

Turing-Maschinen

1936 stellte Alan Turing (1912-1954) in seiner berühmten Arbeit "On computable numbers" lange vor der Erfindung der Computer ein Modell einer Rechenmaschine vor.

Ihre Eigenschaften:

- sie soll möglichst einfach konzipiert sein
- und alles leisten können, was man durch einen Algorithmus (einer Lösungsstrategie) formulieren kann. (Dieser Punkt ist nicht beweisbar, aber allgemein akzeptiert)

Wir nennen einen solchen Automaten eine Turing-Maschine TM.

Warum lohnt sich ihre Untersuchung?

- 1.) Wegen ihrer Einfachheit ist die TM 'durchschaubarer' als ein realer Computer
- 2.) Weils sie alles leisten kann, was ein üblicher Computer kann, ist sie ein vollwertiger Stellvertreter auch für den komplexesten und kompliziertesten Computer (man kann jeden Prozessorbefehl als Algorithmus für eine TM formulieren)
- 3.) Was für eine TM gilt, gilt allgemein für alle Computer und Algorithmen

Das Konzept

Die TM ist ein Automat, der schrittweise arbeitet und aus folgenden Komponenten besteht:

Hardware:

- einem unendlich (= beliebig) langen Band, das mit Symbolen (etwa 0 und 1) gefüllt ist
- einem Schreib/Lesekopf, der in jedem Schritt ein Zeichen lesen und aufs Band drucken kann
- einem Leitwerk mit endlich vielen inneren Zuständen (ihre Anzahl heißt die **Ordnung** der TM)
- die TM kann drei Aktionen ausführen: einen Schritt auf dem Band nach rechts vorrücken, einen Schritt nach links gehen oder nach Ausführung der Schreib/Lese-Aktion anhalten.

und **Software:**

- dies ist eine Sammlung von Vorschriften, wann die TM welche Aktion ausführen soll, und in welchen Zustand sie dabei wechselt (die Programmierung)

Die einzelnen TM unterscheiden sich nur durch die Software (ihr Programm).

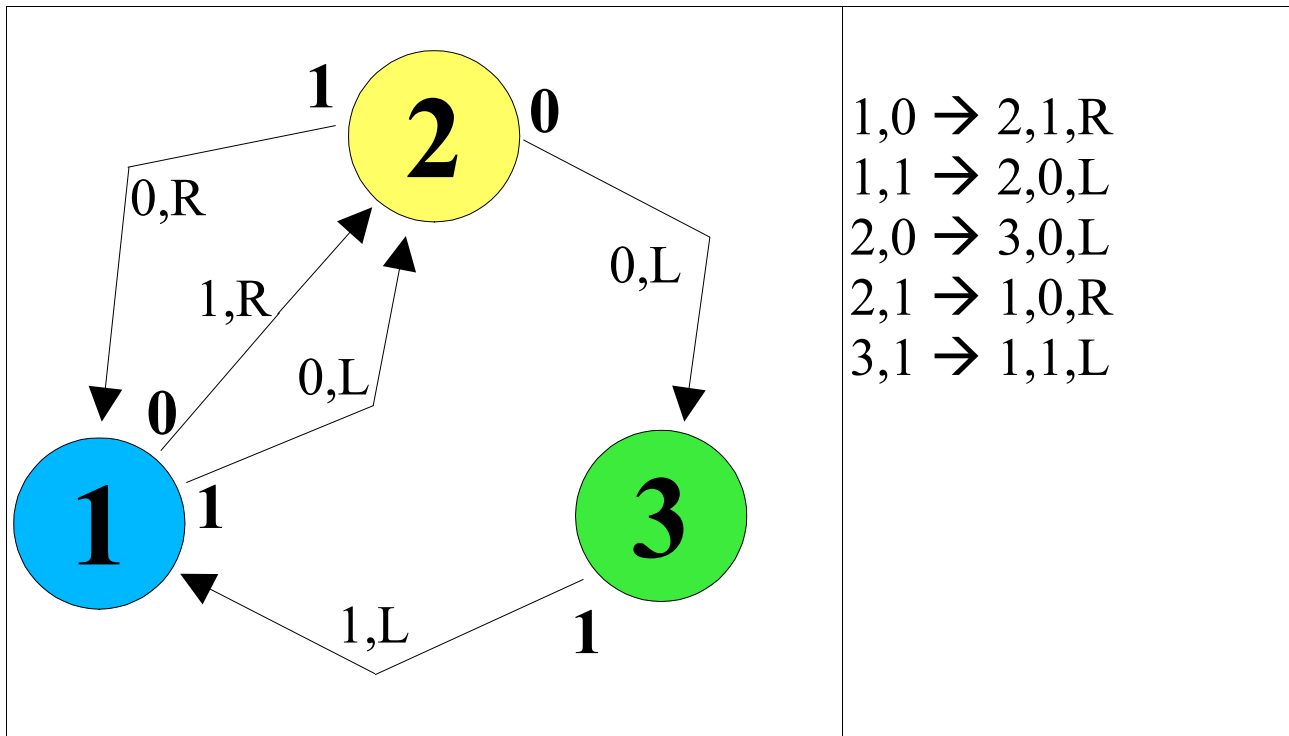
Variationen:

