Ausarbeitung zum Thema Turing Maschine
bei
Prof. Dr. Detlef Wotschke
Veranstaltung: Beschreibungskomplexität
(Proseminar)

ThorstenTimmer,
SS 2005

18. April 2005

Inhaltsverzeichnis

1 Einführung

In dieser Arbeit möchte ich die Motivation, Definition und Anwendbarkeit des Modells der Turing Maschine aufzeigen. Dazu beschreibe ich im Folgenden nicht nur die geschichtliche Motivation, die Turing Maschine zu entwickeln, sondern gehe auch auf die formale Definition, einige Beispiele, Techniken und Modifikationen in Bezug auf die Turing Maschine ein.
Ganz informell ausgedrückt, ist die Turing Maschine (im Folgenden auch TM genannt) ein einfaches mathematisches Modell für einen Computer, wie wir ihn uns heute vorstellen. Eine TM modelliert die durch einen Allzweck-Computer gegebene Berechenbarkeit. Wie man schon erahnen kann, ist die TM ein sehr mächtiges Modell. Tatsächlich könnte man auf ihr alle Programme, die von heutigen Computern berechnet werden, lösen. Ihre Struktur ist jedoch so einfach aufgebaut, dass es viel zu lange dauern würde heutige Programme in der Programmiersprache der TM zu erstellen. Auch die Berechnung würde ein erhebliches Maß an Zeitaufwand benötigen, da sie nicht auf Effizienz, sondern auf eine Einfachheit ausgelegt ist, um Probleme der Berechenbarkeit mit diesem theoretischen Modell für einen Menschen verständlich auszudrücken.

Mit diesem einfachen theoretischen mathematischen Modell ist es möglich Probleme z.B. auf ihre Berechenbarkeit durch einen Algorithmus zu untersuchen. Ist ein Problem mit einem endlichen Algorithmus lösbar, existiert dazu auch eine Turing Maschine und umgekehrt. Die Motivation der Entwicklung der Turing Maschine kann bis zu David Hilbert zurückverfolgt werden. Er versuchte um die Jahrhundertwende des 19 Jahrhunderts einen Algorithmus zu entwickeln, der die Wahrheit, bzw. Falschheit einer jeden mathematischen Aussage bestimmt. Er hatte damit keinen Erfolg. Alaan Turing entwickelte 1936 das Modell der Turing Maschine, um zu beweisen, dass es nicht möglich ist einen solchen Algorithmus zu erstellen. Da die Turing Maschine alle endlichen berechenbaren Probleme lösen kann, ist es auch möglich mit ihrer Hilfe den Beweis zu führen, das etwas nicht berechenbar ist. Zumindest im Sinne der TM.

2 Die Turing Maschine

In diesem Kapitel gehe ich auf die formale Definition und die Funktionsweise der TM ein. Das Basis Modell der TM besteht aus:

- einer `endlichen Kontrolle`
- einem nach links beschränkten und nach rechts unendlichen `Eingabeband`
- einem `Schreib-/Lesekopf`

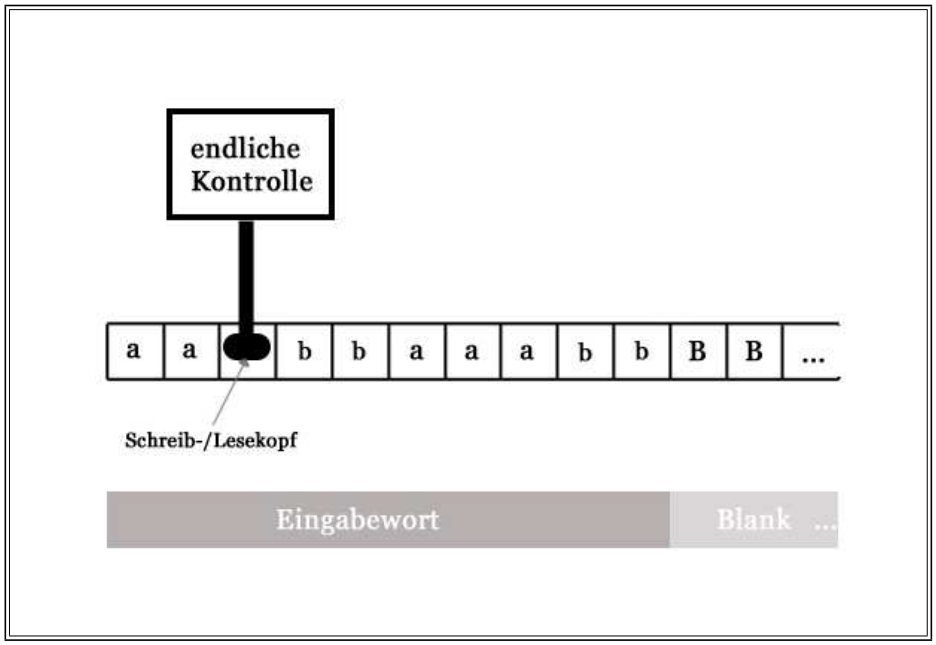

Abbildung 1: Basismodell der Turing Maschine

Die Endliche Kontrolle überwacht, bzw. kontrolliert die Bewegungen der Turing Maschine. Dazu befindet sie sich in einem Zustand von einer endlichen Menge von Zuständen. Jeder Zustand definiert, wie sich die TM bei einem gelesenen Eingabesymbol zu verhalten hat. Das Eingabeband ist nach links beschränkt, aber nach rechts unendlich lang. Es ist, vergleichbar mit einem Array, in Felder aufgeteilt, wobei pro Feld ein Symbol aus der Menge der Bandsymbole stehen kann. Ist ein Feld leer, steht in ihm das Blanksymbol (B). Auf den ersten n Feldern des Eingabebandes steht zu Beginn der Berechnung, also vor der ersten Bewegung der TM, die Eingabe. Die Eingabe ist ein Wort w mit $|w| = n$ und $w = \alpha_1 , \ldots , \alpha_n$, wobei $\alpha_1, \ldots , \alpha_n$ aus der Menge der Eingabesymbole sind. Die Eingabesymbole sind eine Teilmenge der Bandsymbole. Das Blanksymbol ist z.B. in der Menge der Bandsymbole enthalten, aber nicht in der Menge der Eingabesymbole. Die Felder rechts des letzten Symbols des Eingabewortes enthalten alle ein Blank.

