Einige Beispiele zur Turingmaschine

Beispiel 1: Addition von 1 zu einer Dualzahl

Aufgabe:

> Auf dem Eingabe-Band einer Turingmaschine steht eine Dualzahl (= Binärzahl, bestehend aus 0-en und 1-en, links steht die höchstwertigste Ziffer, rechts die niederwertigste, jenseits der Zahl stehen links und rechts nur (unendlich viele) Blank-Zeichen („#“)).
>
> Diese Zahl soll um 1 erhöht werden.
>
> Am Ende soll der Lese-/Schreibkopf auf der ersten (linkesten) Stelle der Zahl stehen.

$$\ldots\ \#\#\#\ 1\ 0\ \overset{\downarrow}{0}\ \#\#\#\ \ldots \quad\rightarrow\quad \ldots\ \#\#\#\ \overset{\downarrow}{1}\ 0\ 1\ \#\#\#\ \ldots$$

$$\ldots\ \#\#\#\ 1\ 1\ \overset{\downarrow}{1}\ \#\#\#\ \ldots \quad\rightarrow\quad \ldots\ \#\#\ \overset{\downarrow}{1}\ 0\ 0\ 0\ \#\#\#\ \ldots$$

Programm-Skizze:

a) O.B.d.A. soll der Lese-/Schreibkopf der Turingmaschine zu Anfang bereits auf der rechtesten (also niederwertigsten) Ziffer der Dualzahl stehen.

   (Keine wirkliche Einschränkung: Wenn wir am Anfang auf einer beliebigen Ziffer (0 oder 1) stehen, müssten wir sonst nur solange nach rechts gehen, bis wir an ein Blank (#) kommen – und dann wieder eins zurück, da wir nun eins zu weit gegangen sind.)

b) Falls wir an dieser Stelle eine 0 auf dem Band vorfinden,

   > können wir diese einfach durch eine 1 ersetzen und sind bereits (fast) fertig
   > (wir müssen noch bis ganz nach links gehen – aber dies erfolgt später ab Schritt (d)).

c) Falls wir an dieser Stelle eine 1 auf dem Band vorfinden,

   > können wir diese durch eine 0 ersetzen,
   > verbleiben jedoch diesmal im „Zustand“, dass die Addition noch nicht vollendet wurde
   > und gehen mit dem Lese-/Schreibkopf um eine Stelle nach links.
   > Ab hier wiederholt sich das Ganze ab Schritt (b).

d) Nachdem die eigentliche Addition abgeschlossen ist, gehen wir mit dem Lese-/Schreibkopf Schritt für Schritt nach Links, bis das erste Blank (#) erreicht ist.

e) Dann müssen wir jedoch wieder einen Schritt zurück (nach rechts) gehen, da wir einen Schritt zu weit gegangen sind.

   (Den mussten wir jedoch zu weit gehen, da wir sonst das Blank-Zeichen nicht hätten sehen können.)

Turing-Programm:

Zu Anfang befindet sich der Lese-/Schreibkopf ganz rechts auf der letzten (niederwertigsten) Ziffer der Zahl; die Maschine befindet sich zu Anfang im Zustand q_0.

$q_0\ 0 \quad \rightarrow \quad 1\ L\ q_1$

Wenn sich die Maschine im Zustand q_0 befindet und eine 0 auf dem Band steht, wird diese durch 1 ersetzt, der Lese-/Schreibkopf fährt um einen Schritt nach links und die Maschine wechselt in den Zustand q_1 (von jetzt ab ist die Addition abgeschlossen).

$\left.\begin{array}{l} q_1\ 0 \quad \rightarrow \quad 0\ L\ q_1 \\ q_1\ 1 \quad \rightarrow \quad 1\ L\ q_1 \end{array}\right\}$

Wenn sich die Maschine im Zustand q_1 befindet (unabhängig, ob 0 oder 1 auf dem Band steht), fährt der Lese-/Schreibkopf um einen Schritt nach links (und die Maschine verbleibt im Zustand q_1).

