# Lösungen zur Probeklausur mit Kommentaren

## Aufgabe 1

**a.** Ein Wort $w \in \{a,b\}^*$ liegt genau dann in $L$, wenn es entweder mit $a$ anfängt und mit $b$ endet oder umgekehrt. Also erhält man

$$L \quad = \quad L(a(a|b)^*b \,|\, b(a|b)^*a)$$

**b.** Ein DEA, der die Sprache $L$ akzeptiert, sollte sich im aktuellen Zustand das erste und das letzte Zeichen des bisher gelesenen Wortes merken. Also benötigt er neben dem Startzustand (in dem er noch gar kein Zeichen gelesen hat) vier Zustände $p_{aa}, p_{ab}, p_{ba}$ und $p_{bb}$, wobei $p_{z_1z_2}$ bedeutet, dass das bisher gelesene Wort mit $z_1$ beginnt und mit $z_2$ endet.

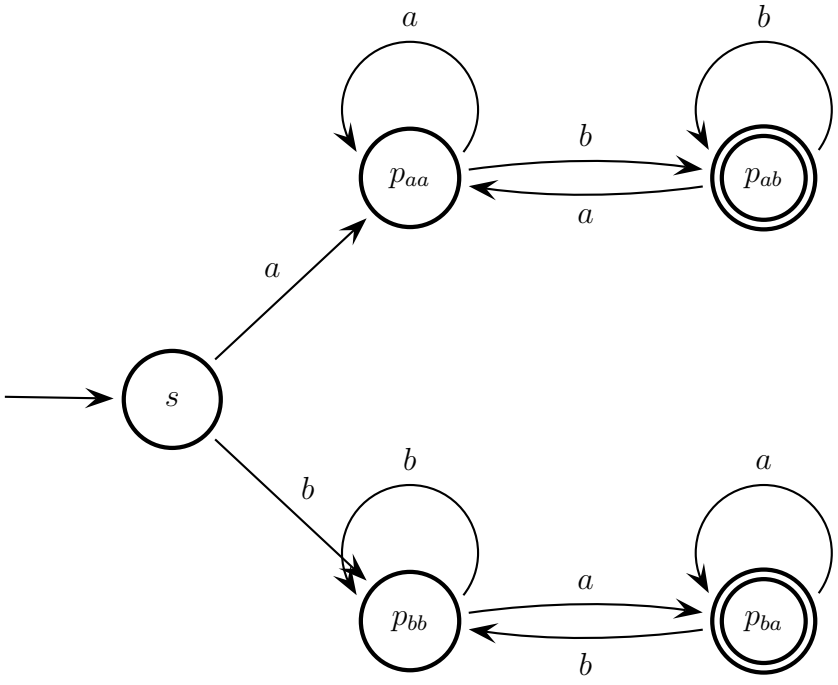
Man wählt also $A = (\Sigma, Q, s, F, \delta)$ mit $Q = \{s, p_{aa}, p_{ab}, p_{ba}, p_{bb}\}$, $F = \{p_{ab}, p_{ba}\}$ und

$$\begin{array}{ll} \delta : Q \times \Sigma \to Q & \\ \delta(s, z) = p_{zz} & \text{für alle } z \in \{a,b\} \\ \delta(p_{z_1z_2}, z) = p_{z_1z} & \text{für alle } z_1, z_2, z \in \{a,b\} \end{array}$$

In Tabellenform:

| $\delta$ | $a$ | $b$ |
|---|---|---|
| $s$ | $p_{aa}$ | $p_{bb}$ |
| $p_{aa}$ | $p_{aa}$ | $p_{ab}$ |
| $p_{ab}$ | $p_{aa}$ | $p_{ab}$ |
| $p_{ba}$ | $p_{ba}$ | $p_{bb}$ |
| $p_{bb}$ | $p_{ba}$ | $p_{bb}$ |

Graphisch:



**c.** Es wird zunächst gezeigt, dass der Automat $A$ tatsächlich die in **b.** beschriebene Eigenschaft besitzt, d.h. dass

$$\delta^*(s, w) \quad = \quad p_{z_1 z_2} \tag{1}$$

für alle $w \in \{a, b\}^+$ mit erstem Zeichen $z_1$ und letztem Zeichen $z_2$. Dies wird durch Induktion über $|w|$ bewiesen.