2.1 Formale Definition

Definition: Eine Turing Machine (TM) M ist ein Tupel

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$,

wobei gilt:

- Q ist eine endliche Menge von Zuständen
- Σ ist eine enliche Menge von Eingabesymbolen.
- Γ ist eine enliche Menge von Bandsymbolen.
- δ ist die Übergangsfunktion, eine Abildung von $Q \times \Gamma$ nach $Q \times \Gamma \times \{L, R\}$
- $q_0 \in Q$ ist der Anfangszustand
- B ist das Blanksymbol
- $F \subseteq Q$ ist die Menge der Endzustände

Die Übergangsfunktion δ beschreibt den Übergang von einer Zustandsbeschreibung in die nächste, in Abhängigkeit des gelesenen Eingabesymbols auf dem Band und dem momentanen Zustand in der endlichen Kontrolle. Eine Zustandsbeschreibung einer TM M ist folgendermaßen definiert:

$$\alpha_1 q \alpha_2$$

wobei, $\alpha_1\alpha_2 \in \Gamma^*$ und $q \in Q$. $\alpha_1\alpha_2$ ist der momentane Bandinhalt, gefolgt von unendlich vielen Blank Symbolen und q ist der Zustand, in der sich die endliche Kontrolle befindet. Der Schreib-/Lesekopf befindet sich bei einer solchen, auch Konfiguration genannten Zustandsbeschreibung auf dem ersten Symbol von α_2. Ist $\alpha_2 = \epsilon$, also leer, befindet sich der Kopf rechts neben der Eingabe auf einem Blank. Eine Bewegung der Turing Maschine ist ein Übergang von einer Konfiguration in ihre Folgekonfiguration. Die Folgekonfiguration hängt dabei von dem Zustand, dem gelesenen Eingabesymbol, also dem ersten Symbol von α_2 und der Bewegung des Kopfes nach Links oder Rechts ab. Beispielsweise beschreibt $\delta(q, X_i) = (p, Y, L)$, dass sich die TM in den Zustand p begibt, das Symbol X_i durch Y ersetzt und sich dann um ein Feld nach Links bewegt, wenn das Eingabesymbols X_i gelesen wird, während sich die endliche Kontrolle im Zustand q befindet. Die Zustandsbeschreibungen sehen folgendermaßen aus:

$$X_1 X_2 \ldots X_{i-1} q X_i \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n$$

Das Symbol $\vdash_M$ besagt, dass die rechts stehende Zustandsbeschreibung aus der links stehenden Zustandsbeschreibung folgt. Das M kann weggelassen werden, wenn klar ist auf welche TM sich $\vdash$ bezieht. Folgt eine Zustandsbeschreibung durch eine endliche Anzahl von Zuständen aus einer anderen, dann kann man auch $\vdash^*$ schreiben. Besitzt eine Konfiguration keine Folgekonfiguration, dann ist sie eine Endkonfiguration. Die TM hält dann in dieser Zustandsbeschreibung, wobei sie akzeptieren oder verwerfen kann, je nachdem, ob der Zustand akzeptierend oder verwerfend ist.

Die von einer TM erkannte Sprachen kann man damit als die Menge

$$\{ w \mid w \in \Sigma^* \wedge q_0 w \vdash^* \alpha_1 p \alpha_2,\, p \in F \wedge \alpha_1, \alpha 2 \in \Gamma^* \}$$

schreiben.

2.2 Berechenbare Sprachen und Funktionen

Die Sprachen, die von einer TM erkannt werden, sind die rekursiv aufzählbaren Sprachen. Die Klasse der rekursiv aufzählbaren Sprachen enthält z.B. die regulären Sprachen und kontextfreien Sprachen komplett. Es gibt Sprachen L in den rekursiv aufzählbaren Sprachen, die von einer TM M erkannt werden und die TM mit der Eingabe $w \in L$ hält, aber für Wörter $w \notin L$ nicht anhält. Solche Sprachen sind semi-entscheidbar, wohingegen die rekursiven Mengen entscheidbar sind. Die rekursiven Mengen sind eine echte Unterklasse der rekursiv aufzählbaren Mengen. Für sie gilt, das die TM, die eine Sprache aus den rekursiven Sprachen akzeptiert, auf jeder Eingabe hält. Diese Sprachen werden nicht nur erkannt sondern auch entschieden, da es durch die TM möglich ist herauszufinden, ob ein Wort in der Sprache ist oder nicht. Bei den Sprachen, die nur erkannt werden, ist dies für Wörter, die nicht in der Sprache sind unmöglich, da die TM ja unendlich lange läuft. Es ist also möglich ein Wort aus L zu erkennen, aber für ein beliebiges Wort im Allgemeinen nicht möglich zu entscheiden, ob es in L enthalten ist, oder nicht.