$q_1\ \# \quad \rightarrow \quad \#\ R\ e$

Wenn sich die Maschine im Zustand q_1 befindet und ein Blank-Zeichen (#) auf dem Band steht, so fährt der Lese-/Schreibkopf um einen Schritt nach rechts und die Maschine geht in den Endzustand e über.

$\left.\begin{array}{l} e\ 0 \quad \rightarrow \quad 0\ H\ e \\ e\ 1 \quad \rightarrow \quad 1\ H\ e \\ e\ \# \quad \rightarrow \quad \#\ H\ e \end{array}\right\}$

Wenn sich die Maschine im Endzustand e befindet (unabhängig, was gerade auf dem Band steht), so Hält die Maschine (und verbleibt auch im Endzustand e).

$q_0\ 1 \quad \rightarrow \quad 0\ L\ q_0$

Wenn sich die Maschine im Zustand q_0 befindet und eine 1 auf dem Band steht, wird diese durch 0 ersetzt, der Lese-/Schreibkopf fährt um einen Schritt nach links und die Maschine verbleibt im Zustand q_0 (die Addition ist noch nicht abgeschlossen).

$q_0\ \# \quad \rightarrow \quad 1\ H\ e$

Wenn sich die Maschine im Zustand q_0 befindet und ein Blank-Zeichen (#) auf dem Band steht (d.h. der Übertrag bei der Addition setzte sich stellenweise bis zum linken Rand fort), wird dieses durch 1 ersetzt, die Maschine geht in den Endzustand e über und hält.

Dieses Programm könnte man auch etwas übersichtlicher in Tabellenform schreiben:

	0	1	#	Eingabe
q_0	$1, L, q_1$	$0, L, q_0$	$1, H, e$	
q_1	$0, L, q_1$	$1, L, q_1$	$\#, R, e$	
e	$0, H, e$	$1, H, e$	$\#, H, e$	
Zustände				

Beispiel 2: Verdopplungsmaschine

Aufgabe:

> Eine Anzahl von 0-en auf dem Band eingeschlossen zwischen zwei 1-en soll von der Maschine verdoppelt werden.
>
> $$\begin{array}{rcl} \#\,1\,0\,1\,\# & \rightarrow & \#\,1\,0\,0\,1\,\# \\ \#\,1\,0\,0\,1\,\# & \rightarrow & \#\,1\,0\,0\,0\,0\,1\,\# \\ \#\,1\,0\,0\,0\,1\,\# & \rightarrow & \#\,1\,0\,0\,0\,0\,0\,0\,1\,\# \\ & \ldots & \end{array}$$

Arbeitsweise der Verdopplungsmaschine (Programm-Skizze):

a) Der Lese-/Schreibkopf wird auf einer Zelle mit nicht-leerem Symbol angesetzt.
Vor dort bewegt er sich zur linken 1 und streicht diese.

b) Der Lese-/Schreibkopf geht 1 Zelle nach rechts.

c) Findet er dort eine 1,

> so gehe bis zur ersten leeren Zelle nach rechts,
> setzt dort 1, stop

d) Findet er dort eine 0,

> so streicht er diese,
> geht bis zur ersten leeren Zelle nach rechts,
> setzt dort 0
> geht um 1 Zelle nach rechts
> setzt dort 0
> geht bis zur ersten nicht-leeren Zelle nach links
> und fährt fort bei Schritt (c).

Turing-Programm:

Der Lese-/Schreibkopf wird auf einer Zelle mit nicht-leerem Symbol angesetzt.