$|w| = 1$, d.h. $w = z$ für ein $z \in \{a, b\}$

Dann ist $z$ erstes und letztes Zeichen von $w$ und per Definition von $\delta$ gilt $\delta^*(s, w) = \delta(s, z) = p_{zz}$. Also ist (1) für $w$ erfüllt.

$|w| > 1$:

Sei $w = vz$ mit $v \in \{a, b\}^*$ und $z \in \{a, b\}$. Wegen $|w| > 1$ ist $|v| \geq 1$. Sei $z_1$ das erste und $z_2$ das letzte Zeichen von $v$. Dann ist $z_1$ das erste und $z$ das letzte Zeichen von $w$. Nach Induktionsannahme für $v$ gilt $\delta^*(s, v) = p_{z_1 z_2}$, also $\delta^*(s, w) = \delta^*(s, vz) = \delta(\delta^*(s, v), z) = \delta(p_{z_1 z_2}, z) = p_{z_1 z}$, wobei die letzte Gleichheit per Definition von $\delta$ gilt. Also ist (1) auch für $w$ erfüllt.

Damit ist (1) bewiesen und wir erhalten zunächst für alle $w \in \{a, b\}^+$:

$$\begin{aligned} w \in L \quad &\Leftrightarrow \quad \delta^*(s, w) = p_{z_1 z_2} \text{ mit } z_1 \neq z_2 \\ &\qquad \text{wegen (1)} \\ &\Leftrightarrow \quad \delta^*(s, w) \in F \\ &\qquad \text{per Definition von } F \\ &\Leftrightarrow \quad w \in L(A) \end{aligned}$$

Außerdem gilt $\varepsilon \notin L$ und $\varepsilon \notin L(A)$ wegen $s \notin F$, d.h. die Äquivalenz

$$w \in L \quad \Leftrightarrow \quad w \in L(A)$$

gilt sogar für alle $w \in \{a, b\}^*$. Also ist $L = L(A)$ bewiesen.

Man beachte, dass (1) eine Aussage über *alle* Wörter $w \in \{a, b\}^+$ macht: Für jedes $w \in \{a, b\}^+$ wird der Zustand $\delta^*(s, w)$ angegeben, zu dem das Wort $w$ führt. Meistens braucht man eine derart allgemeine Aussage, damit der Induktionsbeweis durchgeht. Mit einer spezielleren Aussage wie $L \subseteq L(A)$, die nur etwas über die Wörter aus $L$ aussagt, gelingt der Induktionsschritt üblicherweise *nicht*.

**d.** Mit dem Verfahren aus der Vorlesung erhält man zu jedem (N)DEA $A$ eine rechtslineare Grammatik $G$ mit $L(G) = L(A)$. In unserem Fall erhalten wir also die folgende Grammatik $G$ mit $L(G) = L(A) = L$: $G = (\Sigma, N, S, P)$ mit $\Sigma = \{a, b\}$, $N = Q = \{s, p_{aa}, p_{ab}, p_{ba}, p_{bb}\}$, $S = s$ und

$$\begin{aligned} P = \{ \; & s \to a p_{aa} \mid b p_{bb}, \\ & p_{aa} \to a p_{aa} \mid b p_{ab}, \\ & p_{bb} \to b p_{bb} \mid a p_{ba}, \\ & p_{ab} \to a p_{aa} \mid b p_{ba} \mid \varepsilon, \\ & p_{ba} \to b p_{bb} \mid a p_{ba} \mid \varepsilon \; \} \end{aligned}$$

## Aufgabe 2

Sei $L = \{w_1 a w_2 \mid w_1, w_2 \in \{a, b\}^* \wedge |w_1| = |w_2|\}$.

**a.** Nach dem (starken) Pumping Lemma genügt es zu zeigen: Für jedes $n \geq 1$ existiert ein $x \in L$ mit $|x| \geq n$ so, dass für jede Zerlegung $x = uvw$ mit $v \neq \varepsilon$ und $|uv| \leq n$ ein Pumpfaktor $i \in \mathbb{N}$ existiert mit $uv^i w \notin L$.