2.2.1 Berechnung ganzzahliger Funktionen

Die Turing Maschine kann auch dazu verwendet werden, ganzzahlige Funktionen zu berechnen. Dazu kann eine ganze Zahl i in unärer Darstellung als 0^i dargestellt werden. Hat eine Funktion k Argumente, werden diese durch das Zeichen 1 getrennt, beispielsweise als $0^{i_1}10^{i_2}1 \ldots 10^{i_k}1$. Das Ergebnis der Funktion f liegt nach der Berechnung in Form von der Zeichenkette 0^m auf dem Band vor. Also ist $f(i_1, \ldots, i_k) = m$.
Eine der wohl am einfachsten zu berechnenden Funktionen ist die Nachfolgerfunktion $s(x) = x + 1$. Eine TM muss an die Eingabe 0^x einfach eine 0 dranhängen, um das Ergebnis 0^{x+1} zu berechnen. Diese Funktion gehört zu der Klasse der total rekursiven Funktionen, die eine Unterklasse der partiell rekursiven Funktionen ist. Die total rekursiven Funktionen können analog zu den rekursiven Sprachen aufgefasst werden, da sie von einer TM berechnet werden, die bei jeder Eingabe hält. Genauso sind die partiell rekursiven Funktionen analog zu den rekusiv aufzählbaren Sprachen aufzufassen, da die TM, die eine solche Funktion berechnet nicht halten muss. Viele der gebräuchlichen Funktionen sind total rekursiv, wie z.B. $n!$ oder 2^{2^n}.

Ein weiteres Beispiel ist die echte Subtraktion $m \dot{-} n$, die per Definition $m - n$ für $m \leq n$ und 0 für $m < n$ ist. Die TM

$$M = (\{q_0, \ldots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_6, B, \emptyset),$$

startet mit der Eingabe 0^m10^n auf dem Band und stoppt mit $0^{m \dot{-} n}$ auf dem Band. M liest eine der m Nullen und ersetzt sie durch ein Blank, danach sucht M eine 1, die angibt, das nach den Einsen die n Nullen folgen und ersetzt die erste gefundene Null durch eine Eins. Danach bewegt sich M nach links, bis ein Blank gefunden wird und liest das nächste rechts stehende Symbol neben diesem Blank. Steht wieder eine Null dort, wiederholt M die Prozedur, die Wiederholung endet, wenn eine der folgenden Bedingungen zutreffen:

1. Auf der Suche nach rechts wird, statt eine Null, ein Blank gefunden. Dann sind alle n Nullen durch Einsen ersetzt worden und $n+1$ der m Nullen sind durch B ersetzt worden. Die $n+1$ Einsen werden durch eine Null gefolgt von n Blanks ersetzt. Das Ergebnis ist die Subtraktion $m-n$ in Form von $m-n$ Nullen auf dem Band.

2. Am Anfang der Schleife wird keine Null gefunden, die in Blanks umgewandelt werden sollen, da schon alle m Nullen ersetzt wurden. Dann gilt $n \geq m$. Nun werden alle Einsen und Nullen durch Blanks ersetzt.

Die Übergangsfunktion δ aufgelistet und beschrieben:

1. $\delta(q_0, 0) = (q_1, B, R)$
   Die führende Null wird durch ein Blank ersetzt.

2. $\delta(q_1, 0) = (q_1, 0, R)$
   $\delta(q_1, 1) = (q2, 1, R)$
   Die erste Eins wird gesucht.

3. $\delta(q_2, 1) = (q_2, 1, R)$
   $\delta(q_2, 0) = (q_3, 1, L)$
   Ist die erste Eins schon gefunden, werden alle weiteren übersprungen und die nächste folgende Null durch eine Eins ersetzt.

4. $\delta(q_3, 0) = (q_3, 0, L)$
   $\delta(q_3, 1) = (q_3, 1, L)$
   $\delta(q_3, B) = (q_0, B, R)$
   Alle Nullen und Einsen nach links werden übersprungen, bis ein Blank gefunden ist. Ist es gefunden, bewegt sich die TM um ein Feld nach rechts und geht in q_0 über, um die Schleife zu wiederholen.

5. $\delta(q_2, B) = (q_4, B, L)$
   $\delta(q_4, 1) = (q_4, B, L)$
   $\delta(q_4, 0) = (q_4, 0, L)$
   $\delta(q_4, B) = (q_6, 0, R)$
   Sollte in Zustand q_2 keine Null sondern ein Blank gefunden werden, so bewegt sich die TM in Zustand q_4 und bei einer Bewegung nach links

werden alle Einsen durch B ersetzt, bis ein B gefunden wird. Dieses Blank wird durch eine Null ersetzt, die TM geht in Zustand q_6 über und stoppt.

6. $\delta(q_0, 1) = (q_5, B, R)$
   $\delta(q_5, 0) = (q_5, B, R)$
   $\delta(q_5, 1) = (q_5, B, R)$
   $\delta(q_5, B) = (q_6, B, R)$
   Findet die TM in Zustand q_0 eine Eins statt einer Null, so sind die ersten m Nullen erschöpft. M geht dann in Zustand q_5 über, um das Band von allem überflüssigen Rest zu bereinigen und stoppt dann in Zustand q_6.

2.3 Techniken zur Konstruktion von Turing Maschinen

Wie man sich vorstellen kann, ist es für komplexere Probleme nicht sehr einfach eine TM mit den bisher gegebenen Eigenschaften zu konstruieren. Deshalb betrachten wir nun einige „Tricks“ um die Konstruktion von Turing Maschinen zu vereinfachen.

2.3.1 Speichern in der endlichen Kontrolle

Die endliche Kontrolle kann erweitert werden, um Information zu speichern. Dazu wird jeder Zustand als ein Tupel geschrieben, nicht mehr wie bisher als nur ein Buchstabe. Die Turing Maschine an sich wird dadurch nicht verändert, es ändern sich lediglich die Bezeichner für die Zustände der endlichen Kontrolle.

