

Einige große Zahlen

Ich habe in den Skizzen erwähnt, wie sich ein Wettbewerb um möglichst große natürliche Zahlen anzetteln läßt. Hier ein paar Ratschläge für einen solchen Wettkampf.

Ein naiver Ansatz wäre, Potenzen oder Fakultäten zu iterieren. Eine Fakultät ist so etwas: $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. Eine iterierte Fakultät ist so etwas: $5!! = 120!$. Eine etwas naive Möglichkeit, große Zahlen zu erzeugen, wäre, $9!!!!!!!!!!!!!!!!!!!!$ aufzuschreiben und diesem Term je nach Lust und Laune noch einige weitere Anführungszeichen hinzuzufügen.

Einen ähnlichen Effekt können wir mit iterierten Potenzen erzielen, indem wir etwa $10^{10^{10}}$ schreiben. Diese Schreibweise wird jedoch sehr schnell unübersichtlich. Es wäre besser, wenn wir Potenzen anders als mit hochgestellten verkleinerten Buchstaben schreiben könnten, denn sonst werden die Formeln schnell unlesbar. Wir wollen deshalb vereinbaren, daß wir statt a^b im folgenden $a \uparrow b$ schreiben. Einen Term wie $10 \uparrow 10 \uparrow 10 \uparrow 10$ soll von rechts nach links ausgewertet werden, also als $10 \uparrow (10 \uparrow (10 \uparrow 10))$, denn eine Auswertung von links nach rechts wäre höchst langweilig und würde in kleineren Zahlen resultieren. Generell werden wir versuchen müssen, möglichst hohe Zahlen nach rechts zu schaufeln, da sie rechts weit desaströsere Folgen haben als links. $100 \uparrow 10$ etwa ist viel kleiner als $10 \uparrow 100$, und $10 \uparrow 10 \uparrow 10 = 10 \uparrow (10 \uparrow 10)$ ist sehr viel größer als $(10 \uparrow 10) \uparrow 10$.

Eine lange Reihe $10 \uparrow 10 \uparrow \dots 10$, die besonders hohe Zahlen liefert, können wir weiter abkürzen. Donald Knuth hat für diesen Zweck eine neue Notation eingeführt: wenn wir eine Reihe $a \uparrow a \uparrow \dots a$ mit b Vorkommen von a haben, dann können wir diesen Term als $a \uparrow \uparrow b$ schreiben. $10 \uparrow 10 \uparrow 10 \uparrow 10$ ist in dieser Notation also $10 \uparrow \uparrow 4$. Diese Notation läßt sich leicht verallgemeinern:

$$a \underbrace{\uparrow \dots \uparrow}_m b = a \underbrace{\uparrow \dots \uparrow}_{m-1} \underbrace{a \uparrow \dots \uparrow}_{m-1} \dots a \quad \text{für } m > 1$$

oder, rekursiv ausgedrückt:

$$a \underbrace{\uparrow \dots \uparrow}_m b = a \underbrace{\uparrow \dots \uparrow}_{m-1} \underbrace{a \uparrow \dots \uparrow}_m b - 1 \quad \text{für } m > 1, b > 1$$

$$a \underbrace{\uparrow \dots \uparrow}_m \mathbf{I} = a$$

Die meisten Skalen und Zahlen in der Physik haben eine Reichweite von weniger als $10 \uparrow 100$ (diese Zahl hat übrigens einen eigenen Namen, „Googol“): die Zahl der Atome im Universum ist wohl um etliches kleiner. Selbst irgendwelche kombinatorischen Probleme, die die Anzahl möglicher Zustände eines Systems angeben, haben in aller Regel eine Größenordnung, die unterhalb von $10 \uparrow 10 \uparrow 100$ („Googolplex“ genannt) liegt. Mit der Knuthschen Notation lassen sich leicht Zahlen erzeugen, die alles übersteigen dürften, was Nichtmathematikern jemals untergekommen ist oder unterkommen wird. Wenn wir nicht gerade $2 \uparrow \dots \uparrow 2$ wählen (was für jede beliebige Anzahl von Pfeilen immer unveränderlich 4 ergibt), dann können wir durch einen Haufen von Pfeilen leicht wirklich, wirklich sehr, sehr große Zahlen erzeugen. $3 \uparrow \uparrow \uparrow 3$ etwa ist ein Turm von Potenzen 3^{\dots^3} mit 7.625.597.484.987 Vorkommen von 3:

$$3 \uparrow \uparrow \uparrow 3 = 3 \uparrow \uparrow 3 \uparrow \uparrow 3 = 3 \uparrow \uparrow 3^{3^3} = 3 \uparrow \uparrow 3^{27} = \underbrace{3^{\dots^3}}_{3^{27}}$$

