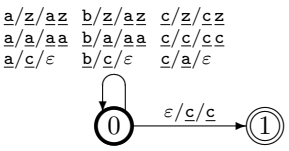
Theoretische Informatik 1 – Übungsblatt 5 (SS2006)
Lösungen

Aufgabe 5.1 Sei L die Sprache $\{w \in \{\underline{\mathtt{a}}, \underline{\mathtt{b}}, \underline{\mathtt{c}}\}^* \mid n_{\underline{\mathtt{a}}}(w) + n_{\underline{\mathtt{b}}}(w) < n_{\underline{\mathtt{c}}}(w)\}$, wobei der Ausdruck $n_s(w)$ die Anzahl der Vorkommnisse des Symbols s im Wort w bezeichnet. Geben Sie einen Kellerautomaten $\mathcal{K}$ an, der L akzeptiert, d.h., für den $\mathcal{L}(\mathcal{K}) = L$ gilt.

Lösung Wir zählen im Keller mit, ob bisher mehr $\underline{\mathtt{a}}$'s und $\underline{\mathtt{b}}$'s oder ob mehr $\underline{\mathtt{c}}$'s vorgefunden wurden. Wir benötigen neben $\underline{\mathtt{z}}$ zwei Kellersymbole, um zwischen einem $\underline{\mathtt{a}}/\underline{\mathtt{b}}$-Überhang bzw. einem $\underline{\mathtt{c}}$-Überhang unterscheiden zu können. Ein Wort wird akzeptiert, wenn der Keller am Wortende einen $\underline{\mathtt{c}}$-Überhang anzeigt.



In Tupelnotation: $\mathcal{K} = \langle\{0,1\}, \{\underline{\mathtt{a}}, \underline{\mathtt{b}}, \underline{\mathtt{c}}\}, \{\underline{\mathtt{a}}, \underline{\mathtt{c}}, \underline{\mathtt{z}}\}, \delta, 0, \underline{\mathtt{z}}, \{1\}\rangle$, wobei die Übergangsfunktion δ definiert ist durch

$$\begin{array}{llll}
\delta(0,\underline{\mathtt{a}},\underline{\mathtt{z}}) = \{(0,\underline{\mathtt{a}}\underline{\mathtt{z}})\} & \delta(0,\underline{\mathtt{b}},\underline{\mathtt{z}}) = \{(0,\underline{\mathtt{a}}\underline{\mathtt{z}})\} & \delta(0,\underline{\mathtt{c}},\underline{\mathtt{z}}) = \{(0,\underline{\mathtt{b}}\underline{\mathtt{z}})\} & \delta(0,\varepsilon,\underline{\mathtt{c}}) = \{(1,\underline{\mathtt{c}})\} \\
\delta(0,\underline{\mathtt{a}},\underline{\mathtt{a}}) = \{(0,\underline{\mathtt{a}}\underline{\mathtt{a}})\} & \delta(0,\underline{\mathtt{b}},\underline{\mathtt{a}}) = \{(0,\underline{\mathtt{a}}\underline{\mathtt{a}})\} & \delta(0,\underline{\mathtt{c}},\underline{\mathtt{b}}) = \{(0,\underline{\mathtt{b}}\underline{\mathtt{b}})\} & \\
\delta(0,\underline{\mathtt{a}},\underline{\mathtt{c}}) = \{(0,\varepsilon)\} & \delta(0,\underline{\mathtt{b}},\underline{\mathtt{c}}) = \{(0,\varepsilon)\} & \delta(0,\underline{\mathtt{c}},\underline{\mathtt{a}}) = \{(0,\varepsilon)\} &
\end{array}$$

Aufgabe 5.2 Die Syntax der fiktiven, regelbasierten Datenbanksprache RDS sei folgendermaßen definiert. Ein *Programm* ist eine Folge von bedingten und atomaren Aussagen in beliebiger Reihenfolge, die voneinander durch Strichpunkte getrennt sind; die Folge endet mit einem Punkt. Eine *bedingte Aussage* besteht aus einer atomaren Aussage gefolgt von den Zeichen $\underline{\leq=}$, denen wiederum eine nicht-leere Folge von atomaren Aussagen folgt, die untereinander durch Beistriche getrennt sind. Eine *atomare Aussage* ist ein Name, dem optional eine Argumentliste folgt. Eine *Argumentliste* ist eine in runden Klammern eingeschlossene, nicht-leere Folge von Namen und Variablen in beliebiger Reihenfolge; die einzelnen Namen und Variablen sind durch Beistriche voneinander getrennt. Ein *Name* ist eine Folge von Buchstaben und Ziffern, die mit einem Kleinbuchstaben beginnt. Eine *Variable* ist eine Folge von Buchstaben und Ziffern, die mit einem Großbuchstaben beginnt.

