	-	-	-	(s_k, X, L)	(s_k, Y, L)
s_z	-	(s_z, Z, L)	(s_z, a, L)	(s_z, b, L)	-	(s_0, X, R)	(s_z, Y, L)
s_f	-	-	-	-	-	-	-

Darstellung von δ als Graph



Grundidee von M (im Beispiel)

- Schreib-/Lesekopf wandert von links nach rechts und ersetzt dabei a durch X, b durch Y und c durch Z.
- Anschließend wird der Kopf an die Anfangsposition zurückgeführt und der oben beschriebene Rechtslauf erfolgt von neuem.
- M erreicht den Endzustand nur, wenn die Anzahlen der drei Eingabealphabetzeichen übereinstimmen und in der Reihenfolge $a - b - c$ sind.

Grundidee von M (im Beispiel)

- Schreib-/Lesekopf wandert von links nach rechts und ersetzt dabei a durch X, b durch Y und c durch Z.
- Anschließend wird der Kopf an die Anfangsposition zurückgeführt und der oben beschriebene Rechtslauf erfolgt von neuem.
- M erreicht den Endzustand nur, wenn die Anzahlen der drei Eingabealphabetzeichen übereinstimmen und in der Reihenfolge $a - b - c$ sind.

Grundidee von M (im Beispiel)

- Schreib-/Lesekopf wandert von links nach rechts und ersetzt dabei a durch X, b durch Y und c durch Z.
- Anschließend wird der Kopf an die Anfangsposition zurückgeführt und der oben beschriebene Rechtslauf erfolgt von neuem.
- M erreicht den Endzustand nur, wenn die Anzahlen der drei Eingabealphabetzeichen übereinstimmen und in der Reihenfolge $a - b - c$ sind.

TM berechnen Funktionen

Akzeptor $\longrightarrow$ Funktionswert-Berechner $f(x_1, x_2, \ldots, x_n) = ?$
(Berechenbarkeitstheorie; Effizienz)

Verabredungen:

- Bandinschrift: Wörter für $x_1, x_2, \ldots, x_n$ durch je genau ein Vorbelegungszeichen getrennt
- Kopf steht auf erstem Zeichen von x_1

Interpretation des Resultats: Kopf steht auf dem ersten Zeichen des Wortes für $f(x_1, x_2, \ldots, x_n)$.

TM berechnen Funktionen

Akzeptor $\longrightarrow$ Funktionswert-Berechner $f(x_1, x_2, \ldots, x_n) = ?$
(Berechenbarkeitstheorie; Effizienz)

Verabredungen:

- Bandinschrift: Wörter für $x_1, x_2, \ldots, x_n$ durch je genau ein Vorbelegungszeichen getrennt
- Kopf steht auf erstem Zeichen von x_1

Interpretation des Resultats: Kopf steht auf dem ersten Zeichen des Wortes für $f(x_1, x_2, \ldots, x_n)$.

TM berechnen Funktionen

Akzeptor $\longrightarrow$ Funktionswert-Berechner $f(x_1, x_2, \ldots, x_n) = ?$
(Berechenbarkeitstheorie; Effizienz)

Verabredungen:

- Bandinschrift: Wörter für $x_1, x_2, \ldots, x_n$ durch je genau ein Vorbelegungszeichen getrennt
- Kopf steht auf erstem Zeichen von x_1

Interpretation des Resultats: Kopf steht auf dem ersten Zeichen des Wortes für $f(x_1, x_2, \ldots, x_n)$.

TM berechnen Funktionen

Akzeptor $\longrightarrow$ Funktionswert-Berechner $f(x_1, x_2, \ldots, x_n) = ?$
(Berechenbarkeitstheorie; Effizienz)

Verabredungen:

- Bandinschrift: Wörter für $x_1, x_2, \ldots, x_n$ durch je genau ein Vorbelegungszeichen getrennt
- Kopf steht auf erstem Zeichen von x_1

Interpretation des Resultats: Kopf steht auf dem ersten Zeichen des Wortes für $f(x_1, x_2, \ldots, x_n)$.

Nachfolger-Maschine

Problem

Definieren Sie eine Nachfolgermaschine, d.h. eine TM, die zu einer in unärer Darstellung auf das Band geschriebenen Zahl deren Nachfolger berechnet und in unärer Darstellung auf dem Band hinterlässt.