Zahlen der Form $n \uparrow \dots \uparrow n$ mit n Pfeilen in einer Reihe heißen „Ackermann-Zahlen“, da sie ähnlich definiert und von ähnlicher Größenordnung sind wie Zahlen $A(n, n)$ der Ackermannfunktion. Die Folge der Ackermannzahlen ist nicht „primitiv rekursiv“, das heißt, sie läßt sich nicht mit einem Programm berechnen, das eine primitiv rekursive Sprache benutzt, in der jedes Programm garantiert endet.

Sehr lange Reihen von Pfeilen sind etwas umständlich zu notieren, insbesondere, wenn die Zahl der Pfeile, wie bei den Ackermannzahlen, auch noch systematisch anwachsen soll. Es liegt also nahe, die Notation zu erweitern:

$$a \rightarrow b \rightarrow c = a \underbrace{\uparrow \dots \uparrow}_c b$$

Diese Notation stammt von John Horton Conway. Wir können die Ackermannzahlen in dieser Notation definieren als $n \rightarrow n \rightarrow n$.

Es sind nun verschiedene Möglichkeiten denkbar, wie wir auch diese Notation erweitern, in der Art, wie Knuth die \uparrow -Notation erweitert hat. Wir könnten etwa versuchen, für diese Notation

„das selbe“ zu machen wie für Knuths Notation, was freilich nicht übermäßig originell wäre (abgesehen davon, daß wir uns darauf verständigen müßten, was genau denn „das selbe“ in diesem Fall wäre). Vielleicht mag die Leserin sich ja selbst einmal daran versuchen, ehe ich im nächsten Absatz die von Conway vorgeschlagene Lösung vorstelle.

Wir sagten, um möglichst große Zahlen zu erzeugen, sei es wichtig, möglichst früh weit rechts möglichst große Zahlen zu erzeugen. Conways Notation leistet auf ingeniöse Weise genau das. Zunächst einmal haben wir eine Kürzungsregel, die noch nicht übermäßig ingeniös anmutet, aber dafür sorgt, daß wir Ketten mit mehr als drei Gliedern letztendlich immer auf Ketten mit drei Gliedern reduzieren können:

$$a_1 \rightarrow a_2 \rightarrow \dots a_n \rightarrow 1 = a_1 \rightarrow a_2 \rightarrow \dots a_n$$

Die entscheidende verbleibende Frage: was geschieht bei Ketten, die mehr als drei Glieder enthalten und für die die letzte Zahl größer als 1 ist? Die Antwort hängt von der Größe der vorletzten Zahl ab.

$$a_1 \rightarrow \dots a_n \rightarrow 1 \rightarrow z = a_1 \rightarrow \dots a_n$$

$$a_1 \rightarrow \dots a_n \rightarrow 2 \rightarrow z = a_1 \rightarrow \dots a_n \rightarrow (a_1 \rightarrow \dots a_n) \rightarrow z - 1$$

$$a_1 \rightarrow \dots a_n \rightarrow 3 \rightarrow z = a_1 \rightarrow \dots a_n \rightarrow (a_1 \rightarrow \dots a_n \rightarrow (a_1 \rightarrow \dots a_n) \rightarrow z - 1) \rightarrow z - 1$$

...

für $z > 1$.