Beispiel: Als Beispiel betrachten wir eine TM M, die sich das erste Symbol des Eingabewortes w anschaut, es in der endlichen Kontrolle speichert und mit den folgenden Symbolen von w vergleicht. Kommt das entsprechende Symbol nicht mehr in w vor, akzeptiert M, andernfalls verwirft M. Sei

$$M = (Q, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, F)$$

mit $Q \in \{q_0, q_1\} \times \{0, 1, B\}$. Q ist also die Menge der Paare

$$\{[q_0, 0], [q_0, 1], [q_0, B], [q_1, 0], [q_1, 1], [q_1, B]\}.$$

Die Menge F ist $\{[q_0, B]\}$. Die erste Komponente der Zustände kontrolliert also die nächste Aktion und die zweite Komponente läßt die TM sich an das erste gelesene Eingabesymbol „erinnern“. δ ist nun folgendermaßen definiert:

1. (a) $\delta([q_0, B], 0) = ([q_1, 0], 0, R)$
   (b) $\delta([q_0, B], 1) = ([q_1, 1], 1, R)$
   Zu Beginn befindet sich M in Zustand q_0 und liest entweder eine 0, oder eine 1 und bewegt sich dann nach rechts. Die erste Komponente wird nun zu q_1, während in der zweiten das gelesene Symbol gespeichert wird.
2. (a) $\delta([q_1, 0], 1) = ([q_1, 0], 1, R)$
   (b) $\delta([q_1, 1], 0) = ([q_1, 1], 0, R)$
   In diesen Zuständen wird geprüft, ob das erste gelesene und in der zweiten Komponente gespeicherte Symbol ein zweites mal gelesen wird oder nicht. Wird jeweils ein anderes Symbol gelesen, bewegt sich die TM um ein Feld nach rechts, ohne etwas zu verändern.
3. (a) $\delta([q_1, 0], B) = ([q_1, B], 0, L)$
   (b) $\delta([q_1, 1], B) = ([q_1, B], 1, L)$
   Wird bis zum ersten Blank Symbol das erste Symbol nicht mehr gelesen, so geht M in den akzeptierenden Zustand über.

Die Übergänge von den Zuständen $[q_1, 0]$ und einer gelesenen 0 und $[q_1, 1]$ und einer gelesenen 1 sind nicht definiert, so dass die TM in diesen Fällen ohne zu akzeptieren anhält.
Die endliche Kontrolle kann aus k Komponenten bestehen, wobei eine Komponente immer die Aktion angibt und die restlichen zum Speichern von Information verwendet werden können.

2.3.2 Mehrspurige Turing Maschinen

Man kann sich das Band der TM in k Spuren aufgeteilt vorstellen, wobei die Bandsymbole dann als k-Tupel aufgefasst werden.

Abbildung 2 zeigt eine mehrspurige TM, auf dessen Spuren Zahlen in binärdarstellung stehen. Auf der ersten Spur ist die Zahl 47 zu sehen, die überprüft werden soll, ob sie eine Primzahl ist oder nicht. Dazu wird die Zahl der ersten Spur auf die dritte Spur kopiert und auf die zweite Spur die Zahl 2 in Binärdarstellung geschrieben. Danach wird von der dritten Spur die zwei so oft wie möglich subtrahiert. Bleibt ein Rest auf Spur drei übrig, so wird die Zahl auf Spur zwei um eins erhöht und die ursprüngliche Zahl, die noch auf Band eins vorhanden ist wieder auf das dritte Band kopiert. Danach beginnt die Prozedur von vorne, bis die Zahl selbst auf Band zwei steht. Wird Band drei vorher in einem Schritt Null, so ist die Zahl keine Primzahl, tritt dies nicht ein, so ist sie eine Primzahl. Es ist klar, wie die Tupel beispielsweise vor der ersten Bewegung der TM aussehen. Es handelt sich dabei um die Eingabesymbole, die $[\S, B, B], [0, B, B], [1, B, B]$ oder $[\$, B, B]$ sein können. Das Blanksymbol ist $[B, B, B]$.

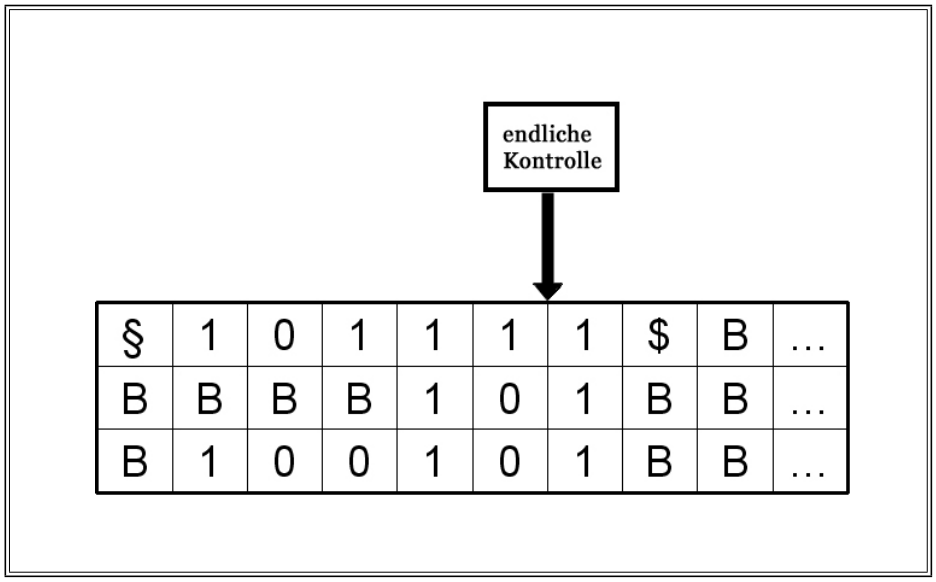

Abbildung 2: Mehrspurige Turing Maschine

2.3.3 Markierungssymbole

Markierungssymbole sind ein gutes Hilfsmittel, um aufzuzeigen, wie eine TM Wörter erkennt, in denen bestimmte Zeichenketten sich wiederholen. Beispielsweise

