Die Turing-Maschine im praktischen Einsatz

Dargestellt ist jeweils die Zustands- bzw. Übergangstabelle. Die Zustände sind mit Z1, Z2,... bezeichnet. S ist der Zustand, der die Maschine stoppt. L bedeutet eine Bewegung des Schreib-Lese-Kopfes nach links, R eine Bewegung nach rechts. Die Turing-Maschine soll immer im Zustand Z1 starten.

1. Beispiel:

Gelesenes Zeichen	0			1		
Zustand	Schreibe	Neuer Zustand	Nach	Schreibe	Neuer Zustand	Nach
Z1	1	Z2	L	0	Z1	L
Z2	0	S		1	Z2	R
S (Stopp)						

Die folgenden Bänder sollen benutzt werden, wobei zur Orientierung eine Positionsnummer angegeben ist, die für die Turing-Maschine irrelevant ist. Die Maschine starte jeweils auf der fettgedruckten Ziffer.

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	0	0	0	0	0	0	0	0	**0**	0	0	0

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	0	0	0	0	0	0	0	0	**1**	0	0	0

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	0	0	0	0	0	0	0	1	**1**	0	0	0

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	0	0	0	1	0	1	1	1	**1**	0	0	0

Was bewirkt diese Turing-Maschine?

2. Beispiel:

Gelesenes Zeichen	0			1		
Zustand	Schreibe	Neuer Zustand	Nach	Schreibe	Neuer Zustand	Nach
Z1	1	Z2	R	1	Z3	L
Z2	1	Z1	L	1	Z2	R
Z3	1	Z2	L	1	S	
S (Stopp)						

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	0	0	0	0	0	0	**0**	0	0	0	0	0

Die Vorgänge macht man sich am besten mit einer Programmablauftabelle klar.

Positionsnr.	Zustand	gelesenes Zeichen	zu schreibendes Zeichen	Neuer Zustand	Kopf-bewegung
7	Z1	0			

3. Beispiel:

Gelesenes Zeichen	0			1		
Zustand	Schreibe	Neuer Zustand	Nach	Schreibe	Neuer Zustand	Nach
Z1	0	Z1	R	0	Z2	R
Z2	1	Z3	R	1	Z2	R
Z3	0	S		1	S	
S (Stopp)						

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	**0**	0	1	1	1	1	0	1	1	0	0	0

Die Vorgänge macht man sich wieder am besten mit einer Programmablauftabelle klar.

Positionsnr.	Zustand	gelesenes Zeichen	zu schreibendes Zeichen	Neuer Zustand	Kopf-bewegung
1	Z1	0			

4. Beispiel:

Beim vorliegenden Band soll an die 1er-Kette rechts eine 1 angefügt werden. Bestimme eine geeignete Zustands- bzw. Übergangstabelle zur Lösung dieses Problems.

Positionsnr.	01	02	03	04	05	06	07	08	09	10	11	12
Inhalt	**0**	0	0	1	1	1	1	0	0	0	0	0

Gelesenes Zeichen	0			1		
Zustand	Schreibe	Neuer Zustand	Nach	Schreibe	Neuer Zustand	Nach
Z1						
Z2						
Z3						
S (Stopp)						

Erweiterung: Stelle diese Turing-Maschine mit KARA dar. Ein Kleeblatt sei eine 1, ein leeres Feld eine 0. Zur Erinnerung: Kara ist nicht Javakara, sondern arbeitet als endlicher Automat mit Zuständen.
Besser: Benutze **Turing-Kara!**

Fleißige Biber (busy beaver):

Turing-Maschinen, die entlang ihres Bandes hin- und herlaufen, hier ein Symbol unverändert lesen und dort ein Symbol schreiben, könnten uns an fleißige Biber erinnern, die den Fluss zwischen Damm und Wald fleißig durchqueren und bei jeder Tour ein Stöckchen (eine 1) für die Konstruktion ihres Dammes heranschaffen. Wie fleißig kann nun eine Turing-Maschine sein? Wie viele Stöckchen können sie legen? Dabei ist aber zu bedenken, dass eine Turing-Maschine anhalten muss. Eine Maschine zu entwerfen, die unendlich viele 1en auf ein Band schreibt, ist nicht schwer. Wie ist es aber, wenn die Maschine nur möglichst viele 1en schreiben soll und irgendwann anhalten soll?
TIBOR RADO, ein ungarischer Mathematiker, dachte sich das aus, was heute als *Busy-beaver-Problem* bezeichnet wird: Gegeben sei eine Turing-Maschine mit *n* Zuständen und einem zweielementigen Alphabet bestehend aus 0 und 1. Was ist nun die maximale Anzahl an Einsen, die eine solche Turing-Maschine auf das Band schreiben kann?
Der augenblickliche Wissensstand (2002) sieht so aus:

Anzahl der Zustände	maximale Anzahl der 1en
1	1
2	4
3	6
4	13
5	≥ 4098

1984 erschien in *Scientific American* ein Artikel über den damals fleißigsten bekannten 5-Zustands-Biber von UWE SCHULT, einem deutschen Informatiker. Sein Biber konnte 501 Einsen vor dem Anhalten erzeugen. Als Antwort auf den Artikel führte GEORGE UHING, ein amerikanischer Programmierer, eine Computersuche nach fleißigen Bibern mit 5 Zuständen durch und fand einen,

der 1915 Einsen produzieren konnte. 1989 führten JÜRGEN BUNTROCK und HEINER MARXEN in Deutschland auf einem Hochgeschwindigkeitsrechner in Deutschland ein Programm zur Suche ein, dass nach drei Tagen einen Biber mit 4096 Einsen fand. Hier ist seine Turing-Tabelle:

Gelesenes Zeichen	0			1		
Zustand	Schreibe	Neuer Zustand	Nach	Schreibe	Neuer Zustand	Nach
Z1	1	Z2	L	1	Z1	L
Z2	1	Z3	R	1	Z2	R
Z3	1	Z1	L	1	Z4	R
Z4	1	Z1	L	1	Z5	R
Z5	1	S	R	0	Z3	R
S (Stopp)						

Es ist natürlich berechtigt, zu fragen, was „dieser Quatsch“ soll, mit dem sich Informatiker mit „viel Freizeit“ beschäftigen. Ich kann mich selbst daran erinnern, dass zu der Zeit, als ich in die Informatik Anfang der 80er Jahre einstieg, in diversen Computerzeitschriften Busy-Beaver-Kolumnen existierten, was ich damals überhaupt nicht verstehen konnte und wollte.
Warum ist dieses Problem nun so berühmt geworden? Die Antwort ist kurz und ernüchternd:

> Das Busy-beaver-Problem ist mit einer Turing-Maschine nicht berechenbar und damit überhaupt nicht berechenbar.

Kein Computer kann jemals herausfinden, wieviel Stöckchen ein fleißiger Biber mit einer bestimmten Anzahl von Zuständen maximal legen kann. Die Informatik hat also ihre Grenzen und damit muss man sich leider abfinden.
Der Beweis des Satzes sei auf das Informatik-Studium verschoben.