Dabei sind hier die Klammern nicht auflösbar: Klammern können erst entfernt werden, wenn die Terme in ihnen vollständig ausgewertet wurden. Wie sich leicht sehen läßt, wird durch diese Regel eine Kette mit m Termen in eine Kette überführt, die ebenfalls m Terme hat, bei der aber der Wert des letzten Terms um 1 reduziert wurde (und alle internen Terme haben ebenfalls wieder nur Länge m , mit einem letzten Term der Größe $z - 1$), so daß die Rekursion tatsächlich zu einem Abschluß kommt, aber nicht ohne zuvor für sehr, sehr große Werte zu sorgen.

In dieser Notation nun läßt sich so ziemlich alles ausdrücken, womit Mathematiker gewöhnlich zu tun haben. Die größte derzeit bekannte Zahl, die tatsächlich in einem mehr oder weniger praktischen Zusammenhang aufgetaucht ist (also nicht bei dem Versuch, möglichst große Zahlen zu erzeugen), ist Grahams Zahl, die beim Färben ziemlich spezieller Graphen in der Ramsey-

Theorie vorkommt. In Conways Notation liegt Grahams Zahl irgendwo zwischen $3 \rightarrow 3 \rightarrow 64 \rightarrow 2$ und $3 \rightarrow 3 \rightarrow 65 \rightarrow 2$.

Wir können versuchen, auch diese Notation wieder zu erweitern. Eine naheliegende Möglichkeit wäre, Ketten der Form $a \rightarrow a \rightarrow \dots a$, mit b Vorkommnissen von a , in einer neuen Notation zusammen zu fassen. Wir können ein solches Gebilde etwa mit $a \overset{2}{\uparrow} b$ bezeichnen. Dann ist etwa

$3 \rightarrow 3 \rightarrow 3 \rightarrow 3 = 3 \overset{2}{\uparrow} 4$ eine Zahl, die sehr viel größer als Grahams Zahl ist. Wir können dann auch für das Symbol $\overset{2}{\uparrow}$ die selben Regeln einführen wie für das Symbol \uparrow . Insbesondere können wir dieses Symbol Conwayisieren, indem wir ein neues Symbol $\overset{2}{\rightarrow}$ einführen, mit

$a \underbrace{\overset{2}{\uparrow} \dots \overset{2}{\uparrow}}_c b = a \overset{2}{\rightarrow} b \overset{2}{\rightarrow} c$, für das dann auch wieder die Rechenregeln des Symbols \rightarrow gelten. Als

nächstes können wir eine Kette $\underbrace{a \overset{2}{\rightarrow} a \overset{2}{\rightarrow} \dots a}_b$ mit Hilfe eines neuen Symbols zu $a \overset{3}{\uparrow} b$ zusammen

fassen, und so weiter. $3 \overset{3}{\uparrow} 3$ ist dann eine Zahl jenseits aller bisher bekannter Nützlichkeit.

Der so eingeschlagene Weg ist freilich nicht übermäßig originell. Im Vergleich zu den Ansätzen Knuths oder Conways haben wir keine bahnbrechend neuen Ideen entwickelt. Wir wollen deshalb statt dessen noch auf einen anderen Aspekt verweisen.

Die ursprüngliche Funktion $a \uparrow b$ wurde definiert als a^b . Wir hätten jedoch auch, etwas systematischer, mit $a \uparrow b = a + b$ beginnen können, und alle übrigen Definitionen unverändert lassen. In diesem Fall hätten wir zwei Schritte später die Potenz erreicht, mit $a \uparrow \uparrow b = a^b$. Wir hätten sogar, noch bescheidener, $a \uparrow b = a + 1$ definieren können. In diesem Fall hätte dann $a \uparrow \uparrow b = a + b - 1$ gegolten, aber auch für diesen Fall hätten wir wieder unsere Hierarchie entwickeln können. Wir können als Ausgangspunkt irgend eine beliebige zweistellige Funktion

$$f: \mathbb{N} \setminus \{0\} \times \mathbb{N} \setminus \{0\} \rightarrow \mathbb{N} \setminus \{0\}$$

verwenden (aus systematischen Gründen sind allerdings solche Funktionen f vorzuziehen, für die $f(a, 1) \equiv a$ gilt.). Wir definieren dann

$$a \overset{f}{\uparrow} b = f(a, b)$$

Ansonsten können wir den Aufbau wie oben durchführen, also

$$a \underbrace{\uparrow \dots \uparrow}_m b = a \underbrace{\uparrow \dots \uparrow}_{m-1} \underbrace{\uparrow \dots \uparrow}_{m-1} \dots a$$

Analog lassen sich auch Operatoren \xrightarrow{f} , $\overset{n}{\uparrow}_f$ und $\overset{n}{\rightarrow}_f$ definieren.