Sei $n \geq 1$. Man wähle $x = b^n a b^n$. Dann ist $x \in L$ und $|x| = 2n + 1 \geq n$. Sei $x = uvw$ mit $v \neq \varepsilon$ und $|uv| \leq n$. Dann besteht $v$ nur aus $b$s, d.h. $v = b^k$ für ein $k \geq 1$ und es folgt $uv^2w = b^{n+k}ab^n \notin L$.

Wie kommt man auf ein geeignetes Wort x? Man sollte natürlich ein Wort wählen, das sich leicht "aus der Sprache heraus pumpen lässt". Da $L$ alle Wörter enthält, die ein $a$ in der Mitte besitzen, wäre es eher ungeschickt, ein Wort mit vielen $a$s auszuwählen, denn dann könnte beim Pumpen eines dieser $a$s 'versehentlich' in die Mitte gelangen. Deshalb wählt man ein Wort, das nur das eine (unbedingt notwendige) $a$ in der Mitte enthält. Dieses eine $a$ lässt sich beim Pumpen leicht aus der Mitte heraus schieben.

**b.** Nach dem Satz von Myhill-Nerode genügt es, unendlich viele Wörter in $\Sigma^*$ zu finden, die paarweise $L$-unterscheidbar sind. Man wähle die unendlich vielen Wörter $b^n a$ mit $n \in \mathbb{N}$. Für alle $m \neq n$ gilt $b^n a b^n \in L$ und $b^m a b^n \notin L$, also sind $b^n a$ und $b^m a$ $L$-unterscheidbar.

Der Satz von Myhill-Nerode ist nichts anderes als eine abstrakte Formulierung des *Schubfachprinzips*, wie es zu Beginn der Vorlesung verwendet wurde. Eine Argumentation, die explizit mit dem Schubfachprinzip arbeitet, sieht so aus:

Angenommen, es existiert ein DEA $A = (\Sigma, Q, s, F, \delta)$ mit $L = L(A)$. Man betrachte die unendlich vielen Wörter $b^n a$ mit $n \in \mathbb{N}$. Da $Q$ endlich ist, müssen mindestens zwei dieser Wörter zum gleichen Zustand führen, d.h. es existieren zwei Zahlen $m \neq n$ mit $\delta^*(s, b^n a) = \delta^*(s, b^m a)$. Dann folgt aber $\delta^*(s, b^n a b^n) = \delta^*(\delta^*(s, b^n a), b^n) = \delta^*(\delta^*(s, b^m a), b^n) = \delta^*(s, b^m a b^n)$, d.h. auch $b^n a b^n$ und $b^m a b^n$ führen zum gleichen Zustand. Das ist ein Widerspruch zu $L = L(A)$ weil $b^n a b^n \in L$ und $b^m a b^n \notin L$. Also existiert kein DEA, der $L$ akzeptiert, d.h. $L$ ist nicht regulär.

Ein beliebter Fehler bei der Anwendung des Satzes von Myhill-Nerode besteht darin, dass man nur zeigt, dass je zwei 'benachbarte' Wörter unter den unendlich vielen ausgewählten $L$-unterscheidbar sind, z.B. bei unserer Sprache die Wörter $b^n a$ und $b^{n+1} a$. Diese Argumentation genügt *nicht*, denn sie beweist nur, dass mindestens zwei $L$-Äquivalenzklassen existieren: Es wäre möglich, dass alle $b^{2n} a$ in der einen und alle $b^{2n+1} a$ in der anderen $L$-Äquivalenzklasse liegen.