Beispiel: Das RDS-Programm

$$\underline{\mathtt{vater}}\underline{(}\underline{\mathtt{max}}\underline{,}\underline{\mathtt{tim}}\underline{)}\underline{;}\ \underline{\mathtt{vater}}\underline{(}\underline{\mathtt{tim}}\underline{,}\underline{\mathtt{tom}}\underline{)}\underline{;}\ \underline{\mathtt{grossvater}}\underline{(}\underline{\mathtt{X}}\underline{,}\underline{\mathtt{Z}}\underline{)}\underline{\leq=}\underline{\mathtt{vater}}\underline{(}\underline{\mathtt{X}}\underline{,}\underline{\mathtt{Y}}\underline{)}\underline{,}\underline{\mathtt{vater}}\underline{(}\underline{\mathtt{Y}}\underline{,}\underline{\mathtt{Z}}\underline{)}\underline{.}$$

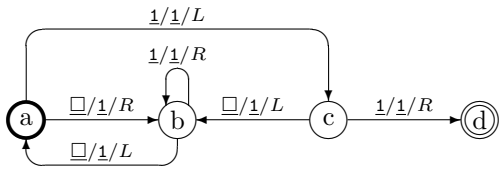
besteht aus zwei atomaren und einer bedingten Aussage. Die bedingte Aussage besteht wiederum aus drei atomaren Aussagen, einer vor den Symbolen $\underline{\leq=}$ und zweien danach.

Geben Sie für die Sprache der RDS-Programme eine kontextfreie Grammatik in EBNF an. Verwenden Sie so weit als möglich EBNF-Notationen, um die Grammatik übersichtlich zu halten.

Lösung Eine mögliche Lösung ist $\langle V, T, P, Prog\rangle$ mit

$$
\begin{array}{lcll}
V & = & \{Prog, Aussage, Bedingt, Atomar, Argliste, NV, Name, Var, GB, KB, Ziff\} & , \\
T & = & \{\underline{\mathtt{A}}, \ldots, \underline{\mathtt{Z}}, \underline{\mathtt{a}}, \ldots, \underline{\mathtt{z}}, \underline{\mathtt{0}}, \ldots, \underline{\mathtt{9}}, \underline{.}, \underline{,}, \underline{;}, \underline{\leq}, \underline{=}, \underline{(}, \underline{)}\} & , \\
P & = & \{Prog & \Rightarrow \quad Aussage\ \{\ \underline{;}\ Aussage\ \}\ \underline{.}\ , \\
 & & Aussage & \Rightarrow \quad Bedingt \mid Atomar\ , \\
 & & Bedingt & \Rightarrow \quad Atomar \underline{\leq} \underline{=} Atomar\ \{\ \underline{,}\ Atomar\ \}\ , \\
 & & Atomar & \Rightarrow \quad Name\ [\ Argliste\ ]\ , \\
 & & Argliste & \Rightarrow \quad \underline{(}\ NV\ \{\ \underline{,}\ NV\ \}\ \underline{)}\ , \\
 & & NV & \Rightarrow \quad Name \mid Var\ , \\
 & & Name & \Rightarrow \quad KB\ \{\ KB \mid GB \mid Ziff\ \}\ , \\
 & & Var & \Rightarrow \quad GB\ \{\ KB \mid GB \mid Ziff\ \}\ , \\
 & & GB & \Rightarrow \quad \underline{\mathtt{A}} \mid \cdots \mid \underline{\mathtt{Z}}\ , \\
 & & KB & \Rightarrow \quad \underline{\mathtt{a}} \mid \cdots \mid \underline{\mathtt{z}}\ , \\
 & & Ziff & \Rightarrow \quad \underline{\mathtt{0}} \mid \cdots \mid \underline{\mathtt{9}}\ \}\ .
\end{array}
$$