Wir können dabei sogar auf die Definition des einfachen Pfeils verzichten und uns rein auf die Definition des liegenden Pfeils beschränken. Zunächst einmal führen wir dazu folgende Konvention ein:

$$a \overset{1}{\xrightarrow{f}} b = a \overset{1}{\xrightarrow{f}} b \overset{1}{\xrightarrow{f}} 1 = a \overset{1}{\uparrow}_f b = f(a, b)$$

Die grundlegende Regel für den Pfeil

$$a \underbrace{\uparrow \dots \uparrow}_c b = a \underbrace{\uparrow \dots \uparrow}_{c-1} \underbrace{\uparrow \dots \uparrow}_{c-1} \dots a$$

können wir auch rekursiv formulieren mit

$$a \underbrace{\uparrow \dots \uparrow}_c b = a \underbrace{\uparrow \dots \uparrow}_{c-1} \underbrace{\uparrow \dots \uparrow}_c b - 1 \quad \text{für } b > 1, c > 1$$

und

$$a \underbrace{\uparrow \dots \uparrow}_c 1 = a$$

Diese Regel läßt sich nun auch anders formulieren:

$$\begin{aligned}
& a \rightarrow b \rightarrow c \\
& = \underbrace{a \uparrow \dots \uparrow}_c b \\
& = \underbrace{a \uparrow \dots \uparrow}_{c-1} \underbrace{a \uparrow \dots \uparrow}_c b - I \quad \text{für } b > 1, c > 1 \\
& = a \rightarrow (\underbrace{a \uparrow \dots \uparrow}_c b - I) \rightarrow c - I \\
& = a \rightarrow (a \rightarrow b - I \rightarrow c) \rightarrow c - I
\end{aligned}$$

Auch den Schritt von $\xrightarrow{n-1}$ nach \uparrow^n können wir neu formulieren:

$$a \xrightarrow{n} b = a \xrightarrow{n} b \xrightarrow{n} I = a \uparrow^n b = \underbrace{a \xrightarrow{n-1} a \xrightarrow{n-1}}_b$$

Im folgenden geben wir nun einen systematischen Überblick über die verwendeten Regeln:

$$(A_0) \quad a \xrightarrow[f]{I} b = f(a, b)$$

$$(A_1) \quad a \xrightarrow[f]{n} b = \underbrace{a \xrightarrow[f]{n-1} a \xrightarrow[f]{n-1}}_b \quad \text{für } n > 1$$

$$(A_2) \quad a_1 \xrightarrow[f]{n} \dots a_{m-1} \xrightarrow[f]{n} I \xrightarrow[f]{n} a_{m+1} \xrightarrow[f]{n} \dots a_k = a_1 \xrightarrow[f]{n} \dots a_{m-1} \quad \text{für } m > 1$$

$$(A_3) \quad a_1 \xrightarrow[f]{n} \dots a_{m-1} \xrightarrow[f]{n} a_m \xrightarrow[f]{n} a_{m+1} = \begin{cases} a_1 \xrightarrow[f]{n} (a_1 \xrightarrow[f]{n} a_m - I \xrightarrow[f]{n} a_{m+1}) \xrightarrow[f]{n} a_{m+1} - I & \text{für } m = 2 \\ a_1 \xrightarrow[f]{n} \dots a_{m-1} \xrightarrow[f]{n} (a_1 \xrightarrow[f]{n} \dots a_{m-1} \xrightarrow[f]{n} a_m - I \xrightarrow[f]{n} a_{m+1} - I) \xrightarrow[f]{n} a_{m+1} - I & \text{für } m > 2 \end{cases}$$

für $a_m > 1, a_{m+1} > 1$

Bemerkung: die Regel A2 erlaubt auch das Kürzen einer 1 an letzter Stelle (falls $n = m$). Die Regel A4 beschreibt für den Fall $m = 2$ die Regeln des Knuthschen Pfeils, für den Fall $m > 2$ die Regeln des Conwayschen Pfeils.