## Aufgabe 3

Sei $A = (\Sigma, Q, S, F, \Delta)$ der NDEA aus der Aufgabenstellung. Dann ist der Potenzautomat $A' = (\Sigma, Q', s', F', \delta)$ zu $A$ definiert durch

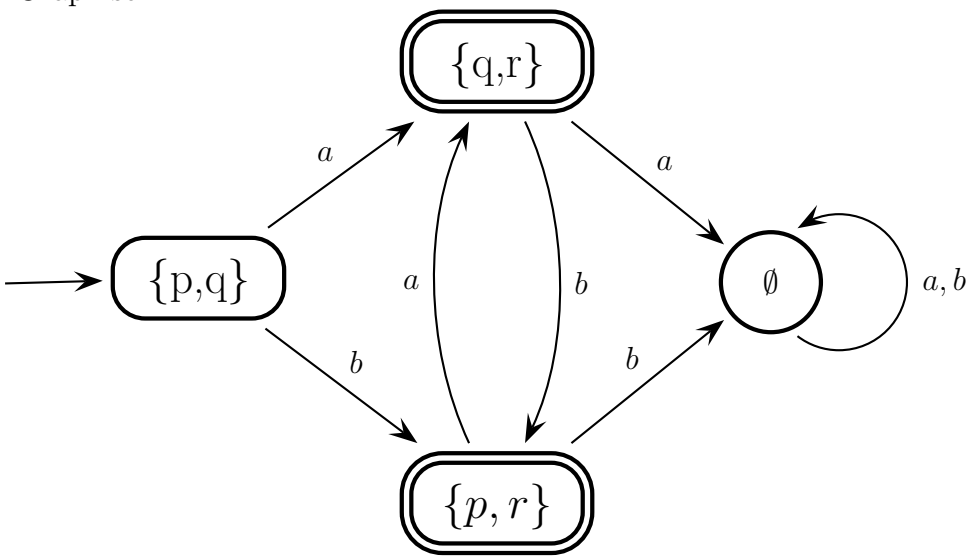
- $Q' = \wp(Q)$
- $s' = S = \{p, q\}$
- $F' = \{P \subseteq Q \mid P \cap F \neq \emptyset\} = \{P \subseteq Q \mid r \in P\}$
- $\delta : Q' \times \Sigma \to Q'$
  $\delta(P, z) = \{q \in Q \mid \exists\, p \in P.\ (p, z, q) \in \Delta\}$

Den erreichbaren Teil von $A'$ erhält man also wie folgt:

- $\delta(s', a) = \delta(\{p, q\}, a) = \{q, r\}, \quad \delta(s', b) = \delta(\{p, q\}, b) = \{p, r\}$
- $\delta(\{p, r\}, a) = \{q, r\}, \quad \delta(\{p, r\}, b) = \emptyset$
- $\delta(\{q, r\}, a) = \emptyset, \quad \delta(\{q, r\}, b) = \{p, r\}$
- $\delta(\emptyset, a) = \emptyset, \quad \delta(\emptyset, b) = \emptyset$

wobei $\{p, r\}$ und $\{q, r\}$ die erreichbaren Endzustände sind.

Graphisch:




Man sollte nicht vergessen, die Endzustände zu markieren. Außerdem sollte man den toten Zustand $\emptyset$ nicht weglassen (wenn er erreichbar ist) und auch nicht die Übergänge von $\emptyset$ nach $\emptyset$ (denn formal gehören sie zum DEA, weil wir fordern, dass $\delta$ eine totale Funktion ist).

## Aufgabe 4

Sei $L \subseteq \Sigma^*$ regulär.

**a.** In den Übungen wurde bereits bewiesen, dass für jede reguläre Sprache $L$ auch die Sprachen $\mathit{Pref}(L)$ und $\mathit{Suff}(L)$ regulär sind, also ist auch $\mathit{Sub}(L) = \mathit{Pref}(\mathit{Suff}(L))$ regulär.

Eine alternative Lösung, bei der man explizit mit endlichen Automaten argumentiert, sieht so aus:

Sei $A = (\Sigma, Q, S, F, \Delta)$ ein NDEA mit $L = L(A)$. Man wähle $A' = (\Sigma, Q, S', F', \Delta)$ mit

- $S' = \{p \in Q \mid \exists\, s \in S, u \in \Sigma^*.\ \delta^*(s, u) = p\}$
  (die Menge aller erreichbaren Zustände)
- $F' = \{q \in Q \mid \exists\, f \in F, v \in \Sigma^*.\ \delta^*(q, v) = f\}$
  (die Menge aller lebendigen Zustände)

Dann gilt $\mathit{Sub}(L) = L(A')$, wobei man für einen exakten Beweis eine ähnliche Argumentation wie für $\mathit{Pref}(L)$ und $\mathit{Suff}(L)$ (Übungsblatt 2, Aufgabe 4) verwenden müsste.