$$\{ww|w \in \Sigma^*\}$$

oder

$$\{wcy|w, y \in \Sigma^*, w \neq y\}. \qquad (*)$$

Dazu erweitern wir das Band einfach auf eine zweite Spur, die entweder ein Blank (B) oder einen Haken ($\surd$) enthalten kann. Ein Haken wir dazu verwendet, ein Symbol des Eingabebandes zu markieren. Im Falle von $(*)$ wird ein Symbol im ersten Teilwort w abgehakt, bzw. markiert, wenn es gelesen wird, danach wird das erste Symbol von y gesucht und mit Hilfe der Erweiterung der endlichen Kontrolle verglichen. Sind die beiden Symbole ungleich, so wird auch das entsprechende Symbol in y abgehakt.

Um Teile einer Zeichenkette bezüglich ihrer Länge zu vergleichen, ist das Abhaken ebefalls ein nützliches Konzept.

Beispiel:
$$\{a^i b^j c^k | i \neq j, j \neq k\}$$

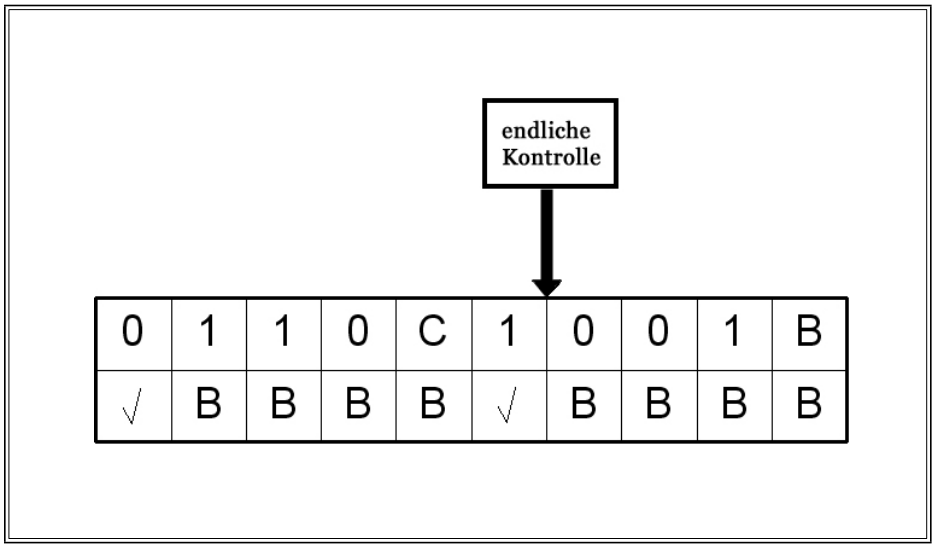

Abbildung 3: TM mit Markierungssymbolen

2.3.4 Verschiebungen

Um auf dem Band Platz zu schaffen, ist es möglich alle von Blank verschiedenen Symbole um eine endliche Anzahl von Feldern nach rechts zu verschieben. Dazu verwenden wir wieder das Speichern in der endlichen Kontrolle. Können beispielsweise drei Symbole in der endlichen Kontrolle gespeichert werden, können die am weitesten links liegenden zwei Symbole in die endliche Kontrolle aufgenommen und durch Blanks ersetzt werden, wobei das erste gelesene Symbol in dem Speichertripel an zweiter Stelle steht und das zweite an dritter. Die erste Komponente der endlichen Kontrolle ist wieder das Kontrollsymbol. Das dritte Symbol des Bandes, das danach gelesen wird, wird nun an die dritte Stelle des Tripels geschrieben, das vorher dort gespeicherte an die zweite verschoben und auf das dritte Feld des Bandes das ehemals erste Symbol geschrieben. Danach bewegt sich die TM nach rechts und wiederholt die ganze Prozedur. Wird ein Blank gelesen, so werden die gespeicherten Symbole einfach auf das Band geschrieben, bis die endliche Kontrolle leer ist. Eine vorraussetzung dafür ist natürlich, dass in dem Wort auf dem Band keine Blanksymbole vorkommen.

2.3.5 Unterprogramme

Es ist nicht verwunderlich, dass man mit einer TM Unterprogramme schreiben kann, die, falls sie benötigt werden, immer wieder aufgerufen werden können. Dazu schreiben wir das Unterprogramm so, dass es einen Anfangszustand und einen Rückkehrzustand hat. Der Rückkehrzustand beinhaltet im allgemeinen Fall keine Bewegung, sie wird erst beim Einfügen in ein ande-

res Programm bestimmt, da wir vorher nicht wissen, wo das entsprechende Programm nach Beendigung des Unterprogramms weiterarbeiten wird. Ein gutes Beispiel dafür ist das Unterprogramm COPY, das z.B. für die total rekursive Funktion Multiplikation verwendet werden kann. Bei der Multiplikation steht zu beginn auf dem Band die Zeichenkette 0^m10^n gefolgt von unendlich vielen Blanks und am Ende 0^{mn} umgeben von Blanks. Anfangs wird eine Eins hinter 0^m10^n geschrieben, im folgenden m-mal die n Nullen hinter diese eins gesetzt und in jedem Schritt eine der m Nullen gelöscht. Danach befindet sich auf dem Band 10^n10^{mn} und das Präfix 10^n1 wird dann gelöscht. Das Unterprogramm COPY ist nun dazu da, die n Nullen im jeden der m Schritte zu kopieren. Es beginnt mit der Zustandsbeschreibung $0^m1q_10^n10^i$ und endet mit $0^m1q_10^n10^{i+n}$.