Aufgabe 5.3 Sei $\mathcal{T}$ die folgende Turing-Maschine:



Die Kantenbeschriftung $x/y/z$ bedeutet, dass beim Zustandsübergang das Symbol x vom Band gelesen und durch das Symbol y überschrieben wird und sich der Schreib-/Lesekopf anschließend in Richtung z bewegt. $\underline{\square}$ bezeichnet das Leersymbol.

a) Beschreiben Sie die Turing-Maschine in Tupel-Notation.

b) Geben Sie alle Konfigurationen an, die die Turing-Maschine durchläuft, wenn sie auf dem leeren Band gestartet wird.

Lösung Die Turing-Maschine wird durch das 7-Tupel $\langle\{a, b, c, d\}, \{\underline{1}\}, \{\underline{1}, \underline{\square}\}, \delta, a, \underline{\square}, \{d\}\rangle$ beschrieben, wobei die Übergangsfunktion definiert ist durch die Tabelle

δ	$\underline{\square}$	$\underline{1}$
a	$(b, \underline{1}, R)$	$(c, \underline{1}, L)$
b	$(a, \underline{1}, L)$	$(b, \underline{1}, R)$
c	$(b, \underline{1}, L)$	$(d, \underline{1}, R)$

Die Angabe macht keine Aussage über das Eingabealphabet; man hätte statt $\{\underline{1}\}$ auch $\{\}$ festlegen können.
Auf dem leeren Band gestartet ergibt sich folgende Konfigurationsfolge:

$$
\begin{array}{llllllll}
a & \vdash \underline{1}\,b & \vdash a\,\underline{1}\,\underline{1} & \vdash c\,\underline{\square}\,\underline{1}\,\underline{1} & \vdash b\,\underline{\square}\,\underline{1}\,\underline{1}\,\underline{1} & & & \\
 & \vdash a\,\underline{\square}\,\underline{1}\,\underline{1}\,\underline{1}\,\underline{1} & \vdash \underline{1}\,b\,\underline{1}\,\underline{1}\,\underline{1}\,\underline{1} & \vdash \underline{1}\,\underline{1}\,b\,\underline{1}\,\underline{1}\,\underline{1} & \vdash \underline{1}\,\underline{1}\,\underline{1}\,b\,\underline{1}\,\underline{1} & & & \\
 & \vdash \underline{1}\,\underline{1}\,\underline{1}\,\underline{1}\,b\,\underline{1} & \vdash \underline{1}\,\underline{1}\,\underline{1}\,\underline{1}\,\underline{1}\,b & \vdash \underline{1}\,\underline{1}\,\underline{1}\,\underline{1}\,a\,\underline{1}\,\underline{1} & \vdash \underline{1}\,\underline{1}\,\underline{1}\,c\,\underline{1}\,\underline{1}\,\underline{1} & \vdash \underline{1}\,\underline{1}\,\underline{1}\,\underline{1}\,d\,\underline{1}\,\underline{1} & &
\end{array}
$$

Anmerkung: Diese Turing-Maschine ist ein *Busy Beaver* mit drei Zuständen (der Endzustand wird nicht mitgezählt). Ein Busy Beaver ist eine Turing-Maschine, die auf dem leeren Band gestartet irgendwann hält (also den Endzustand erreicht) und unter allen derartig haltenden Turing-Maschinen mit derselben Zustandszahl die meisten Einser auf das Band schreibt. Sei E die Funktion, die jeder Zustandszahl die maximale Zahl der Einser zuordnet; es gilt also etwa $E(3) = 6$. Die Funktion E wächst sehr stark. Man kann zeigen, dass E nicht berechenbar ist, d.h., es gibt keine Turing-Maschine (und auch kein Computerprogramm), die für beliebiges n den Funktionswert