Man beachte, dass man bei der Definition von $A'$ nicht einfach $S' = F' = Q$ wählen kann: Wenn $A$ unerreichbare und/oder tote Zustände enthält, dann könnte der so konstruierte Automat $A'$ Wörter akzeptieren, die *nicht* in $\mathit{Sub}(L)$ liegen.

**b.** Offensichtlich gilt $\mathit{Super}(L) = \Sigma^* \circ L \circ \Sigma^*$. Also ist $\mathit{Super}(L)$ regulär, weil $\Sigma^*$ und $L$ regulär sind und die Klasse der regulären Sprachen abgeschlossen ist unter Konkatenation.

Auch hier könnte man explizit mit Automaten argumentieren. Eine solche Argumentation würde aber darauf hinauslaufen, dass man die Konstruktion aus der Vorlesung wiederholt: Aus einem Automaten $A$ für $L$ und einem Automaten $A'$ für $\Sigma^*$ setzt man einen Automaten für die Konkatenation $\Sigma^* \circ L \circ \Sigma^*$ zusammen. Deshalb ist es sinnvoller, mit den Sprachen selbst zu argumentieren.

## Aufgabe 5

Sei $A_{even}$ ein DEA, der die Sprache $L_{even} = \{w \in \Sigma^* \mid |w| \text{ ist gerade}\}$ erkennt.

**a.** Der Algorithmus konstruiert aus $A$ und $A_{even}$ einen DEA $A'$ mit $L(A') = L(A) \cap L(A_{even}) = \{w \in L(A) \mid |w \text{ ist gerade}\}$ und testet dann (mit dem bereits bekannten Algorithmus), ob $L(A') \neq \emptyset$.

**b.** Der Algorithmus braucht nur zu testen, ob $L(A) \subseteq L(A_{even})$. Ein solcher Algorithmus existiert laut Vorlesung.

Alternative Lösung:

Sei $A_{odd}$ ein DEA, der die Sprache $L_{odd} = \{w \in \Sigma^* \mid |w| \text{ ist ungerade}\}$ erkennt. Dann braucht man nur zu testen, ob $L(A) \cap L(A_{odd}) = \emptyset$.

Natürlich ist es auch möglich (aber nicht ganz so einfach), Algorithmen anzugeben, die *nicht* auf die bereits bekannten Verfahren zurückgreifen: Den aus der Vorlesung bekannten Algorithmus zur Berechnung der erreichbaren Zustände kann man so verfeinern, dass er (simultan) die Mengen

$$\begin{aligned} R^{even} &= \{\delta^*(s,w) \mid w \in \Sigma^* \wedge |w| \text{ ist gerade}\} \text{ und} \\ R^{odd} &= \{\delta^*(s,w) \mid w \in \Sigma^* \wedge |w| \text{ ist ungerade}\} \end{aligned}$$

berechnet. Dann braucht man anschließend nur noch zu überprüfen, ob

**a.** $R^{even} \cap F \neq \emptyset$

**b.** $R^{odd} \cap F = \emptyset$

## Aufgabe 6

Sei $G = (\Sigma, N, S, P)$ mit $\Sigma = \{a, b, c\}$, $N = \{S, A\}$ und

$$P = \{ \begin{array}{lcll} S & \to & aSc, & (1) \\ S & \to & A, & (2) \\ A & \to & bAc, & (3) \\ A & \to & \varepsilon \ \} & (4) \end{array}$$

Es gilt $L = L(G)$, denn:

‘$\subseteq$’: Sei $w = a^m b^n c^{m+n} = a^m b^n c^n c^m$. Dann gilt

$$S \Rightarrow^m a^m S c^m \Rightarrow a^m A c^m \Rightarrow^n a^m b^n A c^n c^m \Rightarrow a^m b^n c^n c^m = w$$