Findet COPY in Zustand q_1 eine 0, so ersetzt es diese durch eine 2 und geht in Zustand q_2 über. Von dort aus wird das nächste Blank gesucht, durch eine 0 ersetzt, worauf eine Bewegung nach Links folgt und der Übergang in q_3. Alle links stehenden Zeichen werden nun übersprungen, bis eine Zwei gefunden wird und das ganze beginnt wieder in Zustand q_1. Wird in q_1 keine 0 mehr gefunden, sondern eine 1, geht die TM in Zustand q_4 über und bewegt sich nach Links auf die am weitesten rechts stehende 2. Alle 2-Symbole werden wieder durch 0 ersetzt, bis eine 1 gefunden wird, worauf die TM in Zustand q_5 wechselt, welcher der Endzustand ist und das Unterprogramm damit verlässt.

	0	1	2	B
q_1	$(q_2, 2, R)$	$(q_4, 1, L)$		
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$		$(q_3, 0, L)$
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	$(q_1, 2, R)$	
q_4		$(q_5, 1, R)$	$(q_4, 0, L)$	

Abbildung 4: Übergänge des Unterprogramms COPY

Um von der anfänglichen Zustandsbeschreibung $q_00^m10^n$ in $B0^{m-1}1q_10^n1$ zu kommen müssen noch zusätzliche Zustände hinzugefügt werden.

$$\delta(q_0, 0) = (q_6, B, R)$$
$$\delta(q_6, 0) = (q_6, 0, R)$$
$$\delta(q_6, 1) = (q_1, 1, R).$$

Um von $B^i0^{m-i}1q_50^n10^{ni}$ in $B^{i+1}0^{m-i-1}1q_10^n10^{ni}$ zu kommen, wodurch COPY erneut gestartet wird und um zu prüfen, ob $i = m$ gilt, also alle m Nullen gelöscht sind, brauchen wir noch weitere Übergänge. Gilt $i = m$, so muss auch das Präfix 10^n1 gelöscht werden und ein haltender Endzustand definiert werden.

	0	1	2	B
q_5	$(q_7, 0, L)$			
q_7		$(q_8, 1, L)$		
q_8	$(q_9, 0, L)$			(q_{10}, B, R)
q_9	$(q_9, 0, L)$			(q_0, B, R)
q_{10}		(q_{11}, B, R)		
q_{11}	(q_{11}, B, R)	(q_{12}, B, R)		

3 Modifikationen von Turing Maschinen

Es gibt verschiedene Möglichkeiten die TM zu erweitern, so dass es bei einem ersten kurzen Blick auf die Erweiterungen vorstellbar wäre, dass diese die Berechenbarkeit verändern. Wir werden jedoch feststellen, dass die TM erstaunlich robust gegenüber Erweiterungen ist.

3.1 Beidseitig unendliches Band

Wie der Name schon ausdrückt, ist die hauptsächliche Veränderung einer TM mit beidseitigem unendlichem Band die Erweiterung des Bandes, so dass es zu beiden Seiten unendlich lang ist. Die formale Definition bleibt größtenteils zur oben genannten gleich. Die Ausnahmen sind folgende:

Für $\delta(q, X) = (p, Y, L)$ gilt $qX\alpha \vdash_M pBY\alpha$ und für $\delta(q, X) = (p, B, R)$ gilt $qX\alpha \vdash_M p\alpha$. Im ersten Fall konnte ursprünglich keine Bewegung nach links gemacht werden, im zweiten Fall tauchte B links von von p auf. Die anfängliche Zustandsbeschreibung bleibt q_0w, die TM kann sich jetzt aber unbeschränkt nach links bewegen.

Es gilt folgender Satz:

Satz 3.1 L wird von einer TM mit beidseitg unendlichem Band erkannt $\Leftrightarrow$ L wird von einer TM mit einem nach einer Seite unendlichen Band erkannt

Beweis:
Es gilt beide Richtungen zu beweisen:
Seien M_1 die TM mit einseitig unendlichem Band und M_2 die TM mit beidseitig unendlichem Band.
1. L wird von einer TM mit beidseitig unendlichem Band erkannt $\Leftarrow$ L wird von einer TM mit einem nach einer Seite unendlichen Band erkannt.
Um M_1 durch M_2 zu simulieren, fügen wir einfach ein Zeichen auf dem Band von M_2 ein, das das Ende des Bandes in die linke Richtung bezeichnet. Wird dieses Symbol erreicht, so hält M_2 ohne zu akzeptieren, wie es auch M_1 tun würde.
2. L wird von einer TM mit beidseitg unendlichem Band erkannt $\Rightarrow$ L wird

von einer TM mit einem nach einer Seite unendlichen Band erkannt.
Als Beweis konstruieren wir eine TM M_1, die M_2 simuliert. In dem M_1 einfach mit zwei Spuren ausgestattet wird, wobei auf der ersten Spur der Bandinhalt rechts und inklusive des Startsymbolt von M_2 steht und auf der zweiten Spur im ersten Feld das Symbol § gefolgt vom Bandinhalt links des Startsymbols von M_2. Sollte sich der Kopf, während er auf Spur 2 arbeitet, nach links über das Symbol § bewegen, so arbeitet die TM auf Spur 1 in die entgegengesetzte Richtung , also rechts, weiter.
Dadurch, dass wir wissen, dass einzelne Spuren einzeln bearbeitet werden können, ist es klar, das damit M_2 simuliert werden kann.