$q_0\ 0 \rightarrow 0\ L\ q_0$		Von dort bewegt er (der Lese-/Schreibkopf) sich nach links,
$q_0\ 1 \rightarrow \#\ R\ q_1$		bis er eine 1 findet und streicht diese. Der Lese-/Schreibkopf geht eine Zelle nach rechts.
$q_1\ 1 \rightarrow 1\ R\ q_2$		Findet er dort eine 1,
$q_2\ 0 \rightarrow 0\ R\ q_2$ $q_2\ 1 \rightarrow 1\ R\ q_2$	}	so geht er bis zur ersten leeren Zelle nach rechts,
$q_2\ \# \rightarrow 1\ H\ e$		setzt dort eine 1, stop.
$q_1\ 0 \rightarrow \#\ R\ q_3$		Findet er dort eine 0, so streicht er diese,
$q_3\ 0 \rightarrow 0\ R\ q_3$ $q_3\ 1 \rightarrow 1\ R\ q_3$	}	geht bis zur ersten leeren Zelle nach rechts,
$q_3\ \# \rightarrow 0\ R\ q_4$		setzt dort eine 0, geht um eine Zelle nach rechts,
$q_4\ \# \rightarrow 0\ L\ q_5$		setzt dort eine 0,
$q_5\ 0 \rightarrow 0\ L\ q_5$ $q_5\ 1 \rightarrow 1\ L\ q_5$	}	geht bis zur ersten nicht-leeren Zelle nach links
$q_5\ \# \rightarrow \#\ R\ q_1$		(eins zuviel, also eins zurück) (aber sonst wäre die leere Zelle nicht entdeckt worden)

Das vorliegende Programm wurde fast wörtlich aus der Programm-Skizze erstellt. Eine genaue Betrachtung zeigt jedoch, dass ein Fall bisher „vergessen“ wurde:

Befindet sich der Lese-/Schreibkopf zu Anfang über der rechten 1, so versagt das Programm!

Um dieses Problem zu lösen und das bestehende Programm zu „reparieren“, könnte z.B. ein neuer Zustand q_a eingeführt werden, der diesen Fall abprüft und später auf bereits bestehende Zustände des Programmes verweist (bzw. mit diesen fortsetzt).

$q_a\ 1$	$\rightarrow$	$1\ L\ q_a$	Der Lese-/Schreibkopf könnte zu Anfang auf einer 1 stehen. Dann aber muss zunächst herausgefunden werden, ob es sich um die linke oder die rechte 1 handelt. Dazu fährt der Lese-/Schreibkopf zunächst nach links.
$q_a\ 0$	$\rightarrow$	$0\ L\ q_0$	Ist das nächste Zeichen eine 0, so handelte es sich um die rechte 1. Spätestens jetzt befindet sich der Lese-/Schreibkopf über einer 0, und es kann mit Zustand q_0 fortgesetzt werden.
$q_a\ \#$	$\rightarrow$	$\#\ R\ q_0$	Ist das nächste Zeichen ein Blank-Zeichen (#), so handelte es sich hingegen um die linke 1. Dann ist der Lese-/Schreibkopf um einen Schritt zu weit gegangen und fährt wieder um eins zurück (nach rechts). Die weitere Vorgehensweise entspricht genau der, die mit Zustand q_0 und Eingabe 1 bereits abgedeckt wurde, daher kann mit Zustand q_0 fortgefahren werden.

Auch hier lässt sich das Programm etwas kompakter in Tabellenform schreiben:

	0	1	#	← Eingabe / ↓ Kommentar
q_a	$0, L, q_0$	$1, L, q_a$	$\#, R, q_0$	prüfe, ob zu Anfang auf rechter 1
q_0	$0, L, q_0$	$\#, R, q_1$	–	gehe nach links bis zur linken 1, lösche diese
q_1	$\#, R, q_3$	$1, R, q_2$	–	falls dort 0, lösche diese ($\rightarrow q_3$), sonst $\rightarrow q_2$
q_2	$0, R, q_2$	$1, R, q_2$	$1, H, e$	gehe bis ganz nach rechts, schreibe 1, stop.
q_3	$0, R, q_3$	$1, R, q_3$	$0, R, q_4$	gehe bis ganz nach rechts, schreibe 0
q_4	–	–	$0, L, q_5$	und noch eine 0
q_5	$0, L, q_5$	$1, L, q_5$	$\#, R, q_1$	dann gehe wieder ganz nach links, $\rightarrow q_1$
e	–	–	–	
Zustände				

Es fällt unter anderem auf, dass nicht alle Felder in der Tabelle besetzt sind. Diese speziellen Fälle treten im vorliegenden Beispiel nicht auf (oder zumindest sollten sie nicht auftreten).