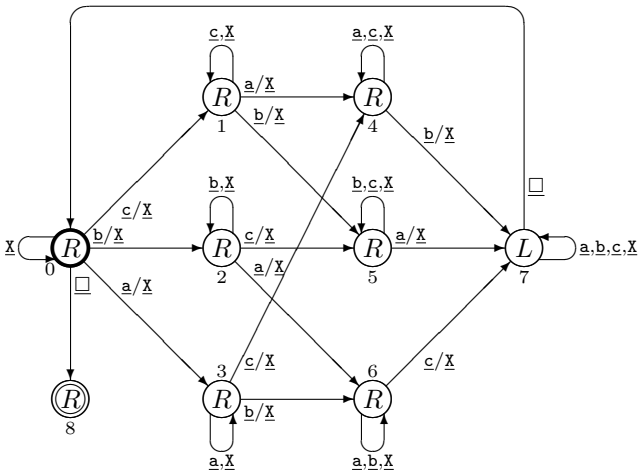
$E(n)$ berechnet. Busy Beaver sind schwer zu finden: Einerseits wächst die Zahl der verschiedenen Turing-Maschinen exponentiell mit der Zahl der Zustände, andererseits lässt sich im Allgemeinen nicht feststellen, ob eine Turing-Maschine irgendwann anhalten oder in eine Endlosschleife geraten wird. Einige bekannte Werte von E:

$$\begin{array}{ll} E(1) = 1 & E(4) = 13 \\ E(2) = 4 & E(5) \geq 4098 \\ E(3) = 6 & E(6) \geq 95{,}524{,}079 \end{array}$$

Die Bedeutung dieser Funktion liegt darin, dass sie ein Beispiel für eine einfache nicht-berechenbare Funktion darstellt.

Aufgabe 5.4 Sei L die Sprache $\{w \in \{\underline{\mathtt{a}}, \underline{\mathtt{b}}, \underline{\mathtt{c}}\}^* \mid n_{\underline{\mathtt{a}}}(w) = n_{\underline{\mathtt{b}}}(w) = n_{\underline{\mathtt{c}}}(w)\}$, d.h., L enthält alle Worte, die dieselbe Anzahl an $\underline{\mathtt{a}}$'s, $\underline{\mathtt{b}}$'s und $\underline{\mathtt{c}}$'s enthalten. Geben Sie eine Turing-Maschine $\mathcal{T}$ an, die L akzeptiert.

Lösung Die unten angegebene Turing-Maschine arbeitet folgendermaßen. Das Wort auf dem Eingabeband wird mehrmals von links nach rechts durchlaufen, wobei in jedem Durchlauf jeweils das erste vorgefundene $\underline{\mathtt{a}}$, $\underline{\mathtt{b}}$ und $\underline{\mathtt{c}}$ gelöscht wird. Das Löschen eines Symbols wird dadurch bewerkstelligt, dass es durch das Hilfssymbol $\underline{\mathtt{X}}$ ersetzt wird, welches in jedem Zustand ignoriert wird (kenntlich an der mit $\underline{\mathtt{X}}$ beschrifteten Schleife in jedem Zustand). Die Symbole $\underline{\mathtt{a}}$, $\underline{\mathtt{b}}$ und $\underline{\mathtt{c}}$ können in sechs verschiedenen Reihenfolgen auftreten: $\underline{\mathtt{a}} \cdots \underline{\mathtt{b}} \cdots \underline{\mathtt{c}}$, $\underline{\mathtt{a}} \cdots \underline{\mathtt{c}} \cdots \underline{\mathtt{b}}$, $\underline{\mathtt{b}} \cdots \underline{\mathtt{c}} \cdots \underline{\mathtt{a}}$, usw. Dementsprechend gibt es sechs Pfade vom Startzustand 0 zum Zustand 7. Im Zustand 7 wird der Kopf wieder nach links an den Anfang des Wortes bewegt, danach beginnt ein neuer Durchlauf. Der Startzustand 0 kontrolliert am Ende auch, ob die Anzahl der $\underline{\mathtt{a}}$'s, $\underline{\mathtt{b}}$'s und $\underline{\mathtt{c}}$'s gleich groß war: In diesem Fall muss nämlich nach dem Überspringen der gelöschten Symbole ein Leersymbol auftreten, was zum Übergang in den Endzustand 8 führt.