Bevor wir in unserer Diskussion fortfahren, wollen wir noch kurz einige Regeln für die Iteration einstelliger Funktionen aufstellen. Sei f eine einstellige Funktion $f: \mathbb{N} \setminus \{0\} \rightarrow \mathbb{N} \setminus \{0\}$. Dann bezeichnet

$$f^1(x) = f(x)$$

$$f^{n+1}(x) = f(f^n(x))$$

Nun können wir in den Exponenten auch wieder Funktionsergebnisse von f schreiben, etwa $f^{f(x)}(x)$. Diesen Vorgang können wir wiederholen und so Ausdrücke der Form $f^{f^{f(x)}(x)}(x)$ erhalten. Einen solchen Ausdruck, in dem n -fach das Symbol f auftaucht, wollen wir mit $f^{+n}(x)$ abkürzen. Es gilt also

$$f^{+1}(x) = f(x)$$

$$f^{+(n+1)}(x) = f^{f^{+n}(x)}(x)$$

Nun können wir auch in solche Ausdrücke wieder einen weiteren Term $f(x)$ hineinstopfen, etwa $f^{+f(x)}(x)$. Auch hier lassen sich wieder Türme bilden der Form $f^{+f^{f(x)}(x)}(x)$. Solche Gebilde bezeichnen wir mit $f^{++n}(x)$, wobei n die Zahl der Symbole f rechts des Symbols $+$ im Term $f^{+f^{f(x)}(x)}(x)$ angibt. Allgemein schreiben wir

$$f^{\overbrace{+ \dots +}^n}(x) = f^{\overbrace{+ \dots +}^{f^{+n}(x)}}(x)$$

Nach diesem Zwischenspiel wenden wir uns wieder der Definition großer Zahlen zu. Im Prinzip können wir jede Definition erweitern, indem wir etwa zu jeder Funktion $f(x)$ eine „größere“ Funktion $f^*(x) = f(x) + 1$ definieren (oder $= 2 \cdot f(x)$, oder $= 2^{f(x)}$, oder $= f(f(x))$, oder was immer uns in den Sinn kommt). Dabei wird es aber immer schwieriger, etwas wirklich neues und originelles zu finden, das nicht einfach die bestehenden Definitionen kopiert, sondern einen echten Schritt über das bestehende System hinaus tut. Die Reihe 1, 2, 3, ... können wir überbieten, indem wir die Reihe verdoppeln: 2, 4, 6, ... Ist dieser Trick erst einmal entdeckt, läßt er sich durch verdreifachen, also 3, 6, 9, ... überbieten, und so weiter, aber diese Überbietung zeigt wenig tiefere Einsicht, sie stellt keine neue Abstraktion dar. Etwas neues wäre es, wenn wir zu Zweierpotenzen 2, 4, 8, ... übergehen. Die lassen sich zwar wieder durch Dreierpotenzen, Viererpotenzen &c. überbieten, diese bringen aber nichts neues. Als nächstes könnten wir Potenztürme der Höhe 2

einführen, also $1^1, 2^2, 3^3, \dots$ Diese können wir wieder durch Potenztürme der Höhe 3 überbieten, und so weiter. Es ist klar, daß sich hier eine neue Reihe abzeichnet: Multiplikation, Potenzen, Potenztürme. Sei $f(a, b) = a + b$. In diesem Fall können wir eine neue Reihe definieren, die wie $1 \underset{f}{\uparrow} 1, 2 \underset{f}{\uparrow} \underset{f}{\uparrow} 2, 3 \underset{f}{\uparrow} \underset{f}{\uparrow} \underset{f}{\uparrow} 3, \dots$ wächst und die das bisherige Schema durchbricht und transzendiert. Diese Reihe nun wächst bereits mit der Geschwindigkeit der Ackermannzahlen. Es ist möglich, auch diese Reihe wieder zu transzendieren und etwa zur Conway-Notation überzugehen. Jedesmal, wenn wir das Bildungsprinzip einer Reihe durchschauen, können wir die eingefahrene Spur verlassen und neue Funktionen bilden, die Abstraktionen der bisherigen Funktionen sind. Aber diese Abstraktionen werden zugleich auch immer schwieriger zu bilden. Wie können wir die Conway-Notation in gleicher Weise erweitern, wie die Conway-Notation die Knuth-Notation erweitert? Und wenn wir die Reihe Knuth, Conway, ... tatsächlich beliebig fortsetzen können, wie können wir dann auch diese Reihe wieder transzendieren und etwas finden, das diese Reihe an Abstraktion übertrifft?