Treten sie dennoch auf, so ist dies ein Fehler im Programm – und es kommt ganz auf die Definition der Turingmaschine an, wie in diesem Falle weiter verfahren wird (ob sie abbricht, hängen bleibt, in einen definierten Fehlerzustand übergeht, etc.).

Die etwas sauberere Vorgehensweise wäre, sich einen weiteren Fehlerzustand zu definieren und von allen diesen Fällen, die eigentlich nicht vorkommen sollten, einen Verweis auf diesen Fehlerzustand zu setzen.

Ableitungen: (Handsimulation)

$$\overset{q_a}{\#1001\#} \vdash \overset{q_a}{\#1001\#} \vdash \overset{q_0}{\#1001\#} \vdash \overset{q_0}{\#1001\#} \vdash \overset{q_1}{\#\#001\#} \vdash \overset{q_3}{\#\#\#01\#} \vdash \overset{q_3}{\#\#\#01\#}$$
$$\vdash \overset{q_3}{\#\#\#01\#} \vdash \overset{q_4}{\#\#\#010\#} \vdash \overset{q_5}{\#\#\#0100\#} \vdash \overset{q_5}{\#\#\#0100\#} \vdash \overset{q_5}{\#\#\#0100\#} \vdash \overset{q_5}{\#\#\#0100\#}$$
$$\vdash \overset{q_1}{\#\#\#0100\#} \vdash \overset{q_3}{\#\#\#\#100\#} \vdash \overset{q_3}{\#\#\#\#100\#} \vdash \overset{q_3}{\#\#\#\#100\#} \vdash \overset{q_3}{\#\#\#\#100\#}$$
$$\vdash \overset{q_4}{\#\#\#\#1000\#} \vdash \overset{q_5}{\#\#\#\#10000\#} \vdash \overset{q_5}{\#\#\#\#10000\#} \vdash \overset{q_5}{\#\#\#\#10000\#} \vdash \overset{q_5}{\#\#\#\#10000\#}$$
$$\vdash \overset{q_5}{\#\#\#\#10000\#} \vdash \overset{q_1}{\#\#\#\#10000\#} \vdash \overset{q_2}{\#\#\#\#10000\#} \vdash \overset{q_2}{\#\#\#\#10000\#} \vdash \overset{q_2}{\#\#\#\#10000\#}$$
$$\vdash \overset{q_2}{\#\#\#\#10000\#} \vdash \overset{q_2}{\#\#\#\#10000\#} \vdash \overset{e}{\#\#\#\#100001\#} \ \square$$

Beispiel 3: Addieren zweier Dualzahlen

Die folgende Turingmaschine berechnet die Summe zweier Zahlen in Binärdarstellung. Bei Eingabe von $bin(x) \sim bin(y)\$$ und Start im Zustand q_0 hält die Maschine im Zustand q_F mit der Ausgabe $bin(x+y) \sim bin(y)\$$ und dem Kopf auf dem ersten Eingabesymbol. (Die Notation $bin(x) \in \{0,1\}^*$ ist eine Darstellung von $x \in \mathbb{N}$ in Binärdarstellung.) Der Bandinhalt rechts vom \$-Zeichen bleibt dabei unverändert stehen.

Beginnend von rechts, wird jeweils das erste nicht markierte Bit von y und von x markiert. Wenn dieses Bit von y gleich 1 ist, wird es zur entsprechenden Stelle von x addiert und gegebenenfalls ein Übertrag nach links weitergereicht. Ein Blank-Symbol (#) am Anfang von x wird dabei wie eine 0 behandelt. Wenn alle Stellen von y markiert sind, werden alle Markierungen gelöscht, und der Automat hält.