Gelegentlich trifft man auf Variationen dieser Vorschriften:

- das Band kann drei Symbole enthalten: 0, 1 und 'leer' (mit dem Symbol 'B' für blank). Dies ist sehr hilfreich, um etwa die Begrenzung von Zahlen zu erlauben: 'B0B' stellt dann die Zahl Null dar. (Man spart damit zahlreiche Regeln ein, die die nur Zahlenbegrenzungen aufsuchen.)
- Das Band kann die Symbole 0,1,2,3,4,5,6,7,8,9,B zum Rechnen mit Dezimalzahlen enthalten
- wenn die Maschine stoppt, druckt sie kein Zeichen mehr
- es ist kein 'stop'-Kommando nötig, wenn die TM anhält, sobald sie in einen Zustand gelangt, für den keine Verhaltensregel vorliegt
- Man trennt Zustandswechsel und Zeichenschreiben in getrennte Regeln auf.
- wir könnten weitere Aktionen zulassen ('füge hier einen neuen Platz in das Band ein' oder 'gehe 20 Stellen nach links' oder 'verharre am Platz'). Wenn jede dieser Aktionen als TM-Algorithmus formulierbar ist, darf man das auch tun. Man hätte dann eine TM mit 'Makro-Fähigkeit'.

Alle diese Versionen sind absolut gleichwertig, da jede das tun kann, was alle anderen tun können.

Die Vorschriften, wann welche Aktionen durchzuführen sind, und welche Zustände einzunehmen sind, kann man praktischerweise als **Übergangsdiagramm/Graph** oder **Tabelle** darstellen. Die zu entnehmende Information: eine TM im Zustand z liest ein Symbol (0 oder 1). In welchen neuen Zustand wechselt sie nun, was druckt sie und welche Bewegung führt sie aus. (Meist wird das Symbol S für Stop verwendet)



Angenommen, die TM befindet sich im Zustand 2.

Falls sie nun das Symbol '0' vom Band einliest, wechselt sie in den Zustand 3, schreibt eine Null aufs Band (hat also nichts geändert) und macht einen Schritt nach links.

Wäre auf dem Band eine '1' gestanden, so wäre sie in den Zustand 1 gewechselt, hätte eine Null gedruckt (das Band verändert) und wäre dann einen Schritt nach rechts gegangen.

Die Regeln im rechten Bildteil bedeuten also

Zustand,Input → *neuerZustand,Output,Bewegung*

- Wir hätten die Zustände auch mit 'blau', 'gelb', 'grün' bezeichnen können, oder mit A,B,C
- Wir hätten die Bewegungen auch durch '+' und '-' symbolisieren können
- Wir hätten die Zustände auch als 'Aktionsbereitschaft' auffassen können und ihnen erklärende Worte zuordnen können.

z.B. schrittRechts,0 → schrittlinks,1,+

Die Version mit Zahlen ist aber einfacher zu programmieren und bei Mathematikern beliebt.

(Andernfalls bräuchte man einen TM-Assembler zur Compilierung der TM-Programme, der allerdings in Python mithilfe der Dictionaries leicht zu realisieren wäre...)

Unäre Zahlen

Da die TM nur mit den Symbolen 0 und 1 arbeitet, kann sie weder mit dezimalen Zahlen (die Ziffern 2 bis 9 fehlen), noch mit binären (wo fängt die Zahl an, wo hört sie auf) direkt arbeiten. Das unäre Zahlenformat stellt etwa die Zahl 5 mithilfe von fünf Einsern dar: 11111: jede Zahl wird durch so viele Einser repräsentiert, wie ihr Wert angibt.