Wir könnten versuchen, uns dem Problem von einer anderen Seite zu nähern: alle Zahlen, die wir bis hierher definiert haben, wurden durch die Zusammenstellung von Worten definiert, und je größer die Zahlen und je komplizierter die Definitionen wurden, desto mehr Worte brauchten wir. Warum nicht also folgende Reihe eröffnen: die größte natürliche Zahl, die sich mit höchstens zehn Worten definieren läßt, die größte natürliche Zahl, die sich mit höchstens zwanzig Worten definieren läßt, und so weiter. Aber eine solche Definition ist zirkulär: ich könnte ja auch sagen: „der Nachfolger der größten natürlichen Zahl, die sich mit höchstens vierzehn Worten definieren läßt.“ Diese Zahl ist mit bloß vierzehn Worten definiert, muß also kleinergleich der größten mit vierzehn Worten definierbaren Zahl sein, ist aber per Definition größer als jene Zahl.

Um solche Selbstbezüglichkeiten zu vermeiden, brauchen wir ein hierarchisches Modell: eine formale Sprache, in der wir Zahlen definieren, und über die wir dann in einer zweiten Sprache (nämlich Deutsch) sprechen können. Eine mögliche formale Sprache wäre etwa, beispielsweise, die Sprache der Javaprogramme: wir können in Java Programme schreiben, die natürliche Zahlen ausgeben (etwa im Format `java.math.BigInteger`), und diese Programme haben eine bestimmte Länge, und zu jeder vorgegebenen Programmlänge gibt es nur endlich viele verschiedenen Programme. Wir können also $j(n)$ definieren als die größte natürliche Zahl, die sich mit einem Java-Programm mit einem Quellcode mit einer Länge von n Zeichen ausgeben läßt.

In der Berechenbarkeits- und Komplexitätstheorie ist es im allgemeinen üblich, sich zur Erforschung grundsätzlicher Fragen nicht auf so komplizierte Sprachen wie Java zu beziehen. Statt

dessen haben wir hier als Standardmodell die Turingmaschinen. Eine Turingmaschine besteht aus einem Band von Feldern, das sich rechts und links ins unendliche erstreckt, jedes Feld des Bandes kann einen von zwei Zuständen annehmen, darüber hinaus hat die Turingmaschine einen Schreiblesekopf, der auf genau eines dieser Felder zugreift. Und schließlich besteht die Turingmaschine noch aus einer Sammlung von Zuständen. Ein solcher Zustand a kann etwa so aussehen: „Wenn auf dem vorliegenden Feld eine 0 steht, dann schreibe auf das Feld eine 1, gehe ein Feld nach links, und wechsele in den Zustand b ; steht dagegen auf dem vorliegenden Feld eine 1, so laß diese 1 unverändert, gehe ein Feld nach rechts und wechsele in den Zustand c “. Generell besteht ein Zustand aus folgenden Daten: für jeden der beiden möglichen Werte des aktuellen Feldes ein Wert, der auf dieses Feld geschrieben werden soll, eine Richtung, in die der Lesekopf bewegt werden soll, und einen Zustand, in den die Maschine wechseln soll. Lediglich ein Zustand unterscheidet sich von den übrigen Zuständen, insofern er der Maschine befiehlt, in keinen weiteren Zustand mehr zu wechseln, sondern das Programm zu beenden. Diesen Zustand 0 wollen wir jedoch (da er ansonsten keine Daten enthält), bei der Zahl der Zustände einer Maschine unberücksichtigt lassen. Eine Maschine beginnt stets im Zustand 1. Eine simple Maschine mit drei Zuständen könnte etwa so aussehen:

Zustand 1:

Wenn 0, dann schreibe 1, gehe nach rechts, Zustand 2

Wenn 1, dann schreibe 1, gehe nach rechts, Zustand 2

Zustand 2:

Wenn 0, dann schreibe 1, gehe nach rechts, Zustand 3

Wenn 1, dann schreibe 1, gehe nach rechts, Zustand 3

Zustand 3:

Wenn 0, dann schreibe 1, gehe nach rechts, Zustand 0

Wenn 1, dann schreibe 1, gehe nach rechts, Zustand 0

Diese Maschine schreibt, nach rechts wandernd, drei Einsen auf das Band und bleibt dann stehen. Obwohl diese Maschine nicht eben eine komplizierte Berechnung durchführt (sie gibt lediglich stets die Zahl drei aus), gibt es doch auch andere Turingmaschinen, die weit kompliziertere Berechnungen ausführen. Ja, es ist sogar so, daß es höchst glaubwürdig ist, daß alles, was sich überhaupt berechnen läßt, sich mit einer Turingmaschine berechnen läßt.

Ein etwas raffinierteres Beispiel einer Turingmaschine mit nur zwei Zuständen sieht etwa so aus:

Zustand 1:

Wenn 0, dann schreibe 1, gehe nach rechts, Zustand 2.

Wenn 1, dann schreibe 1, gehe nach links, Zustand 2.

Zustand 2:

Wenn 0, dann schreibe 1, gehe nach links, Zustand 1.

Wenn 1, dann schreibe 1, gehe nach links, Zustand 0.

Gegeben sei nun eine Turingmaschine mit n Zuständen und, als Ausgangspunkt, einem leeren (das heißt, mit lauter Nullen gefülltem) Band. Eine solche Maschine kann nun eventuell niemals anhalten, wie das etwa bei der folgenden Maschine mit zwei Zuständen der Fall ist:

Zustand 1:

Wenn 0, dann schreibe 1, gehe nach rechts, Zustand 2

Wenn 1, dann schreibe 0, gehe nach links, Zustand 2

Zustand 2:

Wenn 0, dann schreibe 1, gehe nach links, Zustand 1

Wenn 1, dann schreibe 0, gehe nach rechts, Zustand 1

Diese Maschine rennt auf dem Band hin und her, ohne jemals zur Ruhe zu kommen, wobei sie auf einem immer größeren Bereich ein alternierendes Muster von Nullen und Einsen schreibt, die bei jedem Durchgang wechseln. Der Zustand 0 wird nie erreicht.

Das Problem, wann denn nun eine Turingmaschine anhält und wann sie ewig weiterläuft, ist schwierig und, schlimmer noch, es gibt auch keinerlei allgemeingültiges Verfahren oder Schema, keinen Algorithmus, der es erlauben würde, für eine beliebige Turingmaschine zu entscheiden, ob sie hält oder nicht. Für das hier vorgestellte Exemplar ist klar, daß es niemals anhält (der Beweis ist trivial, da ja weder Zustand 1 noch Zustand 2 jemals auf Zustand 0 verweisen), aber da Turingmaschinen beliebig kompliziert sein können, kann auch der Beweis, daß sie niemals halten, beliebig kompliziert sein, und es gibt kein allgemeines Schema, das einen solchen Beweis liefern könnte.

Unter den Turingmaschinen mit n Zuständen allerdings, die tatsächlich einmal anhalten, gibt es solche, die bis dahin viele oder wenige Einsen auf das Band geschrieben hat. Eine Maschine nun, die die maximal mögliche Anzahl von Einsen auf das Band geschrieben hat, heißt fleißiger Biber (busy beaver). Sei $B(n)$ die Zahl der Einsen, die ein fleißiger Biber mit n Zuständen erzeugt.