‘$\supseteq$’: Es gelte $S \Rightarrow^* w$. Der letzte Schritt in dieser Ableitung kann nur mit (4) erfolgen. Deshalb muss vorher irgendwann ein Ableitungsschritt mit (2) erfolgt sein. Damit ist die gesamte Ableitung von der Form

$$\begin{array}{lll} S & \Rightarrow^* a^m S c^m & \text{mit (1)} \\ & \Rightarrow a^m A c^m & \text{mit (2)} \\ & \Rightarrow^* a^m b^n A c^n c^m & \text{mit (3)} \\ & \Rightarrow a^m b^n c^n c^m & \text{mit (4)} \end{array}$$

Also ist $w \in L$.

## Aufgabe 7

Sei $L = \{a^m b^n c^{mn} \mid m, n \in \mathbb{N}\}$.

Es genügt zu zeigen, dass für jede Zahl $n \geq 1$ ein Wort $z \in L$ mit $|z| \geq n$ existiert so, dass folgendes gilt: Für jede Zerlegung $z = uvwxy$ mit $vx \neq \varepsilon$ und $|vwx| \leq n$ existiert ein $i \in \mathbb{N}$ mit $uv^i wx^i y \notin L$.

Sei $n \in \mathbb{N}$. Man wähle $z = a^n b^n c^{n^2}$. Dann ist $z \in L$ mit $|z| = n^2 + 2n \geq n$. Sei $z = uvwxy$ mit $vx \neq \varepsilon$ und $|vwx| \leq n$. Wir zeigen, dass das Wort $z_2 = uv^2wx^2y$ nicht in $L$ liegt.

1. Fall: $vx$ enthält nur $c$s, also $vx = c^k$ mit $k > 0$.

Dann ist $z_2 = a^n b^n c^{n^2+k} \notin L$.

2. Fall: $vx$ enthält mindestens eines der Zeichen $a$ oder $b$.

Dann ist $\#_a(z_2) > n$ oder $\#_b(z_2) > n$, also $\#_a(z_2) \cdot \#_b(z_2) \geq n \cdot (n+1) = n^2 + n$. Wegen $|vx| \leq n$ ist außerdem $\#_c(vx) < n$ und damit $\#_c(z_2) < n^2 + n$. Also ist $\#_a(z_2) \cdot \#_b(z_2) > \#_c(z_2)$ und damit $z_2 \notin L$.

Diese Fallunterscheidung ist natürlich sehr trickreich; sie fällt einem sicherlich nicht auf Anhieb ein. Bei einer weniger trickreichen Lösung würde man vielleicht so argumentieren: Wegen $|vx| \leq n$ gibt es nur die folgenden 5 Möglichkeiten:

(a) $vx$ besteht nur aus $a$s

(b) $vx$ besteht nur aus $b$s

(c) $vx$ besteht nur aus $c$s

(d) $vx$ besteht aus $a$s und $b$s

(e) $vx$ besteht aus $b$s und $c$s

Die Fälle (a) bis (d) sind harmlos, denn es ist offensichtlich, dass nach dem Pumpen mit $i = 2$ die Anzahl der $a$s, $b$s und $c$s nicht mehr wie gewünscht zusammenpasst. Also bleibt (e) als einzig schwieriger Fall übrig, in dem man nochmals die Voraussetzung $|vx| \leq n$ ausnutzen muss: Da $vx$ mindestens ein $b$ enthält, erhöht sich das Produkt aus der Anzahl der $a$s und der Anzahl der $b$s beim Pumpen mit $i = 2$ um mindestens $n$, aber die Anzahl der $c$s erhöht sich höchstens um $|vx| - 1 \leq n - 1$. Deshalb passt auch in diesem Fall die Anzahl der $a$s, $b$s und $c$s nach dem Pumpen mit $i = 2$ nicht mehr wie gewünscht zusammen.

Die Überlegungen zum Fall (e) zeigen auch, dass $z$ gut gewählt wurde. Mit einer 'einfacheren' Wahl für $z$ wie etwa $z = ab^n c^n$ oder $z = a^n bc^n$ gelingt der Beweis nicht, denn diese Wörter bilden eine kontextfreie Teilsprache von $L$, aus der man durch Pumpen nicht herauskommt.