Braucht man die Null zum Rechnen, so wäre sie auf einem leeren Band nicht aufzufinden. In diesem Fall stellt man die Zahl n durch $n+1$ Ziffern dar.

Genauso, wie die Mathematiker aus den natürlichen Zahlen alle weiteren Bereiche ableiten, genügt es, wenn die TM diesen einen Zahlentyp kennt. Algorithmen etwa für Division von unären Zahlen sind bekannt, genauso wie die Berechnung des Sinus oder Logarithmus. (Allerdings sind diese Programme sehr umfangreich)

Beispiel: Erhöhen einer Zahl um 1

Angenommen, rechts von der Kopfposition steht auf dem Band eine unäre Zahl. Wir sollen ihren Wert um 1 erhöhen, was dem Prozessorbefehl INC eines üblichen Computer entspricht. Der Startzustand der Turingmaschine sei 0.

Wir haben also die Aufgabe, an die unäre Zahl eine Eins hinten anzufügen.

1.) Wir überlesen (d.h. ändern den Wert nicht) Nullen nach rechts, bis wir den Beginn der Zahl finden:

0,0->0,0,R

Erkennen wir die erste 1, so wechseln wir in den nächsten Zustand und lassen die 1 unverändert, indem wir sie wieder aufs Band zurückschreiben.

0,1->1,1,R

2.) Nun überlesen wir alle Einser

1,1->1,1,R

Und wechseln in den nächsten Zustand, sobald eine Null kommt. Dabei geben wir aber eine 1 aus, was dem Anfügen einer weiteren Stelle entspricht.

1,0->2,1,R

3.) Wir sind fertig und stoppen, wobei wir noch die begrenzende Null rechts von der neuen Zahl anschreiben.

2,0->2,0,S

2,1->2,0,S

Addition von unären Zahlen:

Wir nehmen an, das Band enthält zwei durch eine Null getrennte Zahlen, der Kopf der TM steht vor der ersten. Wir wollen die Summe dieser Zahlen mithilfe einer TM bestimmen.

z.B. 011101111000000

Wie gelingt das? Am einfachsten wäre es, die Null zwischen den beiden Zahlen herauszuschneiden. Das kann unsere TM allerdings nicht.

Oder so: ersetze diese Null durch einen Einser, und ersetze dafür den hintersten Einser der zweiten Zahl durch eine Null.

Algorithmus:

1.) finde nach rechts die erste 1 der ersten Zahl

2.) finde nach rechts die erste folgende Null

3.) finde nach rechts die letzte 1 (von Zahl 2)

- 4.) ersetze sie durch eine Null
- 5.) finde nach links eine Null (die zwischen den Zahlen)
- 6.) ersetze sie durch eine 1

Fleißige Biber (busy beavers)

Tibor Rado stellte 1962 eine scheinbar harmlose Frage:

Wie viele Einser kann eine anhaltende TM der Ordnung n höchstens auf ein anfangs leeres Band schreiben?

Die Folge dieser Werte wurde mit $\Sigma(n)$ benannt und sollte für eine so einfache Maschine doch wohl leicht zu finden sein.

Für $n=1$ ist die Antwort naheliegend:

1,0 \rightarrow 1,1,S

(1,1 \rightarrow 1,1,S)

erzeugt einen Einser, also $\Sigma(1)=1$.

(Beachte: die TM muss anhalten. Andernfalls wäre 1,1 \rightarrow 1,1,R ein trivialer Sieger mit unendlich vielen Einsern)

Für $n=2$ ist es auch nicht schwer:

1,0 \rightarrow 2,1,L

1,1 \rightarrow 2,1,R

2,0 \rightarrow 1,1,R

2,1 \rightarrow -,1,S

schreibt viermal den Einser, also $\Sigma(2)=4$

Für $n=3$ hatte Rado gemeinsam mit Shen Lin die TM

1,0 \rightarrow 2,1,R

1,1 \rightarrow 3,1,L

2,0 \rightarrow 1,1,L

2,1 \rightarrow 2,1,R

3,0 \rightarrow 2,1,L

3,1 \rightarrow 3,1,S

gefunden, die $\Sigma(3)=6$ liefert.