Alle Zahlen, die wir weiter oben definiert haben, sind durch irgendwelche Algorithmen definiert, die wir auf irgendwelche Startzahlen angewandt haben. Für alle diese Zahlen gibt es Turingmaschinen, deren Programm diesen Algorithmus nachahmt und die deshalb diese Zahl als Anzahl von Einsen auf ihr Band schreibt. Eine Zahl wie, sagen wir, $3 \uparrow^3 3$, die an sich recht groß erscheint, ist doch kleiner als $B(n)$ für irgend ein relativ kleines n , und auch wenn wir nicht wissen, für welches n die Funktion $B(n)$ die Zahl $3 \uparrow^3 3$ übersteigt, ist doch anzunehmen, daß n deutlich kleiner als, sagen wir, 100 ist. Wir können uns nun zwar allerlei (rekursiv definierte) Funktionen $f(n)$ ausdenken, beispielsweise $f(n) = n \uparrow^n n$, von denen wir uns vielleicht einbilden mögen, sie seien rasch wachsend. Aber ab einem bestimmten n_0 wächst für alle $n > n_0$ $B(n)$ schneller als alle unsere $f(n)$, die wir uns so mühsam ausgedacht haben.

Da es keine rekursiv definierbare Funktion gibt, die es erlauben würde, $B(n)$ zu berechnen, ist das Ermitteln einzelner Werte dieser Funktion ausgesprochen mühsam. Unser Wissen beschränkt sich daher auf die ersten vier Funktionswerte. Diese sind 1, 4, 6 und 13. Für die nächsten beiden Werte liegen lediglich Abschätzungen nach unten vor: diese beiden Werte sind jedenfalls größer als 4098 beziehungsweise 12 914 951 964 730 997 250 673 433 546 819 849 509 549 358 087 128 690 053 958 730 050 912 043 114 050 444 850 240 131 432 687 888 779 698 205 017 959 267 279 467 247 759 159 594 822 175 225 305 432 481 859 864 495 796 137 909 683 447 198 447 312 843 568 880 129 905 330 630 692 235 127 777 655 264 853 382 670 979 398 926 663 934 043 364 554 169 509 365 540 834 461 843 577 841 574 296 433 860 268 929 575 034 928 793 440 722 283 883 496 539 655 948 945 100 141 899 429 447 468 817 367 569 604 813 038 150 912 656 805 487 172 475 593 041 843 712 279 467 853 624 749 989 147 054 303 748 499 093 249 845 855 395 105 658 447 844 504 456 040 960 051 641 314 951 220 524 824 061 775 814 634 738 272 664 481 030 066 727 094 186 265 135 749 803 147 803 742 486 664 009 592 180 119 421 672 821 913 958 840 123 231 130 890 028 804 306 931 645 773 916 087 727 281 493 526 404 733 170 198 979 765 603 424 776 606 833 684 409 529 730 007 044 118 561 624 874 873 652 997 002 032 667 344 587 137 859 002 909 978 872 928 814 647 043 325 481 327 200 728 882 173 015 355 211 743 620 254 438 041 767 965 892 742 035 979 306 286 763 642 267 313 721 146 548 318 330 807 873 ($\approx 1,29 \cdot 10^{865}$).

Wenn wir die Funktion $B(n)$ lediglich dazu mißbrauchen wollen, unsere Freunde mit großen Zahlen zu beeindrucken, ist es ein wenig mißlich, daß diese Funktion nicht sofort in Schwung kommt und das erste halbe Dutzend Werte, wiewohl extrem mühsam zu berechnen, doch nicht so groß ist wie andere von uns bereits definierte Monster (so ist etwa 10^{865} winzig im Vergleich zur dritten Ackermannzahl). Aber dem läßt sich leicht abhelfen, indem wir nur genügend große Startwerte verwenden. Um das zu erreichen, definieren wir kurzerhand (und mit roher Gewalt) eine neue Funktion:

$$b(n, m) = B_{\underbrace{m}^{+\dots+m}}(m) + n - 1$$

Wir erinnern uns, daß im Hexadezimalsystem das Zeichen b für die Zahl 11 steht. Nun definieren wir eine neue Zahl, das Busy-Beaver-Beta:

$$\beta = b \xrightarrow[b]{b} b$$

Diese Zahl dürfen wir wohl als groß bezeichnen, ohne befürchten zu müssen, der Übertreibung gescholten zu werden.

Jan Thor, 1.4.2003