δ	0	1	$\overline{0}$	$\overline{1}$	$\sim$	\$	#	Kommentar
q_0	$0,R,q_0$	$1,R,q_0$	$\overline{0},R,q_0$	$\overline{1},R,q_0$	$\sim,R,q_0$	$\$,L,q_1$	–	fahre nach rechts
q_1	$\overline{0},L,q_2$	$\overline{1},L,q_4$	$\overline{0},L,q_1$	$\overline{1},L,q_1$	$\sim,R,q_7$	–	–	finde erstes nicht markiertes Bit y_0 von y und markiere es
q_2	$0,L,q_2$	$1,L,q_2$	–	–	$\sim,L,q_3$	–	–	falls $y_0 = 0$, finde erstes unmarkiertes Bit von x, markiere es.
q_3	$\overline{0},N,q_0$	$\overline{1},N,q_0$	$\overline{0},L,q_3$	$\overline{1},L,q_3$	–	–	$\overline{0},N,q_0$	
q_4	$0,L,q_4$	$1,L,q_4$	–	–	$\sim,L,q_5$	–	–	falls, $y_0 = 1$, finde erstes Bit von x, markiere es, addiere 1,
q_5	$\overline{1},N,q_0$	$\overline{0},L,q_6$	$\overline{0},L,q_5$	$\overline{1},L,q_5$	–	–	$\overline{1},N,q_0$	
q_6	$1,N,q_0$	$0,L,q_6$	–	–	–	–	$1,N,q_0$	gegebenenfalls Übertrag
q_7	–	–	$\overline{0},R,q_7$	$\overline{1},R,q_7$	–	$\$,L,q_8$	–	Ende, fahre ganz nach rechts
q_8	$0,L,q_8$	$1,L,q_8$	$0,L,q_8$	$1,L,q_8$	$\sim,L,q_8$	–	$\#,R,q_F$	nach links, lösche Markierungen

Ableitungen: (Handsimulation)

$$
\begin{aligned}
&\overset{q_0}{\#}101\sim11\$\# \vdash \#\overset{q_0}{1}01\sim11\$\# \vdash \#1\overset{q_0}{0}1\sim11\$\# \vdash \#10\overset{q_0}{1}\sim11\$\# \vdash \#101\overset{q_0}{\sim}11\$\# \vdash \#101\sim\overset{q_0}{1}1\$\# \\
&\vdash \#101\sim1\overset{q_0}{1}\$\# \vdash \#101\sim11\overset{q_0}{\$}\# \vdash \#101\sim1\overset{q_1}{1}\$\# \vdash \#101\sim\overset{q_4}{1}\overline{1}\$\# \vdash \#101\overset{q_4}{\sim}1\overline{1}\$\# \\
&\vdash \#101\overset{q_5}{\sim}1\overline{1}\$\# \vdash \#10\overset{q_6}{0}\overline{0}\sim1\overline{1}\$\# \vdash \#1\overset{q_0}{1}\overline{0}\sim1\overline{1}\$\# \vdash \#11\overset{q_0}{\overline{0}}\sim1\overline{1}\$\# \vdash \#11\overline{0}\overset{q_0}{\sim}1\overline{1}\$\# \\
&\vdash \#11\overline{0}\sim\overset{q_0}{1}\overline{1}\$\# \vdash \#11\overline{0}\sim1\overset{q_0}{\overline{1}}\$\# \vdash \ldots \vdash \#10\overline{00}\sim\overline{11}\overset{q_0}{\$}\# \vdash \#10\overline{00}\sim\overline{11}\$\# \\
&\vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim\overline{11}\$\# \\
&\vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim\overline{11}\$\# \vdash \#10\overline{00}\sim11\$\# \vdash \#10\overline{00}\sim11\$\# \vdash \#10\overline{00}\sim11\$\# \\
&\vdash \#1000\sim11\$\# \vdash \#1000\sim11\$\# \vdash \#1000\sim11\$\# \vdash \#1000\sim11\$\#
\end{aligned}
$$

Einiges zur formalen Schreibweise

Die folgenden Bemerkungen seien an dieser Stelle nur der Vollständigkeit wegen aufgeführt.