Beachte, dass man von allen anderen TM der Ordnung n zeigen muss, dass sie nicht mehr Einser schreiben! (Oder wenn sie es tun, dass sie nicht anhalten)

Für $n=4$ begann es schon schwieriger zu werden, man entwarf spezialisierte Hardware zum Testen der Kandidaten und entwickelte Ideen, wie man aussichtslose Kandidaten möglichst schnell erkennen könnte.

Es begann sogar ein Wettkampf, wer die beste – die fleißigste – TM finden könnte. Die Sieger-TM sollte den Namen '**fleißiger Biber**' erhalten (denn auch der Biber läuft ständig herum und legt Hölzchen ab).

Endlich fand Weimann 1972 eine Lösung und konnte $\Sigma(4)=13$ zeigen. (Beispiel im Unterricht)

Bemerkung: 'die' Lösung gibt es nicht – schließlich kann man durch umnummerieren der Zustände immer noch einen weiteren Biber mit den selben Qualitäten erzeugen.

Nun freute man sich schon auf die Fortsetzung des Wettkampfes – der Weg zum fleißigen Biber der Ordnung 5 schien nicht weit. (Mathematiker sprechen ganz trocken von der Berechnung von $\Sigma(5)$, meinen aber insgeheim das gleiche.) Und man erwartete Werte, die den bisherigen 1,4,6,13 ähnlich waren.

Viele Mathematiker, Informatiker und interessierte Laien begaben sich auf Spurensuche. Man schrieb vielversprechende Computerprogramme, die etwa so arbeiteten:

- fülle die Übergangstabelle zufällig (oder erzeuge alle Möglichkeiten)
- lass den Biber laufen
- entfernt er sich mehr als z.B. 1000 Schritte vom Startplatz? Dann wird er wohl nicht mehr anhalten.
- machte er bisher mehr als z.B. 1000 Schritte? Dann wird er wohl in eine Schleife geraten sein
- wenn er angehalten hat, dann zähle die hinterlassenen Einser.

Es kam aber ganz anders: Bald schon tauchten Biber auf, die es auf über 100 Einser brachten. Ein so großer Sprung in der Σ -Folge war überraschend und unerwartet.

1983 veröffentlichte Schult einen Biber 5. Ordnung mit 501 Einsern.

1984 zeigte Uhing einen Biber, der mehr als 47 Millionen Schritte ausführte und dabei 4098 Einser hinterließ. Niemand hatte eine derartige Explosion von Σ erwartet! Ist das der gesuchte fleißige Biber? Ist $\Sigma(5)$ berechnet? Das ist bis heute unbeantwortet, sollte aber vielleicht in einiger Zeit einer Lösung zuführbar sein (wenn unsere Computer sehr viel leistungsfähiger sein werden).

Ein Spezialist für Biber ist Heiner Marxen (Internetseite!). Sein aktueller Rekordhalter für Biber sechster Ordnung ist so beschreibbar:

Will man die Anzahl der von ihm erzeugten Einser aufschreiben, so hat diese Zahl 866 Stellen. Und die Zahl der dafür benötigten Schritte besitzt 1731 Stellen.

Die Funktion $\Sigma(n)$ scheint **nichtberechenbar** zu sein. Das heißt es gibt keine andere Möglichkeit zu ihrer Bestimmung, als sämtliche Möglichkeiten auszuprobieren. Doch auch das hat seine Tücken:

- es gibt sehr viele mögliche TM. Für Ordnung n sind es $(6n)^{2n}$ Stück ($2n$ Zeilen sind in der Übergangstabelle, jede kann n Zustände, 2 Symbole 0/1 und 3 Aktionen R/L/S enthalten). Berechne den Wert für $n=5$ und $n=6$!
- wie unterscheidet man, ob die TM anhält oder nicht? Marxen hat gezeigt, dass ein Biber wirklich sehr, sehr weit weg laufen kann. Und das mit 12 Zeilen TM-Programm! Hier manifestiert sich wieder einmal das 'Halteproblem'.

Für Ordnung 12 gelang inzwischen eine grobe Abschätzung für $\Sigma(12)$. Stell Dir vor, wie viele Elementarteilchen unser Weltall enthält. Und jetzt stell Dir vor, jedes davon enthielte noch einmal so viele Teilchen wie unser Weltall. Und jedes davon wieder. Und das darfst Du noch oft wiederholen. $\Sigma(12)$ ist jedenfalls **mit Sicherheit** größer als die Summe all dieser Teilchen.