Man beachte auch, dass man in den Fällen (d) und (e) *nicht* mit der Reihenfolge der Zeichen argumentieren kann: Wenn $v$ nur aus $b$s und $x$ nur aus $c$s besteht, dann gerät die Reihenfolge der Zeichen beim Pumpen *nicht* durcheinander. Mit der Reihenfolge könnte man nur dann argumentieren, wenn *eines* der Wörter $v$ oder $x$ bereits mehrere Zeichen enthält. Solche Fälle getrennt zu betrachten, lohnt sich aber nicht. Es ist einfacher, nur über die *Anzahl* der Zeichen zu argumentieren.

## Aufgabe 8

Eine passende Maschine ist

$$M = (\{q_1, q_2, m_0, m_1, p, r, h\}, \{0, 1\}, \{0, 1, \$, B\}, q_1, \{h\}, \delta)$$

wobei $\delta$ gegeben ist durch

$$\begin{gathered}
\delta(q_1, a) = (p_1, a, R) \quad \text{für } a \in \{0, 1\} \\
\delta(q_1, B) = (q_2, \$, L) \\
\delta(q_2, \$) = (q_2, \$, L) \\
\delta(q_2, 0) = (m_0, \$, R) \\
\delta(q_2, 1) = (m_1, \$, R) \\
\delta(m_0, a) = (m_0, a, R) \quad \text{für } a \in \{0, 1, \$\} \\
\delta(m_0, B) = (p, 0, L) \\
\delta(m_1, a) = (m_1, a, R) \quad \text{für } a \in \{0, 1, \$\} \\
\delta(m_1, B) = (p, 1, L) \\
\delta(p, a) = (p, a, L) \quad \text{für } a \in \{0, 1\} \\
\delta(p, \$) = (q_2, \$, L) \\
\delta(q_2, B) = (r, B, R) \\
\delta(r, \$) = (r, \$, R) \\
\delta(r, a) = (h, a, N) \quad \text{für } a \in \{0, 1\}
\end{gathered}$$

Die Idee ist, dass wir zuerst einen Marker \$ ans rechte Ende des Wortes schreiben. Danach gehen wir im Zustand $q_2$ soweit nach links, bis wir eine 0 oder eine 1 finden. Im ersten Fall gehen wir in Zustand $m_0$, im zweiten in den Zustand $m_1$; merken uns also, welches Zeichen wir gefunden haben. In beiden Fällen überschreiben wir das Zeichen mit \$ und gehen ans rechte Ende, wo wir das gemerkte Zeichen hinschreiben. Im Zustand $p$ suchen wir jetzt nach links das nächste \$ und gehen dann wieder in den Zustand $q_2$. Finden wir irgendwann im Zustand $q_2$ das Blanksymbol sind wir fast fertig und müssen nur noch den Lese-/Schreibkopf an die richtige Stelle bewegen.

## Aufgabe 9

Die Idee ist, dass wiederholt 1 von der Eingabe abziehen und dabei jeweils eine Hilfsvariable zwischen 0 und 1 wechseln lassen. Das Programm

```
LOOP x1 DO
    x3 := 1;
    LOOP x2 DO x3 := x3 − 1 END;
    x2 := x3;
END
x1 := x2
```

ist schon fast korrekt, berechnet allerdings die charakteristische Funktion der ungeraden Zahlen. Es muss also noch um die erste Zeile $x_1 := x_1 + 1$ ergänzt werden.

## Aufgabe 10

Siehe Übungen.

## Aufgabe 11

Sei $h(i,j) = j^i$. Da $h$ $\mu$-rekursiv ist, gibt es einen Index $e$, so dass $u(e, \langle i,j \rangle) = h(i,j)$. Mit dem $s$-$m$-$n$-Theorem gilt dann:

$$j^i = h(i,j) = u(e, \langle i,j \rangle) = u(s(e,1,i), j) \ ;$$

also ist $g = \lambda i.s(e,1,i)$ die gesuchte Funktion.