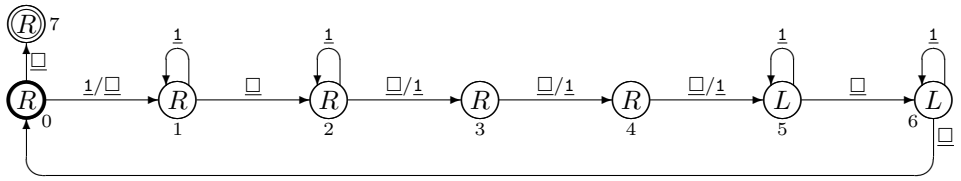


Zur Illustration die Konfigurationsfolge, die beim Wort $\underline{\mathtt{c}}\underline{\mathtt{a}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{b}}\underline{\mathtt{b}}$ durchlaufen wird (spaltenweise zu lesen).

$$\begin{array}{r} 0\underline{\mathtt{c}}\underline{\mathtt{a}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{b}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}1\underline{\mathtt{a}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{b}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}4\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{b}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}4\underline{\mathtt{c}}\underline{\mathtt{b}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}4\underline{\mathtt{b}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}7\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}7\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}7\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ 7\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ 7\square\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \end{array} \qquad \begin{array}{r} 0\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}0\underline{\mathtt{X}}\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}0\underline{\mathtt{a}}\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}3\underline{\mathtt{c}}\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}4\underline{\mathtt{X}}\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}4\underline{\mathtt{b}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}7\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}7\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}7\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}7\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ 7\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ 7\square\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \end{array} \qquad \begin{array}{r} 0\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}0\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}0\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}0\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}0\underline{\mathtt{X}}\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}0\underline{\mathtt{X}} \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}0\square \\ \underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\underline{\mathtt{X}}\square 8 \end{array}$$

Aufgabe 5.5 Zeigen Sie: Die Funktion $f(n) = 3n$ für $n \geq 0$ ist Turing-berechenbar.

Lösung Wir müssen zunächst ein Kodierungsschema festlegen, um die Eingabezahl in eine Zeichenkette auf dem Band der Turing-Maschine umwandeln und dann das Ergebnis der Turing-Maschine (also die Zeichen auf dem Band nach Halten der Turing-Maschine) wieder als Zahl interpretieren zu können. Wir wählen die Unärkodierung, d.h., die Zahl n wird als Zeichenkette bestehend aus n Einsern dargestellt (es könnte auch jedes andere Symbol außer dem Leersymbol sein). Um zu zeigen, dass die Funktion f Turing-berechenbar ist, müssen wir also eine Turing-Maschine angegen, die n Einser auf dem Band in $3n$ Einser umwandelt und dann hält, wobei der Schreib-/Lesekopf zu Beginn und am Ende über dem ersten Einser steht.
Die folgende Turing-Maschine funktioniert folgendermaßen. Der erste Einser der Eingabe wird gelöscht, indem er durch ein Leersymbol ersetzt wird. Danach werden rechts der Eingabe – durch ein Leersymbol getrennt – drei Einser geschrieben. Dann bewegt sich der Kopf wieder nach links zum Beginn der Eingabe. Sobald alle Eingabeeinser abgearbeitet sind, hält die Turing-Maschine.



Zur Illustration die Berechnung von $f(2)$ (spaltenweise zu lesen):

0 1 1 □□□□□□□	□0 1 □ 1 1 1 □□□	□□0□ 1 1 1 1 1 1
□1 1 □□□□□□□	□□1□ 1 1 1 □□□	□□□7 1 1 1 1 1 1
□ 1 1□□□□□□□	□□□2 1 1 1 □□□	
□ 1 □2□□□□□□	□□□ 1 2 1 1 □□□	
□ 1 □ 1 3□□□□□	□□□ 1 1 2 1 □□□	
□ 1 □ 1 1 4□□□□	□□□ 1 1 1 2□□□	
□ 1 □ 1 5 1 1□□□	□□□ 1 1 1 1 3□□	
□ 1 □5 1 1 1□□□	□□□ 1 1 1 1 1 4□	
□ 1 5□ 1 1 1□□□	□□□ 1 1 1 1 5 1 1	
□6 1 □ 1 1 1□□□	□□□ 1 1 1 5 1 1 1	
6□ 1 □ 1 1 1□□□	□□□ 1 5 1 1 1 1 1	
	□□□5 1 1 1 1 1 1	
	□□5□ 1 1 1 1 1 1	
	□6□□ 1 1 1 1 1 1	

Zwecks Übersicht geben wir bei jeder Konfiguration den Inhalt aller neun Felder an, die von der Turing-Maschine im Laufe der Abarbeitung gelesen/beschrieben werden.