Mathematisch kommt bei der Abschätzung eine Zahl vor, die aus lauter exponenzierten 4096 besteht. $4096^{(4096^{(4096^{\dots})})}$ Wie oft hier 4096 übereinander steht? 166 mal!

Berechnet wird von oben nach unten. (Frage Deinen Mathe-Lehrer, wie man 9 hoch 9 hoch 9 berechnet. Das ist nämlich nicht das mickrige (9 hoch 9) hoch 9, sondern 9 hoch (9 hoch 9). Aufschreiben und staunen!

(Quelle: A.K. Dewdney, Scientific American 1984)

Spezielle Biber

Informatiker haben Humor. Und folglich versuchen sie auch, die Realität in der Welt der Turingmaschinen nachzubilden.

Hier drei bekannte Beispiele für ganz und gar nicht fleißige Biber: sie hinterlassen **keine einzige Eins**.

- **castor ministerialis** – der Beamtenbiber: er bemüht sich möglichst weit zu kommen, ohne etwas zu leisten (ohne eine einzige Eins zu hinterlassen). Typischerweise kann ein solcher Beamter der 5. Ordnung 11 Felder weit kommen.
- **castor scientificus** – der Wissenschaftsbiber: er strampelt sich sein Leben lang nach Leibeskräften ab (möglichst viele Schritte). Ein Exemplar der Ordnung 5 macht 187 Schritte, entfernt sich dabei aber nur 8 Felder vom Ausgangspunkt. Er kommt also lange nicht so weit, wie sein Beamtenkollege.
- **castor exflippus** – der Aussteigerbiber: Er will nichts tun, sich möglichst wenig vom Platz rühren (er muss demnach am Startpunkt halten), aber dabei möglichst lange leben. Typisch für Ordnung 5: 67 Schritte

Ihre Programmierung lernst Du im Unterricht kennen.

Spezielle Aufgaben

Erweiterungen gibt es viele:

- Du bekommst einige alte Turing-Bänder geschenkt. Vor ihrer Verwendung sollten sie gelöscht werden. Entwirf eine TM, die a) eine 1 vom Band löscht und anhält, b) alle Einsen vom Band löscht (kann sie anhalten?)
- Es hat sich herausgestellt, dass sich viele Biber-Eigenschaften radikal ändern, wenn sie nicht auf einem leeren Band beginnen, sondern dieses etwa 010101010... ode 001001001001... enthält. Manche Biber erzeugen dann sogar optisch ansprechende Muster
- Raphael Robinson kam auf die Idee, Schults fleißigen Biber (501 Einsen) auf nichtleere Bänder loszulassen. Startete er ihn mit '101010101' so hielt dieser erst nach 12870233 Schritten an und hatte dabei 4911 Einsen erzeugt. Die Fachwelt war verblüfft.

Für Spezialisten: die allgemeine TM

Eine TM besteht aus

- einem endlichen Alphabet A von Symbolen zur Ein- und Ausgabe
- einem endlichen Alphabet $A' \supseteq A$ zur Berechnung von Zwischenschritten
- einem Leerzeichen B (Element von A')
- einem eindimensionalen unbegrenzten Band mit Feldern, von denen jedes genau ein Symbol aus A' enthält

- einem beweglichen Schreib-/Lesekopf, der ein Feld auf einmal auslesen oder beschreiben kann und stets ein Feld nach rechts oder links rücken muss, oder stehen bleiben kann.
- einer endlichen Menge Q von Zuständen, die die TM einnehmen kann
- zwei speziellen Zuständen q_0 und q_e (beide aus Q), die Anfangs- und Endzustand heißen
- einer Funktion δ (auch Programm genannt), die (einigen) Paaren $[q,a]$ ein Tripel $[q_1,a',R|L|S]$ zuordnet

Nachtrag:

Eine einfachere nicht berechenbare Zahlenfolge ist etwa $1, 2^2, 3^{3^3} = 3^{(3^3)}, 4^{4^4}, 5^{5^{5^5}}$

Versuche, Dir das zehnte Element dieser Folge vorzustellen. Wie lange wäre diese Zahl? Kann man diese Frage beantworten?

Beachte, dass von 'oben nach unten' exponenziert wird!

Das dritte Element ist demnach $3^{27} = 7625597484987$ und nicht $27^3 = 19683$

Schlussatz:

Niemand glaubt heute mehr an eine Lösung für $\Sigma(6)$.