Die obigen Beispiele geben den Kern dessen wieder, worauf es bei Betrachtungen von Turing-Beispielen ankommt: das eigentliche Turing-Programm; genauer eine Überführungsfunktion, die die Regeln für die Übergänge von einem Zustand der Maschine zu einem anderen bei einer gegebenen Eingabe beschreibt – und was die Turingmaschine als „Reaktion“ darauf für Aktionen durchführt.

Formal würde man dies noch etwas präzieser aufschreiben, denn bei der bisherigen Schreibweise bleiben noch ein paar Fragen offen:

- Welche Symbole (oder Variablennamen) werden für die Zustände verwendet?
- Welche Zeichen oder Worte können sich auf dem Eingabeband befinden?
- In welchem Zustand beginnt die Turingmaschine?
- Welches Zeichen wird als Blank-Zeichen verwendet?
- Gibt es mehr als einen Zustand, in dem die Maschine enden kann, und welche sind dies?

Diese Fragen waren in den obigen Beispielen eindeutig beantwortbar. Im allgemeinen ist dies nicht der Fall. Daher gehören zur vollständigen Definition einer Turingmaschine folgende Angaben:
(Auszug aus: Uwe Schöning, Theoretische Informatik kurz gefasst, Kap. 1.4, S.75 ff.)

Definition: Eine Turingmaschine (kurz: TM) ist gegeben durch ein 7-Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$$

Hierbei sind:

- Z die eindliche *Zustandsmenge*,
- Σ das *Eingabealphabet*,
- $\Gamma \supset \Sigma$ das *Arbeitsalphabet*,
- $\delta : Z \times \Gamma \longrightarrow Z \times \Gamma \times \{L, R, N\}$ die *Überführungsfunktion*,
- $z_0 \in Z$ der *Startzustand*,
- $\square \in \Gamma - \Sigma$ das *Blank*,
- $E \subseteq Z$ die Menge der *Endzustände*.

Informal erklärt bedeutet

$$\delta(z, a) = (z', b, x) \quad \text{bzw.} \quad \delta(z, a) \ni (z', b, x)$$

folgendes:[1]

Wenn sich M im Zustand z befindet und unter dem Schreib-Lesekopf das Zeichen a steht, so geht M in nächsten Schritt in den Zustand z' über, schreibt (auf den Platz von a) b auf das Band und führt danach die Kopfbewegung $x \in \{L, R, N\}$ aus. (Hierbei steht L für *links*, R für *rechts* und N für *neutral*, also Stehenbleiben).

Beispiel:[1]

Gegeben sei folgende Turingmaschine, die eine Eingabe $x \in \{0, 1\}^*$ als Binärzahl interpretiert und 1 hinzuaddiert: (vergleiche Beispiel 1)

$$M = (\{z_0, z_1, z_2, z_e\}, \{0, 1\}, \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$$

wobei

$$\begin{array}{rcl}
\delta(z_0, 0) & = & (z_0, 0, R) \\
\delta(z_0, 1) & = & (z_0, 1, R) \\
\delta(z_0, \square) & = & (z_1, \square, L) \\
\delta(z_1, 0) & = & (z_2, 1, L) \\
\delta(z_1, 1) & = & (z_1, 0, L) \\
\delta(z_1, \square) & = & (z_e, 1, N) \\
\delta(z_2, 0) & = & (z_2, 0, L) \\
\delta(z_2, 1) & = & (z_2, 1, L) \\
\delta(z_2, \square) & = & (z_e, \square, R)
\end{array}$$

[1] Hinweis: Die Reihenfolge bei den Parametern der δ-Funktion weicht von der Reihenfolge in der Vorlesung ab.
Noch ein Link: `http://www-m10.ma.tum.de/ix-quadrat/turing.html` — Animation Turingmaschine