3.2 Mehrbändige Turing Maschinen

Eine mehrbändige TM besteht aus einer endlichen Kontrolle mit k Bandköpfen und k Bändern. Jedes Band ist in beide Richtungen unendlich. Abhängig von der jeweiligen Konfiguration, kann die TM dann folgendes machen:

1. Den Zustand in der endlichen Kontrolle ändern
2. In jedes der k gelesenen Felder der k Köpfe ein neues Symbol schreiben, oder das Symbol stehen lassen
3. Jeden der k Bandköpfe unabhängig von den anderen Köpfen nach rechts oder links bewegen, oder stehen lassen.

Die Eingabe steht zu Beginn auf dem ersten Band und alle anderen Bänder beinhalten Blanksymbole. Es gilt folgender Satz:

Satz 3.2 L wird von einer mehrbändigen TM akzeptiert $\Leftrightarrow$ L wird von einer einbändigen TM akzeptiert.

Beweis:
Seien M_1 die mehrbändige TM und M_2 die einbändige TM.
1. L wird von einer mehrbändigen TM akzeptiert $\Rightarrow$ L wird von einer einbändigen TM akzeptiert.
Wir konstruieren eine TM M_2 mit $2k$ Spuren, die für jedes Band von M_1 zwei Spuren zur verfügung stellt. Eine Spur ist für den Bandinhalt des jeweiligen Bandes von M_1 zuständig und die andere für eine Markierung der Kopfposition, also des gelesenen Symbols auf M_1. Die endliche Kontrolle von M_2 speichert den Zustand von M_1 zusammen mit der Anzahl rechts vom Bandkopf von M_2 befindlichen Kopfmarkierungen. Um eine Bewegung von M_1 zu simulieren, arbeitet M_2 folgendermaßen:
Anfangs steht der Bandkopf auf dem am weitesten links stehenden Feld, das eine Kopfmarkierung enthält. Nun werden alle Symbole, die markiert sind

nach rechts gesucht und der Bandinhalt jeweils in der endlichen Kontrolle gespeichert. Die Anzahl der Markierungen, die sich rechts vom Kopf befinden, wird bei jedem Erfolg um Eins reduziert, damit die TM weiß, wann alle Markierungen gefunden sind. Sind alle markierten Symbole eingelesen, hat die TM genug Information, um den Übergang, den M_1 angewendet hätte, zu kennen. Jetzt bewegt sie sich wieder nach links und ändert jeden markierten Eintrag, wie es der Übergang beschreibt. Dazu wird die Anzahl der rechts stehenden Markierungen wieder aktualisiert, um zu wissen, wann die am weitesten links stehende Markierung erreicht ist. Ausserdem wir jede Markierung entsprechen des Übergangs nach links oder rechts verschoben. Letztendlich wird der gespeicherte Zustand, der dem Zustand in M_1 entspricht, in der endlichen Kontrolle von M_2 geändert. Damit ist die Bewegung beendet. Ist der neue Zustand ein akzeptierender Zustand in M_1, so akzeptiert auch M_2.
2. L wird von einer mehrbändigen TM akzeptiert $\Leftarrow$ L wird von einer einbändigen TM akzeptiert. Dies sollte klar sein, da es kein Problem darstellt, die mehrbändige TM soweit einzuschränken, dass sie wie eine einbändige TM arbeitet.

3.3 Nichtdeterministische Turing Maschinen

Die Veränderungen bei der nichtdeterministischen TM sind vergleichbar mit den Veränderungen, die bei nichtdeterministischen endlichen Automaten im Vergleich zu deterministischen Automaten auftreten. Die TM hat in diesem Fall, bei gegebenem Zustand in der endlichen Kontrolle und gelesenem Eingabesymbol, die Möglichkeit mehrere bestimmte Bewegungen durchzuführen. Der Übergang geht also nicht nur in eine bestimmte Konfiguration über, sondern in eine bestimmte Menge von Konfigurationen. Um zu testen, ob die TM akzeptiert, müssen alle möglichen Folgen von Konfigurationen der zu testenden Eingabe überprüft werden. Befindet sich eine Folge darunter, die in einem akzeptierenden Zustand endet, so akzeptiert die TM. Wie die vorangehenden Veränderungen, verändert der Nichtdeterminismus nicht die Sprachklassen, die von einer standard TM akzeptiert werden.

Satz 3.3 *L wird von einer nichtdeterministischen TM akzeptiert $\Leftrightarrow$ L wird von einer deterministischen TM akzeptiert.*