Nachfolger-Maschine

Lösung

Kopf bewegt sich an das rechte Ende des Wortes schreibt eine 1 in das erste freie Feld bewegt den Kopf zurück an das linke Ende des Wortes und stoppt

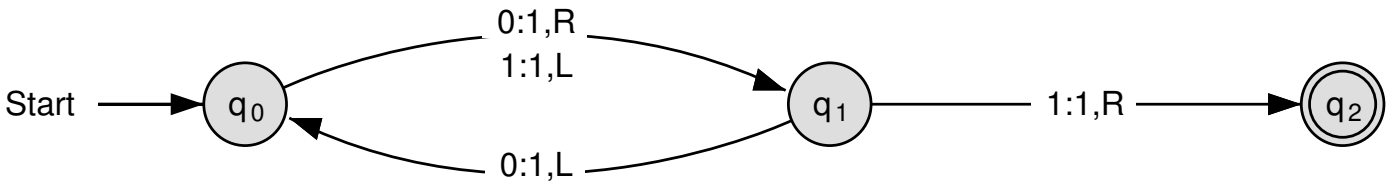
Busy Beaver

Problem

Gesucht ist je eine TM, die mit n Zuständen zzgl. eines Endzustands, d.h. $Z = \{z_1, z_2, \ldots, z_n, z_e\}$, möglichst viele Einsen auf ein anfangs leeres, d.h. mit z.B. $ vorbelegtes, Band schreibt. Die Einsen müssen nicht notwendigerweise zusammenhängend auf dem Band stehen. Auch für die Kopfstellung am Ende wird nichts weiter gefordert. Wichtig ist aber, dass man die TM so konstruiert, dass sie im Zustand z_e (per Crash) stoppt, was bekanntlich lt. Definition ohnehin gefordert ist.

Lösung für $n = 2$

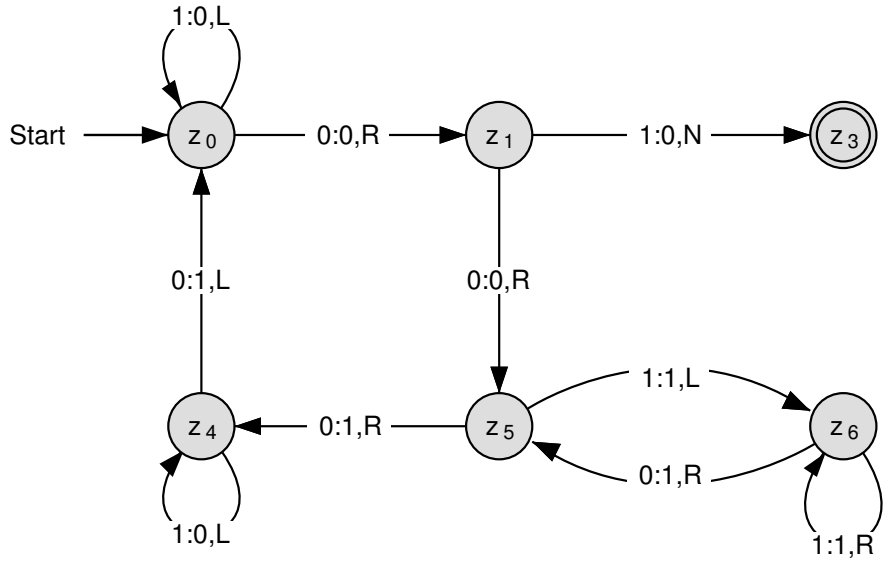
$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1\}, \delta, q_0, 0, \{q_2\})$ mit



Stoppt nach sechs Schritten und schreibt **vier** 1en auf das Band.

Beispiel: Missratener Biber

$M = (\{z_0, z_1, z_3, z_4, z_5, z_6\}, \{0,1\}, \{0,1\}, \delta, z_0, 0, \{z_3\}))$ mit



Stoppt nach 187 Schritten und schreibt keine einzige 1 auf das Band.

Computerübung

Aufgabe 1: Bearbeiten Sie sämtliche TM-Beispiele aus der Vorlesung mit AutoEdit.

Aufgabe 2: Konstruieren Sie eine DTM für eine selbstgewählte reguläre Sprache, für die man ja eigentlich EA verwendet.

Aufgabe 3: Entwickeln Sie eine DTM, die zu zwei in unärer Darstellung auf das Band geschriebenen Wörtern für die Zahlen a und b deren Summe $a + b$ berechnet, indem sie das zugehörige Wort in unärer Darstellung auf dem Band hinterlässt.
Entwerfen Sie zunächst einen Plan auf Konzeptpapier!