Beweis:
Wir konstruieren dazu die TM M_2, die die nichtdeterministische TM M_1 simulieren soll. M_2 benötigt drei Bänder. Auf dem ersten Band steht die Eingabe, das zweite Band ist dazu da, eine Folge von Bewegungen der TM M_1 bei dieser Eingabe für die Simulation zu speichern und das dritte Band wird für die Simulation von M_1 verwendet. Da es nur endlich viele möglichkeiten gibt, den Zustand in der endlichen Kontrolle mit einem Eingabesymbol zu

kombinieren, kann es für eine bestimmten Konfiguration nur eine endliche Menge von Folgekonfigurationen geben. Sei die Anzahl aller möglichen Paare von Zustand und Eingabesymbol r. Dann ist es möglich den Übergang von einer Konfiguration in eine Untermenge von $\{1..r\}$ übergehen zu lassen. Auf Band 2 werden nun systematisch Folgen zwischen 1 und r generiert. Die kürzeste Folge erscheint zu erst und Folgen gleicher Länge werden in numerischer Ordnung erzeugt. Für jede erzeugte Folge wird die Eingabe auf Band 3 kopiert und dann Band 2 benutzt um die Bewegungen von M_1 auf Band 3 zu simulieren. Wenn es eine Folge gibt, die von M_1 akzeptiert wird, dann akzeptiert auch M_2, da alle Folgen auf Band 2 generiert werden. Gibt es keine Folge, die von M_1 erkannt wird, so erkennt auch M_2 nicht.
Die Rückrichtung des Beweises ist wieder sehr einfach, da eine deterministische TM als nichtdeterministische TM aufgefasst werden kann, die aber bei jeder Konfiguration nur eine Auswahlmöglichkeit hat.

3.4 Mehrdimensionale Turing Maschinen

Eine mehrdimensionale TM hat ein Band, das aus einem k-dimensionalem Array besteht, das zu allen $2 * k$ Seiten unendlich lang ist. Befindet sich die TM an einer bestimmten Bandposition, so gibt es $2 * k$ möglichkeiten, wohin sich der Kopf bewegen kann, also schneiden sich jeweils $2k$ Achsen in einer Bandposition. Zu Beginn befindet sich die Eingabe auf einer Achse und der Kopf über dem linkesten Eingabesymbol.

Satz 3.4 L wird von einer k-dimensionalen TM akzeptiert $\Leftrightarrow$ L wird von einer eindimensionalen TM akzeptiert.

Beweis:
Es genügt zu zeigen, dass jeder Bandzustand eines k-dimensionalen Bandes auf ein eindimensionales Band geschrieben werden kann. Stellt man sich vor, dass man alle von einem Blank verschienen Symbole auf einem k-dimensionalen Band mit einem k-dimensionalen Würfel einfangen kann, so ist es auch möglich dies auf einem eindimensionalen Band zu repräsentieren und damit arbeiten zu können. Alle Achsen, die durch den Würfel gehen, haben die gleiche Länge, also können alle Achsen auf das eindimensionale Band geschrieben werden. Als Trennzeichen zwischen den Dimensionen verwenden wir das Symbol $*$. Betrachtet man ein 2-dimensionales Band, so werden die Bandzeilen durch einen $*$ getrennt, bei einem 3-dimensionalen Band, die 2te Dimension wieder durch $*$ und die dritte Dimension durch $**$, usw... Bei einer Bewegung auf einer Achse die parallel zur Eingabeachse ist, ist nichts weiter zu beachten, außer bei erreichen eines Sternes. Der Stern kann als Platzhalter für unendlich viele Blanksymbole aufgefasst werden. Es dann möglich, den Würfel zu vergrößern, indem man hinter jedes der Symbole $*$, $**$, $***$, ... ein Blank schreibt, also jeweils den Inhalt verschiebt, wie wir es schon gesehen haben. Bei einer bewegung in eine andere Richtung ist es

wichtig, die Nummer der Dimension zu kennen, damit man sich an den Sternen orientieren kann. Bewegt man sich z.B. innerhalb der 2ten Dimension, so ist das Symbol $**$ die Grenze in der man sich auf dem eindimensionalen Band bewegen darf. Man befindet sich dann beispielsweise an dritter Position, gesehen von der Startposition, eines Bandes aus der 2ten Dimension und bewegt sich demnach auf eine dritte Position eines Nachbarbandes der 2ten Dimension. Diese Achsen sind durch $*$ getrennt. Also sucht man die Zeichenkette links vom nächsten linken $*$ oder die Zeichenkette rechts vom rechten $*$ und begibt sich dort an die dritte Position.

3.5 Turing Maschine mit mehreren Köpfen

Eine k-köpfige TM hat k Köpfe auf einem Band zu verfügung. Die Köpfe sind von $1 \ldots k$ numeriert. Die Köpfe können sich bei einer Bewegung unabhängig voneinander bewegen, oder auch stehen bleiben. Der Übergang hängt von jedem gelesenem Symbol und dem Zustand ab.

Satz 3.4 *L wird von einer k-köpfigen TM akzeptiert $\Leftrightarrow$ L wird von einer einköpfigen TM akzeptiert.*

Beweis:
Der Beweis ist ähnlich zu dem Beweis für mehrbändige TM. Die zu konstruierende TM hat $k + 1$ Spuren, wobei auf einer Spur die Eingabe steht und auf den anderen Spuren die Positionen der Köpfe vermerkt sind. Bei jeder Bewegung wird jeder Kopf gesucht, sein gelesenes Symbol gespeichert und nachdem alle Köpfe gefunden wurden, die neuen Symbole auf das Band geschrieben und die Köpfe bewegt.

3.6 Off-Line Turing Maschinen

Bei einer Off-Line TM, die eine mehrbändige TM ist, ist die Eingabe auf dem Eingabeband durch zwei Symbole begrenzt. Ein Symbol gibt das linke Ende an und das andere das rechte Ende. Das Eingabeband darf nicht verändert werden. Der Kopf muss sich immer innerhalb der Begrenzungssymbole befinden. Da sie nur ein Spezialfall der mehrbändigen TM ist, ist klar, das sie von einer einbändigen TM simuliert werden kann. Die Off-Line TM kann aber jede andere TM simulieren, indem sie ein Band mehr als die zu simulierende TM verwendet.

Literatur

[1] Hopcraft, Ullman: Introduction to automata theory, languages, and computation, Addison Wesley 1979